__/__/____

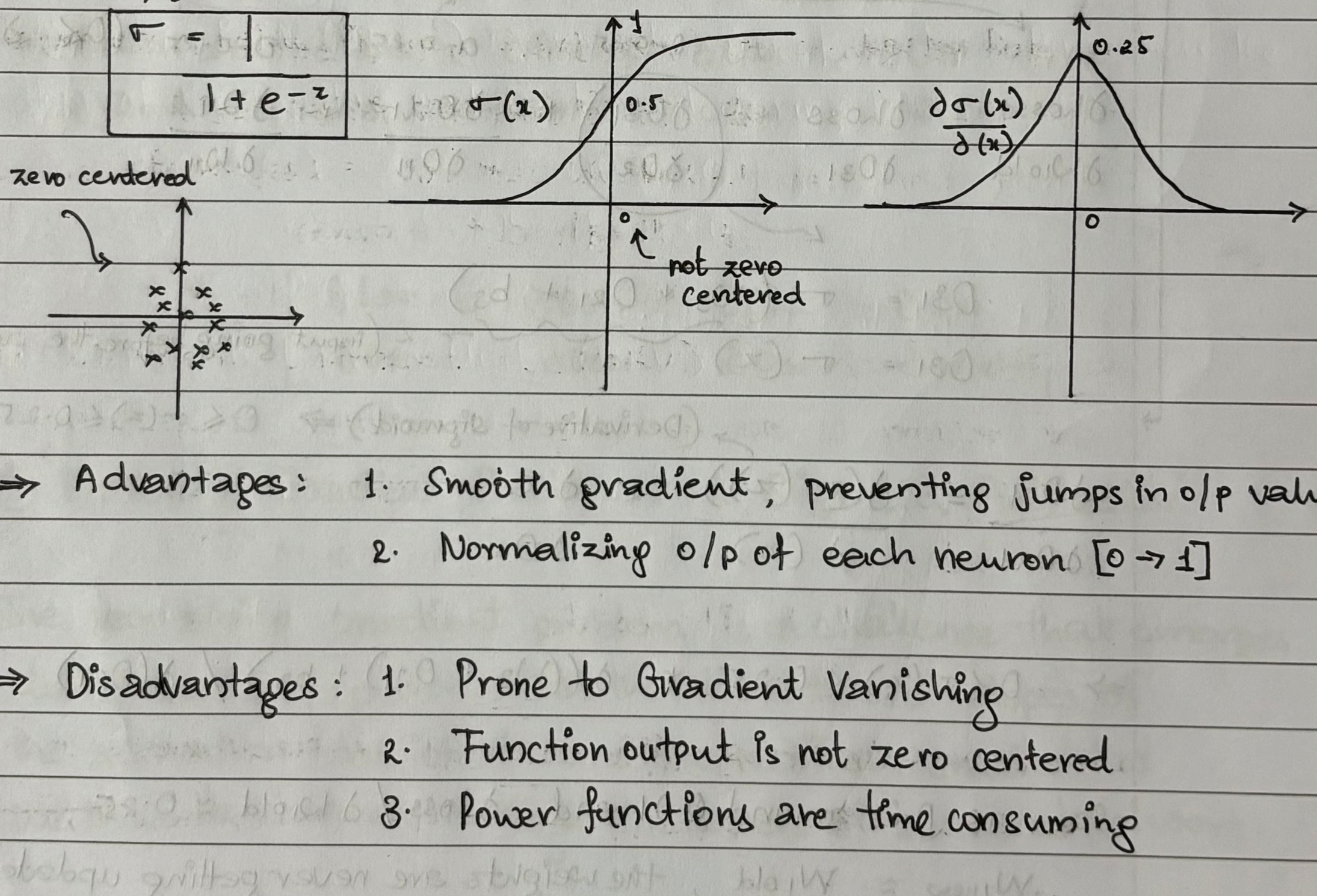
⑤ * Activation Functions *

Activation Functions helps to determine the output of a neural network. These types of functions are attached to each neuron in the network, and determines whether it should be activated or not, based on each neuron's input is relevant for the model's prediction. It introduces non-linearity for linear functions. It also helps to normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

1. Sigmoid Activation Function: Most frequently used act func at the beginning of Deep learning. A smoothing function that is easy to derive.

logistic function

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



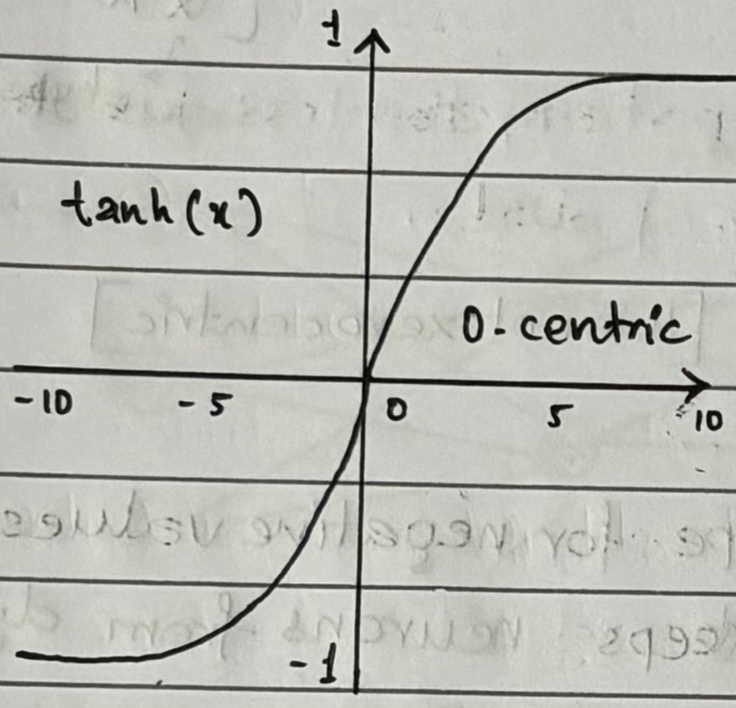
⇒ Advantages: 1. Smooth gradient, preventing jumps in o/p value
2. Normalizing o/p of each neuron [0 → 1]

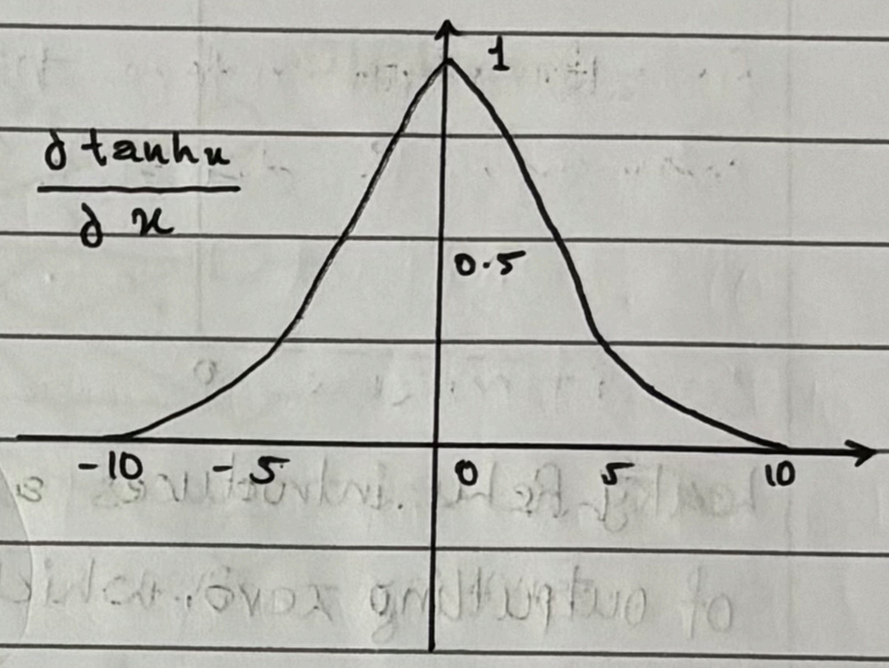
⇒ Disadvantages: 1. Prone to Gradient Vanishing
2. Function output is not zero centered
3. Power functions are time consuming

__/__/____

2. Tanh Activation Function: Hyperbolic Tangent Function is a shifted version of sigmoid, allowing it to strech across the y-axis.

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad \frac{\sinh(x)}{\cosh(x)}$$





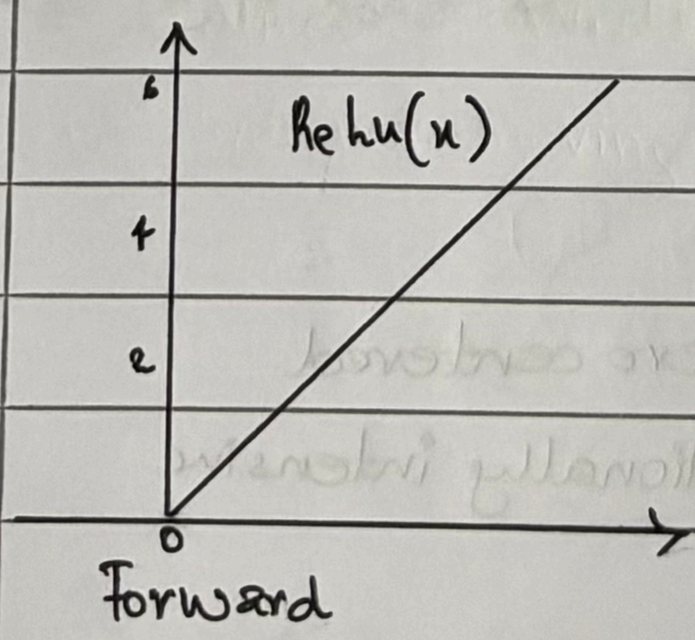
⇒ Advantages: 1. Introduces non-linearity to the model
2. It is centered around zero

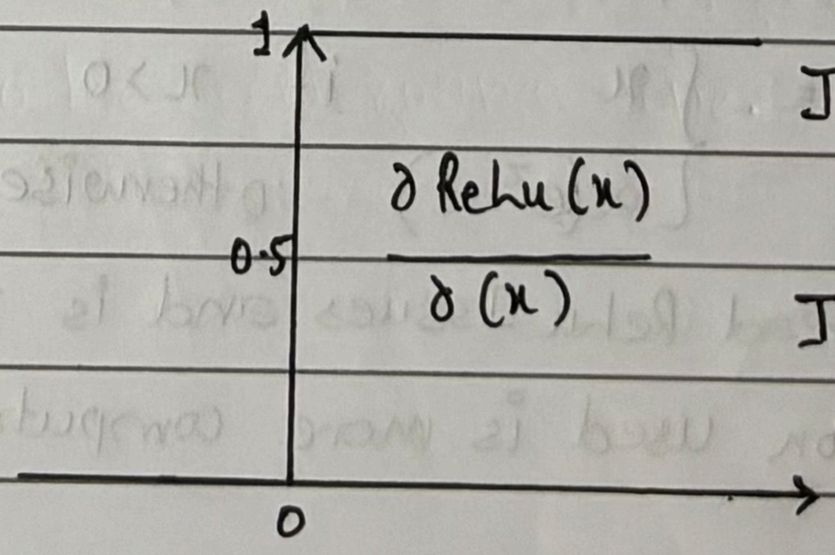
⇒ Disadvantages: 1. Suffers vanishing gradient for large inputs
2. Sensitive to outliers.

3. ReLu Activation Function: Rectified Linear Unit act function has become a default choice due to its simplicity & efficiency.

$$\text{Relu}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad \text{or} \quad \max(0, x)$$

It is a non-linear function which allows the model to learn more complex data patterns. It also avoids vanishing gradient problem.



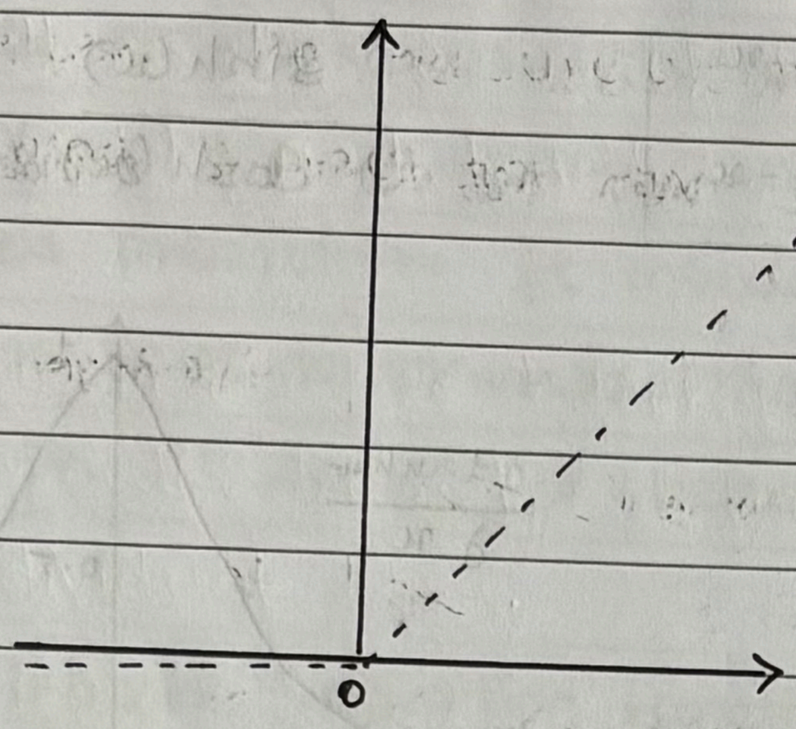


If ∂ of Relu(x) is 0,
$w_{n\,new} \approx w_{n\,old}$ → Dead N
If ∂ of Relu(x) is 1,
weight updation will happe

__/__/__

4. Leaky ReLu and Parametric ReLu Activation Functions

In ReLu, when $\partial(\text{ReLu}(x))$ is 0 → Dead Neuron Problem



$$\text{ReLu}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$

α → hyperparameter

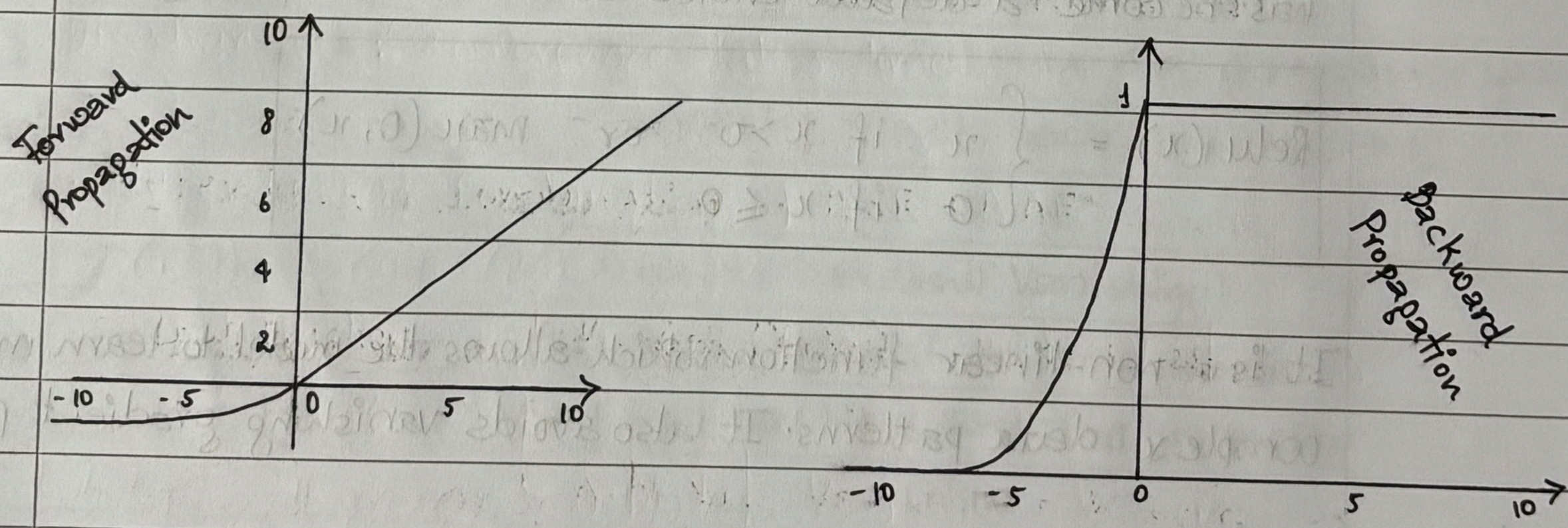
[It is not zero centric]

Leaky ReLu introduces a small slope for negative values instead of outputting zero, which helps keeps neurons from dying.

Parametric ReLu (PReLu) is an extension of Leaky ReLu, where the slope of the negative part is learned during training.

$$\text{PReLu}(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$$

5. Exponential Linear Unit (ELU)



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

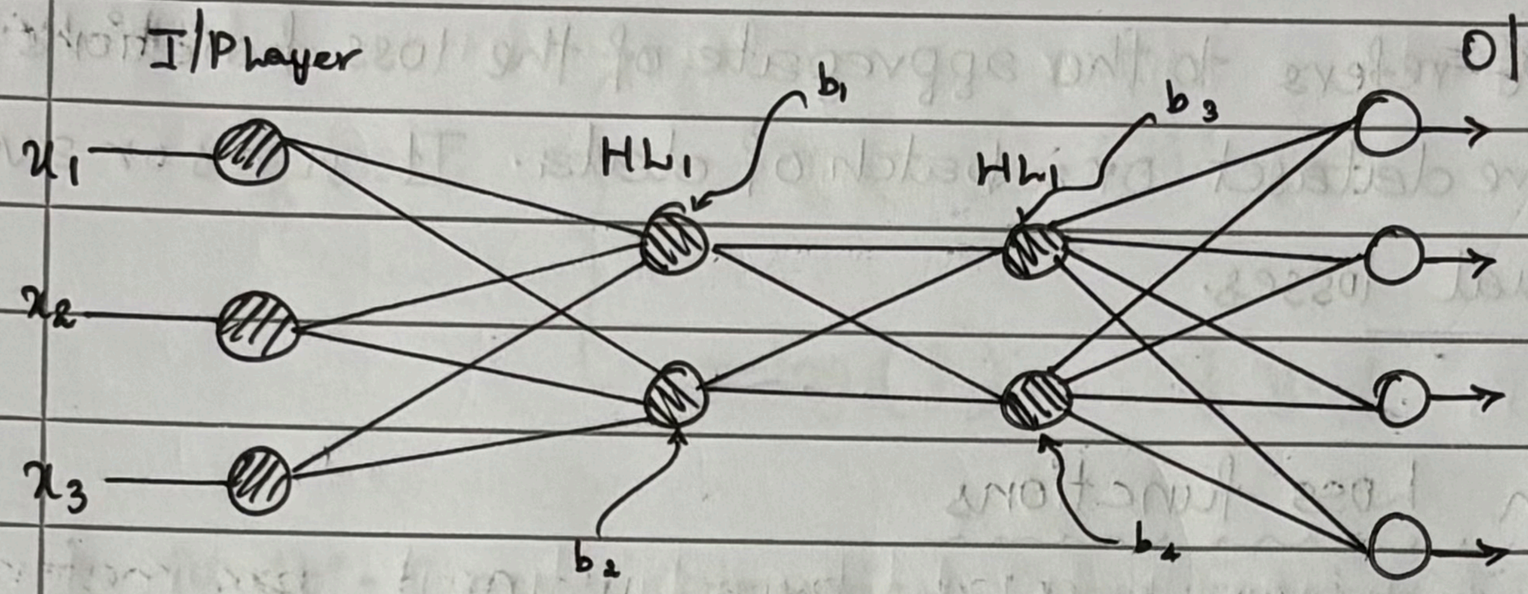
→ Solves Dead ReLu Issues and is zero centered

→ Equation used is more computationally intensive

__/__/____

6. Softmax Activation Function: It is designed to handle multi-class classification problems. Transforms raw output scores from a neural network → probabilities.

Works by squashing the output values of each class into the range of 0 to 1 while ensuring that the sum of probabilities equals 1.



Softmax Activation : $\frac{e^{y_i}}{\sum_{k=0}^{n} e^{y_k}}$

$y_i = 0 * w * b$

⇒ Which Activation Function to use when ?

1. Use Sigmoid or Tanh for hidden layers.

2. If there is vanishing gradient problem use ReLu or any of its varients (PreLu, Leaky ReLu, ELU ...)

3. If it is a binary classification problem use sigmoid.

4. If it is a multi-class classification problem use softmax.

__/__/__

⑥ Loss functions and Cost functions

Loss function measures how far an estimated value is from its true value. It is used to determine which model is better. The measure of the error or penalty associated with model's prediction for a single data instance.

Cost function refers to the aggregate of the loss functions over an entire dataset or a batch of data. It sums or averages the individual losses.

- **Regression Loss functions**

1. Mean Squared Error : $\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$

Advantages: MSE is differentiable, converges faster.

Disadvantages: Not robust to outliers

2. Mean Absolute Error : $\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$

Advantages: Robust to outliers

Disadvantages: Convergance takes time

3. Huber Loss : Combination of MSE + MAE

$$\text{Cost function} = \begin{cases} 1/n \sum_{i=1}^{n}(y_i-\hat{y}_i)^2, & \text{if } |y-\hat{y}| \le \delta \\ \delta|y_i-\hat{y}_i| - 1/2\,\delta^2, & \text{otherwise} \end{cases}$$

(δ: hyperparameter)

4. Root Mean Squared Error : $\sqrt{\sum_{i=1}^{n}\frac{(y_i-\hat{y}_i)^2}{N}}$

__/__/____

- Classification Loss functions

1. Binary Cross Entropy: Also known as log loss used for binary classification

$$\text{log loss} : -y \log(\hat{y}) - (1-y) * \log(1-\hat{y})$$

(y: actual value, $\hat{y}$: predicted value)

$$\text{log loss} : \begin{cases} -\log(1-\hat{y}) & \text{if } y=0 \\ -\log(\hat{y}) & \text{if } y=1 \end{cases}$$

($\hat{y}$) → use sigmoid activation function

2. Multi-Class / Categorical Cross Entropy: Used for multi-class classification problems. Converts categories to One Hot Encoding

$$h(x_i, y_i) = -\sum_{j=1}^{\text{categories}} y_{ij} * \ln(\hat{y}_{ij}), \quad y_{ij} = \begin{cases} 1, & \text{element present} \\ 0, & \text{element not present} \end{cases}$$

($\hat{y}_{ij}$) → We get it using softmax activation function

3. Sparse Categorical Cross Entropy: Similar to Categorical but is used when the target labels are integers instead of one-hot encoded function vectors. Efficient for large datasets

$$L(y_i) = -\sum_{i=1}^{n} \log(y\hat{i}_{,y_i})$$

4. Hinge Loss: Used for training classifiers especially for SVMs. It is suitable for binary classification tasks.

$$\text{Hinge Loss} = \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \cdot \hat{y}_i)$$