Id:20-42834-1

Name: Eva, Arifa AKTER

Sec:A

Topic: Implementation on different activation function

Activation function

An Activation Function decides whether a neuron should be activated or not. This means that it will decide whether the neuron's input to the network is important or not in the process of prediction using simpler mathematical operations.It is a non-linear mathematical function used in artificial neural networks to introduce non-linearity into the output of a neural network node. The purpose of an activation function is to introduce non-linearities to the output of the neural network, allowing it to learn complex patterns in the data.

1. **Step function**
   A step function is a function like that used by the original Perceptron. The output is a certain value, A1, if the input sum is above a certain threshold and A0 if the input sum is below a certain threshold. The values used by the Perceptron were A1 = 1 and A0 = 0.

$$f(x) = \begin{cases} 1 & \text{if } x > \theta \\ 0 & \text{if } x \leq \theta \end{cases}$$

   This equation outputs a value of 1.0 for incoming values of (threshold value) or higher and 0 for all other values. Step functions, also known as threshold functions, only return 1 (true) for values that are above the specified threshold.

   Advantage:
   - computationally efficient and simple
   - Can be used as a binary classifier

   Disadvantage:
   - It is not appropriate for use in backpropagation-based learning algorithms because it is not differentiable at x = 0.
   - Gradient updates are a challenge to improve because they are constant and independent of the input.

2. **Sigmoid function:**

The sigmoid activation function is a mathematical function commonly used in artificial neural networks to introduce non-linearity to the output of a node. It is a smooth, S-shaped function that maps any input value to a value between 0 and 1.

The sigmoid function has a few important properties that make it useful for neural networks. First, it is a continuous function, which means that it has a derivative at every point. This makes it possible to use gradient-based optimization techniques to train the neural network. Second, it is a non-linear function, which means that it can introduce non-linearities into the output of the neural network, allowing it to learn complex patterns in the data.

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

Advantage:

- Well-suited for use in shallow neural networks with few hidden layers.
- Smooth and different.
- It can be applied to probabilistic modeling, which makes it helpful for binary classification issues.

Disadvantage

- Outputs are not zero-cantered, which can slow down convergence during training.
- vulnerable to the vanishing gradient issue, which makes deep learning challenging.

3. Tanh function

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.The hyperbolic tangent (tanh) activation function is a popular mathematical function used in artificial neural networks to introduce non-linearity to the output of a node.

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

Like the sigmoid neuron, its activations saturate, but unlike the sigmoid neuron its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity. Also note that the tanh neuron is simply a scaled sigmoid neuron, in particular the following holds.

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

Advantage:

- Smooth and unique
- Compared to the sigmoid function, the outputs are zero-centered, making it more useful for teaching deep neural networks.
- Well-suited for use in shallow and deep neural networks

Disadvantage:

- vulnerable to the disappearing gradient issue, which makes training very deep neural networks challenging

4. **ReLu function (Rectified Linear Units)**

The rectified linear unit (ReLU) activation function is a popular mathematical function used in artificial neural networks to introduce non-linearity to the output of a node. It is a simple function that returns the input value if it is positive, and returns zero otherwise.

The following equation shows the straightforward ReLU function:

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

In other words, the activation is simply thresholded at zero:

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

Advantage:

- It is a simple function that is computationally efficient to compute.
- It does not suffer from the problem of vanishing gradients for large input values, making it easier to train deep neural networks.
- It can help to sparsity the activation of the network, by setting some of the activations to zero, which can help to prevent overfitting.

Disadvantage:

- Not differentiable at x = 0, which can cause issues during backpropagation.
- vulnerable to the "dying ReLU" problem, in which a significant portion of the network can cease responding and learning.

5. PRELU Function

The parametric rectified linear unit (PReLU) activation function is a variant of the ReLU function, which addresses the problem of "dying ReLU" by allowing for non-zero output for negative input values.

Advantage:

- it can help to prevent the problem of dying ReLU by allowing for non-zero output for negative input values.
- it can help to improve the accuracy of the network, especially for datasets with a high level of noise or variability.
- it is a simple function that is computationally efficient to compute.

Disadvantage:

- Higher processing cost compared to the ReLU function
- May not always improve performance over the ReLU function

6. EReLU Function

The Exponential Rectified Linear Unit (eReLU) activation function is a modified version of the popular ReLU activation function used in artificial neural networks. The eReLU function is designed to address the "dying ReLU" problem, which occurs when ReLU neurons with negative inputs get "stuck" at 0 and stop learning.

Advantage:

- By stopping the gradient from turning zero or negative for negative inputs, it solves the "dying ReLU" issue and speeds up training.
- Outputs are zero-centered, which can enhance learning efficiency
- Smoothness and differentiability support optimization and gradient-based learning

Disadvantage:

- computationally more costly compared to the ReLU function and other variants like the PReLU function
- Tuning extra hyper parameters (alpha)
- There may be less community support and fewer resources accessible for implementation and optimization because it is not as popular or widely used as other activation functions.