

Get started

Open in app

**vinodhkumar baskaran**

Follow

48 Followers

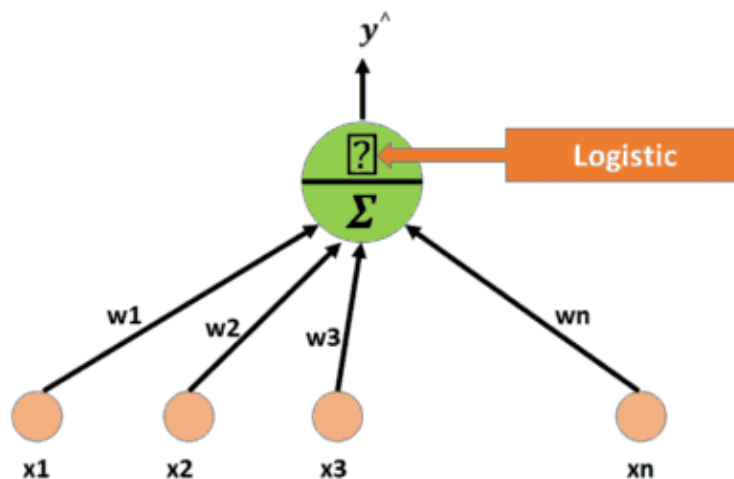
About

You have 1 free member-only story left this month. [Sign up for Medium](#) and get an extra one

Activation functions and its types

v

vinodhkumar baskaran · Apr 14, 2020 · 10 min read ★



Activation Function

What is “Activation function” ?

An activation function is a very important feature of an artificial neural network , they basically decide whether the neuron should be activated or not.

Get started

Open in app



Important use of any activation function is to introduce non-linear properties to our Network.

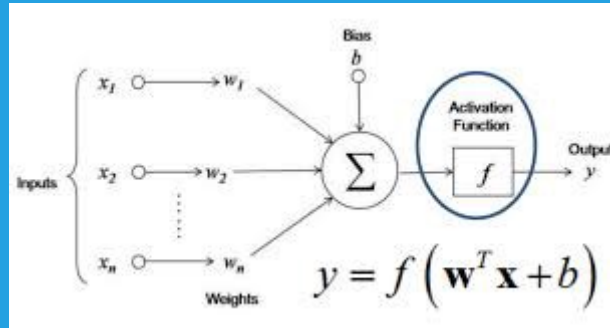


Fig 1

In simple term , it calculates a “weighted sum(W_i)” of its input(x_i), adds a bias and then decides whether it should be “fired” or not.


Explanation of above figure (*Fig 1*),

All the *input X_i 's* are multiplied with their *weight W_i 's* assigned to each link and summed together along with *Bias b* .

Note : X_i 's and W_i 's are vectors and b is scalar.

Let Y be summation of (($W_i * X_i$) + b)

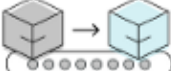
The *value of Y* can be anything ranging from $-\infty$ to $+\infty$. Meaning it has lot of information ,now neuron must know to distinguish between the “*useful*” and “*not-so-useful*” **information**. To build this sense into our network we add ‘**activation function (f)**’— Which will decide whether the information passed is useful or not based on the result it get fired.




Take input and multiply by the neuron's weight.




Add bias*

Feed the result, x , to the activation function: $f(x)$

Take the output and transmit to the next layer of neurons.

Get started

Open in app



Properties that Activation function should hold :

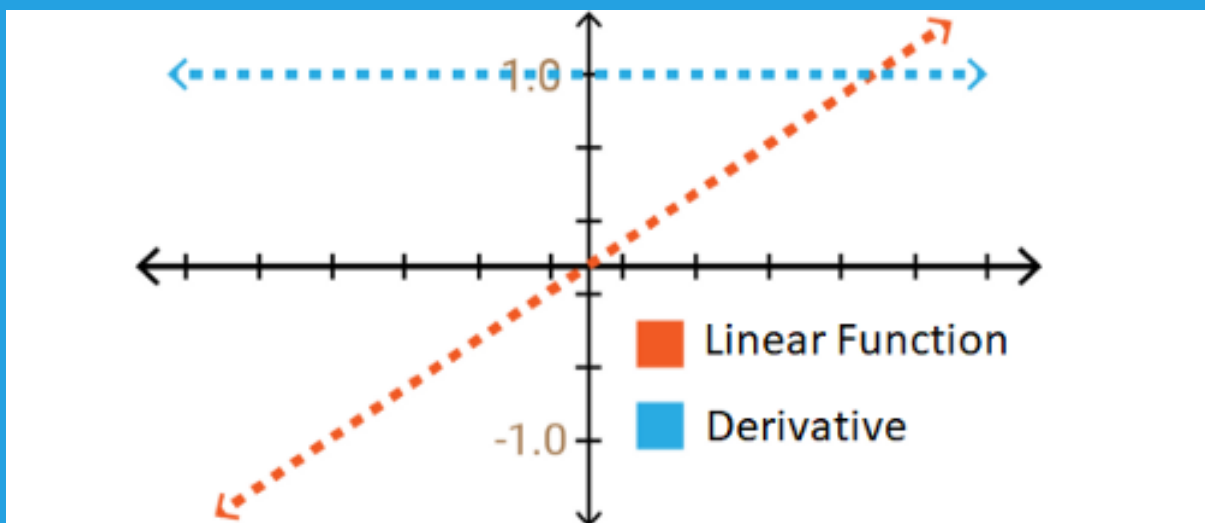
Derivative or Differential: Change in y-axis w.r.t. change in x-axis. It is also known as slope. (Back prop)

Monotonic function: A function which is either entirely non-increasing or non-decreasing.

Activation function Types :

- Linear function
- Binary Step function
- Non-Linear function

Linear Function :



Linear function

A linear activation function takes the form:

$y = mx + c$ (m is line equation represents W and c is represented as b in neural nets so equation can be modified as $y = Wx + b$)

It takes the inputs (X_i 's), multiplied by the weights (W_i 's) for each neuron, and creates an output proportional to the input. In simple term, weighted sum input is proportional to output.

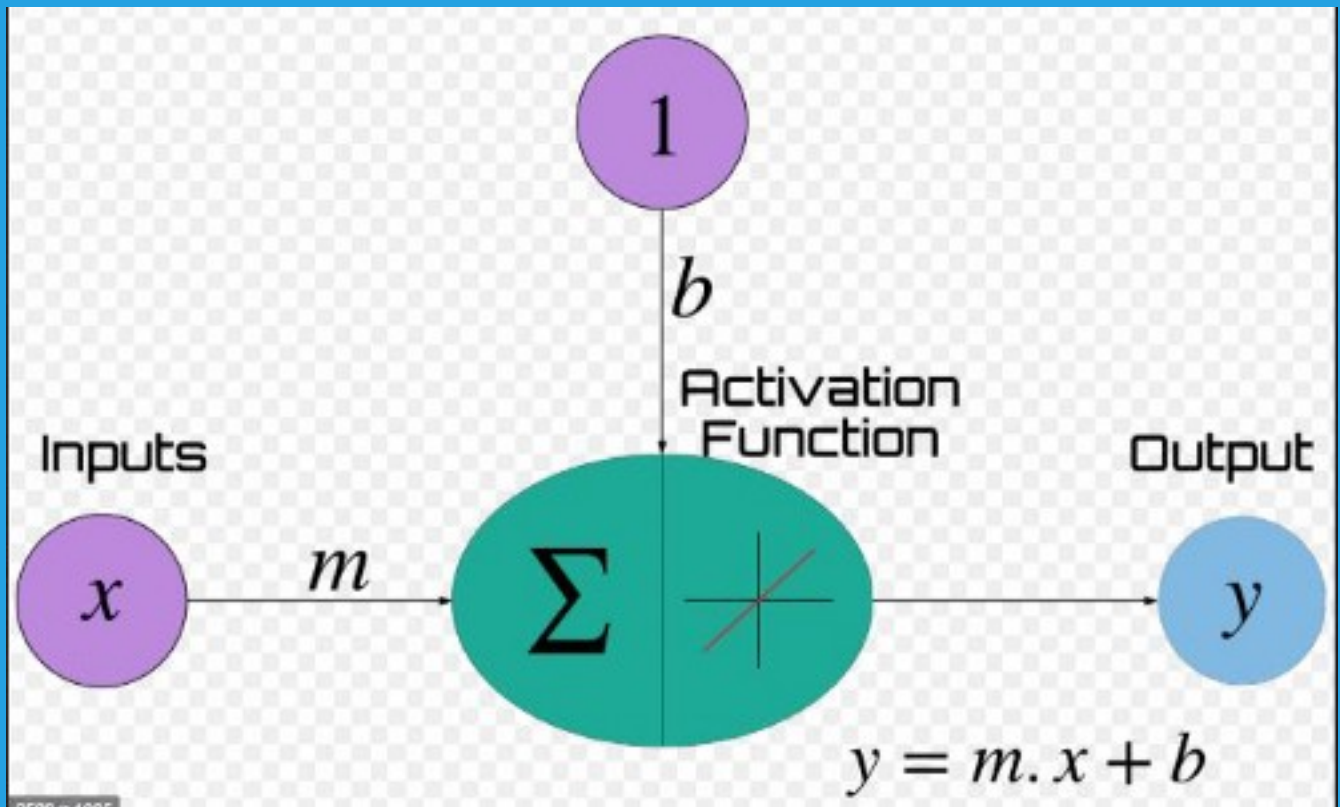
Get started

Open in app



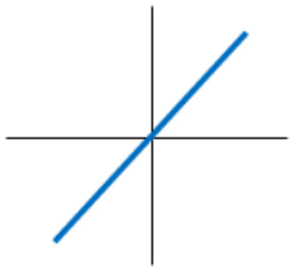
Problem with Linear function,

- 1) **Differential result is constant.**
- 2) All layers of the neural network collapse into one.

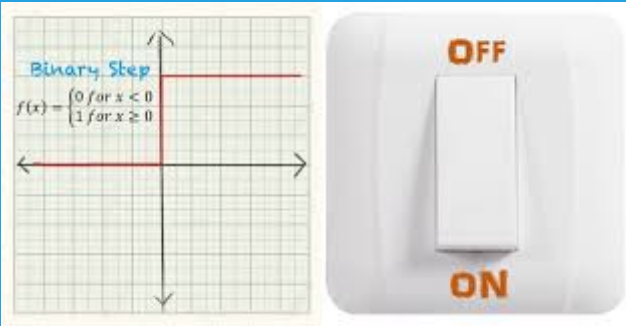


- Differential of linear function is constant and has no relation with the input. Which implies weights and bias will be updated during the backprop but the updating factor (*gradient*) would be the same.
- linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer — Meaning Output of the first layer is same as the output of the nth layer.

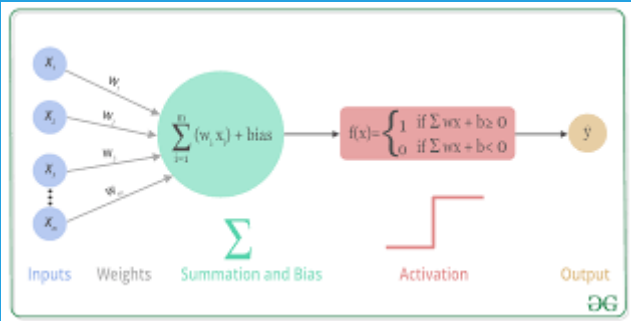
A neural networks with a linear activation function is simply a linear regression model.

Function	Equation	Range	Derivative	
Linear	$f(x) = x$	$-\infty, +\infty$	$f'(x) = 1$	

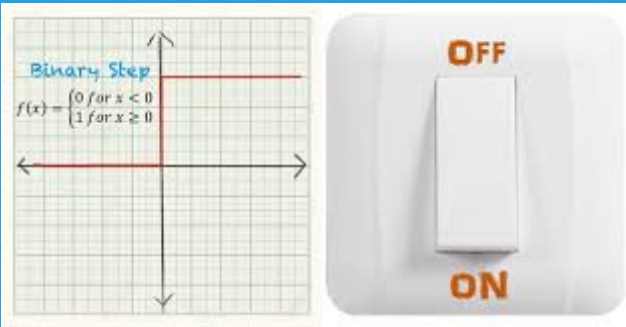
Binary Step Function:



Binary Step function



Binary step function are popular known as “Threshold function”. It is very simple function.



Function	Equation	Range	Derivative
----------	----------	-------	------------

Get started

Open in app



Binary step	$f(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$				
-------------	--	--	--	--	--

Pros and Cons :

- The gradient(differential) of the binary step function is zero,which is the very big problem in back prop for weight updation.
- Another problem with a step function is that it can handle binary class problem alone.(Though with some tweak we can use it for multi-class problem)

Non-Linear function:

The deep learning rocketing to the sky because of the non-linear functions. Most modern neural network use the non-linear function as their activation function to fire the neuron. Reason being they allow the model to create complex mappings between the network's inputs and outputs, which are essential for learning and modeling complex data, such as images, video, audio, and data sets which are non-linear or have high dimensionality.

Advantage of Non-linear function over the Linear function :

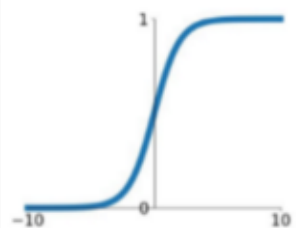
- Differential are possible in all the non -linear function.
- Stacking of network is possible , which helps us in creating the deep neural nets.

Non-linear Types:

The Nonlinear Activation Functions are mainly divided on the basis of their **range or curves**.

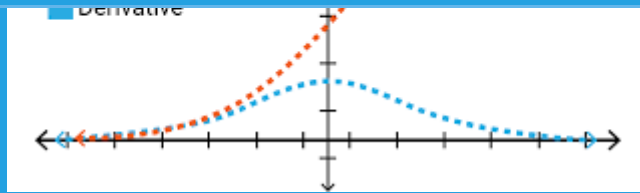
1. Sigmoid or logistic Activation Function:

Function	Equation	Range	Derivative
Sigmoid (Logistic)	$f(x) = \frac{1}{1 + e^{-x}}$	0,1	$f'(x) = f(x)(1 - f(x))$



Get started

Open in app



Sigmoid activation , Derivative

- The output of the sigmoid function always ranges between 0 and 1 .
- Sigmoid is S-shaped , '*monotonic*' & '*differential*' function.
- Derivative /Differential of the sigmoid function ($f'(x)$) will lies between 0 and 0.25.
- Derivative of the sigmoid function is not "*monotonic*".

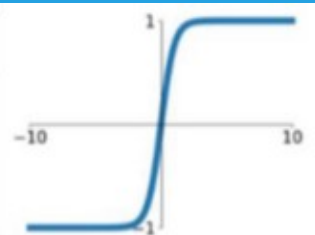
Cons:

- Derivative of sigmoid function suffers "*Vanishing gradient and Exploding gradient problem*".
- Sigmoid function is not "*zero-centric*". This makes the gradient updates go too far in different directions. $0 < \text{output} < 1$, and it makes optimization harder.
- Slow convergence- as its computationally heavy. (Reason use of exponential math function)

“Sigmoid is very popular in classification problems”

2. Tanh Activation Function:

Function	Equation	Range	Derivative
Tanh (Hyperbolic tangent)	$f(x) = \frac{2}{1+e^{-2x}} - 1$	-1,1	$f'(x) = 1 - f(x)^2$

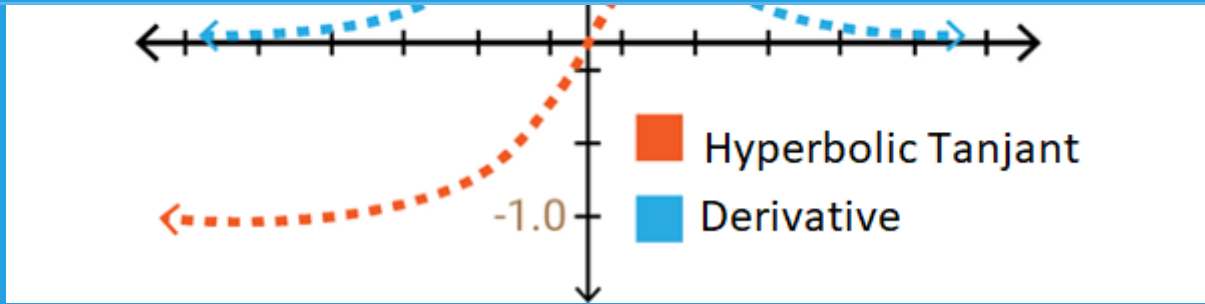


neuron reaches the minimum or maximum value of its range, that



Get started

Open in app



Tanh function and derivative

Tanh is the modified version of sigmoid function. Hence have similar properties of sigmoid function.

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

$$\tanh(x) = 2 \operatorname{sigmoid}(2x) - 1$$

Tanh function(left) , Sigmoidal representation of Tanh(right)

- The function and its **derivative** both are **monotonic**
- Output is zero “*centric*”
- Optimization is *easier*
- Derivative /Differential of the Tanh function ($f'(x)$) will lies between 0 and 1.

Cons:

- Derivative of Tanh function suffers “*Vanishing gradient and Exploding gradient problem*”.
- Slow convergence- as its computationally heavy.(Reason use of exponential math function)

“Tanh is preferred over the sigmoid function since it is zero centered and the gradients are not restricted to move in a certain direction”

3. ReLu Activation Function(ReLu — Rectified Linear Units):

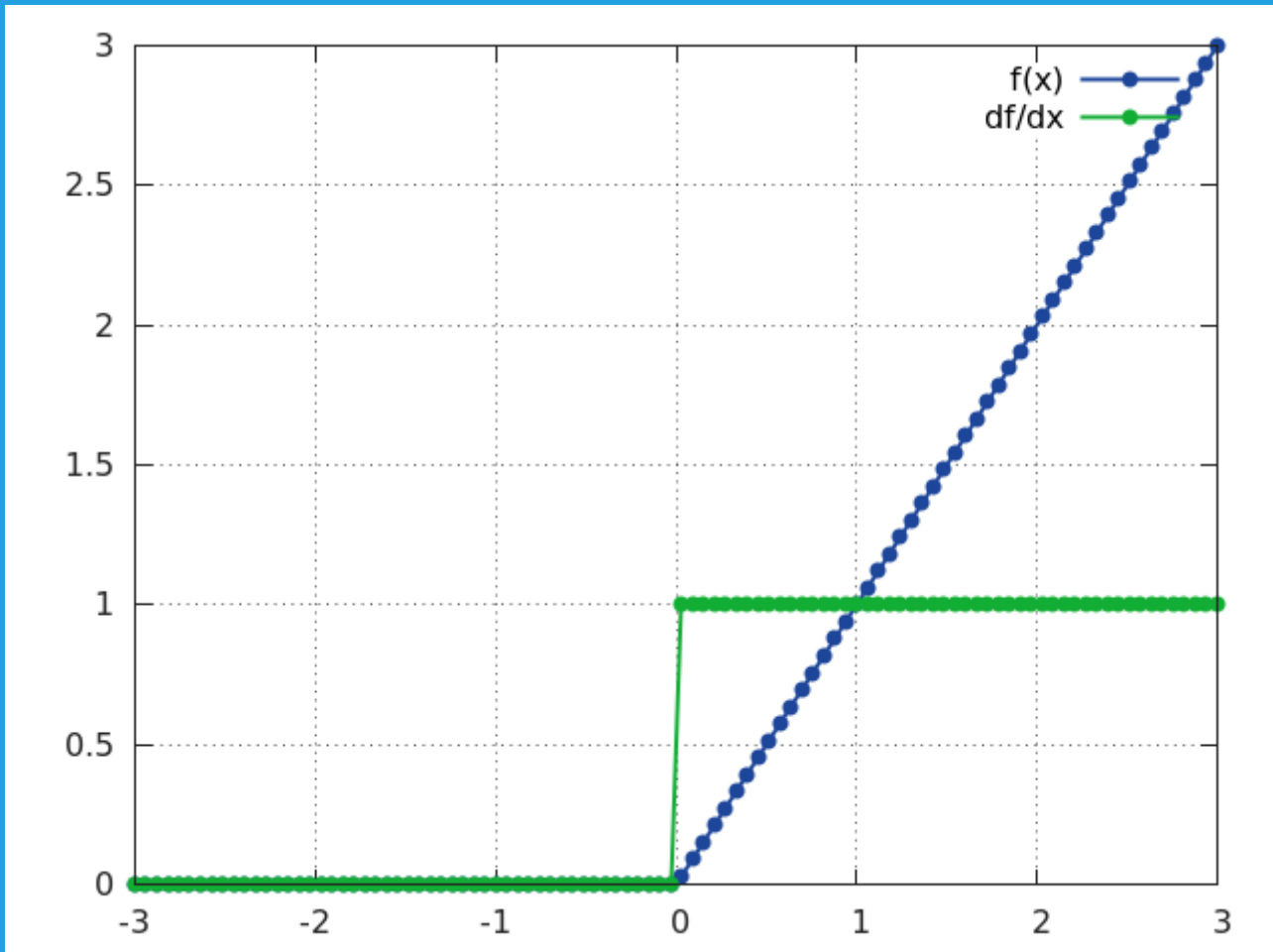
Function	Equation	Range	Derivative

10



Get started

Open in app



ReLU function(Blue) , Derivative of ReLu (Green)

ReLU is the non-linear activation function that has gained popularity in AI. ReLU function is also represented as $f(x) = \max(0, x)$.

- The function and its **derivative** both are **monotonic**.
- Main advantage of using the ReLU function- It does not activate all the neurons at the same time.
- Computationally efficient
- Derivative /Differential of the Tanh function ($f'(x)$) will be 1 if $f(x) > 0$ else 0.
- Converge very fast

Cons:

Get started

Open in app



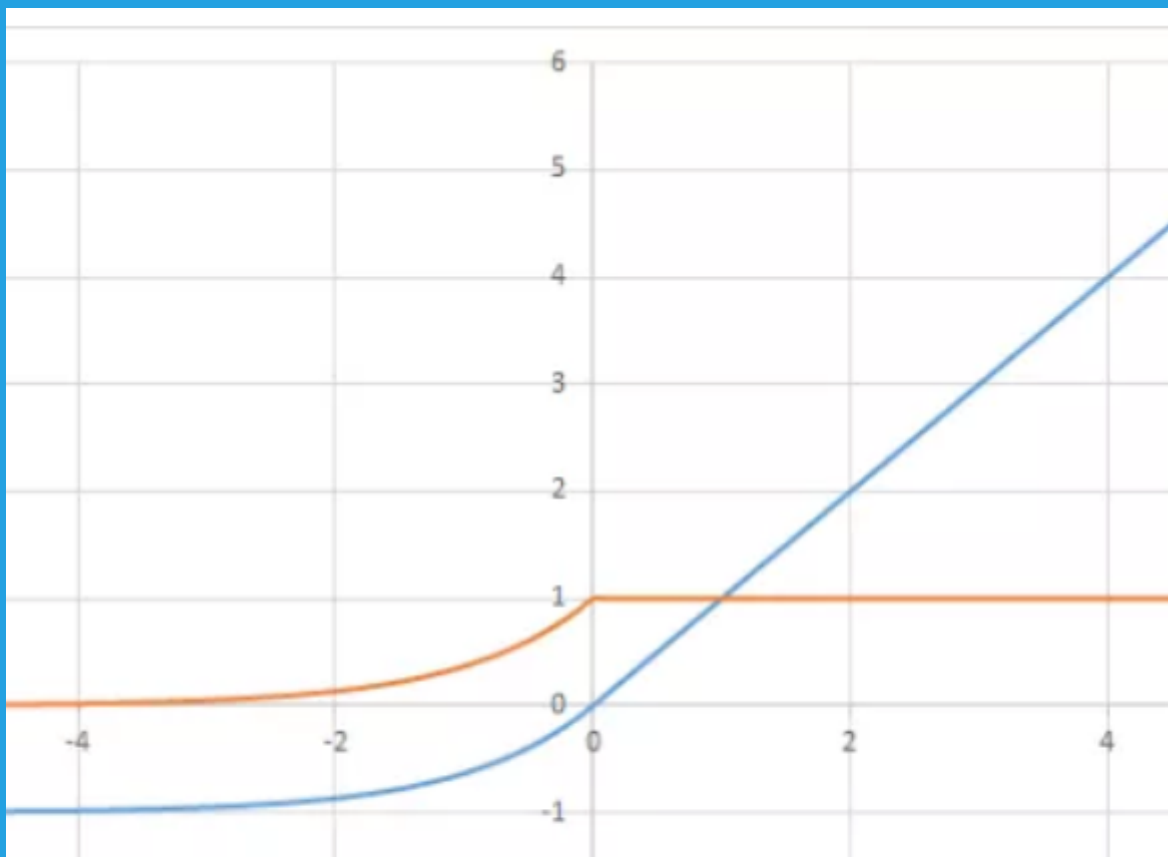
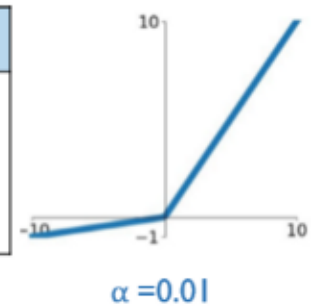
- Dead neuron is the biggest problem. This is due to Non-differentiable at zero.

“Problem of Dying neuron/Dead neuron : As the ReLu derivative $f'(x)$ is not 0 for the positive values of the neuron ($f'(x) = 1$ for $x \geq 0$), ReLu does not saturate (exploid) and no dead neurons (Vanishing neuron) are reported. Saturation and vanishing gradient only occur for negative values that, given to ReLu, are turned into 0- This is called the problem of dying neuron.”

4. leaky ReLu Activation Function:

L leaky ReLU function is nothing but an improved version of the ReLU function with introduction of “constant slope”

Function	Equation	Range	Derivative
Leaky ReLu	$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$-\infty, +\infty$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$



Get started

Open in app



— $f(x)$ — $f'(x)$


Leaky ReLU activation (blue) , Derivative(orange)

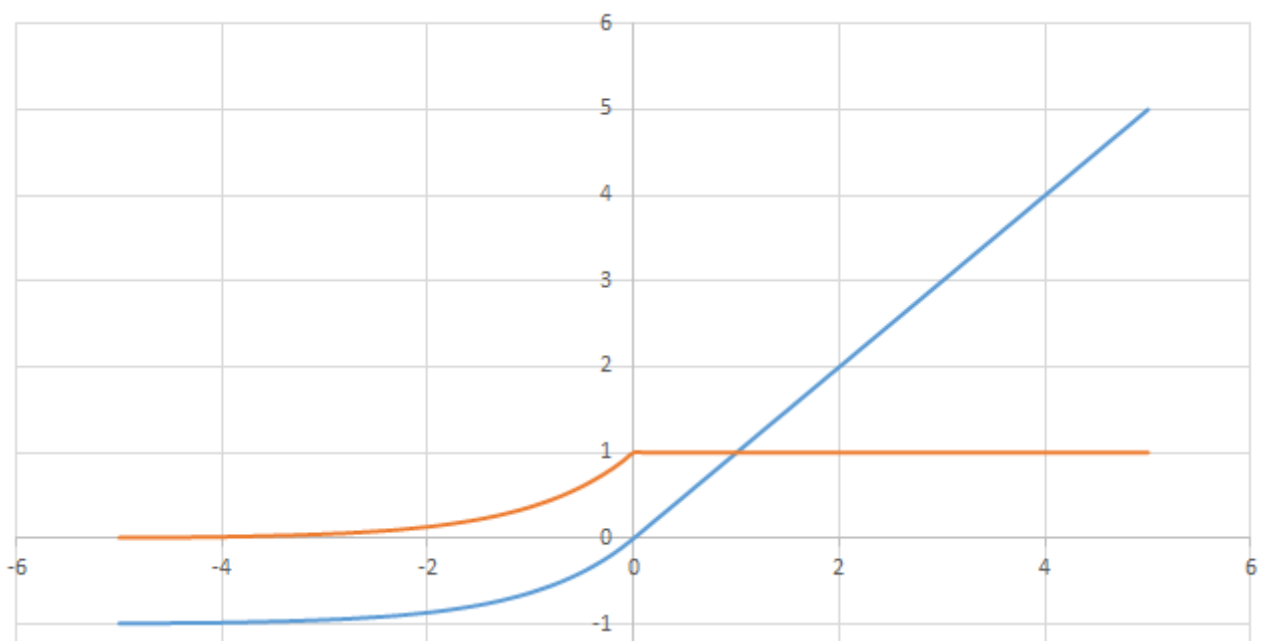
- Leaky ReLU is defined to address **problem of dying neuron/dead neuron**.
- **Problem of dying neuron/dead neuron** is addressed by introducing a small slope having the negative values scaled by α enables their corresponding neurons to “stay alive”.
- The function and its **derivative** both are **monotonic**
- It allows negative value during back propagation
- It is efficient and easy for computation.
- Derivative of Leaky is 1 when $f(x) > 0$ and ranges between 0 and 1 when $f(x) < 0$.

Cons:

- Leaky ReLU does not provide consistent predictions for negative input values.

5. ELU (Exponential Linear Units) Activation Function:

Exponential linear unit (ELU) ^[23]		$f(\alpha, x) = \begin{cases} \alpha(e^x - 1) & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} f(\alpha, x) + \alpha & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$	$(-\alpha, \infty)$
--	---	--	---	---------------------



Get started

Open in app



— $f(x)$ — $f'(x)$


ELU and its derivative

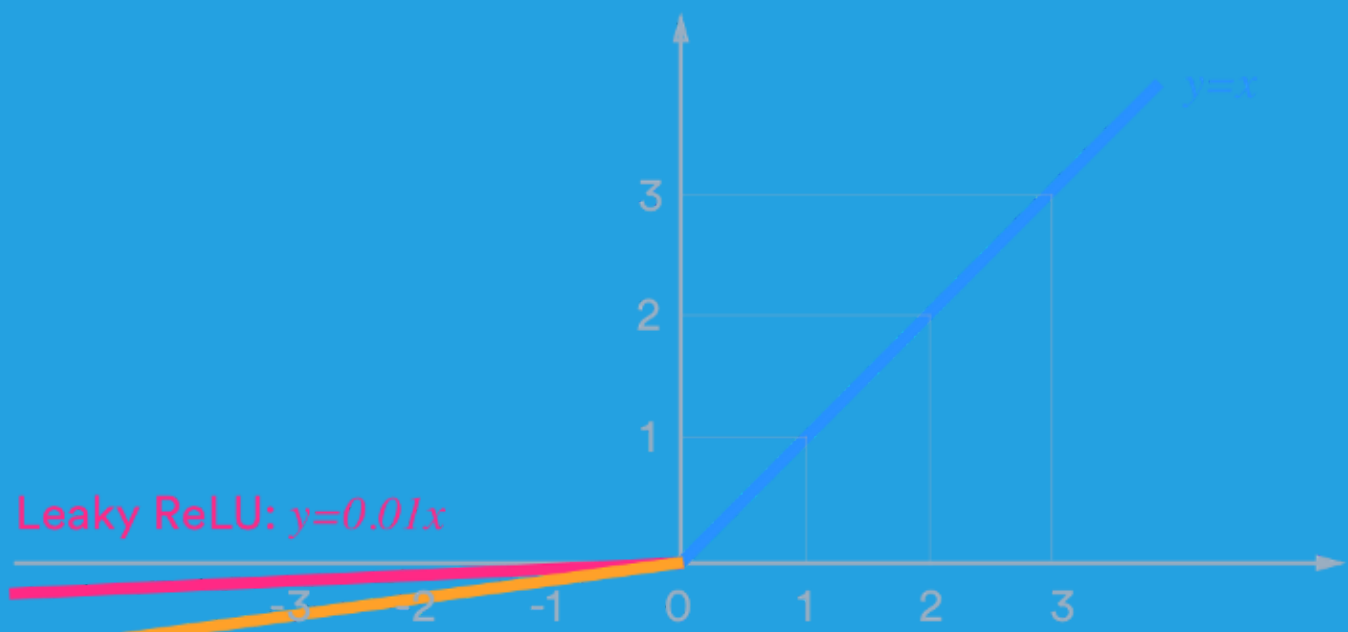
- ELU is also proposed to solve the **problem of dying neuron**.
- No Dead ReLU issues
- Zero-centric

Cons:

- Computationally intensive.
- Similar to Leaky ReLU, although theoretically better than ReLU, there is currently no good evidence in practice that ELU is always better than ReLU.
- $f(x)$ is monotonic only if alpha is greater than or equal to 0.
- $f'(x)$ derivative of ELU is monotonic only if alpha lies between 0 and 1.
- Slow convergence due to exponential function.

6. P ReLu (Parametric ReLU) Activation Function:

Parametric rectified linear unit (PReLU) ^[20]		$f(\alpha, x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$(-\infty, \infty)^{[2]}$
--	---	---	--	---------------------------



Get started

Open in app



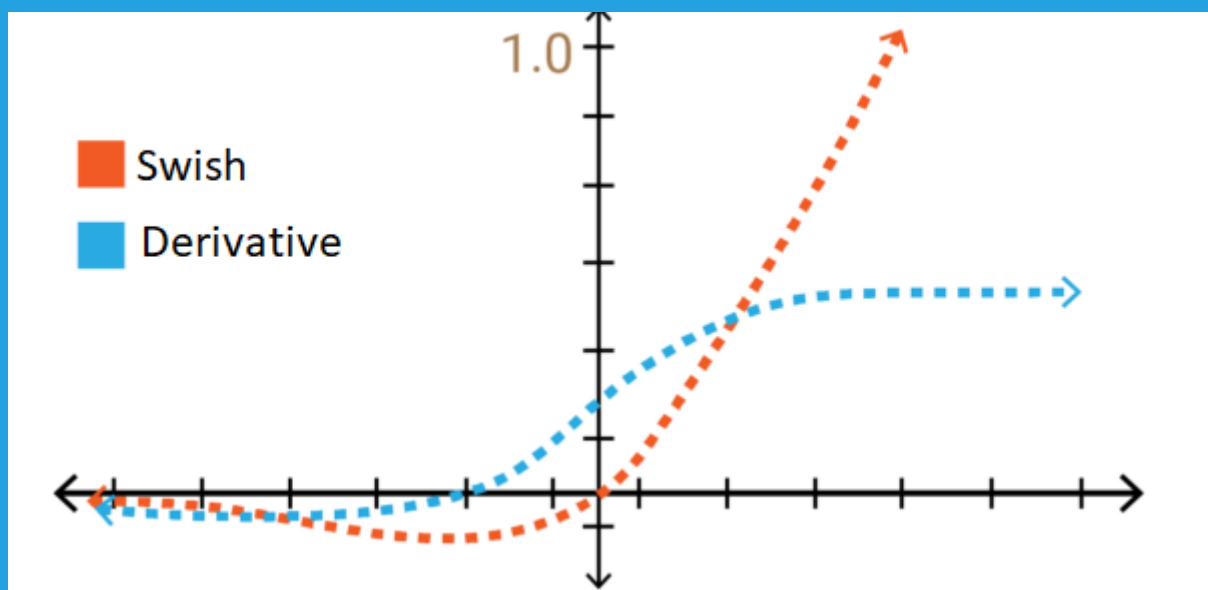
Leaky ReLU vs P Relu

- The idea of leaky ReLU can be extended even further.
- Instead of multiplying x with a constant term we can multiply it with a “**hyperparameter (a-trainable parameter)**” which seems to work better the leaky ReLU. This extension to leaky ReLU is known as **Parametric ReLU**.
- The parameter α is generally a number between 0 and 1, and it is generally relatively small.
- Have slight advantage over Leaky Relu due to trainable parameter.
- Handle the **problem of dying neuron**.

Cons:

- Same as leaky Relu.
- $f(x)$ is monotonic when $\alpha > 0$ or $\alpha = 0$ and $f'(x)$ is monotonic when $\alpha = 1$

7. Swish (A Self-Gated) Activation Function:(Sigmoid Linear Unit)



- Google Brain Team has proposed a new activation function, named **Swish**, which is simply $f(x) = x \cdot \text{sigmoid}(x)$.

Get started

Open in app



- The curve of the *Swish function is smooth* and the function is *differentiable* at all points. This is helpful during the model optimization process and is considered to be one of the reasons that swish outperforms ReLU.
- Swish function is “*not monotonic*”. This means that the value of the function may decrease even when the input values are increasing.
- **Function is unbounded above and bounded below.**

$$\begin{aligned} f(x) &= x * \text{sigmoid}(x) \\ &= x * (1 + e^{-x})^{-1} \end{aligned}$$

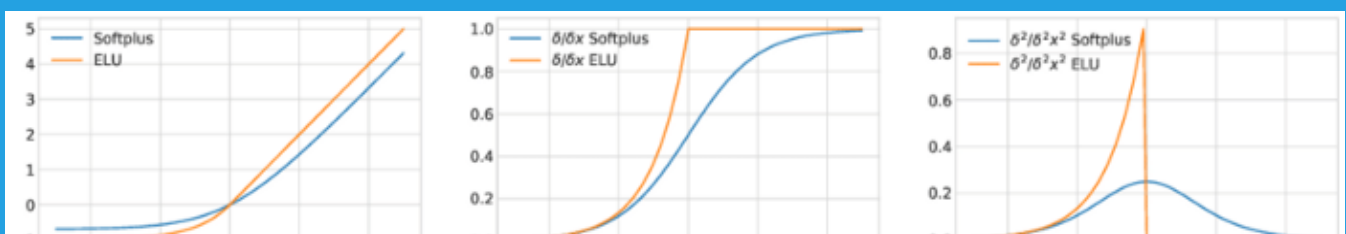
“Swish tends to continuously match or outform the ReLU”

Note that the output of the swish function may fall even when the input increases. This is an interesting and swish-specific feature. (Due to non-monotonic character)

$$f(x) = 2x * \text{sigmoid}(\beta x)$$

If we think that $\beta = 0$ is a simple version of Swish, which is a learnable parameter, then the sigmoid part is always $1/2$ and $f(x)$ is linear. On the other hand, if the β is a very large value, the sigmoid becomes a nearly double-digit function (0 for $x < 0$, 1 for $x > 0$). Thus $f(x)$ converges to the ReLU function. **Therefore, the standard Swish function is selected as $\beta = 1$.** In this way, a soft interpolation (associating the variable value sets with a function in the given range and the desired precision) is provided. Excellent! A solution to the problem of the vanish of the gradients has been found.

8. Softplus



Get started

Open in app



The softplus function is similar to the ReLU function, but it is relatively smoother. Function of Softplus or SmoothRelu $f(x) = \ln(1 + \exp x)$.

Derivative of the Softplus function is $f'(x)$ is logistic function $(1/(1 + \exp x))$.

Function value ranges from $(0, +\infty)$. Both $f(x)$ and $f'(x)$ are **monotonic**.

9. Softmax or normalized exponential function:

Name ↕	Equation ↕	Derivatives ↕	Range ↕
Softmax	$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^J e^{x_j}}$ for $i = 1, \dots, J$	$\frac{\partial f_i(\vec{x})}{\partial x_j} = f_i(\vec{x})(\delta_{ij} - f_j(\vec{x}))$ ^[5]	$(0, 1)$

The “softmax” function is also a type of sigmoid function but it is very useful to handle *multi-class classification problems*.

“Softmax can be described as the combination of multiple sigmoidal function.”

“Softmax function returns the probability for a datapoint belonging to each individual class.”

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \text{ for } j = 1, \dots, K$$

While building a network for a multiclass problem, the output layer would have as many neurons as the number of classes in the target.

For instance if you have *three classes* $[A, B, C]$, there would be three neurons in the output layer. Suppose you got the output from the neurons as $[2.2, 4.9, 1.75]$. Applying the softmax function over these values, you will get the following result — $[0.52, 0.21, 0.27]$. These represent the probability for the data point belonging to each class. From result we can that the input belong to class A.

[Get started](#)[Open in app](#)

Which one is better to use : How to choose a right one:

To be honest there is no hard and fast rule to choose the activation function. We can't differentiate between activation function. Each activation function has its own pro's and con's. All the good and bad will be decided based on the trail.

But based on the properties of the problem we might be able to make a better choice for easy and quicker convergence of the network.

- Sigmoid functions and their combinations generally work better in the case of classification problems
- Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
- ReLU activation function is widely used in modern era.
- In case of dead neurons in our networks due to ReLU then leaky ReLU function is the best choice
- ReLU function should only be used in the hidden layers

“As a rule of thumb, one can begin with using ReLU function and then move over to other activation functions in case ReLU doesn't provide with optimum results”

Reference:

Fundamentals of Deep Learning - Activation Functions and When to Use Them?

Overview Activation function is one of the building blocks on Neural Network. Learn about the different activation...

www.analyticsvidhya.com



MissingLink: Deep Learning for Computer Vision - 10X Faster

Run experiments on hundreds of machines, on and off the cloud, manage huge data sets and gain unprecedented visibility...

missinglink.ai



