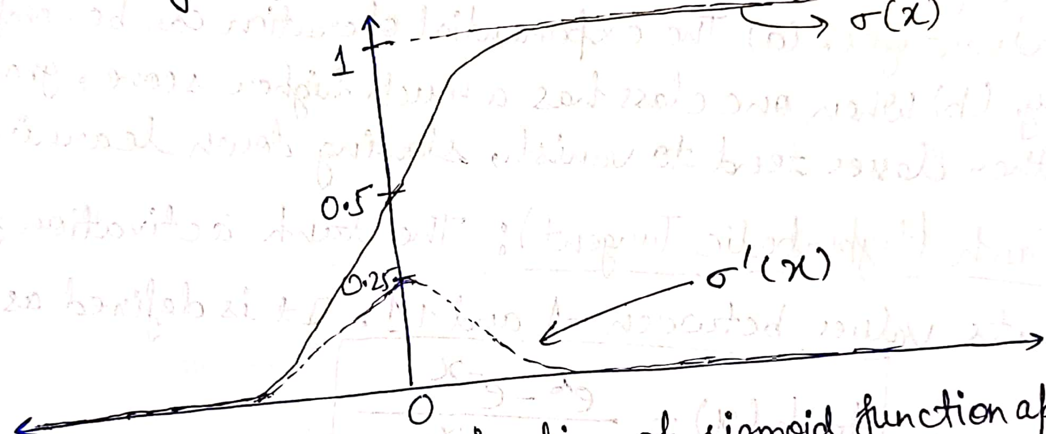
Activation Function

Activation functions in deep learning play a key role by introducing non-linearity into neural networks, which allows them to learn complex patterns and make predictions beyond linear mappings. Some commonly used activation functions in neural networks are—

(1) Sigmoid function ⇒ The sigmoid function outputs a value between 0 and 1 and is defined as:

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

The derivative of sigmoid $\boxed{\sigma'(x) = \sigma(x)\,(1-\sigma(x))}$



For larger values of x, the derivative of sigmoid function approaches towards 0.

⇒ Advantages: (a) It maps inputs to an easily interpretable range between 0 and 1, which is often used as probabilities in the output layer of a neural network for binary classification.

(b) It is a non-linear function, thus helps in capturing non-linearity in data.

(c) It is differentiable.

⇒ Disadvantages: (a) The sigmoid function is saturating in nature and saturates for large values (both positive and negative), causing gradients to diminish and thus suffers from vanishing gradient problem. (b) The outputs are non-zero centered meaning the output is between 0 & 1, which may result in slower convergance because gradients have a consistent bias direction.

(2) **Softmax Function**: Softmax is commonly used in the output layer of classification networks, especially for multi-class classification. It transforms a vector of raw scores z into probabilities, where

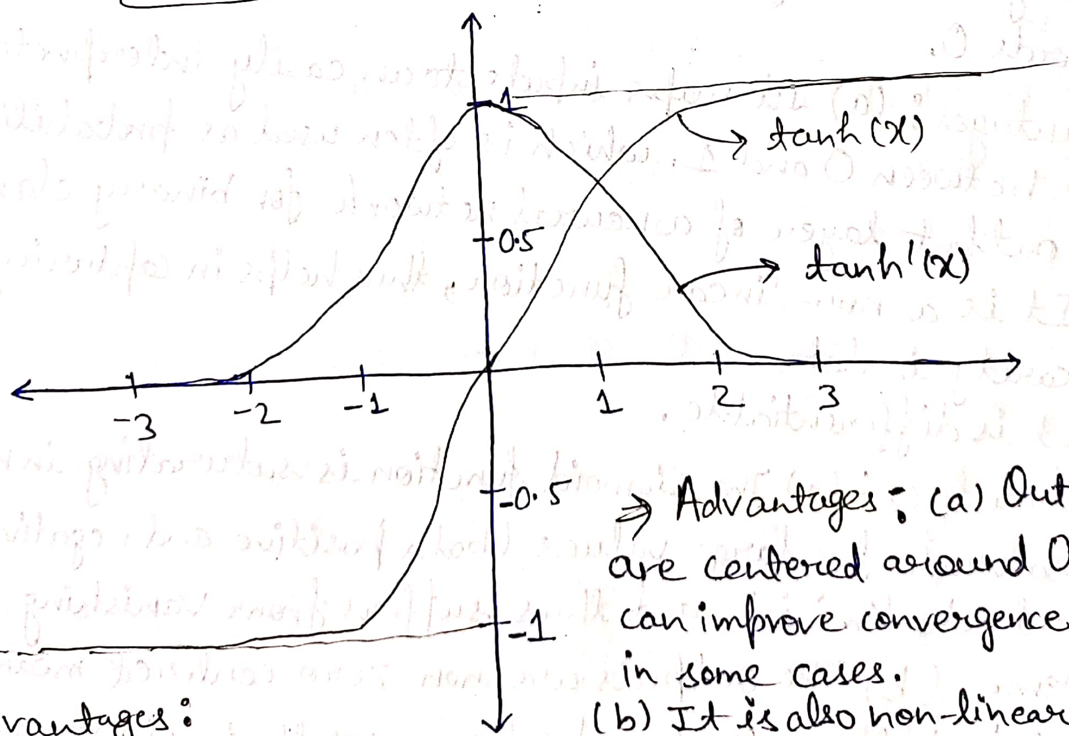
$$\boxed{\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}}$$

⇒ Advantages: (a) Produces a probability distribution over classes, making it suitable for multi-class classification problems.
(b) Interpretable output as each value represents the probability of belonging to a particular class.

⇒ Disadvantages: (a) The exponential operation can be computationally costly. (b) When one class has a much higher score, gradients for other classes tend to vanish, slowing down learning.

(3) **Tanh (Hyperbolic Tangent)**: The tanh activation function outputs values between -1 and +1. It is defined as:

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



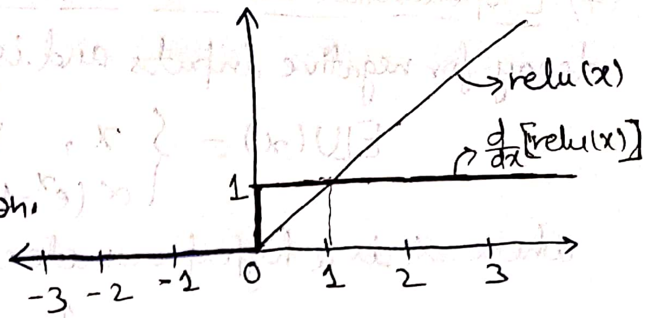
⇒ Advantages: (a) Outputs are centered around 0, which can improve convergence speed in some cases.
(b) It is also non-linear and differentiable in nature.

⇒ Disadvantages:
(a) It also squishes the large positive and negative inputs towards -1 and +1, thus it can also suffer from vanishing gradient problem.
(b) It is also computationally expensive.

(4) ReLU (Rectified Linear Unit) ReLU is defined as:

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

$$ReLU'(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



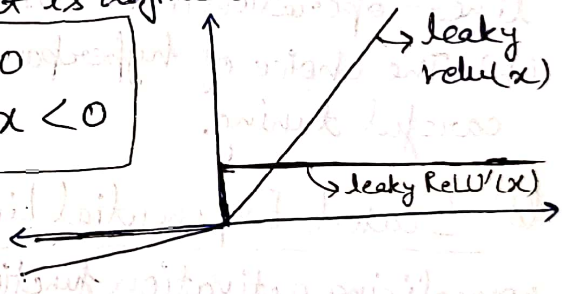
⇒ Advantages: (a) Simple to compute, requiring only a thresholding operation.
(b) It solves vanishing gradient problem and helps with gradient flow for large positive values, enabling faster learning.
(c) Drives some neurons output to zero, creating sparsity in the network, which can improve generalization.

⇒ Disadvantages: (a) It suffers from **dying ReLU** problem. If inputs are negative, gradients become zero, which may "**kill**" some neurons permanently (outputs remain zero for all future inputs).
(b) The ReLU function is not differentiable at 0.
(c) It is also non-zero centered just like sigmoid function.

(5) Leaky ReLU Leaky ReLU modifies ReLU to allow a small, non-zero slope for negative inputs. It is defined as –

$$\text{Leaky ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0.01x, & x < 0 \end{cases}$$

$$\text{Leaky ReLU}'(x) = \begin{cases} 0.01, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



⇒ Advantages: (a) The small slope for negative value allows some gradient to pass through even for negative inputs, thus helps in fixing the **dying ReLU** problem.
(b) It is simple and easy to compute like ReLU.

⇒ Disadvantages: (a) There is no real justification for choosing the slope value as 0.01 for negative values.
(b) Similar to ReLU, it is also not zero-centered, which can slow convergance in some cases.

(6) Parametric ReLU:

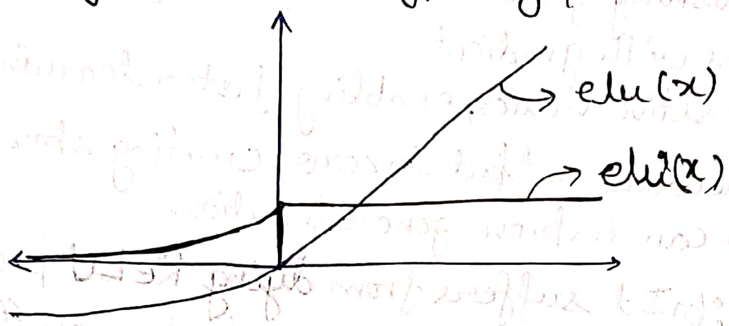
$$\boxed{\text{Parametric ReLU}(x) = \begin{cases} x, & x \geq 0 \\ \underbrace{\alpha}_{\text{hyperparameter}} x, & x < 0 \end{cases}}$$

Scanned with OKEN Scanner

(7) Exponential Linear Unit (ELU): ELU introduces exponential decay for negative inputs and is defined as:

$$ELU(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$$

where α is a hyperparameter (typically set to 1).



⇒ Advantages: (a) It promotes a smooth transition and eleminates sharp transitions at $x=0$, unlike ReLU.

(b) It is zero-centered and helps reduce bias, improving convergence.

(c) Helps in avoiding the vanishing gradient problem.

⇒ Disadvantages: (a) Exponential function is more costly than the linear operation in ReLU.

(b) The choice of hyperparameter α affects performance and requires careful tuning.

(8) Scaled Exponential Linear Unit (SELU): SELU is a self-normalizing activation function that scales its outputs to maintain a mean and variance close to 0 and 1 respectively. It is defined as:

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

where λ and α are constants chosen to ensure self-normalizing properties.

⇒ Advantages: (a) It helps network maintain normalized activations, reducing the need for additional normalization layers.

(b) It promotes stability by improving gradient flow.

⇒ Disadvantages: (a) Computationally expensive

(b) SELU requires careful initialization and specific architectures (eg. using dropouts with SELU is discouraged).