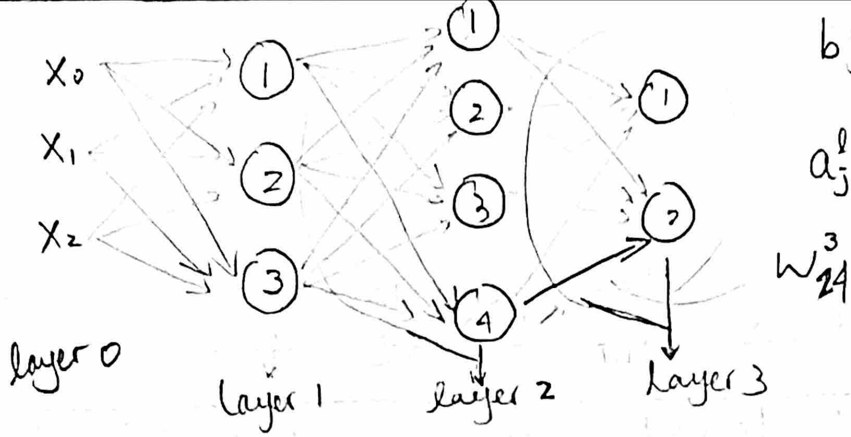


b_j^l : bias of the j^{th} neuron in the l^{th} layer

a_j^l : activation of the j^{th} neuron in the l^{th} layer

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

sum over all nodes in $l-1$

$$a^l = \sigma\left([a_1^{l-1}, a_2^{l-1}, \dots, a_k^{l-1}] \cdot \begin{bmatrix} w_{11}^l & w_{21}^l & \cdots & w_{j1}^l \\ w_{12}^l & w_{22}^l & & \vdots \\ \vdots & \vdots & \ddots & \\ w_{1k}^l & w_{2k}^l & & w_{jk}^l \end{bmatrix} + [b_1^l, b_2^l, \dots, b_j^l]\right)$$

dot product

z_j^l

$$a^l = \sigma\left([a_1^{l-1}, a_2^{l-1}, \dots, a_k^{l-1}, 1] \cdot \begin{bmatrix} w_{11}^l & w_{21}^l & \cdots & w_{j1}^l \\ w_{12}^l & w_{22}^l & & \vdots \\ \vdots & \vdots & & \vdots \\ w_{1k}^l & w_{2k}^l & & w_{jk}^l \\ b_1^l & b_2^l & & b_j^l \end{bmatrix}\right)$$

element-wise activation function

$$z_j^l = a^{l-1} \cdot w_j^l$$

$$a^l = \sigma(\underbrace{a^{l-1} \cdot w^l}) = \sigma(z^l)$$

z^l : weighted input to the neurons in layer l.

Goal of backpropagation: $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$

→ partial derivative of the cost function C w.r.t any w and b in the network

$$C = \frac{1}{2n} \sum_x \| y(x) - a^L(x) \|^2$$

n: total # of training examples.

$\sum_x$: sum over all training data points. x.

$y(x)$: Corresponding output.

L: # of layers in the network

$a^L(x)$: vector output of the network for input x

Assume: Calculating the cost for a single data point x for now.
⇒ x becomes a constant, so does $y(x)$.

Assume: Cost (C) is a function of a^L. i.e., $C = C(a^L)$

i.e., $C = \frac{1}{2} \sum_j (y_j - a_j^L)^2$

Assume: element-wise multiplication. $\begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$

Back propagation: understand how changing weights & biases in a network changes the cost function.

i.e., $\frac{\partial C}{\partial w^l_{jk}}$, $\frac{\partial C}{\partial b^l_j}$

δ^l_j : error in the j^{th} neuron in the l^{th} layer.

→ relate δ^l_j to w^l_{jk} & b^l_j,

$\frac{\partial C}{\partial w^l_{jk}}$ & $\frac{\partial C}{\partial b^l_j}$

$$\delta^l_j = \frac{\partial C}{\partial z^l_j}, \qquad \delta^l = \frac{\partial C}{\partial z^l} \quad \text{(vectorized)}$$

Error in the output layer, δ^L

$$\delta^L_j = \underline{\frac{\partial C}{\partial a^L_j} \sigma'(z^L_j)} = \frac{\partial C}{\partial a^L_j} \frac{\partial a^L_j}{\partial z^L_j} = \frac{\partial C}{\partial z^L_j}$$

↑ component form

z^L_j : compute from $xW + b$.

$\sigma'(z^L_j)$: compute by taking the derivative of $\sigma(z^L_j)$

$\frac{\partial C}{\partial a^L}$: Take derivative of $C = \frac{1}{2}\sum_j (y_j - a^L_j)^2$

↳ $\frac{\partial C}{\partial a^L_j} = (a^L_j - y_j)$

Matrix form:

$$\boxed{\delta^L = \nabla_a C \odot \sigma'(z^L)} \quad (1)$$

~~Error in terms of ther~~

Error δ^l in terms of the error in the next layer δ^{l+1} :

$$\boxed{\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)} \quad (2)$$

Idea:

$(W^{l+1})^T \delta^{l+1}$: propogate the error from $l+1$ <u>backwards</u> to l.

$\odot \, \sigma'(z^l)$: pass the error in l into the activation func.

W/ (1) & (2), δ^l can be calculated in all layers l.

First get $\underset{(1)}{\delta^L}$, then calculate $\delta^{L-1}, \delta^{L-2}, \ldots\ldots, \delta^1$ (2).

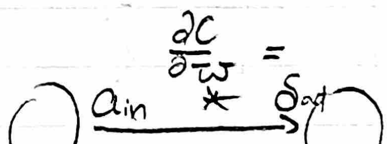
Rate of Change in Cost w.r.t any bias in the network.

$$\boxed{\frac{\partial C}{\partial b_j^l} = \delta_j^l} \ (3), \quad \text{or} \quad \frac{\partial C}{\partial b} = \delta$$

for the same neuron in the same layer

Rate of change in Cost w.r.t any weight in the network

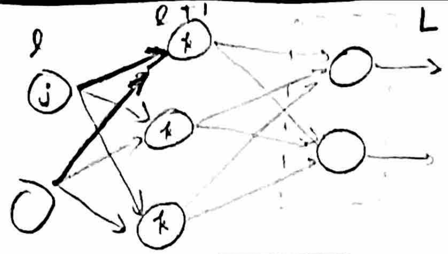
$$\boxed{\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l} \ (4), \quad \text{or}$$

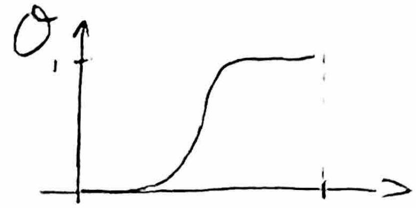


$$\frac{\partial C}{\partial w} = a_{in} \delta_{out}.$$

i.e., $\frac{\partial C}{\partial w}$ = (activation input to w) * (error output of w).

meaning, weights output from low-activation neurons learn slowly.





$$\boxed{\delta^L_j = \frac{\partial C}{\partial a^L_j}\,\sigma'(z^L_j)} \qquad (1)$$

$$\boxed{\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)} \qquad (2)$$

$$\boxed{\frac{\partial C}{\partial b^l_j} = \delta^l_j} \qquad (3)$$

$$\boxed{\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j} \qquad (4)$$

(1) (2): $\sigma'(z^L_j) \approx 0$ when $\sigma(z^L_j) \approx 1$ or $\sigma(z^L_j) \approx 0$

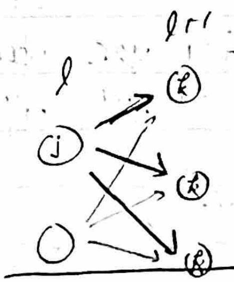
weight in the final layer will learn slowly if the output neuron has low activation $\sigma(z^L_j) \approx 0$ → saturated.

The error δ^L_j will be small when the neuron is saturated

---

(2) proof

$$\delta^l_j = \frac{\partial C}{\partial z^l_j} \quad \text{(definition)}$$



$$\delta^l_j = \sum_k \frac{\partial C}{\partial z^{l+1}_k} \frac{\partial z^{l+1}_k}{\partial z^l_j}$$

$$= \sum_k \delta^{l+1}_k \frac{\partial z^{l+1}_k}{\partial z^l_j} = \boxed{\sum_k \delta^{l+1}_k w^{l+1}_{kj} \sigma'(z^l_j)}$$

$$z^{l+1}_k = \sum_j w^{l+1}_{kj} a^l_j + b^{l+1}_k = \sum_j w^{l+1}_{kj} \sigma(z^l_j) + b^{l+1}_k$$

$$\frac{\partial z^{l+1}_k}{\partial z^l_j} = w^{l+1}_{kj} \sigma'(z^l_j)$$

after differentiating, all terms not equal to j → 0 fit_partial, back propogation:

error = layer_output[-1] − y = $a^3_0 - y \left(= \frac{\partial C}{\partial a^L_0} = \frac{\partial C}{\partial a^3_0} = \frac{\partial C}{\partial a^3}\right)$

$$\text{layer\_gradient} = \begin{bmatrix} \delta^3 = \text{error} * \text{sigmoid\_derivative}(\text{layer\_output}[-2].\text{dot}(\text{self}.W[-1])), & \text{scalar} \\ \delta^2 = \delta^3 (W^3)^T \odot \text{sigmoid\_derivative}(\text{layer\_output}[1].\text{dot}(\text{self}.W[1])), & 1\times 2 \\ \delta^1 = \delta^2 (W^2)^T \odot \text{sigmoid\_derivative}(\text{layer\_output}[0].\text{dot}(\text{self}.W[0])) & 1\times 3 \end{bmatrix}$$

$$\delta^L = \delta^3 = \delta^3_0 = \frac{\partial C}{\partial a^3_0} \cdot \sigma'(z^3_0)$$

$$= \text{error} * \text{sigmoid\_derivative}\left([a^2_0, a^2_1] \cdot \begin{bmatrix} W^3_{00} \\ W^3_{01} \end{bmatrix} \right)$$

= error * sigmoid_derivative(layer_output[-2].dot(self.W[-1])) → scalar

---

for layer in (2, 1)

- layer = 2

$$\text{current\_gradient} \left(= \delta^2 = ((\underbrace{W^3}_{[W^3_{00},\, W^3_{01}]})^T \delta^3) \odot \sigma'(z^2)\right)$$

$$= (\overset{\text{scalar}\ \delta^3}{\text{layer\_gradient}[-1]}).\text{dot}(\text{self}.W[\underset{2}{\text{layer}}].T) * \text{sigmoid\_derivative}\left([a^1_0, a^1_1, a^1_2] \cdot \begin{bmatrix} W^2_{00} & W^2_{10} \\ W^2_{01} & W^2_{11} \\ W^2_{02} & W^2_{12} \end{bmatrix} \right)$$

$$= (\text{layer\_gradient}[-1]).\text{dot}(\text{self}.W[\overset{2}{\text{layer}}].T) * \text{sigmoid\_derivative}($$
$$\text{layer\_output}[\underset{1}{\cancel{\text{layer}-1}}].\text{dot}(\text{self}.W[\underset{1}{\cancel{\text{layer}-1}}])) \rightarrow 1\times 2$$

- layer = 1

$$\text{current\_gradient} \left(= \delta^1 = ((\underset{2\times 3}{W^2})^T \delta^2) \odot \sigma'(z^1)\right)$$

$$= \underbrace{(\overset{1\times 2}{\text{layer\_gradient}[-1]}.\text{dot}(\text{self}.W[\text{layer}].T)}_{\delta^2 \quad 1\times 3} * \text{sigmoid\_derivative}\left(\underbrace{[X^0_0, X^0_1, X^0_2] \cdot \begin{bmatrix} W^1_{00} & W^1_{10} & W^1_{20} \\ W^1_{01} & W^1_{11} & W^1_{21} \\ W^1_{02} & W^1_{12} & W^1_{22} \end{bmatrix}}_{1\times 3} \right)$$

$$= (\text{layer\_gradient}[-1].\text{dot}(\text{self}.W[\overset{\text{layer}\ 1}{\ }].T) * \text{sigmoid\_derivative}($$
$$\text{layer\_output}[\underset{0}{\cancel{\text{layer}-1}}].\text{dot}(\text{self}.W[\underset{0}{\cancel{\text{layer}-1}}]))$$

Weight update:

W^1:
$$\begin{bmatrix} W^1_{00} \\ W^1_{01} \\ W^1_{02} \end{bmatrix} = \begin{bmatrix} W^1_{00} \\ W^1_{01} \\ W^1_{02} \end{bmatrix} - \alpha \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \delta^1_0 \quad \rightarrow \quad \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \begin{bmatrix} \delta^1_0 & \delta^1_1 & \delta^1_2 \end{bmatrix}$$

$$\begin{bmatrix} W^1_{10} \\ W^1_{11} \\ W^1_{12} \end{bmatrix} = \begin{bmatrix} W^1_{10} \\ W^1_{11} \\ W^1_{12} \end{bmatrix} - \alpha \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \delta^1_1$$

$$\begin{bmatrix} W^1_{20} \\ W^1_{21} \\ W^1_{22} \end{bmatrix} = \begin{bmatrix} W^1_{20} \\ W^1_{21} \\ W^1_{22} \end{bmatrix} - \alpha \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \delta^1_2$$

W^2:
$$\begin{bmatrix} W^2_{00} \\ W^2_{01} \\ W^2_{02} \end{bmatrix} = \begin{bmatrix} W^2_{00} \\ W^2_{01} \\ W^2_{02} \end{bmatrix} - \alpha \begin{bmatrix} a^1_0 \\ a^1_1 \\ a^1_2 \end{bmatrix} \delta^2_0 \qquad \begin{bmatrix} a^1_0 \\ a^1_1 \\ a^1_2 \end{bmatrix} \cdot \begin{bmatrix} \delta^2_0 & \delta^2_1 \end{bmatrix}$$

$$\begin{bmatrix} W^2_{10} \\ W^2_{11} \\ W^2_{12} \end{bmatrix} = \begin{bmatrix} W^2_{10} \\ W^2_{11} \\ W^2_{12} \end{bmatrix} - \alpha \begin{bmatrix} a^1_0 \\ a^1_1 \\ a^1_2 \end{bmatrix} \delta^2_1$$

W^3:
$$\begin{bmatrix} W^3_{00} \\ W^3_{01} \end{bmatrix} = \begin{bmatrix} W^3_{00} \\ W^3_{01} \end{bmatrix} - \alpha \begin{bmatrix} a^2_0 \\ a^2_1 \end{bmatrix} \delta^3_0 \qquad \begin{bmatrix} a^2_0 \\ a^2_1 \end{bmatrix} \begin{bmatrix} \delta^3_0 \end{bmatrix}$$

```
W^3 = W^3 - α * layeroutput[2].T.dot(layer_gradient[2])
W^2 = W^2 - α * layeroutput[1].T.dot(layer_gradient[1])
W^1 = W^1 - α * layeroutput[0].T.doT(layer_gradient[0])
```