

| **Title: Activation functions.** |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective:** To implement activation functions of Neural Network.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

CO1 : Identify and describe soft computing techniques and their roles **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

* J.S.R.Jang, C.T.Sun and E.Mizutani, “Neuro-Fuzzy and Soft Computing”, PHI, 2004, Pearson Education 2004.
* Davis E.Goldberg, “Genetic Algorithms: Search, Optimization and Machine Learning”, Addison Wesley, N.Y., 1989.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Pre Lab/ Prior Concepts:**

Neural networks, sometimes referred to as connectionist models, are parallel-distributed models that have several distinguishing features-

1)      A set of processing units;

2)      An activation state for each unit, which is equivalent to the output of the unit;

3)      Connections between the units. Generally each connection is defined by a weight *wjk* that

determines the effect that the signal of unit *j* has on unit *k;*

4)      A propagation rule, which determines the effective input of the unit from its external inputs;

5)      An activation function, which determines the new level of activation based on the effective

input and the current activation;

6)      An external input (bias, offset) for each unit;

7)      A method for information gathering (learning rule);

8)      An environment within which the system can operate, provide input signals and, if necessary, error signals.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Implementation Details:**

Most units in neural network transform their net inputs by using a scalar-to-scalar function called an *activation function*, yielding a value called the unit's activation. Except possibly for output units, the activation value is fed to one or more other units. Activation functions with a bounded range are often called squashing functions. Some of the most commonly used activation functions are :

1. **Identity function**
2. **Binary step function :**
3. **Sigmoid function:**
4. **Bipolar sigmoid function**: this function is defined as follows:
5. **Ramp function**: this function is defined as follows:
6. **ReLU function**: this function is defined as follows

**Code:**

1. Identity function

import numpy as np # type: ignore

import matplotlib.pyplot as plt # type: ignore

from matplotlib import style # type: ignore

style.use('ggplot')

def identity (x):

return x

x = np.linspace(-5, 5, 1000)

y\_identity = identity(x)

plt.figure(figsize=(12, 8))

plt.subplot(3, 2, 1)

plt.plot(x, y\_identity, label='Identity')

plt.title('Identity Function')

plt.legend()

1. Binary step function

import numpy as np # type: ignore

import matplotlib.pyplot as plt # type: ignore

from matplotlib import style # type: ignore

style.use('ggplot')

def binary\_step(x):

return np.where(x >= 0, 1, 0)

x = np.linspace(-5, 5, 1000)

y\_binary\_step = binary\_step(x)

plt.figure(figsize=(12, 8))

plt.subplot(3, 2, 2)

plt.plot(x, y\_binary\_step, label='Binary Step')

plt.title('Binary Step Function')

plt.legend()

1. Sigmoid function

import numpy as np # type: ignore

import matplotlib.pyplot as plt # type: ignore

from matplotlib import style # type: ignore

style.use('ggplot')

def sigmoid(x):

return 1 / (1 + np.exp(-x))

x = np.linspace(-5, 5, 1000)

y\_sigmoid = sigmoid(x)

plt.figure(figsize=(12, 8))

plt.subplot(3, 2, 3)

plt.plot(x, y\_sigmoid, label='Sigmoid')

plt.title('Sigmoid Function')

plt.legend()

1. Bipolar sigmoid function

import numpy as np # type: ignore

import matplotlib.pyplot as plt # type: ignore

from matplotlib import style # type: ignore

style.use('ggplot')

def bipolar\_sigmoid(x):

return (1 - np.exp(-x)) / (1 + np.exp(-x))

x = np.linspace(-5, 5, 1000)

y\_bipolar\_sigmoid = bipolar\_sigmoid(x)

plt.figure(figsize=(12, 8))

plt.subplot(3, 2, 4)

plt.plot(x, y\_bipolar\_sigmoid, label='Bipolar Sigmoid')

plt.title('Bipolar Sigmoid Function')

plt.legend()

1. Ramp function

import numpy as np # type: ignore

import matplotlib.pyplot as plt # type: ignore

from matplotlib import style # type: ignore

style.use('ggplot')

def ramp(x):

return np.piecewise(x, [x < 0, (x >= 0) & (x <= 1), x > 1], [0, lambda x: x, 1])

x = np.linspace(-5, 5, 1000)

y\_ramp = ramp(x)

plt.figure(figsize=(12, 8))

plt.subplot(3, 2, 5)

plt.plot(x, y\_ramp, label='Ramp')

plt.title('Ramp Function')

plt.legend()

1. ReLU function

import numpy as np # type: ignore

import matplotlib.pyplot as plt # type: ignore

from matplotlib import style # type: ignore

style.use('ggplot')

def relu(x):

return np.where(x > 0, x, 0)

x = np.linspace(-5, 5, 1000)

y\_relu = relu(x)

plt.figure(figsize=(12, 8))

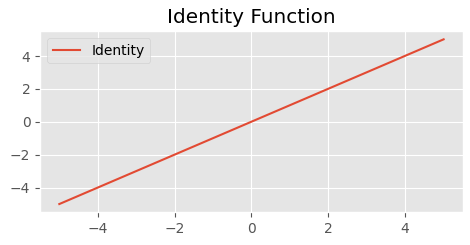
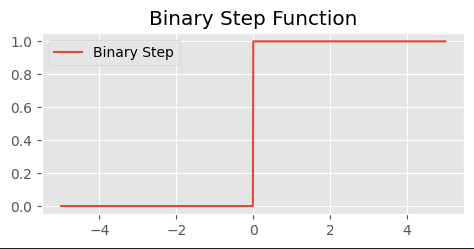
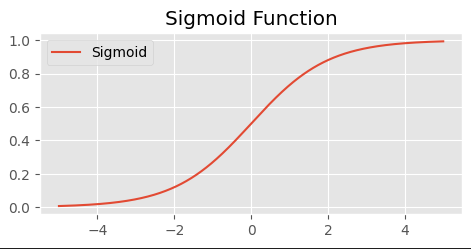
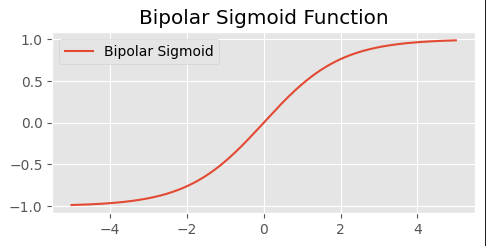
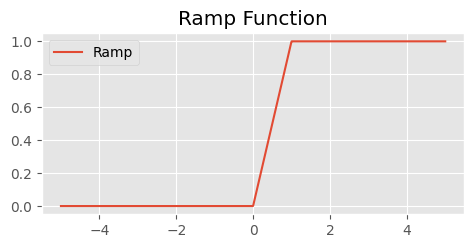
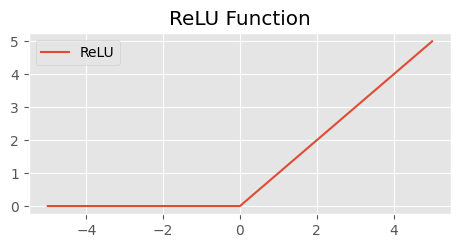
plt.subplot(3, 2, 6)

plt.plot(x, y\_relu, label='ReLU')

plt.title('ReLU Function')

plt.legend()

**Output:**

1. Identity function 
2. Binary step function 
3. Sigmoid Function 
4. Bipolar sigmoid function 
5. Ramp function 
6. ReLU function 

**Conclusion:** Thus, we have successfully implemented 6 Activation Functions of Neural Network.

**Post Lab Descriptive Questions :**

1. Explain the concept behind using Activation function.

Activation functions are a crucial component in artificial neural networks (ANNs), serving as a mathematical operation applied to the output of each neuron. The primary purpose of activation functions is to introduce non-linearity into the network, which enables it to learn and perform complex mappings from input data to output predictions.

Here are the key reasons why activation functions are used:

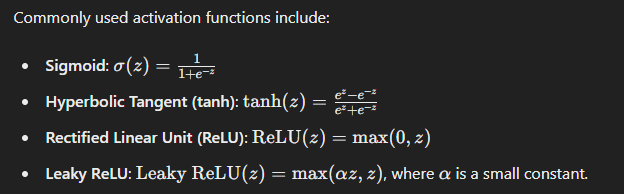
1. Introducing Non-linearity: Without activation functions, a neural network, no matter how deep, would behave like a single-layer perceptron, capable only of learning linear relationships between inputs and outputs. Non-linear activation functions allow the network to learn and approximate any non-linear function, making it powerful enough to solve complex problems like image and speech recognition.

2. Enabling Gradient Descent: Activation functions enable the backpropagation algorithm to work correctly during training. Backpropagation involves computing gradients of the error with respect to the weights of the network, which are used to update the weights in the direction that reduces the error. Activation functions must have derivatives that are computable and well-defined to propagate these gradients backward through the network layers.

3. Controlling Output Range: Different activation functions have different output ranges, which can be important depending on the requirements of the problem. For example, sigmoid activation functions squash the output to the range \([0, 1]\), which is useful for binary classification tasks. On the other hand, tanh activation functions squash the output to \([-1, 1]\), which can be useful in certain contexts to center the outputs around zero.

4. Handling Sparse Activation: In networks with many layers, activation functions can help control the sparsity of activations (i.e., the percentage of neurons that are activated) by determining which neurons are activated based on the input data. This can help in efficient representation and computation within the network.

Commonly used activation functions include:



Choosing the right activation function depends on the specific problem domain and the characteristics of the data being processed. Overall, activation functions play a critical role in the expressive power and training dynamics of neural networks, enabling them to effectively learn and generalize from complex data.

1. Explain the different properties of activation functions.

Activation functions used in neural networks possess various properties that influence their effectiveness in learning and training. Here are the key properties of activation functions:

1. Non-linearity:

- Definition: A non-linear activation function allows the neural network to approximate complex non-linear relationships between inputs and outputs.

- Importance: Without non-linearity, the entire neural network would behave like a single-layer perceptron, unable to model and learn from data that isn't linearly separable.

- Examples: Sigmoid, tanh, ReLU, Leaky ReLU, and many others are non-linear activation functions widely used in neural networks.

2. Range of Output:

- Definition: Activation functions typically have a defined output range, which can affect how well the neural network performs.

- Importance: The output range should ideally match the requirements of the problem at hand. For example, sigmoid outputs are in the range \([0, 1]\), making them suitable for binary classification tasks, while tanh outputs are in \([-1, 1]\), allowing centered outputs around zero.

- Examples: Sigmoid, tanh, and ReLU have different output ranges which can influence network behavior and training dynamics.

3. Differentiability:

- Definition: Activation functions need to be differentiable (or at least piecewise differentiable) so that gradients can be computed during backpropagation.

- Importance: Backpropagation relies on computing gradients of the loss function with respect to the weights and biases of the network. The derivative of the activation function determines how the error is propagated backward through the network layers.

- Examples: Sigmoid and tanh are differentiable everywhere, while ReLU is not differentiable at (z = 0), but its gradient can be defined using subgradients or approximation techniques.

4. Monotonicity:

- Definition: A monotonic activation function preserves the order of inputs, meaning if (a < b), then (f(a) < f(b)) (or (f(a) <= f(b))).

- Importance: Monotonic activation functions can simplify the optimization process because the loss function decreases consistently with respect to the input values.

- Examples: Sigmoid and tanh are monotonic across their entire range, whereas ReLU is monotonic for (z > 0).

5. Saturation:

- Definition: Saturation refers to how quickly the activation function approaches its maximum or minimum output for large positive or negative inputs, respectively.

- Importance: Activation functions that saturate too early (e.g., sigmoid for very positive or negative inputs) can cause the gradients to become very small (vanishing gradients), leading to slow or ineffective learning.

- Examples: Sigmoid and tanh can saturate for extreme inputs, especially outside their main operating range.

6. Zero-Centered Output (or not):

- Definition: Some activation functions produce outputs that are centered around zero (e.g., tanh), while others are not (e.g., ReLU).

- Importance: Zero-centered outputs can help in optimizing the network because they keep the gradients more balanced during backpropagation.

- Examples: tanh produces outputs in the range ([-1, 1]), centered around zero, which can help in certain optimization scenarios.

Understanding these properties helps in choosing the right activation function based on the characteristics of the data, the task at hand, and the network architecture being used. Each property influences how effectively the neural network can learn and generalize from the data during training.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**