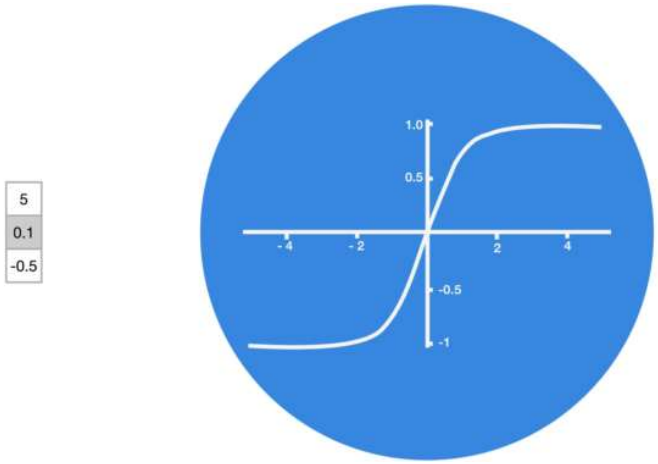
Sukanya Bag

Summary

The provided web content offers a comprehensive overview of various

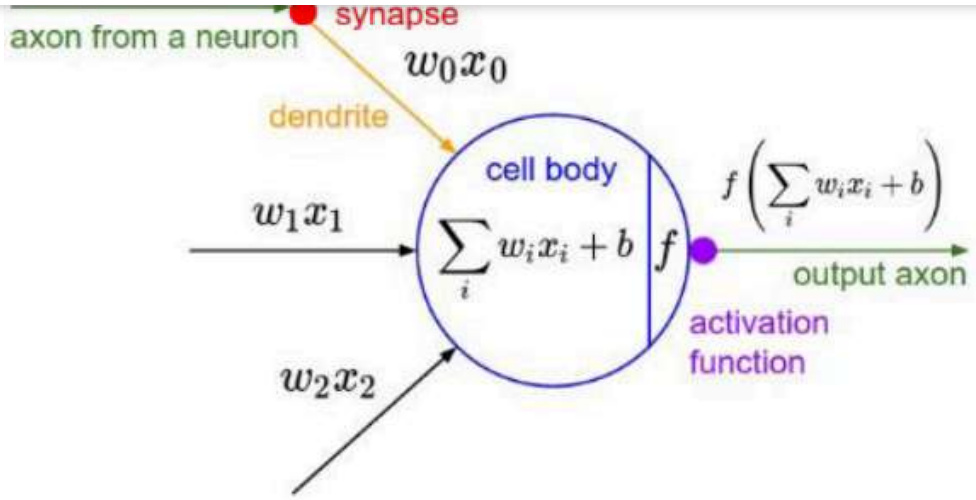
Activation Functions — All You Need To Know!



An **activation function** is a **function** that is added to an **artificial neural network** in order to help the network learn **complex patterns in the data**. When comparing with a neuron-based model that is in our brains, the **activation function** is at the end deciding what is to be fired to the next neuron.

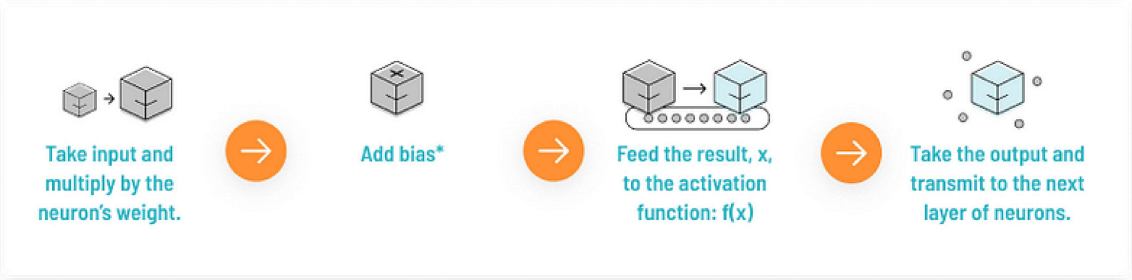
> In **artificial neural networks**, the **activation function** of a node defines the output of that node given an input or set of inputs. A standard **integrated circuit** can be seen as a **digital network** of activation functions that can be "ON" (1) or "OFF" (0), depending on input. — **Wikipedia**

So, summing it up, **activation functions are mathematical equations that determine the output of a neural network.**

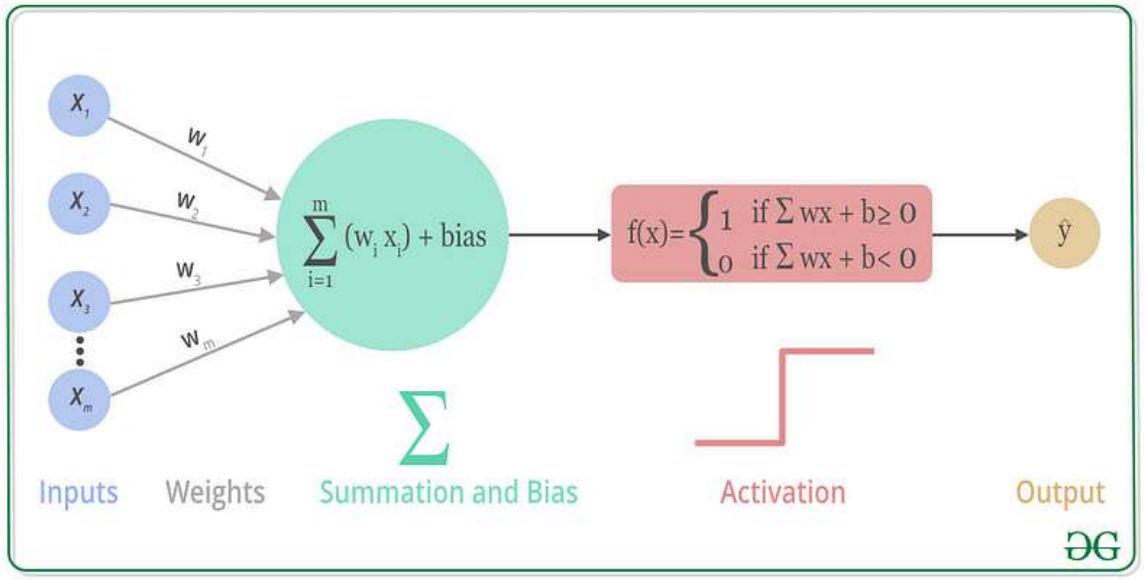


In this blog, we will learn about — the widely used Activation Functions, the backend mathematics behind its working, and discuss various ways on how to choose the best one for your specific deep learning problem statement.

Before jumping in-depth about the different types of activation functions, let's take a quick look into how an artificial neuron works -



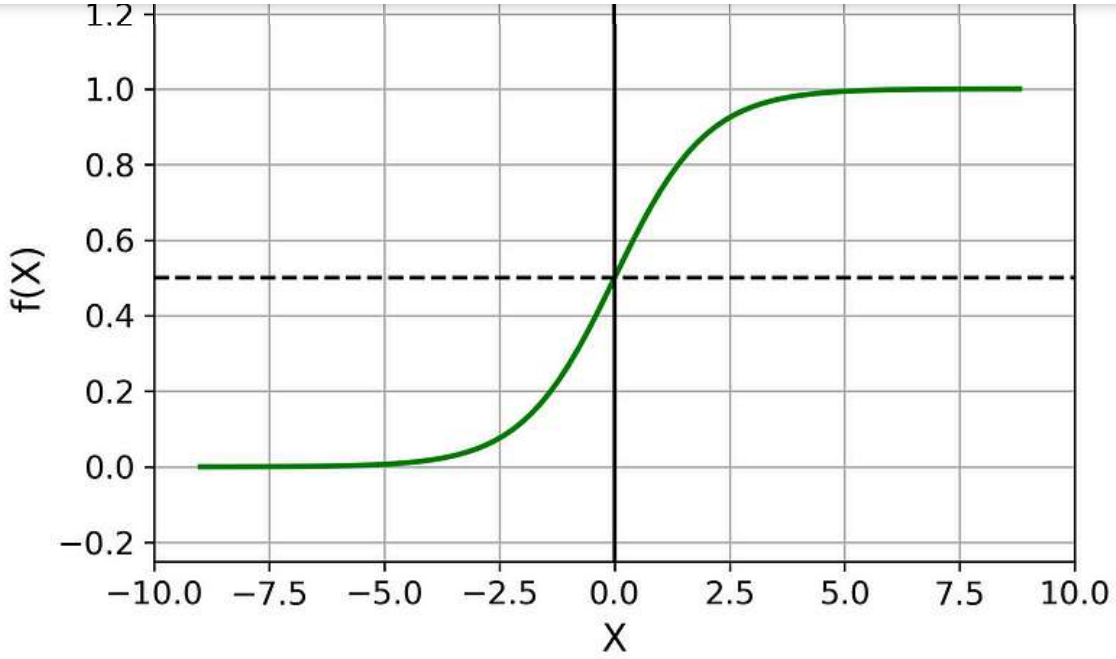
A mathematical visualization of the processes described above can be shown as-



By now, you must be very well acquainted with the process of how an ANN works and what is the role of Activation Function in the process!

So, grab your coffee , and let's begin!

1. Sigmoid Activation Function -



The **Sigmoid Function** looks like an **S-shaped** curve.

Formula : **f(z) = 1/(1+ e^-z)**

Why and when do we use the Sigmoid Activation Function?

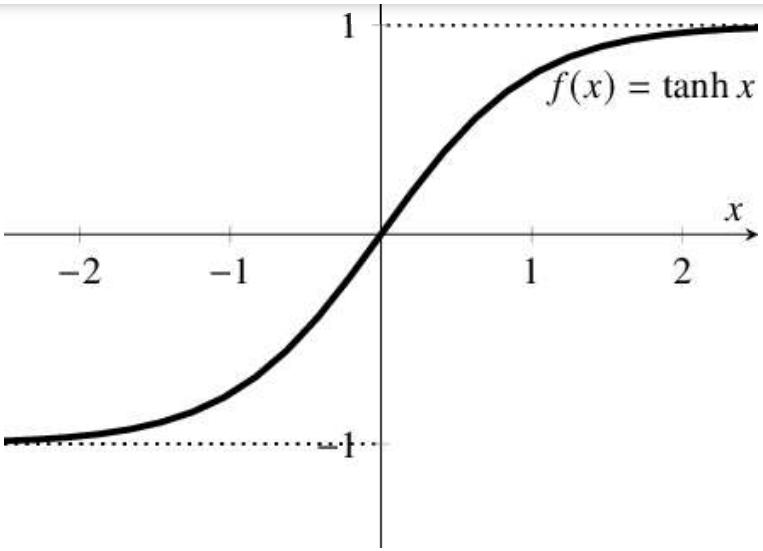
- The output of a sigmoid function **ranges between 0 and 1**. Since, output values bound between 0 and 1, it **normalizes** the output of each neuron.
- Specially used for models where we have to **predict the probability** as an output. Since the probability of anything exists only between the

- **Smooth gradient**, preventing “jumps” in output values.
- The function is **differentiable**.That means, we can find the slope of the sigmoid curve at any two points.
- **Clear predictions**, i.e very close to 1 or 0.

What are some **disadvantages** of the Sigmoid activation function?

- Prone to gradient vanishing (when the **sigmoid** function value is either too high or too low, the derivative becomes very small i.e. << 1. This causes **vanishing gradients** and poor learning for deep networks.)
- The function output is **not centered on 0,** which will reduce the efficiency of weight update.
- The sigmoid function performs exponential operations, which is slower for computers.

2. Tanh or Hyperbolic Tangent Activation Function -



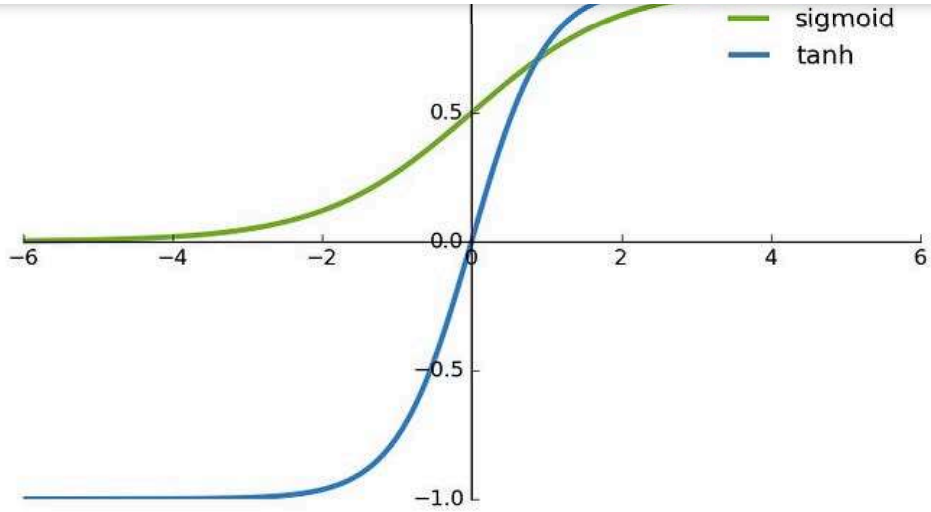
The tanh activation function is also **sort of sigmoidal (S-shaped).**

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

Formula of tanh activation function

Tanh is a hyperbolic tangent function. The curves of tanh function and sigmoid function are relatively similar. But it has some advantage over the sigmoid function. Let's look at what it is.

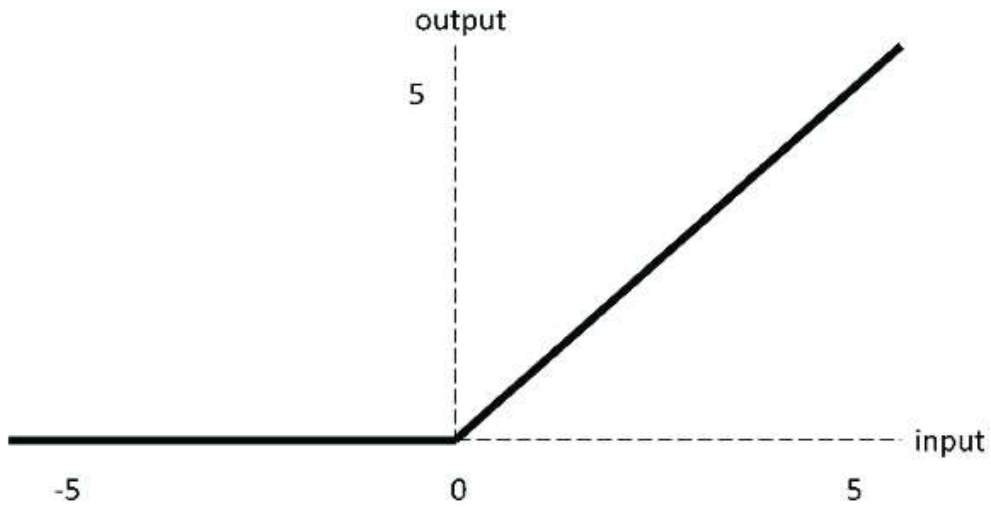
Why is tanh **better compared** to sigmoid activation function?



- First of all, when the input is large or small, the output is **almost smooth and the gradient is small**, which is not conducive to weight update. The difference is the output interval. The output interval of tanh is 1, and the whole function is **0-centric, which is better than sigmoid.**
- The **major advantage** is that the **negative inputs** will be mapped strongly **negative** and the **zero inputs** will be mapped **near zero** in the tanh graph.

> **Note:** In general **binary classification problems**, the tanh function is used for the **hidden layer** and the sigmoid function is used for the **output layer**. However, these are **not static**, and the specific activation

3. ReLU (Rectified Linear Unit) Activation Function-



The ReLU is half rectified (from the 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.

$$\sigma(x) = \begin{cases} max(0, x) & , x >= 0 \\ 0 & , x < 0 \end{cases}$$

The **ReLU (Rectified Linear Unit)** function is an activation function that is currently **more popular** compared to other activation functions in deep learning.

Compared with the sigmoid function and the tanh function, it has the following **advantages**:

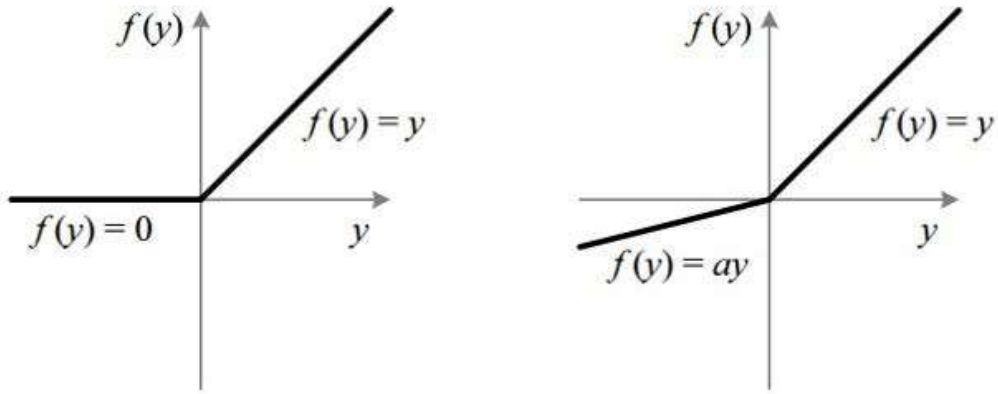
- When the input is positive, there is **no gradient saturation problem.**
- The calculation speed is much **faster**. The ReLU function has only a linear relationship. Whether it is forward or backward, it is much faster than sigmoid and tanh. (Sigmoid and tanh need to calculate the exponent, which will be slower.)

Of course, there are **disadvantages:**

1) **Dead ReLU problem**- When the input is negative, ReLU is completely **inactive**, which means that once a negative number is entered, **ReLU will die**. In this way, in the forward propagation process, it is not a problem. Some areas are sensitive and some are insensitive. But in the **back propagation** process, if you enter a negative number, the **gradient will be completely zero,** which has the **same** problem as the sigmoid function and tanh function.

4. Leaky ReLU Activation Function-

An activation function **specifically designed to compensate** for the dying ReLU problem.



ReLU vs Leaky ReLU

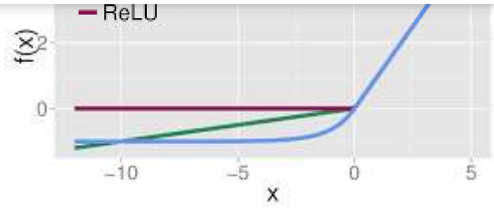
Why Leaky ReLU is **better** than ReLU?

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}.$$

- The leaky ReLU **adjusts the problem of zero gradients** for negative value, by giving a very **small linear component of x** to negative inputs**(0.01x)**.
- The leak helps to increase the range of the ReLU function. Usually, the value of **a** is **0.01** or so.
- Range of the Leaky ReLU is **(-infinity to infinity).**

> **Note :** In theory, Leaky ReLU has all the advantages of ReLU, plus there will be no problems with Dead ReLU, but in actual operation, it has not been fully proved that Leaky ReLU is always better than ReLU.

5. ELU (Exponential Linear Units) function-



ELU vs Leaky ReLU vs ReLU

ELU is also **proposed to solve the problems** of ReLU. In contrast to ReLUs, ELUs have **negative values** which pushes the **mean of the activations** closer to **zero.** Mean activations that are closer to zero **enable faster learning** as they bring the **gradient closer to the natural gradient.**

$$g(x) = \mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leqslant 0 \end{cases}$$

Obviously, **ELU has all the advantages of ReLU**, and:

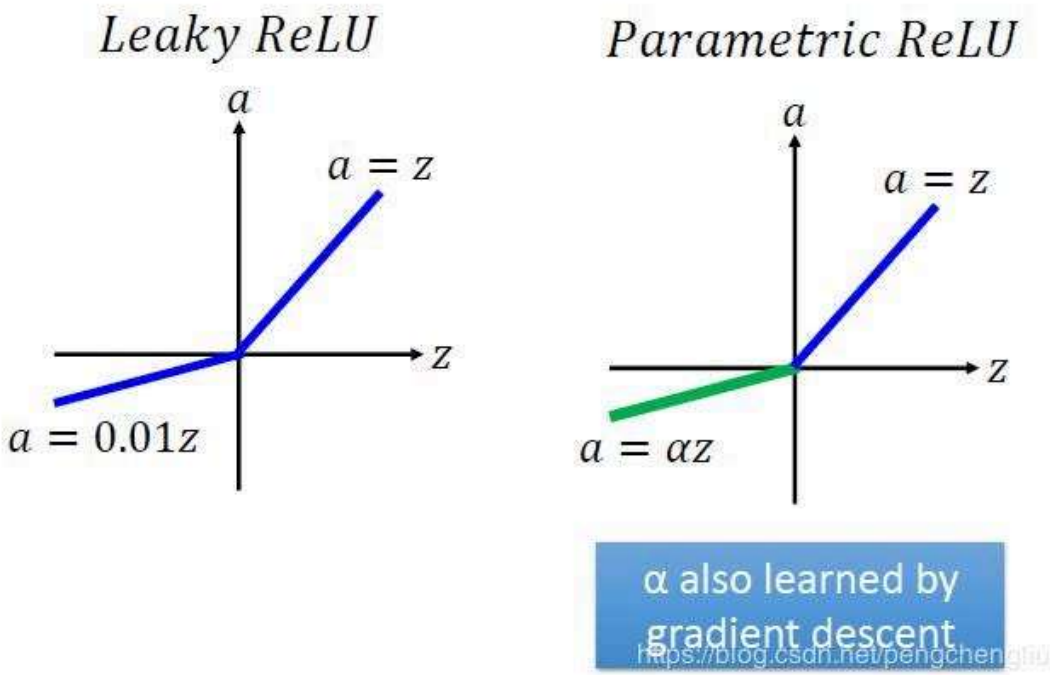
- **No Dead ReLU** issues,the mean of the output is close to 0, **zero-centered**.
- In **contrast** to ReLUs, ELUs have negative values which allows them to push mean unit activations closer to zero like **batch normalization** but

natural gradient because of a **reduced bias shift effect.**

- ELUs **saturate to a negative value** with smaller inputs and thereby decrease the forward propagated variation and information.

One **small problem** is that it is slightly more **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.

6. PRelu (Parametric ReLU)-



PReLU is also an **improved** version of **ReLU.**

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

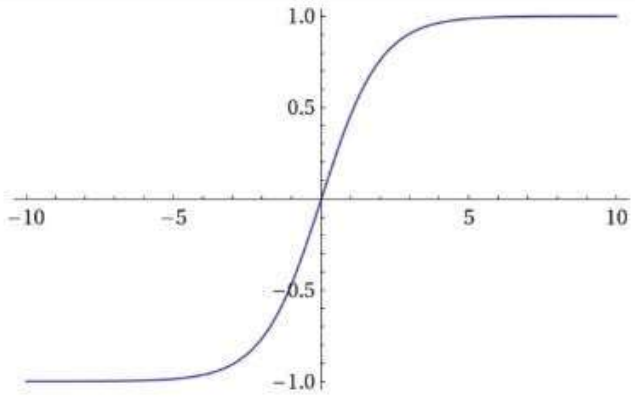
between 0 and 1, and it is generally relatively small, such as a few zeros.

- if $\mathbf{a_i=0}$, f becomes ReLU
- if $\mathbf{a_i>0}$, f becomes leaky ReLU
- if $\mathbf{a_i}$ **is a learnable parameter**, f becomes PReLU

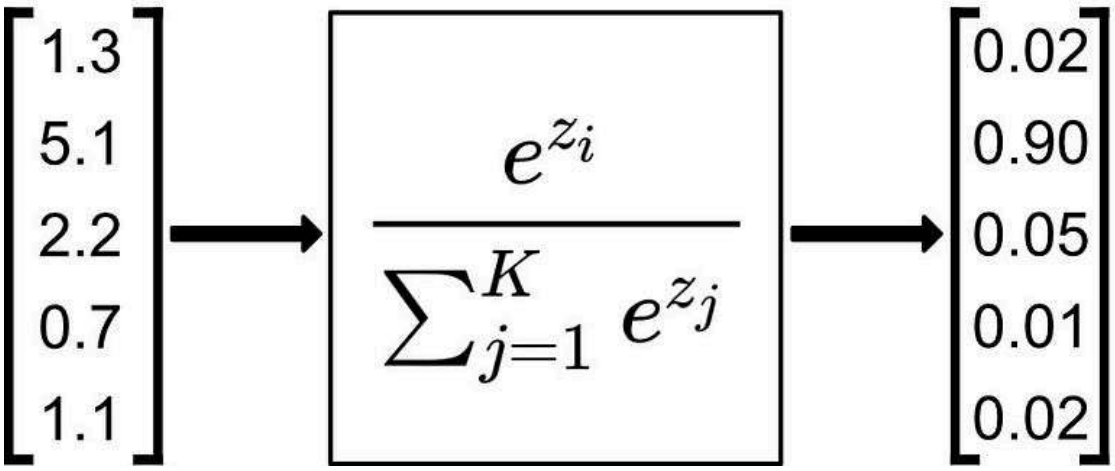
Coming to the **advantages** of PReLU-

- In the negative region, PReLU has a **small slope**, which can also **avoid** the problem of **ReLU death**.
- Compared to ELU, PReLU is a **linear operation** in the negative region. Although the slope is small, it **does not tend to 0**, which is a certain advantage.

7. Softmax



Softmax is used as the **activation function** for multi-class classification problems where class membership is required on more than two class labels. For an arbitrary real vector of length K, Softmax can **compress it into a real vector of length K** with a value in the **range (0, 1)**, and the sum of the elements in the vector is 1.



> Softmax is different from the normal max function: the max function only outputs the largest value, and Softmax ensures that smaller values have a smaller probability and will not be discarded directly. It is a **"max"** that is **"soft"**; it can be thought to be a **probabilistic or "*softer*" version of the argmax function.**

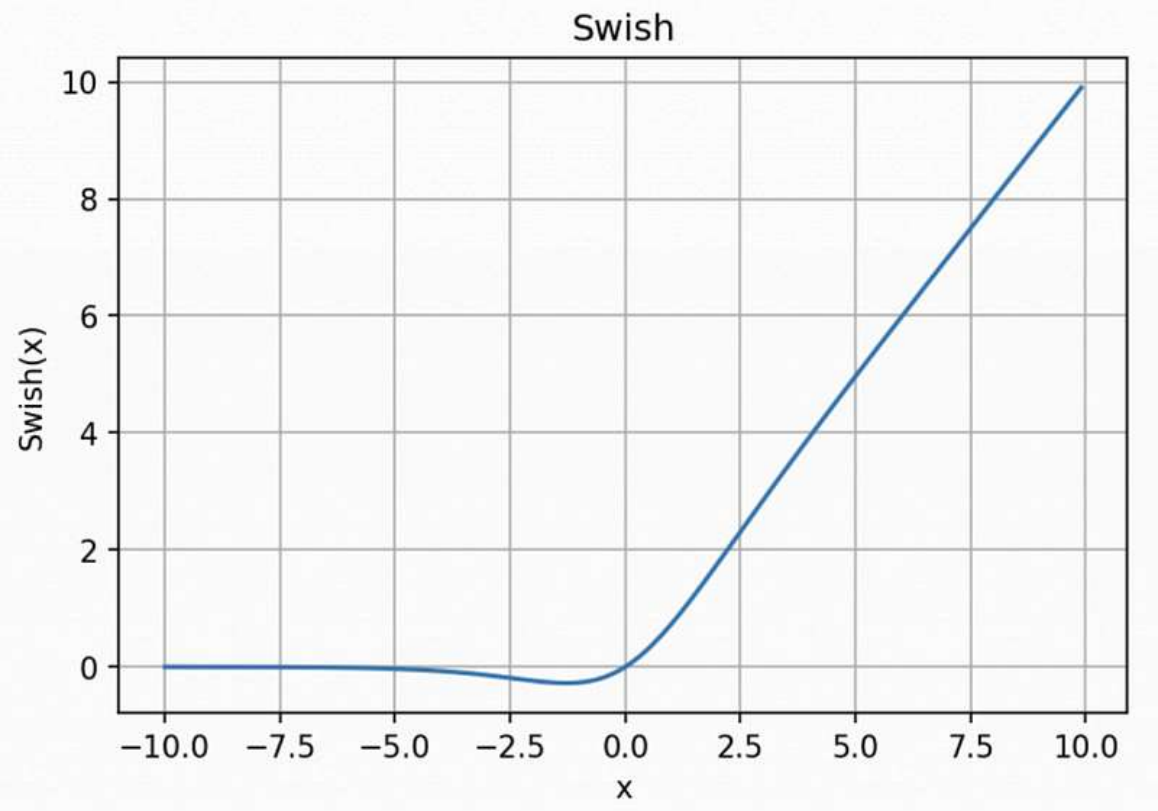
The denominator of the Softmax function combines all factors of the original output value, which means that the different probabilities obtained by the Softmax function are related to each other.

The **major drawback** in the softmax activation function is that it is -

- Non-differentiable at zero and ReLU is unbounded.

create dead neurons that never get activated.

8. Swish (A Self-Gated) Function



The formula is: **y = x * sigmoid (x)**

LSTMs and highway networks. We use the same value for gating to simplify the gating mechanism, which is called **self-gating**.

> The **advantage of self-gating** is that it only requires a **simple scalar input,** while normal gating requires multiple scalar inputs. This feature enables self-gated activation functions such as Swish to easily **replace activation functions that take a single scalar as input (such as ReLU)** without changing the hidden capacity or number of parameters.

Note: Swish activation function can only be implemented when your **neural network is ≥ 40 layers.**

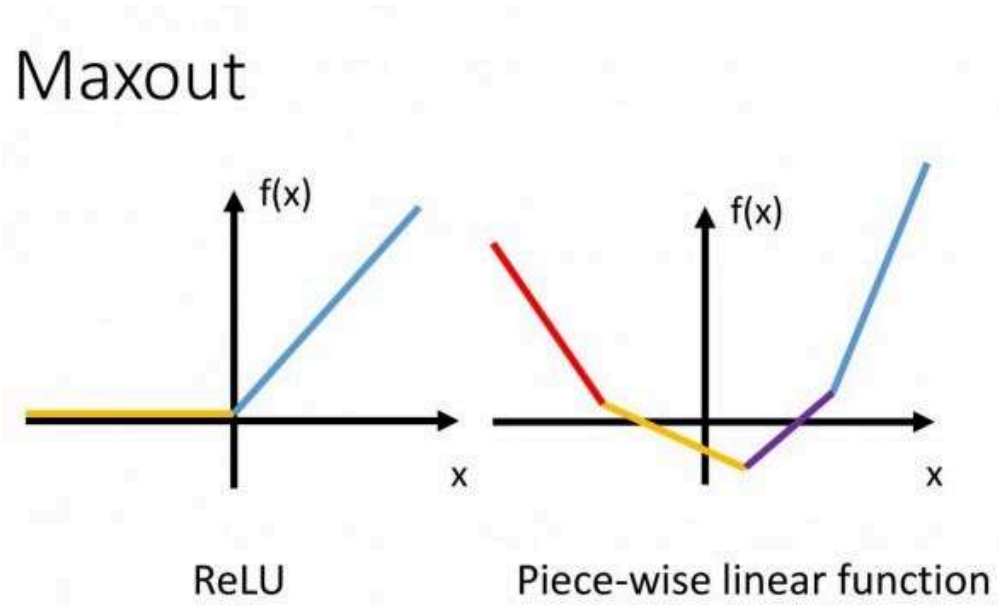
The **major advantages** of the Swish activation function are as follows:

1.**Unboundedness** is helpful to prevent the gradient from gradually approaching 0 during slow training, causing saturation.

(At the same time, being bounded has advantages, because bounded active functions can have strong regularization, and larger negative inputs will be resolved.)

2. Derivative **always > 0.**

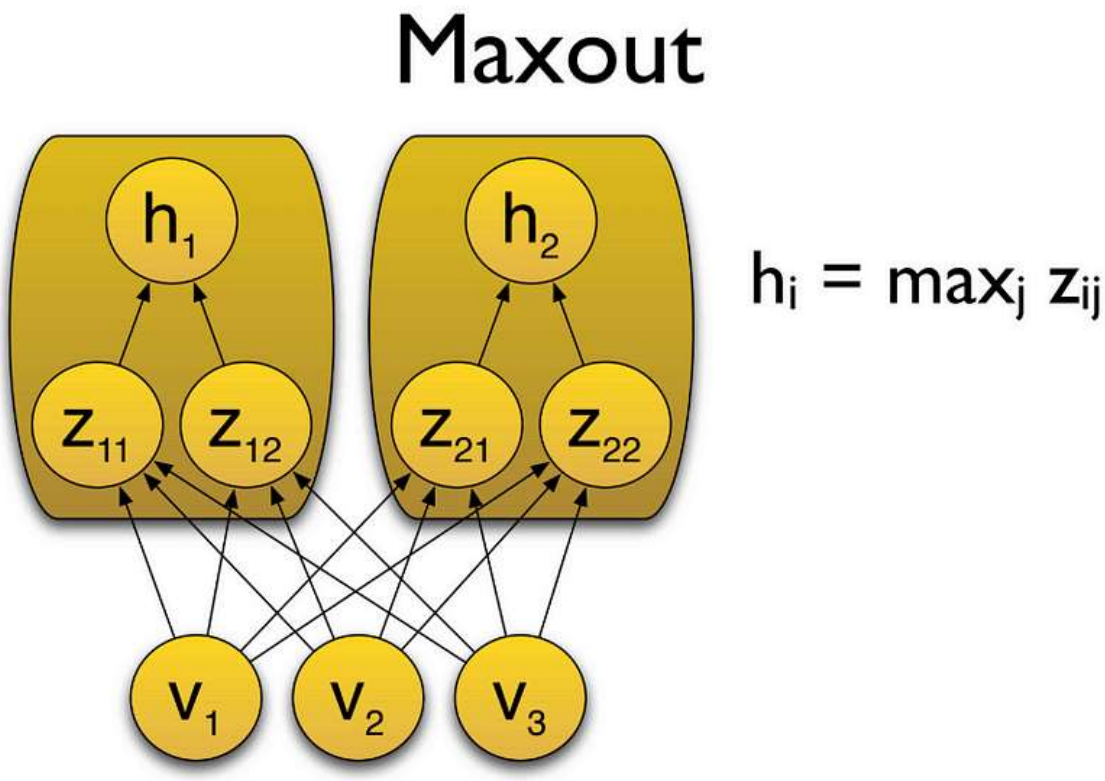
3. **Smoothness** also plays an important role in **optimization and generalization.**



A **Maxout** layer is simply a layer where the activation function **is the max of the inputs.** As stated in the **paper**, even an MLP with **2 maxout units** can **approximate any function.**

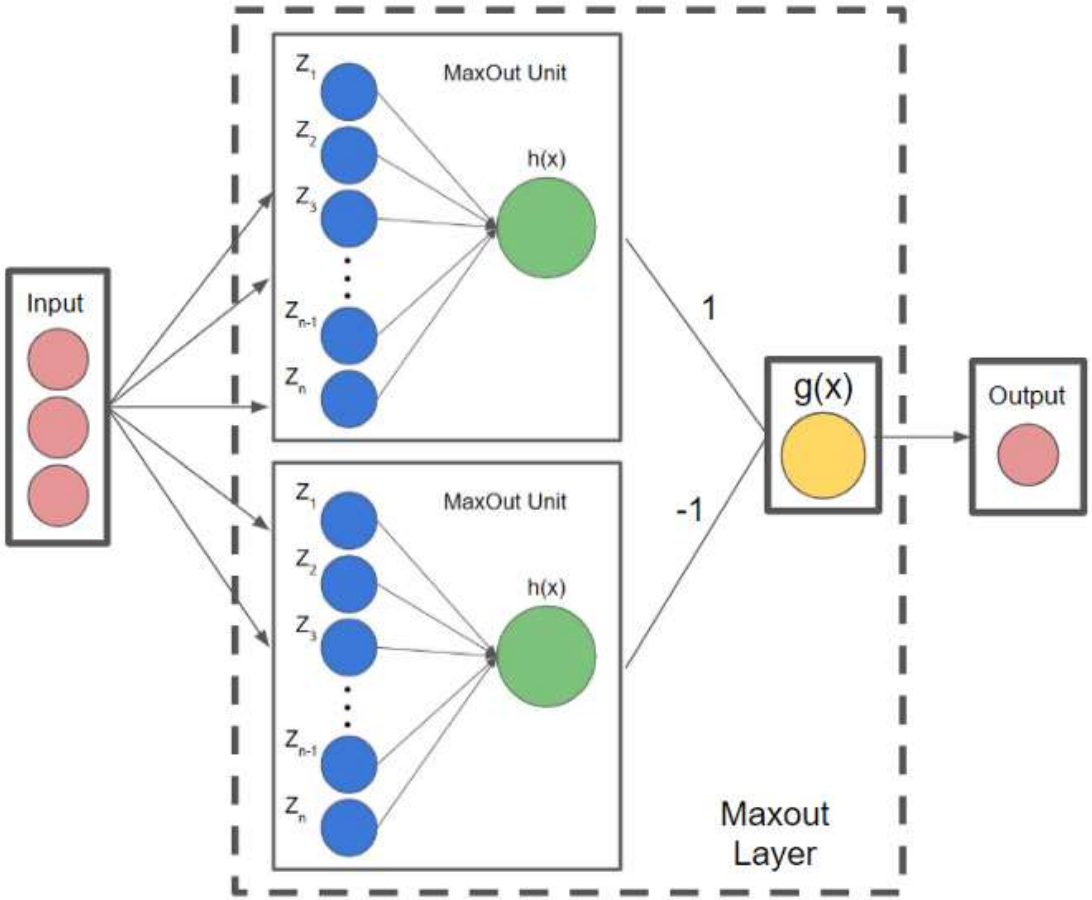
A single Maxout unit can be interpreted as making a **piecewise linear approximation (PWL)** to a **real-valued function** where the line segment between any two points on the graph of the function lies above or on the graph (**convex function**).

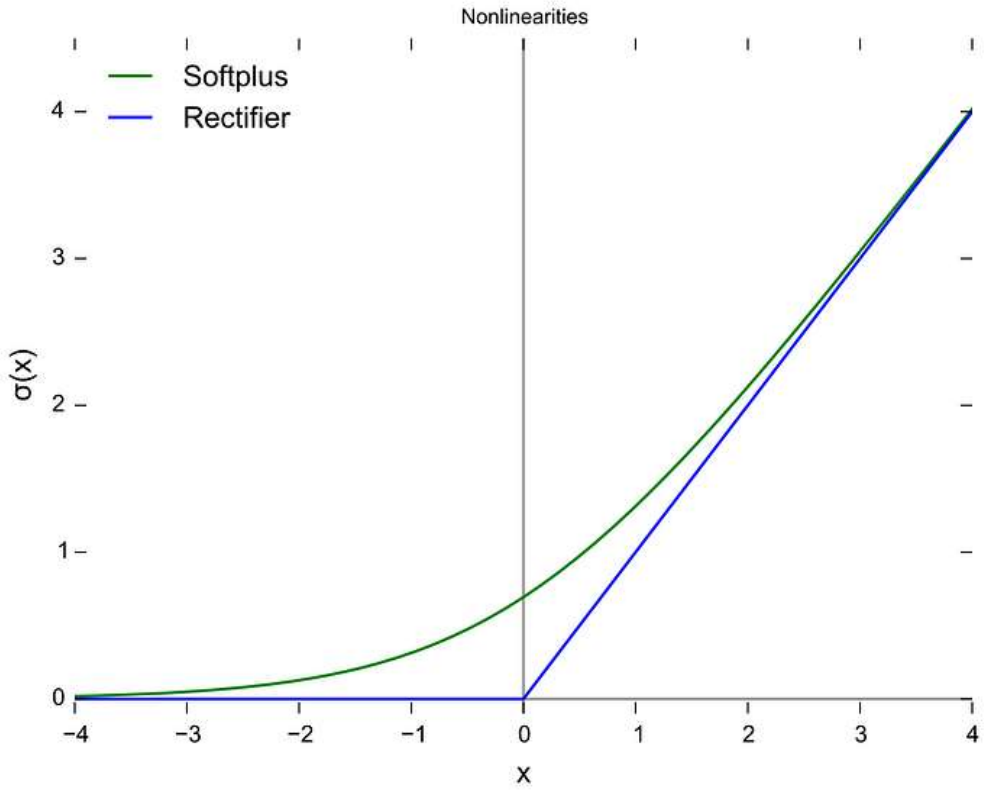
Maxout can also be implemented for a **d-dimensional vector(V).**



Consider, two convex functions **h1(x) and h2(x)**, approximated by two Maxout units. By the above preposition, the function **g(x) is a continuous PWL function.**

Hence, it is found that a **Maxout layer consisting of two Maxout units can approximate any continuous function arbitrarily well.**





Softplus function: f(x) = **ln(1+exp x)**

The **derivative** of softplus is -

f ′(x)=exp(x) / (1+exp x)

= **1/ (1 +exp(−x))**

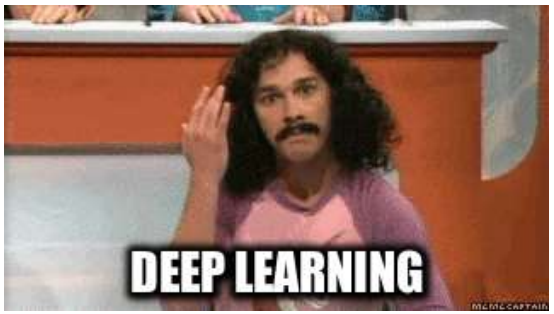
The softplus function is similar to the ReLU function, but it is **relatively smooth.** It is unilateral suppression like ReLU.

It has a wide acceptance range **(0, + inf)**.

> Generally speaking, these activation functions have their own advantages and disadvantages. All the good and bad must be obtained by experimenting them with various problem statements.

And, with this, we come to the end of this blog. Hope this will give you a decent knowledge about the most used activation functions in deep learning.

😎



If you are a beginner in Data Science and Machine Learning and have some specific queries with regard to Data Science/ML-AI,

Interview before your D-Day, feel free to book a 1:1 call here. I will be happy to help!

Until next time! 🤓

LinkedIn

Deep Learning Activation Functions

Recommended from ReadMedium

Code Thulo

Top Large Language Model (LLM) Interview Question | Basic LLM Questions

Large Language Models (LLMs) are deep learning models that are trained on huge amounts of text data. They use a transformer architecture...

7 min read


Juan C Olamendy

Have you ever spent days training a deep learning model, only to find out it performs poorly on new data?

5 min read

Jo Wang

Deep Learning Part 2—Neural Network and the critical Activation Functions

Neural Network Structure

4 min read

Abhishek Jain

Difference between ANN and CNN

The key difference between Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) lies in their architectures and the...

3 min read

Wahab Writer

How to Earn $50 in One Day by Writing Articles

How to Earn $50 in One Day by Writing Articles

5 min read

The Role of AI in Contemporary Art

Artificial Intelligence (AI) is reshaping the artwork international, mixing generation with creativity to provide works that project...

2 min read

LM Po

Understanding LLM Decoding Strategies

The emergence of ChatGPT at the end of 2022 revolutionized the world with its ability to generate human-like text responses. At its core...

11 min read

Vyacheslav Efimov

Understanding Deep Learning Optimizers: Momentum, AdaGrad, RMSProp & Adam

Gain intuition behind acceleration training techniques in neural networks

8 min read

Jorgecardete

Convolutional Neural Networks: A Comprehensive Guide

Exploring the power of CNNs in image analysis

Mikel

Pinterest ML Internship Summer 2025

Looking for a tech internship for the Summer 2025. I share my recent Interview experience, Questions, Solutions and tips.

6 min read

Rohollah

Mastering Feature Selection: Key Applications and Differences—Part 2: Chi-Square Test

Applications of Chi-Square in Feature Selection.How to use Chie-square for feature selection? Using Chi-square in Machine Learning Projects

9 min read