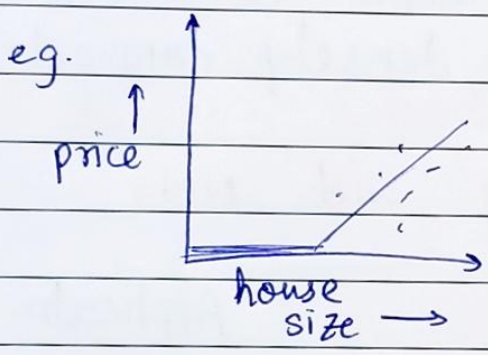
Date : 19/7/24

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☑ ☐ ☐

NEURAL NETWORKS & DEEP LEARNING (C-1)

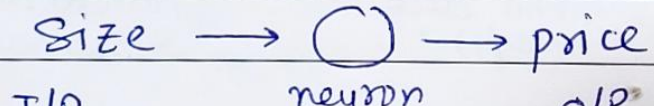
Neural Network



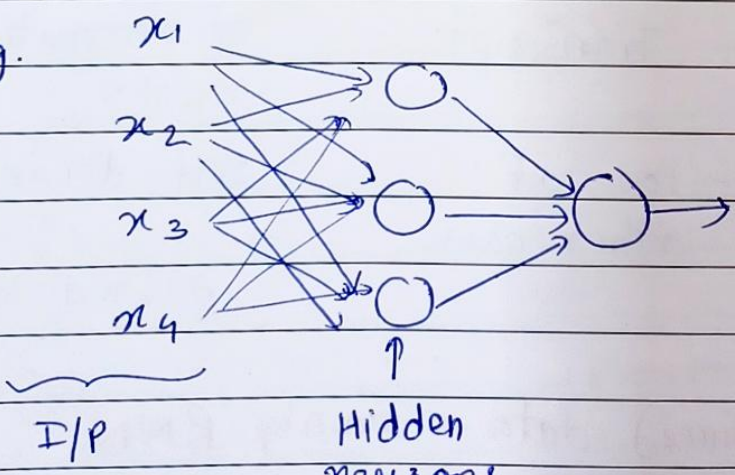
ReLU function.

Rectified Linear Unit

max (–, 0)



eg.



→ (Koala example)

Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

N.N. has ability to decide by itself what I/P features each of the hidden neuron wants/requires to do its computation.

Every I/P feature conn. to every middle neurons ⇒ densely connected.

Supervised Learning with NN.

I/P	O/P	Applicatn.
Ad, User Info.	Will click (0/1)?	Advertising ↳ Standard NN
Image	object	Tagging ↳ CNN
Audio	Transcript	Sp. recognition ↳ RNN
Img, Radar	Pos. of other cars	Self-driving ↳ Custom/ Hybrid NN

Temporal (time) data → mainly RNN.

↑ Data & ↑ Scale (i.e. more layers of NN)

⇓

↑ Performance

Date : .....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

Structured Data

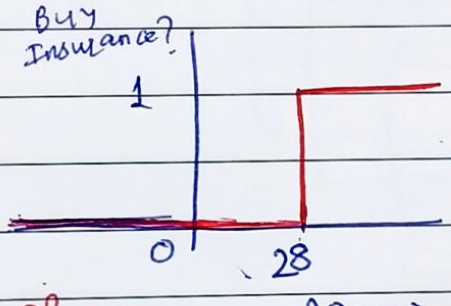
⇒ eg.

Size	#bedrooms	Price
↓	↓	↓

Unstructured Data

⇒ Audio, Image, Text

Basics – Activation functions.

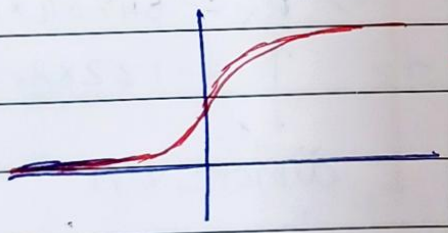


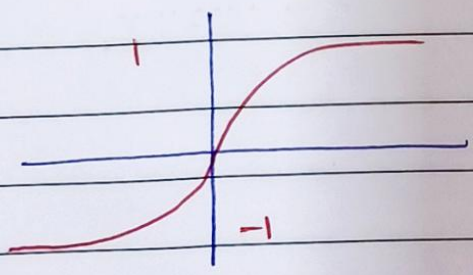
STEP Functn

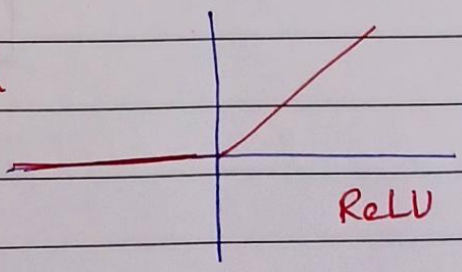
→ eg. People above 28y. will buy & below won't.

But X.

↓







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

sigmoid

Date :.....

MON TUE WED THU FRI SAT SUN

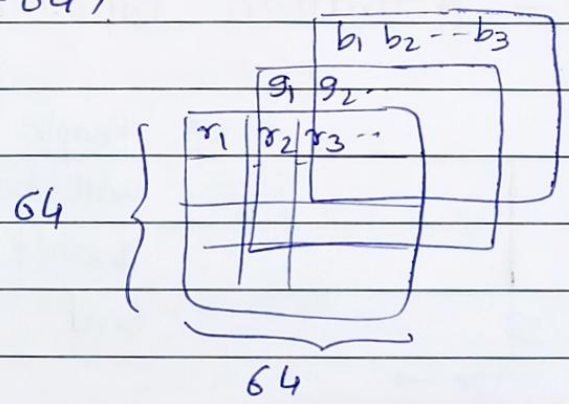
Hidden Layer mostly - ReLU or Leaky ReLU.

O/P → sigmoid mostly.

Binary classification

O/P → 0 or 1

eg. cat image (64 × 64) ⟶ cat OR non-cat



Feature vector

$$X = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ g_1 \\ g_2 \\ \vdots \\ b_1 \\ b_2 \\ \vdots \end{bmatrix} \Rightarrow n_x = 64 \times 64 \times 3 = 12288$$

↑ dimension.

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

(x, y) = Training eg.

$x \in \mathbb{R}^{n_x}$, $y = \{0, 1\}$

$m \rightarrow$ # training examples:

$\{(x^1, y^1), (x^2, y^2) \cdots (x^m, y^m)\}$

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

$\underbrace{\qquad\qquad}_{m}$

compact notation to store m training examples.

For Y also, stack in cols.

$$Y = [y^1, y^2 \cdots y^m]$$

Date :
MON TUE WED THU FRI SAT SUN

Logistic Regression

→ Supervised ML Algo. used for binary class. tasks.

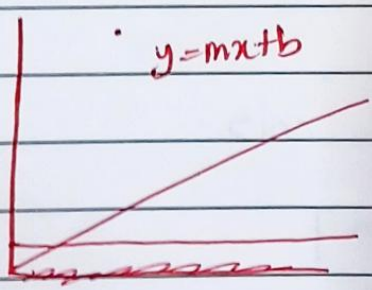
Given I/P x, $\hat{y} = P(y=1|x)$ probability

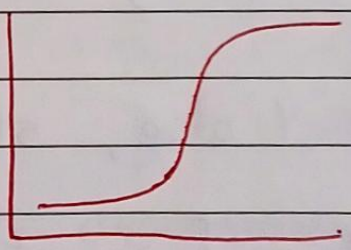
Params: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

O/P: $\hat{y} = w^T x + b$? is not bet^n [0,1]

$\therefore \hat{y} = \sigma(w^T x + b)$

z ↳ limits 0 to 1





$y = \frac{1}{1+e^{-(mx+b)}}$

no. of weights (w) = no. of I/P features.

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

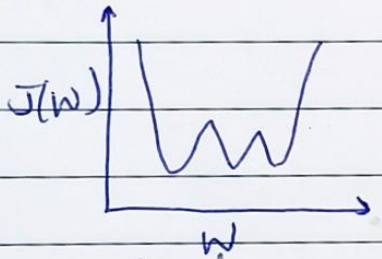
Loss Function:

→ Measures how far an estimated value is from its true value.

→ For general Linear Regr:

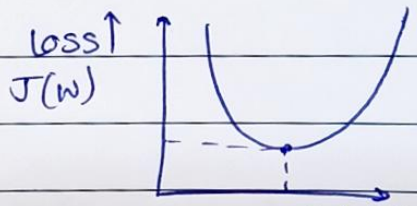
$$\text{Loss} = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

→ But for Logistic Regr. ↑ ✱ don't work.



Gradient Descent

Curve with multiple local minima



optimal Grad. Desc.

Loss fn applied to single training eg.
Cost fn: Cost of parameters.

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

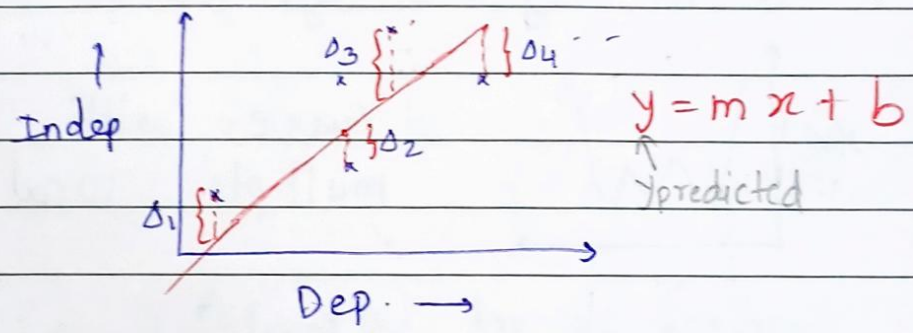
$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} L(\hat{y}^i, y^i)$$

Date : 20/7/24
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☑ ☐

Linear Regression

eg. House area ⟶ Price.

Useful when the relationship betn dependent & independent var can be assumed to be linear.



Lin. Reg. with multiple variables.

price = m_1 × area + m_2 × bedrooms
+ m_3 × age + b

Mean Squared Error

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - y_{predicted})^2$$

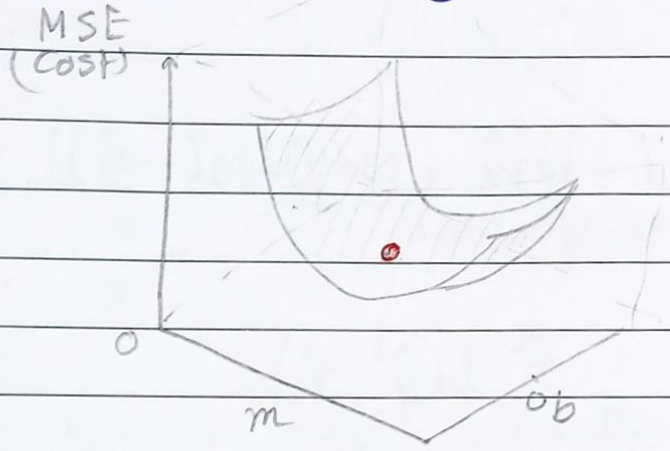
n = no. of data pts avail

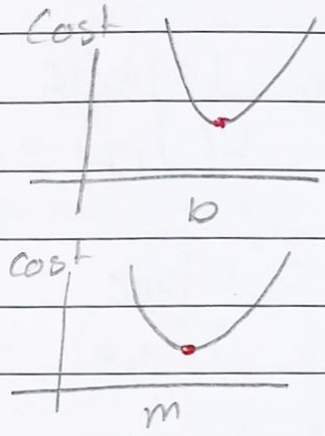
Also called Cost function

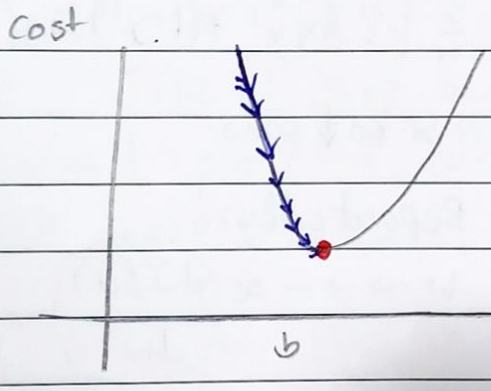
Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Gradient Descent

↳ algo to find best fit line for given training dataset







We will take baby steps to move towards minima & slowly decrease the length of babystep

Achieved using derivative

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - (mx_i + b))^2$$

$$\partial/\partial m = \frac{2}{n}\sum_{i=1}^{n} -x_i\,(y_i - (mx_i + b))$$

$$\partial/\partial b = \frac{2}{n}\sum_{i=1}^{n} -(y_i - (mx_i + b))$$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

$m = m - \text{Learning rate} \cdot \partial/\partial m$

$b = b - \text{"} \cdot \partial/\partial b$

In Logistic :

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

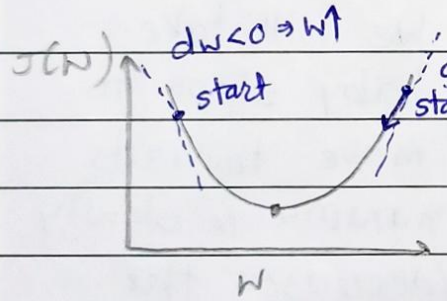
↳ for each individual data

Cost func

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

for entire training

$$= -1/m \sum_{i=1}^{m} [y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)]$$



Repeatedly:

$$w \to w - \alpha \frac{dJ(w)}{dw}$$

dw

$$w = w - \alpha \, \partial J(w,b)/\partial w$$

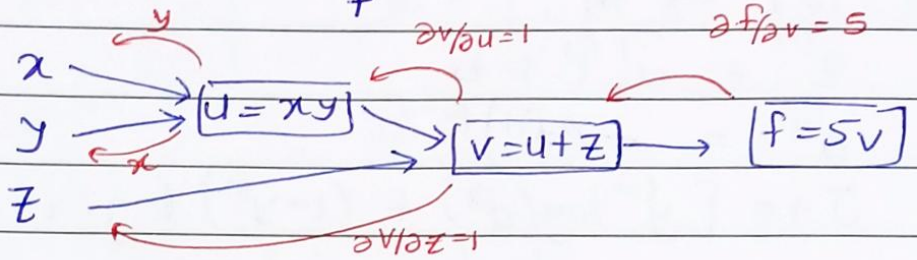
$$b = b - \alpha \, \partial J(w,b)/\partial w$$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Computation Graphs:

$f(x,y,z) = 5(xy+z)$

$u = xy$, $v = u+z$, $f = 5v$



This is useful in cases where we need to optimize final o/p variable.

i.e. $J(w,b)$ cost function ✓

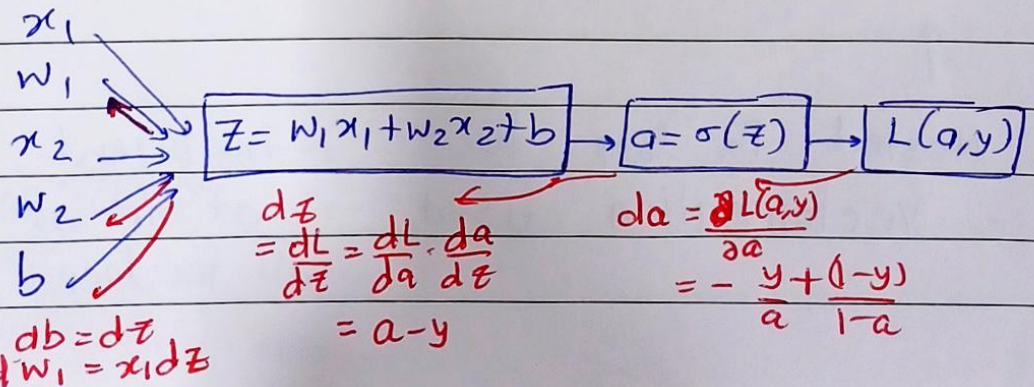
Forward prop → calc. J values.

Back prop → calc. derivatives

$$\therefore \frac{\partial f}{\partial x} = \frac{\partial f}{\partial v} \times \frac{\partial v}{\partial u} \times \frac{\partial u}{\partial x} = 5 \times 1 \times y = 5y$$

$\frac{dJ}{da}$ or "da" in short

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



Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

But this was only for 1 training ex.
For m examples:

$J=0;\ dw_1=0;\ dw_2=0;\ db=0$

for $i=1 \rightarrow m$:

$$z^{(i)} = w_x^{T(i)} + b$$
$$a^{(i)} = \sigma(z^{(i)})$$
$$J\ += \left[y^{(i)} \log(a^{(i)}) + (1-y^{(i)}) \log(1-a^{(i)})\right]$$

$$dz^{(i)} = a^{(i)} - y^{(i)}$$
$$dw_1\ += x_1^{(i)}\, dz^{(i)}$$
$$dw_2\ += x_2^{(i)}\, dz^{(i)}$$
$$db\ += dz^{(i)}$$

this is not separate for indiv. eg.

$n=2$ for loop run 2 times.

to eliminate for-loop:
$dw = np.zeros((n_x,1))$
$dw\ += x^{(i)} dz^{(i)}$

$J\ /= m$

$dw_1\ /= m \quad dw_2\ /= m \quad db\ /= m$

$w_1 := w_1 - \alpha\, dw_1$
$w_2 := w_2 - \alpha\, dw_2$
$b := b - \alpha\, db$

} update parameters (weights)

↑

Nested for loop here ✗ Inefficient
∴ Vectorization used. → got rid of all for loops.

np.zeros((0,1)) creates 2D arr with
0 rows 1 col ⇒ no elem.
→ To initialize empty array

Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Vectorization

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

$$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix} \quad x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$$

$w \in \mathbb{R}^{n_x}$ $\quad x \in \mathbb{R}^{n_x}$

To calc. dot product:

Non-vectorized	Vectorized
z = 0 for i in range n_x z += w[i] · x[i] z += b	z = np.dot(w, x) + b

GPU & CPU → Parallelization instructions (SIMD)

Vectorizing Log. Reg.

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

$z_1 = w^T x^1 + b$ — $a_1 = \sigma(z^1)$

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

$\begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix}$

$= w^T X + b$

w^T: $(1, n_x)$; b: $(1, m)$

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

transpose. — automatically converts to $(1, m)$

$A = [a^{(1)}\ a^{(2)}\ a^{(3)} \cdots a^{(m)}] = \sigma(Z)$

For backpropagation vectorizing:

$A = [a^{(1)}\ a^{(2)} \cdots a^{(m)}]$

$Y = [y^{(1)}\ y^{(2)} \cdots y^{(m)}]$

Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

$$dZ = [dz^{(1)} \; \cdots \; dz^{(m)}]$$
$$= A - Y$$

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

$$dw = \frac{1}{m} \cdot X(dZ)^T$$
$$= \frac{1}{m} \begin{bmatrix} x^{(1)} & \cdots & x^{(m)} \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ dz^{(m)} \end{bmatrix}$$

$$w := w - \alpha dw$$
$$b := b - \alpha db$$

Broadcasting

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 100 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix} = \begin{bmatrix} 101 \\ 102 \\ 103 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 100 & 200 & 300 \end{bmatrix}$$

$$= \begin{bmatrix} \quad & \quad & \quad \\ \quad & \quad & \quad \end{bmatrix} + \begin{bmatrix} 100 & 200 & 300 \\ 100 & 200 & 300 \end{bmatrix}$$

Date :
MON TUE WED THU FRI SAT SUN

(m,n) matrix	$\pm$	(1,n)	$\longrightarrow$ (m,n)
	$\times$ /	(m,1)	$\longrightarrow$ (m,n)

Numpy vectors

```
a = np.random.randn(5)
```

O/P : [-- , -- , -- , -- , --]

a.shape → (5,)

Rank 1 Array

DON'T USE LIKE THIS

a.T same.

⇒ It doesn't behave like either row / column vector.

Instead, `a = np.random.randn(5,1)`

To check: `assert(a.shape == (5,1))`

If there is Rank 1 : `a.reshape(5,1)`

$\hat{y} = P(y=1 \mid x)$

If $y=1$: $P(y|x) = \hat{y}$

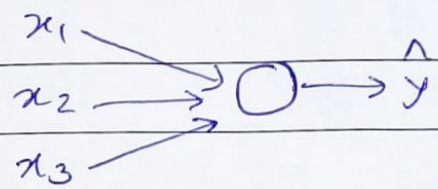
If $y=0$: $P(y|x) = 1-\hat{y}$

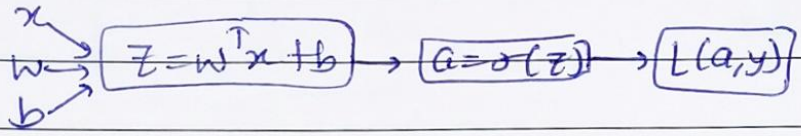
$\Rightarrow P(y|x) = (\hat{y})^{y} \cdot (1-\hat{y})^{(1-y)}$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

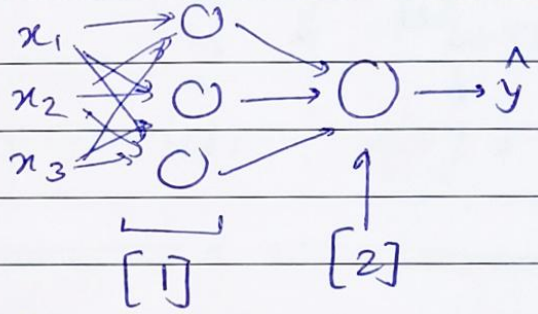
Neural Network Representation:

Logistic Regr:





Neural Net:

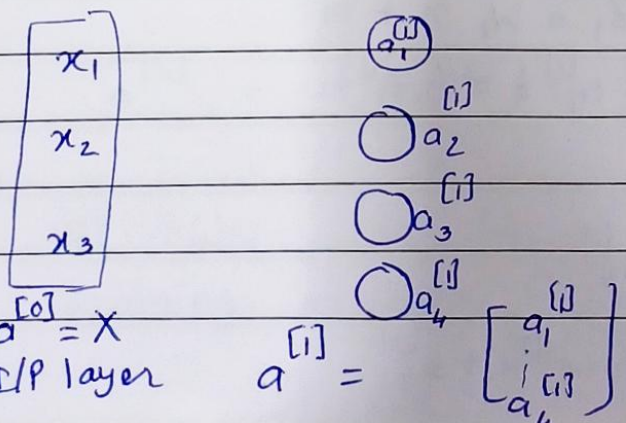


$z^{[1]} = W^{[1]} x + b^{[1]} \rightarrow a^{[1]} = \sigma(z^{[1]})$

$L(a^{[2]}, y) \leftarrow a^{[2]} = \sigma(z^{[2]}) \leftarrow z^{[2]} = W^{[2]} x + b^{[2]}$

2 Layer NN representation:

↳ 1 hidden + 1 o/p.



$a^{[0]} = X$

I/P layer

$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix}$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Hidden layer parameters:

$w^{[1]}$ & $b^{[1]}$

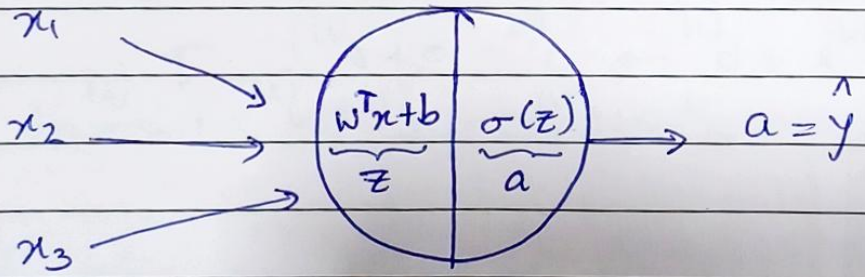
shape: $(4,3)$ $(4,1)$

bcz 4 nodes in hidden layer. bcz 3 I/P features

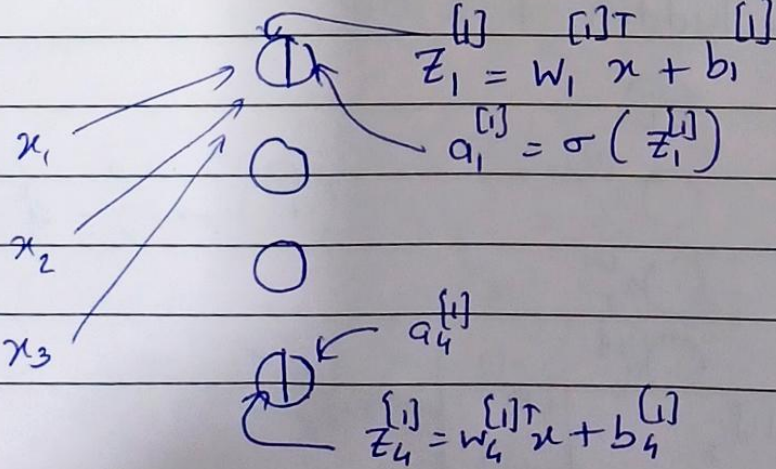
O/P Layer Parameter:

$w^{[2]}$ & $b^{[2]}$

$(1,4)$ $(1,1)$



Logistic



[l] ← layer no.

a_i ← node in layer

Date : .....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

Vectorising again:

$$\begin{bmatrix} - & w_1^{[1]T} & - \\ - & w_2^{[1]T} & - \\ & \vdots & \\ - & w_4^{[1]T} & - \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_4^{[1]} \end{bmatrix}$$

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

$$\therefore Z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ \vdots \\ z_4^{[1]} \end{bmatrix} \qquad a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

Given I/P X: ($X = a^{[0]}$)

$$z^{[1]} = W^{[1]} \cdot a^{[0]} + b^{[1]}$$

(4,1) (4,3) (3,1) (4,1)

$$a^{[1]} = \sigma(z^{[1]})$$

(4,1) (4,1)

$$z^{[2]} = W^{[2]} \cdot a^{[1]} + b^{[2]}$$

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

All this was for a single training example.

training eg. →
hidden units ↓

Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

for $i = 1$ to m,

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$

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

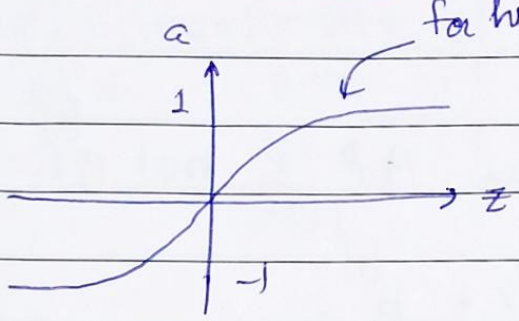
$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \in (n_x, m)$$

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

$$A^{[1]} = \begin{bmatrix} | & | & & | \\ a^{[1](1)} & a^{[1](2)} & \cdots & a^{[1](m)} \\ | & | & & | \end{bmatrix}$$

Date :.....
MON TUE WED THU FRI SAT SUN
Slow learning.

Activation Functions:



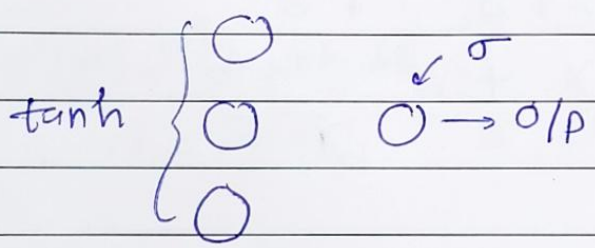
for high z slopes → 0

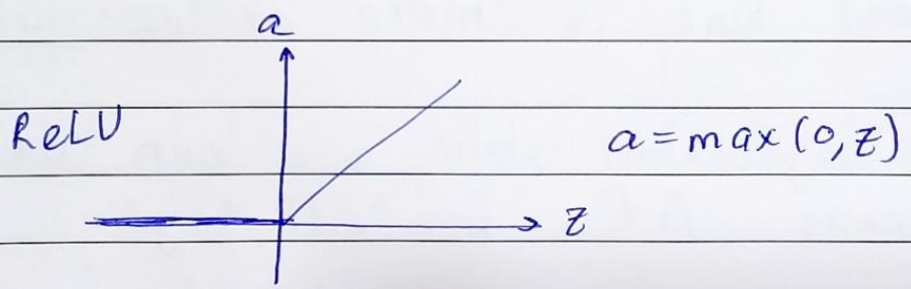
$$a = \tanh(z)$$

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

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

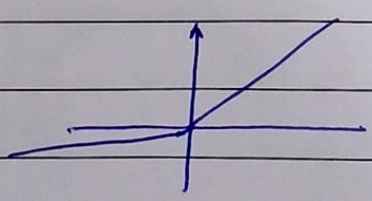
almost
tanh always superior than sigmoid





If binary classif. → in its o/p layer generally σ used.
In all other layers → ReLU can be used.

Try all a.f.



Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

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

↑ Act. func.

Why non-linear A.F & not $g^{[1]}(z) = z$

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

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

$= w^{[2]}(w^{[1]}x + b^{[1]}) + b^{[2]}$

$= \underbrace{(w^{[2]}w^{[1]})}_{w'}x + \underbrace{w^{[2]}b^{[1]} + b^{[2]}}_{b'}$

Linear func. is more or less USELESS

For regression pblm, we can use Linear A.F. for O/P layer.

* ReLU → $g(z) = \max(0, z)$

$$g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \\ \text{undef} & \text{if } z = 0 \end{cases}$$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

* Leaky ReLU:

$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z<0 \\ 1 & \text{if } z>0 \\ \text{undef} & \text{if } z=0 \end{cases}$$

* tanh func: $g(z) = \tanh z = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

$$g'(z) = 1-(\tanh(z))^2$$

$$a = g(z) \qquad g'(z) = 1-a^2$$

* Sigmoid func: $g(z) = \frac{1}{1+e^{-z}}$

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

$$= a(1-a)$$

Gradient descent for NN:

$$dW^{[1]} = \frac{dJ}{dW^{[1]}}, \quad db^{[1]} = \frac{dJ}{db^{[1]}}$$

$$W^{[1]} = W^{[1]} - \alpha\, dW^{[1]}$$

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

Keepdims
- prevents Rank 1 array

Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Back Propagation:

$$dZ^{[2]} = A^{[2]} - Y \qquad Y = [y^{(1)}\ y^{(2)} \dots y^{(m)}]$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

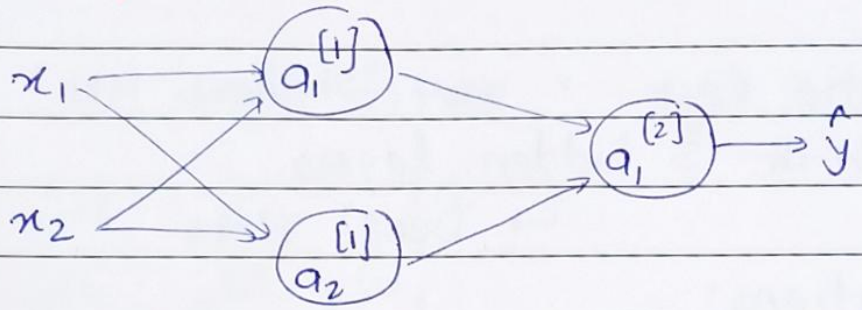
$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis=1, keepdims=True)$$

$$dZ^{[2]} = a^{[2]} - y$$
$$dW^{[2]} = dZ^{[2]} a^{[1]T}$$
$$db^{[2]} = dZ^{[2]}$$
$$dZ^{[1]} = W^{[2]T} dZ^{[2]} \cdot g^{[1]'}(Z^{[1]})$$
$$dW^{[1]} = dZ^{[1]} \cdot x^{T}$$
$$db^{[1]} = dZ^{[1]}$$

$$dZ^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$
$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis=1, keepdims=T)$$
$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^{T}$$
$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis=1, keepdims=T)$$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Why initialise weights randomly?



If $W^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ $\quad b^{[1]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

↳ $a_1^{[1]} = a_2^{[1]}$

Both these units will compute for the same function.

∴

$W^{[1]} = np.random.randn((2,2)) * 0.01$

$b^{[1]} = np.zeros((2,1))$

so that

so that in sigmoid derivative don't → 0 ←

$z^{[1]} = W^{[1]}x + b^{[1]}$

↳ is small

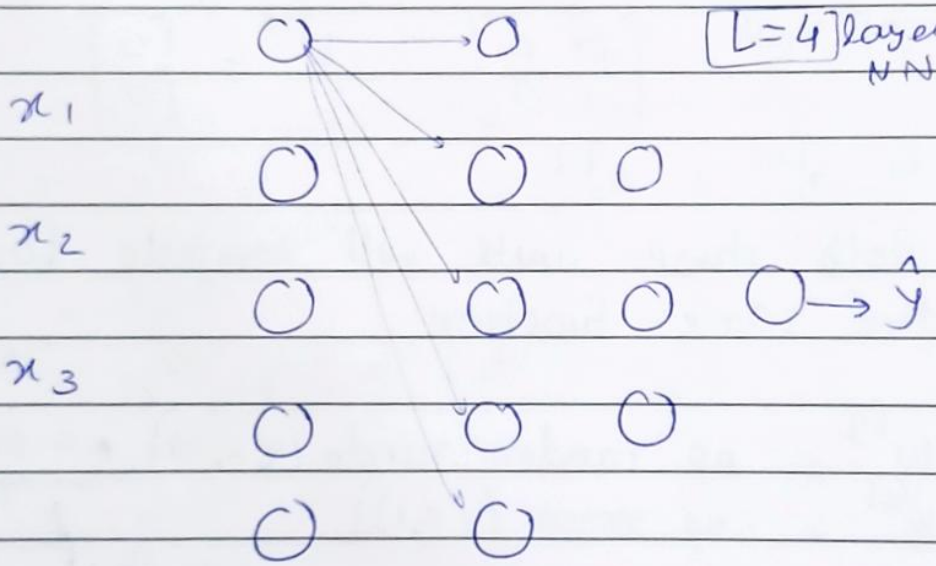
$a^{[1]} = g^{[1]}(z^{[1]})$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Deep Neural Networks

Logistic Regr → very "shallow" NN.
NN with 5 hidden layers
↳ "Deep" NN.

Notations:



$n^{[l]}$ = no. of units in layer l.

$n^{[1]} = 5 \quad n^{[2]} = 5 \quad n^{[3]} = 3$

$n^{[4]} = n^{[L]} = 1 \quad n^{[0]} = n_x = 3$

$a^{[l]}$ = activations in layer l

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

$w^{[l]}$ = weights for $z^{[l]}$

$x = a^{[0]}$

$\hat{y} = a^{[L]}$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Forward Prop in DNN (Deep)

(1) For single tr. ex. → all z, a, b are small.

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

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

⋮

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

(2) Vectorized:

$$Z^{[1]} = W^{[1]} X + b^{[0]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
} For L=1

$$Z^{[2]} = W^{[2]} A^{[1]} + b^{[1]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$
} For L=2

⋮

General:
$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$
$$A^{[l]} = g^{[l]}(Z^{[l]})$$

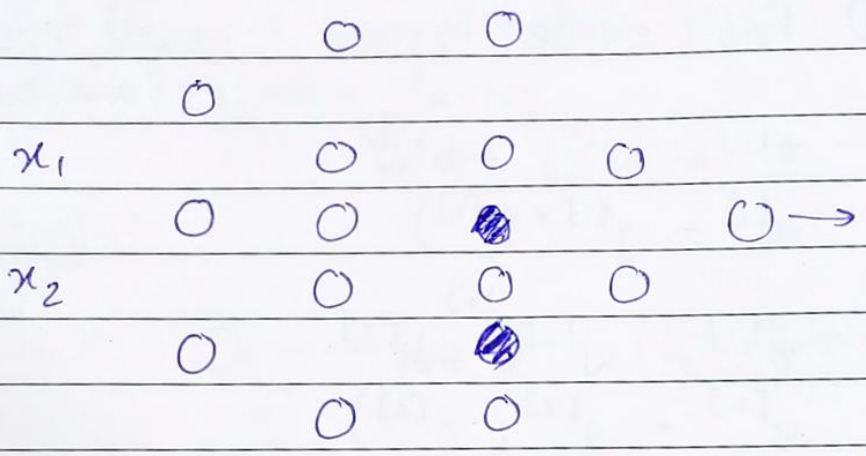
A for loop $l = 1$ to L is necessary //

$Z \rightarrow$ called as vector activations.

Date : .....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Knowing dimensions of Parameters $W^{[l]}$ & $b^{[l]}$:

eg.



L : 0 | 1 | 2 | 3 | 4 | 5

$n^{[0]} = 2$ $n^{[1]} = 3$ - ——

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

$Z^{[1]}$: (3,1) or $(n^{[1]}, 1)$

$W^{[1]}$: (3,2) or $(n^{[1]}, n^{[0]})$

X : (2,1) or $(n^{[0]}, 1)$

$b^{[1]}$: (3,1) or $(n^{[1]}, 1)$

General:

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

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

Z & a same dimension.

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

For Backprop: $dw^{[l]}$: same as $w^{[l]}$
: $(n^{[l]}, n^{[l-1]})$
$db^{[l]}$: " : $(n^{[l]}, 1)$

For vectorized Implementation:

capital $Z^{[1]} = w^{[1]} X + b^{[1]}$

$(n^{[1]}, m)$ $(n^{[1]}, n^{[0]})$ $(n^{[0]}, m)$ $(n^{[1]}, 1)$

↓ broadcasting

$(n^{[1]}, m)$

General: small $z^{[l]}, a^{[l]} : (n^{[l]}, 1)$

caps. $Z^{[l]}, A^{[l]} : (n^{[l]}, m)$
$A^{[0]} = X = (n^{[0]}, m)$

$dZ^{[l]}, dA^{[l]} = (n^{[l]}, m)$

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Forward & Backward functions:

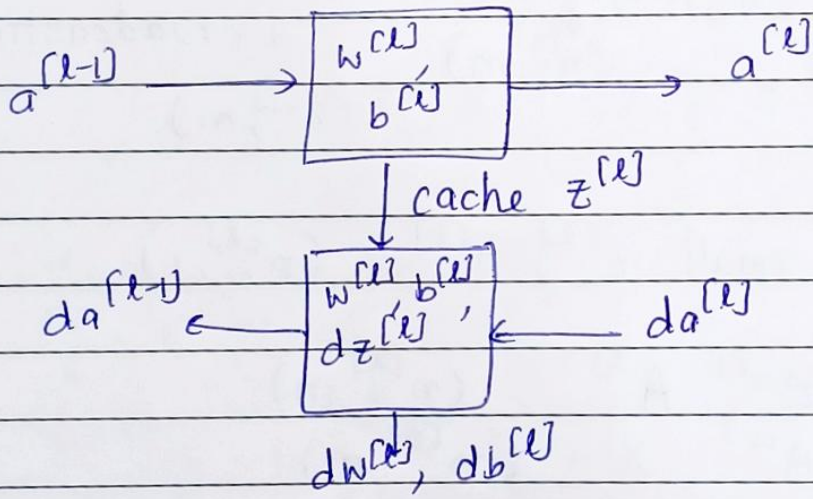
Forward: Layer l:

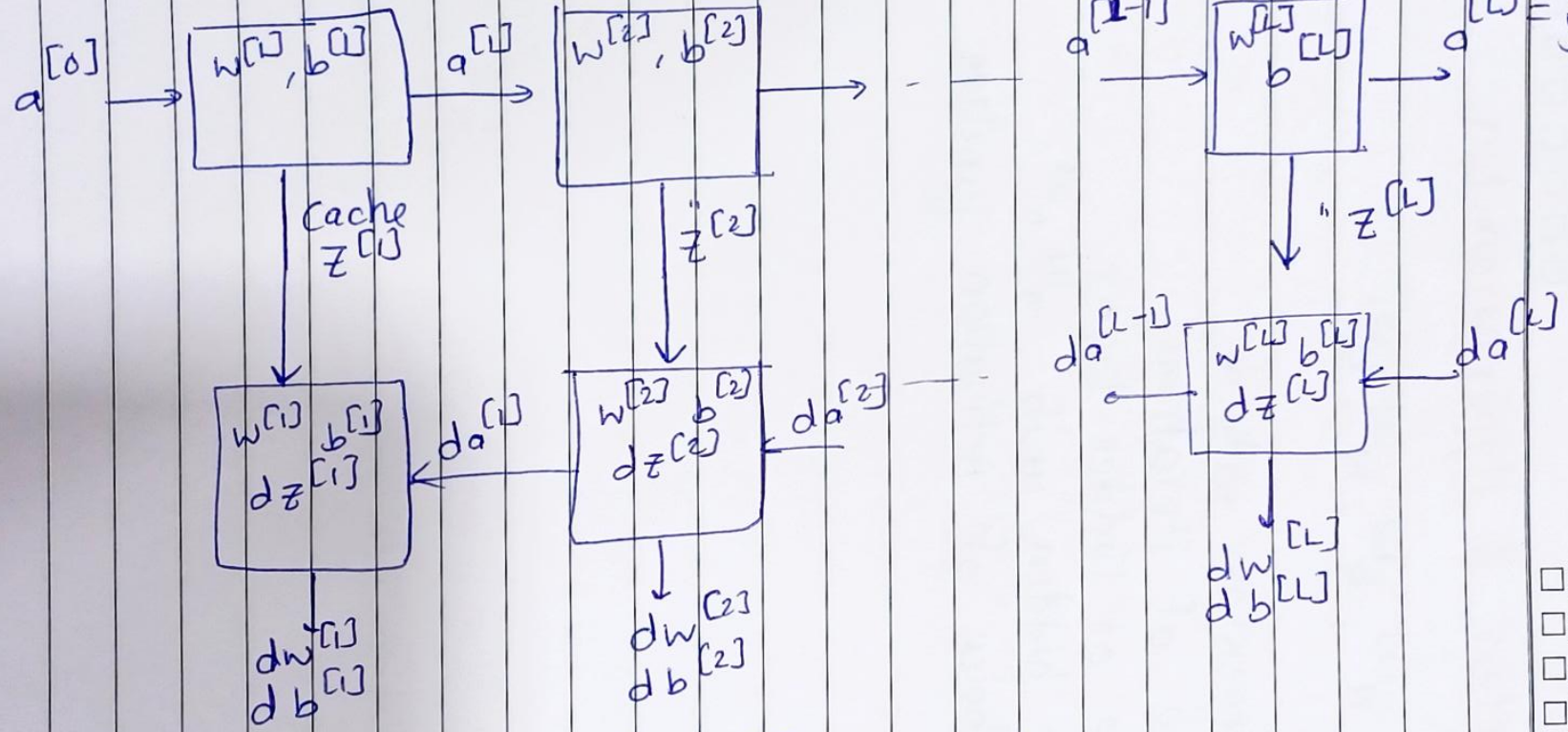
I/P: $a^{[l-1]}$ O/P $a^{[l]}$

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

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

Backward: I/P: $da^{[l]}$ O/P: $da^{[l-1]}$





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

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

Date .....

MON TUE WED THU FRI SAT SUN

☐ ☐ ☐ ☐ ☐ ☐ ☐

Date :.....
MON TUE WED THU FRI SAT SUN
☐ ☐ ☐ ☐ ☐ ☐ ☐

Parameters & Hyperparameters:

$W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]}, \ldots$

Hypar: Learning rate α,
no. of iterations,
no. of hidden layers,
hidden units $n^{[1]}, n^{[2]}, \ldots$
choice of Activation function.