Lab - 1 | 22 January 2026

↳ 6 cells hidden

Lab - 2 | 29 January 2026

↳ 14 cells hidden

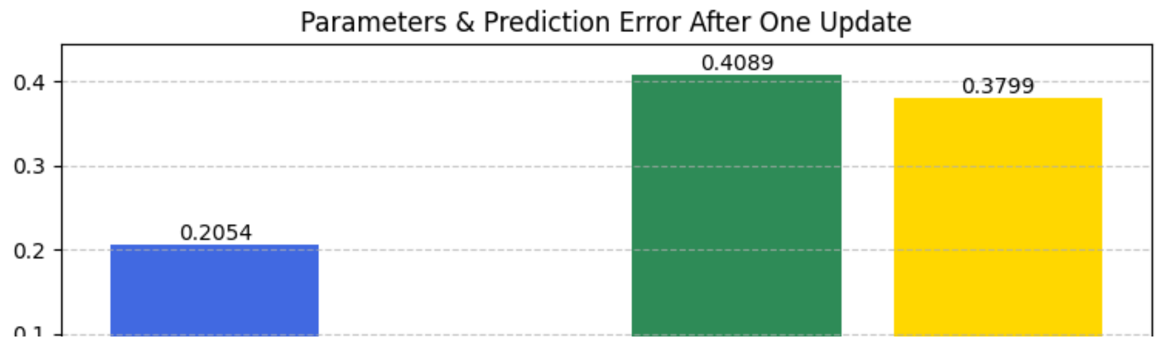
Lab - 3 | 05 February 2026

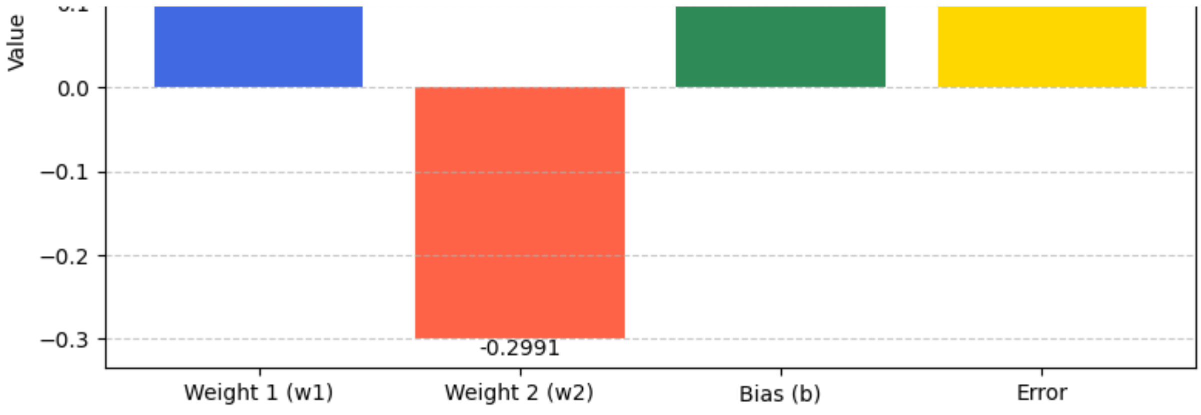
Forward Pass + Backpropagation

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

# ------------------ Input & Target ------------------
x1, x2 = 0.6, 0.1
y = 1

# ------------------ Initial Parameters ------------------
w1, w2, b = 0.2, -0.3, 0.4
lr = 0.1

# ------------------ Sigmoid Function ------------------
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

# ------------------ Forward Pass ------------------
z = w1 * x1 + w2 * x2 + b
y_pred = sigmoid(z)

# Error BEFORE update
error = abs(y - y_pred)

# ------------------ Backpropagation ------------------
dL_dy_pred = y_pred - y
dy_pred_dz = y_pred * (1 - y_pred)
dL_dz = dL_dy_pred * dy_pred_dz
```

```
    dL_dw1 = dL_dz * x1
    dL_dw2 = dL_dz * x2
    dL_db = dL_dz

    # ------------------ Parameter Update ------------------
    w1 -= lr * dL_dw1
    w2 -= lr * dL_dw2
    b  -= lr * dL_db

    print("---------------- Updated Parameters ----------------")
    print(f"w1: {w1:.4f}, w2: {w2:.4f}, b: {b:.4f}")
    print(f"Prediction Error (before update): {error:.4f}")
    print("----------------------------------------------------")

    # ------------------ Visualization ------------------
    labels = ['Weight 1 (w1)', 'Weight 2 (w2)', 'Bias (b)', 'Error']
    values = [w1, w2, b, error]
    colors = ['royalblue', 'tomato', 'seagreen', 'gold']  # Different co

    plt.figure(figsize=(8, 5))
    bars = plt.bar(labels, values, color=colors)

    plt.title('Parameters & Prediction Error After One Update')
    plt.ylabel('Value')
    plt.grid(axis='y', linestyle='--', alpha=0.7)

    # Add value labels on bars
    for bar in bars:
        height = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, height,
                 f'{height:.4f}', ha='center',
                 va='bottom' if height >= 0 else 'top')

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

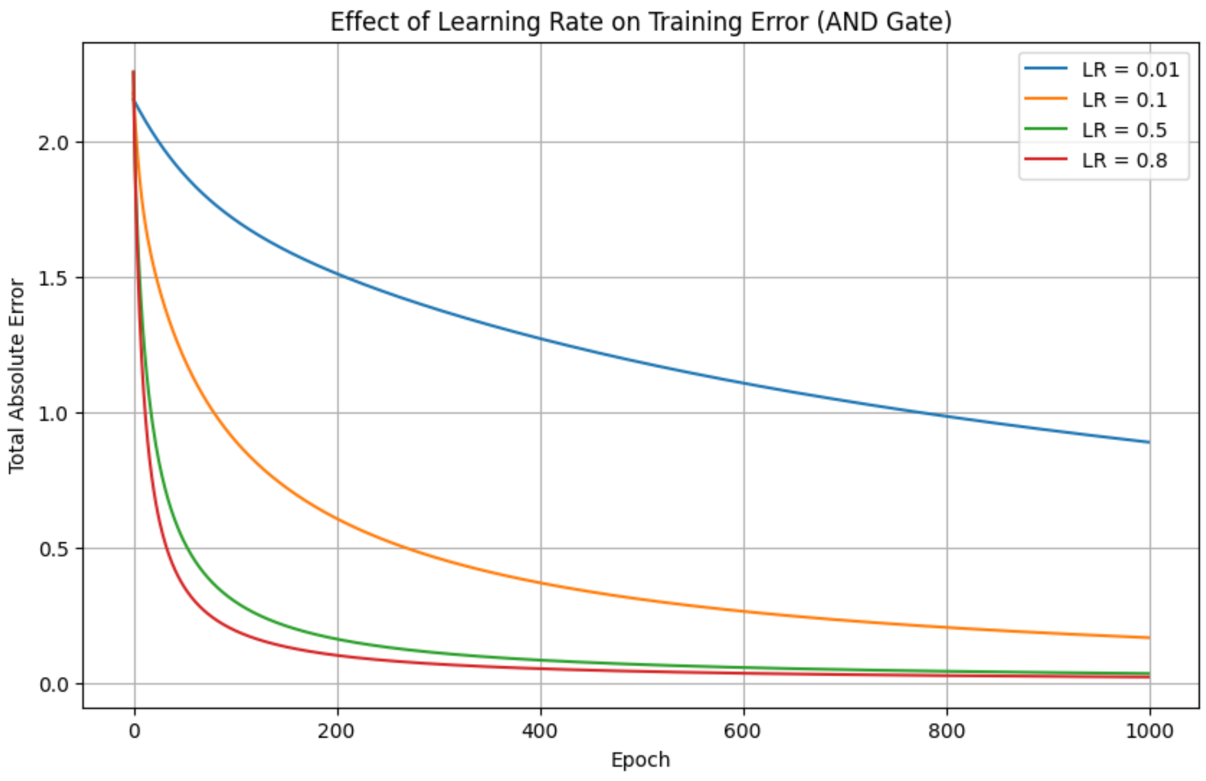
```
---------------- Updated Parameters ----------------
w1: 0.2054, w2: -0.2991, b: 0.4089
Prediction Error (before update): 0.3799
----------------------------------------------------
```





Perceptron using sigmoid activation to learn the AND logic gate using gradient descent

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

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

learning_rates = [0.01, 0.1, 0.5, 0.8]  # 4 different LRs
epochs = 1000

def sigmoid(z):
    return 1/(1+np.exp(-z))

plt.figure(figsize=(10,6))

for lr in learning_rates:
    # Reset weights for each LR
    w = np.array([0.1, 0.2], dtype=float)
    b = 0.3

    errors_per_epoch = []
    for epoch in range(epochs):
        total_absolute_error = 0

        for i in range(len(X)):
            z = np.dot(X[i], w) + b
            y_pred = sigmoid(z)

            total_absolute_error += abs(y[i] - y_pred)

            dz = y_pred - y[i]
            dw = dz * X[i]
            db = dz

            w -= lr * dw
            b -= lr * db

        errors_per_epoch.append(total_absolute_error)

    plt.plot(errors_per_epoch, label=f"LR = {lr}")

plt.title("Effect of Learning Rate on Training Error (AND Gate)")
plt.xlabel("Epoch")
plt.ylabel("Total Absolute Error")
plt.legend()
plt.grid(True)
plt.show()
```


Effect of Learning Rate on Training Error (AND Gate)
LR = 0.01
LR = 0.1
LR = 0.5
LR = 0.8
Total Absolute Error
Epoch