⇒ Week-5

⇒ Cost function for neural nets

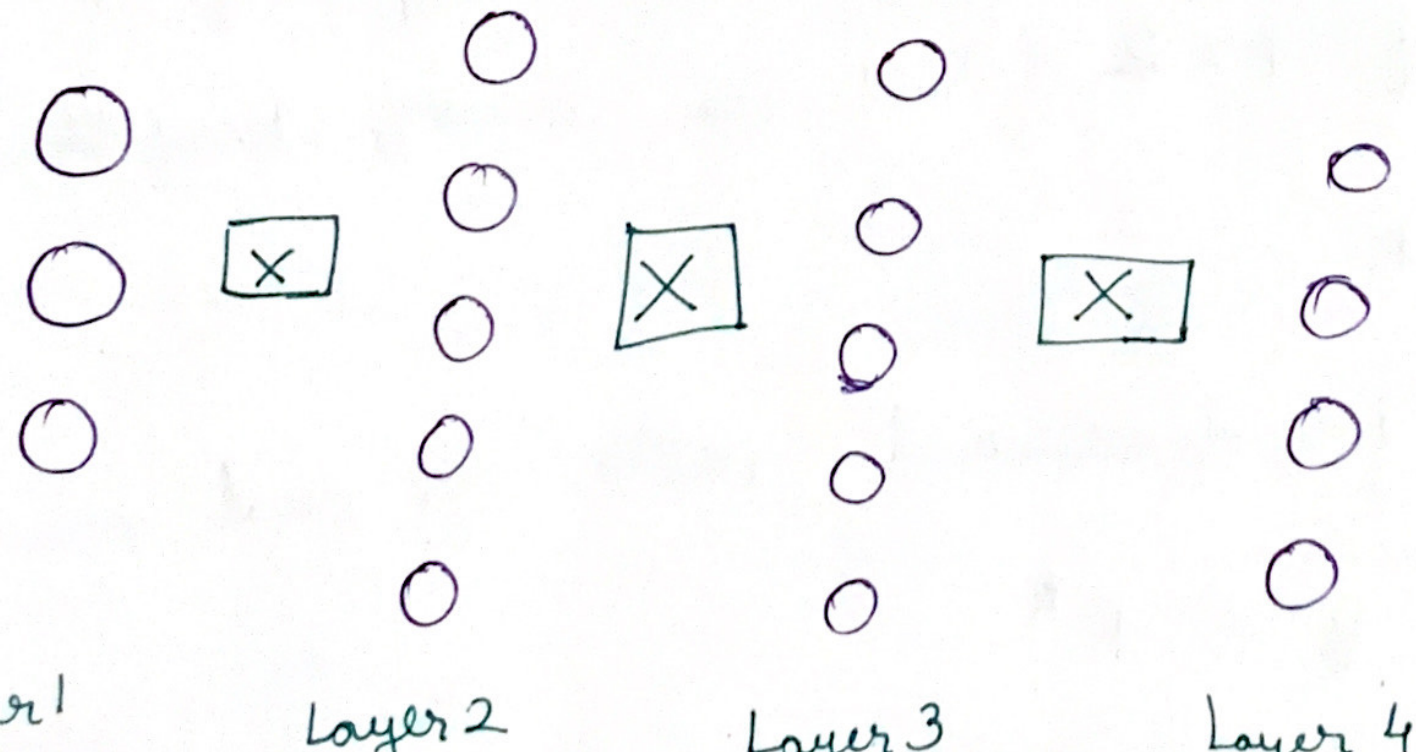
* Lets define a few variables that we can use:

→ L = total number of layers in the network

→ s_l = number of units (not counting bias unit) in layer l

→ K = no. of output units/classes

eg:



Layer 1 Layer 2 Layer 3 Layer 4

→ [X] : denotes that every ^(each) neuron in the layer before ^(the) of the layer which is before the 'mark' is connected to each neuron of the layer present after the 'mark'

⇒ Here, $L = 4$, $s_1 = 3$, $s_2 = 5$, $s_3 = 5$, $s_4 = 4$

$K = 4$

⇒ Types of classification problems with neural nets

→ 1st type : Binary classification

* 1 output (0 or 1)
* $K = 1$
* $S_L = 1$

→ 2nd type : Multiclass classification

* k distinct classifications. (Typically $k \geq 3$)
* → If $k = 2$ (Just use binary classification)
* $S_L = k$
* $y \rightarrow k$ dimensional vector of real numbers

or $y \in R^k$

eg: $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Pedestrian Car Bike Truck

⇒ Cost function

* Recall that the cost func^n for regularized logistic regression was :

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^i \log(h_\theta(x^i)) + (1-y^i)\log(1-h_\theta(x^i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

* For neural nets, our cost function is a generalization of the eq^n above. Instead of one output we generate k outputs.

$$h_\theta(x) \in R^{*K} \; ; \; (h_\theta(x))_i = i^{th} \text{ output}$$

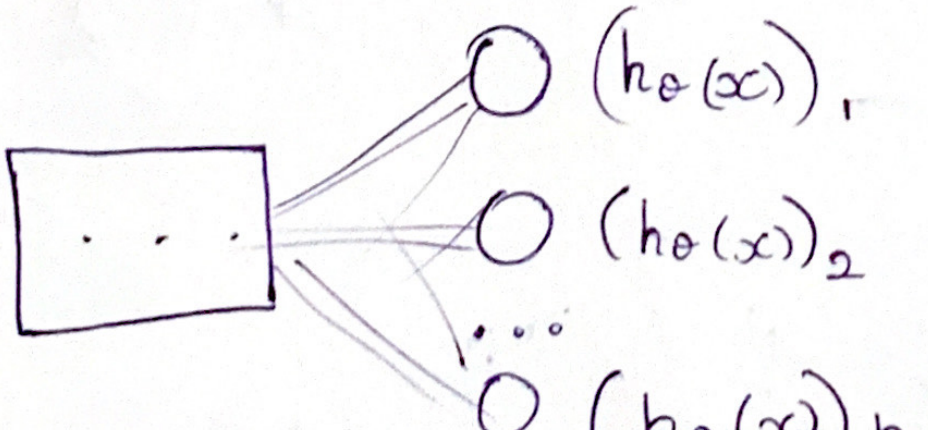
$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k{}^i \log(h_\theta(x^i))_k + (1-y_k{}^i)\cdot \log(1-(h_\theta(x^i))_k)\right] + \frac{\lambda}{2m}\sum_{L=1}^{L-1}\sum_{i=1}^{S_L}\sum_{j=1}^{S_{L+1}}(\theta_{ji}{}^L)^2$$

... Daymmmm..

→ There are 2 halves to the neural net logistic regression cost function

→ 1st half: $-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^i \log(h_\theta(x^i))_k + (1-y_k^i)\cdot \log(1-(h_\theta(x^i))_k)\right]$

* This is just saying for each training example (i.e. 1 to m - the first summation), calculate the logistic regression for every output neuron in the output layer (i.e. 1 to K - the second summation) & averaging the summation over m → total training examples)



TL;DR: The double sum simply adds up the logistic regression costs calculated for each cell in the output layer.

→ 2nd half : $\frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{l})^2$

The triple sum simply adds up the squares of all the individual θs in the entire network.

* Note the i in the triple sum does NOT refer to training example i.

* Similar to logistic regression, the bias terms are not included in the triple sum.

⇒ Backpropagation algorithm : Algorithm used to minimize the cost function of our neural network.

(just like what we were doing with gradient descent in logistic & linear regression)

* Our goal is to compute :

$$\min_{\Theta} J(\Theta)$$

* To compute this ↑ we can use algorithms such as gradient descent & advance ~~optimization~~ optimization algorithms; which require 2 things of the following:

1) $J(\Theta)$ → i.e. the cost function itself! Just calculate it manually & code it!

2) $\frac{\partial J(\Theta)}{\partial \Theta_{i,j}}$ → This is NOT AT ALL easy!

* Θ is indeed in 3 dimensions because we have separate parameter values for each node in each layer going to each node in the following layer.

* Remember $\frac{\partial}{\partial \Theta_{ij}^{L}} J(\Theta)$ is a REAL number (not a vector or a matrix) for each Θ.

⇒ Mechanics behind back propagation

* For each node, we can calculate (δ_j^l) - this is the error of node j in layer L.

* $a_j^L \rightarrow$ activation of node j in layer l.

Totally theoretical value, so we can expect some error compared to the "real" value

* The delta term captures this error!

* But the problem is, "What is this "real" or ideal value that'd give us optimum results) & how do we calculate it?"

The NN is a totally artificial construct. So, the only "real" value we have is our actual classification (our y value) — so that's where we start!

* Looking at the example on pg-1 (Week 5) & looking at the fourth (or output) layer,

$$\delta_j^4 = a_j^4 - y_j$$

$$= \left[\begin{array}{l}\text{Activation of} \\ \text{the unit}\end{array}\right] - \left[\begin{array}{l}\text{Actual value observed} \\ \text{in training example}\end{array}\right]$$

$$= h_\theta(x)_j - y_j$$

* $\boxed{\delta^4 = a^4 - y}$ → Vectorized implementation

Vector of errors in 4^{th} layer

Vector of activation values for the 4^{th} layer

* The idea behind backpropagation is that once you what the error in your output nodes is, you backpropagate this error to further calculate error in nodes present in the previous layer(s). by using the below formula (derived later):

$$\boxed{\delta^L = ((\Theta^L)^T \delta^{L+1}) \overset{*}{.} g'(z^L)}$$

Element wise multiply

* → If $g(z) \equiv$ Sigmoid func$^n = 1/1+e^{-z}$

then $\boxed{g'(z) = g(z) .* (1 - g(z))}$ [Derived later]

$\therefore$ $\boxed{\delta^L = ((\Theta^L)^T \delta^{L+1}) .* a^L .* (1 - a^L)}$

* Using this we can calculate $\frac{\partial J(\theta)}{\partial \theta_{i,j}^{L}}$ in the following way: (ignoring the regularization term, will derive it later)

$$\frac{\partial J(\theta)}{\partial \theta_{i,j}^{L}} = a_j^{L} \; \delta_i^{L+1}$$

→ For m training examples, we can just calculate the average (mean) of all m examples:

$$\therefore \frac{\partial J(\theta)}{\partial \theta_{i,j}^{L}} = \frac{1}{m} \sum_{t=1}^{m} a_j^{t\,L} \; \delta_i^{t\,(L+1)}$$

For each weight and this'll be calculated ↑ Will be fed into advanced optimization algos to perform gradient descent. eg: In Matlab, we use fminunc

* Intuitively, the gradient $\frac{\partial J(\theta)}{\partial \theta_{i,j}^{L}}$ shows how much the weight $\theta_{i,j}^{L}$ needs to change (increase or decrease) in order to minimize $J(\theta)$.

* Now, once we know (Upar Upar se :-),

~~we actually~~ the mechanics of backprop,

we can finally put these together into a

formal backprop algorithm:

→

Given training set $\{(x^1, y^1), \ldots, (x^m, y^m)\}$

Set $\Delta_{ij}^{l} = 0 \quad \forall\ l, i, j$

For training sample $t = 1$ to m:

Set $a^1 = x^t$

Perform forward propagation to compute a^l for $l = 2, 3, \ldots, L$

Using y^t, compute $\delta^L = a^L - y^t$

Compute $\delta^{L-1}, \delta^{L-2}, \ldots, \delta^2$

$\Delta_{ij}^{l} := \Delta_{ij}^{l} + a_j^{l} \delta_i^{l+1} \quad \forall\ i, j, l$

$D_{ij}^{l} := \frac{1}{m} \Delta_{ij}^{l} + \lambda \theta_{ij}^{l}$ if $j \neq 0$

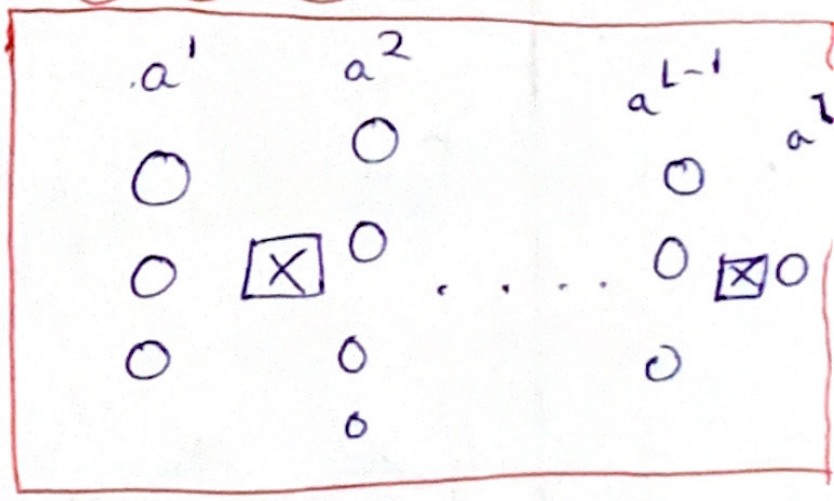
$D_{ij}^{l} := \frac{1}{m} \Delta_{ij}^{l}$ if $j = 0$

* $\Delta_{ij}^{l} = 0 \quad \forall i, j, l$: Used to compute $\frac{\partial J(\theta)}{\partial \theta_{ij}^{l}}$

Just initialize a matrisc full of 0s.

* For each training example $t = 1$ to m :

1) Set $a^1 := x^t$



2) Perform forward propagation to compute

a^l for $l = 2, 3, \ldots, L$

→ For one training example (x, y) :

Forward propagation :

$$a^1 = x$$
$$z^2 = \theta^1 a^1$$
$$a^2 = g(z^2)$$
$$\vdots$$
$$z^L = \theta^{L-1} a^{L-1}$$

3) Using y^t, compute $\delta^L = a^L - y^t$

→ L = Total number of layers

→ a^L = Vector of outputs of the activation units for the last layer

→ Error values for last layer are simply b/w our ~~theoretical~~ value & actual one.

4) Compute $\delta^{L-1}, \delta^{L-2}, \ldots, \delta^2$

→ Use $\boxed{\delta^L = \left((\Theta^L)^T \delta^{L+1}\right) .* a^L .* (1 - a^L)}$

5) $\Delta_{ij}^L := \Delta_{ij}^L + a_j^L \delta_i^{L+1}$

→ $\boxed{\Delta^L := \Delta^L + \delta^{L+1} (a^L)^T}$ → Vectorized implementation

Update

→ The 'D' matrix

* [illegible] 'D' : Calculates mean of all the values of [illegible] Δ_{ij}^{l} for all training samples m.

* $D_{ij}^{l} := \frac{1}{m} \left(\Delta_{ij}^{l} + \lambda \theta_{ij}^{l} \right)$ if $j \neq 0$

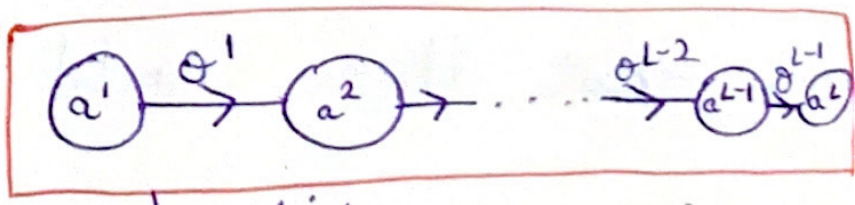
$D_{ij}^{l} := \frac{1}{m} \Delta_{ij}^{l}$ if $j = 0$ } NO regularization for bias terms !

⇒ Exploration of derivatives used in backpropagation

* We know that for a logistic regression classifier (which is what all of the output neurons in a neural network are), we use the cost func^n :

$$J(\theta) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

& apply this over the 'K' output neurons, and for all 'm' samples. In the full sample



say our NN looks like this

* The eq^n to compute the partial derivatives of the theta terms in the output neurons :

$$\frac{\partial J(\theta)}{\partial \theta^{(L-1)}} = \frac{\partial J(\theta)}{\partial a^L} \frac{\partial a^L}{\partial z^L} \frac{\partial z^L}{\partial \theta^{L-1}}$$

[From chain rule]

Notice we are computing this for a single neuron & we've dropped i completely for simplification.

* The eq^n to compute partial derivatives of the theta terms in the [last] hidden layer neurons (i.e. neurons are of layer L-1):

$$\frac{\partial J(\theta)}{\partial \theta^{L-2}} = \frac{\partial J(\theta)}{\partial a^L} \frac{\partial a^L}{\partial z^L} \frac{\partial z^L}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial z^{L-1}} \frac{\partial z^{L-1}}{\partial \theta^{L-2}}$$

* Clearly, the 2 eq^{n}s share some pieces in common, so a delta term (δ^L) can be used for the common pieces b/w the output layer & the hidden layer immediately before it (with the possibility that there could be many hidden layer if we wanted):

$$\delta^L = \frac{\partial J(\theta)}{\partial a^L} \frac{\partial a^L}{\partial z^L}$$

* And we can go ahead & use another delta term (δ^{L-1}) for the pieces that would be shared by the final hidden layer & a hidden layer before that, if we had one.

$$\delta^{L-1} = \frac{\partial J(\theta)}{\partial a^L} \frac{\partial a^L}{\partial z^L} \frac{\partial z^L}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial z^{L-1}}$$

$$\therefore \boxed{\delta^{L-1} = \delta^L \frac{\partial z^L}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial z^{L-1}}}$$

* With these delta terms, our equations become:

$$\boxed{\frac{\partial J(\theta)}{\partial \theta^{L-1}} = \delta^L \frac{\partial z^L}{\partial \theta^{L-1}}}$$

&

$$\boxed{\frac{\partial J(\theta)}{\partial \theta^{L-2}} = \delta^{L-1} \frac{\partial z^{L-1}}{\partial \theta^{L-2}}}$$

⇒ Now, lets calculate these derivatives

→ Deriving the sigmoid gradient fun^n

$\sigma(x) \equiv$ Sigmoid function $= \dfrac{1}{1+e^{-x}}$

$\sigma'(x) = \dfrac{d}{dx}\left(1+e^{-x}\right)^{-1}$

$= -1\left(1+e^{-x}\right)^{-2} \dfrac{d}{dx}\left(1+e^{-x}\right)$

$= \dfrac{-1}{\left(1+e^{-x}\right)^2} e^{-x}(-1)$

$\therefore \quad \sigma'(x) = \dfrac{e^{-x}}{\left(1+e^{-x}\right)^2} = \left(\dfrac{1}{1+e^{-x}}\right)\left(\dfrac{e^{-x}}{1+e^{-x}}\right)$

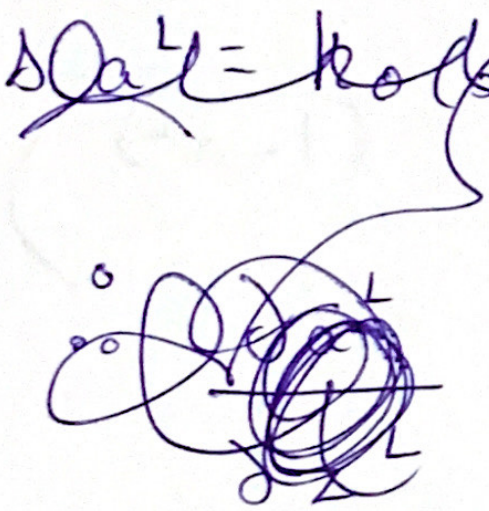
$= \left(\dfrac{1}{1+e^{-x}}\right)\left(\dfrac{1+e^{-x}}{1+e^{-x}} - \dfrac{1}{1+e^{-x}}\right)$

$\therefore \quad \boxed{\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))}$

→ Lets start with the output layer :

$$\frac{\partial J(\theta)}{\partial \theta^{L-1}} = \delta^{L} \frac{\partial z^{L}}{\partial \theta^{L-1}} \quad ; \quad \delta^{L} = \frac{\partial J(\theta)}{\partial a^{L}} \frac{\partial a^{L}}{\partial z^{L}}$$

* Given $J(\theta) = -y \log(a^{L}) - (1-y) \log(1-a^{L})$



$$\therefore \frac{\partial J(\theta)}{\partial a^{L}} = \frac{-y}{a^{L}} + \frac{(1-y)}{(1-a^{L})}$$

* Given $a = g(z)$ where $g = \frac{1}{1+e^{-z}}$

$$\therefore \frac{\partial a^{L}}{\partial z^{L}} = a^{L}(1-a^{L})$$

{ Using the derivation on prev. page

* Substituting values of $\frac{\partial a^L}{\partial z^L}$ & $\frac{\partial J(\theta)}{\partial a^L}$ in the eqn of δ^L :

$$\delta^L = \frac{\partial J(\theta)}{\partial a^L} \frac{\partial a^L}{\partial z^L}$$

$$= \left(\frac{1-y}{1-a^L} - \frac{y}{a^L} \right) \left(a^L (1-a^L) \right)$$

$$= \frac{a^L (1-y) - y(1-a^L)}{a^L (1-a^L)} \cdot \left[a^L (1-a^L) \right]$$

$$= a^L - y a^L - y + a^L y$$

$$\therefore \boxed{\delta^L = a^L - y}$$

* Given $z^L = \theta^{L-1} a^{L-1}$

$$\therefore \boxed{\frac{\partial z^L}{\partial \theta^{L-1}} = a^{L-1}}$$

→ Putting it all together for the final layer

$$\frac{\partial J(\theta)}{\partial \theta^{L-1}} = \delta^L \frac{\partial z^L}{\partial \theta^{L-1}}$$

$$\therefore \boxed{\frac{\partial J(\theta)}{\partial \theta^{L-1}} = (a^L - y)\ (a^{L-1})}$$

→ Lets continue on for the hidden layer :

$$\frac{\partial J(\theta)}{\partial \theta^{L-2}} = \delta^{L-1} \frac{\partial z^{L-1}}{\partial \theta^{L-2}}$$

$$* \quad \frac{\partial z^{L}}{\partial a^{L-1}} = \theta^{L-1} \quad \{ \text{As } z^{L} = \theta^{L-1} \cdot a^{L-1} \}$$

$$* \quad \frac{\partial a^{L-1}}{\partial z^{L-1}} = a^{L-1}(1 - a^{L-1}) \quad \{ \text{As } \sigma'(z) = \sigma(z)[1-\sigma(z)] \}$$

$$\therefore \delta^{L-1} = \delta^{L} \frac{\partial z^{L-1}}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial z^{L-1}}$$

$$\therefore \boxed{\delta^{L-1} = \delta^{L} \theta^{L-1} a^{L-1}(1 - a^{L-1})}$$

$$\therefore \frac{\partial J(\theta)}{\partial \theta^{L-2}} = \left(\delta^{L-1}\right) \frac{\partial z^{L-1}}{\partial \theta^{L-2}} \quad \{ \text{Use } z^{L-1} = \theta^{L-2} a^{L-2} \}$$

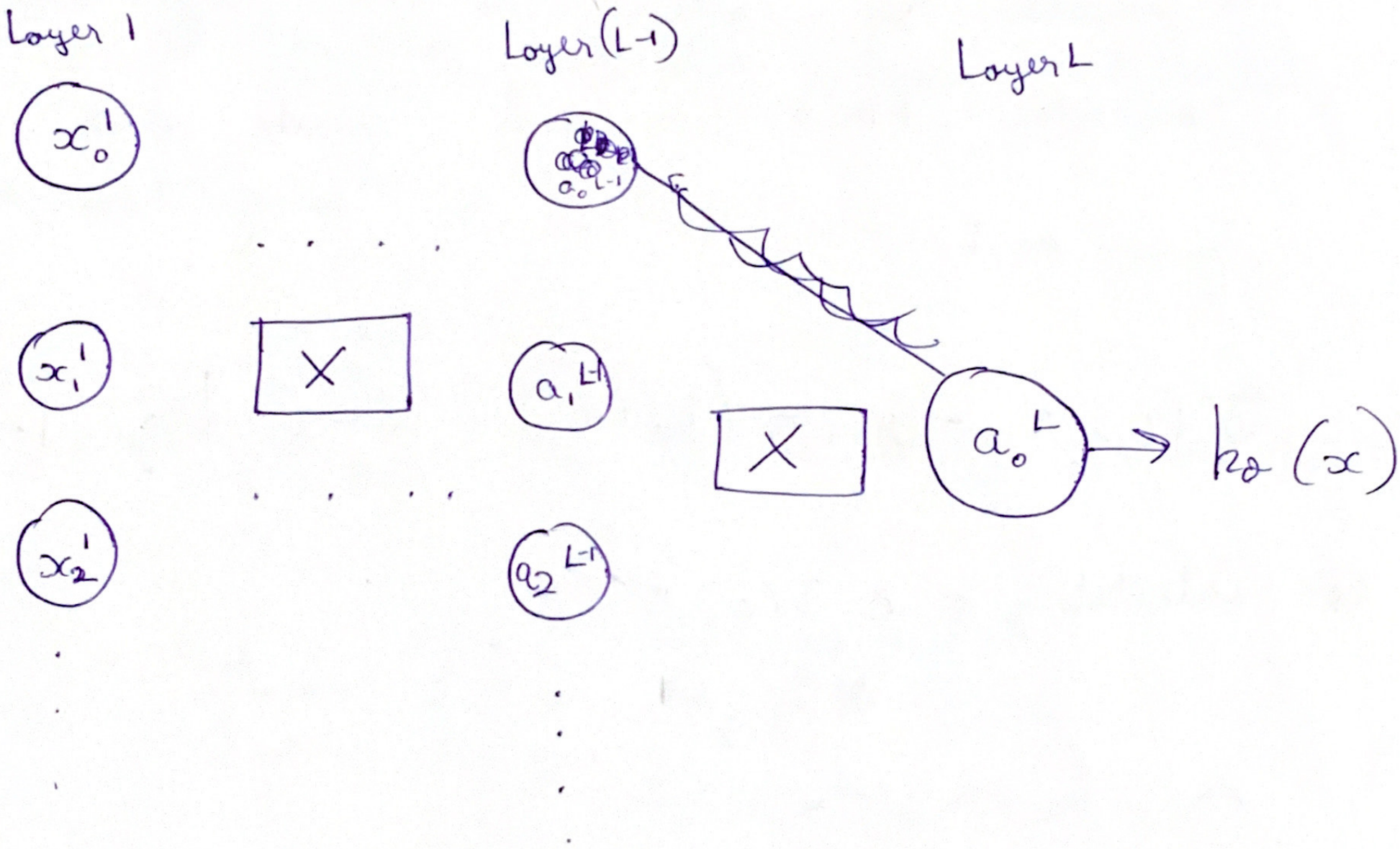
$$= \left(\delta^{L} \frac{\partial z^{L}}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial z^{L-1}}\right) a^{L-2} \quad \{ \text{Use } z^{L} = \theta^{L-1} a^{L-1} \}$$

$$\therefore \boxed{\frac{\partial J(\theta)}{\partial \theta^{L-2}} = \left[(a^{L} - y)(\theta^{L-1})(a^{L-1})(1-a^{L-1})\right](a^{L-2})}$$

$$= \delta^{L-1}$$

has fully connected layers

* Suppp. our neural net ^ looks something like this :



For any arbitrary node j in layer L

$$a_j^L = g(z_j^L) = g(\theta_{j,0}^{L-1} \cdot a_0^{L-1} + \theta_{j,1}^{L-1} a_1^{L-1} + \dots + \theta_{j,s_{L-1}}^{L-1} a_{s_{L-1}}^{L-1})$$

where s_{L-1} = no. of nodes in layer $(L-1)$

a_k^{L-1} = k^{th} node/neuron of layer $(L-1)$

* Since we have only one output node in our network, we use (or drop) the using similar arguments as earlier (in the single neuron connected NN) we can write our cost func^{n} as:

$$J(\theta) = -y \log(h_\theta(x)) - (1-y) \log(1-h_\theta(x))$$

* Calculating $\frac{\partial J(\theta)}{\partial \theta_{i,y}^{L-1}}$:

$$\frac{\partial J(\theta)}{\partial \theta_{i,y_L}^{L-1}} = \frac{\partial J(\theta)}{\partial a_{y_L}^{L}} \frac{\partial a_{y_L}^{L}}{\partial z_{y_L}^{L}} \frac{\partial z_{y_L}^{L}}{\partial \theta_{i,y_L}^{L-1}}$$

* For prev. layer or [last] hidden layer :

$$\frac{\partial J(\theta)}{\partial \theta_{i,y}^{L-2}} = \frac{\partial J(\theta)}{\partial a}$$