```python
import numpy as np
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return x * (1 - x)
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])
y = np.array([[0],
              [0],
              [0],
              [1]])
np.random.seed(42)
input_neurons = 2
hidden_neurons = 2
output_neurons = 1
learning_rate = 0.5
epochs = 10000
weights_input_hidden = np.random.uniform(size=(input_neurons, hidden_neurons))
weights_hidden_output = np.random.uniform(size=(hidden_neurons, output_neurons))
bias_hidden = np.random.uniform(size=(1, hidden_neurons))
bias_output = np.random.uniform(size=(1, output_neurons))
```

```
for epoch in range(epochs):
    hidden_input = np.dot(X, weights_input_hidden) + bias_hidden
    hidden_output = sigmoid(hidden_input)
    final_input = np.dot(hidden_output, weights_hidden_output) + bias_output
    final_output = sigmoid(final_input)
    error = y - final_output
    d_output = error * sigmoid_derivative(final_output)
    error_hidden = d_output.dot(weights_hidden_output.T)
    d_hidden = error_hidden * sigmoid_derivative(hidden_output)
    weights_hidden_output += hidden_output.T.dot(d_output) * learning_rate
    bias_output += np.sum(d_output, axis=0, keepdims=True) * learning_rate
    weights_input_hidden += X.T.dot(d_hidden) * learning_rate
    bias_hidden += np.sum(d_hidden, axis=0, keepdims=True) * learning_rate
print("Trained Feedforward Neural Network Output:\n")
print(final_output)
predictions = (final_output > 0.5).astype(int)
print("\nPredicted Output (after thresholding):\n", predictions)
```

```
Trained Feedforward Neural Network Output:

[[0.00430516]
 [0.01255233]
 [0.0124792 ]
 [0.98350436]]

Predicted Output (after thresholding):
 [[0]
 [0]
 [0]
 [1]]
```