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

class Perceptron:
    def __init__(self, weights, bias):
        self.weights = np.array(weights)
        self.bias = bias

    def activation(self, x):
        return 1 if x > 0 else 0

    def predict(self, inputs):
        total = np.dot(self.weights, inputs) + self.bias
        return self.activation(total)

weights = [1, 1]
bias = -0.5  # Adjusted bias for OR gate

or_perceptron = Perceptron(weights, bias)

inputs = np.array([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])
outputs = np.array([0, 1, 1, 1])  # OR gate outputs

predictions = np.array([or_perceptron.predict(x) for x in inputs])

plt.figure(figsize=(8, 6))

for i, label in enumerate(outputs):
    if label == 0:
        plt.scatter(inputs[i, 0], inputs[i, 1], color='red', label='Class 0' if i == 0
else "")
    else:
        plt.scatter(inputs[i, 0], inputs[i, 1], color='green', label='Class 1' if i ==
3 else "")

x_values = np.linspace(-0.1, 1.1, 100)
y_values = -(weights[0] * x_values + bias) / weights[1]
```

```
plt.plot(x_values, y_values, label="Decision Boundary", color='blue')

plt.title("OR Gate with Perceptron")
plt.xlabel("Input x1")
plt.ylabel("Input x2")
plt.axhline(0, color="black", linewidth=0.5)
plt.axvline(0, color="black", linewidth=0.5)
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend()
plt.xlim(-0.1, 1.1)
plt.ylim(-0.1, 1.1)

plt.show()
```