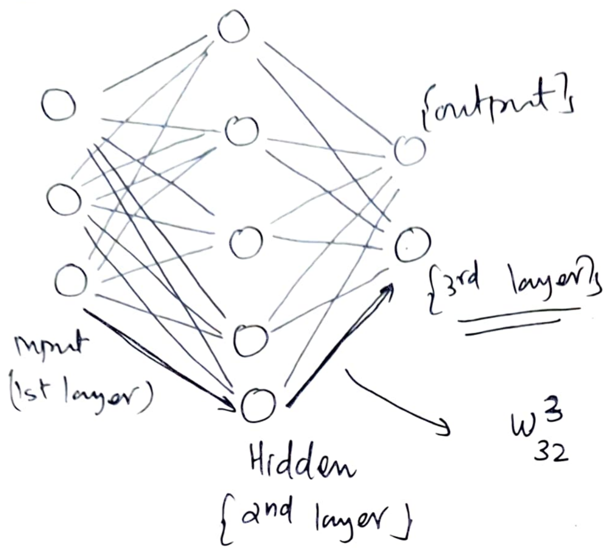
Notations :-




$l$ = final layer no.

$j$ = neuron no. in final layer

$k$ = neuron no. in initial layer.

$w^{l}_{jk}$ =

$b^{l}_{j}$ = bias in j-th neuron in $l$th layer

$a^{l}_{j}$ = activation of jth neuron in $l$th layer.

⇒ Vectorizing activation

activation of $a^{l}_{j} = \sigma\left(\underbrace{\sum_{k} w^{l}_{jk}\, a^{l-1}_{k}}_{\text{dot product.}} + b^{l}_{j}\right)$

this implies

$$\boxed{\vec{a}^{l} = \sigma\left(W^{l}\vec{a}^{l-1} + \vec{b}^{l}\right)}$$

Remember,

weight matrix is of form, $w^{l}_{jk}$ $\begin{cases} j = \text{no. rows} \\ k = \text{no. coloums.} \end{cases}$

$$W = \begin{vmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \vdots & \vdots & & \vdots \\ A_{j1} & A_{j2} & \cdots & A_{jk} \end{vmatrix}$$

& $\boxed{W\vec{a} + \vec{b}}$

$$= \begin{vmatrix} A_{11} & \cdots & A_{1k} \\ \vdots & \ddots & \vdots \\ A_{j1} & \cdots & A_{jk} \end{vmatrix} \cdot \begin{vmatrix} a_{1} \\ \vdots \\ a_{k} \end{vmatrix} + \begin{vmatrix} b_{1} \\ \vdots \\ b_{k} \end{vmatrix}$$

Results in coloum matrix
⇒ a vector of activation.

# Cost functions :

1. quadratic cost :

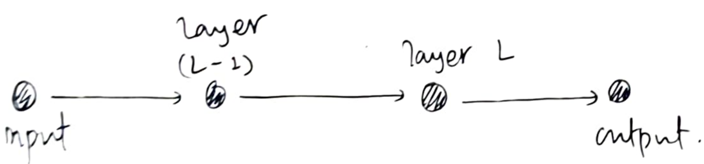
$$C = \frac{1}{2n} \sum_x \| \underbrace{y(x)}_{\text{"desired output"}} - \underbrace{a^L(x)}_{\text{"activation of last layer"}} \|^2 \quad \Big\} \; x = \text{no. of input.}$$

- defines how good / bad is our model performing with some weights & bias over a Example input.

- Other's later in hands on Experimentation.

# Foward pass & Back propogation :-

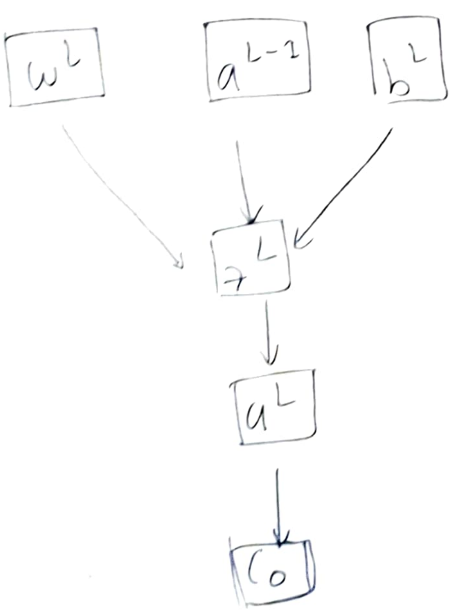
imagine a simple network of neurons,




The cost of the final layer $C_0 = (a^L - y)^2 / 2$

activation $a^L = \sigma(z^L)$ $\{ z^L = (w^L a^{L-1} + b^L)$

the computational graph is,




input with weight of L-1-th layer gives $a^{L-1}$ and that activation with subsequant weights & biases gives us $z^L$ weighted input from last layer which againg with activation function generates output.

→ Cost $C_0$ is something we want to minimize, and $W$'s & $b$'s are knobs we can turn & twist

→ Lets look at how changes in $W_L$ changes $C_0$

$$\frac{\partial C_0}{\partial W_L} = \frac{\partial}{\partial W_L}\left[\left(\sigma(z^L) - y\right)^2/2\right]$$

$$\frac{\partial C_0}{\partial W_L} = \frac{\partial z^L}{\partial W^L}\,\frac{\partial a^L}{\partial z^L}\,\frac{\partial C_0}{\partial a^L} \quad \Big\} \text{ chain rule.}$$

$$\boxed{\frac{\partial C_0}{\partial W_L} = a^{L-1}\,\sigma'(z^L)\cdot(a^L - y)}$$

Similarly,

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial z^L}{\partial b^L}\,\frac{\partial a^L}{\partial z^{(L)}}\,\frac{\partial C_0}{\partial a^L}$$

$$\boxed{\frac{\partial C_0}{\partial b^L} = W^L\,\sigma'(z^L)\,(a^L - y)}$$

$C_0$ depends on $W_L$ & $b$ ⟹ $\nabla C_0 = \underbrace{\left(\frac{\partial C}{\partial W^L}, \frac{\partial C}{\partial b}\right)^T}_{\text{gradient decent step!!}}$

For our simple network,

Total cost $\boxed{\frac{\partial C}{\partial W^L} = \frac{1}{n}\sum_{k=0}^{n-1}\frac{\partial C_k}{\partial W^L}}$ → we can make the errors smaller & make correct-er prod.

# Formal - 4 Equations of back propagation

definition : $\delta_j^l$ - "Error" is,

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l} \quad \} \text{ Error Eq.}$$

Eq 1. : "Error in output layer"

$$\delta_j^L = \underbrace{\frac{\partial C}{\partial a_j^L}}_{\substack{\text{change of} \\ C \text{ w.r.t activations}}} \underbrace{\sigma'(z_j^L)}_{\text{change in } \sigma \text{ w.r.t } z_j^L}$$

These are the result of chain rule in ~~of~~ Error Eq.

→ converting it to matrix form,

$$\boxed{\delta_j^L = (a^L - y) \underbrace{\odot}_{\substack{\text{Hadamard} \\ \text{product.}}} \sigma'(z^L)}$$

Eq 2. Error $\delta^l$ as $\delta'(\delta^{l+1})$ :

$$\boxed{\delta^l = \left( (w^{l+1})^T \delta^{l+1} \right) \odot \sigma'(z^L))}$$

propagating error backwards.

Eq 3. Rate of change of cost wrt bias:

$$\boxed{\frac{\partial C}{\partial b_j^l} = \delta_j^l}$$

Eq 4. Rate of change of cost wrt weight:

$$\boxed{\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\,\delta_j^l}$$

{a in δout?}

# Back propagation Algorithm

① Input $x$ : get activations in $a^1$

② Feed forward : for each layer compute

$$z^l = w^l a^{l-1} + b^l \quad \& \quad a^l = \sigma(z^l)$$

③ Error : $\delta^L = (a^L - y) \odot \sigma'(z^L) \rightarrow$ Eq 1.

④ Back propagate errors : $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$

⑤ adjust weights & Biases :

Update Eq.

$$\boxed{\begin{aligned} w^l &\rightarrow w^l - \eta\, \delta^{l} (a^{l-1})^T \\ b^l &\rightarrow b^l - \eta\, \delta^{l} \end{aligned}}$$