Neural Network

VISHAL GHASOLIYA- 24UADS1056

Propmpt:

Write a Python program that creates a perceptron to learn the AND gate using numpy and matplotlib. Start by setting all weights and bias to zero. While the program is training, show a table for each round that displays the inputs, what output we want, what output we got, the error, and the current weights. Draw a graph that shows a line separating the data points, and update this graph after every training round. Mark the correct predictions and wrong predictions with different colors or shapes so we can tell them apart. Add a 2-second wait time between each training round so we can watch how the learning happens step by step. After the training is complete and the perceptron has learned correctly, show a final graph that displays how the error changed over time during training. Also include a simple explanation in plain words about how the perceptron learns and makes its decisions.

Code:

```
import numpy as np

import matplotlib.pyplot as plt

import time


# Perceptron for AND Gate

class Perceptron:

    def __init__(self, learning_rate=0.1):

        # Initialize weights and bias to zero

        self.weights = np.zeros(2)  # Two input weights

        self.bias = 0.0

        self.learning_rate = learning_rate

        self.epoch_errors = []
```

```
    def activation(self, x):
        """Step activation function"""
        return 1 if x >= 0 else 0

    def predict(self, inputs):
        """Make prediction for given inputs"""
        summation = np.dot(inputs, self.weights) + self.bias
        return self.activation(summation)

    def train(self, X, y, max_epochs=100):
        """Train the perceptron"""
        print("\n" + "="*80)
        print("PERCEPTRON TRAINING FOR AND GATE")
        print("="*80)
        print("\nInitial Weights: w1={:.2f}, w2={:.2f}, bias={:.2f}\n".format(
            self.weights[0], self.weights[1], self.bias))

        # Setup interactive plotting
        plt.ion()
        fig, ax = plt.subplots(figsize=(8, 6))
        for epoch in range(max_epochs):
            total_error = 0
            print(f"\n{'='*80}")
            print(f"EPOCH {epoch + 1}")
            print(f"{'='*80}")
            print(f"{'Input':<15} {'Target':<10} {'Predicted':<12} {'Error':<10} {'Weights &
Bias':<30}")
```

```
        print("-" * 80)
        # Train on each sample
        for i in range(len(X)):
            inputs = X[i]
            target = y[i]
            # Predict
            prediction = self.predict(inputs)
            # Calculate error
            error = target - prediction
            total_error += abs(error)
            # Print current state
            weights_str = f"w1={self.weights[0]:.2f}, w2={self.weights[1]:.2f}, b={self.bias:.2f}"
            print(f"{str(inputs):<15} {target:<10} {prediction:<12} {error:<10}
{weights_str:<30}")
            # Update weights if there's an error
            if error != 0:
                self.weights += self.learning_rate * error * inputs
                self.bias += self.learning_rate * error

        # Store epoch error
        self.epoch_errors.append(total_error)
        # Print final weights for this epoch
        print("-" * 80)
        print(f"Epoch {epoch + 1} - Total Error: {total_error}")
        print(f"Updated Weights: w1={self.weights[0]:.2f}, w2={self.weights[1]:.2f},
bias={self.bias:.2f}")
        # Plot decision boundary
        self.plot_decision_boundary(X, y, ax, epoch + 1, total_error)
        plt.pause(2)  # 2 second delay
```

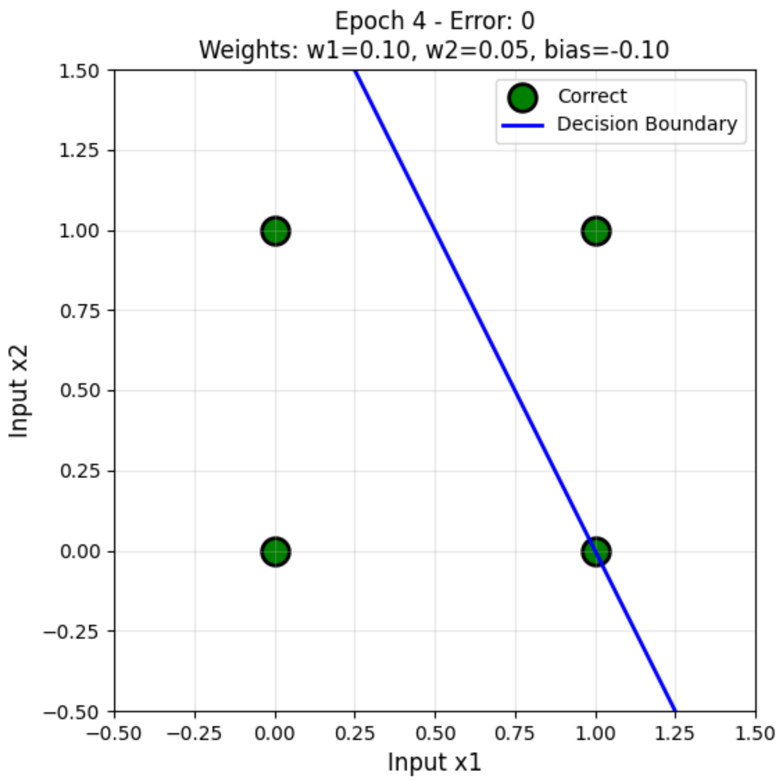
```
        # Check for convergence
        if total_error == 0:
            print(f"\n{'='*80}")
            print(f"CONVERGENCE ACHIEVED AT EPOCH {epoch + 1}!")
            print(f"{'='*80}")
            break
    plt.ioff()


    # Plot error vs epoch
    self.plot_error_graph()
    # Show explanation
    self.explain_perceptron()


def plot_decision_boundary(self, X, y, ax, epoch, error):
    """Plot decision boundary and data points"""
    ax.clear()
    # Separate correct and incorrect predictions
    correct_x = []
    correct_y = []
    wrong_x = []
    wrong_y = []
    for i in range(len(X)):
        prediction = self.predict(X[i])
        if prediction == y[i]:
            correct_x.append(X[i][0])
            correct_y.append(X[i][1])
        else:
            wrong_x.append(X[i][0])
```

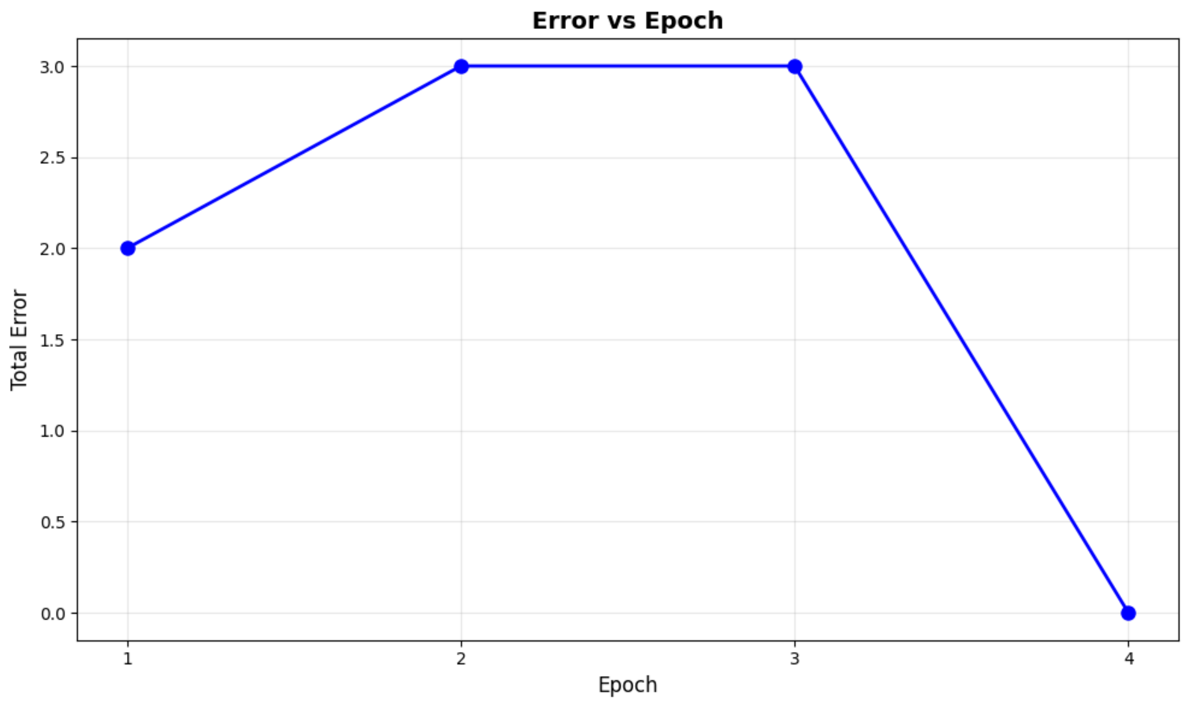
```
            wrong_y.append(X[i][1])

        # Plot correct predictions (green circles)

        if correct_x:

            ax.scatter(correct_x, correct_y, c='green', marker='o', s=200,

                      label='Correct', edgecolors='black', linewidth=2)

        # Plot wrong predictions (red X)

        if wrong_x:

            ax.scatter(wrong_x, wrong_y, c='red', marker='X', s=200,

                      label='Wrong', edgecolors='black', linewidth=2)

        # Plot decision boundary

        if self.weights[1] != 0:

            x_boundary = np.linspace(-0.5, 1.5, 100)

            # Decision boundary: w1*x1 + w2*x2 + bias = 0

            # Solving for x2: x2 = -(w1*x1 + bias) / w2

            y_boundary = -(self.weights[0] * x_boundary + self.bias) / self.weights[1]

            ax.plot(x_boundary, y_boundary, 'b-', linewidth=2, label='Decision Boundary')

        # Formatting

        ax.set_xlim(-0.5, 1.5)

        ax.set_ylim(-0.5, 1.5)

        ax.set_xlabel('Input x1', fontsize=12)

        ax.set_ylabel('Input x2', fontsize=12)

        ax.set_title(f'Epoch {epoch} - Error: {error}\nWeights: w1={self.weights[0]:.2f}, 
w2={self.weights[1]:.2f}, bias={self.bias:.2f}',

                    fontsize=12)

        ax.grid(True, alpha=0.3)

        ax.legend(loc='upper right')

        ax.set_aspect('equal')

        plt.draw()
```

```
    def plot_error_graph(self):
        """Plot error vs epoch after convergence"""
        plt.figure(figsize=(10, 6))
        epochs = range(1, len(self.epoch_errors) + 1)
        plt.plot(epochs, self.epoch_errors, 'bo-', linewidth=2, markersize=8)
        plt.xlabel('Epoch', fontsize=12)
        plt.ylabel('Total Error', fontsize=12)
        plt.title('Error vs Epoch', fontsize=14, fontweight='bold')
        plt.grid(True, alpha=0.3)
        plt.xticks(epochs)
        plt.tight_layout()
        plt.show()
# Main execution
if __name__ == "__main__":
    # AND gate training data
    # Inputs: [x1, x2]
    X = np.array([[0, 0],
                  [0, 1],
                  [1, 0],
                  [1, 1]])

    # Outputs for AND gate
    y = np.array([0, 0, 0, 1])

    # Create and train perceptron
    perceptron = Perceptron(learning_rate=0.03)
    perceptron.train(X, y, max_epochs=100)
```

```
# Test the trained perceptron
print("\n" + "="*80)
print("FINAL TESTING")
print("="*80)
print(f"{'Input':<15} {'Prediction':<15} {'Expected':<15}")
print("-" * 80)
for i in range(len(X)):
    prediction = perceptron.predict(X[i])
    print(f"{str(X[i]):<15} {prediction:<15} {y[i]:<15}")
print("="*80)
```

output:




Error vs Epoch
Total Error
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
Epoch

```
PS C:\New folder (2)> & C:\Python314\python.exe "c:/New folder (2)/practic/src/practic1.py"

================================================================================
EPOCH 1
================================================================================
Input          Target     Predicted    Error      Weights & Bias
--------------------------------------------------------------------------------
[0 0]          0          1            -1         w1=0.00, w2=0.00, b=0.00
[0 1]          0          0            0          w1=0.00, w2=0.00, b=-0.05
[1 0]          0          0            0          w1=0.00, w2=0.00, b=-0.05
[1 1]          1          0            1          w1=0.00, w2=0.00, b=-0.05
--------------------------------------------------------------------------------
Epoch 1 - Total Error: 2
Updated Weights: w1=0.05, w2=0.05, bias=0.00

================================================================================
EPOCH 2
================================================================================
Input          Target     Predicted    Error      Weights & Bias
--------------------------------------------------------------------------------
[0 0]          0          1            -1         w1=0.05, w2=0.05, b=0.00
[0 1]          0          1            -1         w1=0.05, w2=0.05, b=-0.05
[1 0]          0          0            0          w1=0.05, w2=0.00, b=-0.10
[1 1]          1          0            1          w1=0.05, w2=0.00, b=-0.10
--------------------------------------------------------------------------------
Epoch 2 - Total Error: 3
Updated Weights: w1=0.10, w2=0.05, bias=-0.05

================================================================================
EPOCH 3
================================================================================
Input          Target     Predicted    Error      Weights & Bias
--------------------------------------------------------------------------------
[0 0]          0          0            0          w1=0.10, w2=0.05, b=-0.05
[0 1]          0          1            -1         w1=0.10, w2=0.05, b=-0.05
[1 0]          0          1            -1         w1=0.10, w2=0.00, b=-0.10
[1 1]          1          0            1          w1=0.05, w2=0.00, b=-0.15
--------------------------------------------------------------------------------
Epoch 3 - Total Error: 3
Updated Weights: w1=0.10, w2=0.05, bias=-0.10

================================================================================
EPOCH 4
================================================================================
Input          Target     Predicted    Error      Weights & Bias
--------------------------------------------------------------------------------
[0 0]          0          0            0          w1=0.10, w2=0.05, b=-0.10
[0 1]          0          0            0          w1=0.10, w2=0.05, b=-0.10
[1 0]          0          0            0          w1=0.10, w2=0.05, b=-0.10
[1 1]          1          1            0          w1=0.10, w2=0.05, b=-0.10
--------------------------------------------------------------------------------
Epoch 4 - Total Error: 0
Updated Weights: w1=0.10, w2=0.05, bias=-0.10

================================================================================
CONVERGENCE ACHIEVED AT EPOCH 4!
```