UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Devan Saragoi, Teaching Assistant

PES UNIVERSITY

Unit 1: Introduction to Deep Learning

Activation Functions

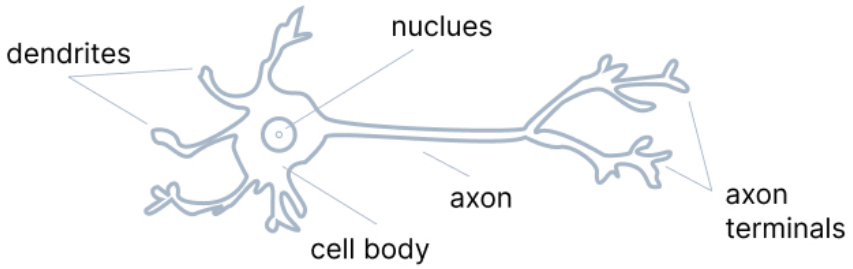
Devang Saraogi
Teaching Assistant

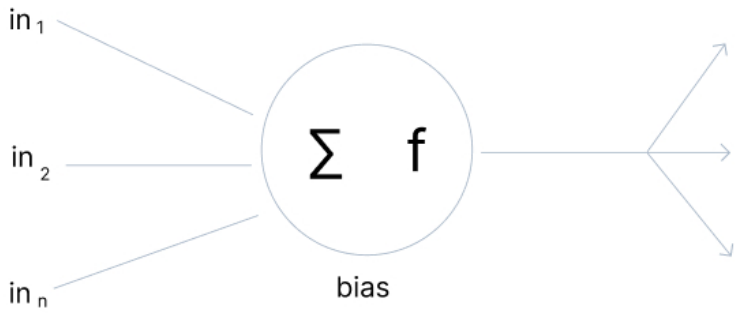
Activation Function

As we know, everything from the name to the structure of a Neural Network is inspired by the human brain; mimicking the way biological neurons signal one another.

Diving into the structure, a node is a replica of a neuron that receives a set of input signals. Depending on the nature and intensity of these input signals, the brain processes them and decides whether the neuron should be **activated ("fired")** or not.

Similarly, in deep learning, this is the role of the Activation Function—that's why it's often referred to as a **Transfer Function** in Artificial Neural Network.





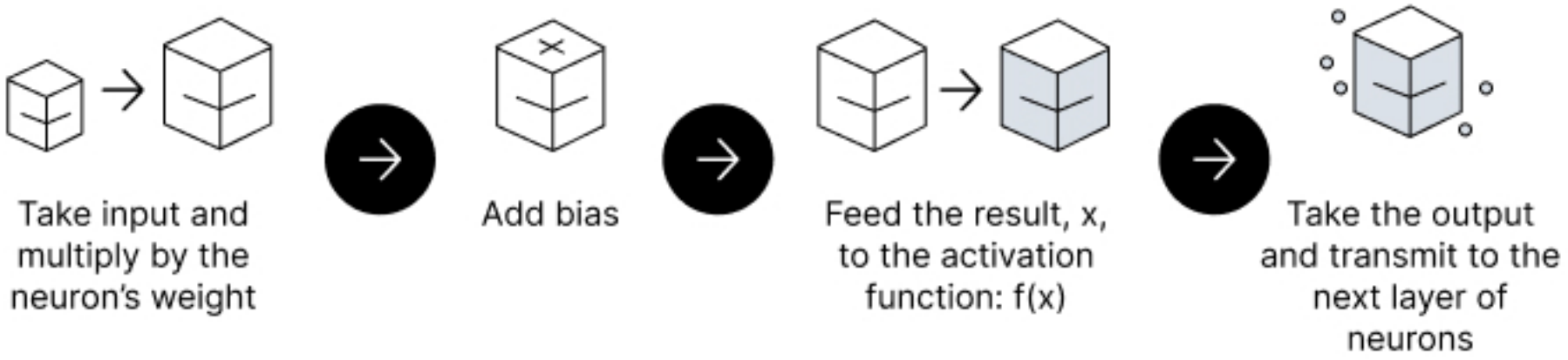
Activation Function

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.

- The activation function introduces **non-linearity** into the output of a neuron.*
- Activation functions also help to normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

* A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Activation Function



Activation Function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer

Types of Activation Function

Typically the type of problems we consider are either classification or regression problems.

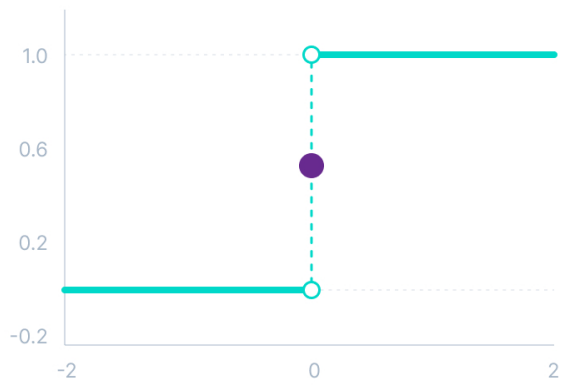
So we need to know

- **What should be the activation function for a regression problem?**
- **What should be the activation function for a classification problem?**

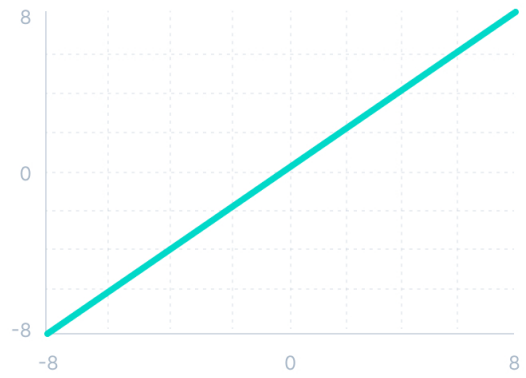
Types of Activation Function

The most popular types of activation functions include…

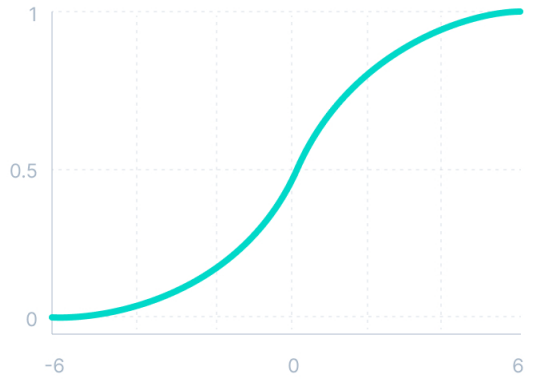
Binary Step Function



Linear Activation



Non-Linear Activation

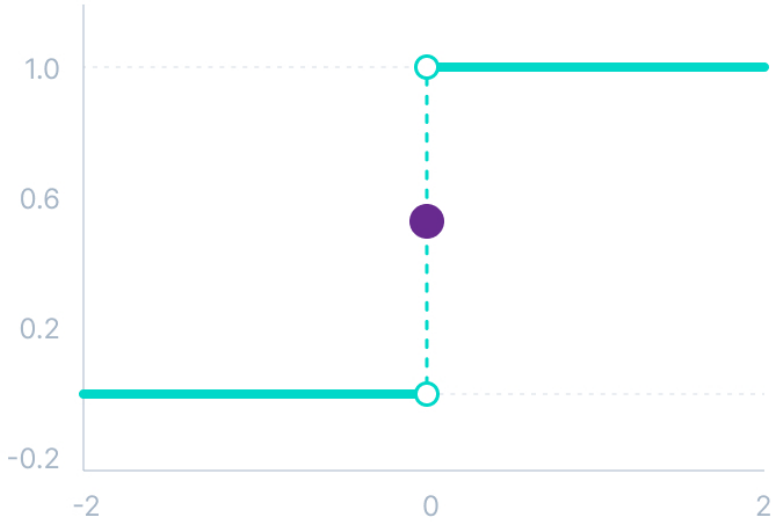


Binary Step Function

Input: weighted sum of weights and biases of the neurons in a layer

Working: the input is compared to a certain threshold; if the input exceeds the threshold than the neuron is activated, else it is deactivated

$$f(x) = \begin{cases} 0 \ for x < 0 \\ 1 \ for x \geq 0 \end{cases}$$



Output: a value of 0 or 1

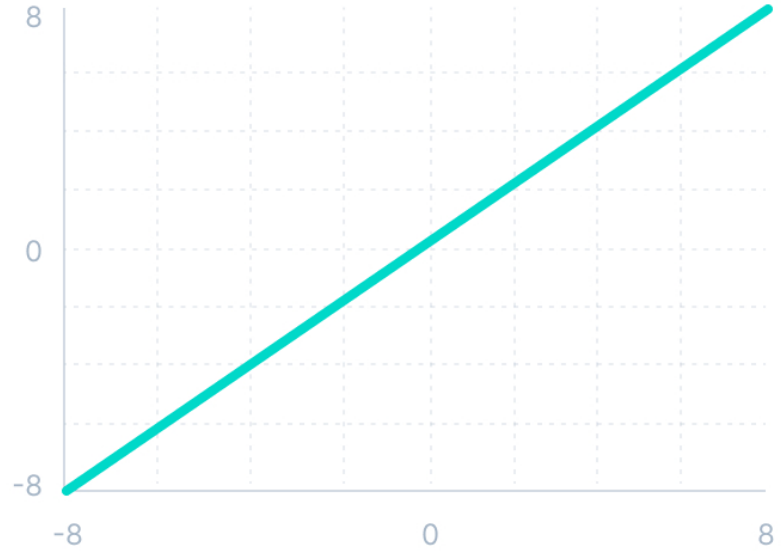
Disadvantage: the step function does not allow multi-value outputs—for example, it cannot be used for multi-class classification problems

Linear Activation Function

Input: weighted sum of weights and biases of the neurons in a layer

Working: the function doesn't do anything to the weighted sum of the input, it simply spits out the value it was given

$$f(x) = x$$



Output: outputs multiple values not just 1 or 0

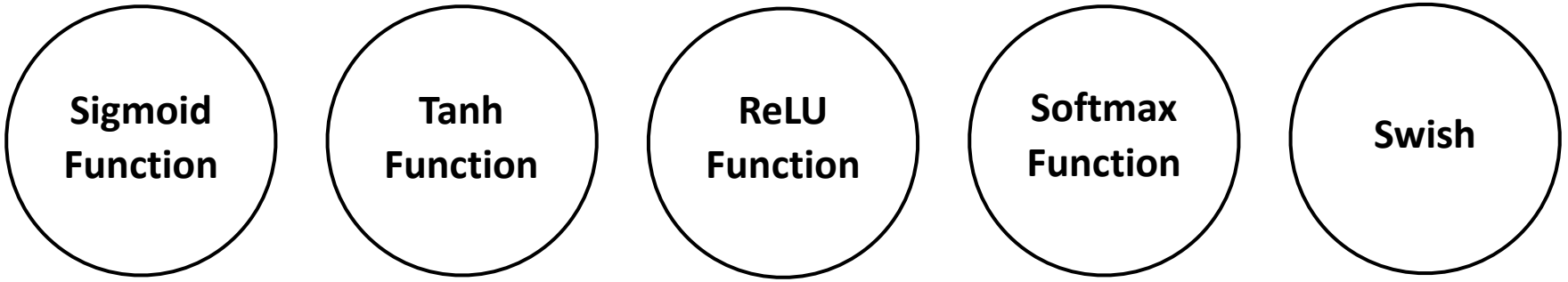
Disadvantage: cannot map non linear data, essentially turning neural networks into single layer of neurons

Types of Activation Function – Non-Linear Activation Function

Linear Activation Functions turn the neural network into a **linear regression** model. This does not allow the model to create complex mappings between the inputs and outputs.

Non-linear Activation Functions solve the following limitations of Linear Activation Functions:

- they allow backpropagation; the derivative function is 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 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.



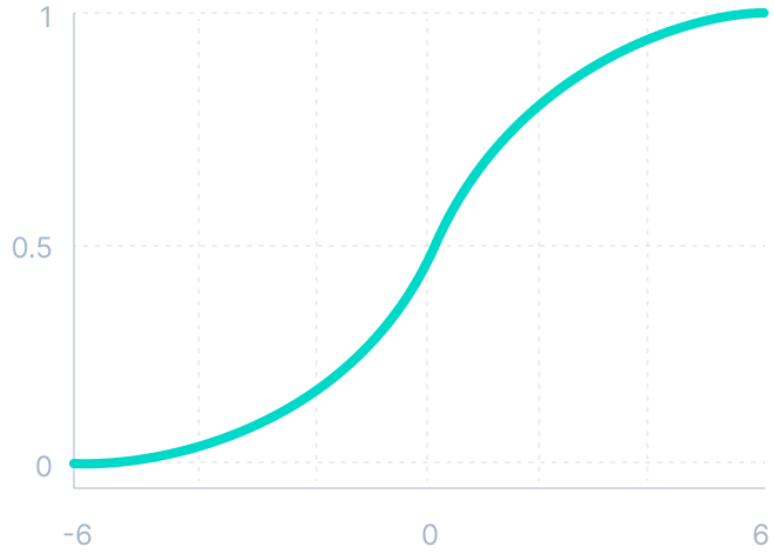
Non-Linear Activation Function

Sigmoid Activation Function (Logistic)

Input: weighted sum of weights and biases of the neurons in a layer

Working: the larger the input the closer the output is to 1 whereas smaller the input closer the output is to 0

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



Output: values between 0 and 1

Advantages:

- suitable for problems involving probabilities
- function is differentiable; smooth gradient

Non-Linear Activation Function

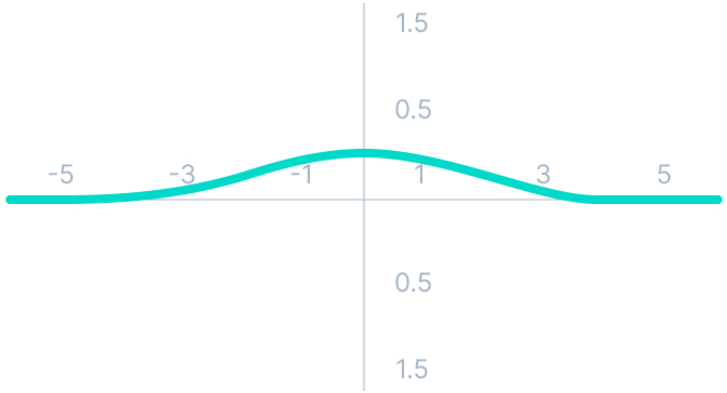
Sigmoid Activation Function

Disadvantages:

derivative of sigmoid function is

$$f'(x) = g(x) = sigmoid(x) \times \left(1 - sigmoid(x)\right)$$

as we can see in the figure, the gradient values are only significant in the central region, the curve gets flatter in the other regions, this implies that the values will change very slowly in those regions



- as gradient values approach zero, the network stops learning and suffers from the **Vanishing Gradient** problem
- the output of the function is **not centered around zero** i.e. the outputs of all neurons will be of the same sign; this makes training the neural network hard

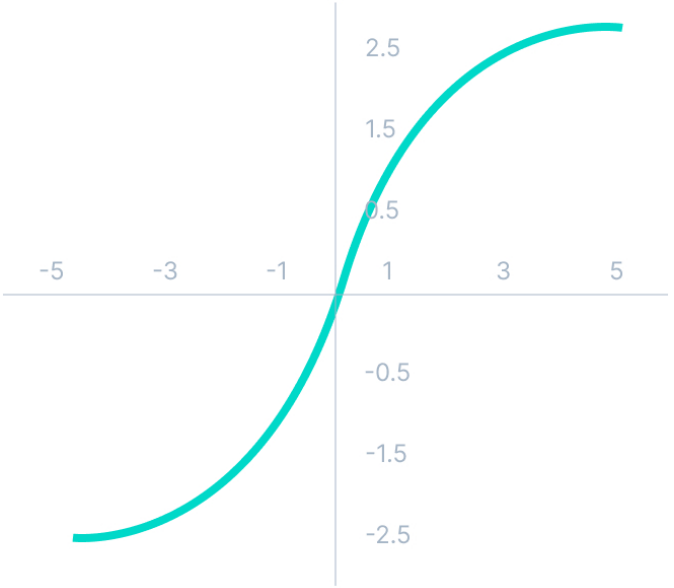
Non-Linear Activation Function

Tanh Function (Hyperbolic Tangent)

Input: weighted sum of weights and biases of the neurons in a layer

Working: the larger the input the closer the output is to 1 whereas smaller the input closer the output is to -1

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



Output: values bounded between -1 and 1

Advantages:

- **Zero Centered**; easier to model inputs that have strongly negative, neutral, and strongly positive values

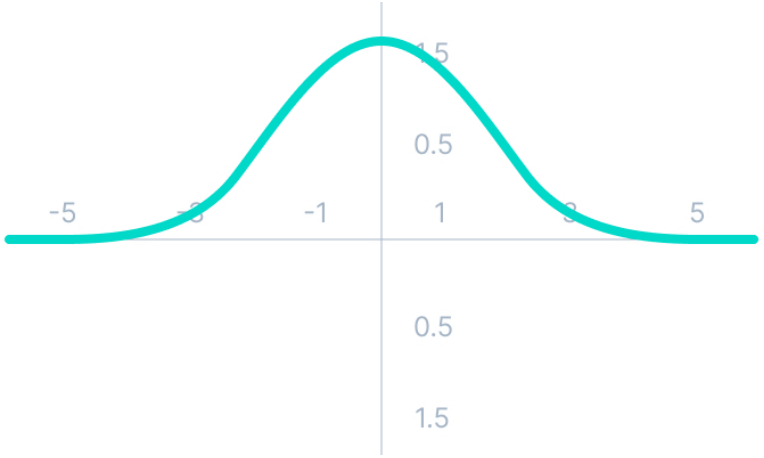
Non-Linear Activation Function

Tanh Function

Disadvantages:

derivative of tanh function is

$$f'(x) = g(x) = 1 - \tanh^2(x)$$



as we can see— it also faces the problem of vanishing gradients similar to the sigmoid activation function

- the gradient of the tanh function is much steeper as compared to the sigmoid function

* although both sigmoid and tanh face the vanishing gradient issue, tanh is zero centered, and the gradients are not restricted to move in a certain direction, therefore, in practice, tanh nonlinearity is always preferred to sigmoid nonlinearity

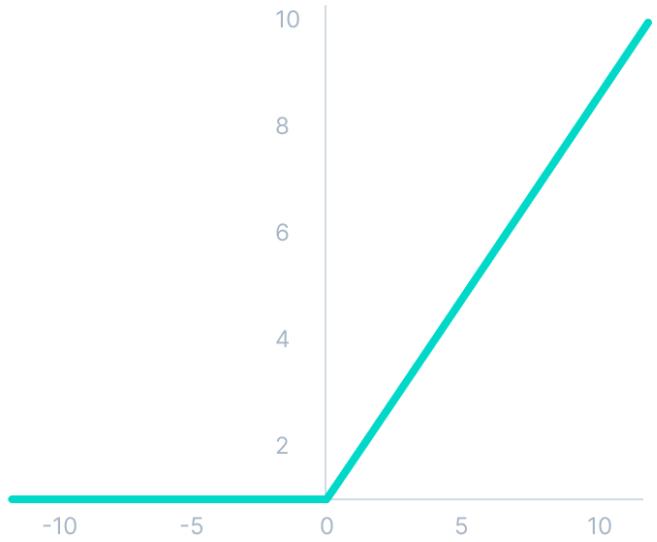
Non-Linear Activation Function

ReLU Activation (Rectified Linear Unit)

Input: weighted sum of weights and biases of the neurons in a layer

Working: he function outputs a 0 if it receives a negative input, but for positive values it returns the same values (like a Linear Activation Function)

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



Output: max of 0 and input

Advantages:

- solves the gradient descent problem
- computationally efficient because of sparse network i.e. **only those neurons are activated which have a non-zero value**

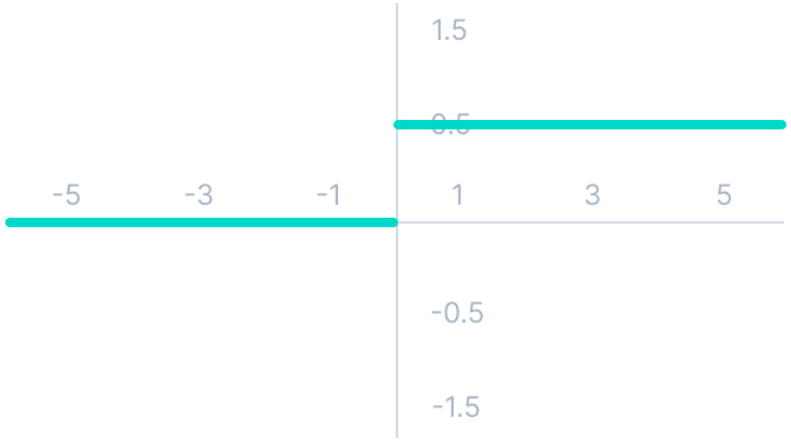
Non-Linear Activation Function

ReLU Activation

Disadvantages: Dying ReLU Problem

derivative of the function is

$$f'(x) = g(x) = \begin{cases} 0 \ for x < 0 \\ 1 \ for x \geq 0 \end{cases}$$



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

Non-Linear Activation Function

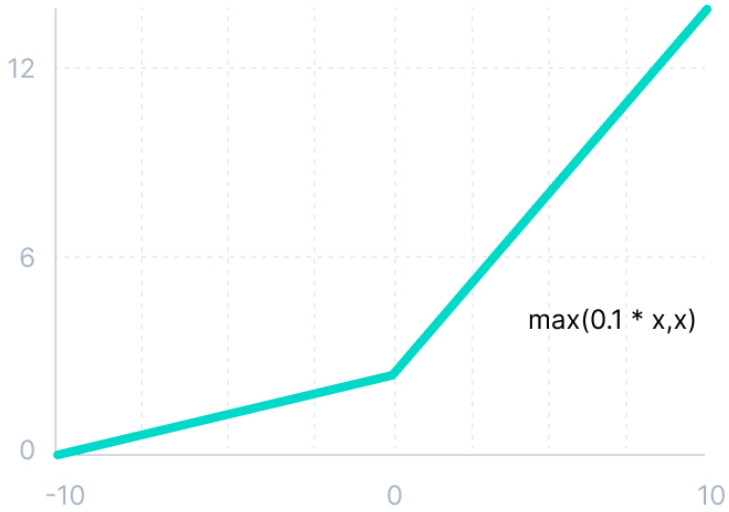
Leaky ReLU Function

Leaky ReLU is an improved version of ReLU, allowing a fixed small positive slope for negative inputs.

Input: weighted sum of weights and biases of the neurons in a layer

Working: the function has a small positive slope in the negative region producing non-zero values for negative inputs; the function performs similarly in positive regions

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



Output: max of **0.1*input** and input

Advantages:

- same as ReLU
- does not suffer from Dying ReLU

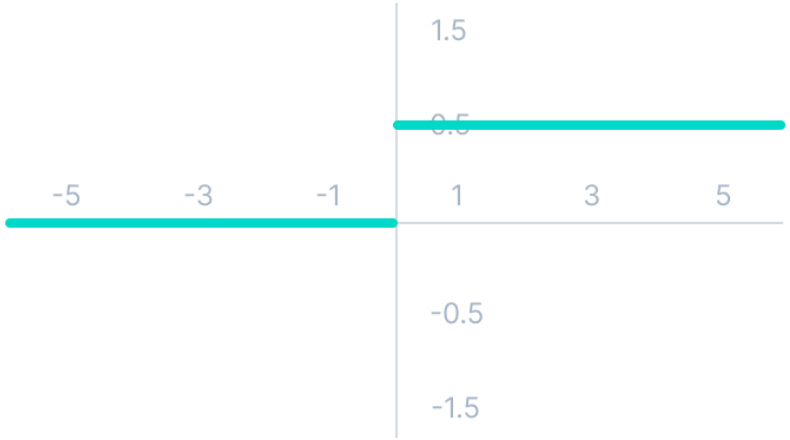
Non-Linear Activation Function

Leaky ReLU Function

How does Leaky ReLU solve the Dying ReLU problem?

it is an improved version of ReLU, a small positive slope is introduced in the negative region **enabling backpropagation**

$$f'(x) = g(x) = \begin{cases} 0.1 & for x < 0 \\ 1 & for x \geq 0 \end{cases}$$



making this minor alteration for negative inputs, the gradient of the negative region results in non-zero value; hence **reducing the occurrences of dead neurons**

Disadvantages: learning process in time consuming and predictions are not consistent for negative values

Non-Linear Activation Function

Parametric ReLU Function (PReLU)

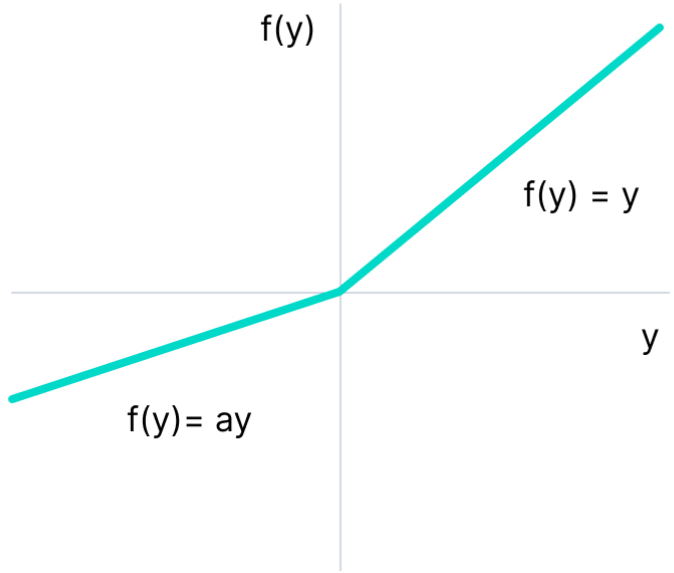
Parametric ReLU is another variant of ReLU that solves the problem of gradient's becoming zero for negative values

Input: weighted sum of weights and biases of the neurons; learnable parameter a

Working: a learnable parameter a represents the slope in the negative region; during training, this parameter is adjusted through backpropagation to optimize the overall performance of the model

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

Output: max of a***input** and input



PES UNIVERSITY

Non-Linear Activation Function

Parametric ReLU Function

Advantages:

- adaptive negative slope
- does not suffer from Dying ReLU

Disadvantages:

- limitation is that it may perform differently for different problems depending upon the value of slope parameter α
- there is a risk of overfitting

Non-Linear Activation Function

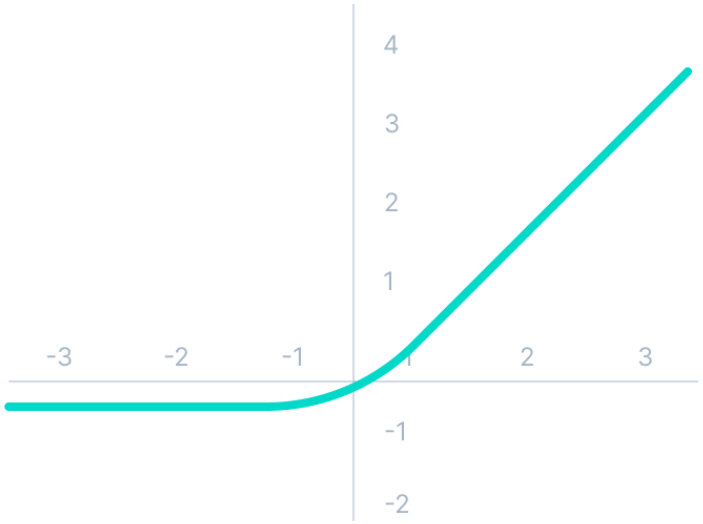
Exponential Linear Units (ELUs)

ELU is also a variant of ReLU that modifies the slope of the negative region of the function

Input: weighted sum of weights and biases of the neurons

Working: ELU uses a logarithmic curve to define the negative values unlike the leaky ReLU and Parametric ReLU functions with a straight line.

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



Advantages:

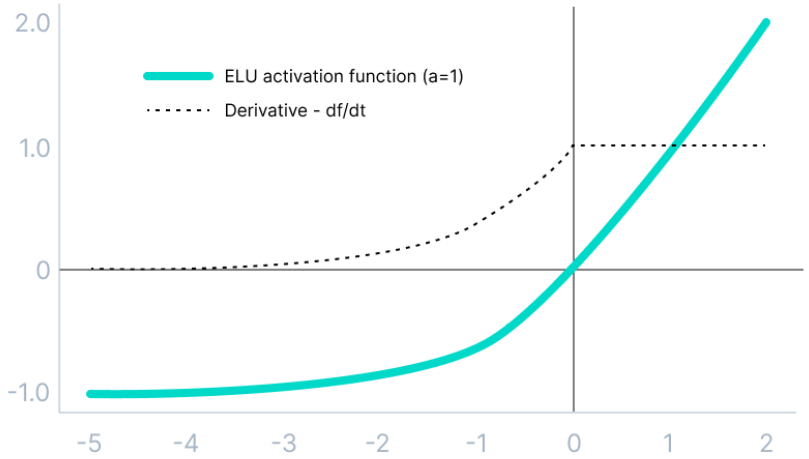
- ELU becomes smooth slowly until its output to $-\alpha$
- avoids dead ReLU problem by introducing log curve for negative values; it helps the network nudge weights and biases in the right direction

Non-Linear Activation Function

Exponential Linear Units

derivative of the activation function

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



Disadvantages:

- increases the computational time because of the exponential operation
- (α) is not a learnable parameter
- exploding gradient problem

Non-Linear Activation Function

Gaussian Error Linear Unit (GELU)

The Gaussian Error Linear Unit (GELU) 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

Input: weighted sum of weights and biases of the neurons

$$f(x) = xP(X \leq x) = x\phi(x)$$
$$= 0.5x\left(1 + \tanh\left[\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right]\right)$$

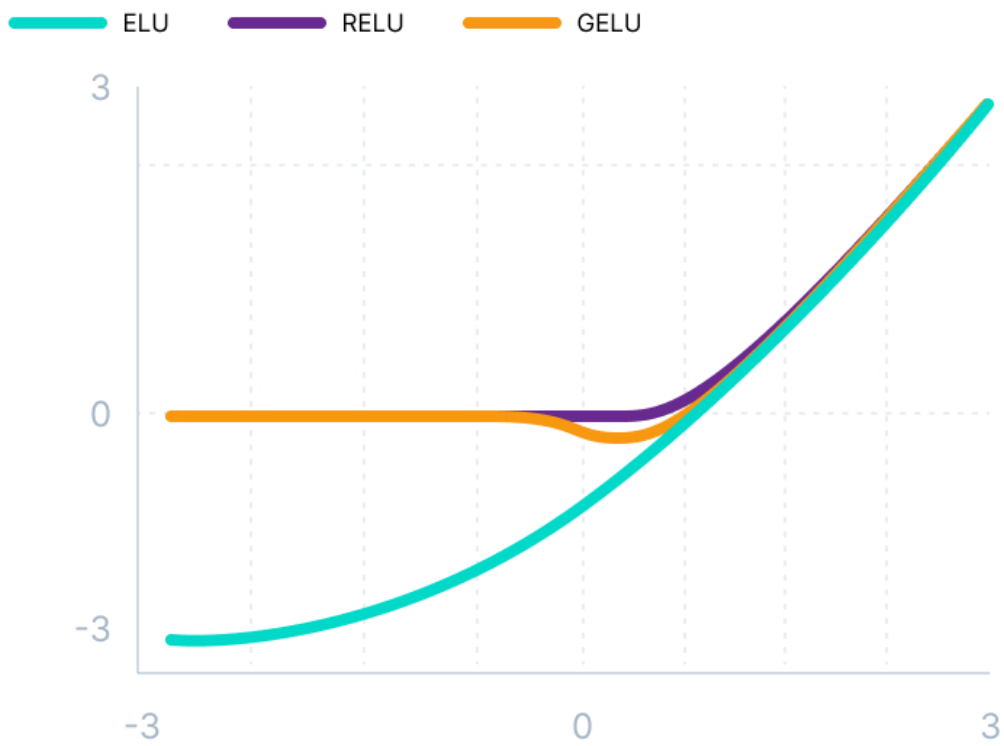
Working: activation functions (ReLU) activate a neuron by multiplying with 0's or 1's; dropout (a regularization technique) randomly drops neurons by multiplying activations with 0; a new RNN regularizer called *Zoneout* stochastically multiples the inputs by 1

This together decides the neuron's output yet they work independently; GELU aims to combine them

Non-Linear Activation Function

Gaussian Error Linear Unit (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



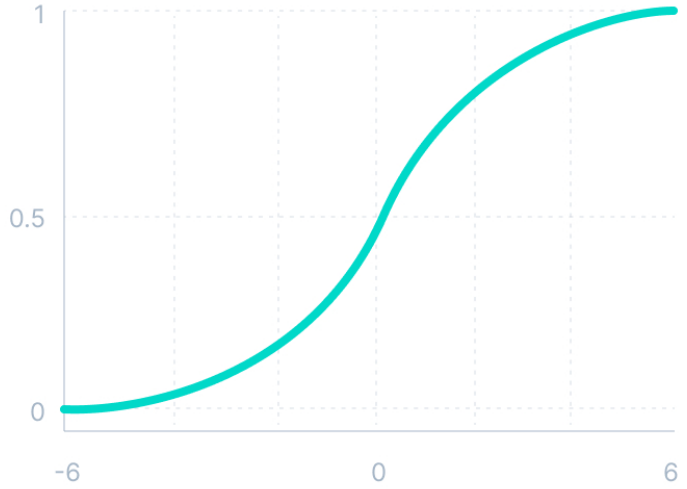
Non-Linear Activation Function

Softmax Activation

Input: weighted sum of weights and biases of the neurons in a layer

Working: softmax function is defined as the combination of multiple sigmoid functions, it calculates the relative probabilities

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$



Output: maps the output to a [0,1] range and at the same time makes sure the summation of the output is 1.

The output of Softmax is therefore a probability distribution.

Non-Linear Activation Function

Swish Function

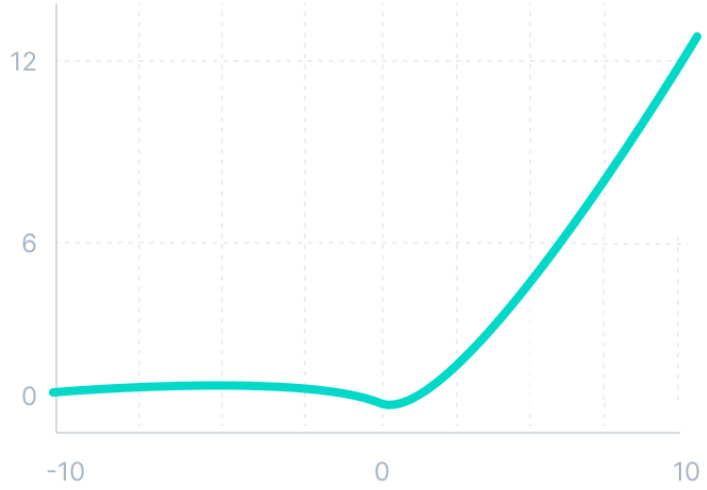
The Swish activation performs well in various deep learning tasks. It introduces a non-linearity that allows the model to capture complex patterns while maintaining some of the desirable properties of ReLU.

Input: weighted sum of weights and biases of the neurons in a layer,

β hyperparameter to control the slope of the swish activation

Working: the function self gates its output i.e. the magnitude of the output is influenced by the input; the sigmoid function acts as a gating mechanism

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



Output: the function approaches positive infinity as the input approaches positive infinity; as the input approaches negative infinity the function settles down to a constant value (~0.5)

Non-Linear Activation Function

Swish Function

Advantages:

- smooth function, does not change value direction abruptly like ReLU; bends smoothly from 0 towards values < 0 and then upwards again
- lays emphasis on small negative values but zeroes out large negative values (win-win)
- non-monotonous nature enhances the expression of input data and weights to be learnt

Disadvantages:

- beneficial in some cases, its superiority over other activation function is not universally established
- finding optimal value for β may require additional experimentation

Non-Linear Activation Function

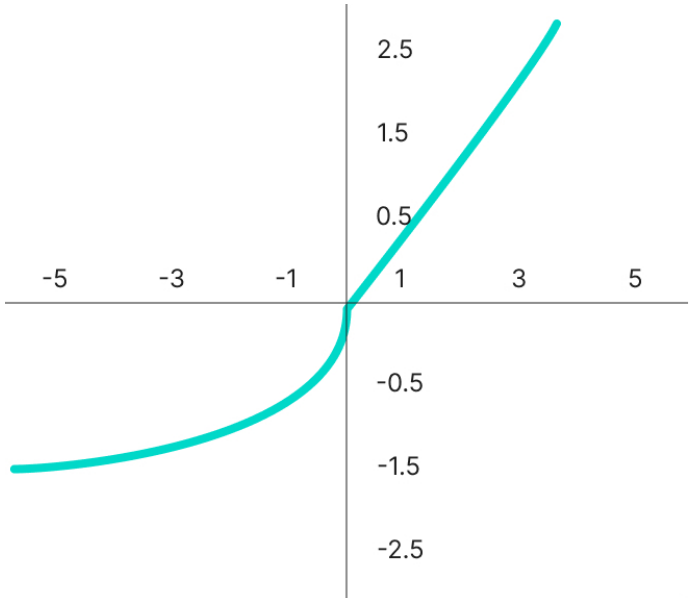
Scaled Exponential Linear Unit (SELU)

SELU is a **self-normalizing activation function**. It is a variant of the ELU . SELU's self-normalizing behavior makes sure that the output is always in a **standardized** range.

Input: weighted sum of weights and biases of the neurons in a layer, scaling factor λ and scaling factor α *

Working: scaling factor λ is applied to the positive region of the function to ensure the standard deviation of the output is close to 1 and scaling factor α is applied to the negative region contributing to the non-linearity of the function.

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



* λ (~ 1.505) and α (~ 1.673) are predefined. These values are based on mathematical considerations to encourage the self normalization. When set to these values, SELU helps the network to stabilize

Non-Linear Activation Function

Scaled Exponential Linear Unit

Advantages:

- like ReLU, SELU does not have vanishing gradient problem
- compared to ReLU, SELU cannot die
- SELUs learn faster and better than other activation functions without needing further procession; moreover, other activation function combined with batch normalization cannot compete with SELUs

Disadvantages:

- SELU is relatively new so it is not yet used widely in practice; ReLU stays as the preferred option
- more research on architectures such as CNNs and RNNs using SELUs is needed for wide-spread industry use

Choice of Activation Function

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

- if we encounter a case of dead neurons in our networks variants of ReLU activation functions seem to be the best choice
- ReLU should only be used in hidden layers and sigmoid/tanh should not be used in hidden layers because they make the model more susceptible to problems (vanishing gradiant)
- sigmoids/tanh functions are generally avoided due to the vanishing gradient problem
- swish function is used in neural networks having a depth greater than 40 layers

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

Choice of Activation Function

Depending upon the type of the prediction problem it is advised to choose these activation functions for the **output layer**

- **Regression** – Linear Activation Function
- **Binary Classification** – Sigmoid Activation Function
- **Multiclass Classification** – Softmax Activation
- **Multilabel Classification** – Sigmoid Activation Function

PES UNIVERSITY

Recap

- Activation Functions are used to introduce **non-linearity** in the network.
- A neural network will almost always have the **same activation function in all hidden layers**. This activation function should be **differentiable** so that the parameters of the network are learned in backpropagation.
- While selecting an activation function, you must consider the problems it might face: vanishing and exploding gradients.
- Use Softmax or Sigmoid function for the classification problems.

PES UNIVERSITY

Further Reading

In addition to the popular activation functions mentioned earlier, such as the Sigmoid and ReLU functions, there are many other activation functions that can be used in neural networks based on the specific problem statement and desired optimization of the learning process.

To gain a deeper understanding and familiarize yourself with different activation functions, you can refer to the following link:

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

PES UNIVERSITY

References

- https://www.v7labs.com/blog/neural-networks-activation-functions
- https://pub.towardsai.net/what-is-parametric-relu-2444a2a292de
- https://machinelearningmastery.com/activation-functions-in-pytorch/
- https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
- https://medium.com/@vinodhb95/activation-functions-and-its-types-8750f1287464
- https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e
- https://pytorch.org/docs/stable/nn.html#non-linear-activations-other

UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack: Devan Saragoi, Teaching Assistant