key use case

Seq to Seq Models

→ Transcription, chatgpt, translation etc
LLMs comes under this.

Reinforcement learning

→ very hard to train, very active
→ Games are classical use case.

Then came the craziness
↳ language models, speech recognition, machine translation, conversation modeling, objection detection recognition, etc. visual tracking, image/video captioning, generating authentic faces / new audio etc.

LARI

Forward pass: input → layers → output

Backward pass: final output → layers → update weight

① Tanh:

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

$$f'(z) = \frac{(e^z+e^{-z})\frac{d}{dt}(e^z-e^{-z}) - (e^z-e^{-z})\frac{d}{dt}(e^z+e^{-z})}{(e^z+e^{-z})^2}$$

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

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

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

② ReLU fn:

$$r(z) = \max(0, z)$$

$$r(z) = \begin{cases} az & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

$$r'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

③ leaky ReLU

$$r(z) = \max(az, z)$$

z input, a → constant

$$r(z) = \begin{cases} a\,0.01(z) & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases}$$

$$r'(z) = \begin{cases} 0.01 & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases}$$

④ Softmax:

$$s(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

→ specific element in z.

→ sum of exp of z.

Assume, $\sum_{j=1}^{K} e^{z_j} = S$, $s(z)_i = \frac{e^{z_i}}{S}$

Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial s_1}{\partial x_1} & \frac{\partial s_1}{\partial x_2} & \frac{\partial s_1}{\partial x_3} \\ \frac{\partial s_2}{\partial x_1} & \frac{\partial s_2}{\partial x_2} & \frac{\partial s_2}{\partial x_3} \end{bmatrix}_{m \times n}$$

$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ —softmax→ $\begin{bmatrix} j_1 \\ j_2 \\ j_3 \end{bmatrix}$

eg: $x = \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix}_{3 \times 2}$

1) compute softmax output
2) compute jacobian output.

$$s_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

① $s = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}_{3 \times 1}$ Sum = 11.212

$$s_1 = \frac{7.389}{11.212} = 0.66 \qquad s_2 = \frac{2.718}{11.212} = 0.24 \qquad s_3 = \frac{1.105}{11.212} = 0.1$$

$$s = \begin{bmatrix} 0.66 \\ 0.24 \\ 0.1 \end{bmatrix}$$

$\frac{\partial s_i}{\partial x_j} = s_i(1 - s_i)$ $\Rightarrow J_{ij} = \frac{\partial s_i}{\partial x_j}$

(diag = 1)

In case 1: when $i = j$

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

Case 2: $i \neq j$ (off diag = 0)

$$\frac{\partial s_i}{\partial x_j} = -s_i s_j$$

$$J = \begin{bmatrix} 0.2244 & -0.1584 & -0.066 \\ -0.1584 & 0.1824 & -0.027 \\ -0.066 & -0.027 & 0.01 \end{bmatrix}$$

Now how did the derivatives come?

$$s(\vec{x}_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} = \frac{e^{x_i}}{sum}$$

(sum = $x_1 + x_2 + x_3$)

$$\frac{\partial s_i}{\partial x_j}$$

$i = j$

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

$(n = 3)$

$$s_1 = \frac{e^{x_1}}{e^{x_1} + e^{x_2} + e^{x_3}}, \quad s_2 = \frac{e^{x_2}}{e^{x_1} + e^{x_2} + e^{x_3}}, \quad s_3 = \frac{e^{x_3}}{e^{x_1} + e^{x_2} + e^{x_3}}$$

$$s_1 = \frac{2.718}{30.192}, \quad s_2 = \frac{7.389}{30.192}, \quad s_3 = \frac{20.085}{30.192}$$

Partial derivatives: $(i = j)$

$$\frac{\partial s_1}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\frac{e^{x_i}}{sum}\right) = \frac{\frac{d}{dx_1}(e^{x_i}) \cdot sum - e^{x_i}\left(\frac{d}{dx_1}(S)\right)}{(sum)^2}$$

$$= \frac{e^{x_i} \cdot sum - e^{x_i} \cdot e^{x_1}}{(sum)^2} = \frac{e^{x_1}(sum - e^{x_1})}{(sum)^2}$$

$$\frac{\partial s_1}{\partial x_1} = s(x_i)\,(1 - s(x_i))$$

(2) $\frac{\partial s_1}{\partial x_2}$

$$s_1 = \frac{e^{x_1}}{sum}$$

$$sum = x_1 + x_2 + x_3$$

$$\frac{\partial s_1}{\partial x_2} = \frac{\left(\frac{d}{dx_2}(e^{x_1}) = 0 \times sum\right) - e^{x_1} \cdot \frac{d}{dx_2}(sum)}{(sum)^2}$$

$$= \frac{0 \cdot sum - e^{x_1} \cdot e^{x_2}}{(sum)^2} = -\left(\frac{e^{x_1}}{sum}\right)\left(\frac{e^{x_2}}{sum}\right)$$

$$= -s_1 \cdot s_2$$

---

(1) $i = j$

$$s = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}, \quad \frac{e^{x_i}}{sum}$$

$$\frac{\partial s_1}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\frac{e^{x_1}}{sum}\right)$$

$$= \frac{\frac{d}{dx_1}(e^{x_1}) \cdot sum - e^{x_1} \frac{d}{dx_1}(sum)}{(sum)^2}$$

$$= \frac{e^{x_1} \cdot sum - e^{x_1} \cdot e^{x_1}}{(sum)^2} = \frac{e^{x_1}(sum - e^{x_1})}{(sum)^2}$$

$$= s\,(1 - s)$$

(2) $i \neq j$

$$\frac{\partial s_1}{\partial x_2} = \frac{\partial}{\partial x_2}\left(\frac{e^{x_1}}{sum}\right)$$

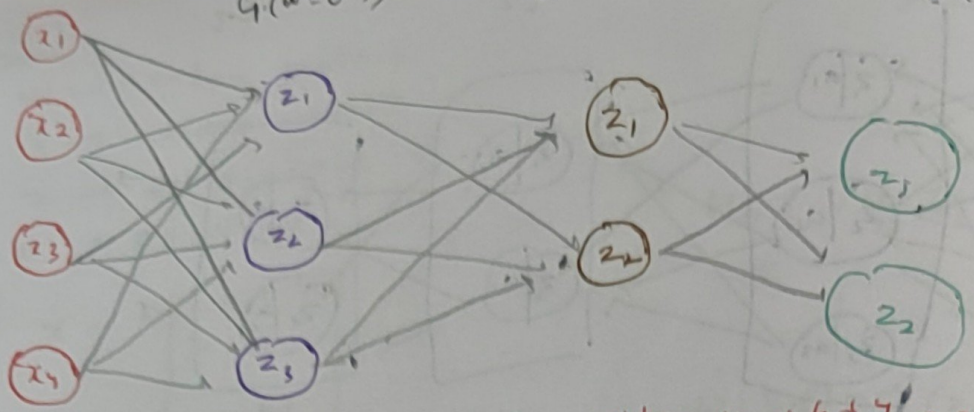
$$= \frac{\frac{d}{dx_2}(e^{x_1}) \cdot sum - e^{x_2} \cdot \frac{d}{dx_2}(sum)}{(sum)^2} = \frac{0 \cdot sum - e^{x_2} \cdot e^{x_1}}{(sum)^2}$$

$$= -s \cdot s$$

So derivative for softmax:

if $i \neq j$, $s(1-s)$ $-s \cdot s$

if $i = j$, $s(1-s)$



Total weights = 12 + 6 + 4 = 22

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2.4 \\ 1.2 \\ -0.8 \\ 1.1 \end{bmatrix}$$

L1

$$z_1 = \begin{bmatrix} -2.4 \\ 1.2 \\ -0.8 \\ 1.1 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \\ 0.1 \end{bmatrix} = \begin{bmatrix} -0.24 \\ 0.12 \\ -0.08 \\ 0.11 \end{bmatrix} = -0.09$$

instead of each time, do matrix multiplication

$$z_1 = w \cdot x = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}_{4\times4} \begin{bmatrix} -0.24 \\ +0.12 \\ -0.08 \\ +0.11 \end{bmatrix}_{4\times1}$$

$$= \begin{bmatrix} -0.09 \\ -0.09 \\ -0.09 \\ -0.09 \end{bmatrix}$$

$$z_2 = w \cdot z_1 \; L_2 \begin{bmatrix} 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.01 \end{bmatrix} \begin{bmatrix} -0.09 \\ -0.09 \\ -0.09 \end{bmatrix}$$

$$=$$

$$\text{Relu} \begin{bmatrix} -0.09 \\ -0.09 \\ -0.09 \end{bmatrix} = 0.$$

$$z_2 = w \cdot L_2 \cdot \begin{bmatrix} 0.001 & 0.001 & 0.001 \\ 0.001 & 0.001 & 0.001 \\ 0.001 & 0.001 & 0.001 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \qquad \text{Relu} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = 0.$$

$$z_3 \cdot \begin{bmatrix} 0.01 & 0.01 \\ 0.01 & 0.01 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0 \qquad \text{softmax} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = ?$$

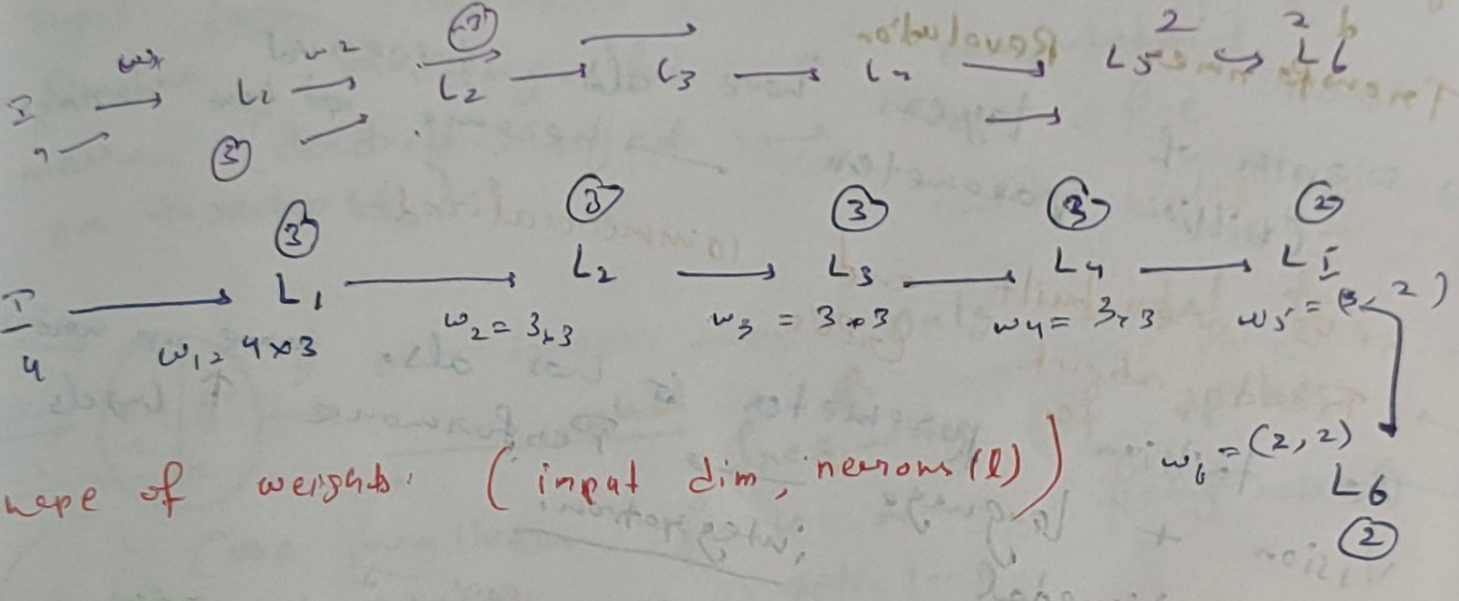
$$\hat{y} = \frac{e^0}{e^0 + e^0} = \frac{1}{1+1} = \frac{1}{2} = 0.5$$

$\hat{y} = 0.7$

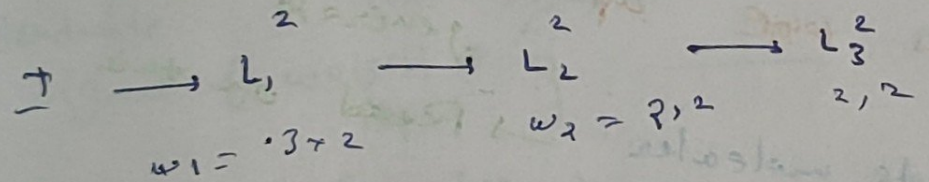
→ Bias terms are also introduced to the forward pass

→ if 4 input, 3 neuron → matrix (4, 3)

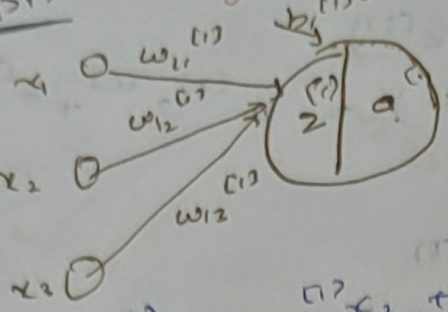
if 3 input, 2 neuron (3, 2) matrix.



shape of weights: (input dim, neurons (l))



Perceptron : single layer single neuron network with No hidden layer



here A binary classifica

$$z^{[1]} = w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2 + w_{13}^{[1]} x_3 + b_1^{[1]}$$

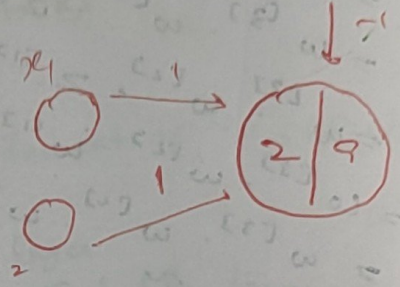
$$a^{[1]} = g(z^{[1]})$$

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

example :

AND

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



$$z^{[1]} = w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2 + b_1^{[1]}$$

① $z^{[1]} = (0)(1) + (0)(1) + (-1)$

$z^{[1]} = -1$ $\quad g = 0$ (bcoz $z = -1 < 0$)

② $z^{[1]} = (0)(1) + (1)(1) + (-1)$

$= 0 + 1 * 1 = 0$ $\quad g = 0$

③ $z^{[1]} = (1)(1) + (1)(0) + (-1) = 1 - 1 = 0$ $\quad g = 0$

④ $z^{[1]} = (1)(1) + (1)(1) + (-1) = 1$ $\quad g = 1$ $(z > 0)$

OR

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

$b_1 = -1$

① ② $z^{[1]} = (1)(0) + (1)(0) + (-1)$ $\quad g(-1) = 0$

$= 0 + 0 + (-1) = -1$

③ $z^{[1]} = (1)(0) + (1)(1) + (-1)$ $\quad g(0) = 0$ ✗

$= 1 - 1 = 0$

④ $z^{[1]} = (1)(1) + (1)(0) + (-1)$ $\quad g(0) = 1$ ✓

$= 1 - 1 = 0$

bias = 0

now, ① $z = 0$

② $z = 0$

③ $z = 0$

④ $z = 0$

① $z = (1)(0) + (1)(0) + (-1) = -1$, $g(-1) = 0$.

② $z = (1)(0) + (1)(1) + (-1) = 0$, $g(0) = 0$

③ $z = (1)(1) + (1)(0) + (-1) = 0$, $g(x) = 0$

④ $z = (1)(1) + (1)(1) + (-1) = 1$, $g(1) = 1$

→ we can change the bias

$z = (0)(1) + (0)(0) + (-2)$

$z = 0$ → problem

XOR

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

(bias = -1)

④ $z^{[1]} = (1)(1) + (1)(1) + (-1)$

$= 2 - 1 = 1$

$g(1) = 1$

(bias = -2)

$z = (1)(1) + (1)(1) + (-2)$

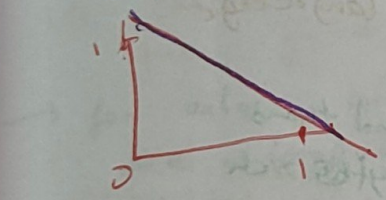
$= 2 - 2 = 0$

$z^{[1]} = w_{11} x_1 + w_{12} x_2 + b_1^{[1]}$

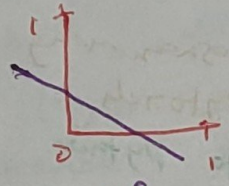
$z = (1)(1) + (1)(1) + (-2)$

$= 1 + 1 - 2 = 0$

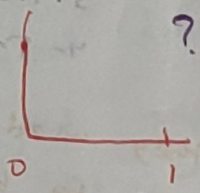
→ We can come up with a linear boundary for AND and OR.



AND



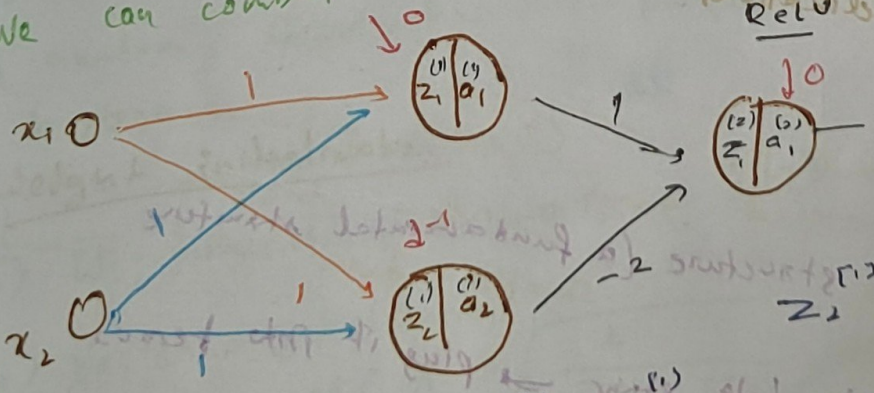
OR



XOR

this problem with a neural network

We can combat



ReLU

Input = (0, 1)

$z_1^{[1]} = w_{11}^{(1)} a_1^{[0]} + w_{12}^{(1)} a_2^{[0]} + b_1^{(1)}$

$= (1)(0) + (1)(1) + 0 = 1$

$g(1) = 1$

$z_2^{[1]} = (1)(0) + (1)(1) + (-1)$

$= 0$ $g(0) = 0$

$z_1^{[2]} = (1)(1) + (-2)(0) + 0$

$= 1 = +1$

$g_1^{(2)} = 1$

(2) Input (1, 1)

$z_1^{[1]} = (1)(1) + (1)(1) + 0 = 2$ $\quad z_2^{[1]} = (1)(1) + (1)(1) + (-1)$

$g(2) = 1$ $\quad = 1$, $g(1) = 1$

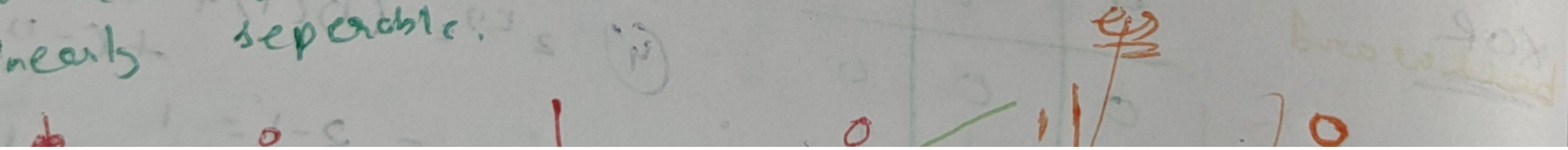
$z_1^{[2]} = (1)(1) + (1)(-2) = 1 - 2 = -1$, $\quad g(-1) = 0$

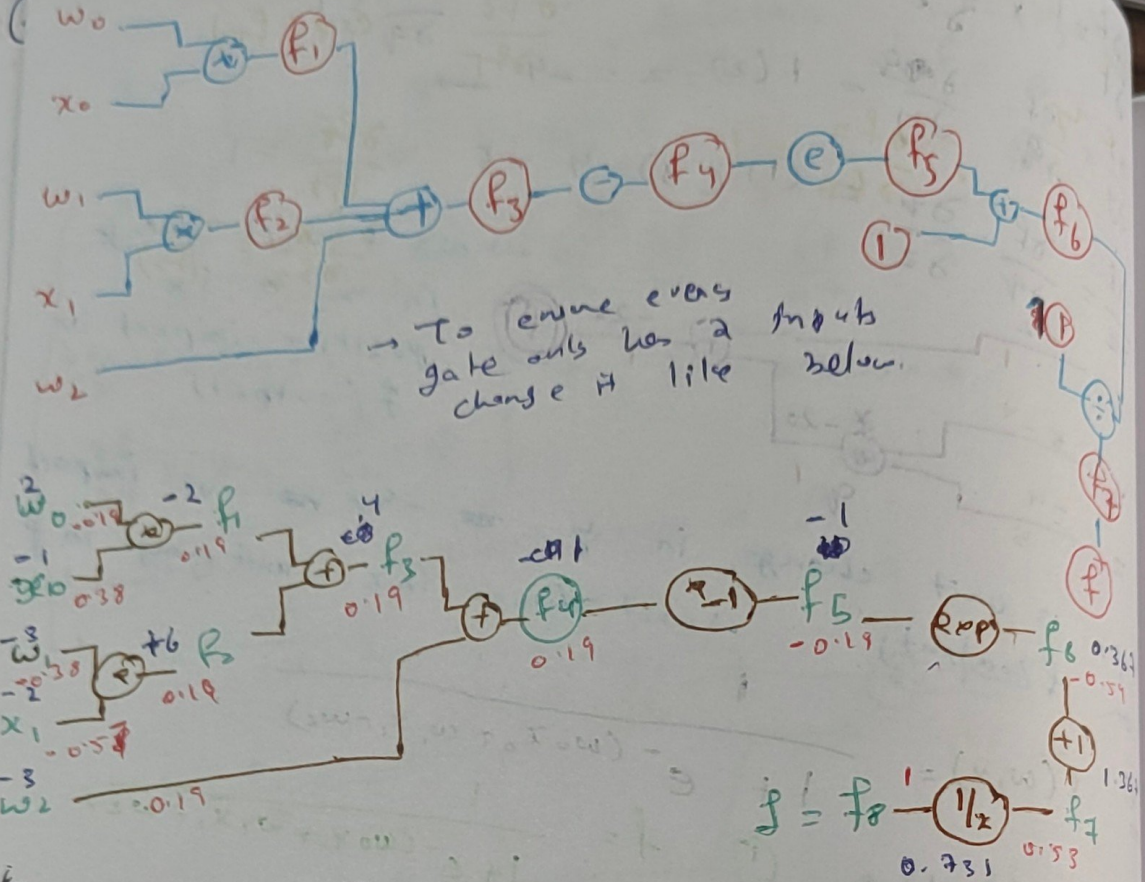
$+0$ $\quad y = 0$ //.

Now it give a correct answer. with a hidden layer.

why is this working?.

Is we transform our input data (from original space to new space) $\longrightarrow$ in this space there are linearly seperable.





Computing gradients. ③ forward pass:

$f_1 = 2(-1) = -2$, $f_2 = -6$, $f_3 = -8$, $f_4 = -11$,

$f_5 = 11$, $f_6 = 59.8744$, $f_7 = 59.8744$

$f_8 =$

$f_1 = -2$, $f_2 = 6$, $f_3 = 4$, $f_4 = 1$, $f_5 = 1$, $f_6 = 0.36$, $f_7 = 1.36$

$f_8 = 0.73$

① compute gradients

$\frac{\partial f}{\partial f_8} = 1$, $\frac{\partial f}{\partial f_7} = \frac{\partial f}{\partial f_7}\left(\frac{1}{f_7}\right)$

$\frac{\partial f_1}{\partial \cdot}$ $f_1 = w_0 x_0$, $\frac{\partial f_1}{\partial w_0} = x_0$, $\frac{\partial f_1}{\partial x_0} = w_0 = 2$

$f_2 = w_1 x_1$, $\frac{\partial f_2}{\partial w_1} = x_1$, $\frac{\partial f_2}{\partial x_1} = w_1$

$= -2$ $= -3$

$f_3 = f_1 + f_2$, $\frac{\partial f_3}{\partial f_1} = 1$, $\frac{\partial f_3}{\partial f_2} = 1$

$f_4 = f_3 + w_2$, $\frac{\partial f_4}{\partial f_3} = 1$, $\frac{\partial f}{\partial w_2} = 1$

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

$f_6 = \frac{\partial f_6}{\partial f_5} = \exp(f_5)$

$f_6 = \exp(f_5)$ $\quad \frac{\partial f_6}{\partial f_5} = \exp(f_5)$

$f_7 = f_6 + 1$, $\frac{\partial f_7}{\partial f_6} = 1$ $\quad f_8 = f_8 \quad \frac{\partial f_8}{\partial f_8} = 1$

$f_8 = f_8 \frac{1}{f_7}$, $\frac{\partial f_8}{\partial f_7} = \frac{-1}{(f_7)^2}$

Now cal gradient

$f_8 = 1, \quad f_7 = \frac{1}{(f_7)^2} = \frac{1}{(0.36)^2} = -0.53$

$f_6 = \frac{\partial f_8}{\partial f_7} = \text{local} \times \text{global} = -\frac{1}{f_7^2} \times 1 = -0.534$

$\frac{\partial f_8}{\partial f_6} = \frac{\partial f_8}{\partial f_7} \cdot \frac{\partial f_7}{\partial f_6} = (-0.534)(1) = -0.54$

$\frac{\partial f_8}{\partial f_5} = \frac{\partial f_8}{\partial f_7}$ of local $\times$ global $= \frac{\partial f_6}{\partial f_5} \times \frac{\partial f_8}{\partial f_6} = \exp(-1) \times (-0.54)$

$= (0.36)(-0.54)$

$= -0.1944$

$= -0.19$

$\frac{\partial f_8}{\partial f_4} = \frac{\partial f_5}{\partial f_4} \cdot \frac{\partial f_8}{\partial f_5} = (-1)(-0.19) = 0.19$

$\frac{\partial f_8}{\partial f_3} = \frac{\partial f_4}{\partial f_3} \cdot \frac{\partial f_8}{\partial f_4} = (1)(0.19) = 0.19$

$\frac{\partial f_8}{\partial w_{b2}} = \frac{\partial f_8}{\partial} \frac{\partial f_4}{\partial w_{b2}} \cdot \frac{\partial f_3}{\partial} = (1)(0.19) = 0.19$

$\frac{\partial f_8}{\partial f_1} = \frac{\partial f_3}{\partial f_1} \cdot \frac{\partial f_8}{\partial f_3} = (1)(0.19) = 0.19$

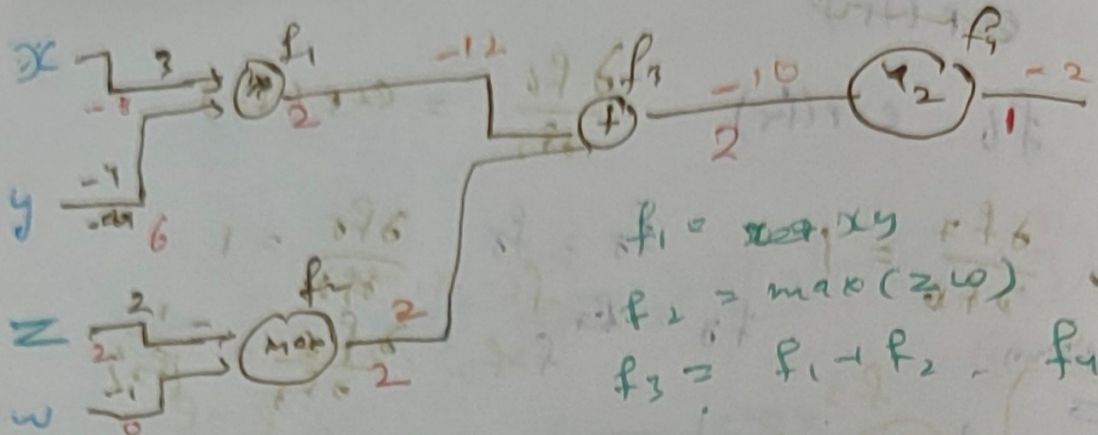
$\frac{\partial f_8}{\partial f_2} = \frac{\partial f_3}{\partial f_1} \cdot \frac{\partial f_8}{\partial f_3} = (1)(0.19) = 0.19$

$\frac{\partial f_8}{\partial w_0} = \frac{\partial f_1}{\partial w_0} \cdot \frac{\partial f_8}{\partial f_1} = (-1)(0.19) = -0.19$

$\frac{\partial f_8}{\partial x_0} = \frac{\partial f_1}{\partial x_0} \cdot \frac{\partial f_8}{\partial f_1} = (2)(0.19) = 0.38$

$\frac{\partial f_8}{\partial w_1} = \frac{\partial f_2}{\partial w_0} \cdot \frac{\partial f_8}{\partial f_2} = (x_1)(0.19) = -0.38$

$\frac{\partial f_8}{\partial x_1} \equiv \frac{\partial f_2}{\partial x_1} \cdot \frac{\partial f_8}{\partial f_2} = (w_1)(0.19) = -0.57$



$f_1 = x \cdot y$, $f_2 = \max(z, w)$, $f_3 = f_1 - f_2$, $f_4 = (f_3)^2$

local: $f_1 = xy$

$\frac{\partial f_1}{\partial x} = y$, $\frac{\partial f_1}{y} = x$,

$f_2 = \max(z, w)$

$f_2 = \begin{cases} z & \text{if } z > w \\ w & \text{if } z < w \\ \text{undefined} & \text{if } z = w \end{cases}$

$f_3 = f_1 - f_2$

$\frac{\partial f_3}{\partial f_1} = 1$, $\frac{\partial f_2}{\partial f_2} = 1$

$\frac{\partial f_2}{\partial z} = 1$ if $z > w$

$= 0$ if $z < w$

$=$ undefined if $z = w$

$f_4 = (f_3)^2$, $\frac{\partial f_4}{\partial f_3} = 2$

$\frac{\partial f_2}{\partial w} = 1$ if $w > z$

0 if $w < z$

undefined if $w = z$

arbitrary

chain:

$\frac{\partial f_4}{\partial f_3} = \frac{\partial f_4}{\partial f_3} = 1$, $\frac{\partial f_4}{\partial f_3} = \frac{\partial f_4}{\partial f_3} = 2$, $\frac{\partial f_4}{\partial f_1} = \frac{\partial f_4}{\partial f_1} = \frac{\partial f_3}{\partial f_1} \cdot \frac{\partial f_4}{\partial f_3}$

$= (1)(2) = 2$

$\frac{\partial f_4}{\partial x} = \frac{\partial f_4}{\partial x} \cdot \frac{\partial f_1}{\partial f_1} = (y)(2)$

$= (4)(2) = 8$, $\frac{\partial f_4}{\partial f}$

$\frac{\partial f_4}{\partial y} = \frac{\partial f_1}{\partial y} \cdot \frac{\partial f_4}{\partial f_1} = (x)(2) = (3)(2) = 6$

now / $\frac{\partial f_4}{\partial f_2} = \frac{\partial f_3}{\partial f_2} \cdot \frac{\partial f_4}{\partial f_3} = (1)(2) = 2$

$\frac{\partial f_4}{z} = \frac{\partial f_2}{\partial z} \cdot \frac{\partial f_4}{\partial f_2} = \begin{pmatrix} 0 \\ 1 \\ \text{undefined} \end{pmatrix}(2) = 2$ if $z > w$

$= 0$ if $z < w$

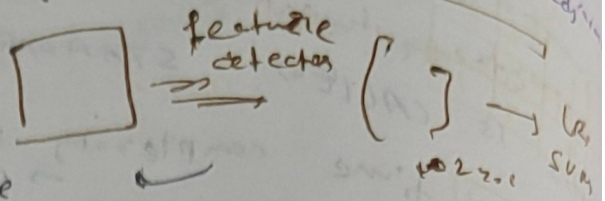
$\frac{\partial f_4}{\partial w} = \frac{\partial f_2}{\partial w} \cdot \frac{\partial f_4}{\partial f_2} = \begin{pmatrix} 0 \\ 1 \\ \text{undefined} \end{pmatrix}(2) = 0$ if $z > w$

$= 2$ if $z < w$

Histogram of oriented gradients (HOG)
Histogram of oriented flows (HOF)
Optical flow features
} Features used at first (Classical)

Most natural images → more?
↳ gradient pixel values
or generally: 437, 240, 1, -240

adjacent gradients are usually homogeneous



only 2 if strong "edge comes in

this become saturated to

CNN

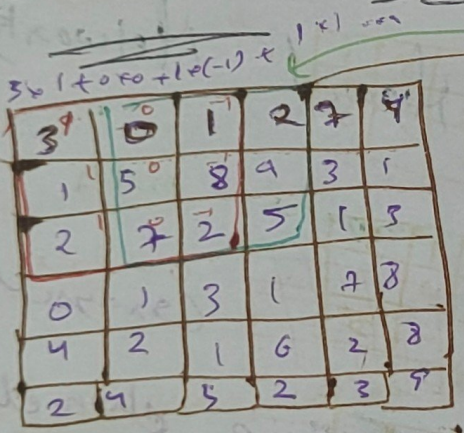
→ Scaling is a problem if we wanna use a FFN

Suppose, 1000 × 1000 × 3 image → 3M × 1000
= 3 billion parameters

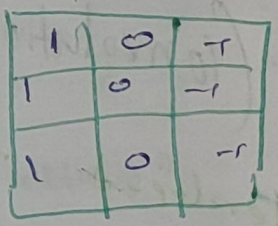
↳ so CNN is better, weight depends on size heavily here which is a problem, overfitting due

Convolution

1 Convolution Operation



input 6×6 image



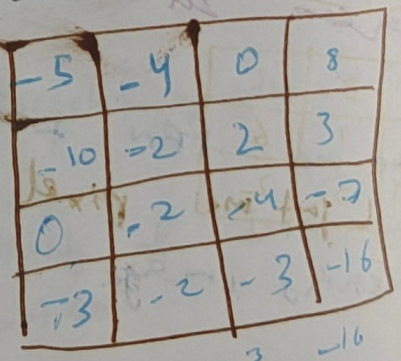
1 filter w learned (3×3)

superimpose

convolution operation

neg values and constant increase

(3×1) + (0×0) + (1×-1)
+ (1×1) + (5×0) + (8×-1)
+ (2×1) + (7×0) + (2×-1)
= 3+0-1+1+0-8+2+0-2
= -5



→ slide after bygone position.

6	1	2
5	8	9
7	4	3

→ now slide again

1	2	7
8	9	3
2	3	1

2	7	4
9	3	1
5	1	3

1	5	8
2	7	2
0	1	5

5	8	9
7	2	5
1	3	1

9	3	1
5	1	3
1	7	8

7	2	5
1	3	1
2	1	6

5	1	3
1	7	8
6	2	8

1	3	1
2	1	6
4	5	2

= (0×1) + (0×1) + (2×-1) + (5×1) + (8×0) + (9×-1) + (7×1) + (2×0) + (5×-1)

= 0 + 0 - 2 + 5 + 0 - 9 + 7 - 5

= -4.

= (1×1) + (2×0) + (7×-1) + (8×1) + (5×0) + (3×-1) + (9×1) + (3×0) + (1×-1)

= 1 + 7 + 8 - 3 + 2 - 1 = 0

1	1	1
1	0	1
1	0	1

= 2 + 0 - 4 + 9 + 0 - 1 + 5 + 0 - 3

= 8

= 1 - 8 + 2 - 2 + 0 - 3 = -10

8	9	3
2	5	1
3	1	7

= 8 - 3 + 2 - 1 + 3 - 7

= 2

2	7	2
0	1	3
4	2	1

= 2 - 2 + 0 - 3 + 4 - 1

= 0

2	5	1
3	1	2
1	6	2

= 2 - 1 + 3 - 7 + 1 - 2

= -4

0	1	3
4	2	1
2	4	5

= 0 - 3 + 4 - 1 + 2 - 5 + 1

= -3

3	7	7
1	6	2
5	2	3

= 3 - 7 + 1 - 2 + 5 - 3

= -3

1	2	8
6	2	8
2	3	9

= 1 - 8 + 6 - 8 + 2 - 9

= -16

we got 6×6 → image back.

formula to compute,

o/p size = $\left(\frac{n+2p-f}{s}+1\right) \times \left(\frac{n+2p-f}{s}+1\right)$

choice is a hyperparameter

"Valid" convolution ⟹ I/P size ≠ O/P size

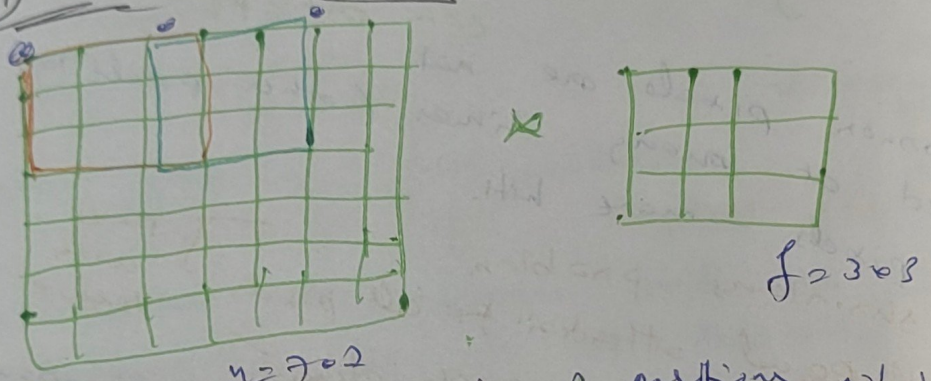
"Same" convolution → I/P size = O/P size

what should value of p to set same convolution

↳ $p = \frac{f-1}{2}$

(3)

Strided Convolution



f = 3 × 3

n = 7 × 7

stride, s = 2 (move by 2 positions, not 1)

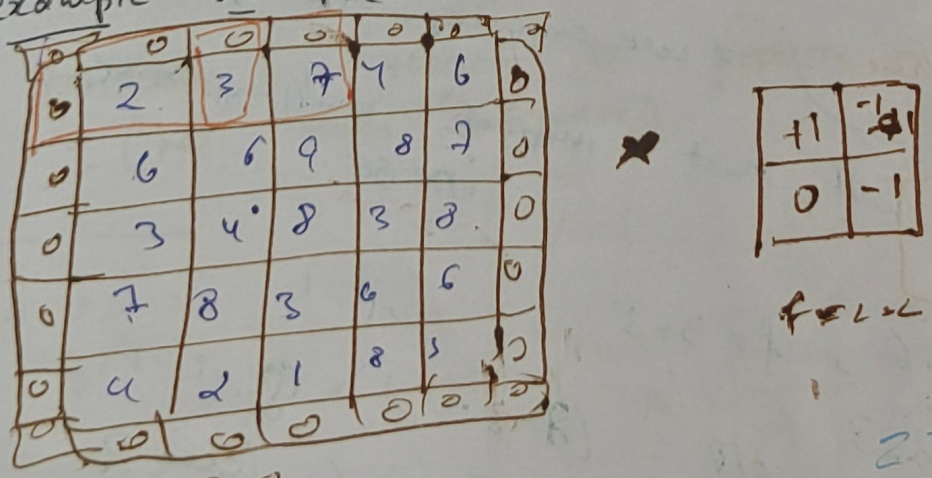
↳ even if we lose info (cannot cover all)

is it okay?

→ output size ↓ → Add padding?

Stride is based on how much attention you wanna give to the pixels (which pixels?)

Example on padding and striding

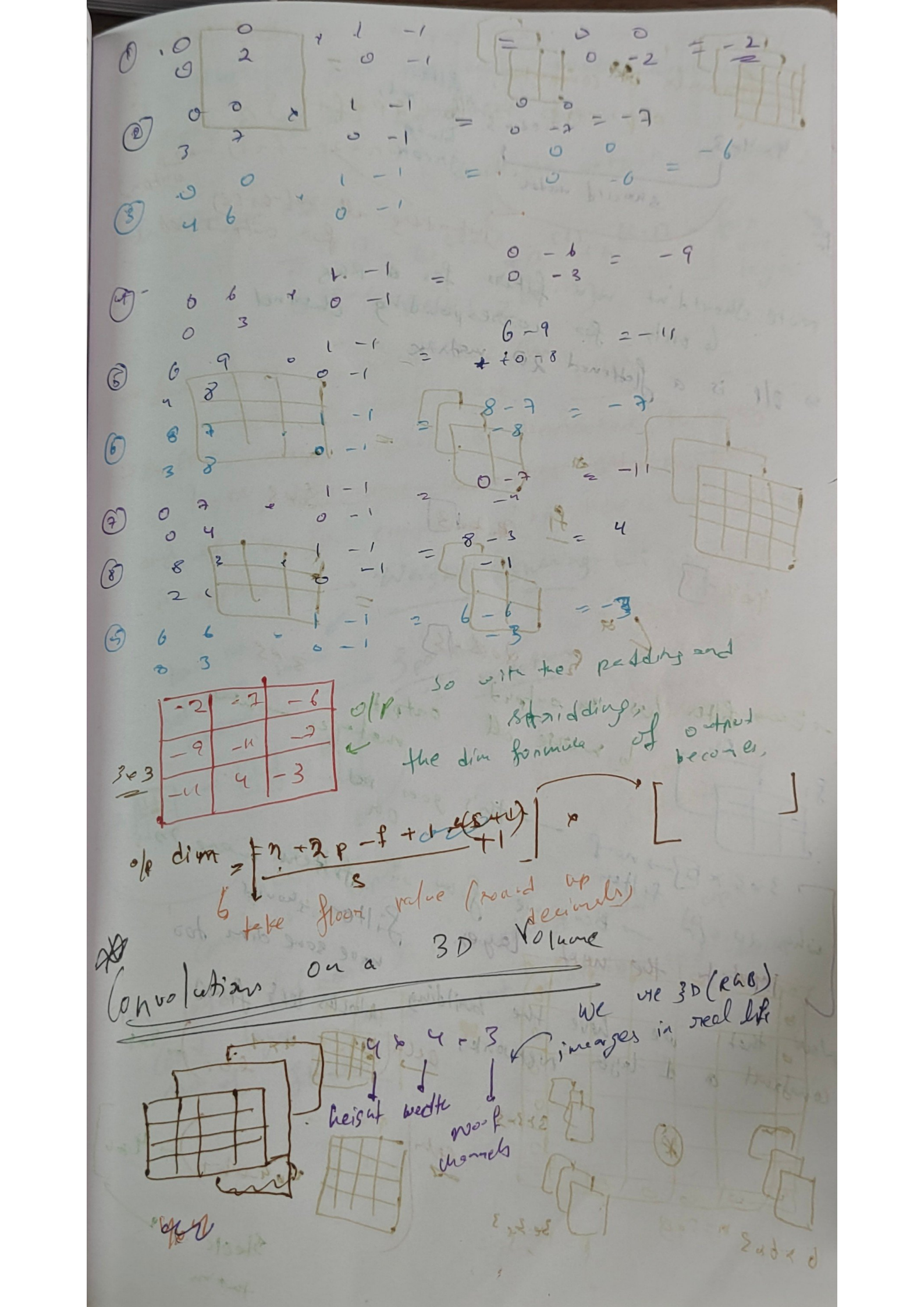


n = 5 × 5

f = 2 × 2

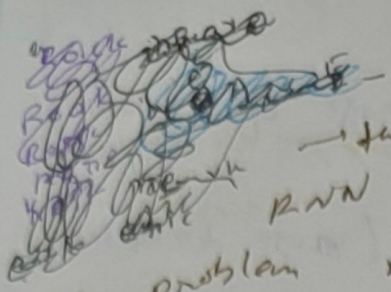
p = 1, s = 2

3 × 3


-2 -7 -6
-9 -11 -7
-11 4 -3
3x3
o/p
so with the padding and stridding, the dim formula of output becomes,
o/p dim = floor((n + 2p - f + d(s+1))/s + 1)
take floor value (round up decimals)
Convolutions on a 3D Volume
we use 3D (RGB) images in real life
4 x 4 x 3
height
width
no. of channels

temporal dependency → handled by RNN

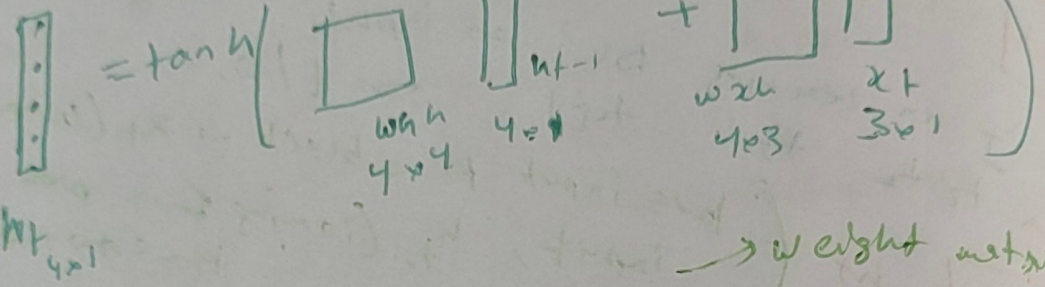
→ we need some input to generate something.



→ the choice of y coming out of a RNN block is optional - depending on problem network

$$\vec{h}_t = f_W(h_{t-1}, x_t)$$

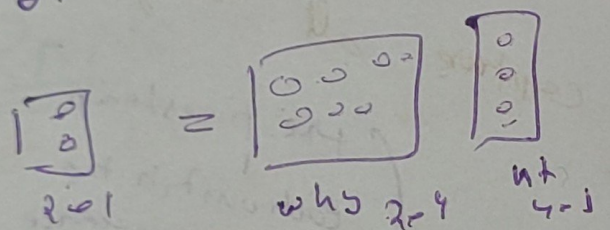
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$



→ weight matrix is shared across timesteps

→ input is fixed size.

$$y_t = W_{hy} h_t$$



Character-level language Model :-

T = 2 (two time steps)

$x^{(1)}$: $x_1 \in \mathbb{R}^2 \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $x^{(2)}$: $x_2 \in \mathbb{R}^2 \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

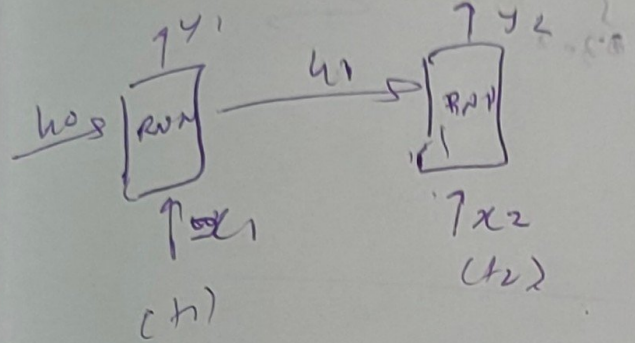
input dim = 2 ⇒ $x_t \in \mathbb{R}^2$

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

$h_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{3\times 1}$, $W_{xh} = \begin{bmatrix} 0.5 & -0.3 \\ 0.8 & 0.2 \\ 0.1 & 0.4 \end{bmatrix}_{3\times 2}$,

$y^{(t)} \in \mathbb{R}^2$ → $W_{hy} = \begin{bmatrix} 1.0 & -1.0 & 0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix}_{2\times 3}$

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



Compute, h_1, y_1
h_2, y_2

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

$y_t = W_{hy} h_t$

① $y_t =$

$$h_t = \tanh\left(\begin{bmatrix} 0.1 & 0.4 & 0.0 \\ -0.2 & 0.3 & 0.2 \\ 0.05 & -0.1 & 0.2 \end{bmatrix}_{3\times 3} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{3\times 1} + \begin{bmatrix} 0.5 & -0.3 \\ 0.8 & 0.2 \\ 0.1 & 0.7 \end{bmatrix}_{3\times 2} \begin{bmatrix} 1 \\ 2 \end{bmatrix}_{2\times 1}\right)$$

$$= \tanh\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}_{3\times 1} + \begin{bmatrix} -0.1 \\ 1.2 \\ 0.9 \end{bmatrix}_{3\times 1}\right)$$

$$= \tanh\begin{bmatrix} -0.1 \\ 1.2 \\ 0.9 \end{bmatrix} = \begin{bmatrix} -0.099 \\ 0.833 \\ 0.716 \end{bmatrix}_{3\times 1}$$

$$y_t = \begin{bmatrix} 1.0 & -1.0 & 0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix}_{2\times 3} \begin{bmatrix} -0.099 \\ 0.833 \\ 0.716 \end{bmatrix}_{3\times 1}$$

$$= \begin{bmatrix} -0.57 \\ 0.008 \end{bmatrix}$$

$$\begin{bmatrix} -0.48 \\ 0.05 \end{bmatrix} = \begin{bmatrix} -0.48 \\ 0.05 \end{bmatrix}_{2\times 1}$$

② $h_t = \tanh\left(\begin{bmatrix} 0.1 & 0.4 & 0.0 \\ -0.2 & 0.3 & 0.2 \\ 0.05 & -0.1 & 0.2 \end{bmatrix}_{3\times 3} \begin{bmatrix} -0.01 \\ 0.83 \\ 0.72 \end{bmatrix}_{3\times 1} + \begin{bmatrix} 0.5 & -0.3 \\ 0.8 & 0.2 \\ 0.1 & 0.7 \end{bmatrix}_{3\times 2} \begin{bmatrix} -1 \\ 1 \end{bmatrix}_{2\times 1}\right)$

$$= \tanh\left(\begin{bmatrix} 0.1331 \\ 0.395 \\ \end{bmatrix} + \begin{bmatrix} -0.8 \\ 0.6 \\ \end{bmatrix}_{3\times 1}\right) = \tanh\begin{bmatrix} -0.409 \\ 0.995 \\ 0.3605 \end{bmatrix}$$

$$h_t = \begin{pmatrix} -0.437 \\ 0.259 \\ 0.345 \end{pmatrix}$$

$$y_t = \begin{bmatrix} 1 & -1 & 0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix}_{2\times 3} \begin{bmatrix} -0.437 \\ 0.259 \\ 0.345 \end{bmatrix}_{3\times 1}$$

$$= \begin{Bmatrix} \\ \\ \end{Bmatrix}^{U}$$

Timestamp ②

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

$$= \tanh\left(\begin{bmatrix} 0.1 & -0.4 & 0.0 \\ -0.2 & 0.3 & 0.2 \\ 0.05 & -0.1 & 0.2 \end{bmatrix} \begin{bmatrix} -0.029 \\ 0.83 \\ 0.716 \end{bmatrix} + \begin{bmatrix} 0.5 & -0.3 \\ 0.8 & 0.2 \\ 0.1 & 0.4 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right)$$

$$= \tanh\left(\begin{pmatrix} 0.3221 \\ 0.112 \\ 0.05525 \end{pmatrix} + \begin{pmatrix} -0.8 \\ -0.6 \\ 0.3 \end{pmatrix} \right) = \tanh\begin{pmatrix} -0.4779 \\ 0.1188 \\ 0.35525 \end{pmatrix}$$

$$h_t = \begin{bmatrix} -0.443 \\ -0.188 \\ 0.370 \end{bmatrix}$$

$$h_t = \begin{bmatrix} -0.4434 \\ -0.1845 \\ 0.370 \end{bmatrix}_{3\times 1}$$

$$y_t = W_{hy} h_t$$

$$= \begin{bmatrix} 1 & -1 & 0.5 \\ 0.5 & 0.5 & -0.5 \end{bmatrix}_{2\times 3} \begin{bmatrix} -0.4434 \\ -0.1845 \\ 0.370 \end{bmatrix}_{3\times 1}$$

$$= \begin{bmatrix} -0.0885 \\ -0.49415 \end{bmatrix}_{2\times 1}$$

Example:

$x_t = [0.5, -0.1]^T$

$h_{t-1} = [0, 0.1]^T$

$c_{t-1} = [0.2, -0.2]^T$

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

$w_{hi} = \begin{bmatrix} 0.1 & 0.2 \\ -0.2 & 0.5 \end{bmatrix}$

$w_{xf} = \begin{bmatrix} -0.4 & 0.2 \\ 0.5 & 0.3 \end{bmatrix}$

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

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

$w_{ho} = \begin{bmatrix} 0.15 & 0.05 \\ 0.1 & -0.2 \end{bmatrix}$

$w_{xg} = \begin{bmatrix} -0.5 & 0.4 \\ 0.2 & 0.3 \end{bmatrix}$

$w_{hg} = \begin{bmatrix} 0.2 & 0.1 \\ -0.1 & 0.5 \end{bmatrix}$

$h_1 = ?$

$c_1 = ?$

$\odot$ → element wise multiplication.

Equations:

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

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

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

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

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

$= \tanh(w_{hs} h_{t-1} + w_{xs} x_t)$

$g_t = \sigma(w_{hg} h_{t-1} + w_{xg} x_t)$

$= \sigma\left(\begin{bmatrix} 0.05 & -0.1 \\ 0.3 & 0.3 \end{bmatrix}\begin{bmatrix} 0 \\ 0.1 \end{bmatrix} + \begin{bmatrix} -0.4 & 0.2 \\ 0.2 & 0.5 \end{bmatrix}\begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix}\right)$

$= \sigma\left(\begin{bmatrix} 0 & -0.01 \\ 0 & 0.03 \end{bmatrix} + \begin{bmatrix} -0.9 \\ \end{bmatrix}\right)$

$= \sigma\left(\begin{bmatrix} -0.01 \\ 0.03 \end{bmatrix} + \begin{bmatrix} 0.22 \\ -0.15 \end{bmatrix}\right)$

$= \sigma\left(\begin{bmatrix} -0.23 \\ -0.12 \end{bmatrix}\right) = \begin{bmatrix} 0.44 \\ 0.42 \\ 0.53 \end{bmatrix}$

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

$= \sigma\left(\begin{bmatrix} 0.1 & 0.2 \\ -0.2 & 0.5 \end{bmatrix}\begin{bmatrix} 0 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.5 & -0.3 \\ 0.4 & 0.1 \end{bmatrix}\begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix}\right)$

$= \sigma\left(\begin{bmatrix} 0.02 \\ 0.05 \end{bmatrix} + \begin{bmatrix} 0.28 \\ 0.18 \end{bmatrix}\right) = \sigma\begin{bmatrix} 0.53 \\ 0.23 \end{bmatrix}$

$= \begin{bmatrix} 0.57 \\ 0.54 \end{bmatrix}$

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

$= \sigma\left(\begin{bmatrix} 0.15 & 0.05 \\ 0.1 & -0.2 \end{bmatrix}\begin{bmatrix} 0 \\ 0.1 \end{bmatrix} + \begin{bmatrix} 0.3 & 0.25 \\ 0.2 & 0.2 \end{bmatrix}\begin{bmatrix} 0.5 \\ -0.1 \end{bmatrix}\right)$

$= \sigma\left(\begin{bmatrix} 0.005 \\ -0.02 \end{bmatrix} + \begin{bmatrix} 0.125 \\ 0.08 \end{bmatrix}\right) = \sigma\begin{bmatrix} 0.13 \\ 0.06 \end{bmatrix}$

$= \begin{bmatrix} 0.53 \\ 0.51 \end{bmatrix}$ 0.46

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

$= \tanh\left(\begin{bmatrix} 0.2 & 0.1 \\ -0.1 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} + \begin{bmatrix} -0.5 & 0.7 \\ 0.2 & 0.7 \end{bmatrix} \begin{bmatrix} 0.7 \\ -0.4 \end{bmatrix} \right)$

$= \tanh\left(\begin{bmatrix} 0.1 \\ 0.5 \end{bmatrix} + \begin{bmatrix} -0.29 \\ 0.07 \end{bmatrix} \right) = \tanh \begin{bmatrix} -0.28 \\ 0.12 \end{bmatrix}$

$= \begin{bmatrix} -0.27 \\ 0.12 \quad 0.13 \end{bmatrix}$

$C_t = f_t \odot C_{t-1} + i_t \odot g_t$

$= \begin{bmatrix} 0.49 \\ 0.53 \end{bmatrix} \odot \begin{bmatrix} 0.2 \\ -0.2 \end{bmatrix} + \begin{bmatrix} 0.57 \\ 0.54 \end{bmatrix} \odot \begin{bmatrix} -0.27 \\ 0.13 \end{bmatrix}$

$= \begin{bmatrix} 0.088 \\ -0.106 \end{bmatrix} + \begin{bmatrix} -0.1539 \\ 0.0702 \end{bmatrix}$

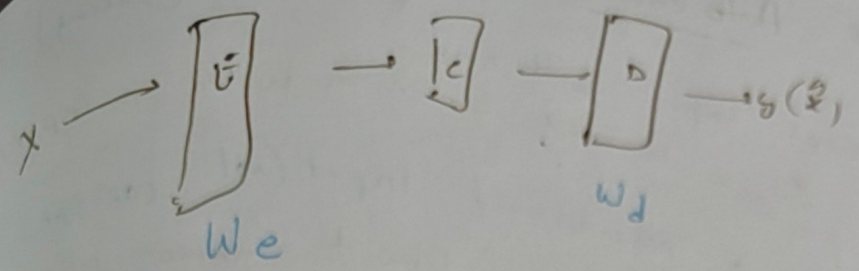
$= \begin{bmatrix} -0.06 \\ -0.03 \end{bmatrix}$

$\tanh(C_t) = \begin{bmatrix} -0.059 \\ -0.029 \end{bmatrix}$

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

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

$= \begin{bmatrix} -0.03 \\ -0.01 \end{bmatrix}$



perform backpropagation from loss $\left[L = \frac{1}{2}\|x - \hat{x}\|^2\right]$ to update weights → which helps us learn

$\frac{\partial L}{\partial We}$, $\frac{\partial L}{\partial W_d}$

Eg: $x \in \mathbb{R}^2 \Rightarrow x = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$

$h \in \mathbb{R}^1$ ← central vector.

$We = 1 \times 2$, $W_d = 2 \times 1$

$= [0.5 \quad -1.0]$ $\begin{bmatrix} 1.0 \\ 0.5 \end{bmatrix}$

loss = MSE

$= \frac{1}{2} \|\hat{x} - x\|^2$

Forward pass

encoder: $h = We\, x$ $\begin{bmatrix} 2 \\ 0 \end{bmatrix}_{2\times1}$ $[0.5 \quad -1.0]_{1\times2} = \begin{bmatrix} 1 & -2 \\ 0 & -0 \end{bmatrix}$

$\hat{x} = [0.5 \quad -1.0]_{1\times2} \begin{bmatrix} 2 \\ 0 \end{bmatrix}_{2\times1} = [1]$

decoder: $\hat{x} = W_d\, h$

$= [0.5]_{2\times1}$ $[1]_{1\times1} = \begin{bmatrix} 0.5 \\ \end{bmatrix}_{2\times1}$

$L = \frac{1}{2} \|x - \hat{x}\|^2$

$= \frac{1}{2} \left[(2-1)^2 + (0-0.5)^2 \right] = \frac{1}{2} 1 + 0.25 = \frac{1.25}{2} = 0.625$

$$\alpha_{t,j} = \frac{\exp\left(\text{score\_fn}(s_{t-1}, h_j)\right)}{\sum_{j=1}^{T} \exp\left(\text{score\_fn}(s_{t-1}, h_j)\right)}$$

we can come up with diff scoring fns (not just dot product like above)

example

$h_1 = [1, 0, 1]$

$h_2 = [0, 1, 1]$

$h_3 = [1, 1, 0]$

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

second fn = dot product

$$\alpha_{t,j} = \frac{\exp\left(\text{score\_fn}(s_{t-1}, h_j)\right)}{\sum_{j=1}^{T} \exp\left(\text{score\_fn}(s_{t-1}, h_j)\right)}$$

$(T = 3)$

$$c_t = \sum_{j=1}^{T} \alpha_{t,j} h_j$$

$$= \alpha_{t,1} h_1 + \alpha_{t,2} h_2 + \alpha_{t,3} h_3$$

$$\alpha_{t,1} = \frac{\exp\left(\text{score\_fn}(s_{t-1}, h_1)\right)}{\exp\left(\text{score}(s_{t-1}, h_1)\right) + \exp\left(\text{score}(s_{t-1}, h_2)\right) + \exp\left(\text{score}(s_{t-1}, h_3)\right)}$$

$$= \frac{\exp\left([1, 0, 1]\,[1, 0, 1]\right)}{\exp\left([1, 0, 1]\,[1, 0, 1]\right) + \exp\left([1, 0, 1]\,[0, 1, 1]\right) + \exp\left([1, 0, 1]\,[1, 1, 0]\right)}$$

$$= \frac{\exp(2)}{\exp(2) + \exp(1) + \exp(1)}$$

$$= \frac{7.389}{7.389 + 2.718 + 2.718}$$

$$= 0.587$$

$$\alpha_{t,2} = \frac{\exp(1)}{\exp(2)+\exp(1)+\exp(1)} = \frac{2.718}{12.6} = 0.2$$

$\alpha_{t,3} = 0.2$

$$C_t = \alpha_{t-1,1} h_1 + \alpha_{t,2} h_2 + \alpha_{t,3} h_3$$

$$= 0.5(0,1) + 0.2(0,1,1) + 0.2(1,1,0)$$

$$= (0.5, 0, 0.5) + (0.2, 0.2, 0.2) + (0.2, 0.2, 0)$$

$$= (0.77, 0.4, 0.77)$$

self attention :

How it different from (RNN attention)

Steps

① create → query vector (q_i)
key vector (k_i)
value vector (v_i)
} From encoder's input vector (x_i)

↳ by initializing weight matrices

② calculate
dotproduct (q_1, k_1)
dotproduct (q_1, k_2)
} similarity with all ?

→ use scaled dot product

③ softmax on dotproduct

example :

compute z_1, z_2 for :

I/P : Playing outside

① ~~word vec~~

Playing → $q_1 = [0.21 \quad 0.04 \quad 0.62 \quad 0.36]^T$

$k_1 = [0.31 \quad 0.84 \quad 0.463 \quad 0.53]^T$

$V_1 = [0.36 \quad 0.83 \quad 0.1 \quad 0.38]^T$

outside → $q_2 = [0.1 \quad 0.14 \quad 0.86 \quad 0.86 \quad 0.44]^T$

$k_2 = [0.45 \quad 0.94 \quad 0.35 \quad 0.58]^T$

$V_2 = [0.31 \quad 0.36 \quad 0.19 \quad 0.72]$

① After softmax:

[0.228956 0.225347 0.300122 0.245575]

[0.36 0.83 0.1 0.8]

= [0.08243856 0.18703801 0.03001224 0.09329119]

② After softmax:

[0.23042 0.225685 0.28368 0.25021]

· [0.31 0.36 0.09 0.72]

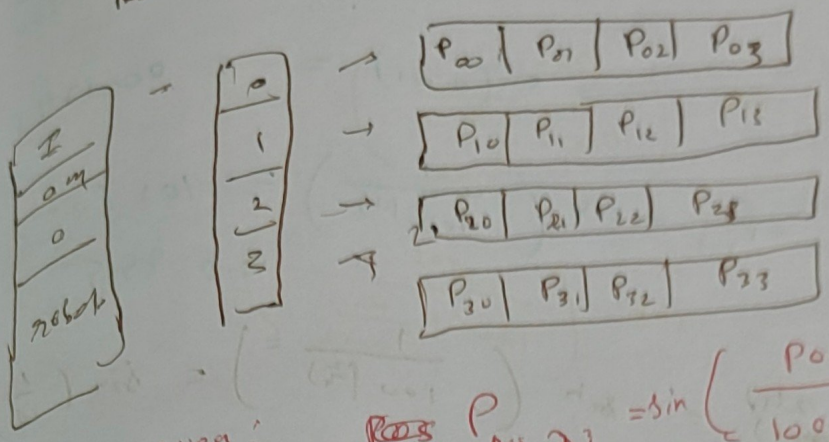
→ [0.073250 0.082686 0.053[illegible] 0.180154]

Z_1 = [0.376 0.[illegible] 0.14 0.54]

Z_2 = [0.30 0.54 0.[illegible] 0.54]

I am a robot

$P_E : d = 4 \Rightarrow \mathbb{R}^4$ (using 100 here but it's 1000)



formula:

even: $P_{pos,2i} = \sin\left(\frac{pos}{100^{\frac{2i}{d_{em}}}}\right)$

odd $= P_{pos,2i+1} = \cos\left(\frac{pos}{1000^{\frac{2i}{d_{em}}}}\right)$

$i = \frac{d}{2} = \frac{4}{2} = 2$

$i = 0$

$P_{pos,2(0)} = \sin\left(\frac{pos}{100^{\frac{2(i)}{dem}}}\right)$

$= \sin(pos,) \Rightarrow \sin(0) = 0$

$P_{pos,2(0)+1} = \cos\left(\frac{pos}{100^{\frac{2(0)}{dm}}}\right) = \cos(0) = 1$

$P_{00} = 0, \quad P_{01} = 1$

$P_{pos,2(1)} = \sin\left(\frac{0}{100^{\frac{2}{4}}}\right) = \sin(0)$

$\left(\frac{0}{100^{\frac{2}{4}}}\right) = \cos(1) = 1$

$P_{pos,2(0)-1} = \cos$

$P_{02} = 0, \quad P_{03} = 1$

$P_{pos,2(2)} = \sin\left(\frac{0}{10^{-1}}\right) = \sin(0)$

DOS = 1

i = 0

$P_{i,2(0)} = \sin\left(\frac{1}{1000\left(\frac{2\pi}{4}\right)}\right) = \frac{1}{100}$

$P_{i,2(0)} = \sin(0.01) = 0.000174$

$P_{i,2(0)+1} = P_{i,1} = \cos\left(\frac{1}{100(1)}\right) = \cos(0.01)$

$= 0.999$

i = 1

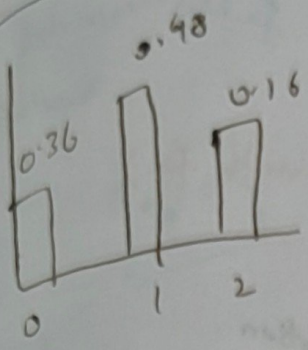
$P_{i,2(1)} = \sin\left(\frac{1}{100\left(\frac{2\pi}{4}\right)}\right) = \sin\left(\frac{1}{10}\right)$

$= \sin(0.1) = 0.01$

$P_{i,2(1)+1} = \cos\left(\frac{1}{10}\right) = \cos(0.1) =$

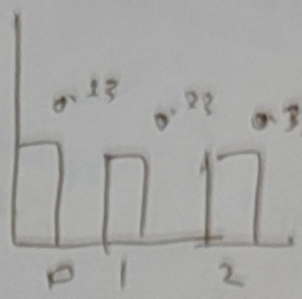
Numerical

P



Binomial dist

Q



uniform dist

distributions

	0	1	2
P(x)	$\frac{9}{25}$	$\frac{12}{25}$	$\frac{4}{25}$
Q(x)	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

Q. compute $D_{KL}(P \| Q)$

$$D_{KL}(P\|Q) = \sum_{x \in X} P(x=x) \ln \frac{P(x=x)}{Q(x=x)}$$

$$= \frac{9}{25} \ln\left(\frac{9/25}{1/3}\right) + \frac{12}{25} \ln\left(\frac{12/25}{1/3}\right) + \frac{4}{25} \ln\left(\frac{4/25}{1/3}\right)$$

$$\approx 0.085296$$

$$D_{KL}(Q\|P) = \sum_{x \in X} Q(x=x) \ln \frac{Q(x=x)}{P(x=x)}$$

$$= 1/3 \ln\left(\frac{1/3}{9/25}\right) + 1/3 \ln\left(\frac{1/3}{12/25}\right) + 1/3 \ln\left(\frac{1/3}{4/25}\right)$$

$$= \frac{1}{3}(-0087) + 1/3(-0.374) + 1/3(0.724)$$

$$= -0.0261 - 0.1122 - 0.2172 = 0.08745$$

$$\Rightarrow -0.0885$$ ✗

Note: $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$