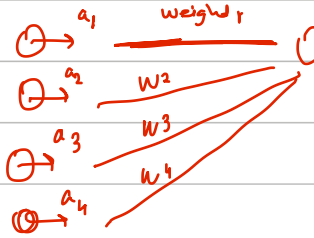
- MNIST → 784 neurons ← 28×28 pixels → 28 ⇒ 784

- final layer classification → 10 neurons for 10 digit classification

- hidden layers can be quite variable
  - ↳ a combination of subcomponents
    - ↳ comprised of subcomponents

pixels → edges → patterns → digits

activation; an output of neuron circle / node



$h = w_1 a_1 + w_2 a_2 + w_3 a_3 \dots w_n a_n$ for one next neuron

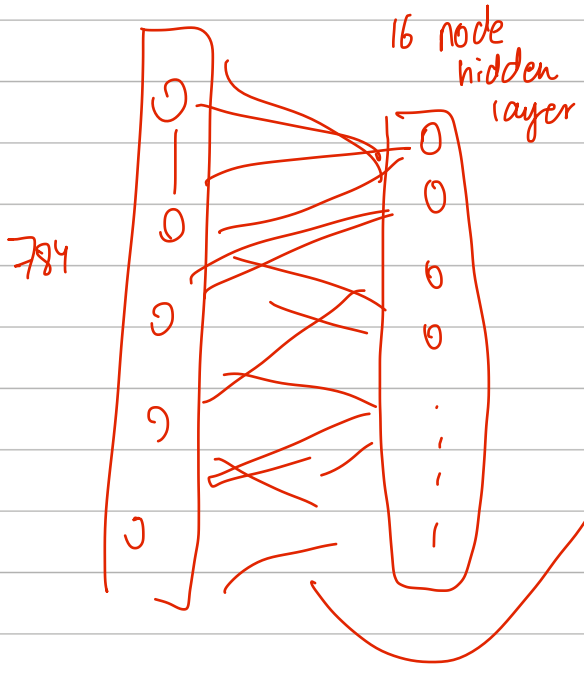
put h through an activation function to squeeze all values between 0 and 1

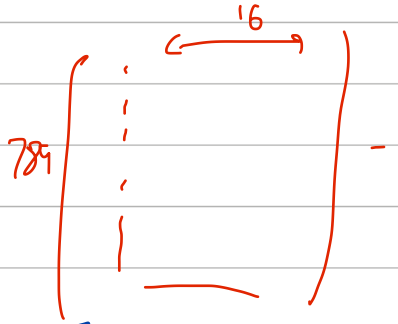
Sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ $\quad 0 \le \sigma(x) \le 1$

bias: only activate when weighted sum > threshold

$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 + \dots w_n a_n - \text{bias})$

↑ Sigmoid





$h = X \; W$

$(784, 1) \cdot (16, 1)^T$

$[784, 16]$

$\sigma([784, 16])$

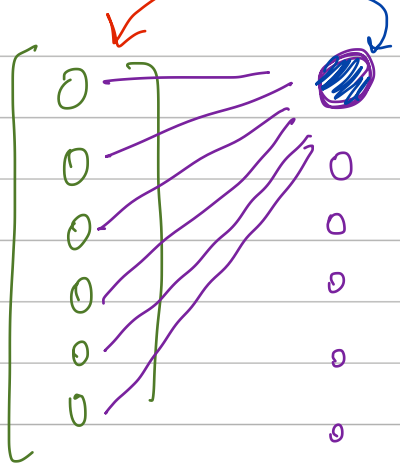
$\Rightarrow$ $[16, 1]$ - $[16, 1]$

activations . biases

Better:

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} = \begin{bmatrix} \\ \\ \\ \end{bmatrix}$$

weights / all the connections between $a_0 \dots a_n$ and one node in hidden layer.



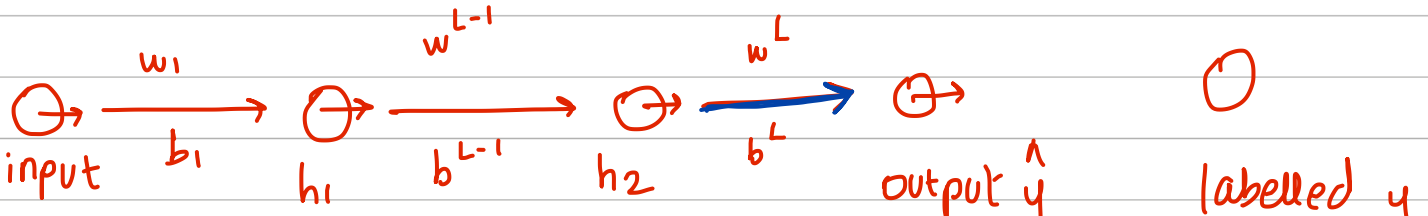
Backprop: for a single final neuron, fix weights

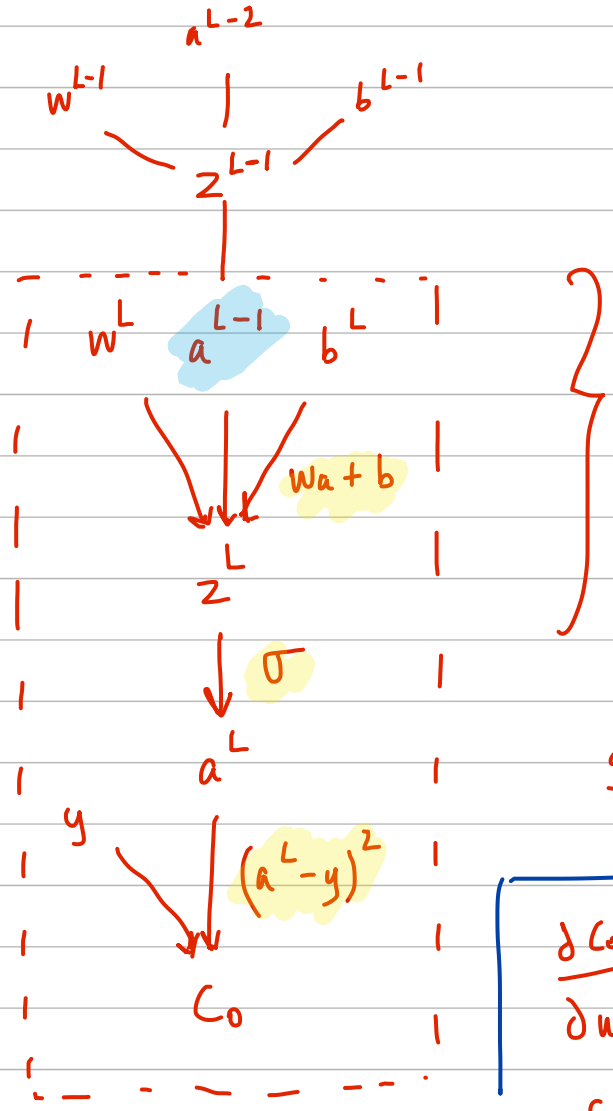
Gradient descent: try to minimize cost of all parameters (all weights across all layers)

$\downarrow C(w)$

Calculus :

Single node layers + single training example for cost function





a^{L-1} a^L

these aren't exponents, just indices

Cost

$$C_0 = (a^L - y)^2$$

$$z^L = w^L a^{L-1} + b^L$$

$$a^L = \sigma(z^L)$$

Question to answer

$\frac{\partial C_0}{\partial w^L}$? How sensitive is the cost to change in w^L

$$\frac{\partial C_0}{\partial w^L} = \frac{\partial z^L}{\partial w^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial C_0}{\partial a^L}$$

③ ② ①

follow the chain backwards

① $C_0 = (a^L - y)^2$

$$\frac{\partial C_0}{\partial a^L} = 2(a^L - y)(1)$$

$$= 2(a^L - y)$$

② $a^L = \sigma(z^L)$ deriv. of sigmoid / relu etc

$$\frac{\partial a^L}{\partial z^L} = \sigma'(z^L)$$

③ not exponents, just indices

$$z^L = w^L a^{L-1} + b^L$$

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

the previous activation determines how strongly the weight nudges the layer

Same neural net structure but all training examples

$$\frac{\partial C}{\partial w^L} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial C_k}{\partial w^L}$$

k == the k^{th} training example

"stochastic" split examples into batches

But we also want to minimize cost for all weights, not just last layer

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^1} \\ \frac{\partial C}{\partial b^1} \\ \vdots \\ \frac{\partial C}{\partial w^L} \\ \frac{\partial C}{\partial b^L} \end{bmatrix}$$

gradient of the cost function

Gradient descent:

$$\vec{w} = \begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix} - \begin{bmatrix} \\ \vdots \\ \\ \end{bmatrix}$$

$\vec{w}$: new weights & biases; first vector: all weights & biases; second vector: gradient ∇C; all the "nudges"

Similarly: for a single example

$$\frac{\partial C_0}{\partial b^L} = \frac{\partial z^L}{\partial b^L} \cdot \frac{\partial a^L}{\partial z^L} \cdot \frac{\partial C_0}{\partial a^L}$$

stays same

work backwards

$$= (1) \cdot \sigma'(z^L) \cdot 2(a^L - y)$$

(1): the changed part

Similarly for a single example

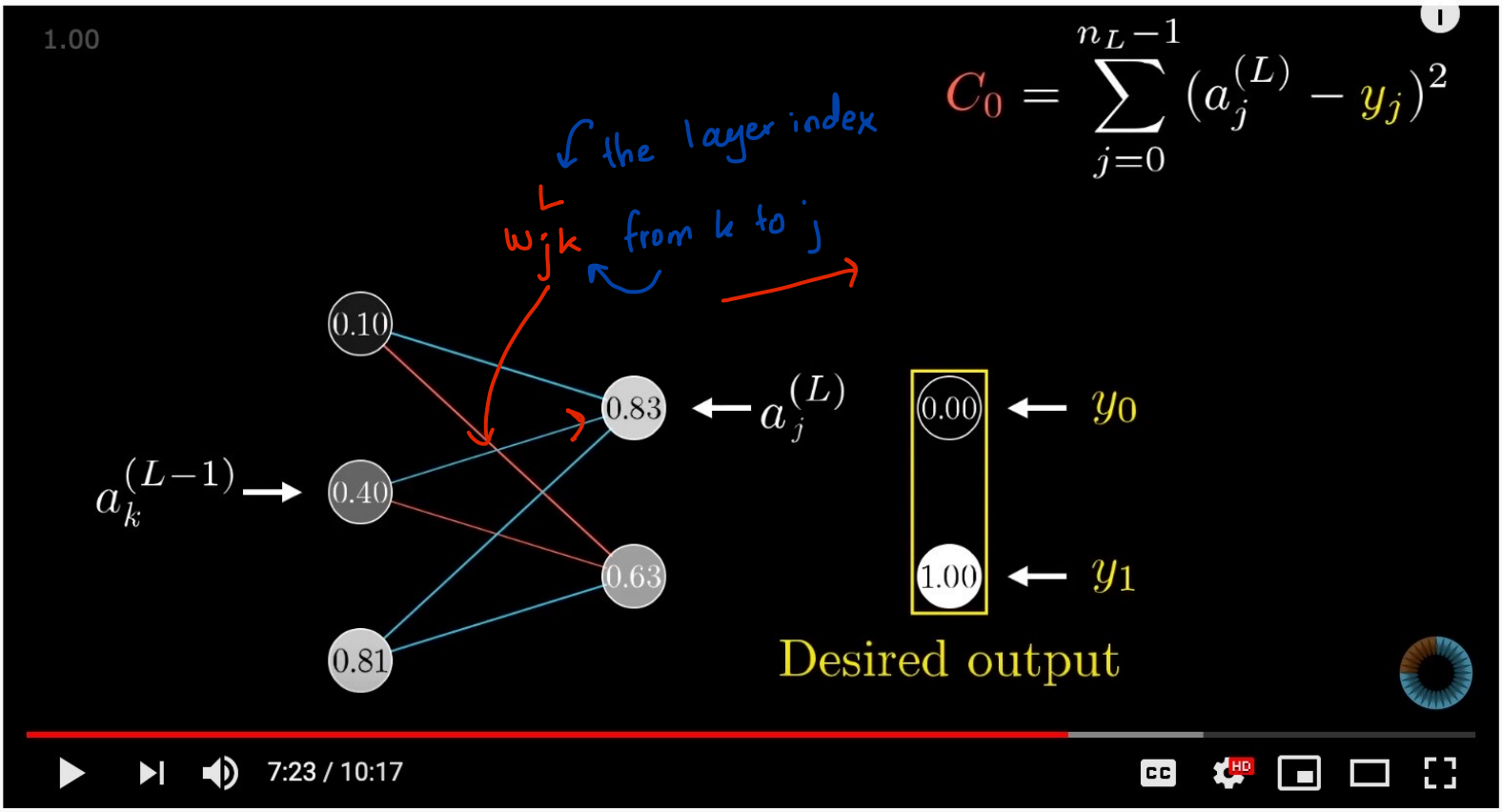
$$\frac{\partial C_0}{\partial a^{L-1}} = \frac{\partial z^L}{\partial a^{L-1}} \cdot \boxed{\frac{\partial a^L}{\partial z^L} \cdot \frac{\partial C_0}{\partial a^L}}$$

stays same

$$= w^L \cdot \sigma'(z^L) \cdot 2(a^L - y)$$

w^L: the changed part

keep going until you hit w^1 (the backprop part)

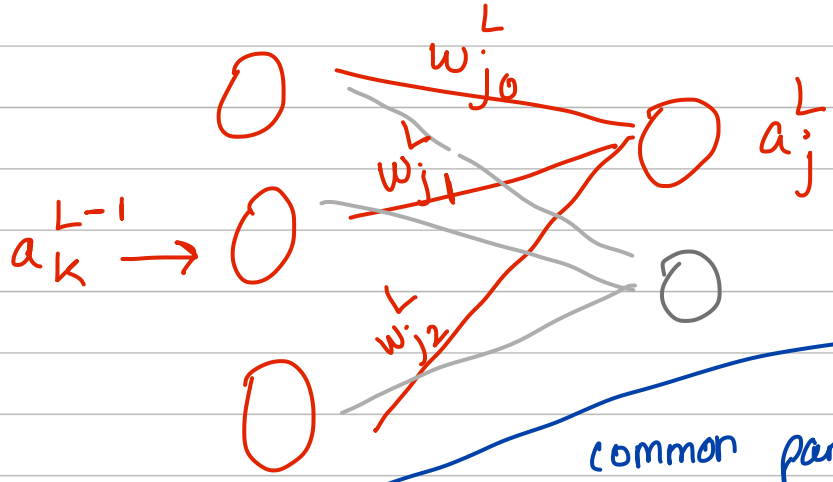


Multilayer network with one training example

$$C_0 = \sum_{j=0}^{n_{L-1}} \left(a_j^L - y_j\right)^2$$

for all j in the output layer

$$z_j^L = w_{j0}^L a_0^{L-1} + w_{j1}^L a_1^{L-1} + w_{j2}^L a_2^{L-1} + b_j^L$$



$$\frac{\partial C_0}{\partial w_{jk}^L} = \frac{\partial C_0}{\partial a_j^L} \times \frac{\partial a_j^L}{\partial z_j^L} \times \frac{\partial z_j^L}{\partial w_{jk}^L}$$

common part

variable part

change in cost for training example 0 / change in weight connecting node k in L-1 layer to node j in L layer = change in cost for training e.g 0 / change in activation of j node in L layer × change in activation of j node in L layer / change in linear combination (i.e pre-squishing with σ) of node j in L layer × (you get it) .... / the weight ....

very similar setup for

$$\frac{\partial C_0}{\partial b_j^L} \quad \text{and for} \quad \frac{\partial C_0}{\partial a_k^{L-1}}$$

$$\frac{\partial C_0}{\partial a_k^{L-1}} = \sum_{j=0}^{n_{L-1}} \frac{\partial z_j^L}{\partial a_k^{L-1}} \frac{\partial a_j^L}{\partial z_j^L} \cdot \frac{\partial C_0}{\partial a_j^L}$$

changed part

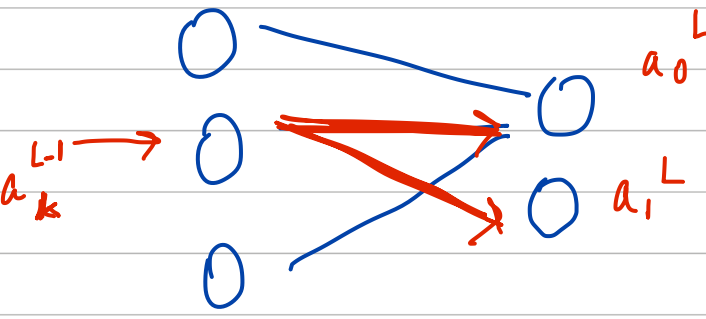
same as before

$$\frac{\partial C_0}{\partial b_j^L} = \frac{\partial z_j^L}{\partial b_j^L} \cdot \frac{\partial a_j^L}{\partial z_j^L} \cdot \frac{\partial C_0}{\partial a_j^L}$$

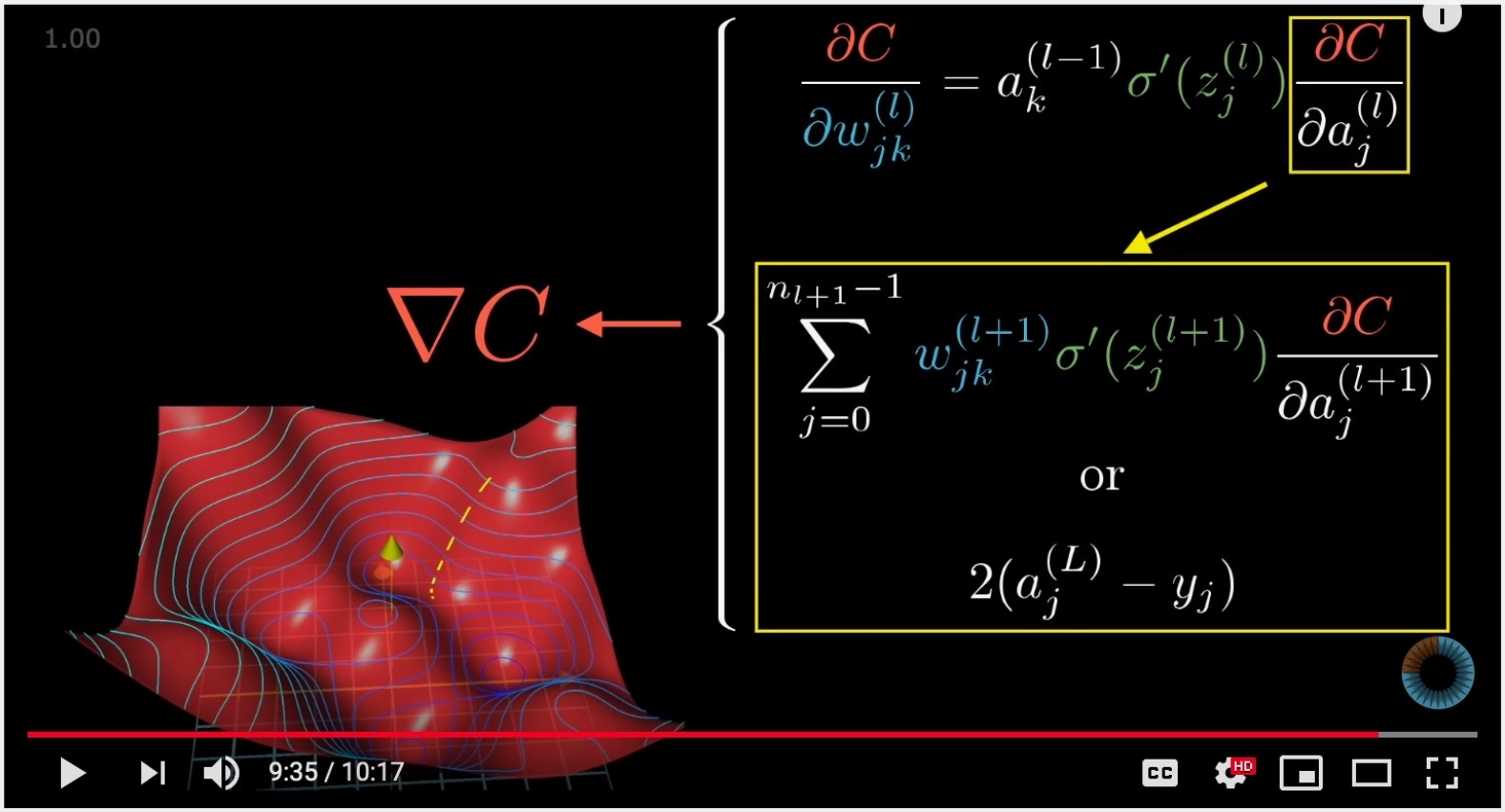
changed part

same as before

why the sum?



the previous activation a_k^{L-1} not only influences a_0^L but also every other node in layer L like a_1^L


1.00
$\frac{\partial C}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \sigma'(z_j^{(l)}) \frac{\partial C}{\partial a_j^{(l)}}$
$\nabla C \leftarrow$
$\sum_{j=0}^{n_{l+1}-1} w_{jk}^{(l+1)} \sigma'(z_j^{(l+1)}) \frac{\partial C}{\partial a_j^{(l+1)}}$
or
$2(a_j^{(L)} - y_j)$
9:35 / 10:17