Neural Network

Inputs : (x, y) ; $x \in \mathbb{R}^d$, $y \in \mathbb{R}^m$ where 'd' is the number of features and 'm' is the number of classes in the final output.

$$x = [x_1, x_2, x_3, x_4 \dots x_n] \quad y = [y_1, y_2 \dots y_m]$$

There are N number of examples [tuple (x, y)] but for simplicity we will just focus on one example i.e $(x, y) \leftarrow$ one data point.

* Notation

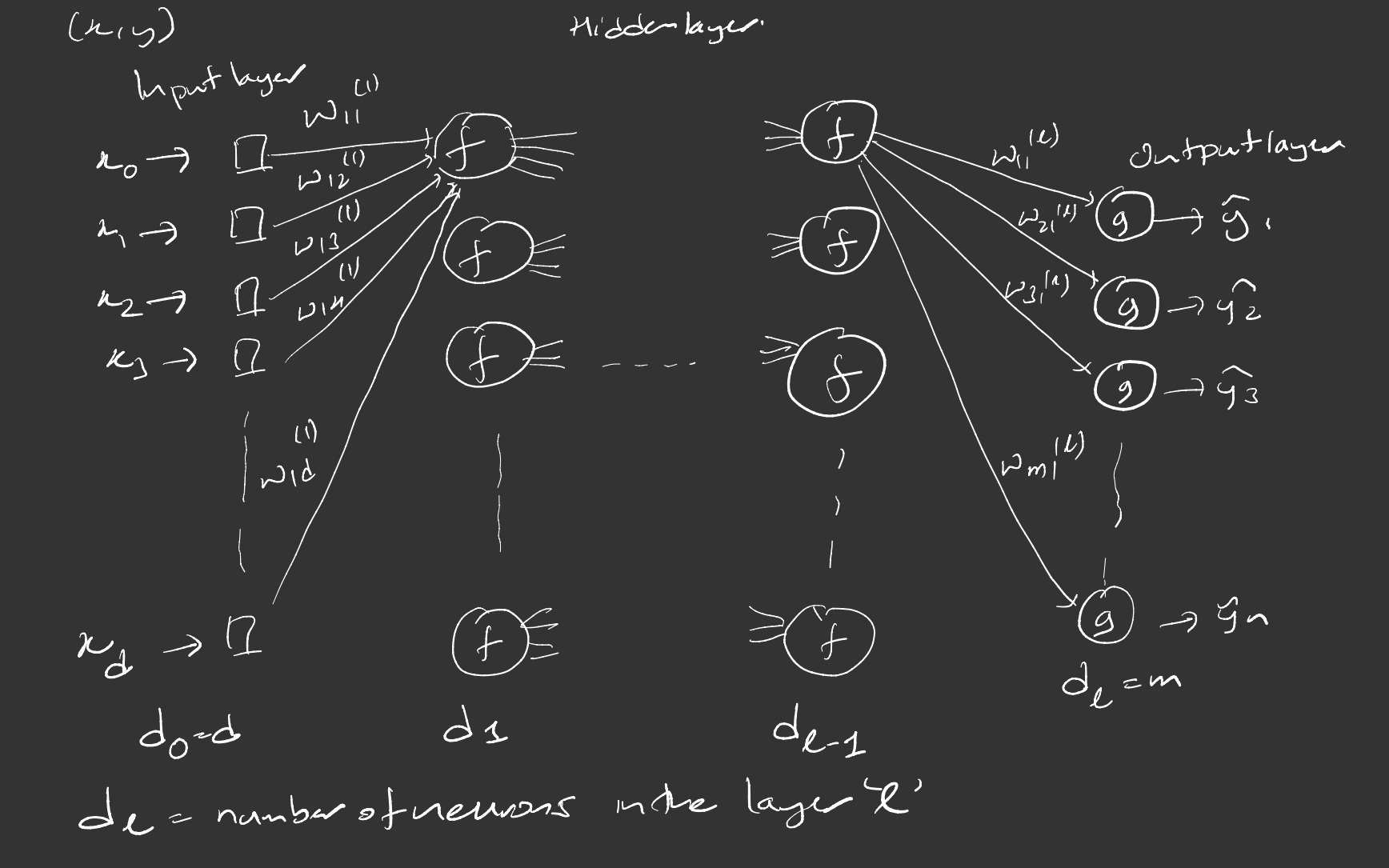
For inputs $x_i^{(l)}$: $l \leftarrow$ layer, $i \leftarrow i^{th}$ value in vector

$$1 \le l \le N \qquad 0 \le i \le d_l$$

For weights $w_{ji}^{(l)} \leftarrow$ layer

$i \rightarrow$ from i^{th} neuron of $(l-1)^{th}$ layer

$j \rightarrow$ going to j^{th} neuron of l^{th} layer


(x, y)
Input layer
Hidden layer
Output layer
x_0
x_1
x_2
x_3
x_d
$w_{11}^{(1)}$
$w_{12}^{(1)}$
$w_{13}^{(1)}$
$w_{14}^{(1)}$
$w_{1d}^{(1)}$
$w_{11}^{(L)}$
$w_{21}^{(L)}$
$w_{31}^{(L)}$
$w_{m1}^{(L)}$
f
g
$\hat{y}_1$
$\hat{y}_2$
$\hat{y}_3$
$\hat{y}_n$
$d_0 = d$
d_1
d_{L-1}
$d_L = m$
d_L = number of neurons in the layer 'L'

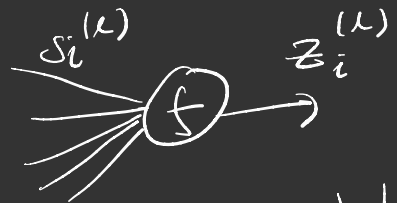
* More Notation

For targets : y_i ← i^{th} value of vector y. y is the actual output

For output : $\hat{y}_i$ ← i^{th} value of vector $\hat{y}$. $\hat{y}$ is the predicted output of example 'x'

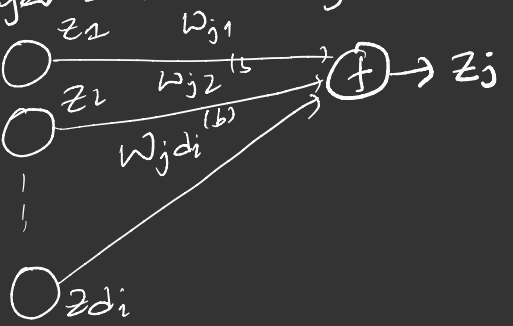
for hidden neurons:

- weighted sum : $s_i^{(l)}$ ← i^{th} neuron, (l) ← layer
- output of a neuron : $z_i^{(l)}$ ← i^{th} neuron, (l) ← layer

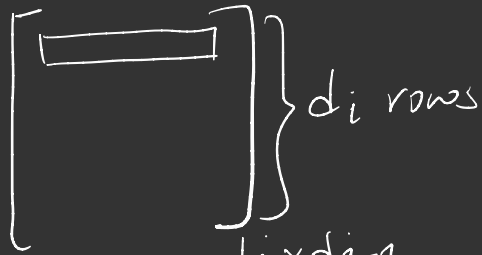


Then, $z_j^{(b)} = f(s_j^{(b)})$ where f is an activation function.

And, $s_j^{(b)} = \sum_{i=0}^{d_{b-1}} w_{ji}^{(b)} z_i^{(b-1)}$

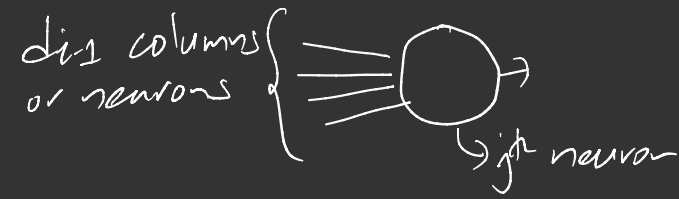


Then in vector notation.

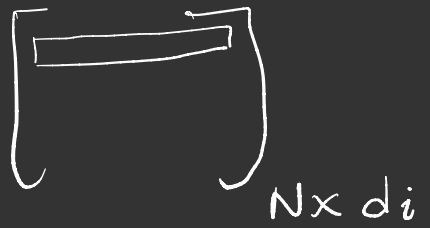
$W_{i \leftarrow i^{th} \text{ layer}}$ = 

each row has weights of j^{th} neuron in layer i

layer-i



Similarly,

$X_{i \leftarrow i^{th} \text{ layer}}$ = 

layer i

$\rightarrow z_1$

$\rightarrow z_2$

$\rightarrow z_3$

$\rightarrow z_{d_i}$

N number of examples in N rows.

Each row contains d_i values as the output at i^{th} layer.

Then, $X_i = f(X_{i-1} N_i^T)$

$N \times d_{i-1}$ $\quad$ $d_{i-1} \times d_i$

$\rightarrow N \times d_i$

* Functions

i) Logistic : $f(s_i) = \frac{1}{1+e^{-s_i}}$ $\quad f'(s_i) = f(s_i)[1-f(s_i)]$

ii) Softmax: normalizing function : $\mathbb{R}^N \rightarrow \mathbb{R}^N$

$$g(s_i) = \frac{e^{s_i}}{\sum_{i=1}^{m} e^{s_i}} = \hat{y}_i$$

$$\begin{bmatrix} \frac{\partial \hat{y}_1}{\partial s_1} & \frac{\partial \hat{y}_1}{\partial s_2} & \cdots & \frac{\partial \hat{y}_1}{\partial s_n} \\ \vdots & & & \vdots \\ \frac{\partial \hat{y}_n}{\partial s_1} & \cdots & & \frac{\partial \hat{y}_n}{\partial s_n} \end{bmatrix}_{i \times j}$$

Now, finding the derivative of softmax.

$$\frac{\partial g(s_i)}{\partial s_j} = \frac{\sum_{i=1}^{m} e^{s_i} \times e^{s_i} \{i=j\} - e^{s_i} \times e^{s_j}}{\left(\sum_{i=1}^{m} e^{s_i}\right)^2}$$

$$= \frac{\Sigma \times e^{s_i} \{i=j\} - e^{s_i} \times e^{s_j}}{\Sigma^2}$$

$$= \frac{e^{s_i} \left(\Sigma \{i=j\} - e^{s_j}\right)}{\Sigma \times \Sigma}$$

$$= \frac{e^{s_i}}{\Sigma} \times \frac{\left(\Sigma \{i=j\} - e^{s_j}\right)}{\Sigma}$$

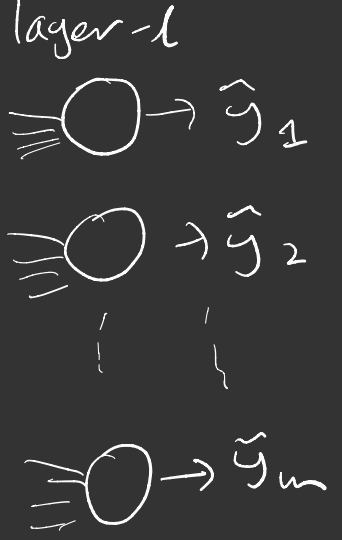
$$= g(s_i) \times \left[1\{i=j\} - g(s_j)\right]$$

So, when $i=j$, $g'(s_i) = g(s_i)\,[1 - g(s_i)]$

when $i \neq j$, $g'(s_j) = g(s_i)\,[-g(s_j)]$

$= -g(s_i)\,g(s_j)$

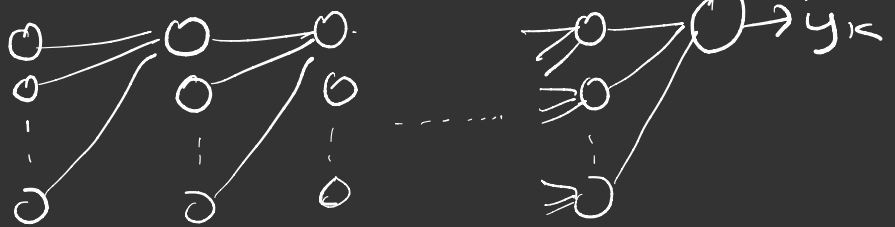
* Forward Propagation (Inference).



$$\hat{y}_k = g(z_k^{(l)})$$

$$= g\left(\sum_{j=0}^{d_{l-1}} w_{kj}^{(l)} z_j^{l-1}\right)$$

$$= g\left[\sum_{j=0}^{d_{l-1}} w_{kj}^{(l)} f\left(\cdots\cdots f\left(\sum_{p=0}^{d} w_{qp}^{(l)} x_p\right)\right)\right]$$



* Cost function : Cross entropy

$$C(w) = -\sum_{n=1}^{N} \sum_{K=1}^{m} y_K \log(\hat{y}_K) \quad \Leftarrow \text{for all examples.}$$

+ for a single example. (x, y)

$$C_i(w) = -\sum_{K=1}^{m} y_K \log(\hat{y}_K) \qquad 1 \le K \le m$$

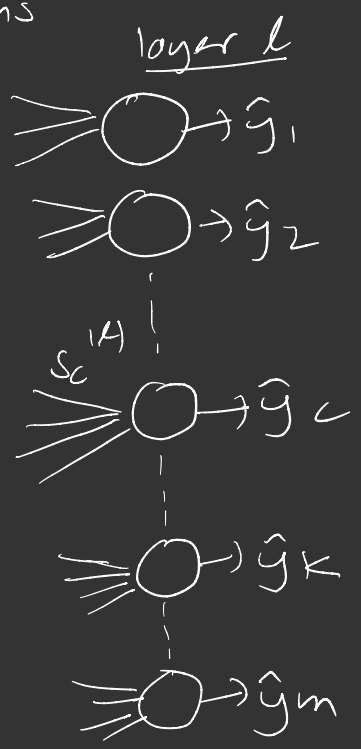
* Now, computing derivatives using back propagation.

$$\frac{\partial C(w)}{\partial w_{cj}^{(l)}} = \frac{\partial C}{\partial s_c^{(l)}} \times \frac{\partial s_c^{(l)}}{\partial w_{cj}^{(l)}} \qquad 1 \le c \le m$$

Here, $\frac{\partial s_c^{(l)}}{\partial w_{cj}^{(l)}} = z_j^{(l-1)}$ $\left[\because s_c^{(l)} = \sum_{j=0}^{d_{l-1}} w_{cj}^{(l)} z_j^{(l-1)} \right]$

Then, computing $\frac{\partial C}{\partial s_c^{(L)}}$ w.r.t weighted sum.

Fixing 'c' between $1 \le c \le m$ and 'k' changes between $1 \le k \le m$
They both represent the output layer neurons



$$\frac{\partial C}{\partial s_c^{(L)}} = \frac{\partial}{\partial s_c^{(L)}} \left[-\sum_{k=1}^{m} y_k \log(\hat{y}_k) \right]$$

$$= -\sum_{k=1}^{m} y_k \frac{\partial}{\partial s_c^{(L)}} \log(\hat{y}_k)$$

$$= -\sum_{k=1}^{m} y_k \frac{\partial}{\partial \hat{y}_k} (\log \hat{y}_k) \times \frac{\partial \hat{y}_k}{\partial s_c^{(L)}}$$

$$= -\sum_{k=1}^{m} y_k \times \frac{1}{\hat{y}_k} \times \hat{y}_k \left(1\{k=c\} - \hat{y}_c\right)$$

$$\left[\because \hat{y}_k = g(s_c^{(a)}) \text{ and } \frac{\partial g(s_i)}{\partial s_j} = g(s_i)\left(1\{i=j\} - g(s_j)\right)\right]$$

$$= -\sum_{k=1}^{m} y_k \left(1\{k=c\} - \hat{y}_c\right)$$

$$= \left[-\sum_{k \neq c}^{m} y_k (-\hat{y}_c)\right] - y_c(1 - \hat{y}_c)$$

$$= \sum_{k \neq c}^{m} y_k \hat{y}_c - y_c + y_c \hat{y}_c$$

$$= \sum_{k=1}^{m} y_k \hat{y}_c - y_c$$

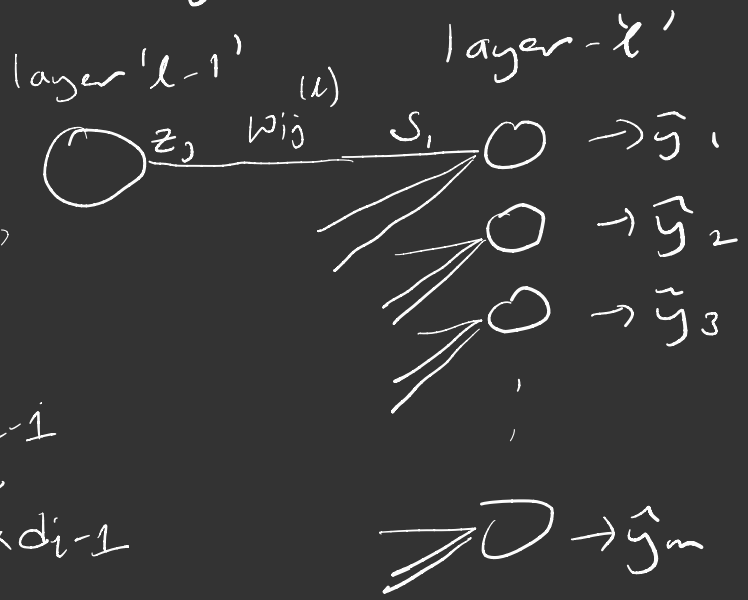
$$= \hat{y}_c \sum_{k=1}^{m} y_k - y_c$$

$$= \hat{y}_c - y_c \qquad \left[\sum_{k=1}^{m} y_k = 1 \text{; sum of probabilities is 1}\right]$$

$$\therefore \frac{\partial C}{\partial w_{cj}^{(L)}} = (\hat{y}_c - y_c) \times z_j^{(L-1)}$$

$\Downarrow$

weights of last layer 'L'
between layers 'L' and 'L-1'



So,

$$W_L = W_L - \frac{\alpha}{|B|} (\hat{Y} - Y)^T X_{L-1}$$

W_L: $m \times d_{i-1}$; $(\hat{Y} - Y)^T$: $m \times B$; X_{L-1}: $B \times d_{i-1}$

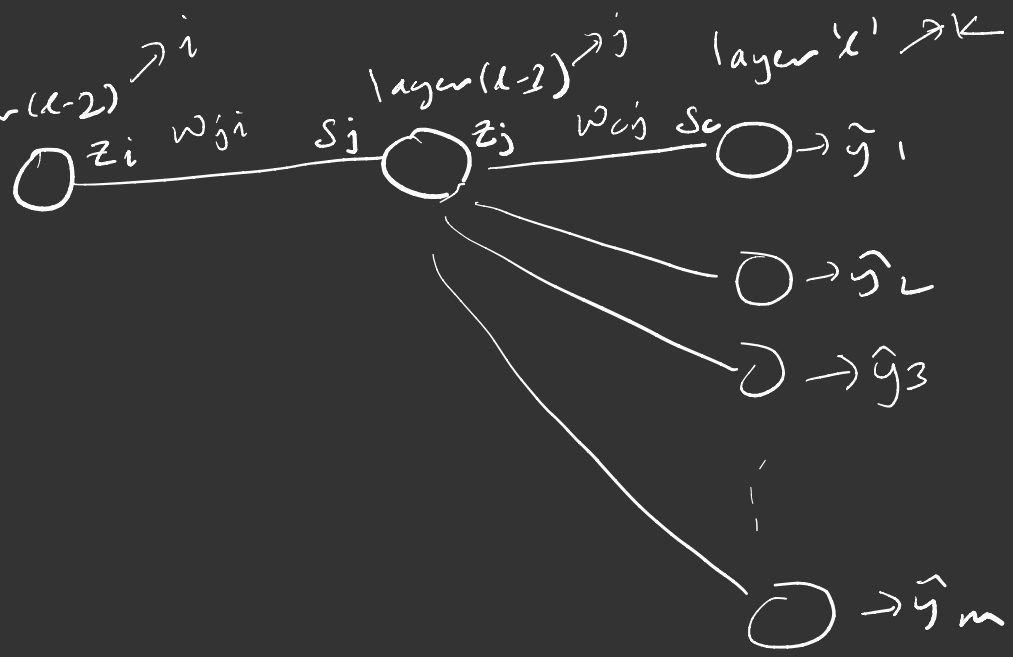
* Now, computing gradients $\frac{\partial C}{\partial w_{ji}^{(l-1)}}$ for the second last layer weights between layer '$l-2$' and '$l-1$'.

$$\frac{\partial C}{\partial w_{ji}^{(l-1)}} = \sum_{c=1}^{m} \frac{\partial C}{\partial s_c^{(l)}} \times \frac{\partial s_c^{(l)}}{\partial z_j^{(l-1)}} \times \frac{\partial z_j^{(l-1)}}{\partial s_j^{(l-1)}} \times \frac{\partial s_j^{(l-1)}}{\partial w_{ji}^{(l-2)}}$$

$$= \sum_{c=1}^{m} (\hat{y}_c - y_c) w_{cj}^{(l)} \times \left[z_j^{(l-1)} \times (1 - z_j^{(l-1)}) \right] \times z_i^{(l-2)}$$

We used $\frac{\partial C}{\partial s_c^{(l)}}$ from the previous computation and then $\frac{\partial z_j^{(l-1)}}{\partial s_j^{(l-1)}} = z_j^{(l-1)}(1 - z_j^{(l-1)})$ is the derivation of sigmoid activation function. $z_j^{(l-1)} = f(s_j^{(l-1)})$ and $s_j^{(l-1)} = \sum_{i=0}^{di} w_{ji} z_i^{(l-1)}$.

Pictorially,



So, in general, if we define the error in neuron 'i' in the layer 'l' as $\delta_i^{(l)} = \frac{\partial C}{\partial s_i^{(l)}}$, then

$$\delta_i^{(l)} = \frac{\partial C}{\partial z_i^{(l)}} \times \frac{\partial z_i^{(l)}}{\partial s_i^{(l)}} = \frac{\partial C}{\partial z_i^{(l)}} \times f'(s_i^{(l)})$$

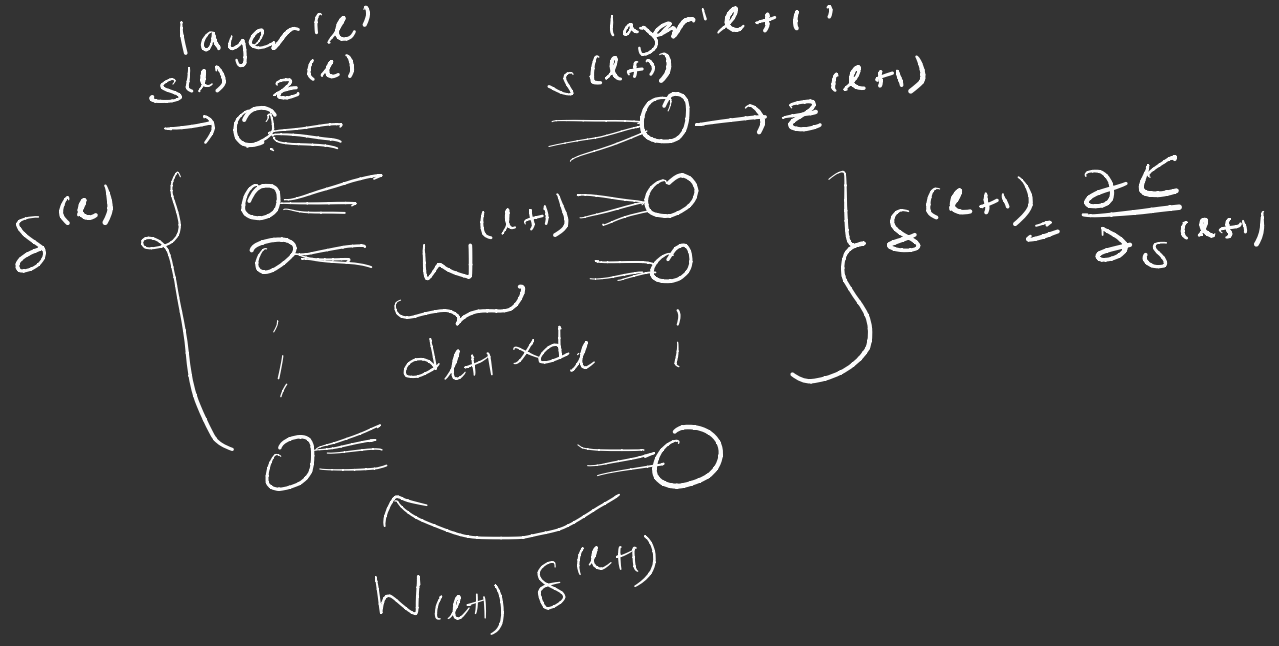
Which can be written as:

$$\delta^L = \nabla_z C \odot f'(z^{(L)})$$ where $\odot$ is the element wise product.

Thus, the error δ^L in terms of error in the next layer δ^{L+1} is given by

$$\delta^L = ((W^{L+1})^T \delta^{L+1}) \odot f'(z^{(L)})$$

The way to understand this is suppose we know the error in the $(L+1)^{st}$ layer i.e δ^{L+1}. Then applying $(W^{L+1})^T$ is intuitively moving back the error through the network, giving some measure of error δ^L at the output of layer L



Then, applying $\odot f'(s^{(l)})$ moves the error backward through the activation function in layer 'l', giving us the error $\delta^{(l)}$ in the weighted input ($s^{(l)}$) to layer 'l'

$$\delta^{(l)} = (W^{(l+1)}\ \delta^{(l+1)}) \odot f'(s^{(l)})$$

Now, $\frac{\partial C}{\partial w_{ji}^{(L)}} = \frac{\partial C}{\partial s_j^{(L)}} \times \frac{\partial s_j^{(L)}}{\partial w_{ji}^{(L)}}$

$$= \delta_j^L \times z_i^{(L-1)}$$

So, generally,

i) Input: $({}^i x, {}^i y)$ for $1 \le i \le N$ N examples.

ii) Feedforward: for each example, each layer $L = 1, 2, 3, \ldots L$

compute $s^L = w^L z^{L-1}$ and $z^{L-1} = f(s^{L-1})$

iii) Output error: $\delta^L = \nabla_z C \odot f'(s^{(L)})$

iv) Back propagate error: For each layer $L = 1, 2, 3, \ldots L$

$$\delta^{(L)} = ((w^{L+1})\ \delta^{L+1}) \odot f'(s^{(L)})$$

v) Gradient of loss function:

$$\frac{\partial C}{\partial w_{ji}^{(l)}} = z_i^{(l-1)} \times \delta_j^{(l)}$$

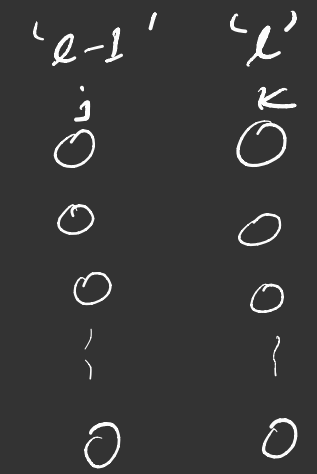
For our case, logistic function in the hidden layer and softmax on the output layer.

i) For last layer 'l'

$$\frac{\partial C}{\partial w_{kj}^{(l)}} = \delta_k^{(l)} \times z_j^{(l-1)}$$

$$= \frac{\partial C}{\partial s_k^{(l)}} \times z_j^{(l-1)}$$

$$= (\hat{y}_k - y_k) \times z_j^{(l-1)}$$



ii) for second last layer '$l-1$'

$$\frac{\partial C}{\partial w_{ji}^{(l-1)}} = \delta_j^{(l-1)} \times z_i^{(l-2)}$$

$$= \left[w_{kj}^{(l)} \delta_k^{(l)} \odot f'(s_j^{(l-1)}) \right] z_i^{(l-2)}$$

$$= \left[\sum_{k=1}^{m} w_{kj}^{(l)} (\hat{y}_k - y_k) \odot f'(s_j^{(l-1)}) \right] z_i^{(l-2)}$$

i : $l-2$ | j : $l-1$ | k : l

$$= \sum_{k=1}^{} \underbrace{(\hat{y}_k - y_k)}_{\delta_k^{(l)}} w_{kj}^{(l)} \times \underbrace{\left[z_j^{(l-1)} \left(1 - z_j^{(l-1)} \right) \right] \times z_i^{(l-2)}}_{\delta_j^{(l-1)}}$$

Confusion - for the third last layer and onward.

$$\frac{\partial L}{\partial w_{ih}^{(l-2)}} = \delta_i^{(l-2)} \times z_h^{(l-3)}$$

$$= \left[w_{ji}^{(l-1)} \delta_j^{(l-1)} \odot f'(s_i^{(l-2)}) \right] \times z_h^{(l-3)}$$

h	i	j	k
l-3	l-2	l-1	l
0	0	0	0
0	0	0	0
0	0	0	0
⋮	⋮	⋮	⋮
0	0	0	0

$$= \delta_j^{(l-1)} \times w_{ji}^{(l-1)} \times f'(s_i^{(l-2)}) \times z_h^{(l-3)}$$

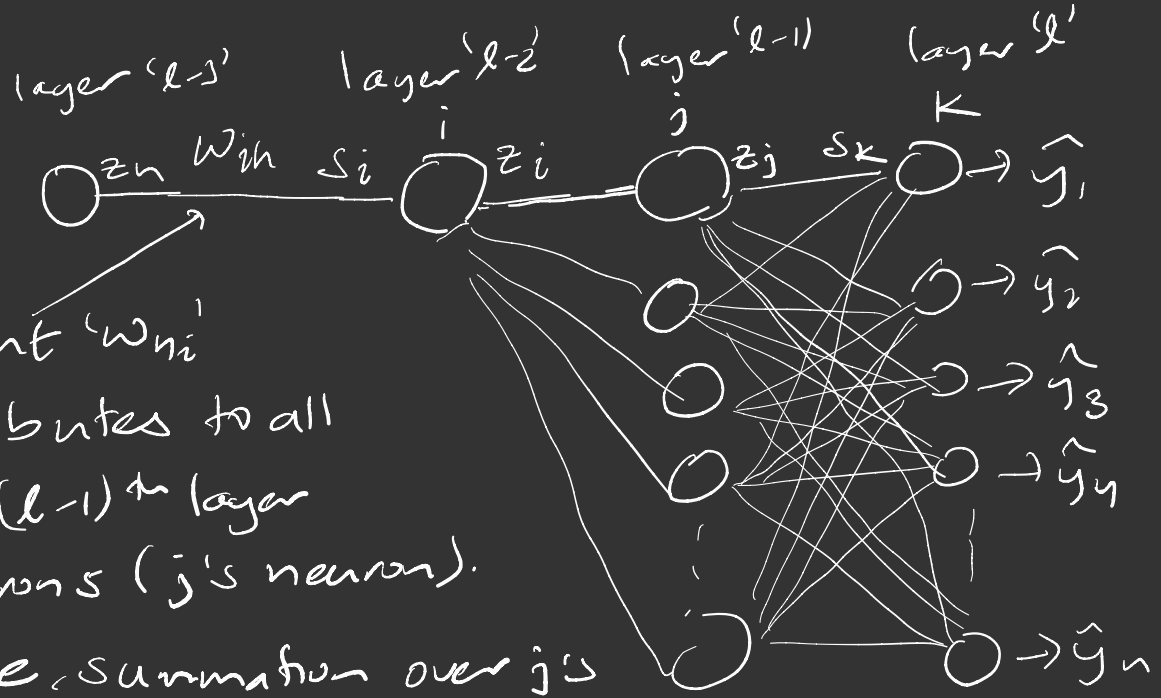
$$= \delta_j^{(l-1)} \times w_{ji}^{(l-1)} \times \left[z_i^{(l-2)} (1 - z_i^{(l-2)}) \right] \times z_h^{(l-3)}$$

$$= \left[\sum_{k=1}^{m} (\hat{y}_k - y_k) w_{kj}^{(l)} \times z_j^{(l-1)} \left[1 - z_j^{(l-1)} \right] \times w_{ji}^{(l-1)} \right.$$

$$\left. \times \left[z_i^{(l-2)} (1 - z_i^{(l-2)} \right] \times z_h^{(l-3)} \right)$$

Should have summation over (j) or not? $\sum_{j=0}^{d_j}$

We should be having because without it what would the 'j' terms in the derivative equation mean.



weight 'w_{ni}' contributes to all the $(l-1)^{th}$ layer neurons (j's neuron).

Hence, summation over j's makes sense. But need to confirm this.