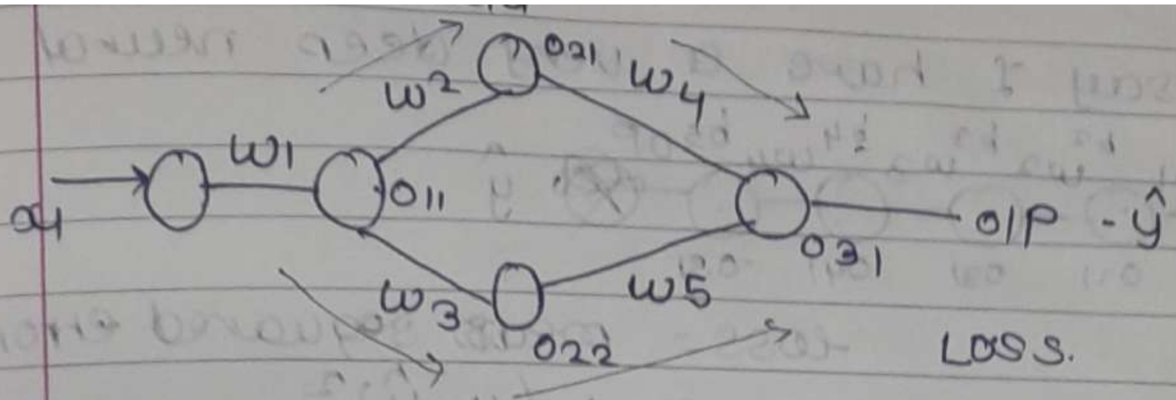


$$w_{1new} = w_{1old} - \eta \frac{\partial L}{\partial w_{1old}}$$ -- ~~wait~~ weigh update formula

$$\frac{\partial L}{\partial w_{1old}} = \frac{\partial L}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1old}}$$

* here there are 2 roots that you can basically see.

① loss is dependent on O31, O31 is dependent on O21, O21 → O11, O11 → w1

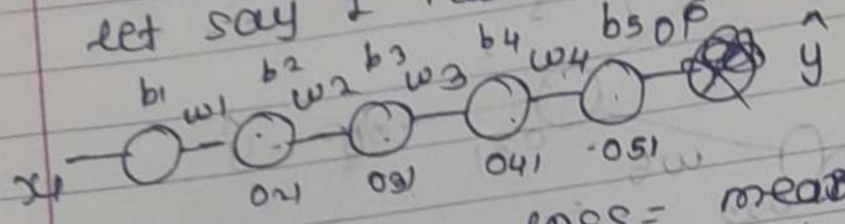
② loss is dependent on O31, O31 is —"— O22, O22 → O11 & O11 → w1

now

$$\frac{\partial L}{\partial w_{1old}} = \left[\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{21}} \times \frac{\partial O_{21}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1old}}\right] + \left[\frac{\partial L}{\partial O_{31}} \times \frac{\partial O_{31}}{\partial O_{22}} \times \frac{\partial O_{22}}{\partial O_{11}} \times \frac{\partial O_{11}}{\partial w_{1old}}\right]$$

③ <u>Vanishing gradient problem</u>

let say I have a very deep neural n/w



loss = mean squared error

$$= \frac{1}{2}(y-\hat{y})^2$$

- Initially we discussed about sigmoid activation f^n

To update w_1

$$w_{1new} = w_{1old} - \frac{\partial L}{\partial w_{1old}}$$

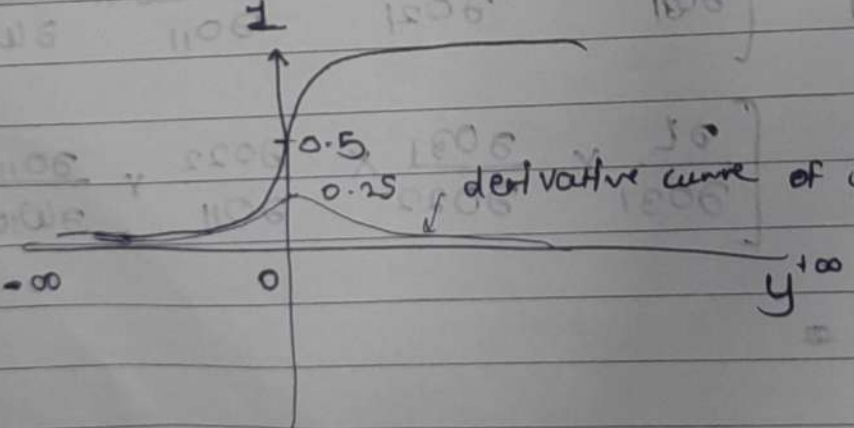
$$\frac{\partial L}{\partial w_{1old}} = \frac{\partial L}{\partial O5} \times \frac{\partial O5}{\partial O41} \times \frac{\partial O41}{\partial O31} \times \frac{\partial O31}{\partial O21} \times \frac{\partial O21}{\partial w_{1old}} \quad -①$$

- we are using here sigmoid activation f^n

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

o/p is 0 or 1

$\geq 0.5 = 1$ & $< 0.5 = 0$



- One Imp property of sigmoid f^n whenever we try to find out derivative of sigmoid it will be ranging from 0 to 0.25

Imp

$0 \leq \sigma(y) \leq 0.25$

- bcoz of this what will happen.
- In each and every value sigmoid is getting used.

e.g. $0.51 = \sigma[(0.41 \times w_4) + b]$

on top of it we applying an activation fn.

- So in backpropagatn when I'm finding the derivative of the value will alway be ranging btn 0 to 0.25

e.g. in eqn ② we will be getting.

$$\frac{\partial L}{\partial w_{1\,new}} = 0.25 \times 0.15 \times 0.10 \times 0.05 \times 0.02$$

- ~~It will~~ The values will keep on decreasing bcoz as we are going to this chain rule, as we going to calculate derivative it is alway going to reduce untill we go the end of the chain
- now what will happen bcoz of this since we are multiplying with smaller values we will be getting very small value.

now bcoz of this small no.

$w_{new} = w_{old} - \eta\,(\text{small no}) \rightarrow$ no change weights.

- at one point of time $w_{new} \approx w_{old}$

It will hardly change and if it is hardly change then weights are not getting updated & this situation where weights are not getting updated or it just getting updated with small value. this problem is basically called as

* a $\boxed{\text{vanishing gradient problem}}$

Then use another activatn fn that is why we came up with another activatn fn. there are so many action fn that we r going to learn.

1. sigmoid
2. Tanh
3. Relu
4. Leaky Relu
5. pre relu.

(1) sigmoid

→ frequently used in the beginning of DL.

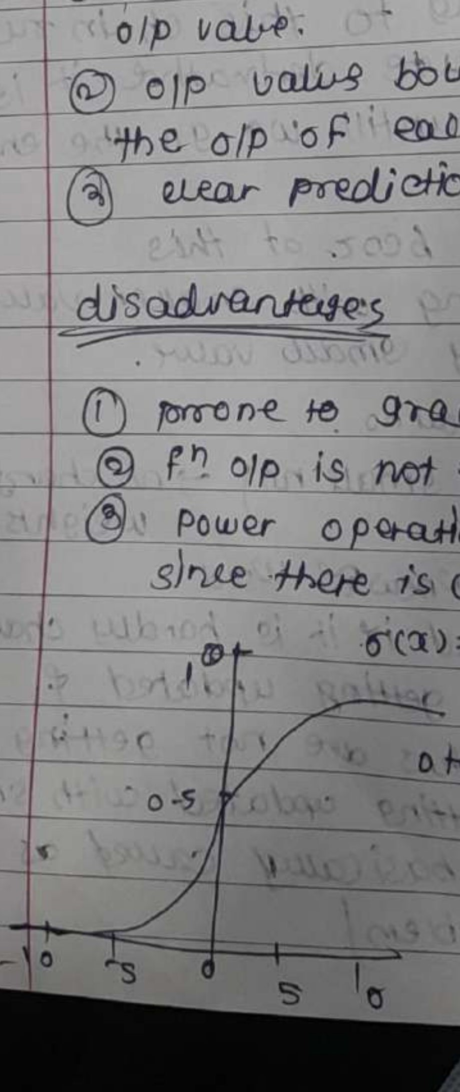
<u>advantages</u>

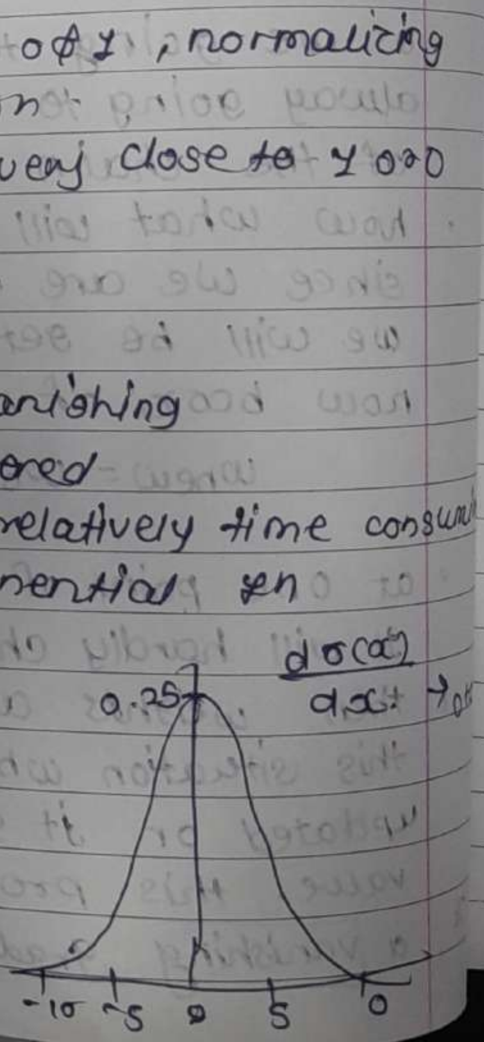
1. smooth gradient, preventing "Jumps in o/p value.
2. o/p values bound betn 0 & 1, normalizing the o/p of each neuron
3. clear predictions, i.e. very close to 1 or 0

<u>disadvantages</u>

1. prone to gradient vanishing
2. fn o/p is not zero centered
3. Power operations are relatively time consuming since there is an exponential fn

$\sigma(x) = \frac{1}{1+e^{-x}}$ $\qquad \frac{d\sigma(x)}{dx} \to 0$ to 0.25

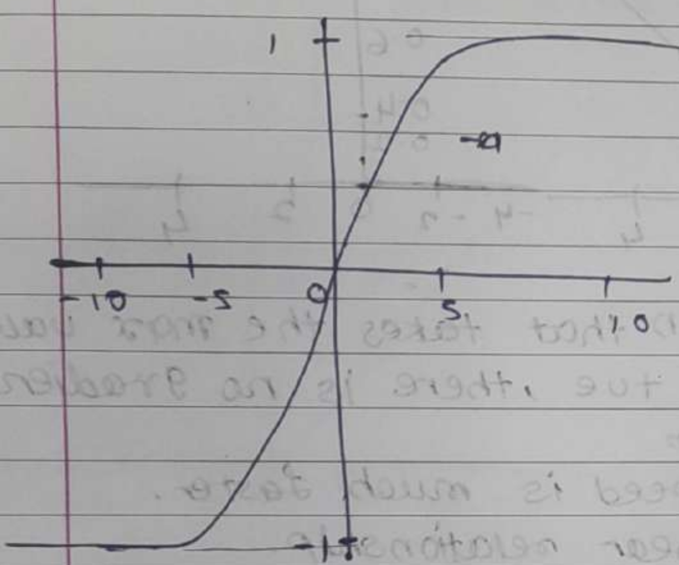




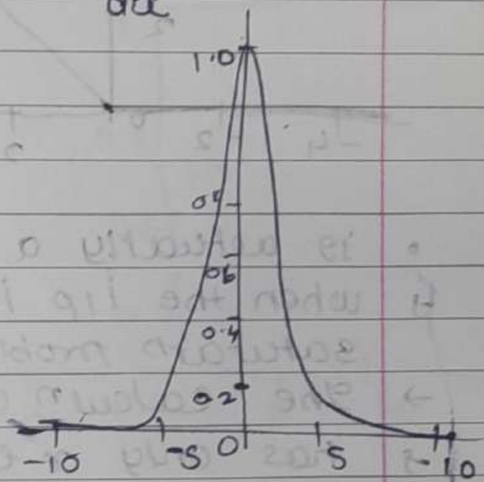
② **tanh fn** (hypebolic tanjent fn

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad -1 \text{ to } +1$$

value ranges from -1 to +1



$\frac{d \tanh(x)}{dx}$ ← 0 to 1



- does this prevent vanishing gradient problem?
- still there will a issue if we go on constructing a deep neural n/w or deep deep neural n/w at one point of time there may be chances that vanishing G. P. still exist. bcoz of that we also not using tanh activan fn

③ The o/p interval of tanh is 1. & the whole fn is 0-centric, which is better than sigmoid

In general binary classifn problem, the tanh fn is used for the hidden layer & sigmoid fn is used for o/p layer

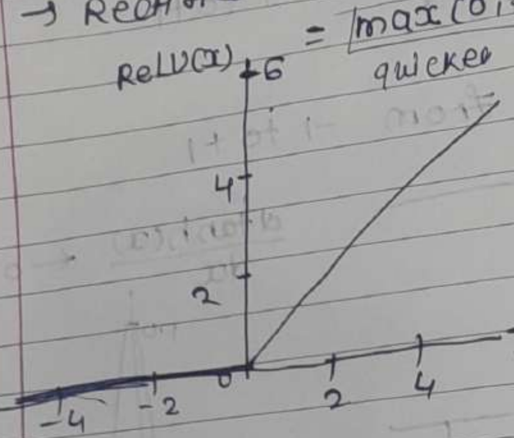
(8) ReLU fn

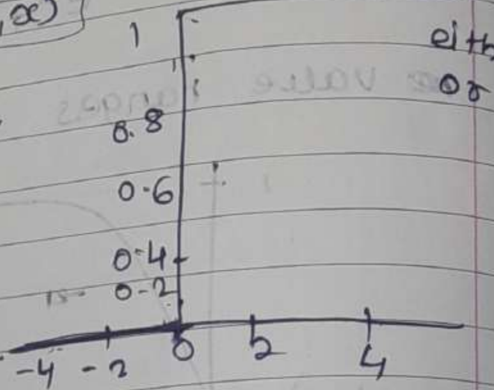
→ most popular using activatn fn

→ Rectified linear unit.

$ReLU(x) = \boxed{max(0, x)}$

quicker





$\frac{d\,ReLU(x)}{dx}$ = either 0 or 1

• is actually a fn that takes the max value

→ when the i/p is +ve, there is no gradient saturatn problem

→ The calcun speed is much faster.

→ has only ~~a~~ linear relationship.
whether it is forward or backword.

→ it is much faster than sigmoid & tanh.
(sigmoid & tanh need to calculate the exponent, which will be slower)

disadvan

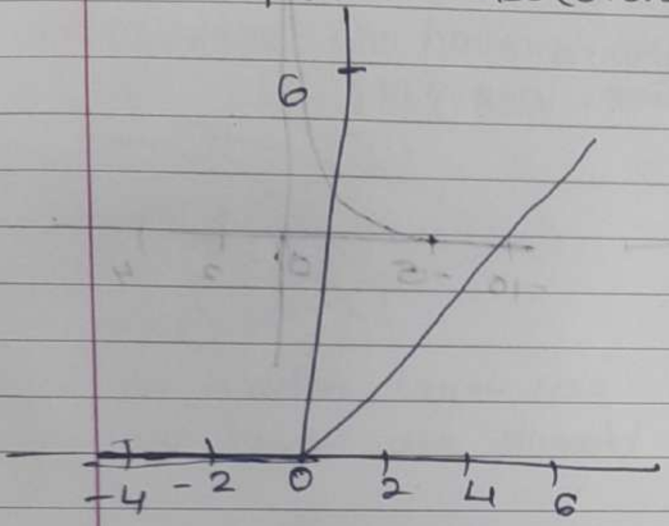
→ when the i/p is -ve. ReLU is completely inactive, which means that once a -ve number is entered, ReLU will die.
In this way, in forward propagatn, it is not a problem. Some areas are sensitive & Some are insensitive.

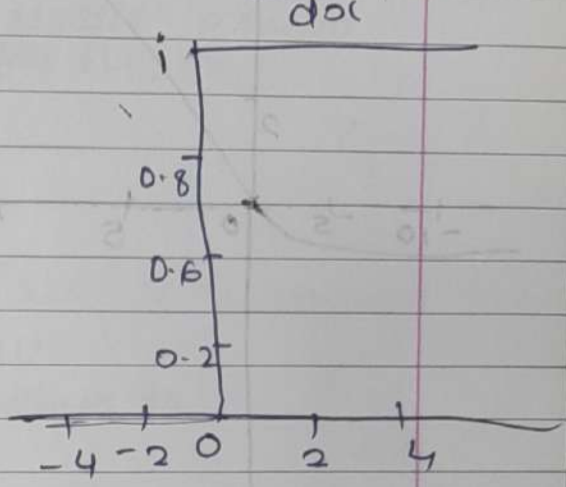
• But in the B.P., if we enter a -ve number the gradient will completely zero, which has the same problem as the sigmoid & tanh fn.

- we find that the o/p of the ReLU fn is either 0 or a +ve number, which means that the ReLU fn is not a 0 centered fn

(4) Leaky ReLU fn

$f(x) = max(0.01x, x)$





- In order to solve the dead ReLU problem, people proposed to set the first half of ReLU 0.01x instead 0.

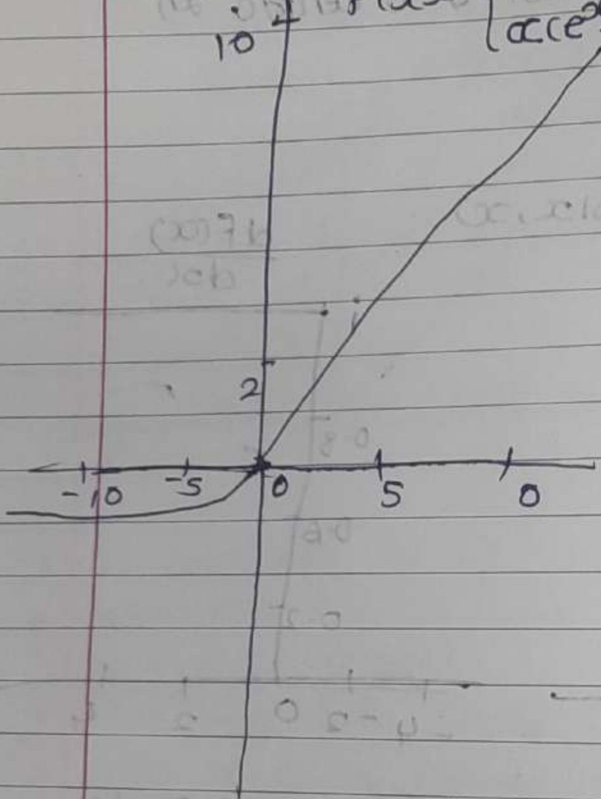
Another intuitive idea is a parameter-based method:

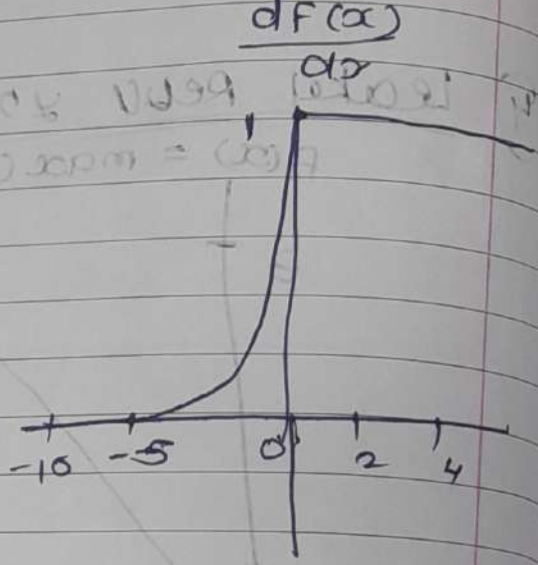
parametric ReLU: $f(x) = max(\text{alpha}\,x, x)$ which alpha can be learned from B.P.

In theory leaky ReLU has all the advantages of ReLU. plus there will be no problems with dead ReLU, but in actual it has not been fully proved that leaky ReLU is always better than ReLU.

(3) ELU (Exponential Linear Units) fn:

$$f(x) = \begin{cases} x, & \text{if } x>0 \\ x(e^{x}-1) & \text{otherwise} \end{cases}$$





ELU is also proposed to solve the problems of ReLU, obviously, ELU has all the advantages of ReLU &

- No Dead ReLU issues
- The mean of the O/P is close to zero, zero centered.

- One small problem is that it is slightly more computationally intensive, similar to leak relu although theoretical better than relu.
- good evidence in practice that ELU is always better than relu.

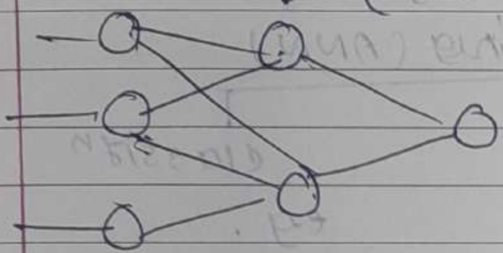
(6) preReLU

$$f(y_i) = \begin{cases} y_i & \text{if } y_i > 0 \\ a_i y_i & \text{if } y_i \le 0 \end{cases}$$

* Technique which Activation fn we should used?

- suppose we have binary classifn problem

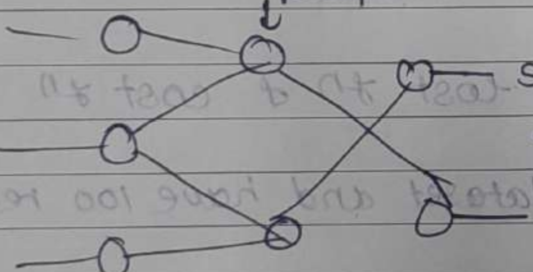
↓HL (ReLU, preReLU, ELU)



- In hidden layer use ReLU
o/p layer we sigmoid activatn fn

multi class classifn
use Sofmax activatn fn in the o/p Layer.

ReLU ↓

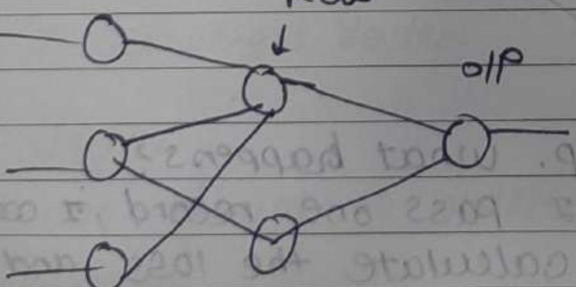


softmax.

- In the cas of Regression problem.

ReLU ↓

o/p



linear activatn fn