ANN → DL

Lecture: 6

ACTIVATION FUNCTION

23rd July 2023.

Date:

* why do we need activation fun^.

→ To achieve non-linearity in the data. (make classification or regression in case of non linear dataset).

→ To activate the neuron (Acts as a gate with triggers when we pass the threshold).

* Types

- linear activation fun^.
  - y = f(x) — x: weighted sum; f: act^ fun^.
  - Binary step act^ fun^.
- non-linear activation fun^.
  - Sigmoid fun^.
  - tanh fun^.
  - Relu
    - Leaky relu
    - Parametric relu
    - ELU
    - SELU
  - Softmax.

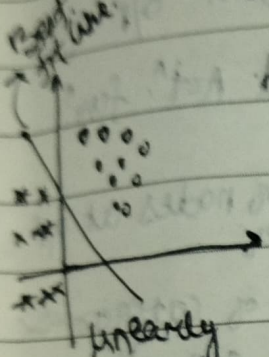
Date:

Note: weighted sum represent the line in 2D space (for linear separable dataset).

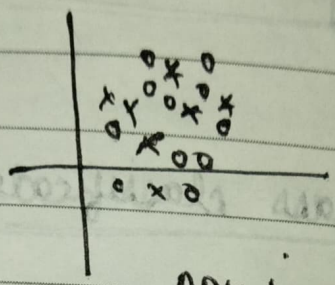
$(x_1 w'_{11} + x_2 w'_{21} + b_{11}) = 0_{12}$

$ax + by + c$.

or $w^T x + b$.



linearly separable dataset.



non linearly separable dataset.

Note: for non-linear separable dataset, weighted sum represents curve in 2D space.

↓ gmp.

*. So, to conclude if dataset is linearly separable then weighted sum alone can be applied on top of regression or classification.

↓ gmp.

*. But if dataset is non linearly separable then we need to pass the weighted sum through some activation fun^n. (non linear act^n fun^n.) so that we are able to implement respective regression/classification use case.

*. Note: Activation fun^n. at o/p node:

Date:

Regression → 1 node/neuron at o/p layer and we use linear act^. fun^.

Binary classification → 1 nodes at o/p layer, Sigmoid Act^. fun^.

Multi class classification → no. of nodes at o/p layer = no. of categories in o/p.
→ Softmax. Act^. fun^.

*. Note: Act^. fun^. for nodes/neuron at hidden layer:

In old days: Sigmoid & tanh

In current time: Relu and its diff^. variants since its so powerful that it replaced the earlier Variants.

*. (Sigmoid & tanh in hidden layer) → linear separable dataset

in hidden layer

Date:

*. (relu & it's variants) → non linear seperable dataset (Regression & classification)

* Sigmoid fun^ } Vanishing Gradients
tanh fun^.

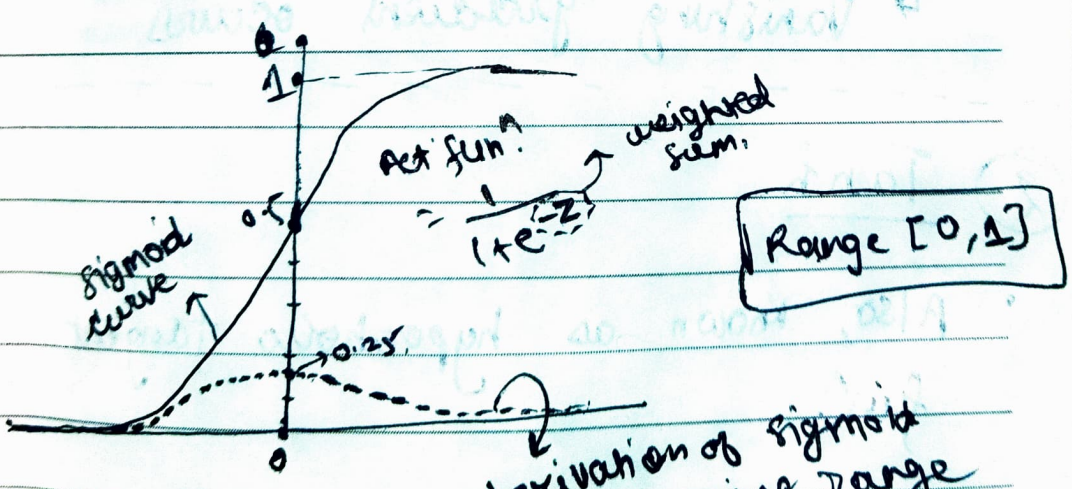
relu & its variants } dying neuron.

*. Properties of Act^ fun^ :

① non-linearity
② Differentiable. (since it will be required to tweak weights in back propagation)
③ computationally inexpensive
④ Zero centered (since then training will be faster same as normalising)
⑤ Non Saturating / (To counter vanishing gradient)
Non-sequezzed

---

① SIGMOID FUN^ :



sequezzed [0-0.25]

This is the squeezing fun^ when. tcs TATA CONSULTANCY SERVICES
we are squeezing values to 0/1 -
This violates the non-sequezzed property

of act^ fun^. Due to this vanishing gradient. This is the reason why we do not use Sigmoid in hidden layer and instead use Relu & its variants.

Date:

- **Advantage:**
  1. Non linear ($\frac{1}{1+e^{-z}}$ and not $y=f(x)=x$)
  2. Differentiable
  3. o/p layer → Binary classification

- **Disadvantage:**
  - Saturated / squeezed fun^.
  - computationally Expensive.
  - non zero centric hence training is slow.
  - Vanishing gradient occurs

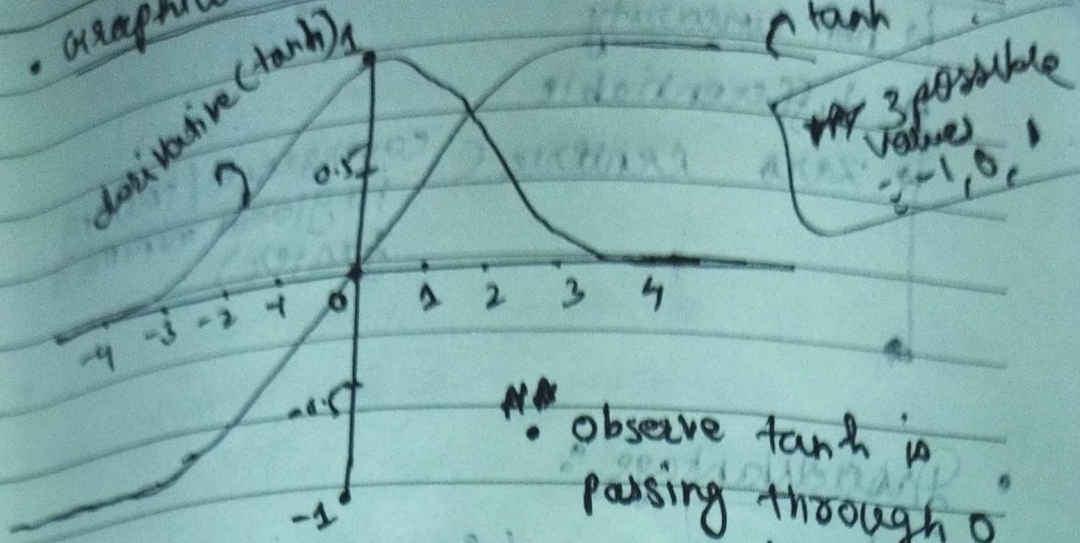
---

(2) Tanh

- Also, known as hyperbolic tangent fun^.

Date:

- Graphical Representation :



- observe tanh is passing through 0 hence it is zero centric Act^n fun^n

- This is saturing in nature since here values are squeezed in the range (-1, +1). In sigmoid it was (0, 1).

i.e, $(-\infty, +\infty) = (-1, +1) \Rightarrow$ tanh.
(provide any value)
$= (0, 1) \Rightarrow$ sigmoid.

- formula :

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

(x: weighted sum; Act^n fun^n; tanh)

$$f(x) = \frac{1}{1+e^{-x}}$$

(sigmoid)

- Range of Derivatives

tanh = (0, 1) and sigmoid = (0, 0.25)

TCS TATA CONSULTANCY SERVICES

Date:

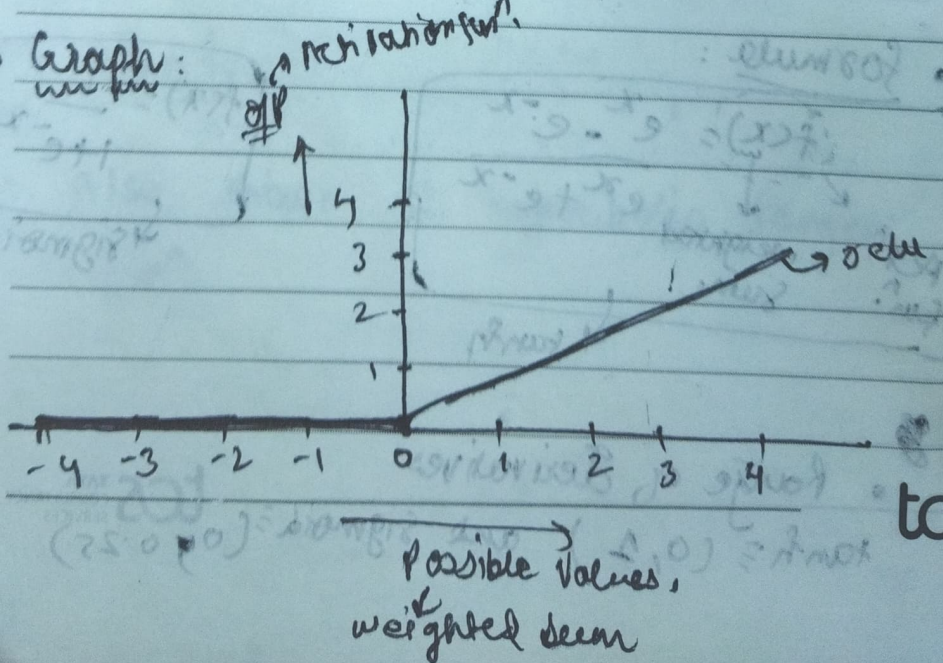
- **Advantage:**
  - non linearity
  - differentiable.
  - zero centeric (convergence / training to global minima is faster)

- **Disadvantage:**
  - Saturated / Sequeezing nature.
  - Vanishing gradient.
  - Computationally Expensive
  - Do not use tanh as activation fun^n at o/p layer for binary classification.

---

③ **RELU:**

- Graph:



Date:

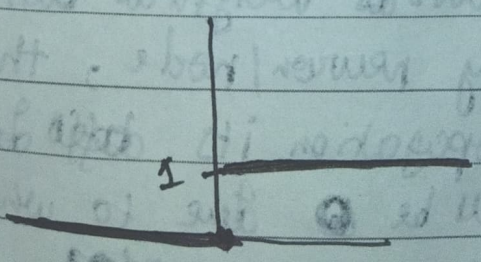
- By observing the graph we can say that RELU produces o/p as 0 for -ve values, whereas it gives same value of +ve values.

- We might think that it is linear in nature. But in actual when we are combining multiple RELU then it becomes non-linear in nature. This is the reason why we use it in hidden layers whereas, others in o/p layer & also since RELU solves vanishing gradient problem.

- Formula:

$$f(x) = \max(0, x)$$

- This is non squeezing.
- Derivative will be either 0 or 1 because of which a problem occurs called dying neuron and then we use its variant leaky relu.



Date:

- **Advantage:**
  - Non linear (on adding multiple relu).
  - non-saturated in the region
  - computational inexpensive
  - Handles vanishing gradient problem
  - Convergence to global minima is fastest

Relu > tanh > Sigmoid

order of convergence

(A) Hence Sigmoid is slowest.

- **Disadvantage:**
  - **Dying neuron Problem.**

* So differentiation of relu act? fun. will be $x \leq 0 \Rightarrow 0$ (weight sum o/p of act. fun.)

$x > 0 \Rightarrow 1$

* So, when there is any -ve I/P or in other words weighted sum is -ve for any neuron/node, then in back propagation its derivative will be 0 due to which weights will not get updated for that neuron and

hence there will be no learning.
This is called Dying neuron problem.

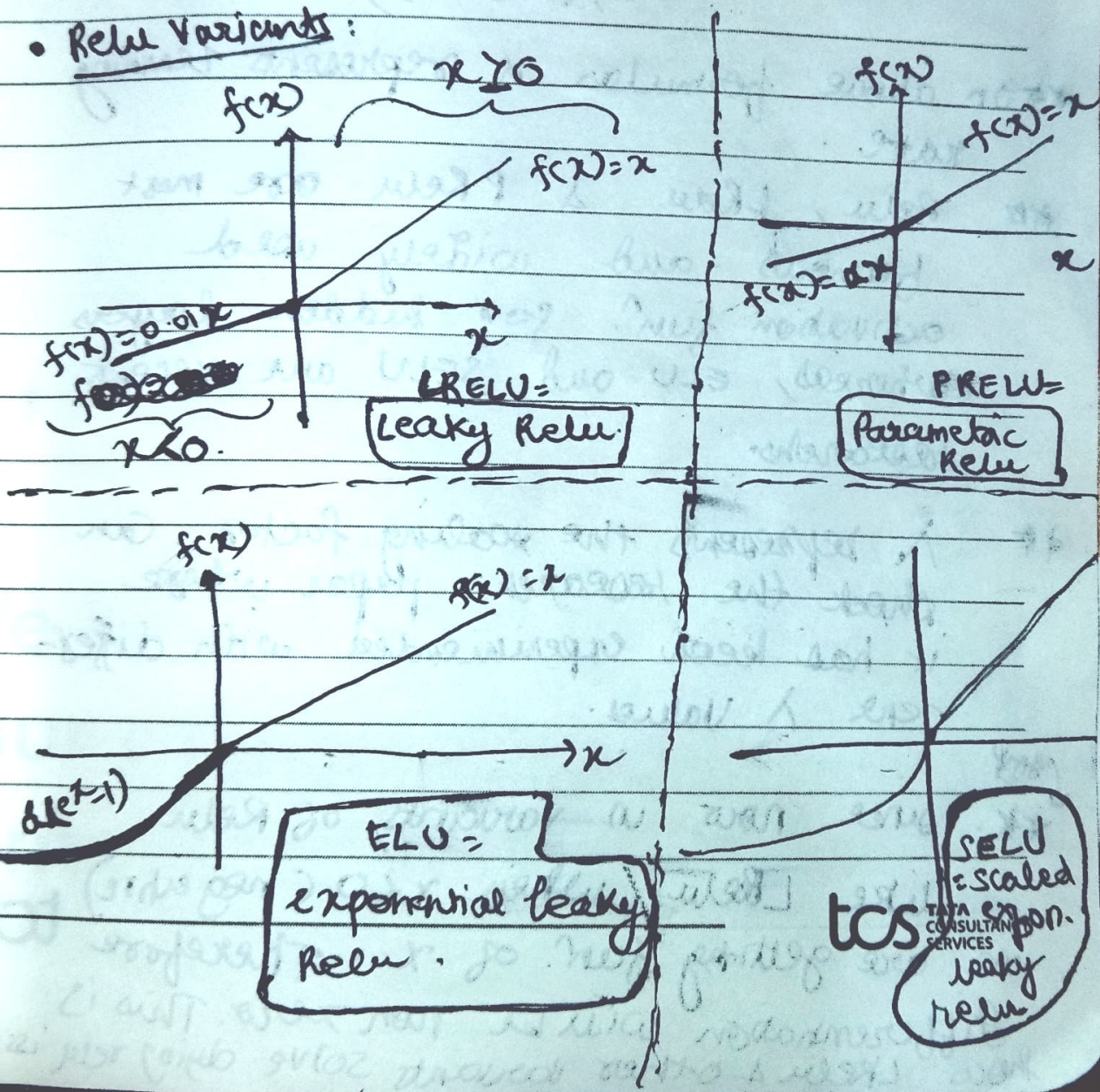
$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W} \rightarrow 0$ (zero)

$\frac{\partial L}{\partial W}$ → chain rule.

$W_{new} = W_{old}$.

↳ Another disadvantage is that is: non zero centric. Relu passed from zero but it is not symmetric that's why non-zero centric. But still it converges faster since we use Batch normalization.

- **Relu Variants:**



- **Relu Variants formula comparison:**

Date:

Relu ⇒ $\max(0, x)$

LRelu ⇒ $\max(0.01x, x)$

PRelu ⇒ $\max(\alpha x, x)$

ELU ⇒ $\begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

SELU ⇒ $\begin{cases} \lambda x & x \geq 0 \\ \lambda\alpha(e^x - 1) & x < 0 \end{cases}$

** In above formulas α represents learning rate.

** Relu, LRelu & PRelu are most famous and widely used activation fun^ for hidden layers whereas, ELU and SELU are recent researchs.

** λ represents the scaling factor. Can check the research paper where it has been experimented with different λ values.

** Since now in variants of Relu like LRelu when $x < 0$ (negative) we are getting fun^ of x. Therefore differentiation will be non zero. This is how LRelu & other variants solve dying relu issue.

Date:

**: Acc. to research value optimal α and λ values are:

$\lambda \approx 1.0507009873\ldots$

$\alpha \approx 1.6732632423\ldots$

**: We use Relu and its variants in the hidden layer since they address the vanishing gradient and dying neuron issue.

**: Can check the derivative graph of different variants of Relu in the ipynb file shared by Sunny.