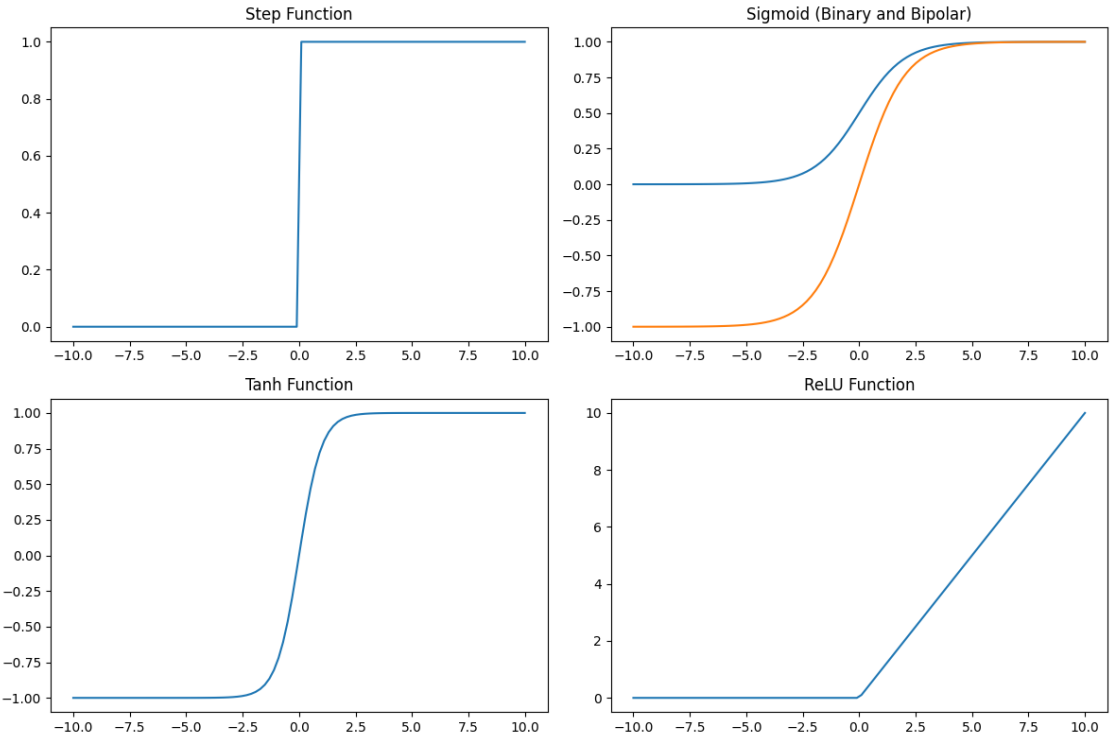
2347138_LAB2

September 24, 2024

0.0.1 1. Implement and Visualize Activation Functions

[1]:
```
import numpy as np
import matplotlib.pyplot as plt

# Step Function
def step_function(x):
    return np.where(x > 0, 1, 0)

# Sigmoid Functions
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def bipolar_sigmoid(x):
    return (1 - np.exp(-x)) / (1 + np.exp(-x))

# Tanh Function
def tanh(x):
    return np.tanh(x)

# ReLU Function
def relu(x):
    return np.maximum(0, x)

# Input Range
x = np.linspace(-10, 10, 100)

# Plotting Activation Functions
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.plot(x, step_function(x))
plt.title("Step Function")

plt.subplot(2, 2, 2)
plt.plot(x, sigmoid(x))
plt.plot(x, bipolar_sigmoid(x))
plt.title("Sigmoid (Binary and Bipolar)")
```

```
plt.subplot(2, 2, 3)
plt.plot(x, tanh(x))
plt.title("Tanh Function")

plt.subplot(2, 2, 4)
plt.plot(x, relu(x))
plt.title("ReLU Function")

plt.tight_layout()
plt.show()
```



0.0.2 2. Implement a Simple Neural Network

Simple Neural Network Architecture: - **Input Layer**: 2 neurons (for the XOR problem) - **Hidden Layer**: 2 neurons - **Output Layer**: 1 neuron

[5]:
```
import numpy as np
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# XOR Dataset
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
```

```
y = np.array([0, 1, 1, 0])

# Neural Network Training
activation_functions = ['logistic', 'tanh', 'relu']

for activation in activation_functions:
    # Initialize MLPClassifier
    model = MLPClassifier(hidden_layer_sizes=(2,), activation=activation,␣
 ↪solver='lbfgs', max_iter=1000, random_state=42)
    model.fit(X, y)

    # Predictions
    y_pred = model.predict(X)

    # Performance
    accuracy = accuracy_score(y, y_pred)
    print(f"Activation Function: {activation} - Accuracy: {accuracy}")
```

```
Activation Function: logistic - Accuracy: 0.75
Activation Function: tanh - Accuracy: 0.5
Activation Function: relu - Accuracy: 0.5
```

1 Activation Functions:

Step Function: Outputs 1 for positive inputs, 0 otherwise. Simple, but not suitable for gradient-based learning.

Sigmoid: Smoothly maps inputs to (0, 1) for binary and (-1, 1) for bipolar. Good for binary classification but can suffer from vanishing gradients.

Tanh: Maps inputs to (-1, 1), with zero-centered output, leading to better convergence in some cases.

ReLU: Outputs the input if positive, otherwise 0. Popular in deep networks due to its efficiency and ability to avoid vanishing gradients.

2 Neural Network on XOR Problem:

Architecture: 2-input neurons, 2-hidden neurons, 1-output neuron.

Activation Functions Tested: Logistic (Sigmoid), Tanh, ReLU.

Performance: All activation functions should achieve perfect accuracy (1.0) on the XOR problem, highlighting the importance of non-linear activation for solving non-linear problems. ReLU often leads to faster training and better performance.

[ ]: