Analytics Vidhya

Types of Activation Functions in Machine Learning

Function Type	*Equation*	*Derivative*
Linear	$f(x) = ax + c$	$f'(x) = a$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)\,(1 - f(x))$
TanH	$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
Parametric ReLU	$f(x) = \begin{cases} \alpha x \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
ELU	$f(x) = \begin{cases} \alpha(e^x - 1) \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$

In machine learning and, more specifically, in neural networks, an activation function is a mathematical function that determines the output of a neuron or node. It adds non-linearity to the network, allowing it to learn from error and make adjustments, which is essential for learning complex patterns.

Here are some of the most commonly used activation functions:

Linear Activation Function

- Equation: f(x)=x

- It simply returns the input as it is. It doesn't make the output non-linear, and therefore, is usually used in single-layer networks or for the output layer in regression problems.

```python
import numpy as np

def linear(x):
    return x
```

Sigmoid Activation Function

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

- It squashes the output between 0 and 1. Historically popular, but less used today due to problems like vanishing gradients in deep networks.

```python
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

ReLU Activation Function

- Equation: f(x)=max(0,x)
- ReLU stands for Rectified Linear Unit
- It replaces all negative values with zero. This function is computationally efficient and has become one of the default activation functions for many types of neural networks.

```
def relu(x):
    return np.maximum(0, x)
```

Leaky ReLU Activation Function

- Equation: $f(x)=x$ (for $x>0x>0$), $f(x)=\alpha x$ (for $x\leq 0x\leq 0$) where α is a small constant.
- Leaky ReLU stands for Leaky Rectified Linear Unit
- It's a variation of ReLU to fix the dying ReLU problem where neurons can sometimes get stuck during training and cease updating.

```
def leaky_relu(x, alpha=0.01):
    return np.where(x > 0, x, alpha * x)
```

Softmax Activation Function

- Used in the output layer of a classification problem with more than two classes. It converts the network's output scores into probabilities for each class.

```
def softmax(x):
    exps = np.exp(x - np.max(x))
    return exps / exps.sum(axis=0, keepdims=True)
```