Lab.

Sigmoid functions

RELU

Tan h

Leaky Relu

Softmax

Sigmoid

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

Derivative → $f'(x) = f(x) * (1 - f(x))$

Tan h

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

derivative of tan h(x) $\Rightarrow 1 - \tanh^2(x)$

2 (sigm) − 1 = tan h

Relu → $f(x) = \max(0, x)$. ⇒ Output range = $[0, \infty)$.

Leaky Relu → $f(x) = \begin{cases} x, & x \geq 0 \\ ax, & x \leq 0 \end{cases}$.

$\frac{d}{dx}\left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)$. derivative of tan(x)

$\frac{uv' - u'v}{v^2}$:

$$\Rightarrow \frac{(e^x - e^{-x})(e^x - e^{-x}) - (e^x + e^{-x})(e^x + e^{-x})}{(e^x + e^{-x})^2}$$

$$\Rightarrow \frac{(e^x - e^{-x})^2 - (e^x + e^{-x})^2}{(e^x + e^{-x})^2}$$

$$\Rightarrow \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^x)^2}$$

$$\Rightarrow \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{((e^x + e^{-x})^2}$$

$$\Rightarrow \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

$$\Rightarrow 1 - \tan(-h)^2$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \xrightarrow{\text{Softmax}} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

$$f(x_i) = \frac{e^{x_i}}{\sum_j x_j}$$

Jacobian matrix :

$$\begin{bmatrix} \frac{\partial s_1}{\partial x_1} & \frac{\partial s_1}{\partial x_2} & \frac{\partial s_1}{\partial x_3} \\ \frac{\partial s_2}{\partial x_1} & \frac{\partial s_2}{\partial x_2} & \frac{\partial s_2}{\partial x_3} \\ \frac{\partial s_3}{\partial x_1} & \frac{\partial s_3}{\partial x_2} & \frac{\partial s_3}{\partial x_3} \end{bmatrix} \rightarrow m \times n \text{ matrix}$$

2) $x = \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}$ (a) Compute softmax o/p (b) Compute the Jacobian

$$S_1 = \left[\frac{e^2}{e^2 + e^1 + e^{0.1}} \right] = \frac{7.38}{7.38 + 2.718 + 1.10} \approx 0.68$$

$$S_2 \Rightarrow \frac{e^1}{e^2 + e^1 + e^{0.1}} = \frac{2.718}{11.38} = 0.23$$

$$S_3 \Rightarrow \frac{e^{0.1}}{e^2 + e^1 + e^{0.1}} = \frac{1.10}{} = 0.096$$

(1) $\begin{bmatrix} 0.68 \\ 0.23 \\ 0.096 \end{bmatrix} \cdot \begin{pmatrix} 0.66 \\ 0.24 \\ 0.1 \end{pmatrix}$

softmax is sofisticated sigmoid.

$$\frac{\partial S_1}{\partial x_1} = S_1(1-S_1) \Rightarrow 0.66(0.24) \Rightarrow 0.2244$$

$$\frac{\partial S_1}{\partial x_2} = -S_1 S_2 \Rightarrow 0.66 \times 0.24 \Rightarrow -0.1584$$

$$\frac{\partial S_1}{\partial x_3} = -S_1 S_3 \Rightarrow 0.66 \times 0.1 \Rightarrow -0.06$$

$$\frac{\partial S_2}{\partial x_1} = -S_2 S_1 \Rightarrow -0.1584$$

$$\frac{\partial S_2}{\partial x_2} = S_2(1-S_2) \Rightarrow 0.1824$$

$$\frac{\partial S_2}{\partial x_3} \Rightarrow 0.24 \times 0.1 = -0.024$$

$$\frac{\partial S_3}{\partial x_1} = 0.1 \times 0.66 \Rightarrow -0.06$$

$$\frac{\partial S_3}{\partial x_2} \Rightarrow 0.24 \times 0.1 \Rightarrow -0.024$$

$$\frac{dS_3}{dx_3} \Rightarrow 0.0024$$

$$J_{ij} = \frac{\partial S_i}{\partial x_j}$$

case 1 : $i = j$ (diagonal element).

$$\frac{\partial S_i}{\partial x_i} = S_i(1-S_i)$$

case II = off diagonal element.

$$\frac{dS_i}{dx_j} = -S_i S_j$$

$$\begin{bmatrix} 0.2244 & -0.1584 & -0.06 \\ -0.1584 & 0.1824 & -0.024 \\ -0.06 & -0.024 & 0.09 \end{bmatrix}_{3\times 3}.$$

1) Datasets
2) Dataloaders.

Tensor is an n-dimensional array; we can plug in the datastructures

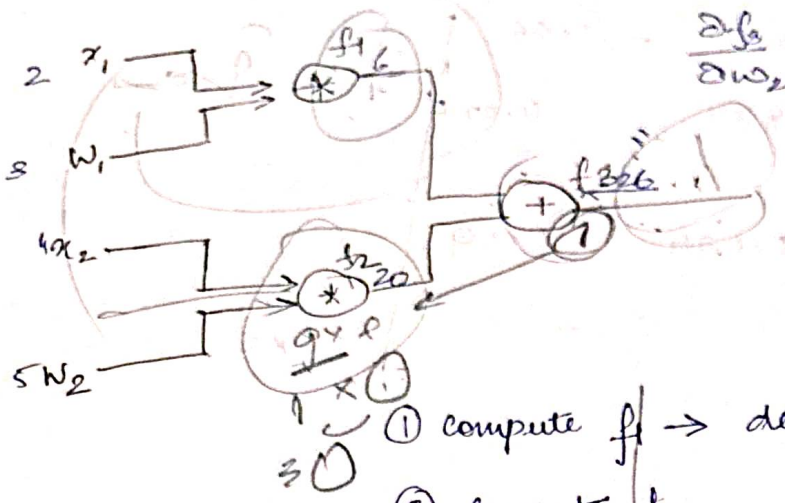
Batches

Epochs → 1 Have seen the dataset once. Higher the number of epochs will increase in learning the datasets.

Forward and backward. One epoch means one pass in the dataset.

Batch size → Hyperparameter.

• Running multiple times do not lead to the overfit.



$\frac{\partial f_3}{\partial w_2} = \frac{\partial f_2}{\partial w_2}$ * global ; $f_1 = w_1 x_1$;

$\frac{df_1}{dx_1} = w_1$; $\frac{\partial f}{\partial w_1} = x_1$;

$\Rightarrow x_2$ * global ;

$f_2 = w_2 x_2$; $\frac{df}{dx_2} = w_2$ $\frac{\partial f}{\partial w_2} = x_2$

$f_3 = f_1 + f_2$ $\frac{df_3}{df_2} = 1$ $\frac{\partial f_3}{\partial f_1} = 1$.

① compute f_1 → def compute $f_1(w_1, x_1)$

② compute f_2 local * global.

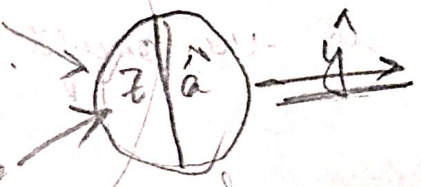
③ compute f_3

④ compute $\frac{df_3}{df_2}$ → def $df_3 f_2$ ( )

Return 1 .

⑤ compute $\frac{\partial f_3}{\partial w_2}$ → def $df_3 w_2$ (local, global)

return local, global



def forward pass()

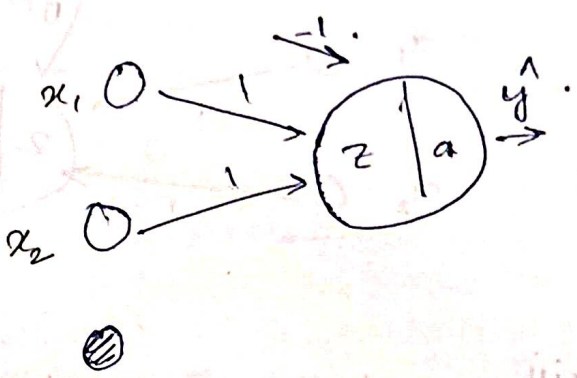
$$a^{[1]} = g(z^{[1]}).$$

$$g = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}.$$

Put this model into logistic regression and minimize the cost entropy loss.

AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



First case = $0 \times 0 + (-1)$

$\Rightarrow -1$.

$g = 0$

$\hat{y} = 0$.

This perceptron works perfectly as AND.

Second = $0 \times 1 - 1$

$\Rightarrow -1$

$\Rightarrow \hat{y} = 0$

Third = $0 - 1$

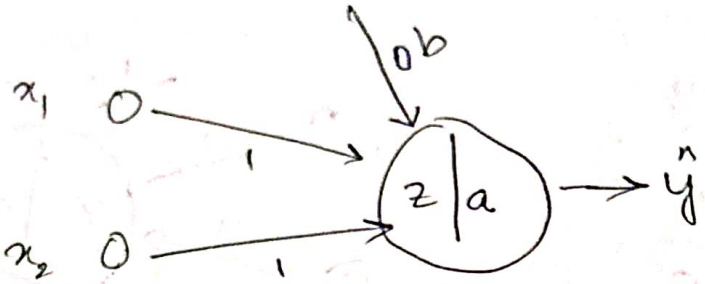
$\hat{y} \Rightarrow 0$

Fourth = $1 + 1 - 1$

$\Rightarrow 1$.

OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



Now; It will work as OR function.

case 1:- 1 + 1 = 2 ⇒ 0 + 0 ⇒ 0.

Case 2 0 + 1 = 1 ⇒ 1

case 3:

1 + 0 + 0 ⇒ 1.

$\hat{y} = 1$.

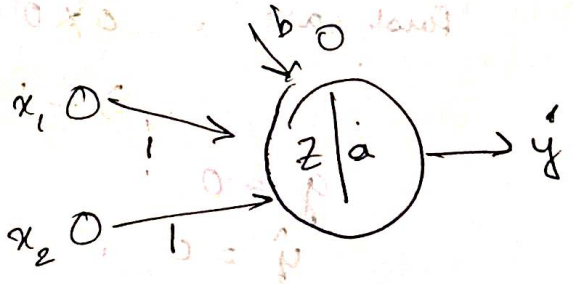
case 4:-

1 + 1 + 0 ⇒ 2

$\hat{y} = 1$.

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

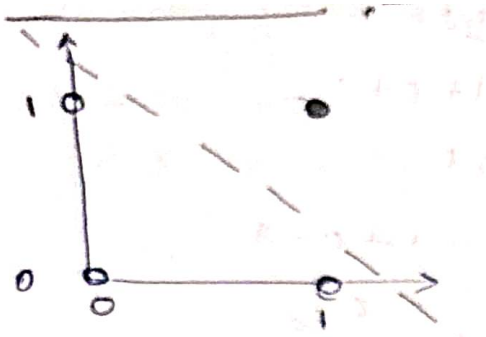


Case 4:- 1 + 1 + 0.

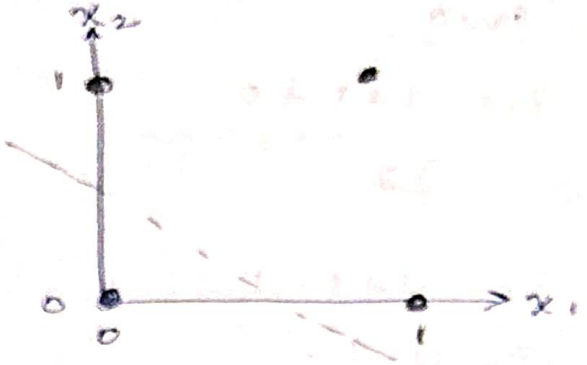
⇒ 2 − 2

$\hat{y} = 1$.

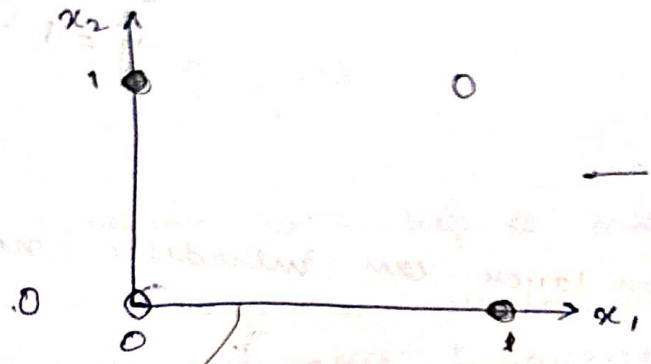
Even if we change bias term it's difficult to satisfy the four conditions to get XOR.



a) x_1 And x_2



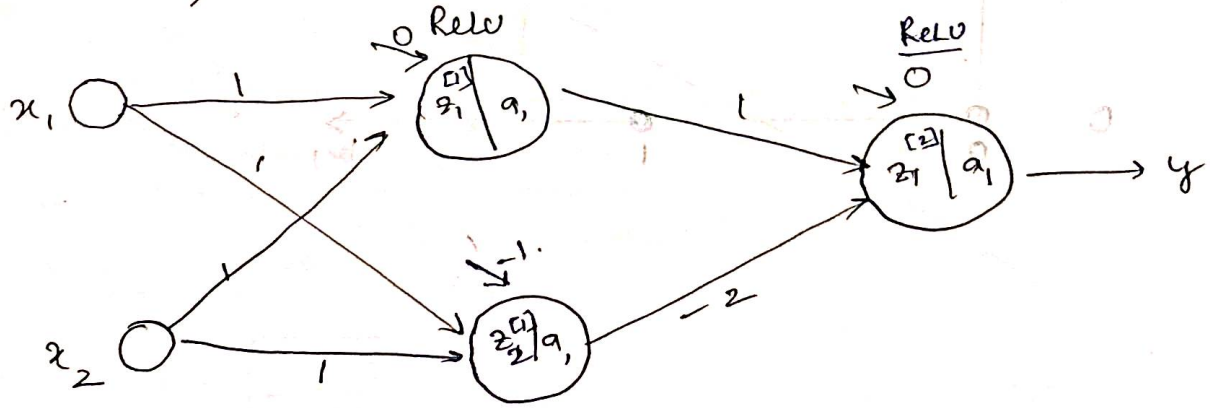
b) x_1 OR x_2 .



a) original space .

c) x_1 XOR x_2 .

It is hard to draw boundary over here as the conditions are not satisfied and perceptron can't act as a XOR ; So we will introduce a hidden layer .



Case 1 :-

$z_1 = 0+0+0$ $0+0$.

$\Rightarrow$ 0 ... $y = 0$

$z_2 = 0+0-1$

$\Rightarrow 0-1$

$\Rightarrow$ $0+2+0$.

$\hat{y} = 2$

Case 1 :-

$z_1 = 1 + 1 + 0$

$\Rightarrow 2$

$z_2 = 1 + 1 - 1$

$z_2 \Rightarrow 1$.

$\Rightarrow 1 \times 2 - 2 \times 1 + 0$.

$\Rightarrow 2 - 2 + 0$

$\hat{y} = 0$.

Case 2 :-

$z_1 = 1 + 0 + 0$

$\Rightarrow 1$

$z_2 = 1 + 0 - 1$

$= 0$

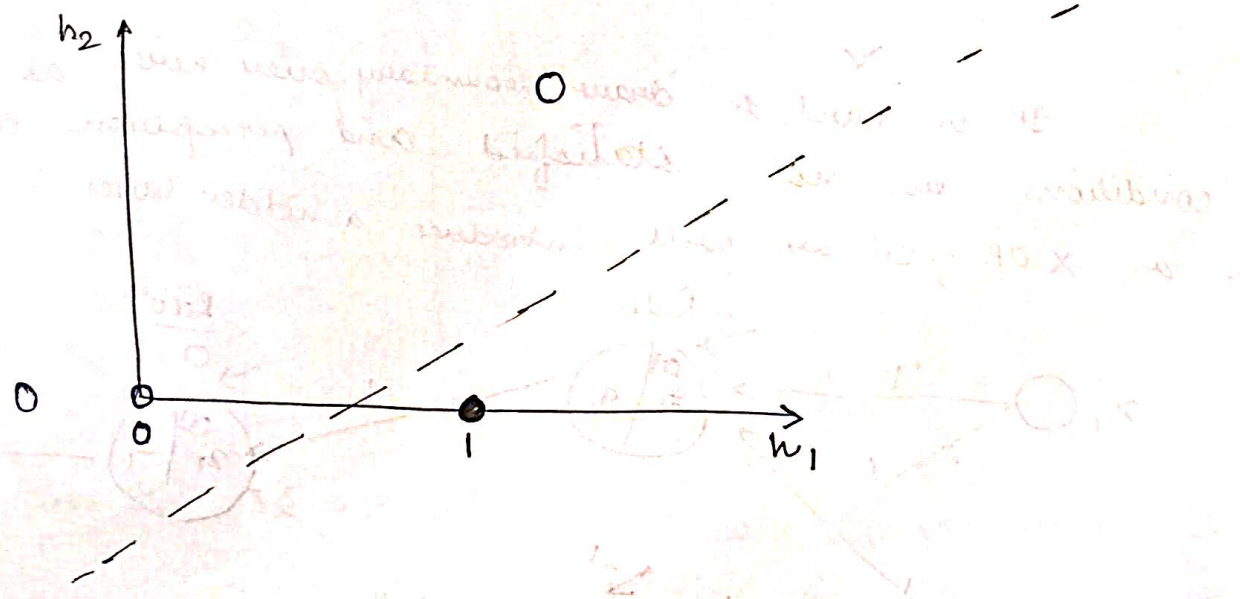
$z_1^{[2]} \Rightarrow 1 \times 1 - 2 \times 0 + 0$

$\Rightarrow 1$.

$\hat{y} = 1$.

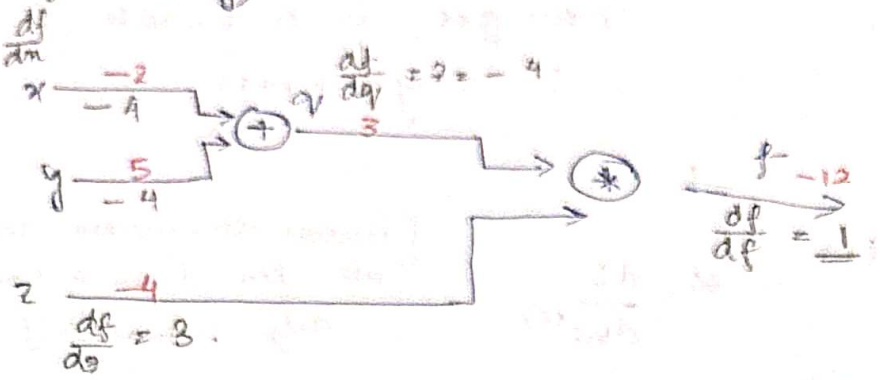
⇒ Introducing a addition layer can introduce non-linearity functions.

(ii) The new (linearly separable) h.space



$f = (x+y)z$

Black one → ~~text~~ number that we get from backward pass.



Let $q = x + y \cdot = \frac{dq}{dx} = 1 \; ; \frac{dq}{dy} = 1$.

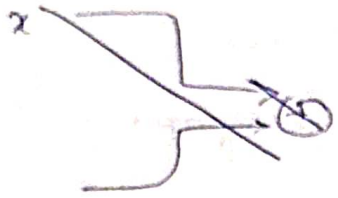
$f = qz \cdot = \frac{df}{dq} \Rightarrow z \qquad \frac{df}{dz} \cdot = q$

$\frac{df}{dx}$ → how much x is influencing x $\int \Rightarrow \underbrace{\frac{dq}{dx}}_{\text{local gradient}} \times \frac{df}{dq} \Rightarrow z \Rightarrow -4$

$\frac{df}{dy} \rightarrow \frac{df}{dq} \times \frac{dq}{dy} \Rightarrow -4 \times 1 \cdot \Rightarrow -4$

↓ global gradient.

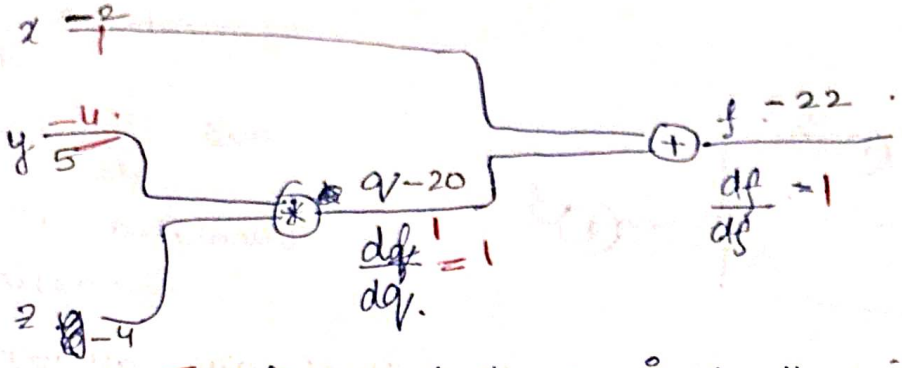
Q)



a) $f = x + q$

$q = yz$

b)



~~5~~ → for a unit change in z there is an output change in f which is 5 times.

$\frac{df}{df} = 1$.

$q = x + yz$.

$\frac{dq}{dx} = 1$

$\frac{dq}{dy} \Rightarrow z$.

$q = yz$

$\frac{dq}{dy} = z$ $\frac{dq}{dz} = y$

$f = x + q$.

$\frac{df}{dx} = 1$

$\frac{df}{dq} = 1$.

⇒ local gradient x global gradient.

$\frac{df}{dz} \Rightarrow \frac{dq}{dz} \times \frac{df}{dq}$ $\frac{df}{dy} \Rightarrow \frac{dq}{dy} \times \frac{df}{dq}$

⇒ $q \times 1$ ⇒ $z \times 1$

⇒ ~~5~~ 5 ⇒ -4.

$$\frac{df}{dz} = \frac{dq}{dx} \times \frac{df}{dq}$$

$\Rightarrow 1 \times 1$

Q) $f(w, x) = \dfrac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$

$w_0 = 2 \quad x_0 = -1 \quad w_1 = 3 \quad x_1 = -2 \quad w_2 = -3.$

$f_1 = w_0 x_0$

$f_2 = w_1 x_1$

$\Rightarrow \dfrac{1}{1} + \dfrac{1}{e^{-f_1}} + \dfrac{1}{e^{-f_2}} \quad \dfrac{1}{e^{-f_3}}.$

$f_3 = f_1 + f_2 + w_2.$

$f_4 = f_3 + w_2.$

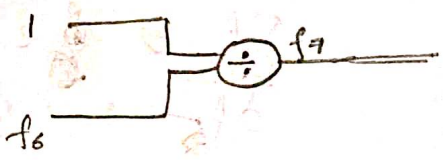
$f_5 = -f_4.$

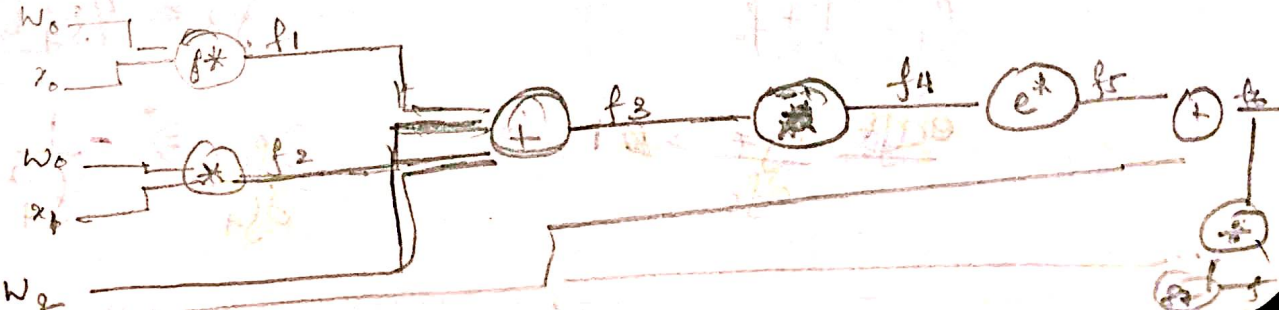
$f_6 = e^{f_5}.$

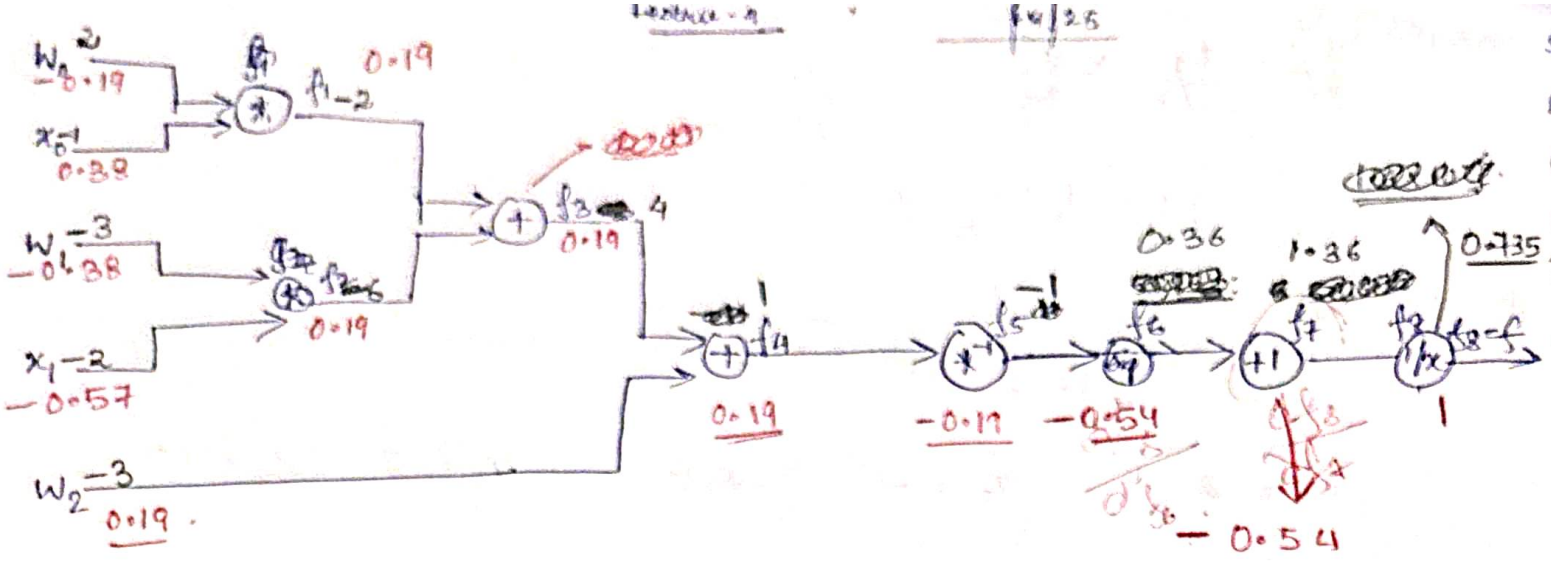
$f_7 = 1 + f_6.$

$f_8 = 1/f_7.$

$\boxed{f = f_7.}$







$$\frac{\delta f_8}{\partial f_8} = 1$$

$f_1 = w_0 x_0$.

$\frac{df_1}{dw_0} = x_0 \quad \frac{df_1}{\partial x_0} = w_0$.

$f_2 = w_1 x_1$.

$\frac{\partial f_2}{\partial w_1} \Rightarrow x_1 \quad \frac{\partial f_2}{\partial x_0} = w_1$.

$f_3 = f_1 + f_2$.

$\frac{\partial f_3}{\partial f_1} = 1 \quad \frac{\delta f_3}{\partial f_2} = 1$

$f_4 = f_3 + w_2$

$\frac{\partial f_4}{\partial f_3} = 1 \quad \frac{\delta f_4}{\partial w_2} = 1$

$f_5 = -f_4$.

$\frac{\partial f_5}{\partial f_4} = -1$.

$f_6 = e^{f_5}$

$\frac{\delta f_6}{\partial f_5} \Rightarrow e^{f_5}$.

$f_7 = 1 + f_6$.

$\frac{\partial f_7}{\partial f_6} = 1$

$f_8 = 1/f_7$.

$\frac{\partial f_8}{\partial f_7} \Rightarrow -\frac{1}{f_7^2}$.

$\frac{df_8}{df_6} \Rightarrow \frac{df_7}{df_6} \times$

$\frac{df_6}{df_7} \Rightarrow \frac{df_6}{df_5} \times \frac{df_5}{df_7}$

$\Rightarrow 1 \times f_5 e^{f_5}$

$\Rightarrow 1 \times (-1) e^{-1}$

$\Rightarrow -0.36$

$\frac{df_5}{df_6} \Rightarrow \frac{df_5}{df_4} \times \frac{df_4}{df}$

$\frac{df_4}{df_5} \Rightarrow \frac{df_5}{df_4} \Rightarrow \frac{df_5}{df_5} \times d$

$\frac{df_8}{df_7} = \text{local} * \text{global}$

$\frac{df_8}{df_7} \Rightarrow -\frac{1}{f_7^2} * 1$

$\Rightarrow \frac{-1}{(1.36)^2} \times 1$

$\Rightarrow -0.054$

$\frac{df_8}{df_5} = \text{local} * \text{global}$

$\frac{df_8}{df_5} \Rightarrow \frac{df_6}{df_5} \times -0.54$

$\Rightarrow 0.34 \times 0.54$

$\Rightarrow -0.19$

$\frac{df_8}{df_6} = \text{local} \times \text{global}$

$\frac{df_8}{df_6} = \frac{df_7}{df_6} \times \frac{df_8}{df_7}$

$\Rightarrow 1 \times -0.054$

$\Rightarrow -0.054$

$\frac{df_8}{df_4} \Rightarrow \text{local} \times \text{global}$

$\frac{df_8}{df_4} \Rightarrow \frac{df_5}{df_4} \times -0.19$

$\Rightarrow -1 \times (-0.19)$

$\Rightarrow 0.19$

$\frac{df_8}{df_{7\&9}}$ = local x global

1×0.19

$\Rightarrow 0.19$

$\frac{df_8}{df_3} = 1 \times 0.19$

$\Rightarrow \underline{0.19}$.

$\frac{df_8}{df_2} = 1 \times 0.19$

$\Rightarrow 0.19$

$\frac{df_8}{df_1} = 1 \times 0.19$

$\Rightarrow 0.19$

$\frac{df_8}{dw_0} \Rightarrow x_0 \times 0.19$

$\Rightarrow -0.19$

$\frac{df_8}{dx_0} \Rightarrow w_0 \times 0.19$

$\Rightarrow 2 \times 0.19$

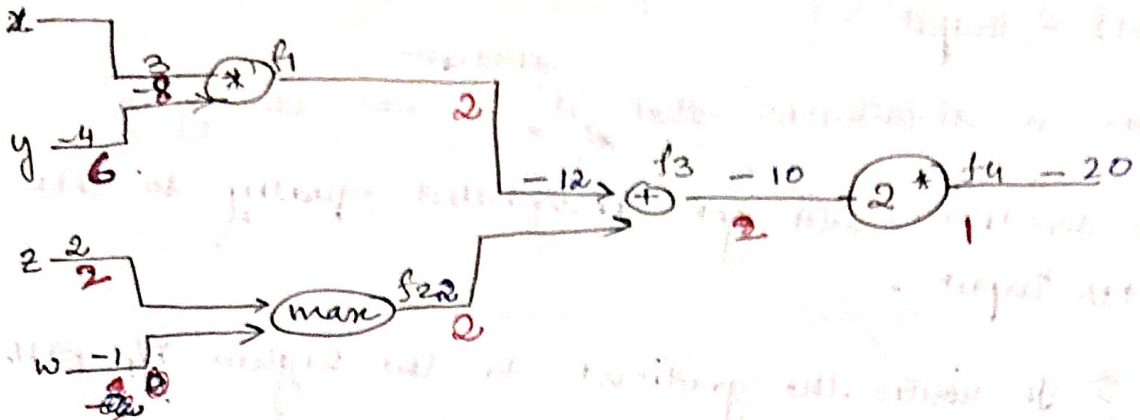
$\Rightarrow 0.38$

$\frac{df_8}{dw_1} \Rightarrow -2 \times 0.19$

$\Rightarrow -0.38$.

$\frac{df_8}{dx_1} \Rightarrow -3 \times 0.19$

$\Rightarrow -0.57$



$f_1 = x * y$

$\frac{df_1}{dx} \Rightarrow y \quad \frac{df_1}{dy} = x$

$f_1 + f_2$

$f_3 \Rightarrow$ ~~max~~ f_2 . ~~max(f1,f2)~~

$\frac{df_3}{df_1} \Rightarrow 1 \quad \frac{df_3}{df_2} = 1$

$f_4 \Rightarrow 2 * f_3$.

$f_2 = z \max w / \max(z, w)$ $\quad \frac{df_4}{df_3} \Rightarrow 2$.

$\frac{df_2}{dz} = 1 \quad \frac{df_2}{w} = 1$

$\frac{df_4}{df_4} = 1$.

$\frac{df_2}{dz} = 1$ if $z > w$
$\Rightarrow 0$ if $z < w$
$\Rightarrow$ undef. if $z = w$

$\frac{df_4}{df_2} \Rightarrow 2 \times 1$.

$\frac{df_4}{df_3} \Rightarrow 2$.

$\frac{df_1}{df_4} \Rightarrow$ local x global .

$\Rightarrow \quad \times 1$

$\frac{df_4}{dx} \Rightarrow y \times 2$.

$\Rightarrow -4 \times 2$

-8

$\frac{df_2}{dw} = 1$ if $w > 2$

$\Rightarrow 0$ if $w < 2$

$\Rightarrow$ undef or arbitrary $\quad w = 2$.

$\frac{df_4}{df_y} \Rightarrow 3 \times 2$

$\Rightarrow 6$

$z_1 = 1 \times 0.5 \Rightarrow 0.5$

$a_1 = \sigma(z_1) = 0.622.$

$z_2 \Rightarrow 0.622 \times 0.5$

$\Rightarrow 0.3110$

$a_2 \Rightarrow \sigma(z_2) \Rightarrow 0.574$

$\Rightarrow$ ~~$\times 0.3110$~~

$z_3 \Rightarrow 0.574 \times 0.5$

≈ 0.2885

$a_3 \Rightarrow \sigma(z_3)$

$\Rightarrow 0.571.$

Loss $\frac{1}{2}(a_3 - y)^2$

$\Rightarrow \frac{1}{2}(0.572 - 1)^2$

≈ 0.0915

≈ 0.0912

$\frac{dL}{da_3} \wedge \frac{da}{dz_3}$

$\frac{da}{dz_3} \Rightarrow g(z)(1 - g(z))$

$\Rightarrow 0.082$

$\frac{dL}{da_3} \Rightarrow a_3 - y = 0.572 - 1$

$\Rightarrow -0.428.$

Layer; L_3 ; Gradient $= -0.105$

Layer L_2 ; Gradient $\Rightarrow -0.013$

Layer; L_1 ; Gradient $\Rightarrow -0.0015.$

for 1024 pixels we need to train the

$$W = 3 \times 1000$$

$$\Rightarrow 8 \text{ million}$$

Convolution operation

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

1	0	-1
1	0	-1
1	0	-1

↓ convolution operated ↓

input = 6×6 image : filter or kernel 3×3

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-6

$2+0-4+9+0-1+5+0$

$-3 = 8$

$3\times1 + 0\times0 + 1\times-1 + 1\times1 + 5\times0 + 8\times-1 + 2\times1 + 7\times0 + 2\times-1$

$\Rightarrow -5$

$0 + 0 + -2 + 9 + 7 + 0 + -6$

$\Rightarrow -4$

$\Rightarrow 1 + 0 + -7 + 8 + 0 - 3 + 2 - 0 - 1$

$= 0$

→ output will be shrinked and less visibility of the pixels.

→ If we want attention to put as some middle pixels or want to miss some pixels then we can skip the layer and use the strided.

→ we can use padding; if we want to more attention at the edges.

$\Rightarrow \frac{n-f}{s}+1$

Q)

2	3	7	4	6
6	6	9	8	7
3	4	8	3	8
7	8	3	6	6
4	2	1	8	3

$n = 5 \times 5$

$p = 1 \; ; s = 2$

$*$

1	-1
0	-1

$=$

$f = 2 \times 2$.

$n+2p-f+1$.

$\Rightarrow n+2-2+1$.

$\Rightarrow \frac{6}{2} \Rightarrow$

$\frac{n+2p-f+1}{5+2-2+1-2}$

$\Rightarrow 3$.

$=$

1	-1
0	-1

0	0	0	0	0	0	0
0	2	3	7	4	6	0
0	6	6	9	8	7	0
0	3	4	8	3	8	0
0	7	8	3	6	6	0
0	4	2	1	8	3	0
0	0	0	0	0	0	0

-2	-7	-6
-9	-11	-7
-11	4	-3

3×3.

$= 0+0+0-2$

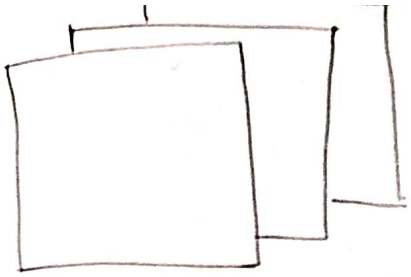
$= 0+0+0$

$= 0-6-3$

$\Rightarrow 6-9+0-8$

$\Rightarrow 6-15$

$\Rightarrow 8-7+0-8$



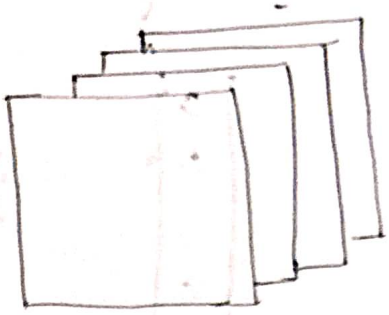
39 x 39 x 3

$f[1] = 3$

$s[1] = 1$

$p[1] = 0$

10 filters



37 x 37 x 10

The size will decrease as padding is zero.

so the new size will become 37 x 37 x 3.

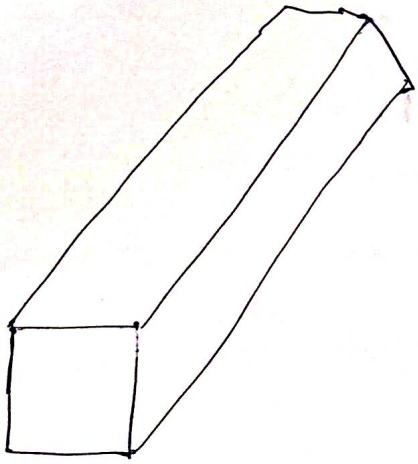
$f[2] = 5$

$s[2] = 2$

$p[1] = 0$

20 filters.

- After applying the convet the size get decreases.
  No. of channels depend on filters

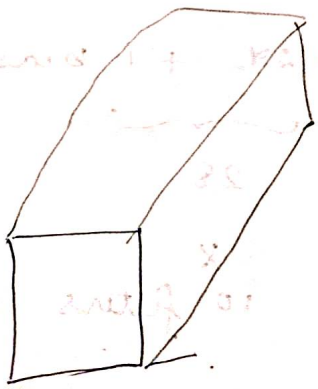


$f[3] = 5$

$s[3] = 2$

$p[1] = 0$

40 filters.



17 x 17 x 20

$\frac{7 \times 7 \times 40}{2} = 1960$ pixels

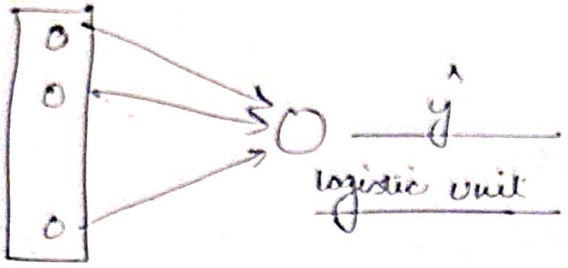
PTO.

formula.

$\Rightarrow \left[\frac{n+2p-f}{s} + 1\right]$

$\Rightarrow \left[\frac{37-5}{2} + 1\right]$

→ Apply a linear layer



I/P → CONV - CONV - CONV - FC].

Discriminatory features :- 19/8/25

We have 10 filters each of size 3×3×3 in one layer of convnet.

How many parameters does this layer have ?

~~9×3 ⇒ 270~~.

3×3×3 → 27 + 1 bias.

28

×

10 filters

⇒ 280 parameters.

If we know that $a[l]$ is better than the output will be $a[l]$.

$$a^{[l+2]} = g\left(w_a^{[l+2][l+1]} + b^{[l+2]} + a^{[l]}\right)$$

$$\approx g\left(z^{[l+2]} + a^{[l]}\right)$$

$$g(256 + 128)$$

$$a^{[l+2]} = \text{ReLU}\left(z^{[l+2]} + a^{[l]}\right)$$

$$32\times1 = 32\times1 + [W] \rightarrow 64\times1$$

↓

This should be $\frac{32}{64}$.

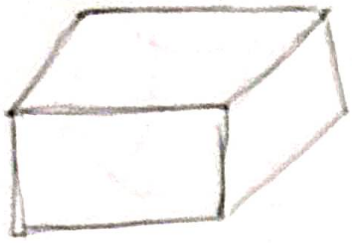
- ResNet can do deeper deeper.

ResNet - 16

" - 32

a) First use the existing resnet and see and then transform the images.

Q)

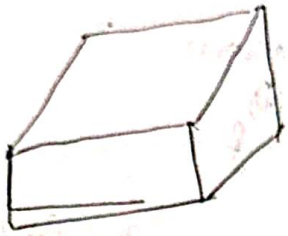


28 × 28 × 192

CONV
1 × 1
× 16 filters

28 × 28 × 16

CONV
5 × 5
same
32 filters.



→ For 1st layer

$1 \times 1 \times 192 = 192$.

$192 \times 16 = 3072$.

$28 \times 28 = 784$.

~~3,80,64~~ 24,08,448.

For 2nd layer ⇒

2) $5 \times 5 \times 16 = 400$.

$400 \times 32 = 12,800$

~~12,46,88~~.

⇒ $784 = 1,00,35,200$

∴ Total = 1,00,35,200 + 2408448.

2) 12,44,36,48 × 12 million.

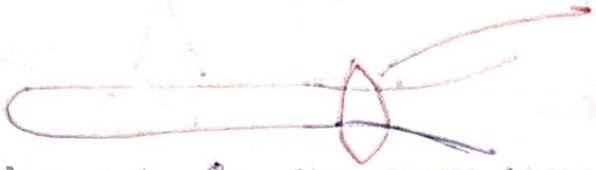
If we need to ① modify Height, width.

→ Padding

→ Striding, Pooling.

② Modify Channels → filters, 1 × 1 CONV NW.

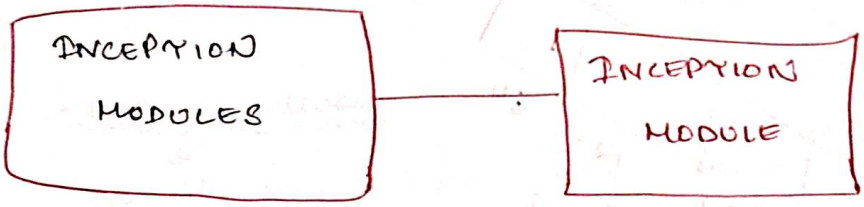
The layers are termed as bottleneck layer.

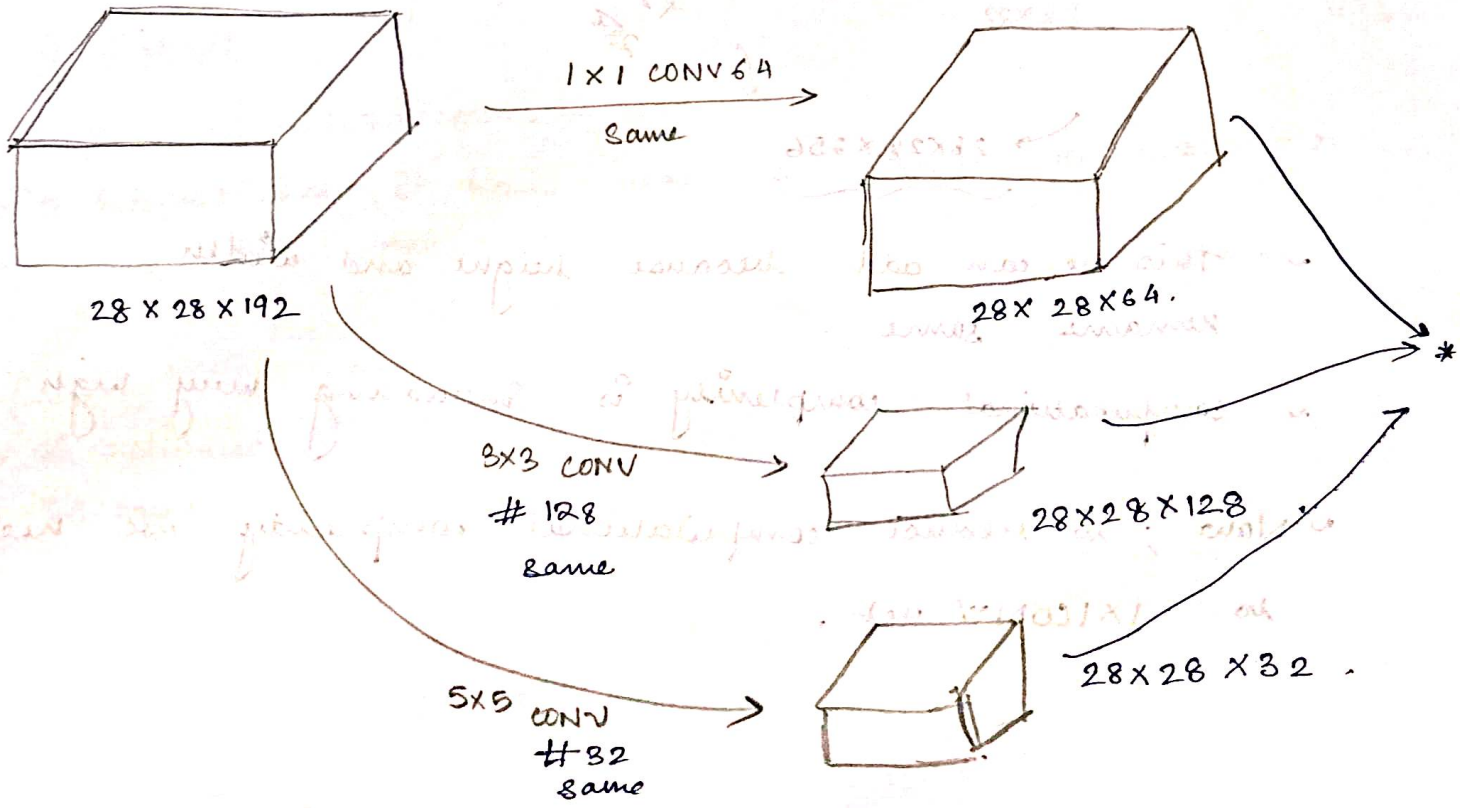


- Relu is included in the CONV layer by default but if we needed we can remove them.

- Relu is important in the conv net, Res net or feed neural network it is important to be there for the non-linearity to be put on them.

INCEPTION NETWORK :-





y^t and x_t)

→ It will be shared across the timesteps

→ Input is of varying length, then the input size is different.

→ Output is there that input is dimension that is ~~size~~ sharing of weights.

Vanilla RNN → to handle variable length

2017/9/25

$\tau = 2$ (two time steps).

$x^{(1)}$: $x_1 \in \mathbb{R}^2$.

$$= \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$x_2 \in \mathbb{R}^2$

$$= \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

input dimensions = 2.

$\Rightarrow x_t \in \mathbb{R}^2$.

hidden state dim = 3 $\Rightarrow h_t \in \mathbb{R}^3$.

$h_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ 3×1

3×1 × 1×2

$W_{xh} = \begin{bmatrix} & \\ & \\ & \end{bmatrix}$ 3×2

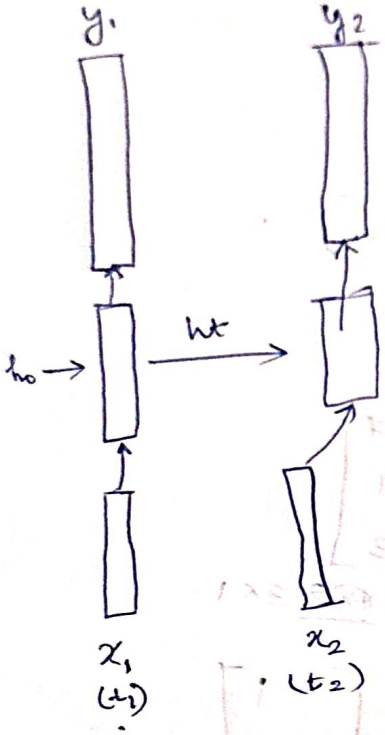
$W_{hh_2} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ 3×1 × 1×3

3×3

$W_{hy} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}$ 3×3

$W_{xh} =$ $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

$W_{hh} = \begin{bmatrix} 0.5 & -0.3 \\ 0.8 & 0.2 \\ 0.1 & 0.4 \end{bmatrix}_{3\times 2}$ $W_{hh} = \begin{bmatrix} 0.1 & 0.4 & 0.0 \\ -0.2 & 0.3 & 0.2 \\ 0.05 & -0.1 & 0.2 \end{bmatrix}_{3\times 3}$ $W_{hy} = \begin{bmatrix} 1.0 & -1.0 \\ 0.5 & 0.5 \end{bmatrix}_{2\times 3}$



$h_t = \tanh\left(W_{hh}\, h_{t-1} + W_{xh}\, x_t\right)$

$y_t = W_{hy}\, x\text{-}h_t$

At timestep = 1; $(t=1)$

$\rightarrow h_t \Rightarrow \tanh($

at x_1 $(t=1)$ $h_{(1-1)} \rightarrow h_0$

$W_{hh} = \begin{bmatrix} 0.1 & 0.4 & 0.0 \\ -0.2 & 0.3 & 0.2 \\ 0.05 & -0.1 & 0.2 \end{bmatrix}_{3\times 3} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{3\times 1}$

$h_y \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ $\left.\begin{matrix} - \\ 0 \\ h_1 \end{matrix}\right\}$

~~$h_t = \tanh(0 +$~~

$\begin{bmatrix} -0.1 \\ 0.2 \\ 0.9 \end{bmatrix} = \begin{bmatrix} 0.5 - 0.6 \\ 0.8 + 0.4 \\ 0.1 + 0.8 \end{bmatrix}$ $\begin{bmatrix} 0.5 & -0.3 \\ 0.8 & 0.2 \\ 0.1 & 0.4 \end{bmatrix}_{3\times 2} \begin{bmatrix} 1 \\ 2 \end{bmatrix}_{2\times 1}$

$$\tanh\left(\begin{bmatrix}0\\0\\0\end{bmatrix} + \begin{bmatrix}-0.1\\1.2\\0.9\end{bmatrix}\right)$$

$$= \tanh\begin{bmatrix}-0.1\\1.2\\0.9\end{bmatrix}$$

For $\tanh(-0.1) = -0.0997$

$\tanh(1.2) = 0.8337$

$\tanh(0.9) \approx 0.7163$

$$y = \begin{bmatrix}-0.0997\\0.8337\\0.7163\end{bmatrix}_{3\times1} \begin{bmatrix}1.0 & -1.0 & 0.5\\0.5 & 0.5 & -0.5\end{bmatrix}_{2\times3}$$

At timestep ($t = 2$)

$$h_2 = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$h_2 = \tanh(W_{hh} h_1 + W_{xh} x_2)$$

$$h_2 = \tanh\left(\begin{bmatrix}0.1 & 0.4 & 0.0\\-0.2 & 0.3 & 0.2\\0.05 & -0.1 & 0.2\end{bmatrix}_{3\times3} \begin{bmatrix}-0.0997\\0.8337\\0.7163\end{bmatrix}_{3\times1} + \begin{bmatrix}0.5 & -0.3\\0.8 & 0.2\\0.1 & 0.4\end{bmatrix}_{3\times2} \begin{bmatrix}-1\\1\end{bmatrix}_{2\times1}\right)$$

$$h_2 = \tanh\left(\begin{bmatrix}\cdot\\\cdot\\\cdot\end{bmatrix} + \begin{bmatrix}0.\cdot\\\cdot\\\cdot\end{bmatrix}\right)$$

$$\Rightarrow \tanh\left(\begin{bmatrix} -0.0997 + 0.33 + 0 \\ 0.0199 + 0.25011 + 0.14326 \\ -0.0049 - 0.8337 + 0.14326 \end{bmatrix} + \begin{bmatrix} -0.5 - 0.3 \\ -0.8 + 0.2 \\ -0.1 + 0.4 \end{bmatrix}\right)$$

$$\tanh\left(\begin{bmatrix} 0.2383 \\ 0.4129 \\ -0.69534 \end{bmatrix} + \begin{bmatrix} -0.8 \\ -0.6 \\ 0.3 \end{bmatrix}\right)$$

$$\tanh\begin{pmatrix} -0.5617 \\ -0.1871 \\ -0.395 \end{pmatrix}$$

$$= \begin{pmatrix} -0.5090 \\ -0.1849 \\ -0.3757 \end{pmatrix}.$$

~~2 ie delay~~ -

$y_1 = W_{hy} \times h_{(1)}$ .. $\rightarrow y_1 \begin{bmatrix} -0.57 \\ 0.008 \end{bmatrix}$.

$$\Rightarrow \begin{bmatrix} 1.0 & -1.0 & 0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 0.0997 \\ 0.8337 \\ 0.7163 \end{bmatrix}$$

2×3 $\quad$ 3×1

$$\Rightarrow \begin{bmatrix} 0.0997 - 0.8337 + 0.35815 \\ 0.0498 + 0.4168 - 0.3581 \end{bmatrix}$$

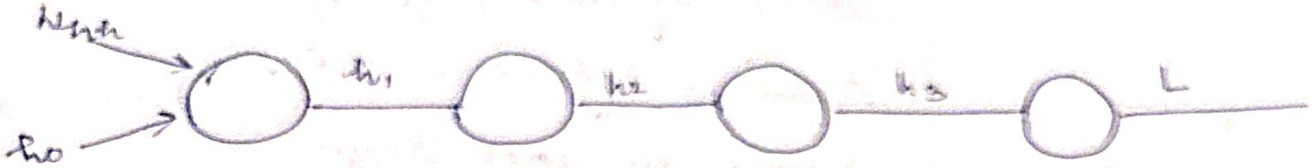
2×1

$$= \begin{bmatrix} -0.37 \\ 0.1085 \end{bmatrix}.$$

Vanishing Gradient Problem :-

↓

Vanilla RNN or multilayer RNN will lead to Vanishing Gradient Problem.



$$\frac{\partial L}{\partial W_{hh}} = \frac{\partial L}{\partial h_3} \cdot \frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1} \frac{\partial L_1}{\partial W_{hh}}$$

$$\Rightarrow \frac{\partial L}{\partial h_3}\left(\frac{\partial h_3}{\partial h_2} \cdot \frac{\partial h_2}{\partial h_1}\right)\frac{\partial h_1}{\partial W_{hh}}$$

$$= \frac{\partial L}{\partial h_3}\left(\prod_{t=2}^{T=3} \frac{\partial h_t}{\partial h_{t-1}}\right)\frac{\partial h_1}{\partial W_{hh}}$$

for T time steps.

$$\frac{\partial L}{\partial W_{hh}} = \frac{\partial L}{\partial h_t}\left(\sum_{t}^{T} \frac{\partial h_t}{\partial h_{t-1}}\right)\frac{\partial h_1}{\partial W_{hh}}$$

$$h_t = \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t\right)$$

→ this is gradient because it deals with multivariable class or vector derivative ≈ single layer.

$$\frac{\partial h_t}{\partial h_{t-1}} = \tanh'\left(W_{hh} h_{t-1} + W_{xh} x_t\right) W_{hh}$$

↓

we need to calculate derivative.

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

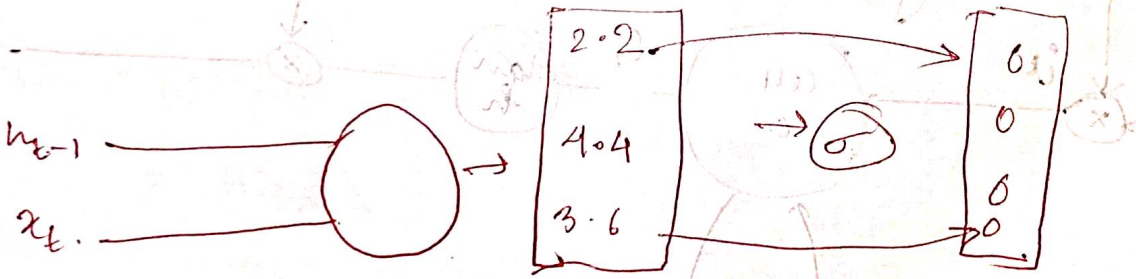
$$f_t = \sigma(W_{hf} h_{t-1} + W_{xf} x_t)$$

$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t)$$

$$o_t = \sigma(W_{ho} h_{t-1} + W_o x_t)$$

$$g_t = \tanh(W_{hg} h_{t-1} + W_{xg} x_t).$$

These gates are sigmoid hence a range between (0,1) that's why they are known as smooth gate.



$$i_t = \sigma(W_{hi} h_{t-1} + W_{xi} x_t)$$

Because h is input and i depends on i_t.

Previous i/p

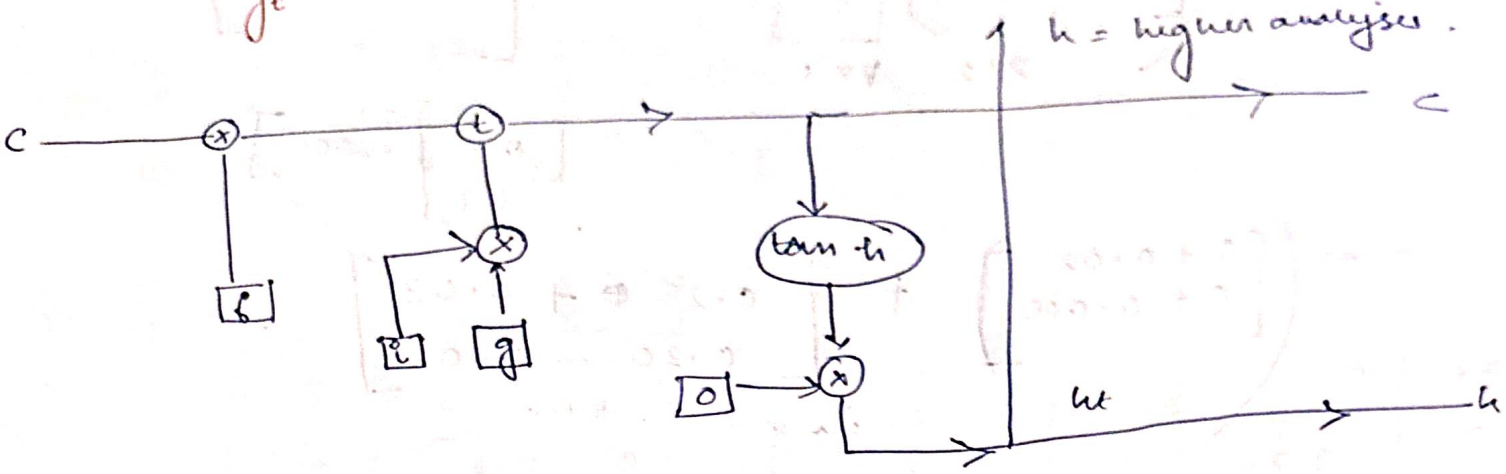
Present output.

→ it is a vector whose values (0, 1).

g_t → another gate.

$$g_t = \tanh(W_{hg} h_{t-1} + W_{xg} x_t).$$

i_t and g_t will be element wise operation.



We use LSTM's (a) to avoid vanishing gradient. (not 100% but minimizes the problem)

Example:

$x_t = \begin{bmatrix} 0.5, & -0.1 \end{bmatrix}^T$ $\quad R^2$

$h_{t-1} = \begin{bmatrix} 0.0, & 0.1 \end{bmatrix}^T$ $\quad R^2$

$C_{t-1} = \begin{bmatrix} 0.2, & -0.2 \end{bmatrix}^T$ $\quad R^2$

$$W_{xi} = \begin{bmatrix} 0.5 & -0.3 \\ 0.4 & 0.1 \end{bmatrix}$$

$$W_{hi} = \begin{bmatrix} 0.1 & 0.2 \\ -0.2 & 0.05 \end{bmatrix}$$

$$W_{xf} = \begin{bmatrix} -0.4 & 0.2 \\ 0.3 & 0.3 \end{bmatrix}$$

$$W_{hf} = \begin{bmatrix} 0.05 & -0.1 \\ 0.2 & 0.1 \end{bmatrix}$$

$$W_{xo} = \begin{bmatrix} 0.3 & 0.25 \\ -0.2 & 0.2 \end{bmatrix}$$

$$W_{ho} \Rightarrow \begin{bmatrix} 0.15 & 0.05 \\ 0.1 & -0.2 \end{bmatrix}$$

$$W_{xg} \Rightarrow \begin{bmatrix} -0.5 & 0.4 \\ 0.2 & -0.3 \end{bmatrix}$$

$$W_{hg} = \begin{bmatrix} 0.2 & 0.1 \\ -0.1 & 0.05 \end{bmatrix}$$

$$\rightarrow i_t = \sigma\left(W_{hi} h_{t-1} + W_{xi} x_t\right)$$

$$\Rightarrow \sigma\left(\begin{bmatrix} 0.1 & 0.2 \\ -0.2 & 0.05 \end{bmatrix}_{2\times 2} \begin{bmatrix} 0.0 \\ 0.1 \end{bmatrix}_{2\times 1} + \begin{bmatrix} 0.5 & -0.3 \\ 0.4 & 0.1 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} 0 + 0.02 \\ 0 + 0.005 \end{bmatrix} + \begin{bmatrix} 0.25 + 0.03 \\ 0.20 - 0.01 \end{bmatrix}\right)$$

$$= \sigma\left(\begin{bmatrix} 0.02 \\ 0.005 \end{bmatrix} + \begin{bmatrix} 0.28 \\ 0.19 \end{bmatrix}\right)$$

$$= \sigma \begin{pmatrix} 0.30 \\ 0.024 \end{pmatrix}$$

$$0.5460$$

$$i_t \Rightarrow \begin{bmatrix} 0.5744 \end{bmatrix}$$

$$O_t = \sigma\left(\begin{bmatrix} 0.15 & 0.05 \\ 0.1 & -0.02 \end{bmatrix} \begin{bmatrix} 0.0 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.3 & 0.25 \\ -0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix}\right)$$

$$= \sigma \begin{bmatrix} 0.0 + 0.005 \\ 0.0 - 0.02 \end{bmatrix} + \begin{bmatrix} 0.15 + 0.025 \\ -0.10 - 0.02 \end{bmatrix}$$

$$= \begin{bmatrix} 0.05 \\ -0.02 \end{bmatrix} + \begin{bmatrix} 0.125 \\ -0.12 \end{bmatrix}$$

$$\Rightarrow \sigma \begin{pmatrix} 0.175 \\ -0.14 \end{pmatrix}$$

$$o_t = [0.5836 \quad 0.46]$$

$$g_t = \tanh\left(\begin{bmatrix} 0.2 & 0.1 \\ -0.1 & 0.05 \end{bmatrix} \begin{bmatrix} 0.0 \\ 0.1 \end{bmatrix} + \begin{bmatrix} -0.5 & 0.4 \\ 0.2 & -0.3 \end{bmatrix} \begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix} \right)$$

$$g_t = \tanh\left(\begin{bmatrix} 0.0 + 0.01 \\ 0.0 + 0.005 \end{bmatrix} + \begin{bmatrix} -0.25 - 0.04 \\ 0.10 + 0.03 \end{bmatrix} \right)$$

$$\tanh\left(\begin{bmatrix} 0.01 \\ 0.005 \end{bmatrix} + \begin{bmatrix} -0.29 \\ 0.13 \end{bmatrix} \right)$$

$$g_t = \tanh\left[\begin{pmatrix} -0.28 \\ 0.135 \end{pmatrix} \right]$$

$$g_t = [-0.273 \quad 0.134]$$

$$h_t = [0.44 \quad 0.53]^T$$

$$c_t = [f_t \odot c_{t-1} + i_t \odot g_t]$$

$$= \left(\begin{bmatrix} 0.44 & 0.53 \end{bmatrix} (0.2, -0.2) + \begin{bmatrix} 0.5744 & 0.5460 \\ -0.273 & 0.13 \end{bmatrix} \right)$$

$$c_t = \left(\begin{bmatrix} 0.088 & -0.106 \end{bmatrix} + \begin{bmatrix} 0 & 0.013 \end{bmatrix} \right)$$

$$c_t = \begin{bmatrix} 0.088 & -0.106 \end{bmatrix} + \begin{bmatrix} -0.1539 & 0.073 \end{bmatrix}$$

$$C_t = \begin{bmatrix} 0.088 & -0.093 \end{bmatrix} \cdot \begin{bmatrix} -0.06 & -0.03 \end{bmatrix}$$

$$h_t = o_t \times \tanh \begin{bmatrix} 0.88 & -0.093 \end{bmatrix}$$

$$h_t = \begin{bmatrix} 0.53 & 0.46 \end{bmatrix} \times \begin{bmatrix} 0.707 & -0.0936 \end{bmatrix}$$

$$h_t = \begin{bmatrix} 0.53 & 0.46 \end{bmatrix} \times \begin{bmatrix} -0.059 & -0.029 \end{bmatrix}$$

$$h_t = \begin{bmatrix} -0.031 & -0.013 \end{bmatrix}$$

RNN

$$\frac{\partial h}{\partial h_{t-1}} = \tanh'\left(W_{hh}\, h_{t-1} + W_{xh}\, x_{t-1}\right)$$

LSTM

$$\frac{\partial h_t}{\partial h_{t-1}} = o_t \tanh'(c_t) \boxed{\frac{\partial c_t}{\partial h_{t-1}}} + \frac{\partial o_t}{\partial h_{t-1}} \tanh(c_t).$$

$$\frac{\partial c_t}{\partial h_{t-1}} = f_t \frac{\partial c_{t-1}}{\partial h_{t-1}} + c_{t-1} \frac{\partial f_t}{\partial h_{t-1}} + g_t \frac{\partial i_t}{\partial h_{t-1}} + i_t \frac{\partial g_t}{\partial h_{t-1}}$$

It is the one in which LSTM prevents vanishing gradient problem is minimized but it can go to zero so rather than neglecting because it is possible that sometimes it can lead to zero.

Disadvantages :- It is computationally very expensive.

$$o_t \tanh^{-1}(c_t) \cdot \frac{dc_t}{d(h_{t-1})} + \text{dot} \frac{\partial h(t-1)}{\partial} \tanh(c_t)$$

$$o_t \tanh' c_t \frac{dc_t}{dh_{t-1}} + \frac{\partial o_t}{\partial h_{t-1}} \tanh(c_t)$$

$h_1 = [1, 0, 1]$

$h_2 = [0, 1, 1]$

$h_3 = [1, 1, 0]$.

$c = \alpha_{t_1} h_1 + \alpha_{t_2} h_2 + \alpha_{t_3} h_3$

score fn = dot product).

$s_{t-1} = [1, 0, 1]$.

$$\alpha_{t_1} \Rightarrow \frac{\exp([1,0,1] \cdot [1,0,1])}{\exp([1,0,1] \cdot [1,0,1]) + \exp([1,0,1] \cdot [0,1,1]) + \exp([1,0,1] \cdot [1,1,0])}$$

$$\alpha_{t_1} \Rightarrow \frac{\exp[1+0+1]}{\exp[1+0+1] + \exp[0+0+1] + \exp[1+0+0]}$$

$$\alpha_{t_1} = \frac{\exp[2]}{\exp[2] + \exp[1] + \exp[1]}$$

$\alpha_{t_1} = \frac{7.4}{7.4 + 2.7 + 2.7}$

$\alpha_{t_1} \Rightarrow \frac{7.4}{12.8}$

$\Rightarrow 0.578$.

$\alpha_{t_2} \Rightarrow \frac{2.7}{12.8}$.

$\Rightarrow 0.21$

$\alpha_{t_3} \Rightarrow 0.21$.

$$c_t = 0.57\,[1, 0, 1] + 0.2\,[0, 1, 1] + 0.2\,[1, 0, 0]$$

$$c_t \Rightarrow [0.57 \quad 0 \quad 0.57] + [0 \quad 0.2 \quad 0.2] + [0.2 \quad 0 \quad 0]$$

$$\Rightarrow [0.57 \quad 0.2 \quad 0.77] + [0.2 \quad 0.2 \quad 0]$$

$$c_t \Rightarrow [0.77 \quad 0.4 \quad 0.77]^T$$

As c_t is closer to h_1 it is paying more attention to h_1.

7/10/25

vector database;

Key	Value
[1.0, 0.5, 0.2]	"Action movie Die hard" → ②
[0.3, 0.0, 0.1]	"Romantic Titanic" → ③
[0.9, 0.4, 0.8]	"Action Movie" John Wick. → ①

Q = looking for action movie

↓ Word 2 Vec

[1.0, 0.3, 0.5].

Dot product

[1 + 0.15 + 0.1]

⇒ 1.25

⇒ 0.3 + 0.3 + 0.05

⇒ 0.65

⇒ 0.9 + 0.12 + 0.40

⇒ 1.42

9/10/25

Transformers

It has encoder and decoder
↓ ↓
2 layer 3 layer.

- Self attention is a part of Multi Head Attention.
- Project the embedding vector into some space.

2) Compute the z_1 & z_2 for the input sentence & playing outside.

Embedding

First step :- word 2 vec.

Multiply that by W_q W_k W_v

and $q_1 = [0.212 \quad 0.04 \quad 0.63 \quad 0.36]^T$.

$k_1 = [0.31 \quad 0.84 \quad 0.963 \quad 0.57]^T$.

$v_1 = [0.36 \quad 0.83 \quad 0.1 \quad 0.38]^T$

outside :-

$q_2 = [0.1 \quad 0.14 \quad 0.86 \quad 0.77]^T$

$k_2 = [0.45 \quad 0.94 \quad 0.73 \quad 0.38]^T$

$v_2 = [0.31 \quad 0.36 \quad 0.19 \quad 0.72]^T$

$q_1 \cdot K_1 = 0.212 \times 0.31 + 0.04 \times 0.84 + 0.63 \times 0.963 + 0.36 \times 0.57$

$q_1 \cdot K_1 = 0.06572 + 0.0336 + 0.60669 + 0.2052$

$q_1 K_1 = 0.9112 \Rightarrow 0.91$

Divide by $\sqrt{dk}$

dimension of key.

$\sqrt{4}$.

2.

Divide by 2 $\quad \frac{0.9112}{2} = 0.45556$.

$\Rightarrow 0.46 \Rightarrow$ score

Softmax.

$q_1 \cdot K_2 = 0.212 \times 0.45 + 0.04 \times 0.94 + 0.63 \times 0.73 + 0.36 \times 0.5$

$\Rightarrow 0.0954 + 0.0376 + 0.4599 + 0.2088$

$\Rightarrow 0.8017$.

Divide by 2 $\Rightarrow 0.4008$. $\Rightarrow$ score.

Softmax $\Rightarrow$ $\quad 0.5138 \quad 0.4862$.

$0.52 \quad 0.49$.

$$0.52 \times \begin{bmatrix} 0.36 & 0.83 & 0.1 & 0.38 \end{bmatrix}$$

$$\begin{bmatrix} 0.1872 & 0.4816 & 0.052 & 0.1976 \end{bmatrix}$$

$$0.12 \times \begin{bmatrix} 0.31 & 0.36 & 0.19 & 0.72 \end{bmatrix}$$

$$\begin{bmatrix} 0.0372 & 0.0432 & 0.0228 & 0.0864 \end{bmatrix}$$

$$0.49 \begin{bmatrix} 0.31 & 0.36 & 0.19 & 0.72 \end{bmatrix}$$

$$z \begin{bmatrix} 0.1519 & 0.1764 & 0.0931 & 0.3528 \end{bmatrix}$$

$$z_1 \; z) \begin{bmatrix} 0.3391 & 0.608 & 0.146 & 0.5504 \end{bmatrix}$$

22

$q_2 \cdot k_1 = [0.1 \quad 0.14 \quad 0.86 \quad 0.77] \; [0.31 \quad 0.84 \quad 0.963 \quad 0.57]$.

$q_2 \cdot k_1 = 0.031 + 0.1176 + 0.828 + 0.4389$.

$q_2 \cdot k = 1.42$.

divide by 2 $= 0.71$.

$q_2 \cdot k_2 = [0.1 \quad 0.14 \quad 0.86 \quad 0.77] \; [0.45 \quad 0.94 \quad 0.730$

$q_2 \cdot k_2 = [0.045 + 0.13 + 0.63 + 0.45]$.

$q_2 \cdot k_2 = 1.255$

$q_2 \cdot k_2 \Rightarrow 0.6275$.

Softmax $= 0.5203$ $\quad 0.4797$

$\Rightarrow 0.52$. $\quad \Rightarrow 0.48$.

$0.52 \times [0.36, \quad 0.83 \quad 0.1 \quad 0.38]$

$= [0.1872 \quad 0.4316 \quad 0.052 \quad 0.1976]$.

$0.48 \times [0.31 \quad 0.86 \quad 0.19 \quad 0.72]$

$\Rightarrow [0.1488 \quad 0.1728 \quad 0.0912 \quad 0.3456]$.

$z_1 = [0.336 \quad 0.60 \quad 0.14 \quad 0.54]$.

$z_2 = [0.30 \quad 0.59 \quad 0.14 \quad 0.54]$.

- Whenever we want to concatenate bring MLP.

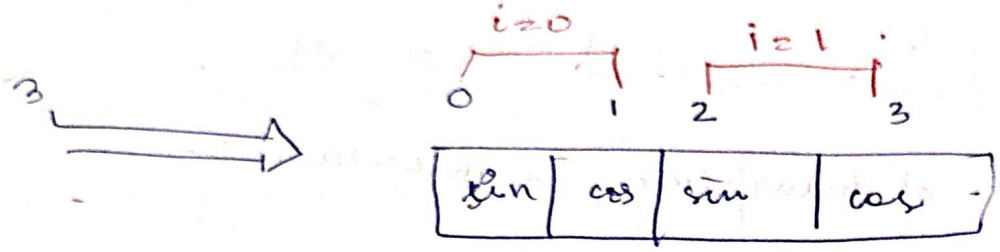
- If we transformer classification we will take out the output of the encoder and then we will feed to XG boost or SVM.

If we want to do transformer translation we will feed the output of the encoder to the decoder.

:> t=4 :> positional embedding vector
i.e. $\in \mathbb{R}^{512}$

:>

For all even function use sin function.
For all odd function use cos function.



positional encoder

$PE_{3,2i}$

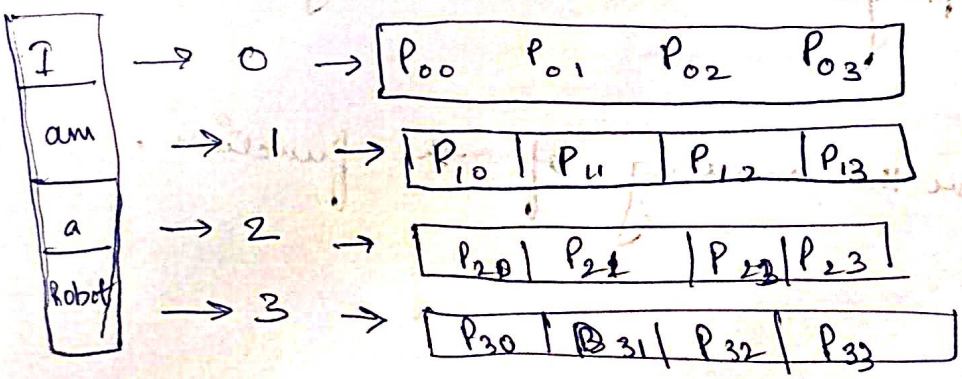
$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{emb}}}\right)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{emb}}}\right)$$

If we have d vector.
$0 < i < d/2$.

I am a robot.

P.E : $d = 4 \Rightarrow \mathbb{R}^4$



$P_{pos,2i} = \sin\left(\frac{Pos}{100}(2i/12^{0})\right)$ $\Rightarrow \sin\left(\frac{?}{100^{?}}\right)$

only for this case.

$\Rightarrow \sin\left(\frac{0}{100}(2\times 0/4)\right)$

$\sin 0$

$\Rightarrow \cos 0$

sin(0) 0	cos(0) 1	sin(0) 0	cos(0) 1

0.017 / 0.018	0.998 / 1	0.001	0.99 / 1

sin(1) cos(1)

$P_{1,20} = \sin\left(\frac{1}{0}\right)$

$\sin 0$

$P_{1,9} = \cos 0$

$P_{1,2} = \sin\left(\frac{1}{100} \times 2/4\ 2\right)$

$\sin\left(\frac{1}{100} \times 0/4\right)$

1

$P_{1,3} = \sin \frac{1}{10}$

$\sin 0.1$

sin(1)

$\Rightarrow 0.099$

$\Rightarrow 0.1$

$P_{1,4} = \cos 0.1$

$\Rightarrow 0.99 \Rightarrow 1.$