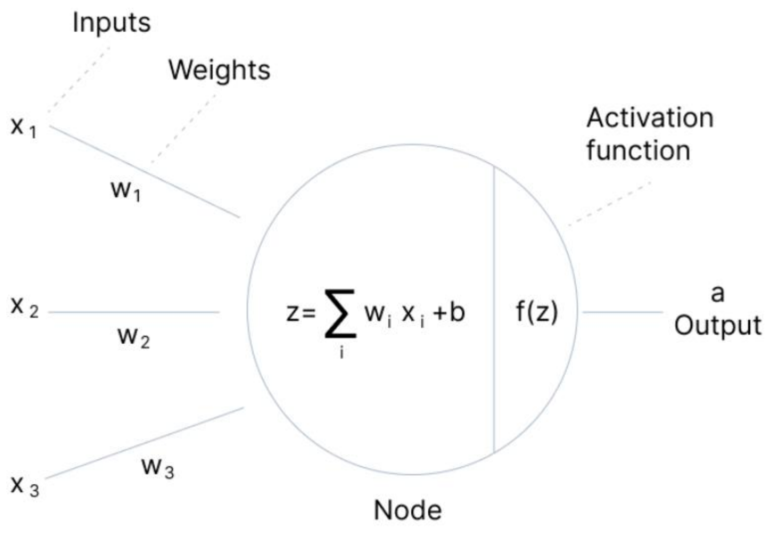
Activation Functions

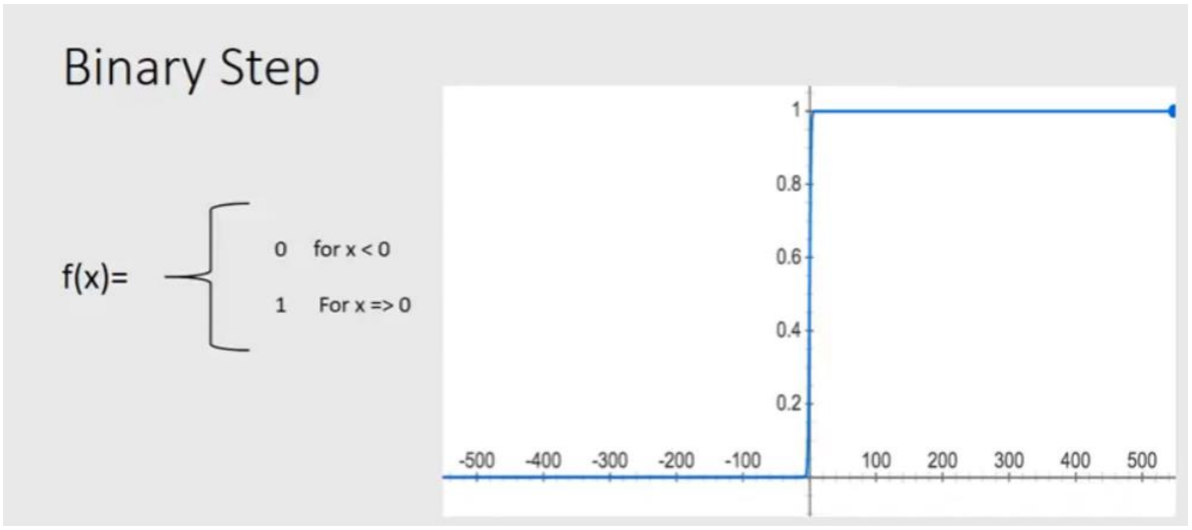
In artificial neural networks, the activation function of a node defines the output of that node given an input or set of inputs. It is also known as **transfer function.**

The primary role of the Activation Function is to transform the summed weighted input from the node into an output value to be fed to the next hidden layer or as output.



There are three types of activation function:
a) Binary Activation function
b) Linear Activation Functions
c) Non-Linear Activation Functions

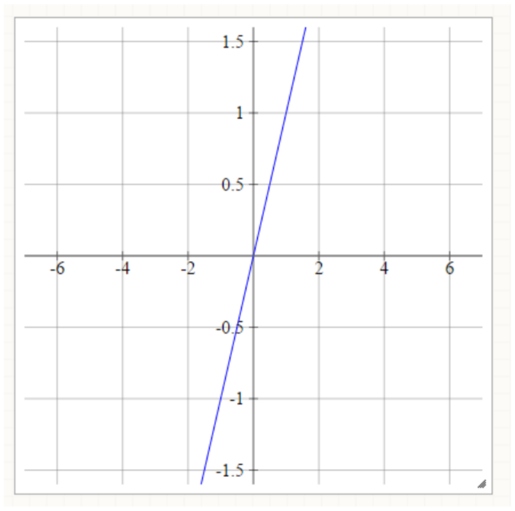
a) Binary Activation Function:



b) Linear Activation Function:

The linear activation function is also known as *Identity Function* where the activation is proportional to the input.

f(x) = x



Linear function

c) Non-Linear Activation Functions

The Nonlinear Activation Functions are the most used activation functions. Non-linear activation functions solve the following limitations of linear activation functions:

- They allow backpropagation because now the derivative function would be related to the input, and it’s possible to go back and understand which weights in the input neurons can provide a better prediction.
- They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers. Any output can be represented as a functional computation in a neural network.

Some mostly used non-linear activation functions are:

1) Sigmoid Function(Logistic Activation Function):

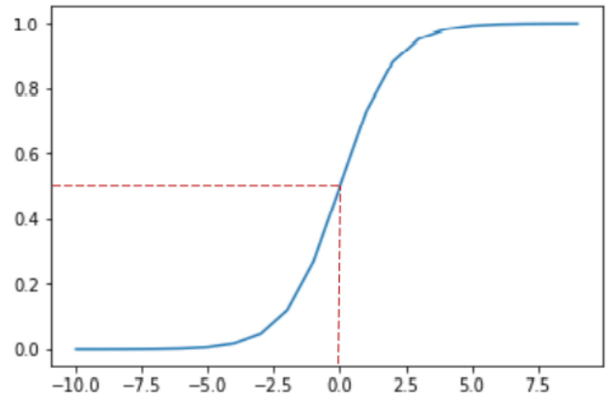
Usually used in output layer.

This function takes any real value as input and outputs values in range of 0 to 1.

Sigmoid/logistic activation function is one of the most widely used functions:

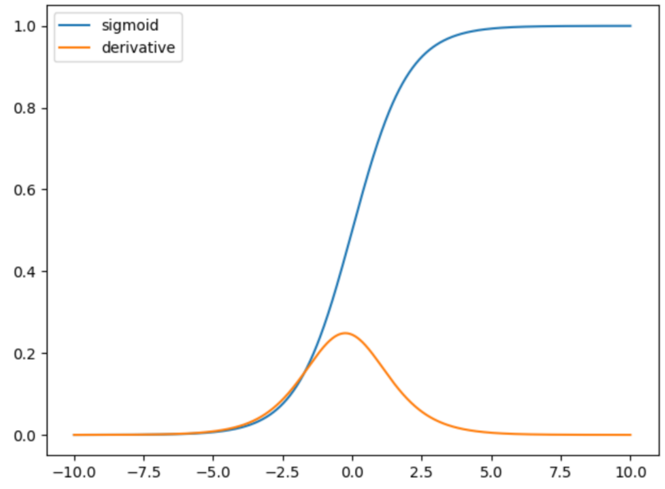
- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range.
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function.

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



Limitations of Sigmoid function:

- The function is **monotonic** but function’s derivative is not.
- The derivative of the function is f'(x) = sigmoid(x)*(1-sigmoid(x)). Gradient values will be significant only for small range (around -3 to 3). After that, graph is getting saturated to a constant value. Therefore, its gradient will approach towards 0 which will lead to vanishing gradient problem. Below graph shows the gradient variation of sigmoid function.



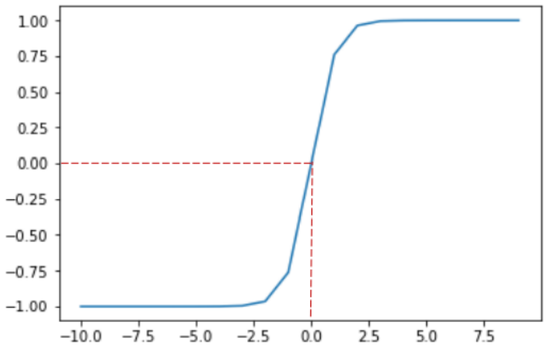
- The output of the logistic function is not symmetric around zero. So, the output of all the neurons will be of the same sign. This makes the training of the neural network more difficult and unstable.

2) Hyperbolic Tan (tanh)

Usually used in hidden layer.

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

$$\tanh(z) = 2\sigma(2z) - 1$$



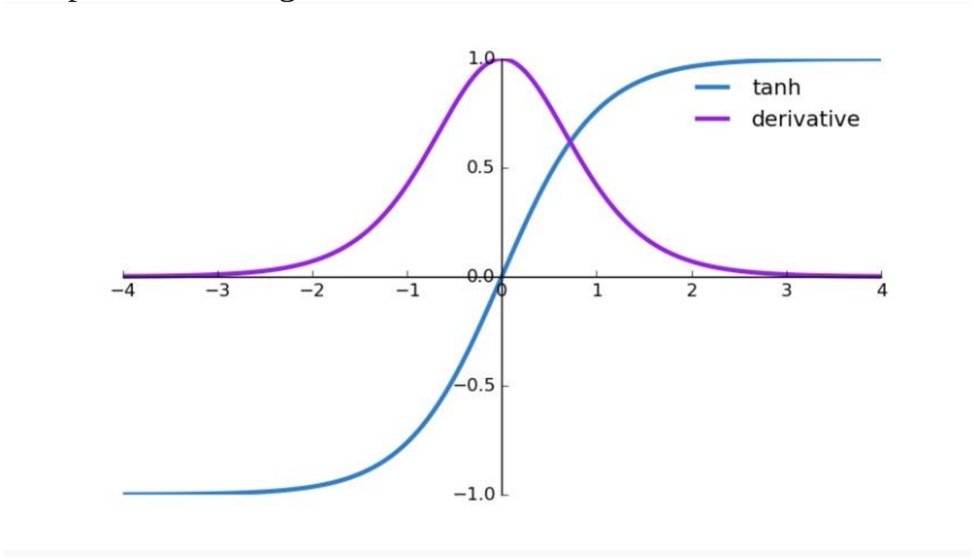
Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

Advantages of Tanh activation function:

- The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.
- Usually used in hidden layers of a neural network as its values lie between -1 to 1; therefore, the mean for the hidden layer comes out to be 0 or very close to it. It helps in centering the data and makes learning for the next layer much easier.

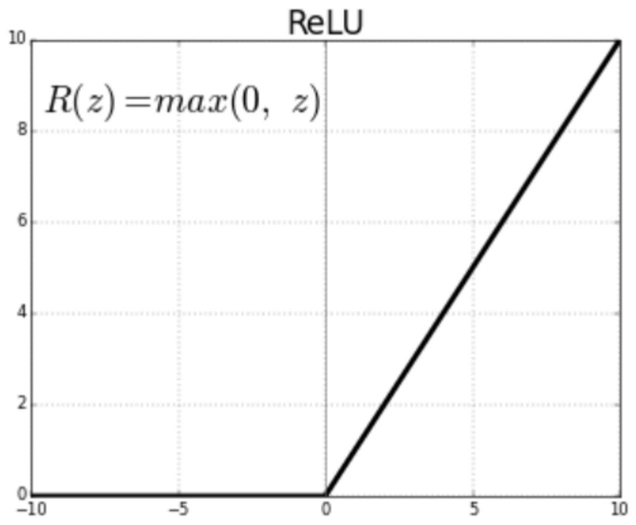
Limitations of Tanh:

- It also faces the problem of vanishing gradients similar to the sigmoid activation function. Even the gradient of the tanh function is much steeper as compared to the sigmoid function.



3) ReLU Activation function (Rectified Linear Unit)

ReLU gives an impression of a linear function, it has monotonic derivative function and allows for backpropagation while simultaneously making it computationally efficient.
The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.



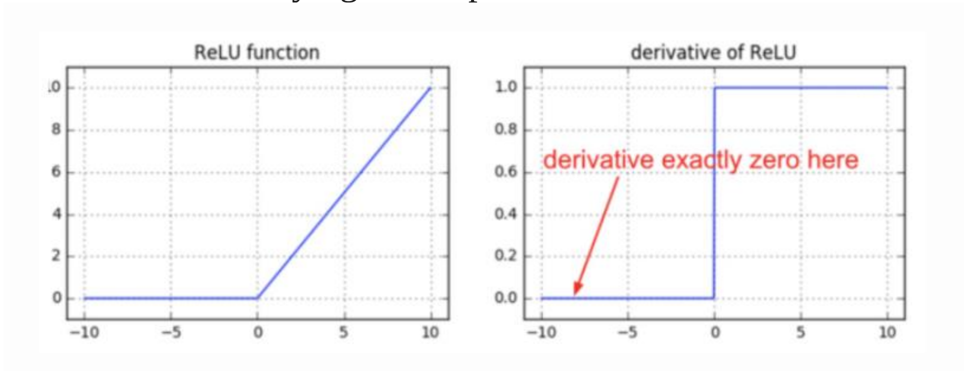
Advantages of ReLU:

- The ReLU function does not activate all the neurons at the same time. The neurons will be activated only if the output of the linear transformation is greater than 0.
- Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh functions.
- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property.

Limitations of ReLU Activation function:

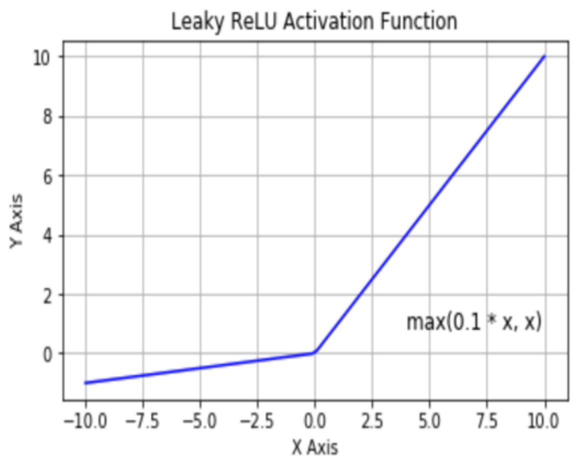
- All the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly.
- The negative side of the graph makes the gradient value zero. Due to this reason, during the backpropagation process, the weights and biases for some neurons are not updated. This can create dead neurons which never get activated.

This is known as Dying ReLU problem.



4) Leaky ReLU Activation Function

Leaky ReLU is an improved version of ReLU function to solve the Dying ReLU problem as it has a small positive slope in the negative area.

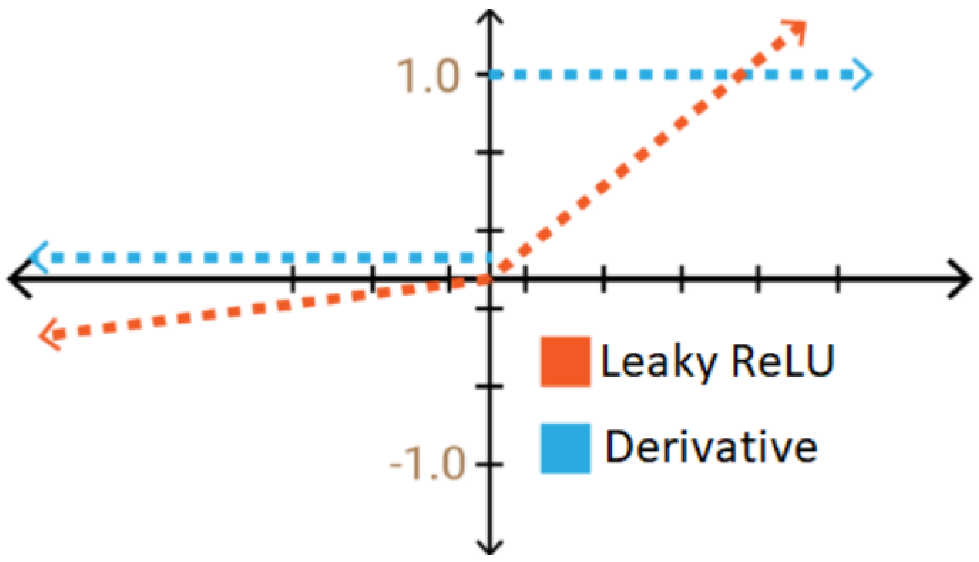


Leaky ReLU

$$f(x) = max\ (0.1x, x)$$

Advantage of Leaky ReLU Function:

- Same as that of ReLU, in addition to the fact that it does enable backpropagation, even for negative input values.
- The gradient of the left side of the graph comes out to be a non-zero value. Therefore, we would no longer encounter dead neurons in that region.



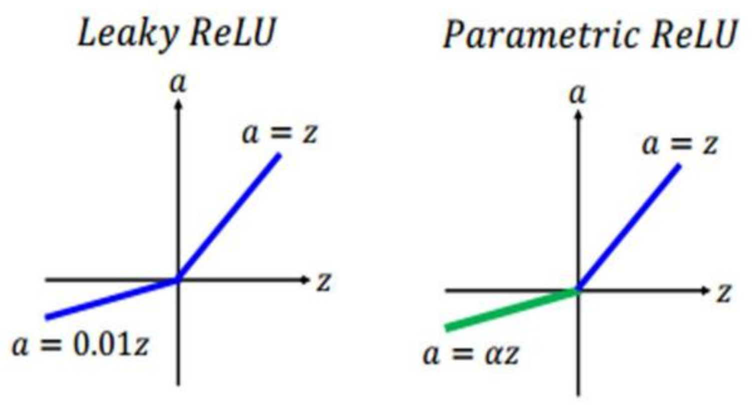
Limitations of Leaky ReLU Function:

- The predictions may not be consistent for negative input values.
- The gradient for negative values is a small value that makes the learning of model parameters time-consuming.

5) Parametric ReLU Activation Function:

Parametric ReLU is another variant of ReLU that aims to solve the problem of gradient's becoming zero for the left half of the axis.

This function provides the slope of the negative part of the function as an argument *a*. By performing backpropagation, the most appropriate value of *a* is learnt.



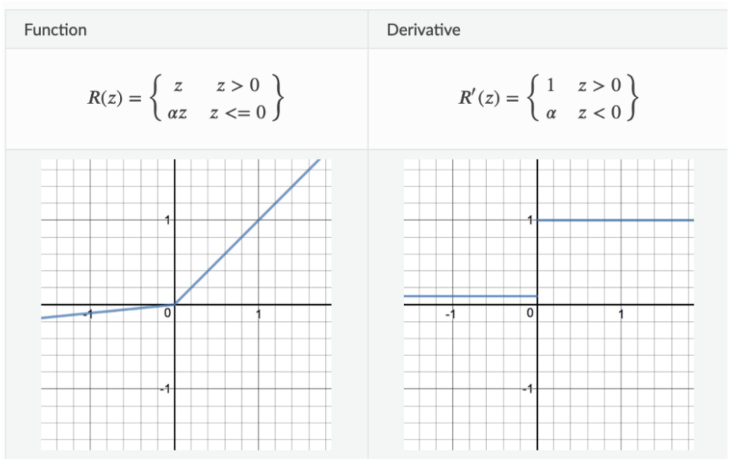
Parametric ReLU

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

where, a is the slope parameter for negative values.

Advantage of Parametric ReLU:

- Used when the leaky ReLU function still fails at solving the problem of dead neurons, and the relevant information is not successfully passed to the next layer.



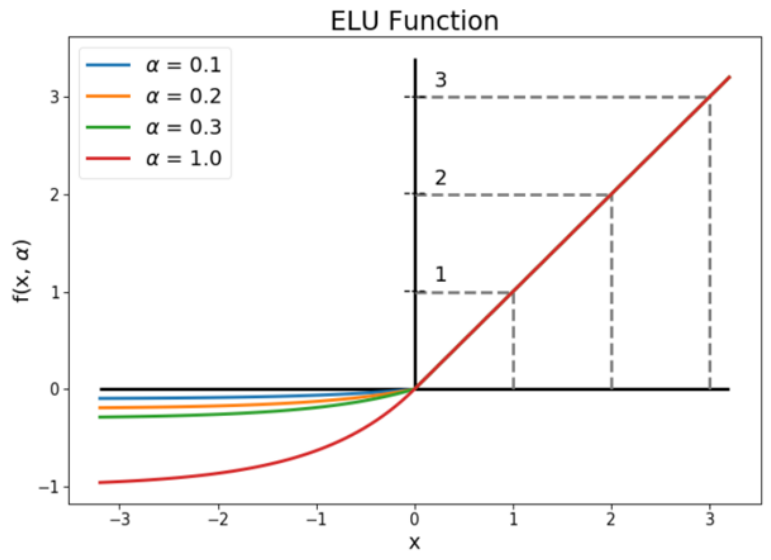
Function	Derivative
$R(z) = \begin{Bmatrix} z & z > 0 \\ \alpha z & z <= 0 \end{Bmatrix}$	$R'(z) = \begin{Bmatrix} 1 & z > 0 \\ \alpha & z < 0 \end{Bmatrix}$

Limitations of Parametric ReLU:

- It may perform differently for different problems depending upon the value of slope parameter ***a.***

6) ELU Activation Function (Exponential Linear Units)

It is also a variant of ReLU that modifies the slope of the negative part of the function. ELU uses a log curve to define the negativ values unlike the leaky ReLU and Parametric ReLU functions with a straight line.



ELU

$$\begin{cases} x & \text{for } x \geqslant 0 \\ \alpha(e^x - 1) & \text{for } x < 0 \end{cases}$$

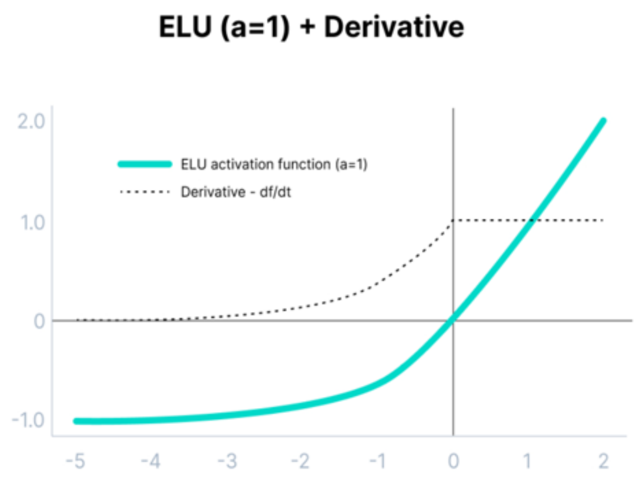
Advantages of ELU:

- ELU becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes.
- Avoids dead ReLU problem by introducing log curve for negative values of input. It helps the network nudge weights and biases in the right direction.

Limitations of ELU:

- It increases the computational time because of the exponential operation included.

- No learning of the ‘a’ value takes place.
- Exploding gradient problem.



ELU derivative

$$f'(x) = \begin{cases} 1 & for\ x \geqslant 0 \\ f(x) + \alpha & for\ x < 0 \end{cases}$$

7) GELU (Gaussian Error Linear Unit)

Activations like ReLU, ELU and PReLU have enabled faster and better convergence of Neural Networks than sigmoid. Also, Dropout regularizes the model by randomly multiplying a few activations by 0.
Both of the above methods together decide a neuron’s output. Yet, the two work independently from each other. GELU aims to combine them.

Also, a new RNN regularizer called *Zoneout* stochastically multiplies input by 1.

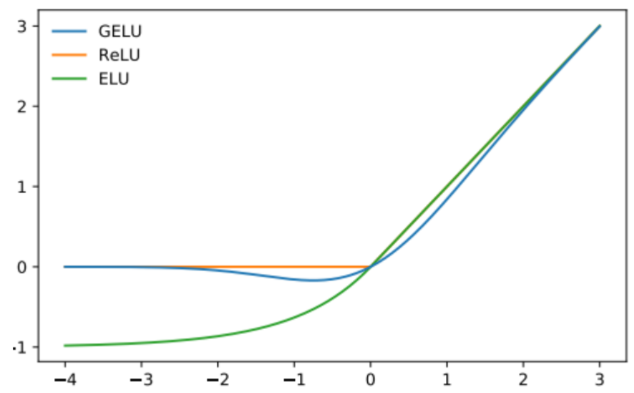
We want to merge all 3 functionalities by stochastically multiplying the input by 0 or 1 and getting the output value (of the activation function) deterministically.

We multiply the neuron input x by

m ~ Bernoulli(Φ(x)), where Φ(x) = P(X ≤x), X ~ N (0, 1) is the cumulative distribution function of the standard normal distribution.

This distribution is chosen since neuron inputs tend to follow a normal distribution, especially with Batch Normalization.

$$\begin{aligned}\mathrm{GELU}(x) &= xP(X \le x) = x\Phi(x)\\ &\approx 0.5x\left(1+\tanh\left[\sqrt{2/\pi}\left(x+0.044715x^3\right)\right]\right)\end{aligned}$$



GELU (μ=0, σ=1), ReLU and ELU (α=1)

Advantages of GELU:

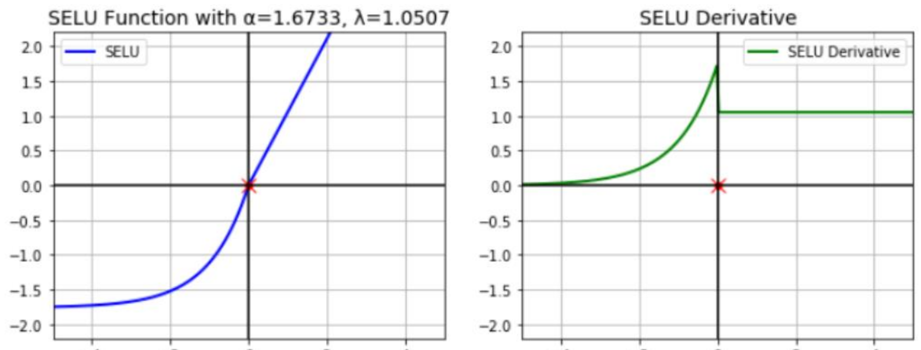
- GELU nonlinearity is better than ReLU and ELU activations and finds performance improvements across all tasks in domains of computer vision, natural language processing, and speech recognition.
- This activation function is compatible with BERT, ROBERTa, ALBERT, and other top NLP models. This activation function is motivated by combining properties from dropout, zoneout, and ReLUs.

8) SELU (Scaled Exponential Linear Unit)

SELU is scaled variant of ELU activation function. It uses two fixed parameters α and λ, and the value of these is derived from the inputs. However, for standardized inputs (mean of 0 and standard deviation of 1) the suggested values are α=1.6733, λ=1.0507.

$$\operatorname{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

Scaled Exponential Linear unit



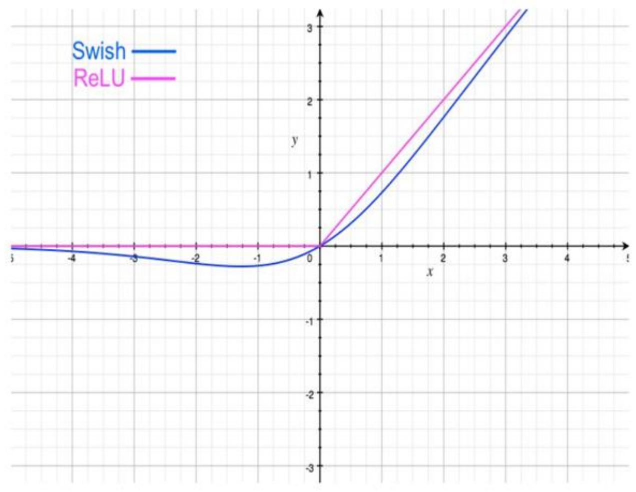
SELU activation function with α=1.6733, λ=1.0507 and its derivative

Advantages of SELU Activation Function:

- It provides self-normalization (that is output from SELU activation will preserve the mean of 0 and standard deviation of 1) and this solves the vanishing or exploding gradients problem.
  SELU will provide the self-normalization if: (a) the neural network consists only a stack of dense layers, (b) all the hidden layers use SELU activation function, (c) the input features must be standardized (having mean of 0 and standard deviation of 1), (d) the hidden layers weight must be initialized with LeCun normal initialization, and lastly, (e) the network must be sequential.
- Gradients can be used to adjust the variance. The activation function needs a region with a gradient larger than one to increase it.
- Internal normalization is faster than external normalization, which means the network converges faster. (compared to RELU)

9) Swish Activation Function

It is a self-gated activation function developed by researchers at Google. (Self gated mean it uses its own value to gate itself, or we can say the gate is actually the '*sigmoid*' of activation itself.
Swish is a smooth, non-monotonic function that consistently matches or outperforms ReLU on deep networks. It is unbounded above and bounded below & it is the non-monotonic attribute that actually creates the difference.



Swish and ReLu activations

Swish

$$f(x) = x * sigmoid(x)$$

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

Advantages of Swish Activation Function:

- Swish is a smooth function that means that it does not abruptly change direction like ReLU does near x = 0. Rather, it smoothly bends from 0 towards values < 0 and then upwards again.
- Small negative values were zeroed out in ReLU activation function. However, those negative values may still be relevant for capturing patterns underlying the data. Large negative values are zeroed out for reasons of sparsity making it a win-win situation.

- The swish function being non-monotonous enhances the expression of input data and weight to be learnt.

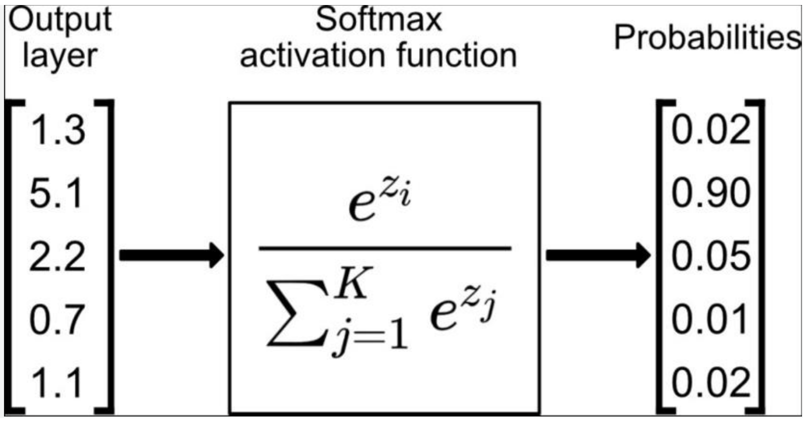
Limitations of Swish Activation function:

- Computationally expensive function (as of Sigmoid).

10) Softmax Activation Function

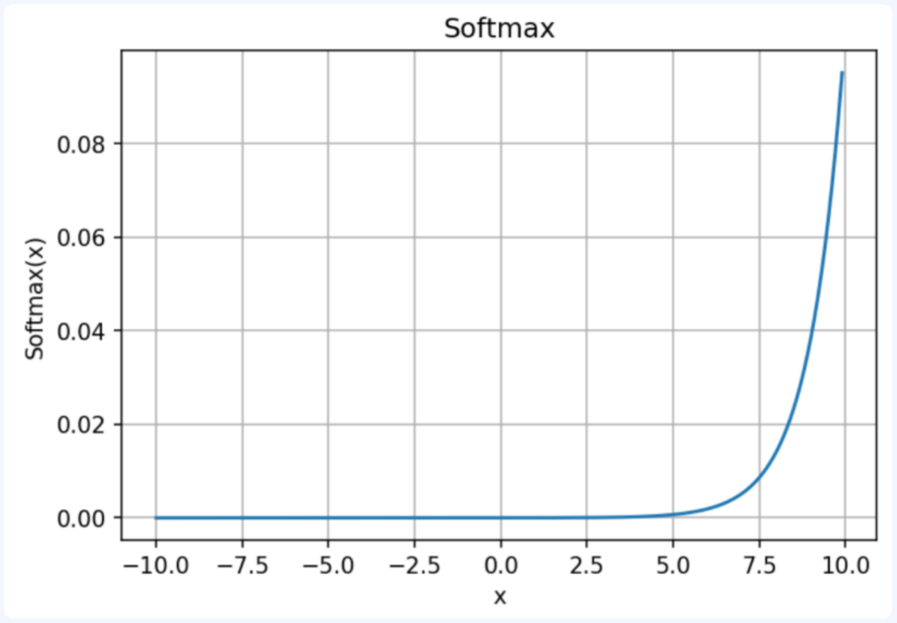
Used in Output layer

The Softmax function is described as a combination of multiple sigmoids.
It calculates the relative probabilities. Similar to the sigmoid/logistic activation function, the SoftMax function returns the probability of each class.
It is most commonly used as an activation function for the last layer of the neural network in the case of multi-class classification.
The softmax function is sometimes called the softargmax function, or multi-class logistic regression.


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

$$f(x_i) = \frac{e^{(x_i)}}{\Sigma_j e^{(x_j)}}$$

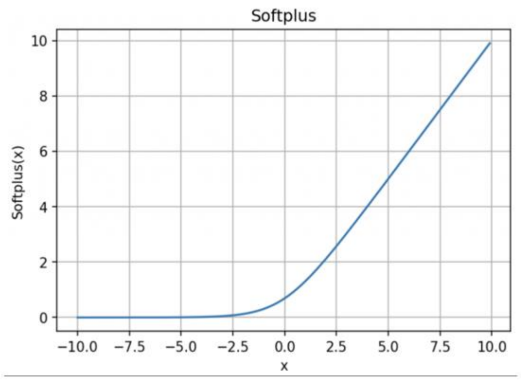
x_j:*data point belonging to* j^{th} *class*



Range: (0, 1)

11) Softplus Activation Function:

It can be viewed as a smooth version of Relu.
Relu and Softplus are largely similar, except near 0(zero) where the softplus is enticingly smooth and differentiable. It’s much easier and efficient to compute ReLU and its derivative than for the softplus function which has log(.) and exp(.) in its formulation.

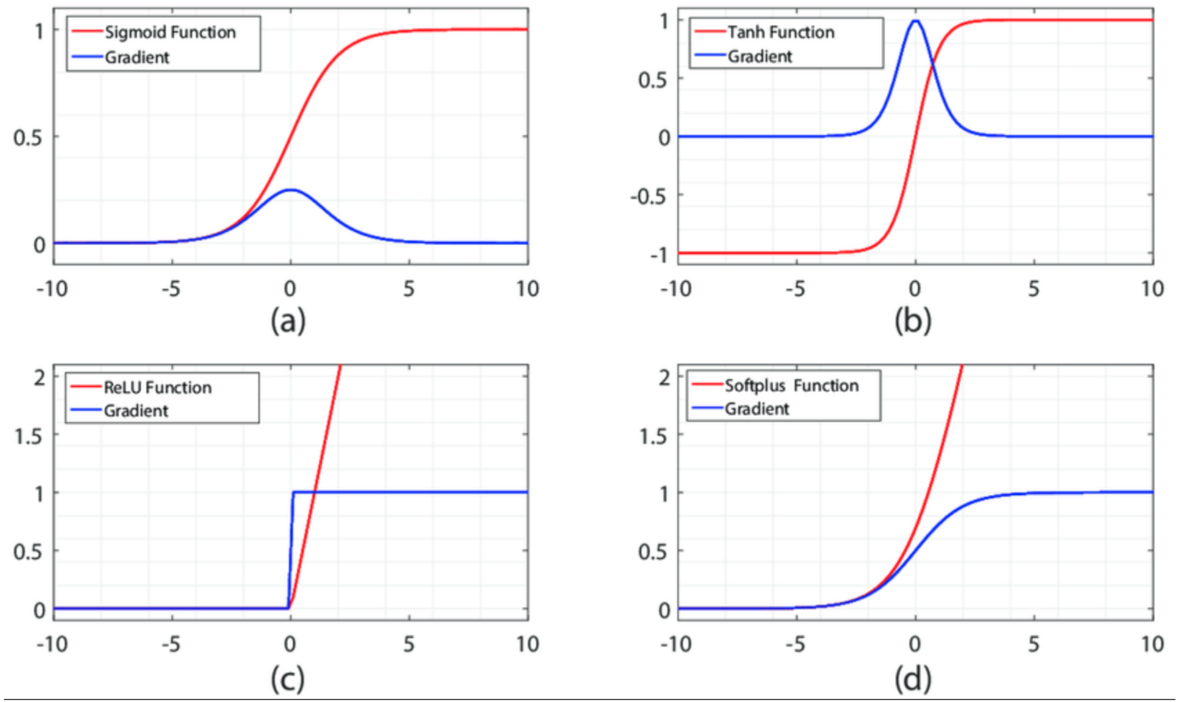


Range: $(0, \infty)$

Softplus: $f(x) = \ln(1 + \exp x)$,

Derivative of the softplus function is the logistic function.

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



Advantages of Softplus:

- It is smooth approximation of RELU function, derivative defined at 0.
- The derivative of softplus(x) is sigmoid function.

Limitations of Softplus:

- Although it reduces vanishing gradient problem but it still persist.

- softplus activation is sound theoretically but doesn’t give any practical advantage over regular ReLU.
- Computationally expensive than RELU (as of exponential in nature).

References:

https://www.v7labs.com/blog/neural-networks-activation-functions

https://medium.com/@cmukesh8688/activation-functions-sigmoid-tanh-relu-leaky-relu-softmax-50d3778dcea5

GELU - https://arxiv.org/pdf/1606.08415v3.pdf

Softmax function- https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

Palak Jain