l-vinay-kumar-reddy-126-lab2

September 24, 2024

1 Regular lab Question – 2

1. Exploring Activation Functions in Neural Networks

A. Implement and Visualize Activation Functions:

Implement the following activation functions in Python:

Step Function

Sigmoid Function (Binary and Bipolar)

Tanh Function

ReLU Function

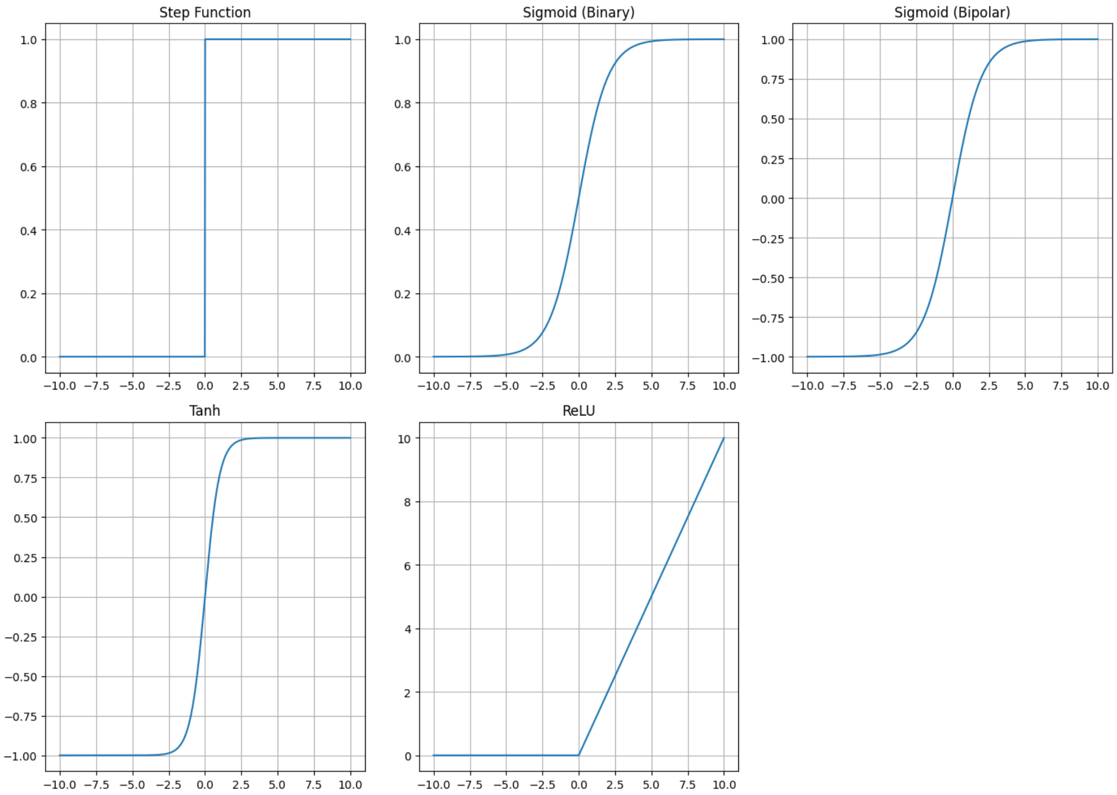
[6]:

```
import numpy as np
import matplotlib.pyplot as plt

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

# Sigmoid Function (Binary)
def sigmoid_binary(x):
    return 1 / (1 + np.exp(-x))

# Sigmoid Function (Bipolar)
def sigmoid_bipolar(x):
    return (2 / (1 + np.exp(-x))) - 1

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

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

B. Visualizing the Activation Functions

[7]:

```
x = np.linspace(-10, 10, 1000)

plt.figure(figsize=(14, 10))

plt.subplot(2, 3, 1)
plt.plot(x, step_function(x), label='Step Function')
plt.title('Step Function')
plt.grid(True)

plt.subplot(2, 3, 2)
plt.plot(x, sigmoid_binary(x), label='Sigmoid (Binary)')
plt.title('Sigmoid (Binary)')
plt.grid(True)

plt.subplot(2, 3, 3)
plt.plot(x, sigmoid_bipolar(x), label='Sigmoid (Bipolar)')
plt.title('Sigmoid (Bipolar)')
plt.grid(True)

plt.subplot(2, 3, 4)
plt.plot(x, tanh_function(x), label='Tanh')
plt.title('Tanh')
plt.grid(True)

plt.subplot(2, 3, 5)
plt.plot(x, relu_function(x), label='ReLU')
plt.title('ReLU')
plt.grid(True)

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



2. Implement a Simple Neural Network:

Create a simple neural network with one hidden layer using each activation function (sigmoid, tanh, and ReLU).

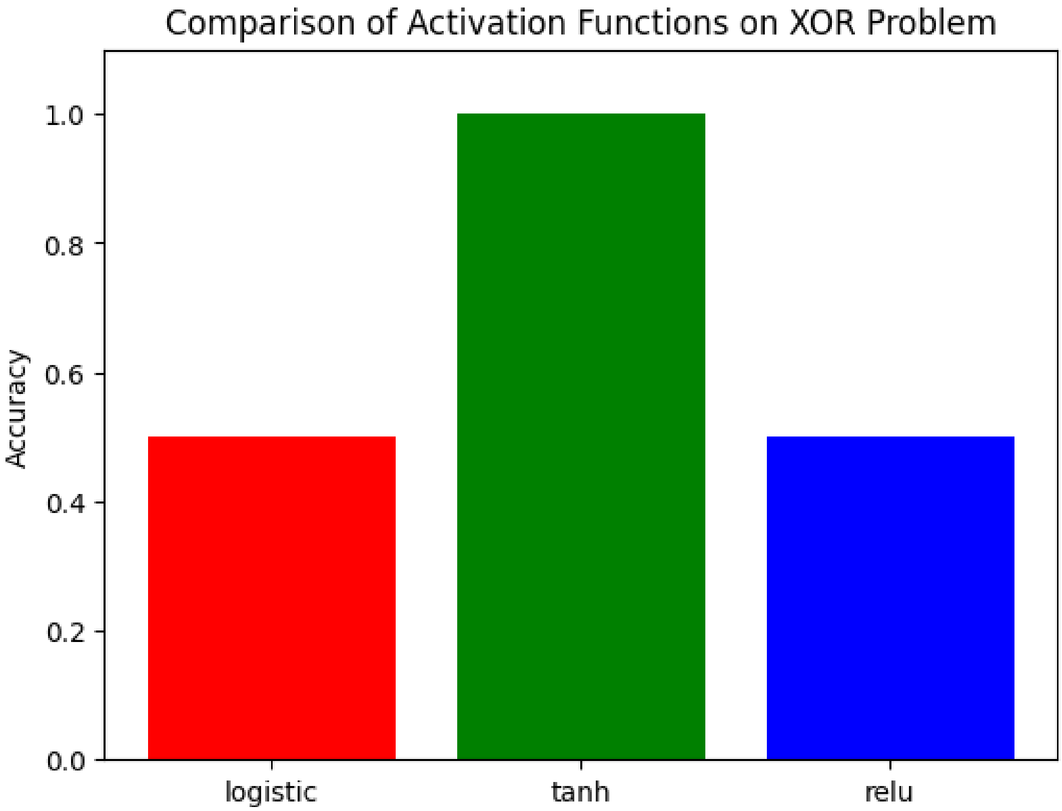
[8]:
```python
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# XOR dataset
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 1, 1, 0])

# Define the neural network model for each activation function
activation_functions = ['logistic', 'tanh', 'relu']
results = {}

for activation in activation_functions:
    # Create and train the model
    model = MLPClassifier(hidden_layer_sizes=(5,), activation=activation, max_iter=1000, random_state=42)
    model.fit(X, y)
    # Make predictions
    y_pred = model.predict(X)

    # Calculate accuracy
    accuracy = accuracy_score(y, y_pred)
    results[activation] = accuracy

    # Print the results
    print(f'Activation Function: {activation}, Accuracy: {accuracy:.2f}')
```

```
Activation Function: logistic, Accuracy: 0.50
Activation Function: tanh, Accuracy: 1.00
Activation Function: relu, Accuracy: 0.50
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:690:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

[9]:

```
# Plotting the results
plt.bar(results.keys(), results.values(), color=['red', 'green', 'blue'])
plt.title('Comparison of Activation Functions on XOR Problem')
plt.ylabel('Accuracy')
plt.ylim(0, 1.1)
plt.show()
```


Comparison of Activation Functions on XOR Problem
Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
logistic
tanh
relu