Double-click (or enter) to edit

```
import numpy as np

# Initialize parameters
w1 = 0.0
w2 = 0.0
learning_rate = 0.01
epochs = 1000

# Synthetic values (simulate small dataset)
x_values = np.array([1, 2, 3, 4])
y_values = np.array([3, 5, 7, 9])  # y = 2x + 1 (ground truth)

# Batch Gradient Descent
for epoch in range(epochs):
    # Predictions
    y_pred = w1 * x_values + w2

    # Compute gradients (mean over all data)
    error = y_pred - y_values
    dw1 = (2 / len(x_values)) * np.dot(error, x_values)
    dw2 = (2 / len(x_values)) * np.sum(error)

    # Update weights
    w1 -= learning_rate * dw1
    w2 -= learning_rate * dw2

    # Compute loss
    loss = np.mean(error ** 2)

    # Print loss every 100 epochs
    if epoch % 100 == 0:
        print(f"Epoch {epoch}: Loss = {loss:.4f}, w1 = {w1:.4f}, w2 = {w2:.4f}")
```

```
print("\nFinal Weights:")
print(f"w1 = {w1:.4f}, w2 = {w2:.4f}")
```

```
Epoch 0: Loss = 41.0000, w1 = 0.3500, w2 = 0.1200
Epoch 100: Loss = 0.0075, w1 = 2.0720, w2 = 0.7883
Epoch 200: Loss = 0.0041, w1 = 2.0534, w2 = 0.8431
Epoch 300: Loss = 0.0023, w1 = 2.0395, w2 = 0.8838
Epoch 400: Loss = 0.0012, w1 = 2.0293, w2 = 0.9139
Epoch 500: Loss = 0.0007, w1 = 2.0217, w2 = 0.9362
Epoch 600: Loss = 0.0004, w1 = 2.0161, w2 = 0.9527
Epoch 700: Loss = 0.0002, w1 = 2.0119, w2 = 0.9650
Epoch 800: Loss = 0.0001, w1 = 2.0088, w2 = 0.9740
Epoch 900: Loss = 0.0001, w1 = 2.0065, w2 = 0.9808

Final Weights:
w1 = 2.0049, w2 = 0.9857
```

```
# Stochastic Gradient Descent (SGD) with simple values and 10 epochs

# Step 1: Initialize weight and bias
w = 0.0     # weight
b = 0.0     # bias

# Step 2: Set learning rate
lr = 0.01

# Step 3: Create small synthetic dataset (x, y)
# True relationship: y = 2x + 1
x = [1, 2, 3, 4]
y = [3, 5, 7, 9]

# Step 4: Training for 10 epochs
for epoch in range(10):
    total_error = 0

    # SGD — update for each individual data point
    for i in range(len(x)):
```

```
        y_pred = w * x[i] + b         # predicted value
        error = y_pred - y[i]         # prediction error

        # Gradients (for one sample only)
        dw = error * x[i]
        db = error

        # Update weights immediately (SGD style)
        w = w - lr * dw
        b = b - lr * db

        # Accumulate error for monitoring
        total_error += error ** 2

    # Step 5: Print results after each epoch
    print(f"Epoch {epoch+1}: Loss = {total_error/len(x):.4f}, w = {w:.4f}, b = {b:.4f}")
```

```
Epoch 1: Loss = 33.2401, w = 0.6266, b = 0.2200
Epoch 2: Loss = 16.2972, w = 1.0649, b = 0.3745
Epoch 3: Loss = 7.9966, w = 1.3715, b = 0.4831
Epoch 4: Loss = 3.9290, w = 1.5858, b = 0.5596
Epoch 5: Loss = 1.9350, w = 1.7357, b = 0.6135
Epoch 6: Loss = 0.9571, w = 1.8404, b = 0.6518
Epoch 7: Loss = 0.4771, w = 1.9135, b = 0.6791
Epoch 8: Loss = 0.2413, w = 1.9645, b = 0.6986
Epoch 9: Loss = 0.1252, w = 2.0000, b = 0.7127
Epoch 10: Loss = 0.0680, w = 2.0247, b = 0.7231
```

```
# Batch Gradient Descent (with simple values and only 10 epochs)

# Step 1: Initialize weight and bias
w = 0.0     # weight
b = 0.0     # bias

# Step 2: Set learning rate
lr = 0.01
```

```
# Step 3: Create small synthetic dataset (x, y)
# Let's say the true relation is: y = 2x + 1
x = [1, 2, 3, 4]
y = [3, 5, 7, 9]

# Step 4: Training for 10 epochs
for epoch in range(10):
    total_error = 0
    dw = 0  # gradient for w
    db = 0  # gradient for b

    # Loop through all data (batch gradient descent)
    for i in range(len(x)):
        y_pred = w * x[i] + b         # predicted value
        error = y_pred - y[i]         # prediction error
        dw += error * x[i]            # accumulate gradient for w
        db += error                   # accumulate gradient for b
        total_error += error ** 2     # accumulate squared error

    # Take average gradients
    dw = dw / len(x)
    db = db / len(x)

    # Step 5: Update weights
    w = w - lr * dw
    b = b - lr * db

    # Step 6: Print results
    print(f"Epoch {epoch+1}: Loss = {total_error/len(x):.4f}, w = {w:.4f}, b = {b:.4f}")
```

```
Epoch 1: Loss = 41.0000, w = 0.1750, b = 0.0600
Epoch 2: Loss = 34.4408, w = 0.3354, b = 0.1150
Epoch 3: Loss = 28.9313, w = 0.4823, b = 0.1655
Epoch 4: Loss = 24.3034, w = 0.6170, b = 0.2118
Epoch 5: Loss = 20.4162, w = 0.7405, b = 0.2542
Epoch 6: Loss = 17.1511, w = 0.8536, b = 0.2932
```

```
Epoch 7: Loss = 14.4084, w = 0.9572, b = 0.3289
Epoch 8: Loss = 12.1047, w = 1.0522, b = 0.3617
Epoch 9: Loss = 10.1697, w = 1.1393, b = 0.3918
Epoch 10: Loss = 8.5443, w = 1.2190, b = 0.4194
```