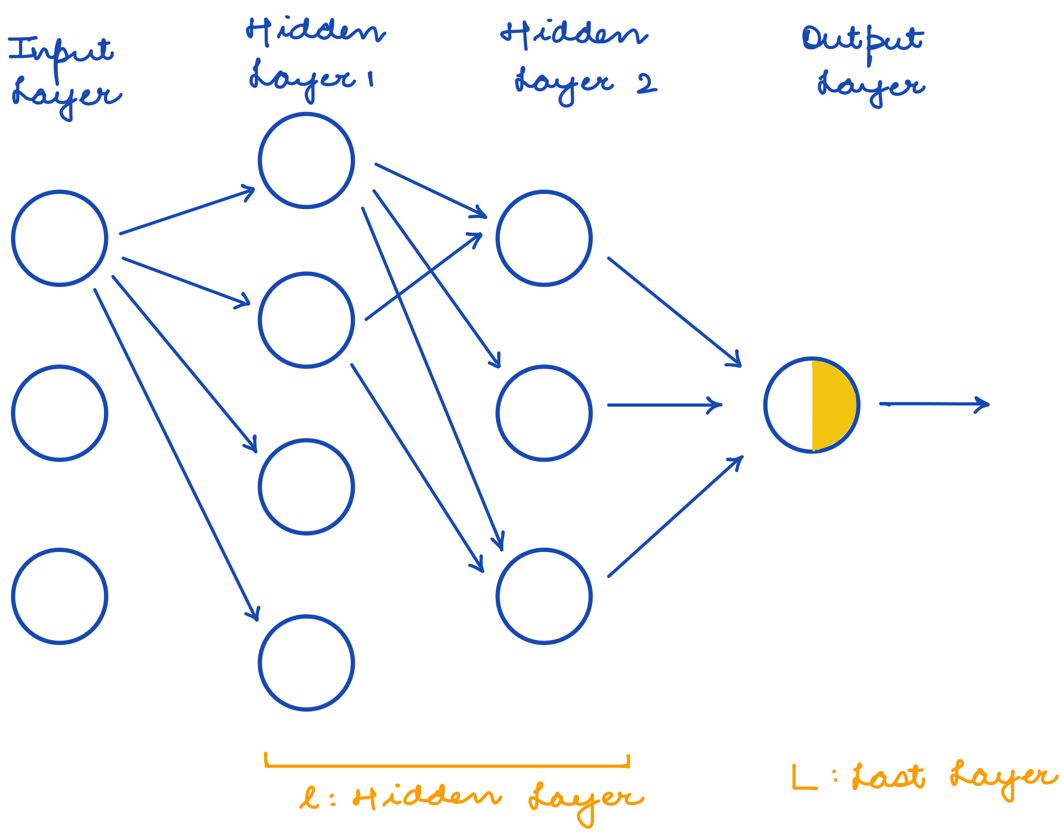
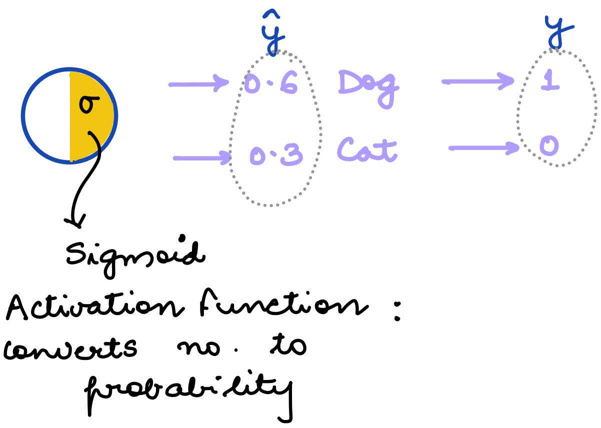
Backpropagation is considered as a challenging problem in Machine Learning, reason being it involves knowledge of Linear Algebra and Calculus. Given the formula it is not very difficult to implement but the important thing is how you reached at the formula.

If you have a neural network then you can do a forward pass, get the prediction but prediction are going to be random because you are doing random initialization of weights. We want that our network should be able to learn and should give good results for regression and classification. Libraries like Tensor flow and PyTorch will automatically do Backpropagation for you.



Backpropagation algo is needed because : lets say your algo said that I am 60% sure that it is a dog but actual output you wanted is 1. Your algo said probability of cat is 0.3, you will take sigmoid and it will become 0.



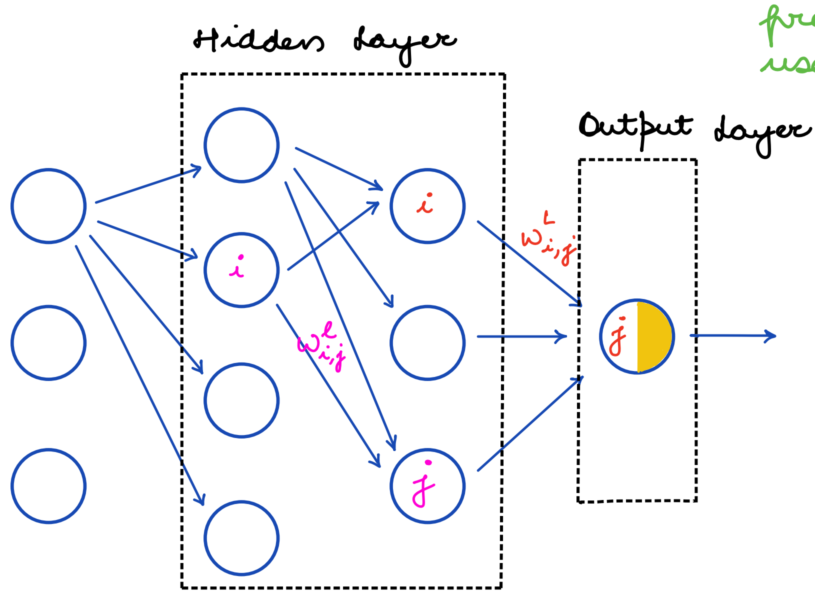

1 means Dog
0 means Cat

we want to figure out the loss. Loss is $\hat{y} - y$.

Loss function we generally use in classification is Cross Entropy. For simplification, we are using mean squared error loss.

$$MSE = L(\hat{y}, y) = \frac{1}{2}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

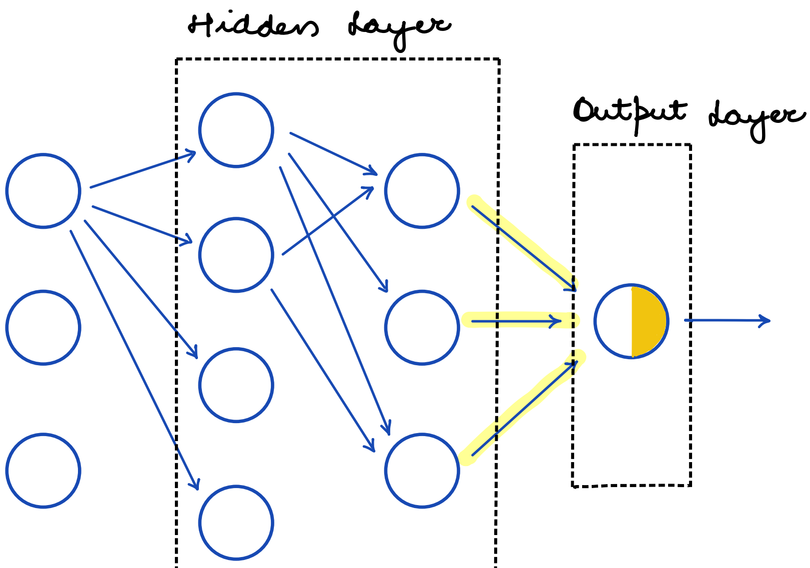
In general, this type of loss is non convex for a given classification problem, so we generally use cross entropy.



I want to analyse how loss function changes on changing the weights, if I make a change in $w^L_{i,j}$ then how does my error, loss and prediction change.

There are 2 type of neurons : one which are in output layer and other neurons are in hidden layer.

Output neurons are directly connected with loss, if you want to change your loss by $\delta L$ then you can find out what change I should make to these weights:




You need to compute 2 things :

**Case 1 : Output Layer**

$$\frac{\partial L}{\partial w^L_{i,j}}$$

$\downarrow$

$\partial L$ wrt $w$ in last layer

$$\frac{\partial L}{\partial b^L}$$

$\downarrow$

$\partial L$ wrt bias in last layer

**Case 2 : Hidden Layer**

$$\frac{\partial L}{\partial w^l_{i,j}}$$

$\downarrow$

$\partial L$ wrt $w$ in hidden layer

$$\frac{\partial L}{\partial b^l}$$

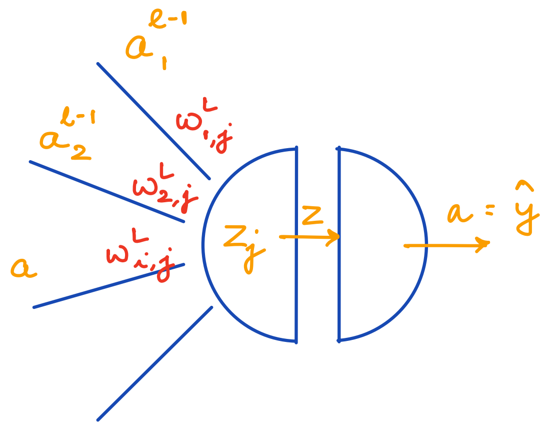
$\downarrow$

$\partial L$ wrt bias in hidden layer

We need to find out how loss changes when weight and bias changes.

## Case 1:

### output layer




$L(y,a)$

$$z_j = \sum_i^{l} w_{i,j}^L \, a_i^{l-1} + b_j^L$$

i iterates over all the neurons in last layer l.

We want to find out how $\hat{y}$ changes with $w_{i,j}$ $\rightarrow$ $\frac{\partial L}{\partial w_{i,j}}$

$$w_{i,j} \rightarrow z_j \rightarrow a_j \rightarrow L(a,y)$$

$$\frac{\partial L}{\partial w_{i,j}} = \underset{1}{\frac{\partial L}{\partial a_j}} \cdot \underset{2}{\frac{\partial a_j}{\partial z_j}} \cdot \underset{3}{\frac{\partial z_j}{\partial w_{i,j}}} \quad \text{(Chain Rule)}$$

$$\Downarrow$$

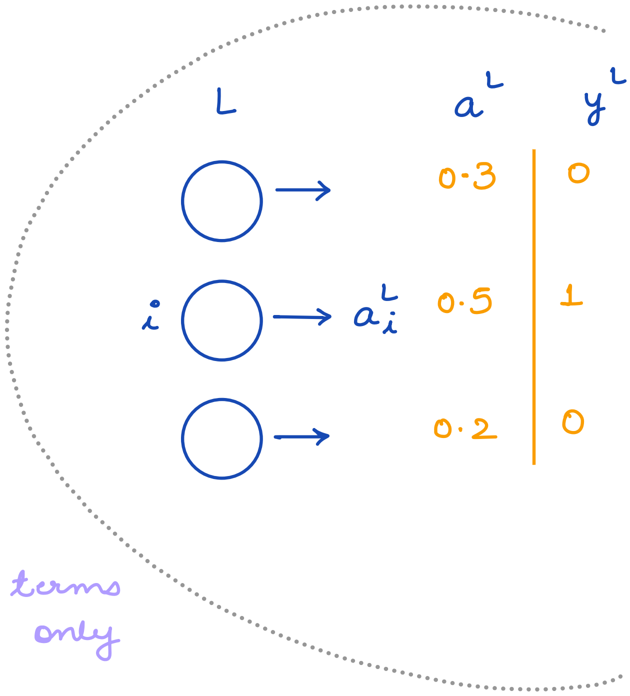
$$\delta_j^L = \frac{\partial L}{\partial z_j}$$

(1) $\frac{\partial L}{\partial a_j}$

$$L = \frac{1}{2} \sum_i \left(y_i^L - a_i^L\right)^2$$

$$\frac{\partial L}{\partial a_j} = \frac{1}{2} \cdot 2 \cdot (y_j - a_j)$$

On differentiating wrt $a_j$, for all other terms it will become 0 and only $(y_j - a_j)$ will be left.




$$\frac{\partial L}{\partial a_j} = -(y_j - a_j)$$

(2) $\frac{\partial a_j}{\partial z_j}$

$$a_j = \sigma(z_j)$$

$$\frac{\partial a_j}{\partial z_j} = \sigma'(z_j)$$

$$= (1 - \sigma(z_j))\,\sigma(z_j)$$

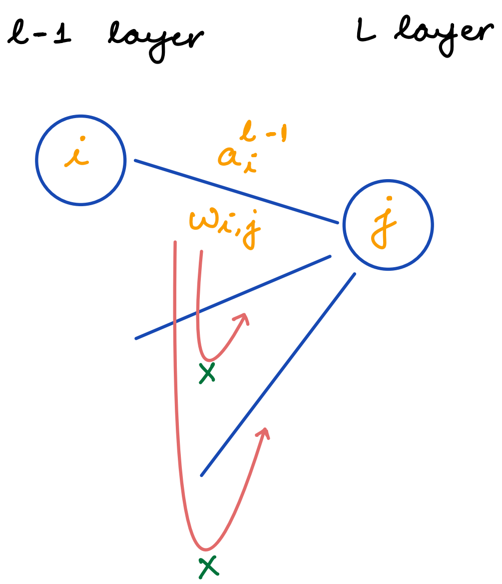
(3) $\frac{\partial z_j}{\partial w_{i,j}}$

$$z_j^l = \sum_i^l w_{i,j} \cdot \underline{a_i^{l-1}} + b_j^l$$

If you take derivative wrt $w_{i,j}$ then you will get activation. For all other terms it will be 0 and only $a_i$ will be left.

$$\frac{\partial z_j}{\partial w_{i,j}} = a_i^{l-1}$$

It is the activation associated with previous layer.




$w_{i,j}$ will depend on $a_i^{l-1}$ activation.

$w_{i,j}$ won't be influenced by other activation values.

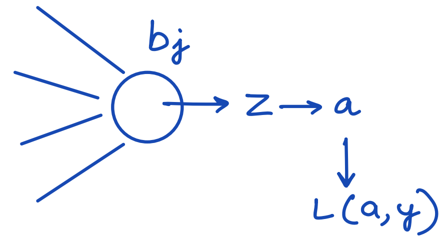
Combining 1, 2 and 3

$$\frac{\partial L}{\partial w_{i,j}} = \underbrace{-(y_j - a_j)\,\sigma'(z_j)}_{\delta_j^L}\; a_i^{l-1}$$

$$\frac{\partial L}{\partial w_{i,j}} = \delta_j^L \cdot a_i^{l-1}$$

**Bias**

$$\frac{\partial L}{\partial b_j} = \underbrace{\frac{\partial L}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j}}_{\delta_j^L} \cdot \frac{\partial z_j}{\partial b_j}$$




$$z_j^l = \underbrace{\sum_i^l w_{i,j}\, a_i^{l-1}}_{\text{will become } 0} + b_j^l$$

will become 1

$$\frac{\partial z_j}{\partial b_j} = 1$$

$$\frac{\partial L}{\partial b_j} = \delta_j^l \cdot 1$$

for output layer,

$$\frac{\partial L}{\partial \omega_{i,j}} = \delta_j^L \cdot a_i^{l-1}$$

$$\frac{\partial L}{\partial b_j} = \delta_j^L \cdot 1$$

$$\delta_j^L = -(y_j - a_j)\,\sigma'(z_j)$$

$$\delta_j^L = (a_j - y_j)\,\sigma'(z_j)$$