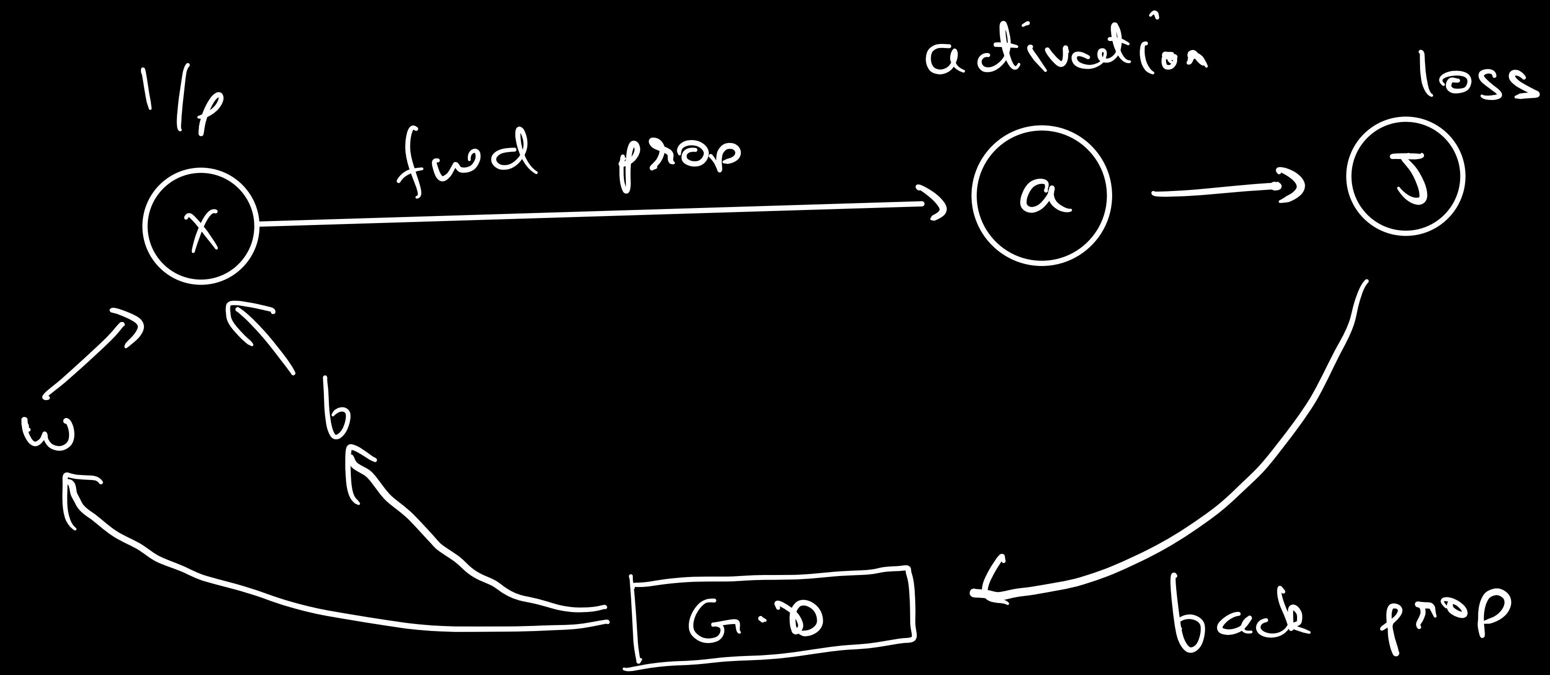
Backpropogation

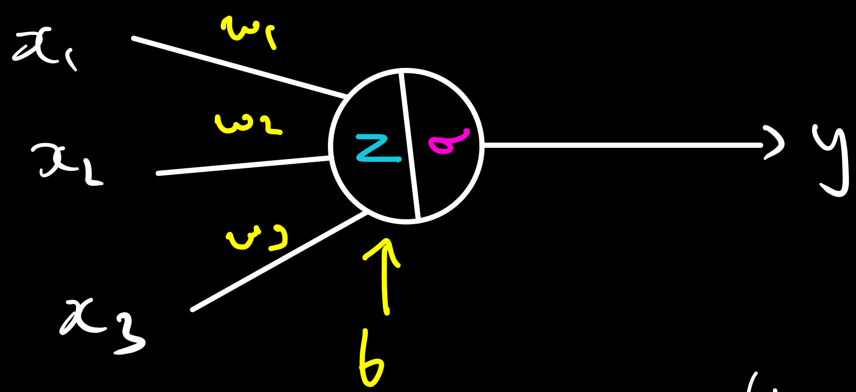
&

Multi Layer Neural Networks

Training a Neuron



Let's consider a single Neuron



L.R.U

Shapes

$w = (1, d)$ $\quad b = (1, 1)$

$x = (d, 1)$ $\quad a = (1, 1)$

$L =$ scalar

Fwd

$$Z = \vec{w} \cdot \vec{x} + b$$

$$\hat{y} = a = \sigma(z)$$

$$J = -\left(y \log(a) + (1-y) \log(1-a)\right)$$

G.D

$$w' = w - \alpha \frac{\partial J}{\partial w}, \quad b' = b - \alpha \frac{\partial J}{\partial b}$$

Bwd

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w}$$

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial b}$$

1) $$\frac{\partial J}{\partial a} = -\left(y \frac{\partial \log(a)}{\partial a} + (1-y) \frac{\partial \log(1-a)}{\partial a}\right)$$

$$= \boxed{-\left(y \cdot \frac{1}{a} + (1-y) \frac{1}{a-1}\right)}$$

2) $$\frac{\partial a}{\partial z} = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right)$$

$$\frac{\partial}{\partial x}\left[(1+e^{-z})^{-1}\right]$$

$$= -\frac{1}{(1+e^{-z})^{2}} \cdot (-e^{-z})$$

$$= \frac{e^{-z}}{(1+e^{-z})^{2}}$$

$$= \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$= \sigma(z) \cdot \left[\frac{e^{-z}+1-1}{1+e^{-z}}\right]$$

$$= \sigma(z)\left[1 - \sigma(z)\right]$$

$$= \boxed{a(1-a)}$$

3) $\dfrac{\partial z}{\partial w} = x$

4) $\dfrac{\partial z}{\partial b} = 1$

$$\gg \frac{\partial J}{\partial w} = -\left(y \cdot \frac{1}{a} + (1-y)\frac{1}{a-1}\right)(a)(1-a)\cdot x$$

-1

$$= -\left(y \cdot \frac{1}{\cancel{a}} \cdot \cancel{a}(1-a) + (1-y)\frac{1}{\cancel{a-1}} a \cdot (\cancel{1-a}) \right) x$$

$$= -\left[y(1-a) - (1-y)a \right]$$

$$= -\left[y - a\cancel{y} - a + a\cancel{y} \right]$$

$$= \boxed{(\vec{a} - y) \cdot \vec{x}}$$

OR $(a-y) \cdot x^T$ [because w shape is transpose of x shape]

$$\frac{\partial J}{\partial y} = (a - y)$$

Eg: $a = \hat{y} = 0.7$

$y = 1$

$\therefore \frac{\Delta w}{\Delta x} = 0.7 - 1 = -0.2$

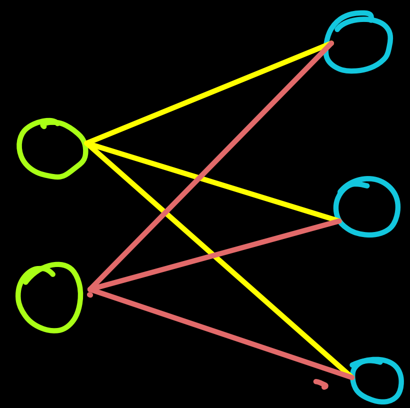
$\therefore$ G.D

$$w' = w - \alpha (a - y) x^T$$

$$b = b - \alpha (a - y)$$

This is for a single point $\vec{x_i}$. When we have entire data X, then we will use $\rightarrow$ TODO

For a multiclass Network



$$\underset{\hat{y}}{\begin{bmatrix}0.7\\0.2\\0.1\end{bmatrix}} \sim \underset{y}{\begin{bmatrix}1\\0\\0\end{bmatrix}}$$

Backprop through Softmax

Long mathematical proof!! → Post Read

Hack →

Sigmoid is a special case of softmax

$$sm(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

[softmax]

for 2 classes

$$sm(z_A) = \frac{e^{z_A}}{e^{z_A} + e^{z_B}}$$

$$= \frac{1}{1 + e^{(z_B - z_A)}}$$

$$= \frac{1}{1 + e^{-(z_A - z_B)}}$$

$$\sim \frac{1}{1+e^{-z'}} \quad \cdots\cdots \text{sigmoid}$$

This is loosely like a sigmoid, hence the derivative also comes out to be similar even for multiple classes:

$$\therefore \frac{\partial J}{\partial a} \cdot \frac{\partial a}{\partial z} = (a-y)$$

$$\Rightarrow \frac{\partial J}{\partial w} = (a-y) \cdot x^T \ , \quad \frac{\partial J}{\partial b} = (a-y)$$

Note that here

$$a = \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_1 \\ x_3 \end{bmatrix}$$

$$\Delta w = (a - y) \cdot x^T$$

$$= K \times 1, \; 1 \times d$$

$$= K \times d$$

Es: 3×2

Please note that shapes depend on initial setting and conventions.

When you have many points,

$$w' = w - \alpha \frac{1}{n} \sum_{i=1}^{n} (a_i - y_i)\, x_i^T$$

$$b = b - \alpha \frac{1}{n} \sum_{i=1}^{n} (a_i - y_i)$$

These calc can also be done in matrix form

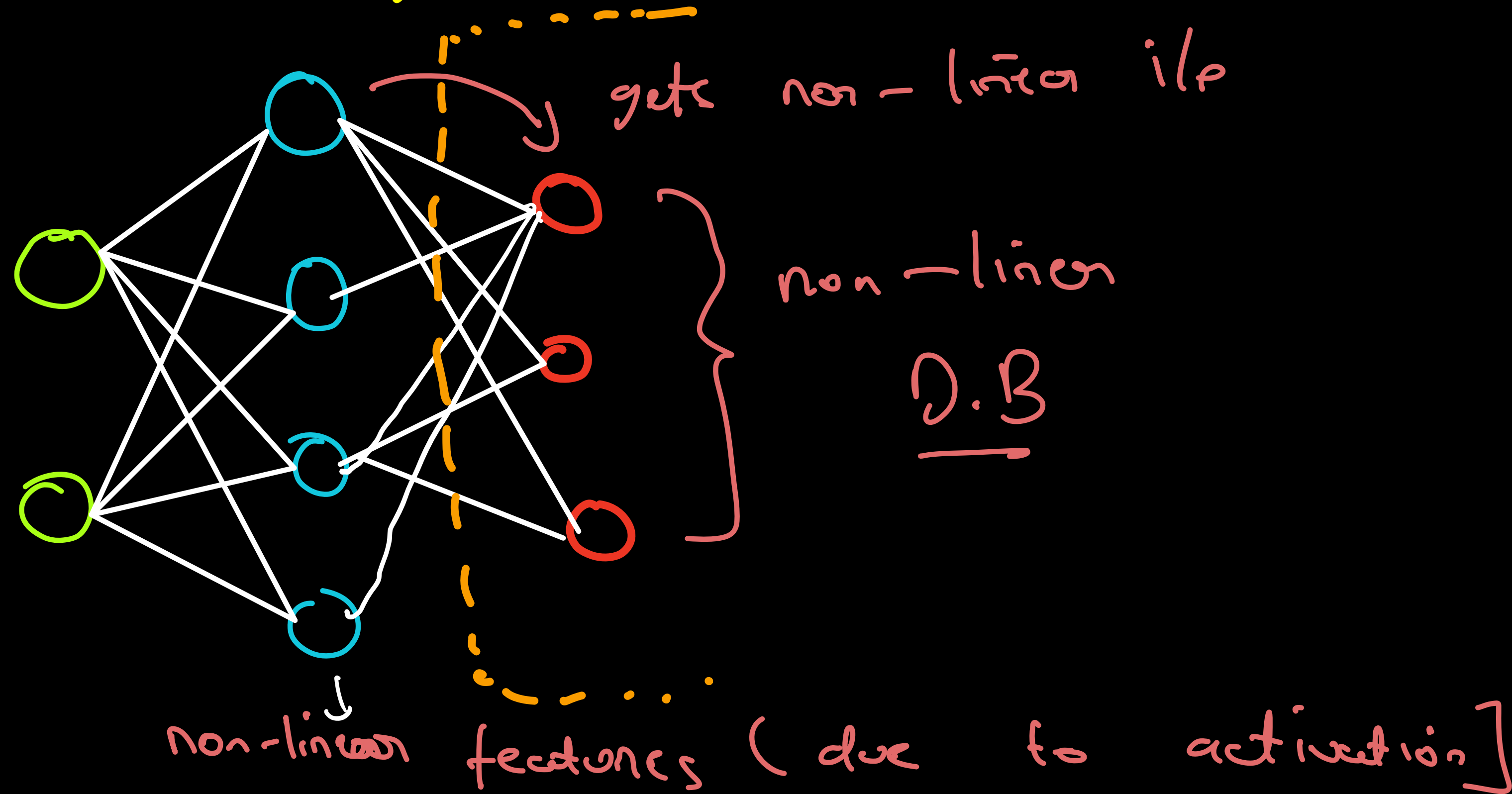
eg: → $w - \alpha \underbrace{(A - Y) \cdot X^T}$ → take mean in code

→ code

Multi-layer networks

We got a linear decision boundary. How to get non-linear?

→ Add another layer



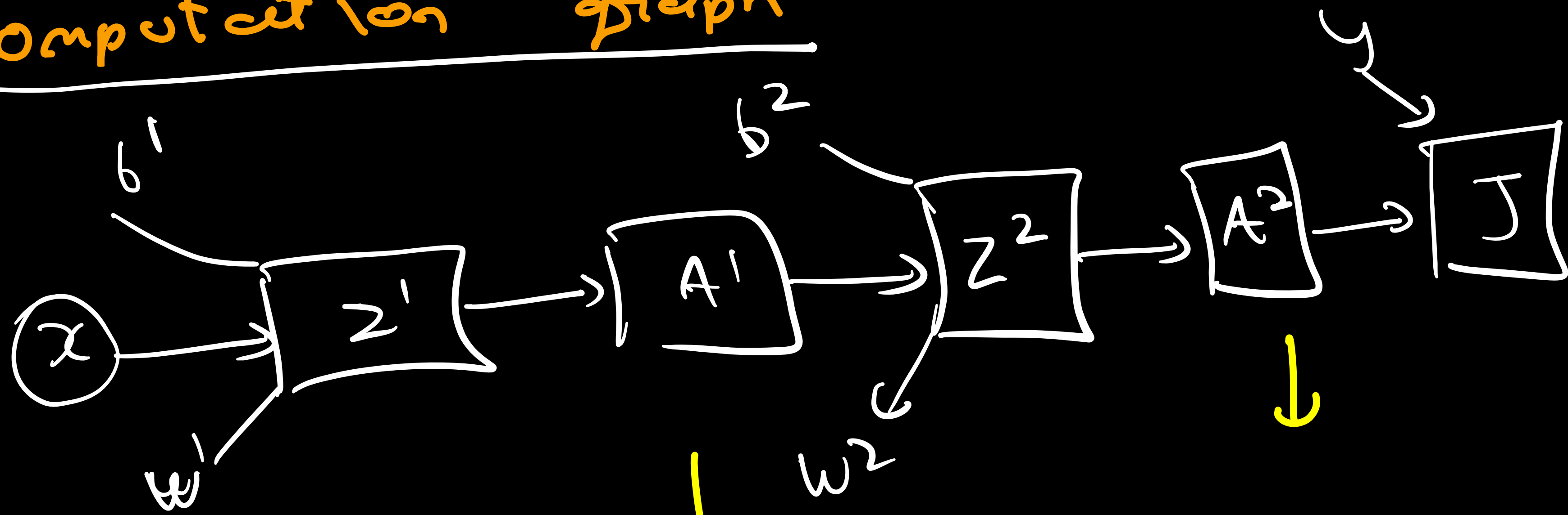
Do not confuse

$$z = wx + b$$
$$(k \times d)(d \times n) + (k \times n)$$
$$= \underline{\underline{k \times n}}$$

$$z = xw + b$$
$$(n \times d)(d \times k) + (1 \times k)$$
$$\underline{\underline{n \times k}}$$

Both are correct, depends on initial shape.

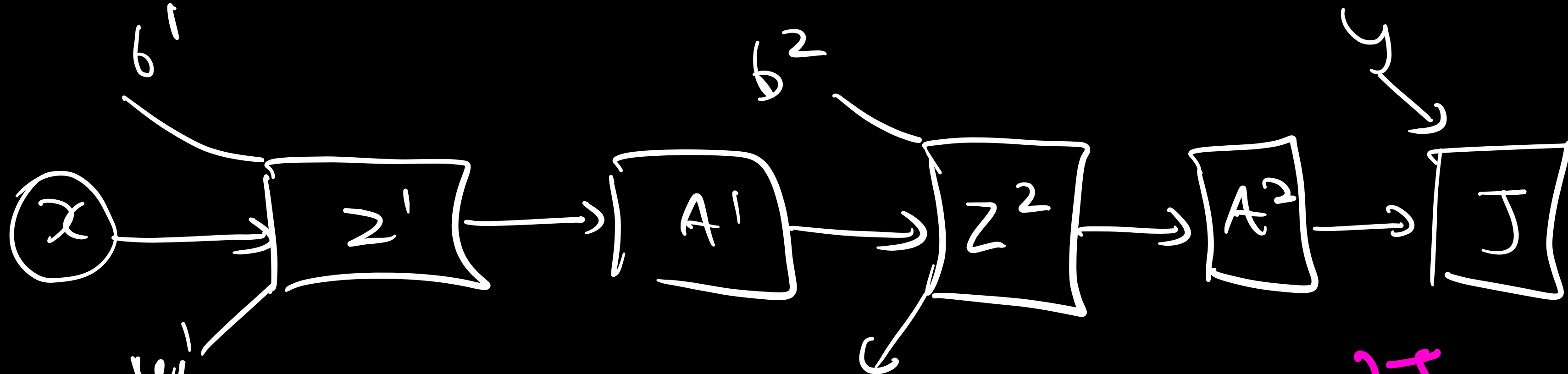
Computation graph



Fwd Prop → $A^1 = a(XW^1 + b^1)$

$A^2 = a(\underbrace{A^1}_{\text{non linear}}W^2 + b^2)$

Back prop



$$\frac{\partial A^1}{\partial Z^1} \leftarrow \frac{\partial Z^2}{\partial A^1} \leftarrow \frac{\partial A^2}{\partial Z^2} \leftarrow \frac{\partial J}{\partial A^2}$$

$$\frac{\partial z^1}{\partial W^1} \qquad \frac{\partial z^1}{\partial b} \qquad \frac{\partial Z^2}{\partial w^2} \qquad \frac{\partial Z^2}{\partial b^2}$$

$$\therefore \frac{\partial J}{\partial W^1} = \frac{\partial J}{\partial A^2} \cdot \frac{\partial A^2}{\partial Z^2} \cdot \frac{\partial Z^2}{\partial A^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial Z^1}{\partial W^1}$$