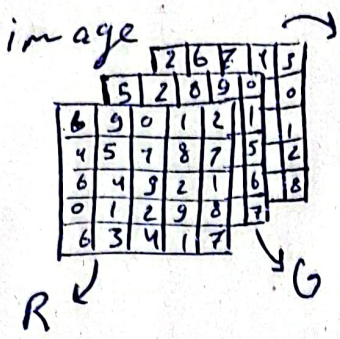
Binary classification (Log reg)



→ an image consists of 3 Layers (RGB)
Devectorizing the image is to take the matrix and turning it into a vector

$X = [69012457 \ldots\ldots 5289o \ldots\ldots 2674].T$

R G B

~~the vector size = R x G x B = 64 x 64 x 64~~

the vector size = Length of the image * width * 3

= 64 * 64 * 3 = 12288

$n_x = 12288$

Notation

single training example (x, y) → Lable $\{0, 1\}$

↳ n-dimensional feature vector

m → number of traing examples → $(x^{(1)}, y^{(1)})$

First training example.

First training example :-

$x^{(1)} = [216 \ 228 \ 172 \ 169 \ldots\ldots\ldots\ldots]$

$y^{(1)} = 0, 1$

Feature Matrix X : Contains all feature vectors x

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}_{n_x}$$

$\leftarrow m \rightarrow$

the height is the number of features.
the width is the number of training examples.

Lable vector Y :-

$$Y = [\, y^{(1)} \;\; y^{(2)} \;\; y^{(3)} \;\; \ldots \;\; y^{(m)} \,]$$

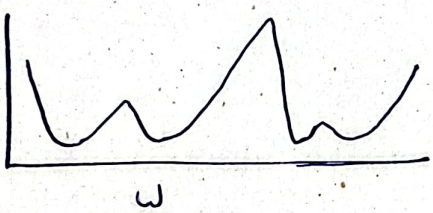
shape of X $\rightarrow (n_x, m)$

shape of Y $\rightarrow (1, m)$

Log reg cost function :-

for log reg we don't use a simple loss function

Like $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ because the curve of the w, b to the loss will be non convex



instead we us something more convex Like the log loss function

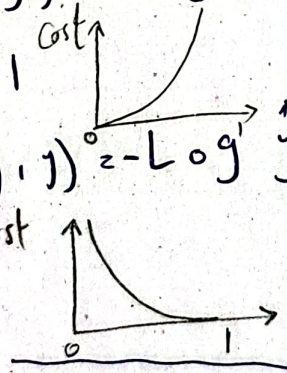
$$\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$$

for $y = 0$

$\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y})$ → for the loss to be (0) you need $-\log(1-\hat{y})$ to be zero so you need $\hat{y}$ to be as small as possible $\simeq 0$

for $y = 1$

$\mathcal{L}(\hat{y}, y) = -\log \hat{y}$ → for the loss to be (0) you need $-\log \hat{y}$ to be 0 and since $-\log(1) = 0$ you need $\hat{y}$ to be as big as possible $\simeq 1$



---

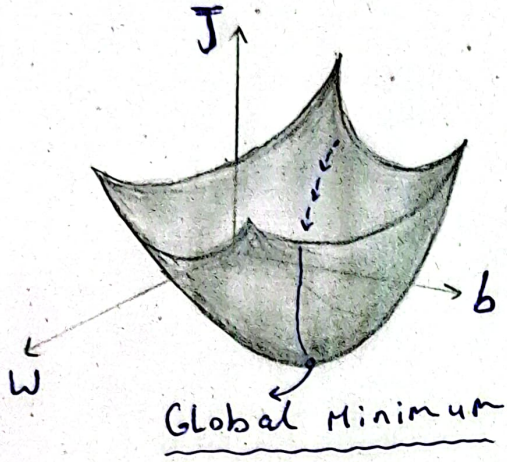
* Loss function works for a single training example but to measure the loss for the whole dataset we'll define a cost function :-

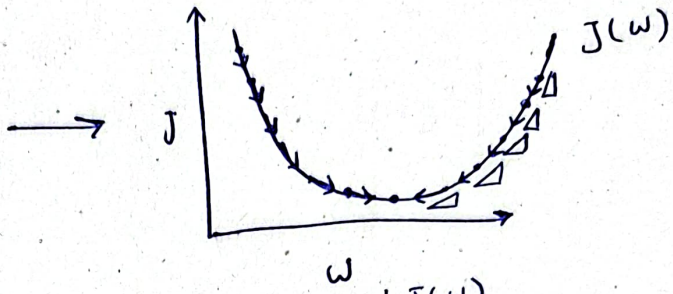
Cost function : $J(w, b) = \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$

(minus; m = num of training examples)

Gradiant descent :

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}, y) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})$$





$$w := w - \alpha \frac{d J(w)}{dw}$$

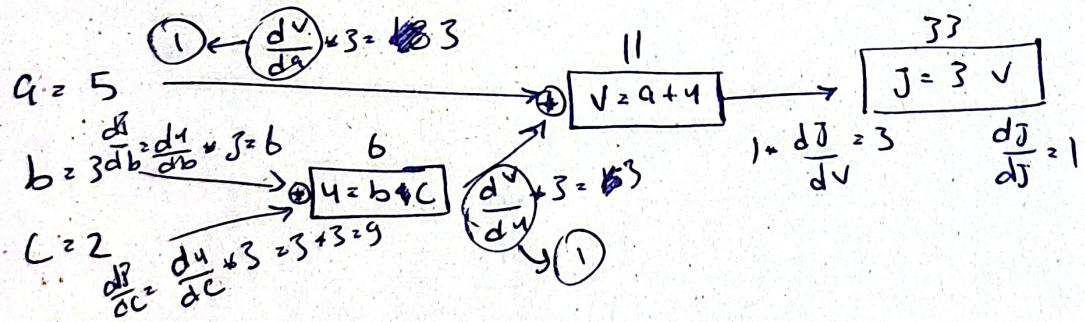
$J(w,b)$

$$w := w - \alpha \frac{d J(w,b)}{d w} = w - \alpha\, dw$$
$$b := b - \alpha \frac{d J(w,b)}{d b} = b - \alpha\, db$$

} in coding

Computation graph

$J(a,b,c) = 3(a+bc)$ || $u = bc$, $v = a+u$



$$\frac{dJ}{dv} = 3(v)$$

$v = a + u \rightarrow \frac{dv}{du} = 1, \frac{dv}{da} = 1$

$v = a + u$

$$\boxed{\frac{dJ}{da} \quad \frac{dJ}{db} \quad \frac{dJ}{dc}}$$

$$\frac{dJ}{dJ} = 1, \quad \frac{dJ}{dv} = \frac{dJ}{dJ}\frac{dJ}{dv} = 1 * 3 = 3$$
$$\frac{dJ}{du} = \frac{dJ}{dv}\frac{dv}{du} = 3 * 1 = 3$$
$$\frac{dJ}{da} = \frac{dJ}{dv}\frac{dv}{da} = 3 * 1 = 3$$

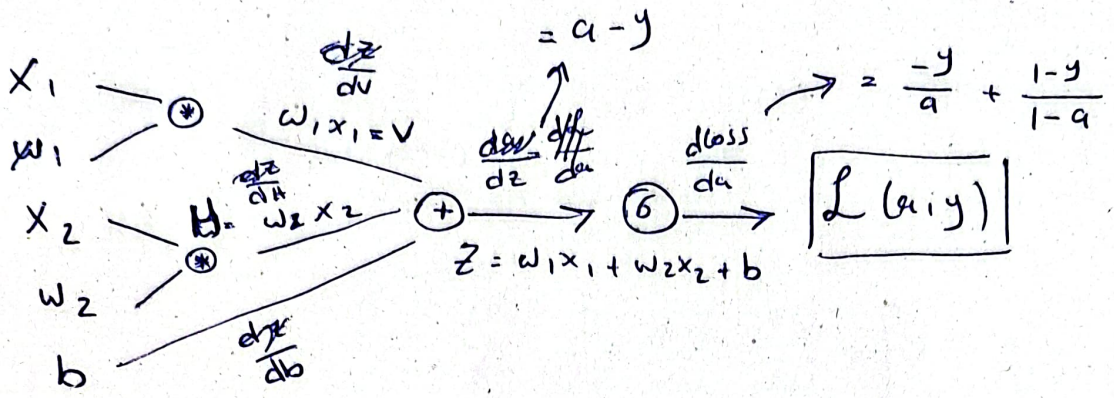
$J = 3v$

$$\frac{dJ}{dv} = 3$$
$$\frac{dv}{du} = 1 \qquad \frac{dv}{da} = 1$$
$$\frac{dJ}{dc} = 3 * 3 = 9$$
$$\frac{dJ}{db} = \frac{dJ}{dv}\frac{dv}{du}\frac{du}{db} = 3 * 1 * 2 = 6$$

$$z = w^T x + b$$

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

$$\mathcal{L}(a, y) = -(y \log(a) + (1-y)\log(1-a))$$



$$da = \frac{d\mathcal{L}}{da}$$

$$dz = \frac{d\mathcal{L}}{da}\frac{da}{dz} = da \cdot \frac{da}{dz}$$

$$dv = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dv} = dz \cdot \frac{dz}{dv}$$

$$dH = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dH} = dz \cdot \frac{dz}{dH}$$

$$db = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{db} = dz \cdot \frac{dz}{db} = dz$$

$$dx_1 = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dv}\frac{dv}{dx_1} = dv \cdot \frac{dv}{dx_1} = dz * w_1$$

$$dw_1 = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dv}\frac{dv}{dw_1} = dv\,\frac{dv}{dw_1} = dz * 1 * x_1 = dz \cdot x_1$$

$$dx_2 = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dv}\frac{dv}{dx_2} = dH \cdot \frac{dH}{dx_2} = dz * w_2$$

$$dw_2 = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dv}\frac{dv}{dw_2} = dH \cdot \frac{dH}{w_2} = dz * x_2$$

~~$db = \frac{d\mathcal{L}}{da}\frac{da}{dz}\frac{dz}{dv}\frac{dv}{db}$~~

$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y^{(i)})$$

$$\rightarrow a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b) \Leftrightarrow [dw_1^{(i)}, dw_2^{(i)}, db^{(i)}]$$

$$\frac{d}{dw_1} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial w_1} \mathcal{L}(a^{(i)}, y^{(i)})$$

dw_1

$$= \frac{df}{da} \frac{da}{dz} \frac{dz}{dv} \frac{dv}{dw_1} = \boxed{dv \frac{dv}{dw_1}}$$

$J=0$, $dw_1=0$, $dw_2=0$, $db=0$

For $i=1$ to m

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -\left[y^{(i)} \log(a^{(i)}) + (1-y) \log(1-a^{(i)})\right]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)}$

$dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)}$ ↕ $n=2$

$db \mathrel{+}= dz^{(i)}$

$J \mathrel{/}= m$

$dw_1 \mathrel{/}= m$

$dw_2 \mathrel{/}= m$

$db \mathrel{/}= m$

after the ~~first~~ loop

$dw_1 = \frac{dJ}{dw_1}$

so dw_1 is the accumulation of derivatives across the training set.

$w_1 := w_1 - \alpha\, dw_1$.

$w_2 := w_2 - \alpha\, dw_2$.] each step

$b := b - \alpha\, db$.

Vectorizing logistic regression

$z^{(1)} = w^T \cdot x^{(1)} + b$, $z^{(2)} = w^T \cdot x^{(2)} + b$, $z^{(3)} = w^T \cdot x^{(3)} + b$

$a^{(1)} = \sigma(z^{(1)})$, $a^{(2)} = \sigma(z^{(2)})$, $a^{(3)} = \sigma(z^{(3)})$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$ ↳ number of features

$m \rightarrow$ number of training examples.

$X.shape = (n_x, m)$

$$Z = \begin{bmatrix} z^{(1)} & z^{(2)} & z^{(3)} & \cdots & z^{(m)} \end{bmatrix}_{1 \times m} = w^T \cdot X + [b_1\ b_2\ b_3 \cdots b_m]$$

$$w^T \cdot \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}_{n_x, m}$$

$$= \underbrace{\left[w^T \cdot x^{(1)} + b\right]}_{z^{(1)}} \underbrace{\left[w^T \cdot x^{(2)} + b\right]}_{z^{(2)}} \cdots \underbrace{\left[w^T \cdot x^{(m)} + b\right]}_{z^{(m)}}$$

$Z = np.dot(w^T, X) + b$ (1,1) ↳ python broadcasting.

$A = \sigma(Z) = [\sigma(z^{(1)})\ \ \sigma(z^{(2)}) \cdots \sigma(z^{(m)})]$

$dz = A - Y$ $(1 \times m)$

$db = \frac{1}{m}\ np.sum(dz)$

$dw = \frac{1}{m} X\, dz^T =$

$Z = np.dot(w.T, X) + b$

$A = \sigma(z)$

$dz = A - Y$

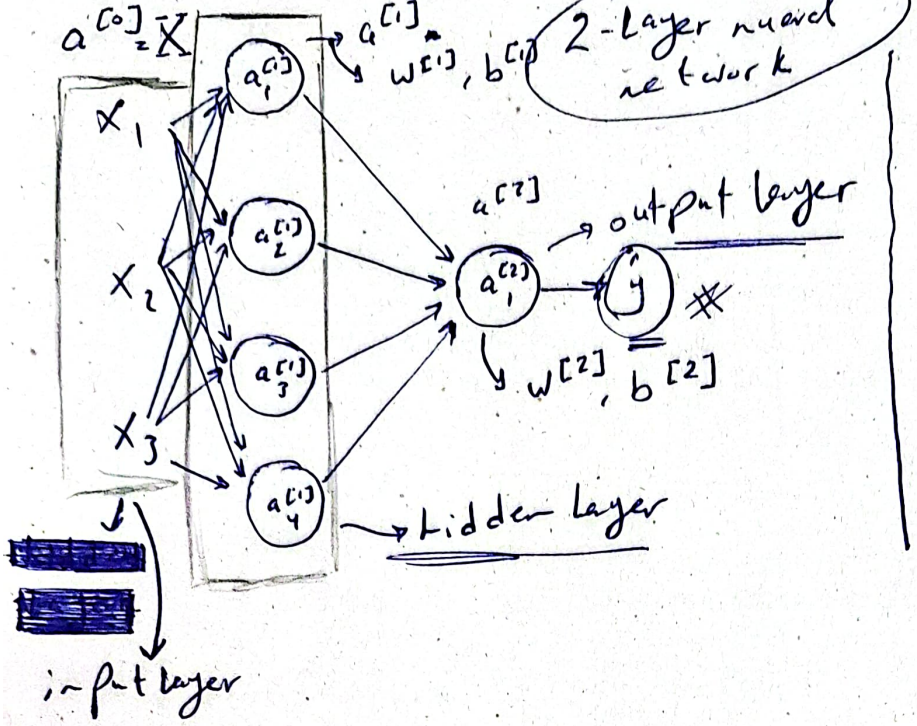
$dw = \frac{1}{m} X \cdot dz^T$

$db = \frac{1}{m}\ np.sum(dz)$

$w := w - \alpha\, dw$ → ✱

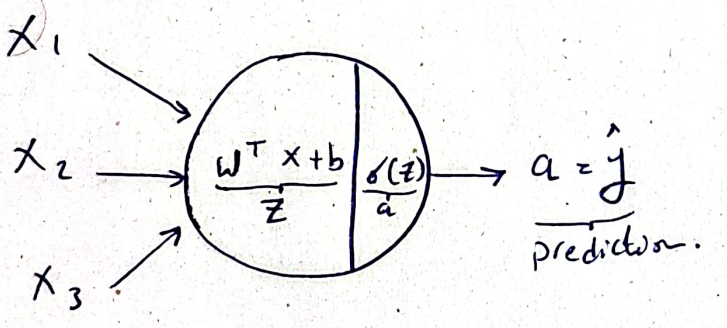
$b := b - \alpha\, db$ → ✱

Neural network representation

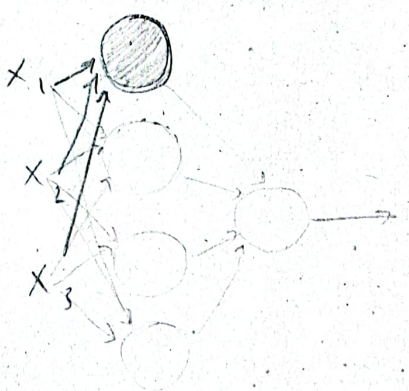


$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ \vdots \end{bmatrix}$$

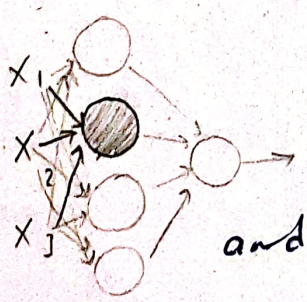
~~$a^{[0]}$~~ Computing the output of a neural network



$z = w^T \cdot x + b \rightarrow a = \sigma(z)$



$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$
$$a_1^{[1]} = \sigma(z_1^{[1]})$$



$$z_2^{[1]} = w_2^{[1]} x + b_2^{[1]}$$
$$a_2^{[1]} = \sigma(z_2^{[1]})$$

and so on

$$z_i^{[1]} = w_i^{[1]T} x + b_i^{[1]}$$
$$a_i^{[1]} = w_i^{[1]T} x + b_i^{[1]}$$

general for —

$$z_{(i)}^{[j]} = w_{(i)}^{[j]T} + b_{(i)}^{[j]}$$
$$a_{(i)}^{[j]} = \sigma(z_{(i)}^{[j]})$$

→ in the previous Logistic regression as a single nueron example::

* X was a matrix consisting of all images stacked horizontally.

$$\bar{X} = \begin{bmatrix} | & | & & | \\ x_{(1)} & x_{(2)} & \dots & x_{(m)} \\ | & | & & | \end{bmatrix} \uparrow n_x \downarrow \quad \longleftrightarrow m$$

and w was a row vector after it was transposed so each feature has a weight

$$Z = W^T \cdot X + b$$

$$= \begin{bmatrix} w_1 & w_2 & w_3 & w_4 \dots \end{bmatrix} \begin{bmatrix} | & | & \\ x_{(1)} & x_{(2)} & \dots \\ | & | & \end{bmatrix} + b \quad \hookrightarrow \text{scaler}$$

* each weight is going to be multiplied with a specific feature resulting in their dot product so we can calculate the overall cost.

$$Z = W^T X + b \rightarrow A = \sigma(Z) \rightarrow \boxed{J = \sum_{i=1}^{m} L(y, A)}$$

---

* when handeling a whole layer then w is going to be multiple vectors - each corresponding to a single nueron — stacked together in a matrix, and b is going to be a vector.

$$W^T = \begin{bmatrix} — w_1^{[1]} — \\ — w_2^{[1]} — \\ — w_3^{[1]} — \\ \vdots \\ — w_n^{[1]} — \end{bmatrix}, \quad b = \begin{bmatrix} b_1^{[1]}, & b_2^{[1]} & \dots & b_n^{[1]} \end{bmatrix}$$

$$Z^{[1]} = \begin{bmatrix} — & w^{[1]T} & — \\ — & w_1^{[1]T} & — \\ — & \vdots & — \\ — & w_n^{[1]T} & — \end{bmatrix} \cdot \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} + \begin{bmatrix} b^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_n^{[1]} \end{bmatrix}$$

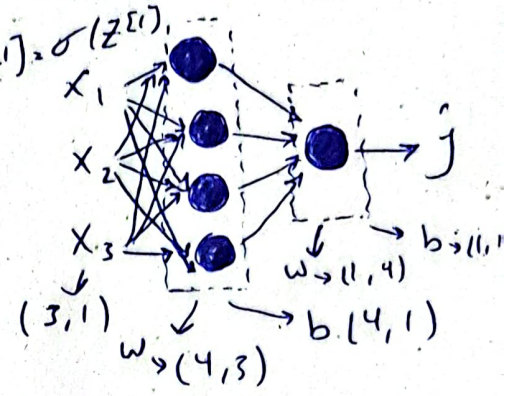
$$Z^{[1]} = W^{[1]T} \cdot X + b \Rightarrow A^{[1]} = \sigma(Z^{[1]})$$

($A^{[0]} = X$)

$(4,1) \quad (4,3) \quad (3,1) \quad (4,1)$

$$Z^{[2]} = W^{[2]T} \cdot A^{[1]} + b$$

$(1,1) \quad (1,4) \quad (4,1) \quad (1,1)$



$$\searrow A^{[2]} = \sigma(Z^{[2]}) = \hat{y}$$

For multiple training examples

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \uparrow n_x \quad \longleftarrow m \longrightarrow$$

$$Z^{[1]} = W^{[1]T} X + b^{[1]}$$
$$A^{[1]} = \sigma(Z^{[1]})$$
$$Z^{[2]} = W^{[2]T} \cdot A^{[1]} + b^{[2]}$$
$$A^{[2]} = \hat{Y} = \sigma(Z^{[2]})$$

$$Z^{[1]} = \begin{bmatrix} | & | \\ Z^{[1](1)} & Z^{[1](2)} \\ | & | \end{bmatrix}$$

$[\ \ ]$

$4 \times 4 \times 3 = 16 \times 3 = 48$

important

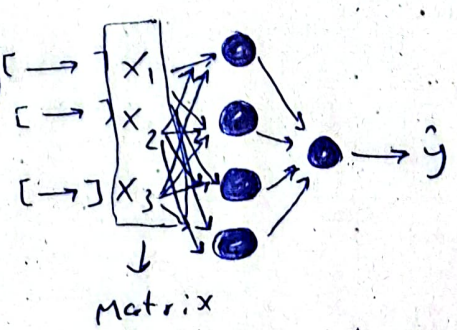
$$Z^{[1]} = W^{[1]T} \cdot X + b^{[1]} \to A^{[1]} = \sigma(Z^{[1]})$$

$(4,10) \quad (4,48) \quad (48,10) \quad (4,10)$

$$Z^{[2]} = W^{[2]T} \cdot A^{[1]} + b^{[2]}$$

$(1,10) \quad (1,4) \quad (4,10) \quad (1,10)$

↳ out put for 10 images prediction.

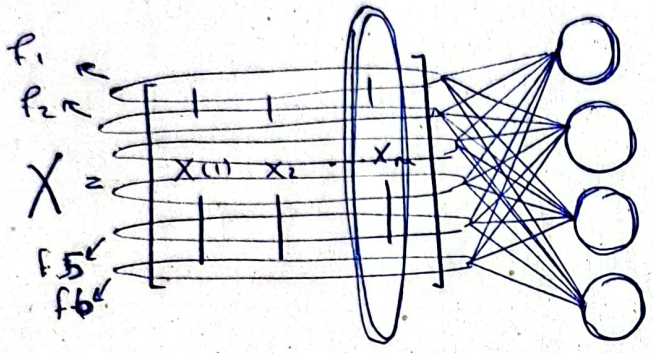


$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix}$$

Justification for the vectorized implementation

$$Z^{[1](1)} = W^{[1]T} \cdot X^{(1)} + b^{[1]}_0$$
$$Z^{[1](2)} = W^{[1]T} \cdot X^{(2)} + b^{[1]}_0$$
$$Z^{[1](3)} = W^{[1]T} \cdot X^{(3)} + b^{[1]}_0$$

$W^{[1]} =$ [ ]



→ Let's imagine that we do this process - one image at a time -.

* First image has 6 features and each feature is going to be an input to 4 neurons of the first layer.

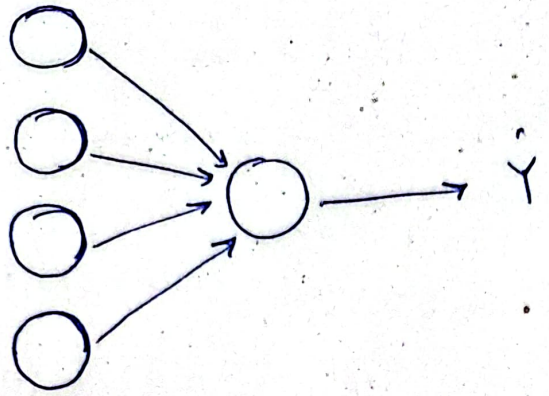
$X_1 \rightarrow (6,1)$, $W^{[1]} \rightarrow (4,6)$
$X_2 \rightarrow (6,1)$, $W^{[1]} \rightarrow (4,6)$
$X_3 \rightarrow (6,1)$, $W^{[1]} \rightarrow (4,6)$

$W^{[1]}X^{(1)} =$ [ ] $W^{[1]}X^{(2)} =$ [ ] ...

$Z^{[1]} =$ [ ] 4 × n

$W^{[1]T}X^{(1)} =$ [ ] (4,1)
neuron 1
neuron 2
... 3
... 4

these are the Activations of the four neurons on the 6 features of the first training example



$X = \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(10)} \\ | & & | \end{bmatrix} \Rightarrow Z^{[1]} = \begin{bmatrix} | & & | \\ \vdots & \cdots & \vdots \\ | & & | \end{bmatrix}$

$X \to (6, 10)$ $\qquad Z^{[1]} \to (4, 10)$

$W^{[1]} = (4, 6)$

$$\boxed{A^{[1]} = \sigma(Z^{[1]} \longrightarrow (4, 10)}$$

$$Z^{[2]} = W^{[2]T} \cdot A^{[1]} + b^{[2]}$$

$$Z^{[2](1)}_{(1,1)} = W^{[2]}_{(1,4)} \cdot A^{[1](1)}_{(4,1)}$$

$$Z^{[2](2)}_{(1,1)} = W^{[2]}_{(1,4)} \cdot A^{[1](2)}_{(4,1)}$$

$Z^{[2](1)} = [\bullet]$, $Z^{[2](2)} = [\bullet]$, $Z^{[2](3)} = [\bullet]$

$Z^{[2]} = [\bullet \quad \bullet \quad \bullet \quad \cdots \quad \bullet]$ ← m →

$A^{[2]} = \hat{Y} = \sigma(Z^{[2]})$

↳ predictions for the classes of the 10 images

Gradient descent for 1-hidden layer network

Params → $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}$

dims → $n_x = n^{[0]}, n^{[1]}, n^{[2]} = 1$

n_x ↳ $n^{[1]}$ ↳ hidden units

→ $w^{[1]} \rightarrow (n^{[1]}, n^{[0]})$

→ $b^{[1]} \rightarrow (n^{[1]}, 1)$

→ $w^{[2]} \rightarrow (n^{[2]}, n^{[1]})$

→ $b^{[2]} \rightarrow (n^{[2]}, 1)$

Cost $= J(w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]})$

$= \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y, A^{[2]} = \hat{Y})$

GD

↳ Repeat

→ Compute the predictions $(\hat{y}^{(1)} \dots \hat{y}^{(m)})$ ⇒ Forward prob

→ ~~dddddd~~ Compute the Cost

→ Compute the derivatives $dw^{[1]} = \frac{d\hat{J}}{dw^{[1]}}$, $db^{[1]} = \frac{d\hat{J}}{db^{[1]}}$

$dw^{[2]}, db^{[2]}$

⇒ back prob

→ update → $w^{[1]} = w^{[1]} - \alpha\, dw^{[1]}$

$b^{[1]} = b^{[1]} - \alpha\, db^{[1]}$

$w^{[2]} = w^{[2]} - \alpha\, dw^{[2]}$

$b^{[2]} = b^{[1]} - \alpha\, db^{[2]}$

---

forward prob

$z^{[1]} = w^{[1]} x + b^{[1]}$ → ~~A[1]~~

$A^{[1]} = g^{[1]}(z^{[1]})$

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

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

back prob

$dz^{[2]} = A^{[2]} - Y$

$dw^{[2]} = \frac{1}{m}\, dz^{[2]}\, A^{[1]T}$

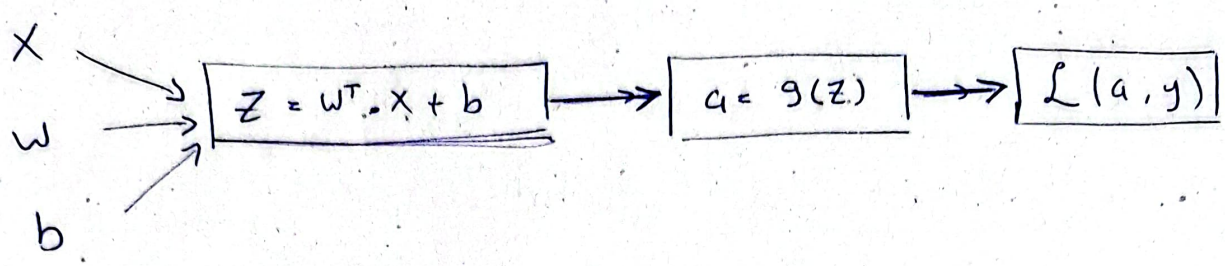
$db^{[2]} = \frac{1}{m}\, np.sum(dz^{[2]})$

$dz^{[1]} = \frac{1}{m} d\, w^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$

$dw^{[1]} =$ _____

$db^{[1]} =$ _____

Backpropagation intuition



Forward prob

$$z = w^T \cdot X + b \rightarrow a = g(z) \rightarrow \mathcal{L}(a,y) = -y \log a - (1-y)\log(1-a)$$

(Cost)

Back prob

$$\frac{d\mathcal{L}}{d\mathcal{L}} = 1$$

$$\frac{d\mathcal{L}}{da} \quad \mathcal{L}(a,y) = \frac{-y}{a} + \frac{1-y}{1-a}$$

$\rightarrow$ "da"

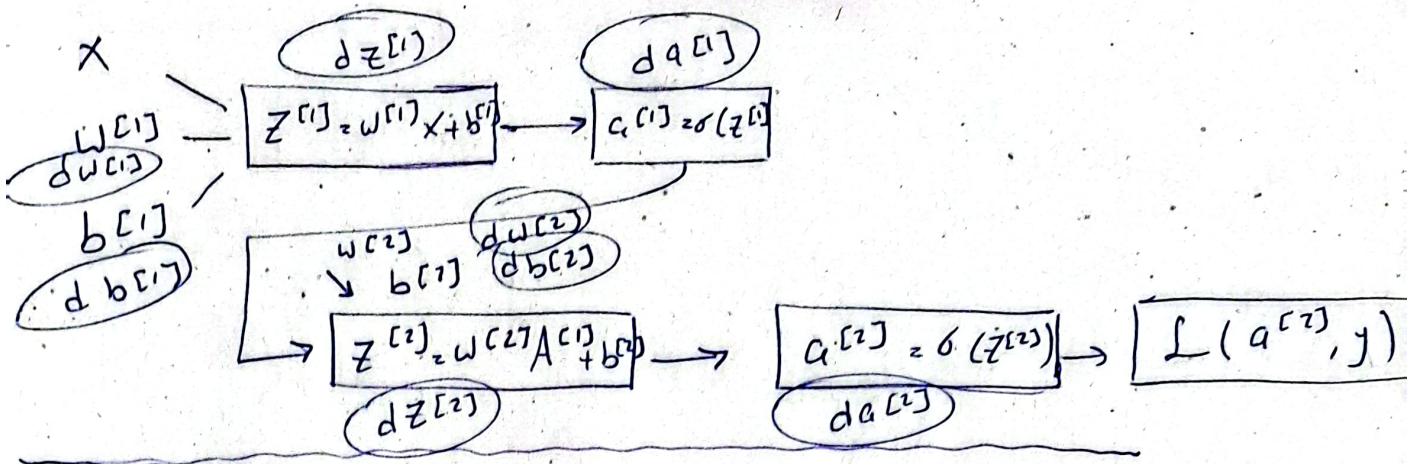
$$dz = a - y = da \cdot g'(z) \quad \Rightarrow g(z) = \sigma(z)$$

$$dz = \frac{d\mathcal{L}}{dz} = \boxed{\frac{d\mathcal{L}}{da}} \cdot \boxed{\frac{da}{dz}} = "da" \cdot \boxed{\frac{d}{dz}(\sigma(z))}\, g'(z)$$

$\rightarrow$ "da"

$$= da \cdot g'(z)$$

$$dw = dz \cdot x$$
$$db = dz$$



Forward prob

$$z^{[1]} = W^{[1]T} X + b^{[1]} \Rightarrow a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = W^{[2]T} a^{[1]} + b^{[2]} \Rightarrow a^{[2]} = \sigma(z^{[2]})$$

$$Cost = -y \log a^{[2]} - (1-y) \log(1 - a^{[2]})$$

Back prob

$$A = \sigma(z)$$

$$\frac{da}{dz} = A\,\sigma(z)\,(1 - \sigma(z))$$

$$A = A(1 - A)$$

$$\rightarrow da^{[2]} = \frac{-y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}$$

$$\rightarrow dz^{[2]} = \frac{dL}{da^{[2]}} \cdot \frac{da^{[2]}}{dz^{[2]}} = \frac{-y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}} * \left(a^{[2]}(1-a^{[2]})\right)$$

$$\rightarrow da^{[2]}$$

$$= a^{[2]} - y$$

$$\rightarrow dW^{[2]} = dz^{[2]} \cdot a^{[1]T}$$

$$\rightarrow db^{[2]} = dz^{[2]}$$

$$\rightarrow da^{[1]} = \frac{dL}{da^{[2]}} \cdot \frac{da^{[2]}}{dz^{[2]}} \cdot \frac{dz^{[2]}}{da^{[1]}} = (a^{[2]} - y) \cdot W^{[2]}$$

$$= a^{[2]} - y$$

$$= a^{[2]} - y \cdot W^{[2]} = dz^{[2]} \cdot W^{[2]}$$

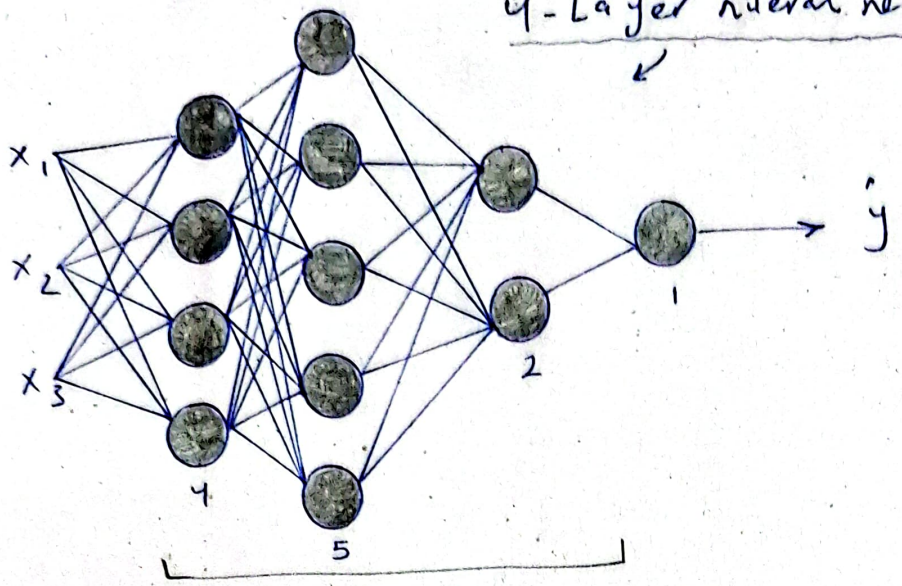
$$\rightarrow \frac{dz^{[1]}}{da^{[1]}} = g^{[1]'}(z^{[1]})$$

$$dz^{[1]} = da^{[1]} \cdot \frac{dz^{[1]}}{da^{[1]}}$$

$$= dz^{[2]} \cdot W^{[2]} * g^{[1]'}(z^{[1]})$$

Deep nueral networks

4-Layer nueral network



3-hidden Layers

→ $L = 4$ (# Layers)

→ $n^{[l]}$ = units in Layer l → $n^{[1]} = 4$, $n^{[2]} = 5$, 2, 1 ...

→ $n^{[0]} = n_x = 3$ → input features

→ $a^{[l]}$ = activations in Layer l

→ $a^{[l]} = g^{[l]}(z^{[l]})$, → $a^{[0]} = X$, $a^{[L]} = \hat{y}$

→ $w^{[l]}$ = weights for $z^{[l]}$

→ $b^{[l]}$ = biases for $z^{[l]}$

Forward probagation

- $z^{[1]} = w^{[1]} \cdot X + b^{[1]}$
- $a^{[1]} = g^{[1]}(z^{[1]})$
- $z^{[2]} = w^{[2]} \cdot a^{[1]} + b^{[2]}$
- $a^{[2]} = g^{[2]}(z^{[2]})$
- $z^{[3]} = w^{[3]} \cdot a^{[2]} + b^{[3]}$
- $a^{[3]} = g^{[3]}(z^{[3]})$
- $z^{[4]} = w^{[4]} \cdot a^{[3]} + b^{[4]}$
- $a^{[L]} = \hat{y} = g^{[4]}(z^{[4]})$

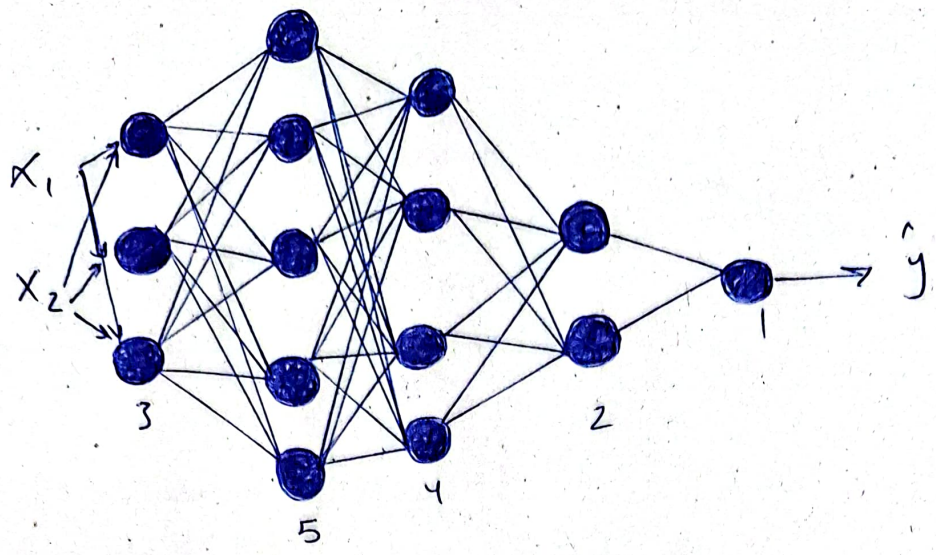
↳ most likely sigmoid.

$$z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

Vectorized form

$$Z^{[1]} = W^{[1]} X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$\vdots$$
$$X = A^{[0]}$$
$$Z^{[4]} = W^{[4]} \cdot A^{[3]} + b^{[4]}$$
$$A^{[4]} = g^{[4]}(Z^{[4]})$$

↳ $\hat{Y}$ → predictions



$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

$(3,1) \quad (3,2) \quad (2,m) \quad (3,1)$

$(n^{[l]}, m) \quad (n^{[l]}, n^{[l-1]}) \quad (n_x, m)$

$W \longrightarrow (n^{[l]}, n^{[l-1]})$

$b \longrightarrow (n^{[l]}, 1)$

- Let is imagine that we have 10 training examples.

$$\rightarrow Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$
$(3,10) \quad (3,2) \quad (2,10) \quad (3,1)$

$$\rightarrow Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$$
$(5,10) \quad (5,3) \quad (3,10) \quad (5,1)$

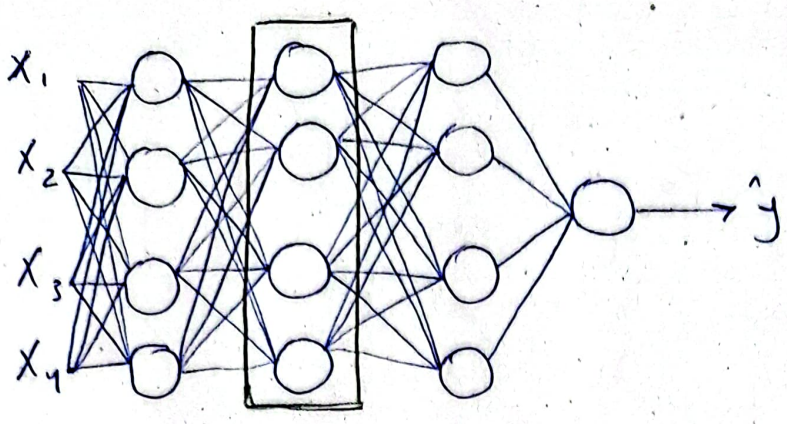
$$\rightarrow Z^{[3]} = W^{[3]} \cdot A^{[2]} + b^{[3]}$$
$(4,10) \quad (4,5) \quad (5,10) \quad (4,1)$

$$\rightarrow Z^{[4]} = W^{[4]} \cdot A^{[3]} + b^{[4]}$$
$(2,10) \quad (2,4) \quad (4,10) \quad (2,1)$

$$\rightarrow Z^{[5]} = W^{[5]} \cdot A^{[4]} + b^{[5]}$$
$(1,10) \quad (1,2) \quad (2,10) \quad (1,1)$

b is broadcasted by python automatically.

$$A^{[5]} = \sigma(Z^{[5]}) \longrightarrow \text{predictions}$$



Layer 2 → $w^{[2]}$, $b^{[2]}$

→ $z^{[2]} = w^{[2]} \cdot A^{[1]} + b^{[2]}$ → $A^{[2]} = g^{[2]}(z^{[2]})$

↳ input: $A^{[1]}$, output: $z^{[2]}$, $A^{[2]}$

Backprob for Layer 2

input, $da^{[2]}$, output: $da^{[1]}$, $dw^{[2]}$, $db^{[2]}$

+ Cache ($z^{[2]}$)

---

for Layer l: $w^{[l]}$, $b^{[l]}$

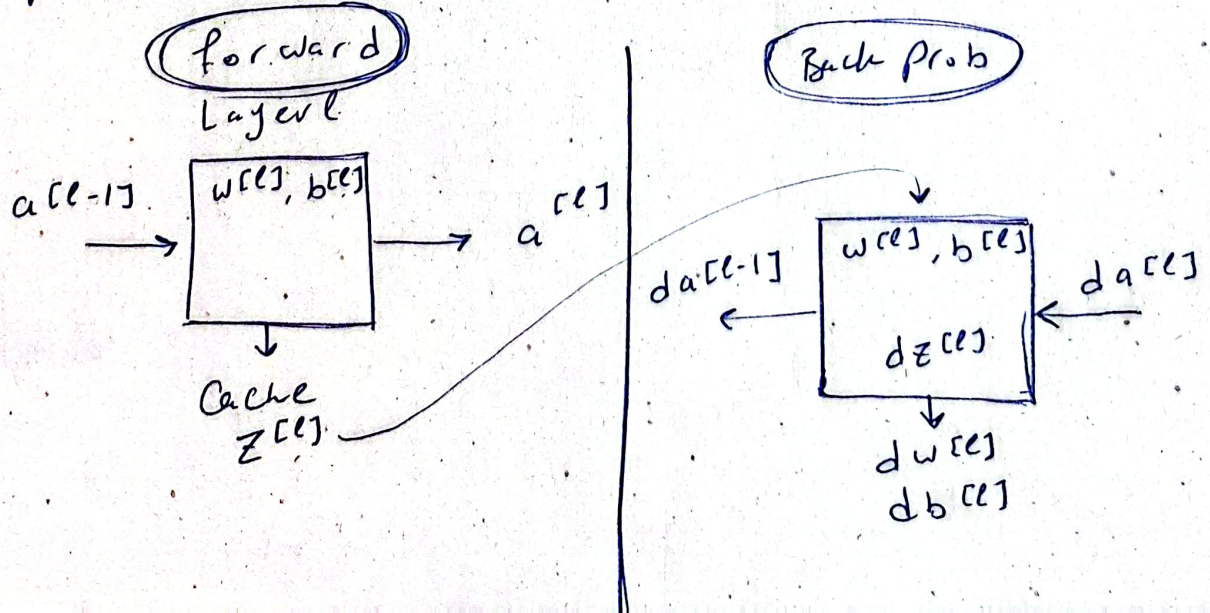
→ $z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$

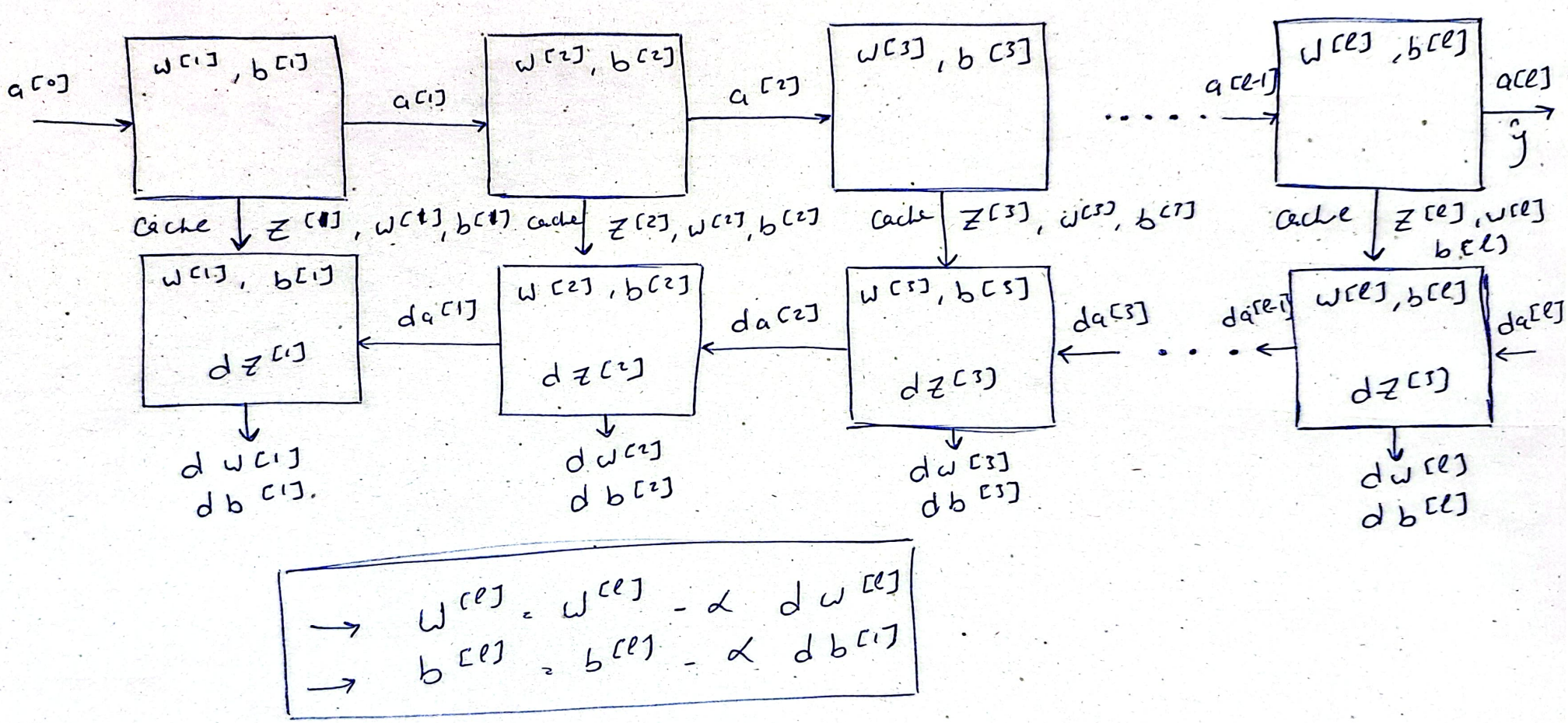
forward

input → $a^{[l-1]}$, output → $a^{[l]}$

Backprop

input → $da^{[l]}$, output → $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$





$$w^{[l]} = w^{[l]} - \alpha \, dw^{[l]}$$

$$b^{[l]} = b^{[l]} - \alpha \, db^{[l]}$$

forward probagation

- input $a^{[l-1]}$
- output $a^{[l]}$, cache ($z^{[l]}$)

$X = a^{[0]}$

→ $z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$

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

Back probagation

- input $da^{[l]}$
- output $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$

→ $dz^{[l]} = da^{[l]} * g^{[l]'}(z^{[l]})$

→ $dw^{[l]} = dz^{[l]} \cdot a^{[l-1].T}$

→ $db^{[l]} = dz^{[l]}$

↳ $da^{[l-1]} = w^{[l]T} \cdot dz^{[l]}$

---

vectorized form

$dz^{[l]} = dA^{[l]} * g^{[l]'}(z^{[l]})$

$dw^{[l]} = (dz^{[l]} \cdot A^{[l-1].T})/m$

$db^{[l]} = (np.sum(dz^{[l]}, axis=1, keepdims=True))/m$

$dA^{[l-1]} = w^{[l].T} \cdot dz^{[l]}$
