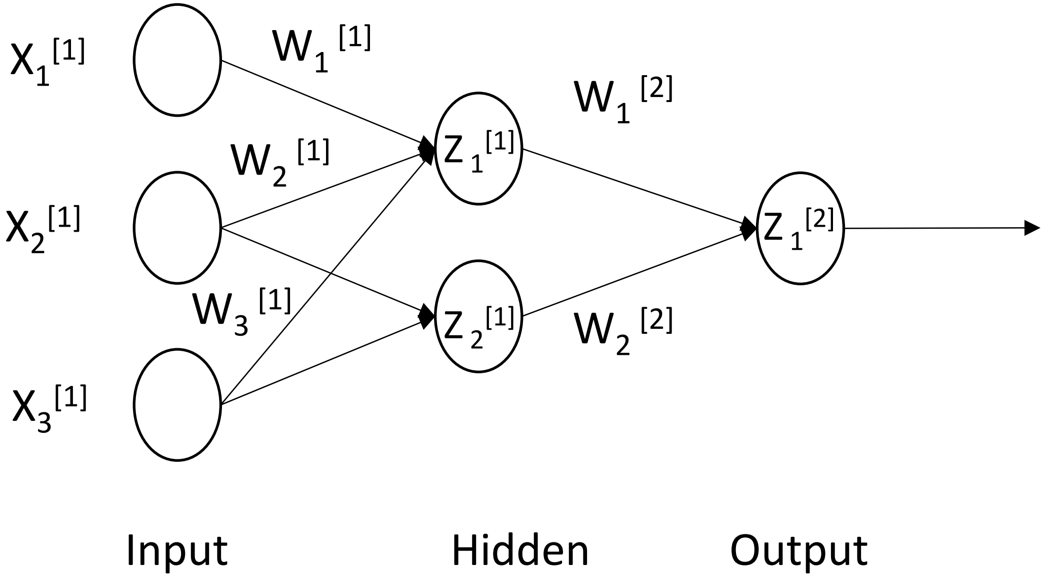
Activation Function

Nomenclature & Linear Activation Function

- **Superscipt** represents the layer
- **Subscript** the sequence in that layer



$Z^{[1]} = W^{[1]T} X^{[1]} + B^{[1]}$

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

$a^{[1]} = C^1 * Z^{[1]}$

$a^{[1]} = C^1 * (W^{[1]T} X^{[1]} + B^{[1]})$

$a^{[1]} = C^1 * W^{[1]T} X^{[1]} + C^1 * B^{[1]}$

$Z^{[2]} = W^{[2]T} a^{[1]} + B^{[2]}$

$Z^{[2]} = W^{[2]T} (C^1 * W^{[1]T} X^{[1]} + C^1 * B^{[1]}) + B^{[2]}$

$a^{[2]} = g(Z^{[2]})$

$a^{[2]} = C^2(Z^{[2]})$

$a^{[2]} = C^2(W^{[2]T} (C^1 * W^{[1]T} X^{[1]} + C^1 * B^{[1]}) + B^{[2]})$

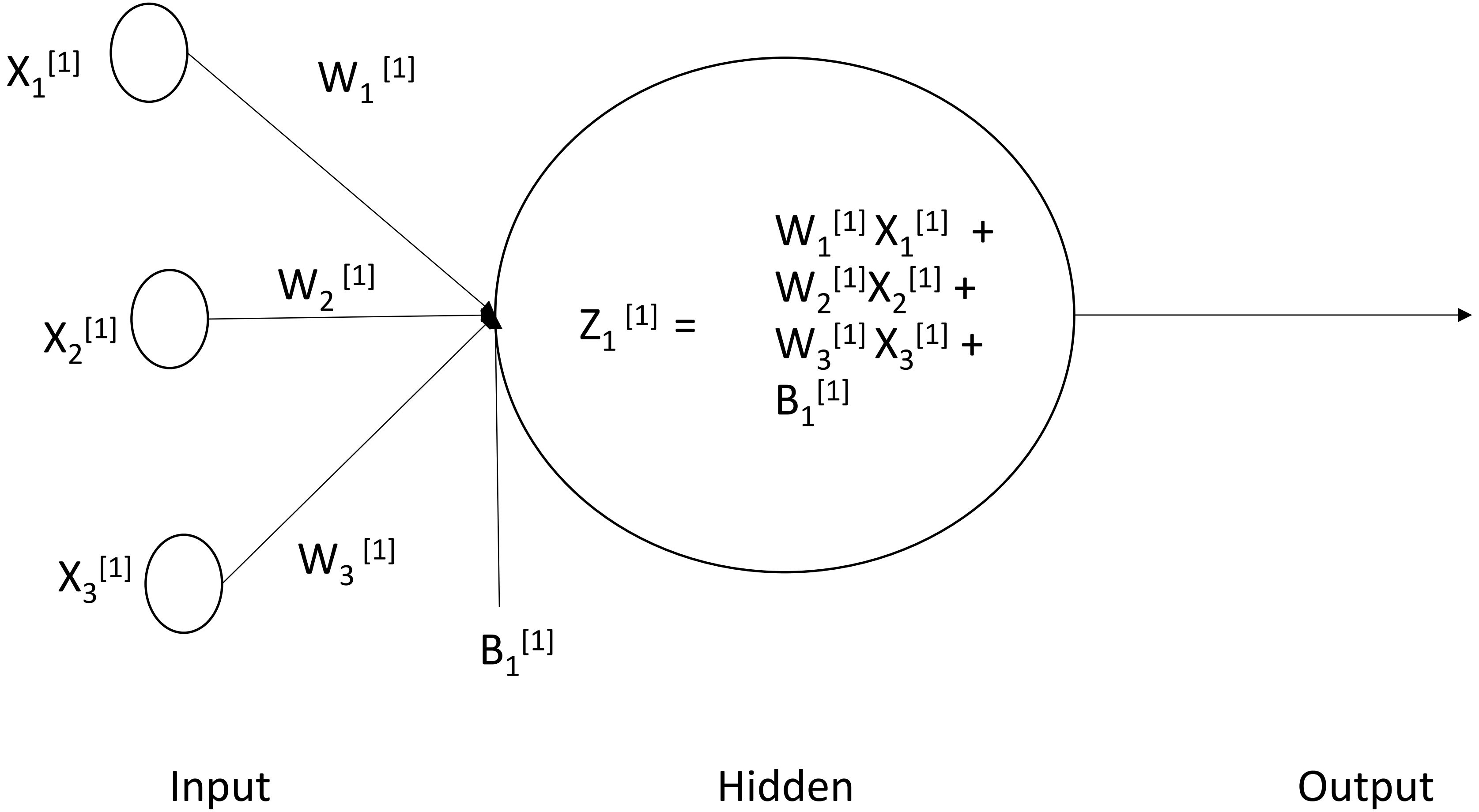
$a^{[2]} = C^2 * W^{[2]T} * C^1 * W^{[1]T} X^{[1]} + C^2 * W^{[2]T} * C^1 * B^{[1]} + C^2 B^{[2]})$

$a^{[2]} = C^2 * W^{[2]T} * C^1 * B^{[1]} + C^2 B^{[2]}) + C^2 * W^{[2]T} * C^1 * W^{[1]T} X^{[1]}$

$a^{[2]} = B^* + W^* X^{[1]}$

Linear Regression

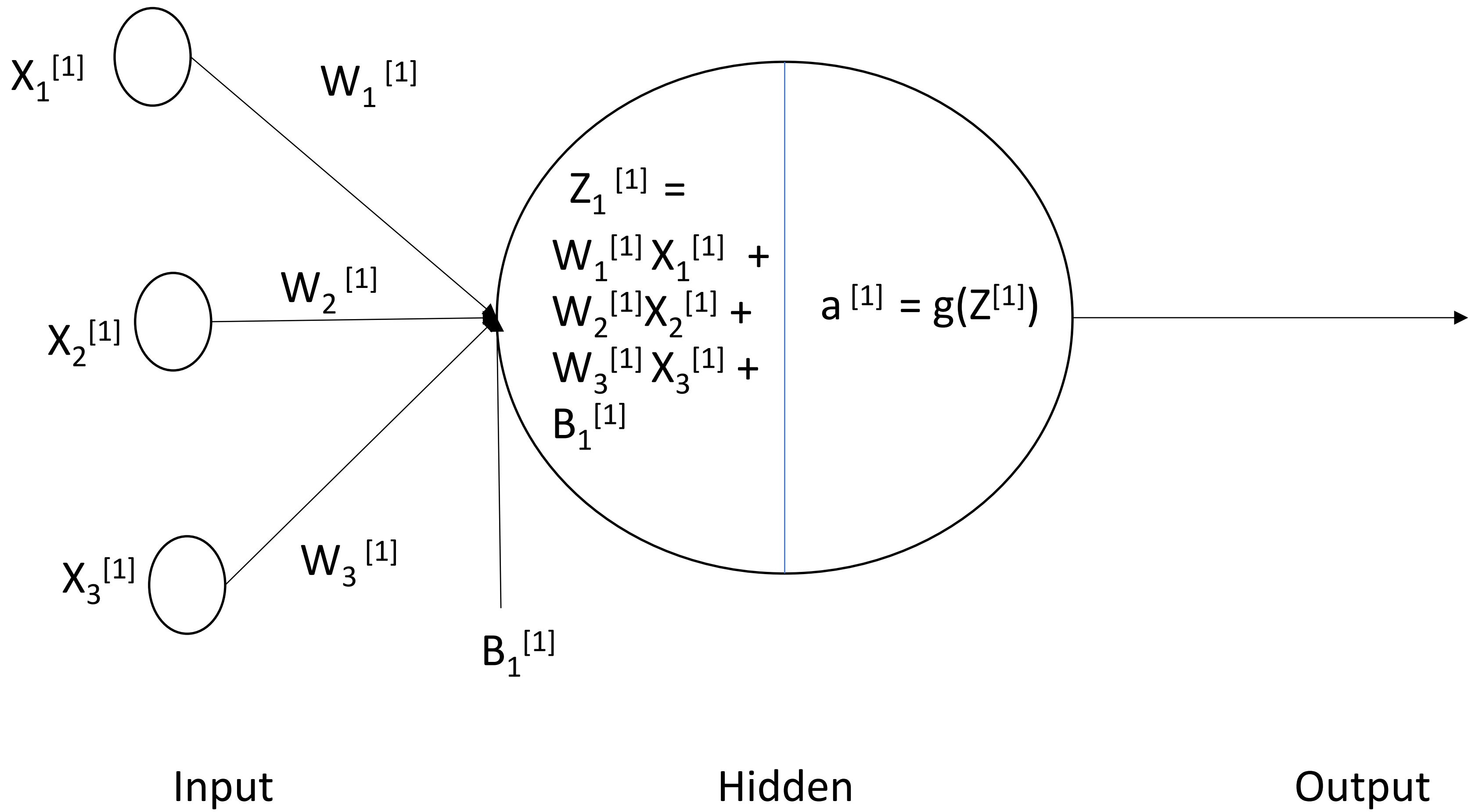
$Z^{[1]} = W^{[1]T} X^{[1]} + B^{[1]}$



Activation Function

$Z^{[1]} = W^{[1]T} X^{[1]} + B^{[1]}$

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



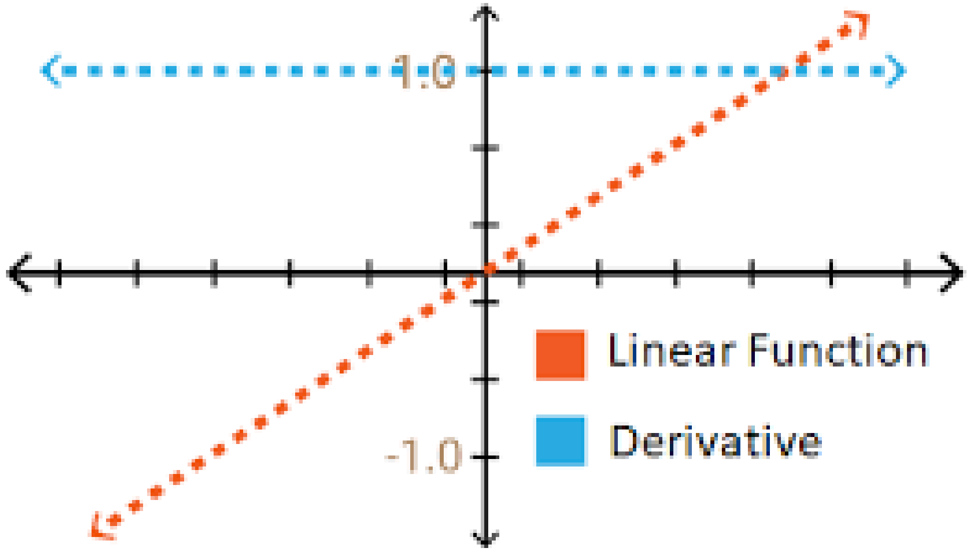
Note:

- **Superscript** Represents the Layer
- **Subscript** Represents the elements(Weight, activation fn, etc) of each layer

Linear

$$f(x) = ax + b$$

$$\frac{df(x)}{dx} = a$$



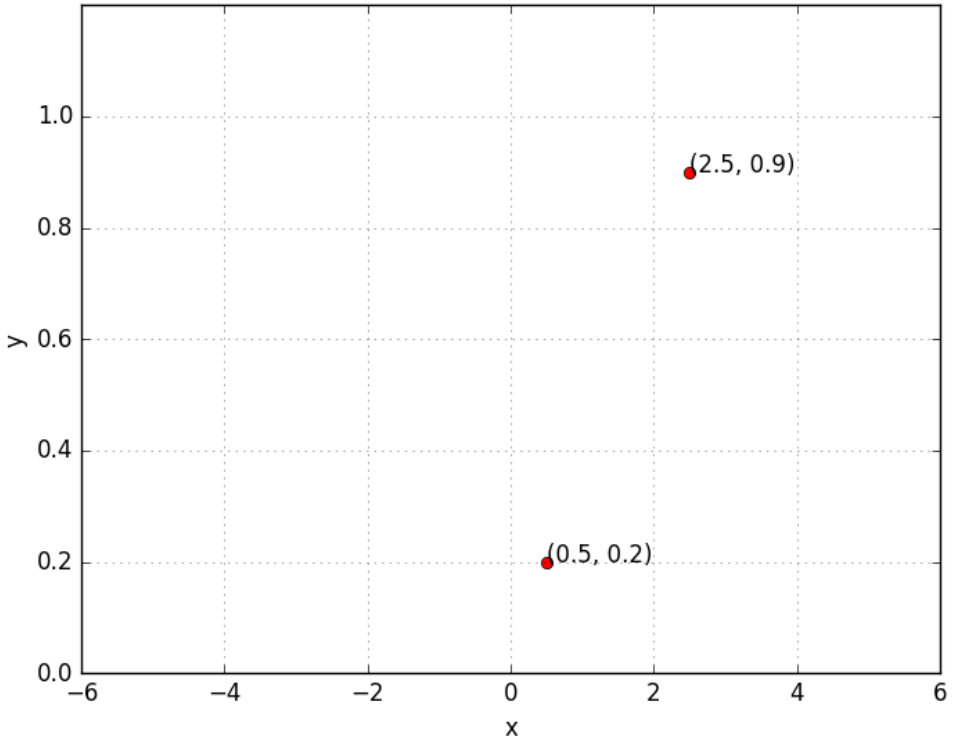
Observations:

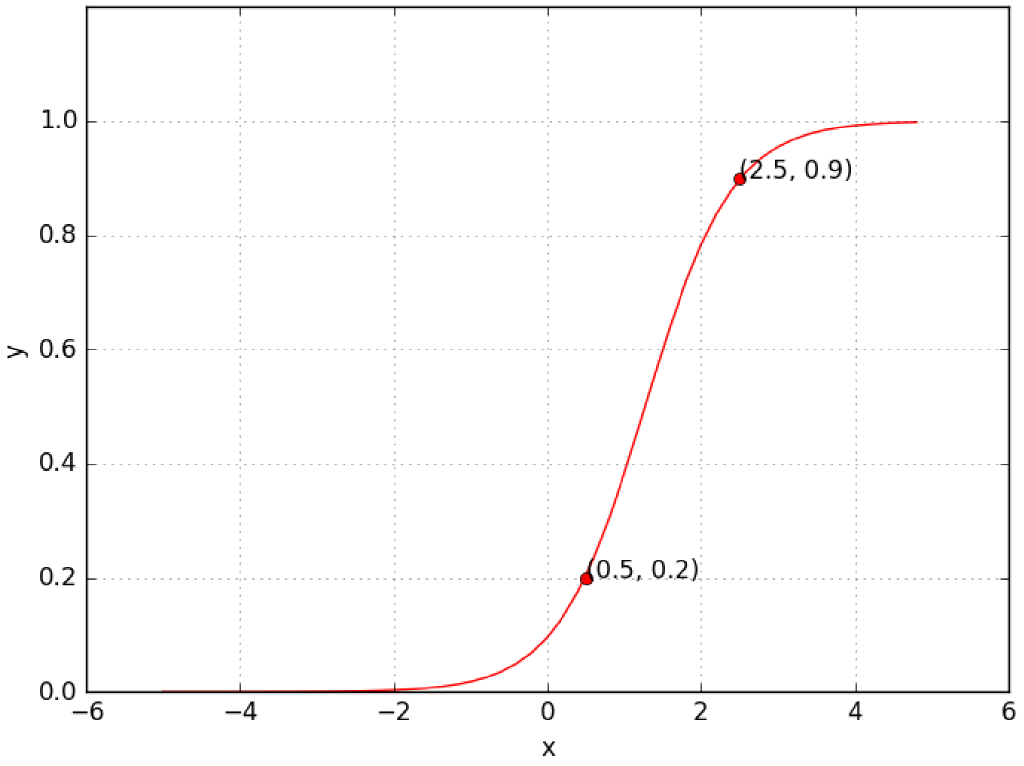
- Constant gradient
- Gradient does not depend on the change in the input

Non-Linear

- Sigmoid (Logistic)
- Hyperbolic Tangent (Tanh)
- Rectified Linear Unit (ReLU)
  - *Leaky Relu*
  - *Parametric Relu*
- Exponential Linear Unit (ELU)

Sigmoid Activation Functions (Logistics)





Suppose we train the network with $(x, y) = (0.5, 0.2)$ and $(2.5, 0.9)$

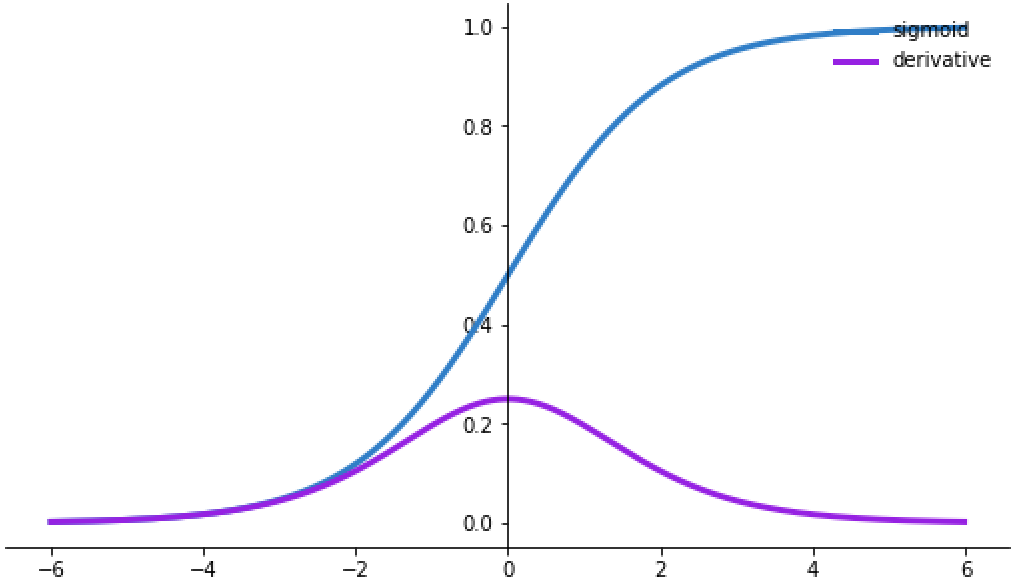
At the end of training we expect to find w^*, b^* such that:

$f(0.5) \rightarrow 0.2$ and $f(2.5) \rightarrow 0.9$

Sigmoid Activation Functions (Logistics)

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

$$\frac{df(x)}{dx} = f(x)(1 - f(x))$$



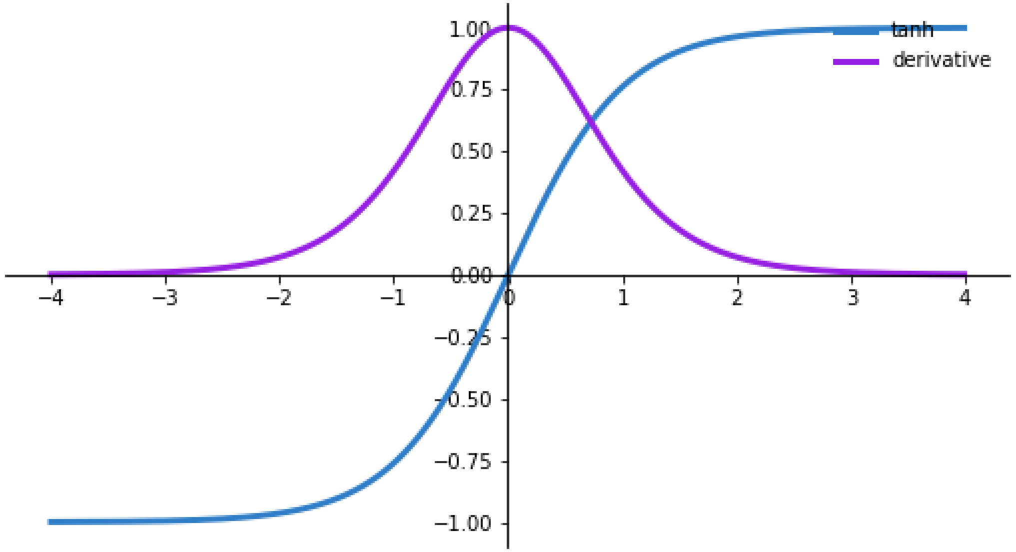
Observations:

- Output: 0 to 1
- Outputs are not zero-centered
- Can saturate and kill (vanish) gradients
- Derivative ranges from 0 to 0.25
- Can saturate and kill (vanish) gradients
- Large or very small inputs, the sigmoid function approaches either 0 or 1. The derivative of the function might become 0.

Tanh Activation Function

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

$$\frac{df(x)}{dx} = 1 - f(x)^2$$



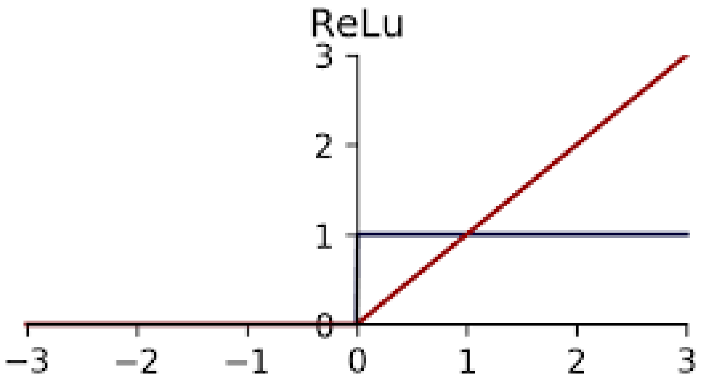
Observations:

- Output: -1 to +1
- Outputs are zero-centered
- Derivative ranges from 0 to 1
- Can Saturate and kill (vanish) gradients
- Gradient is more steeped than Sigmoid, resulting in faster convergence

Rectified Linear Unit(ReLU)

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

$$\frac{df(x)}{dx} = \begin{cases} 0 & x < 0 \\ 1, & x \geq 0 \end{cases}$$



Observations:

- Its of linear nature
- Greatly increase training speed compared to tanh and sigmoid
- Reduces likelihood of killing(vanishing) gradient
- Might Lead to Exploding Gradient
- Dead nodes (Does not update their weights during training)
- Ranges from 0 to infinity

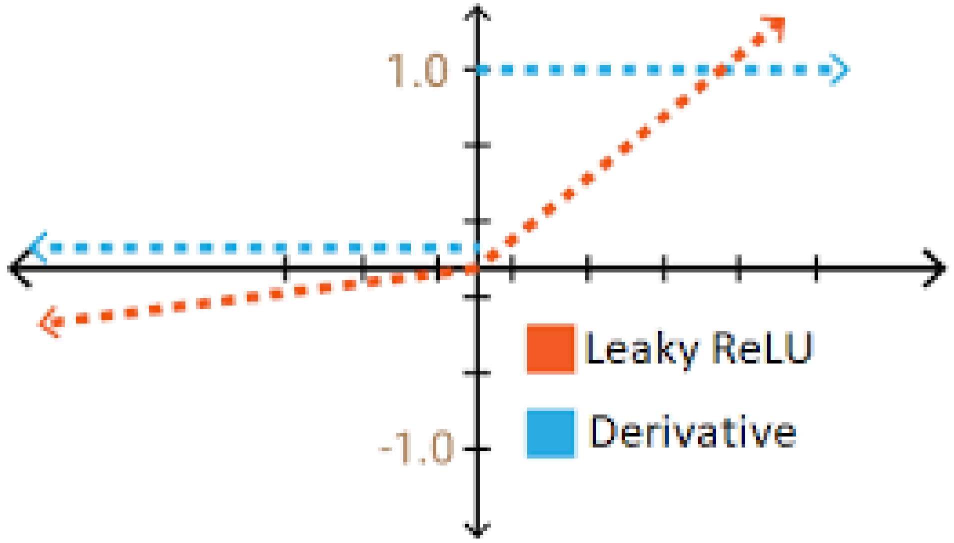
Some Variants of ReLU:

- Leaky ReLU-
- Parameterized ReLU (PReLU)-
- Exponential Linear Unit (ELU)-

Leaky-ReLU

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

$$\frac{df(x)}{dx} = \begin{cases} 0.01, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



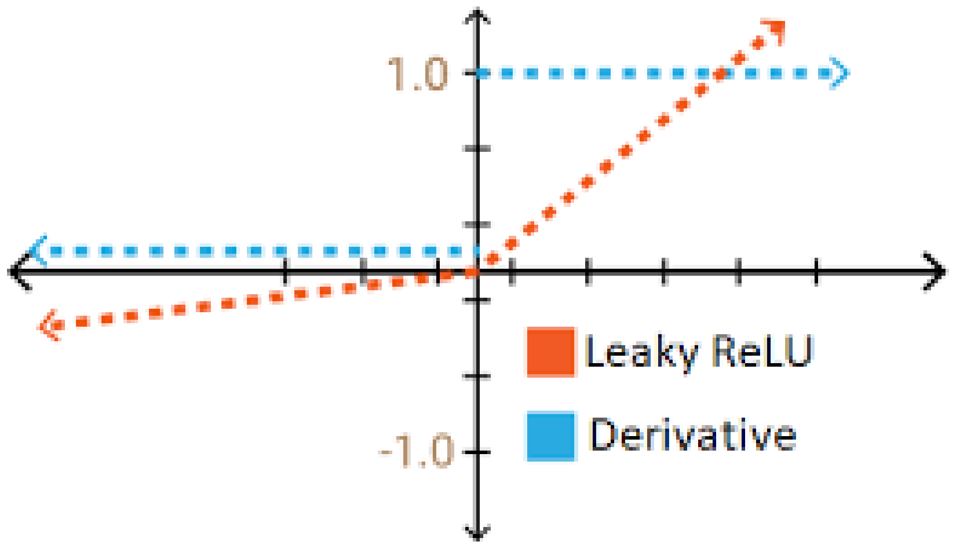
Observations:

- It prevents the neuron from dying
- More useful for Deep Networks
- Coefficient is fixed 0.01
- Slope for the negative values are fixed

Parameterized-ReLU

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

$$\frac{df(x)}{dx} = \begin{cases} \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



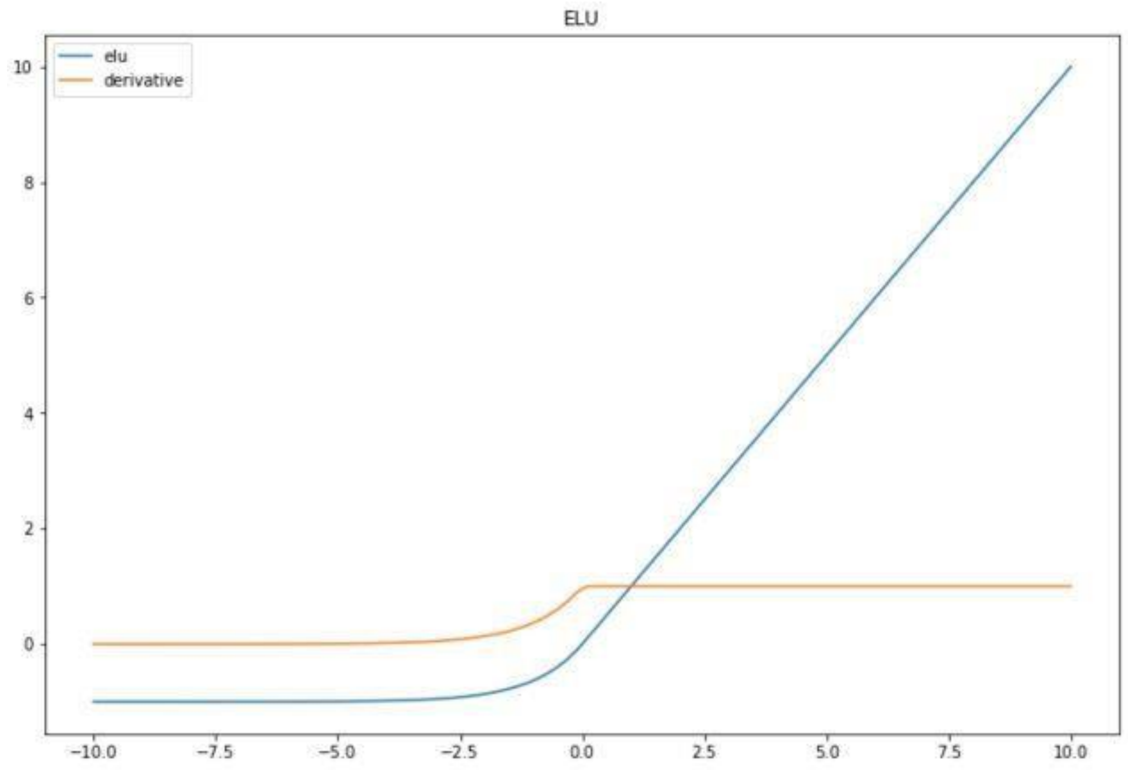
Observations:

- Another Hyperparameter, Learns the slope for the negative values
- Convergence speed is better as compared to Leaky ReLU

Exponential Linear Unit (ELU)

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

$$\frac{df(x)}{dx} = \begin{cases} f(x) + \alpha, & x < 0 \\ 1, & x \geq 0 \end{cases}$$



Observations:

- It can produce –ve output
- It can blow up activation function
- α should be positive
- Derivative ranges from 0 to infinity

Summary

- We learn characteristics of different Activation Functions and their gradient
- The choice of activation function depend on the nature of the problem, nature of the target output and the deepness of the network.