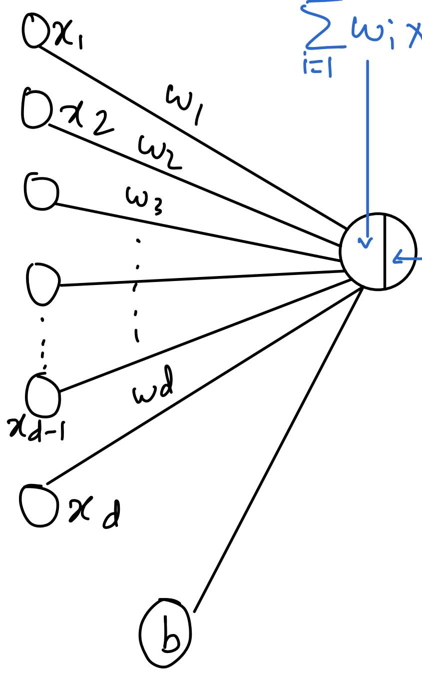
Deep Learning

Neural Networks learn features themselves.

Kernel methods require intuition on the part of the designer

$x \in \mathbb{R}^d$

$\sum_{i=1}^{d} w_i x_i + b$ i.e. $z = w^T x + b$



$a = g(z)$

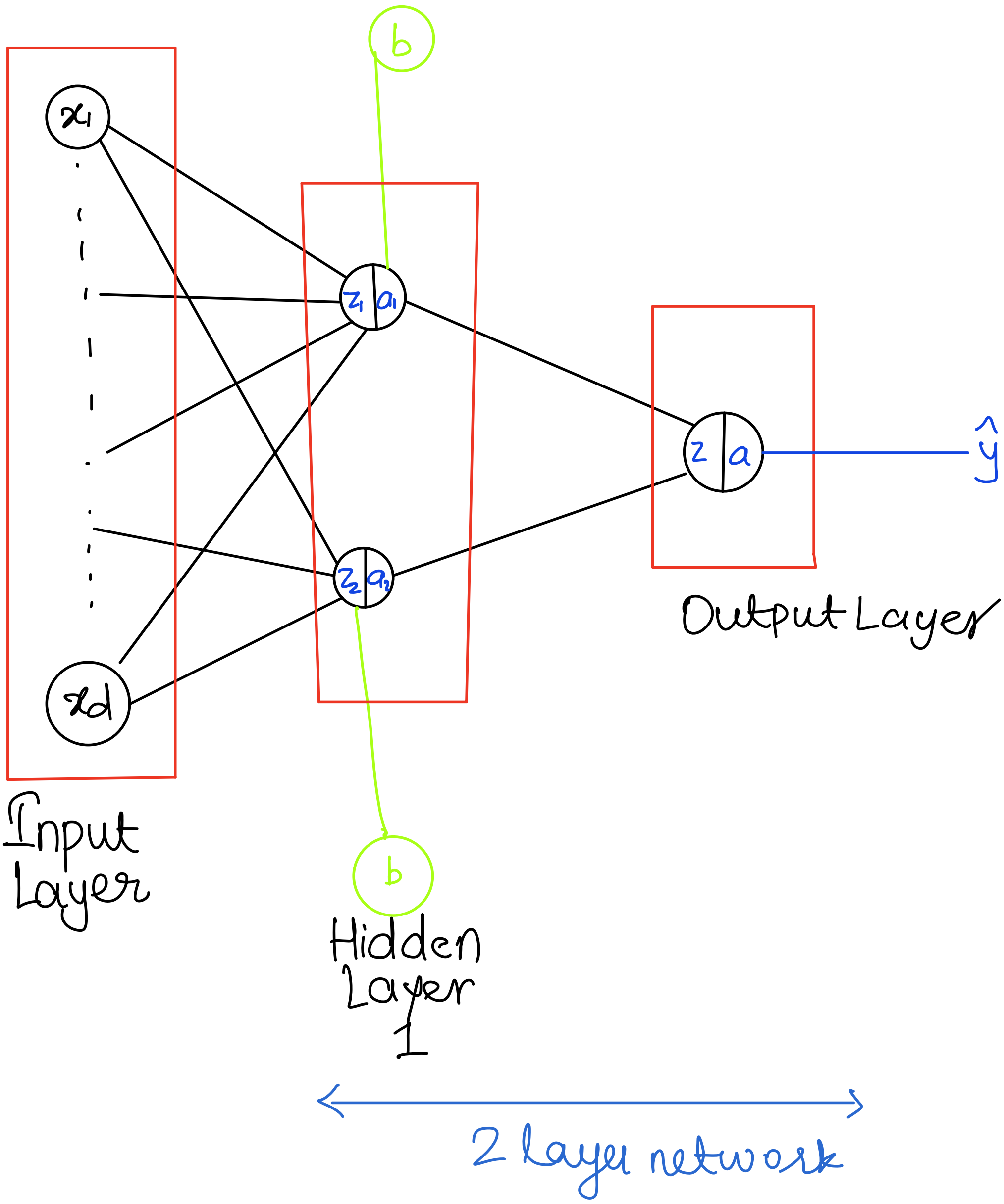
In logistic,

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

θ replaced by w

$\hat{y} = a$

$$l(y,\hat{y}) = y\log\hat{y} + (1-y)\log(1-\hat{y})$$



We don't count input layer

$w_{ij}^{[l]}$ connection

$b_i^{[l]} \rightarrow$ bias

$z_i^{[l]} \rightarrow w \cdot (\ ) + b$

$a_i^{[l]} \rightarrow g(z)$

$W^{[l]}$ is the weight matrix

$$\begin{array}{c c} \text{\# neurons in } l^{th} \text{ layer} & \begin{bmatrix} & & \\ & W^{[l]} & \\ & & \end{bmatrix} \end{array}$$

neurons in l-th layer.

$$x - a^{[0]}$$

$$z_1^{[1]} = \sum_j w_{1j}^{[1]} a_j^{[0]} + b_1$$

no. of colums → no. of neurons in input layer

$$z_2^{[1]} = \sum_j w_{2j}^{[1]} a_j^{[0]} + b_2 = w_2^{[1]T} a^{[0]} + b_2$$

$$a_1^{[1]} = g(z_1^{[1]})$$

$$z_1^{[2]} = \sum_j w_{1j}^{[2]} a_j^{[1]} + b_1 = w_1^{[2]T} a^{[1]} + b_1$$

$$z^{[1]} = w^{[1]} a^{[0]} + b^{[1]} \qquad a^{[1]} = g(z^{[1]})$$

$$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$$

$$a^{[2]} = g(z^{[2]})$$

$$z^{[3]} = w^{[3]} a^{[2]} + b^{[3]}$$

$$a^{[3]} = g(z^{[3]})$$

(Q) What if $g(z) = z$?

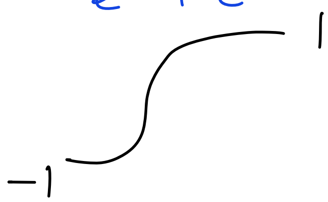
$$\rightarrow a^{[3]} = z^{[3]} = w^{[3]} a^{[2]} = w^{[3]} z^{[2]}$$
$$= w^{[3]} w^{[2]} z^{[1]}$$
$$= w^{[3]} w^{[2]} w^{[1]} x$$
$$= \tilde{w} x$$

g(z) can be non-linear function only

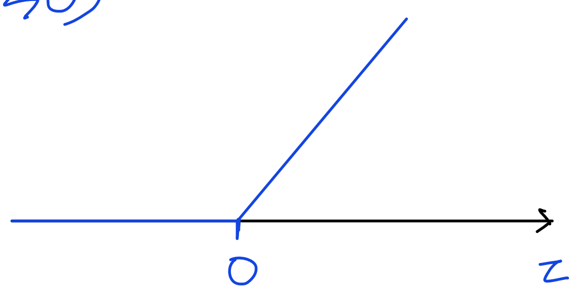
We need it to be monotonic.

$$g = \frac{1}{1+e^{-z}} \qquad g = \tanh = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$

gradient is low
almost 0. learning stops



$g = \text{relu} = \max(z, 0)$



$$a^{[0]} = x^{(i)}$$
$$z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$$
$$a^{[1]} = g(z^{[1]})$$
$$z^{[2]} = w^{[2]} a^{[1]} + b^{[2]}$$
$$a^{[2]} = g(z^{[2]})$$
$$\vdots$$
$$z^{[L]} = w^{[L]} a^{[L-1]} + b^{[L]}$$
$$a^{[L]} = g(z^{[L]}) = \hat{y}^{(i)}$$
$$\mathcal{L}(y^{(i)}, \hat{y}^{(i)}) = -\left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\right]$$

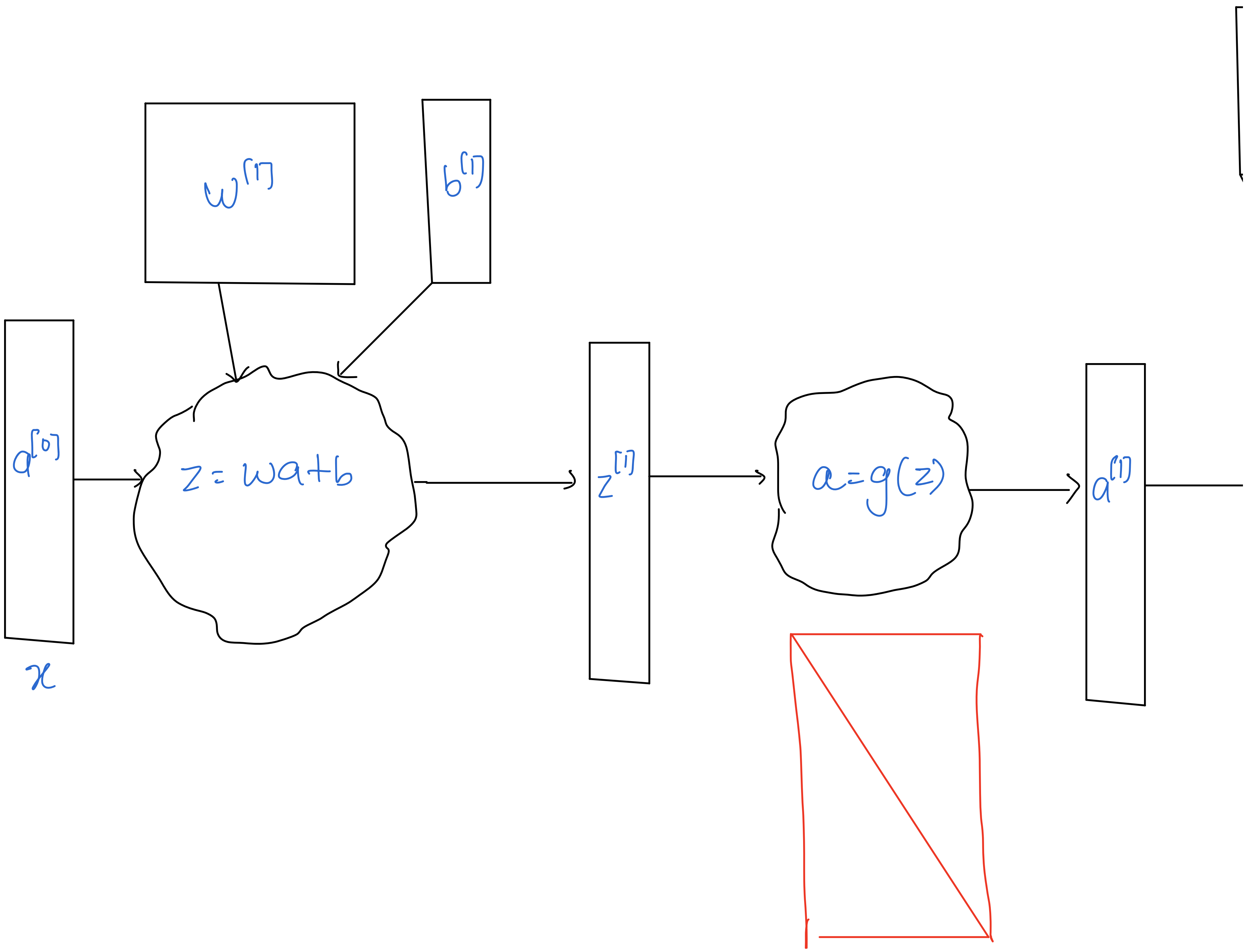
For l in $1, 2 \ldots L$

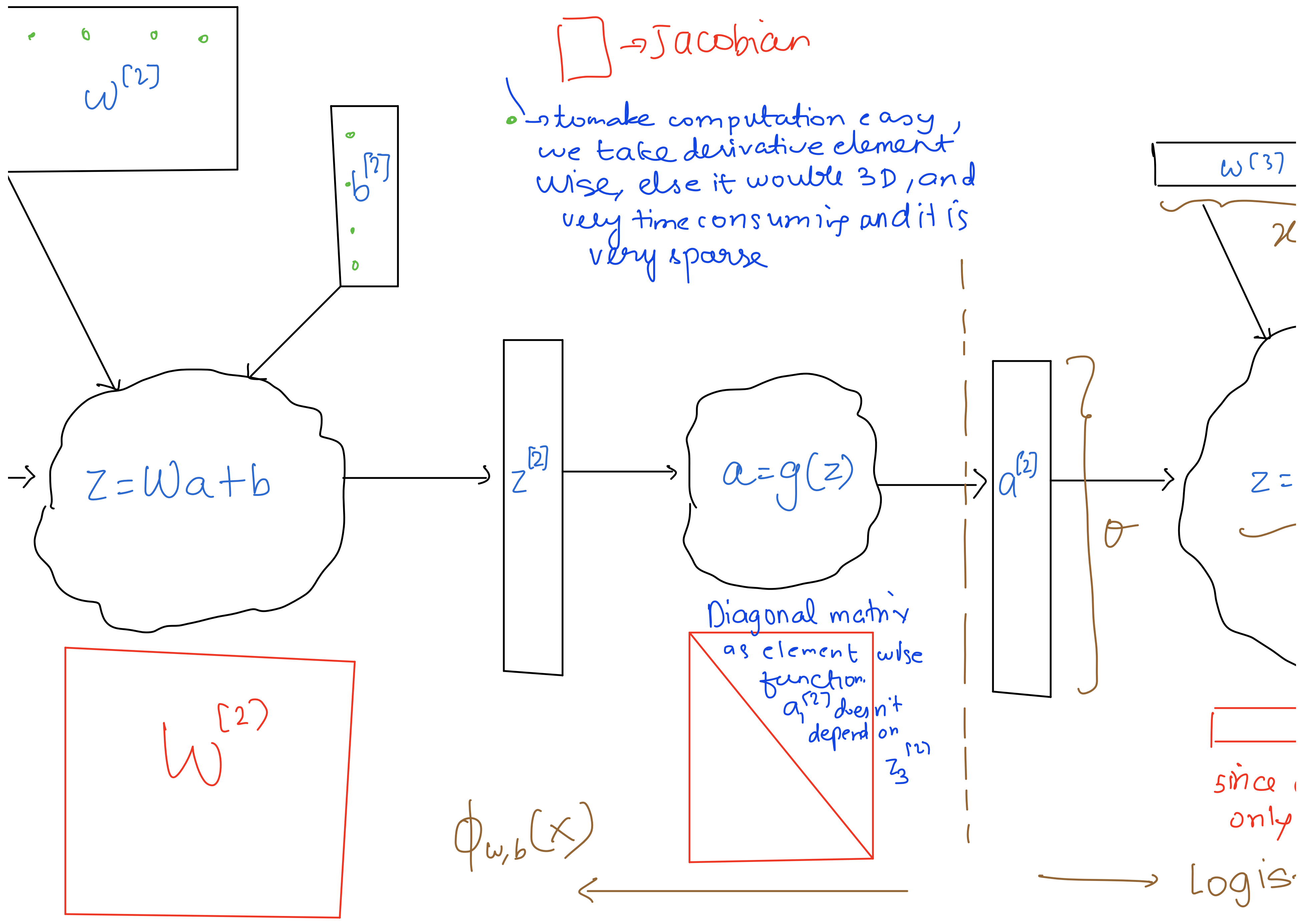
$$\left.\begin{aligned} W^{[l]} &:= W^{[l]} - \alpha \frac{\partial L}{\partial W^{[l]}} \\ b^{[l]} &:= b^{[l]} - \alpha \frac{\partial L}{\partial b^{[l]}} \end{aligned}\right\} \text{Stochastic Gradient}$$

Initialized randomly.
If initialized with all neurons will be same.
By random initialization, we ensure we reach diff. local minima as loss is not convex

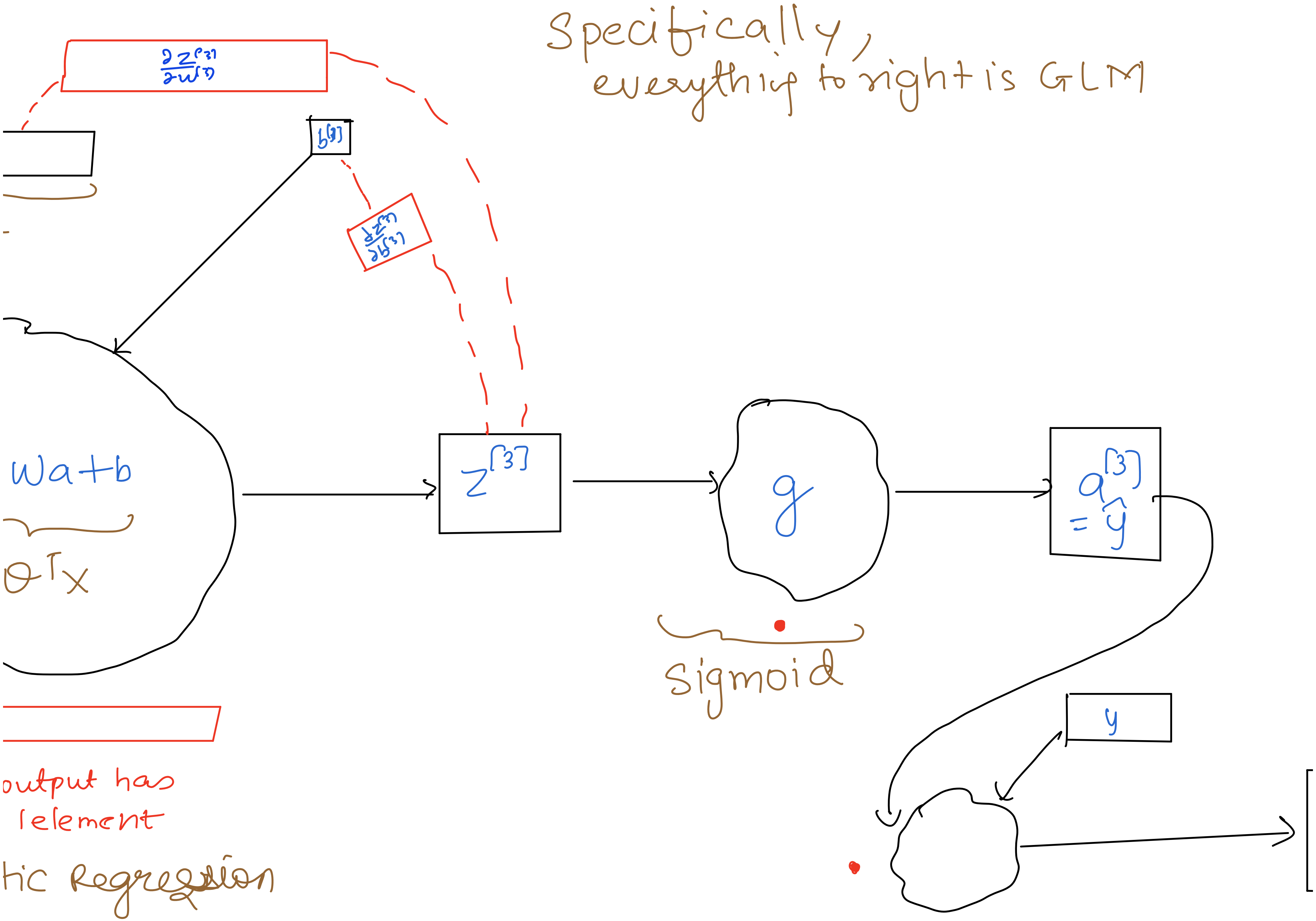
$$w_{ij}^{[l]} \sim N\left(0, \sqrt{\frac{2}{n^{[l]}+n^{[l-1]}}}\right) \quad \text{Xavier Initialization}$$

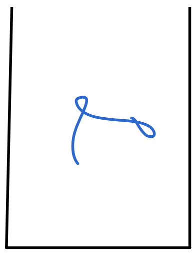
$$\sim \text{Unif}[-0.1, 0.1]$$


$a^{[0]}$
x
$w^{[1]}$
$b^{[1]}$
$z = wa + b$
$z^{[1]}$
$a = g(z)$
$a^{[1]}$



$w^{[2]}$
$b^{[2]}$
$Z = Wa + b$
$z^{[2]}$
$a = g(z)$
$a^{[2]}$
θ
$w^{[3]}$
x
$z =$
$W^{[2]}$
→ Jacobian
→ to make computation easy, we take derivative element wise, else it would be 3D, and very time consuming and it is very sparse
Diagonal matrix as element wise function. $a_1^{[2]}$ doesn't depend on $z_3^{[2]}$
$\phi_{w,b}(x)$
since only
Logis


Specifically,
everything to right is GLM
$\frac{\partial z^{[3]}}{\partial w^{[3]}}$
$b^{[3]}$
$\frac{\partial z^{[3]}}{\partial b^{[3]}}$
$wa+b$
$\theta^T x$
$z^{[3]}$
g
sigmoid
$a^{[3]} = \hat{y}$
y
output has
1 element
tic Regression



$$\frac{\partial \mathcal{L}}{\partial W^{[2]}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{11}^{[2]}} & \cdots & \frac{\partial \mathcal{L}}{\partial w_{13}^{[2]}} \\ \vdots & \ddots & \\ & \cdots & \frac{\partial \mathcal{L}}{\partial w_{23}^{[2]}} \end{bmatrix}$$

$$\frac{\partial \mathcal{L}}{\partial z^{[3]}} = \frac{\partial}{\partial z^{[3]}}\left[-y\log\hat{y} - (1-y)\log(1-\hat{y})\right]$$

$$= \frac{\partial}{\partial z^{[3]}}\left[-y\log\sigma(z^{[3]}) - (1-y)\log(1-\sigma(z^{[3]}))\right]$$

$$\vdots$$

$$= a^{[3]} - y$$