(1) Basic steps for Design

1) Feed input. Data flows from layer to layer and Retrieve the output

$$y = network(x, w)$$

2) Calculate error (E)

eg:
$$E = \frac{1}{2}(y^* - y)^2$$

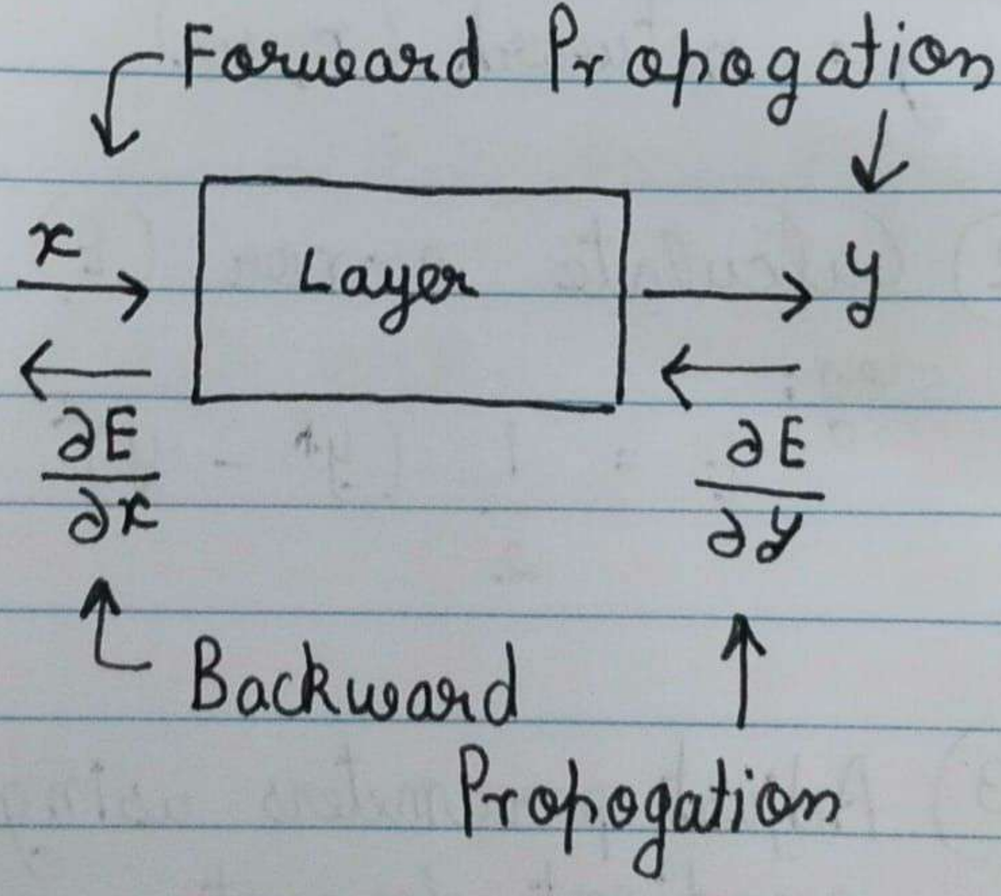
3) Adjust parameters using gradient descent

$$w \leftarrow w - \alpha \frac{\partial E}{\partial w}$$

4) Repeat

Just a big function

- Each layer needs to be implemented in a seprate class



For

- Every trainable layer there are a set of parameters 'w' which changes based on gradient and descent.

and in order to change the layer needs to find the partial derivative wrt the parameter

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial y}\underbrace{\frac{\partial y}{\partial W}}_{\text{specific to computation of layer}} \quad \left(\begin{array}{c}\text{using}\\ \text{chain}\\ \text{rule}\end{array}\right)$$

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial X} \rightarrow \text{specific to computation of layer}$$

$\frac{\partial E}{\partial X}$ ↓ for input

↳ we need it for input as most neural networks are Sequential meaning output of one is input of another.

(2) Create Base layer

2 attribute → I/P → O/P

2 method
- Forward
- backward

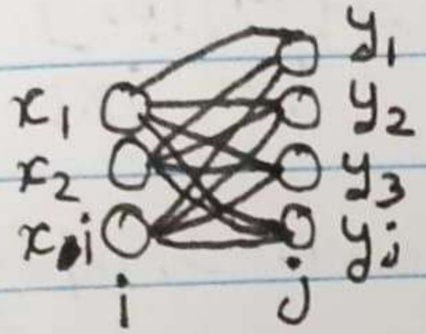
learning rate

additional parameter → can be used to pass optimizer

responsible for 2 thing
- updation of parameters
- return error derivative wrt input

(3) Dense Layer (Forward step)

- Connects set of i I/P neurons to j o/P neurons



- each input is connected to every output
- each connection is called weight → w_{ji}

general output

- every output value is computed as Sum of all input value times the weight connecting them plus bias (b) → trainable parameter

$$y_j = \sum x_i w_{ji} + b_j$$

can be used to write in form of matrix multiplication

$$Y = W \cdot X + B$$

$$\underset{j \times 1}{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \end{bmatrix}} = \underset{j \times i}{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1i} \\ w_{21} & w_{22} & \cdots & w_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j1} & w_{j2} & \cdots & w_{ji} \end{bmatrix}} \underset{i \times 1}{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix}} + \underset{j \times 1}{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_j \end{bmatrix}}$$

Hence, we can simply write it as

$$Y = W \cdot X + B$$

- This layer inherits its from Dense layer
  - takes 2 parameter
    - output and input neurons
  - weight and bias are randomly initialized

(4) Dense Layer (Backward step)

$$\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \frac{\partial E}{\partial y_2} \\ \vdots \\ \frac{\partial E}{\partial y_j} \end{bmatrix}_{j \times 1}$$

and we need to calculate for $\frac{\partial E}{\partial W}$ which is in same shape as that of W of previous page but instead of w its $\frac{\partial E}{\partial w_{ji}}$

:3 ↓ Lazy to draw

eg: $\frac{\partial E}{\partial w_{12}}$

$$= \frac{\partial E}{\partial y_1}\frac{\partial y_1}{\partial w_{12}} + \frac{\partial E}{\partial y_2}\frac{\partial y_2}{\partial w_{12}} + \cdots + \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{12}}$$

to first calculate we need to find where does w_{12} appear on forward propogation equations which comes to be y_1 and since it only comes in y_1 other derivatives are 0.

$$\frac{\partial E}{\partial w_{12}} = \frac{\partial E}{\partial y_1} \frac{\partial y_1}{\underbrace{w_{12}}_{\substack{x_2 \\ \text{Coefficient of } w_{12}}}}$$

we can generalise this

$$\boxed{\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial y_j} x_i \\ &\Rightarrow \frac{\partial E}{\partial Y} X^t \end{aligned}}$$

ok so i didnt really understand the process after this but

↓

ok so PTO

Since x was a column vector to multiply we transposed it

Now

$$\boxed{\frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y}}$$

(Not showing steps)

and

$$\boxed{\frac{\partial E}{\partial X} = W^t \cdot \frac{\partial E}{\partial Y}}$$

⑥ Activation Layer { Backward Prop

$$\frac{\partial E}{\partial Y} \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \vdots \\ \frac{\partial E}{\partial y_i} \end{bmatrix} \quad \frac{\partial E}{\partial X} = \begin{bmatrix} \frac{\partial E}{\partial x_1} \\ \vdots \\ \frac{\partial E}{\partial x_i} \end{bmatrix}$$

$\frac{\partial E}{\partial Y} \downarrow$ given

$\frac{\partial E}{\partial X} \downarrow$ have to find

Now for

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial y_1} f'(x_1)$$

and since this can be generalised we write

$$\frac{\partial E}{\partial y_i} f'(x_i)$$

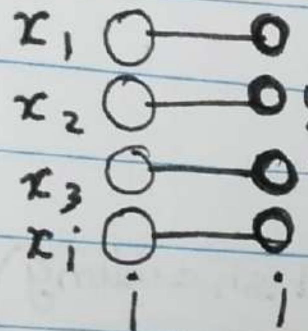
and in vector form

$$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y} \odot f'(X)$$

$\odot$ → hadamard product

(5) Activation Layer

(has no trainable parameters)



$y_1 = f(x_1)$

$y_2 = f(x_2)$

$y_3 = f(x_3)$

$y_i = f(x_i)$

(Forward Prop)

$\Leftrightarrow Y = f(X)$

takes I/P neurons and passes them through an activation function

also inherits from base layer

2 parameters

- Activation layer
- input

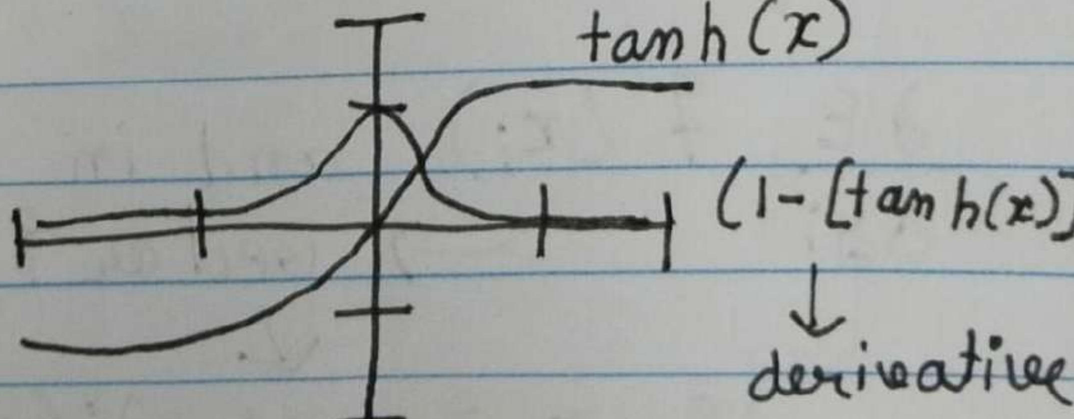
hadamard product ⟹ Multiplication of 2 vectors elements wise

---

7 Specific activation

need function and its derivative

eg: Hyper bolic tangent



- Its non linear
- we inherit from activation layer

7.1 Mean Squared Error

$$E = \frac{1}{n} \sum_i (y_i^* - y_i)^2$$

E — Basically the fuckin error

y_i^* — desired output

y_i — actual output

So since its Sequential,

$$\frac{\partial E}{\partial Y} = \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \vdots \\ \frac{\partial E}{\partial y_i} \end{bmatrix}$$

Now taking y_1 as example

$$\frac{\partial E}{\partial y_1} = \frac{\partial}{\partial y_1} \frac{1}{n} (y_1^* - y_1)^2$$

$$= \frac{2}{n} (y_1 - y_1^*)$$

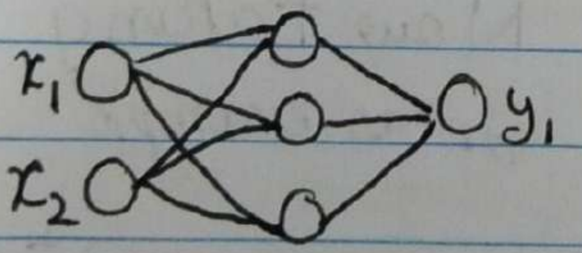
like all cases we can generalise and write in matrix format

$$\frac{\partial E}{\partial Y} = \frac{2}{n}(Y - Y^*)$$

---

8. Solving XOR

diagram



Exclusive OR

x_1	x_2	y_1
0	0	0
0	1	1
1	0	1
1	1	0

IK XOR already is inbuilt but doing this since it is not linearly seprable