LINEAR REGRESSION

Linear Regression

- Hồi quy tuyến tính đơn biến (Simple Linear Regression)
- Hồi quy tuyến tính đa biến (Multiple Linear Regression)

Hồi quy tuyến tính đơn biến

- Bài toán đặt ra:
  - Giả sử ta có tập dữ liệu:

Experience	Salary
3	60
4	55
5	66
6	93
7	?

  - Hãy xây dựng một chương trình ML để tự động dự đoán tiền lương của nhân viên dựa trên số năm kinh nghiệm của họ.
- Câu hỏi đặt ra là, làm cách nào để xây dựng một chương trình có thể tự động dự đoán tiền lương dựa vào số năm kinh nghiệm?

Hồi quy tuyến tính đơn biến

- Quan sát bảng dữ liệu + Trực quan hóa bảng dữ liệu bằng biểu đồ scatter, ta thấy:
  - Khi kinh nghiệm tăng -> lương tăng
  - Các điểm dữ liệu thường tạo thành dạng gần thẳng

$\Rightarrow$Mô hình phù hợp nhất: Hồi quy tuyến tính đơn biến (Simple Linear Regression)

$\Rightarrow$Nghĩa là, phải xây dựng được công thức một đường thẳng có dạng:

$$Salary = W . Experience + b$$

Experience	Salary
3	60
4	55
5	66
6	93
7	?

Hồi quy tuyến tính đơn biến

- Với tập dữ liệu đã có, ta có thể xây dựng các đường thẳng sau:
  - Salary1 = W1 . Experience1 + b1
  - Salary2 = W2 . Experience2 + b2
  - Salary3 = W3 . Experience3 + b3
  - Salary4 = W4 . Experience1 + b4
- Nghĩa là ta có:
  - 60 = 3w1 + b1 => w1 = 16, b1 = 12

  => Salary = 16 . Experience + 12
  - 55 = 4w2 + b2 => w2 = 12, b2 = 7

  => Salary = 12 . Experience + 7
  - 66 = 5w3 + b3 => w3 = 12, b3 = 6

  => Salary = 12 . Experience + 6
  - 93 = 6w4 + b4 => w4 = 15, b4 = 3

  => Salary = 15 . Experience + 3

Experience	Salary
3	60
4	55
5	66
6	93
7	?

Hồi quy tuyến tính đơn biến

- Nghĩa là ta có:
  - Salary = 16 x Experience + 12

    => Experience = 7 => Salary = 16 x 7 +12 = 124
  - Salary = 12 x Experience + 7

    => Experience = 7 => Salary = 12 x 7 + 7= 91
  - Salary = 12 x Experience + 6

    => Experience = 7 => Salary = 12 x 7 + 6 = 90
  - Salary = 15 x Experience + 3

    => Experience = 7 => Salary = 15 x 7 + 3= 108
- Vậy rốt cuộc đường thẳng nào cho câu trả lời phù hợp nhất?

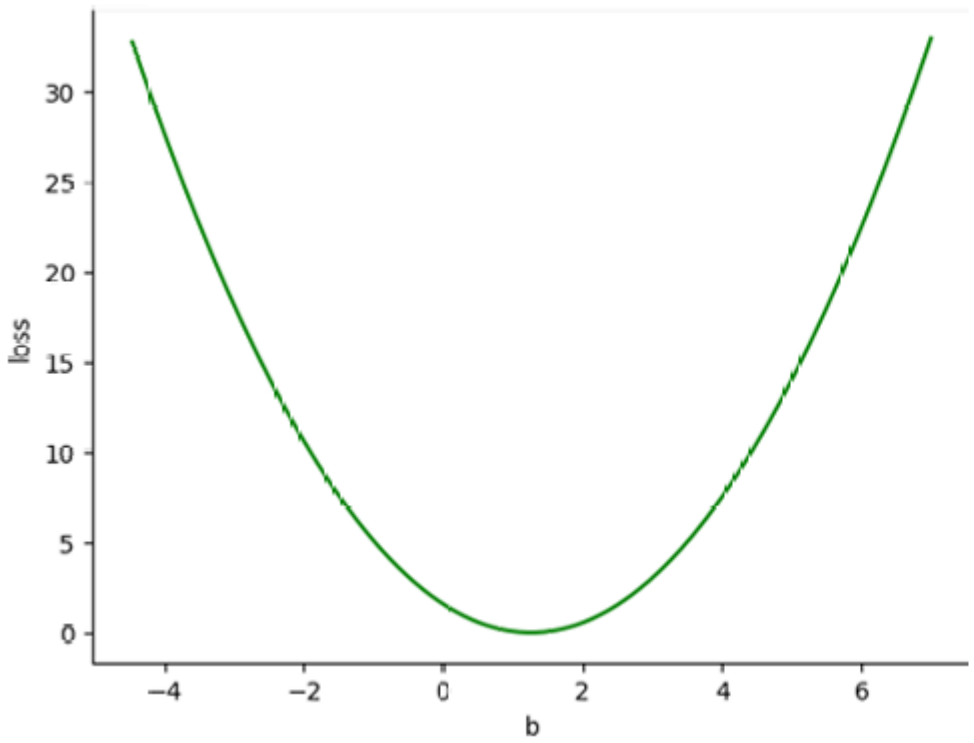
Experience	Salary
3	60
4	55
5	66
6	93
7	?

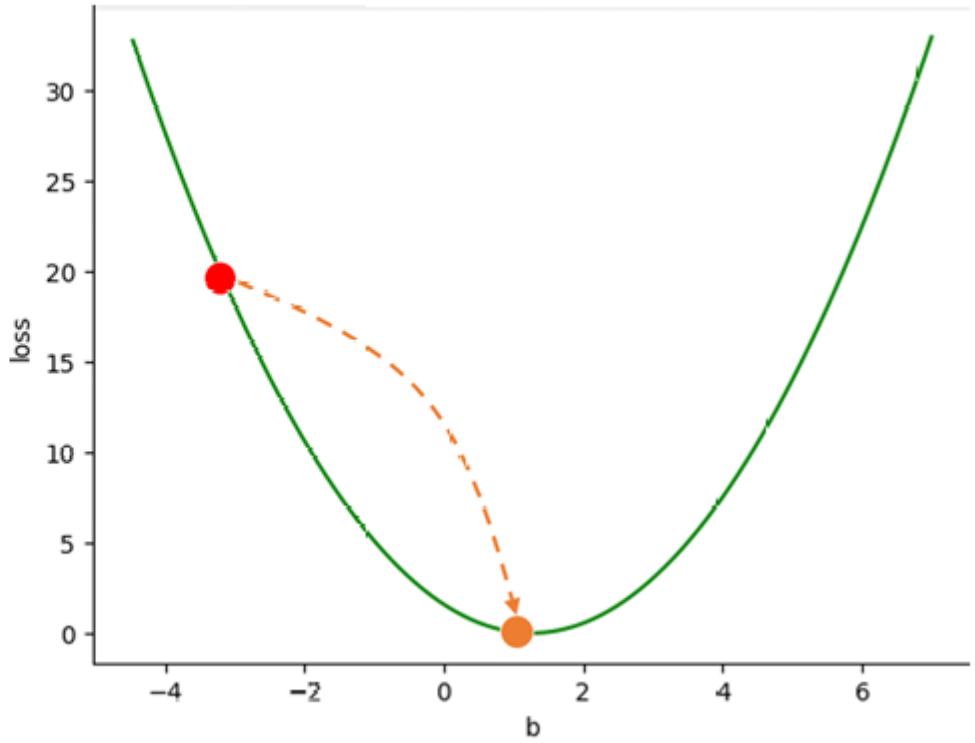
Hồi quy tuyến tính đơn biến

- Mục tiêu của mô hình Simple Linear Regression:
  - Tìm một đường thẳng tốt nhất mô tả mối liên hệ giữa một biến đầu vào (x) và một biến đầu ra (y), để từ đó dự đoán giá trị y cho giá trị x mới.
- Cụ thể:
  - Từ các (x,y) đã có => tìm w, b để xây dựng đường thẳng $\hat{y} = w \times x + b$ sao cho $\mathrm{L} = (\hat{y} - y)^2$ nhỏ nhất

Hồi quy tuyến tính đơn biến

- Từ các phân tích, ta có một bài toán mới được phát biểu như sau: Tìm w và b để giá trị loss (L) là nhỏ nhất.





Hồi quy tuyến tính đơn biến: Quy trình huấn luyện

1. Khởi tạo ngẫu nhiên giá trị cho hai tham số là w và b.

2. Với mỗi mẫu dữ liệu thứ i trong bộ dữ liệu, ta áp dụng các bước tính toán sau:

(a) Thực hiện dự đoán output $\widehat{y_i}$ với input là x_i theo công thức sau: $\widehat{y_i} = f(x_i) = w \times x_i + b$

(b) Để tính toán sự chênh lệch giữa giá trị dự đoán so với giá trị thực tế, ta đưa y_i và $\widehat{y_i}$ vào hàm tính loss như sau: $L(\widehat{y_i}, y_i) = (\widehat{y_i} - y_i)^2$

Hồi quy tuyến tính đơn biến

- (c) Ta tìm giá trị đạo hàm tại mẫu dữ liệu thứ i cho hai tham số w và b như sau:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial b} = 2(\hat{y}_i - y_i), \qquad \frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}_i}\frac{\partial \hat{y}_i}{\partial w} = 2x_i(\hat{y}_i - y_i)$$

- (d) Với giá trị đạo hàm riêng vừa tìm được cho hai tham sô w và b trên mẫu dữ liệu i, ta cập nhật lại giá trị mới cho hai tham số này với công thức sau:

$$w = w - \eta\frac{\partial L}{\partial w}, \qquad b = b - \eta\frac{\partial L}{\partial b}$$

  η là learning rate (tạm dịch: hệ số học).

- Sau đó, như đã đề cập, ta lặp lại bước 2 cho đến khi xử lý hết tất cả các mẫu dữ liệu trong bộ dữ liệu.

THỰC HÀNH

1. Khởi tạo w = 10, b = 5 và hệ số học η = 0.01.
2. Duyệt qua từng mẫu dữ liệu:
   1. Mẫu 0:
   a. $\widehat{y_0} = f(x_0)$ = 10 x 3 +5 =35
   b. $L(\widehat{y_0}, y_0)$ = (35-60)^2=625
   c. $\frac{\partial L}{\partial w} = 2x_0(\widehat{y_0} - y_0)$ = 2x3(35-60)=-150

   $\frac{\partial L}{\partial b} = 2\,(\widehat{y_0} - y_0)$ = 2(35-60)=-50

   d. $w = w - \eta \frac{\partial L}{\partial w}$=10-0.01 x (-150)=11.5

   b $= b - \eta \frac{\partial L}{\partial b}$=5-0.01 x (-50)=5.5

Sau mẫu 0: $w = 11.5, b = 5.5$ và hệ số học η = 0.01
(Dùng bộ này để tính mẫu 1)

Experience	Salary
3	60
4	55
5	66
6	93
7	?

VIẾT CHƯƠNG TRÌNH

Xây dựng mô hình

1. Khởi tạo w và b

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

```
INPUT_DIM = X_train.shape[1] # X is 1-dimensional
OUTPUT_DIM = y_train.shape[1] # y is 1-dimensional
```

```
# Initialize random weights
W = 0.01 * np.random.randn(INPUT_DIM, OUTPUT_DIM)
b = np.zeros((1, 1))
print (f"W: {W.shape}")
print (f"b: {b.shape}")
```

VIẾT CHƯƠNG TRÌNH

Xây dựng mô hình

2. Modeling

2.1. Tính $\widehat{y_0}$

```
# Forward pass [NX1] · [1X1] = [NX1]
y_pred = np.dot(X_train, W) + b
```

2.2. Tính L

```
# Loss
N = len(y_train)
loss = (1/N) * np.sum((y_train - y_pred)**2)
print (f"loss: {loss:.2f}")
```

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

VIẾT CHƯƠNG TRÌNH

Xây dựng mô hình

2. Modeling

2.3. Tính gradient

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

```
# Backpropagation
dW = -(2/N) * np.sum((y_train - y_pred) * X_train)
db = -(2/N) * np.sum((y_train - y_pred) * 1)
```

2.4. Cập nhật w và b

```
LEARNING_RATE = 1e-1
```

```
# Update weights
W += -LEARNING_RATE * dW
b += -LEARNING_RATE * db
```

VIẾT CHƯƠNG TRÌNH

Huấn luyện mô hình theo cơ chế full-sample

```
NUM_EPOCHS = 100
```

```
# Initialize random weights
W = 0.01 * np.random.randn(INPUT_DIM, OUTPUT_DIM)
b = np.zeros((1, ))

# Training loop
for epoch_num in range(NUM_EPOCHS):

    # Forward pass [NX1] · [1X1] = [NX1]
    y_pred = np.dot(X_train, W) + b

    # Loss
    loss = (1/len(y_train)) * np.sum((y_train - y_pred)**2)

    # Show progress
    if epoch_num%10 == 0:
        print (f"Epoch: {epoch_num}, loss: {loss:.3f}")

    # Backpropagation
    dW = -(2/N) * np.sum((y_train - y_pred) * X_train)
    db = -(2/N) * np.sum((y_train - y_pred) * 1)

    # Update weights
    W += -LEARNING_RATE * dW
    b += -LEARNING_RATE * db
```

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

VIẾT CHƯƠNG TRÌNH

Xây dựng mô hình

4. Testing (Evaluation)

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

```
# Predictions
pred_train = W*X_train + b
pred_test = W*X_test + b
```

```
# Train and test MSE
train_mse = np.mean((y_train - pred_train) ** 2)
test_mse = np.mean((y_test - pred_test) ** 2)
print (f"train_MSE: {train_mse:.2f}, test_MSE: {test_mse:.2f}")
```

VIẾT CHƯƠNG TRÌNH

Huấn luyện mô hình theo cơ chế one-sample

```
w, b = initialize_params()
N = len(y_train)
LEARNING_RATE = 1e-2
epoch_max = 100

for epoch in range(epoch_max):
    for i in range(N):
        # get a sample
        x = X_train[i]      # vector shape (d,)
        y = y_train[i]

        # compute output
        y_pred = np.dot(w, x) + b

        # compute gradients
        dw = 2*(y_pred - y) * x
        db = 2*(y_pred - y)

        # update parameters
        w -= LEARNING_RATE * dw
        b -= LEARNING_RATE * db

    # print progress
    y_pred_all = X_train.dot(w) + b
    loss = np.mean((y_pred_all - y_train)**2)

    if epoch % 10 == 0:
        print(f"Epoch {epoch}, loss = {loss:.4f}")
```

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

Hồi quy tuyến tính đa biến

- Bài toán đặt ra:
  - Giả sử ta có tập dữ liệu:

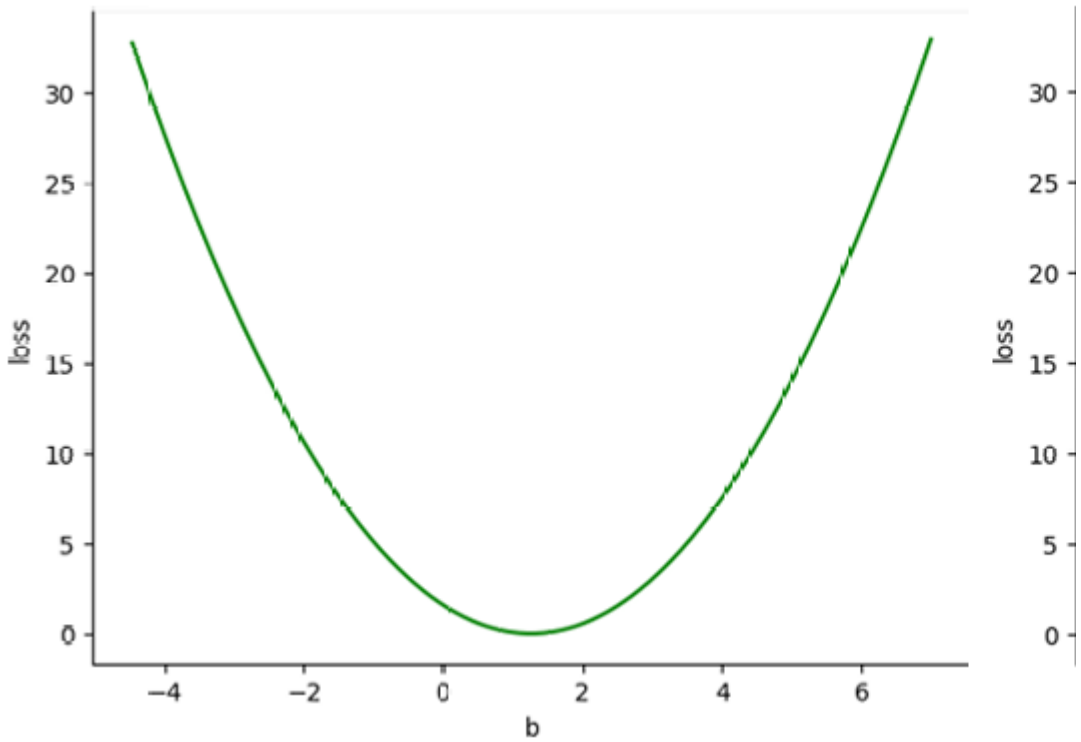
Experience	Degree	Salary
3	1	60
4	1	55
5	2	66
6	2	93
7	?	?

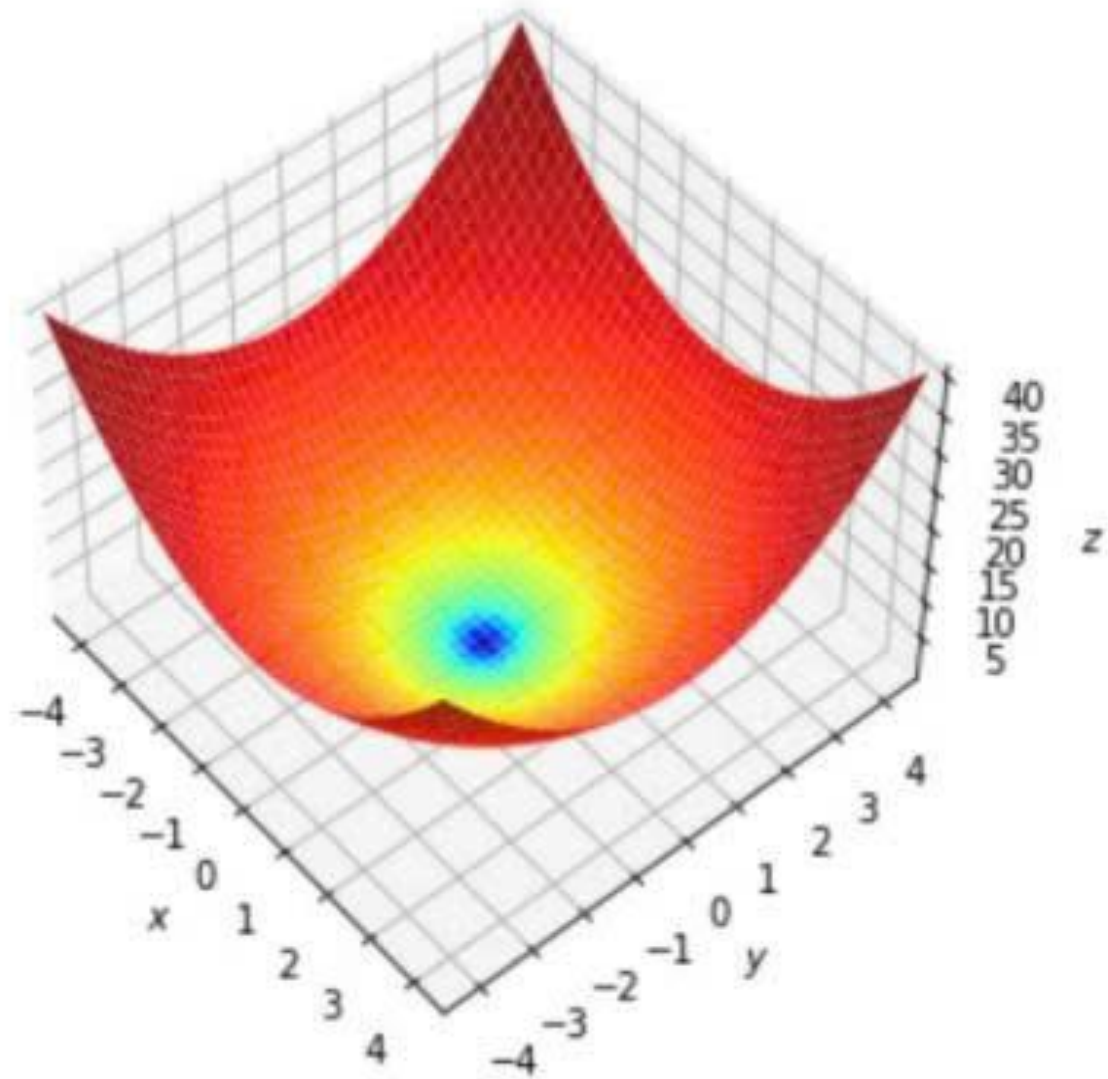
  - Hãy xây dựng một chương trình ML để tự động dự đoán tiền lương của nhân viên dựa trên số năm kinh nghiệm và bậc lương của họ.
- Câu hỏi đặt ra là, làm cách nào để xây dựng một chương trình có thể tự động dự đoán tiền lương dựa vào số năm kinh nghiệm và bậc lương?

Hồi quy tuyến tính đa biến

- Mục tiêu của mô hình Multiple Linear Regression:
  - Tìm một mặt phẳng tốt nhất mô tả mối liên hệ giữa các biến đầu vào (x1, x2) và một biến đầu ra (y), để từ đó dự đoán giá trị y cho các giá trị x mới.
- Cụ thể:
  - Từ các (x,y) đã có => tìm w_1, w_2, b để xây dựng mặt phẳng $\hat{y} = w_1 x_1 + w_2 x_2 + b$ sao cho $\mathrm{L} = (\hat{y} - y)^2$ nhỏ nhất
  - Hoặc viết gọn hơn: $\hat{y} = w^{\mathrm{T}} x + b$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$$


loss
30
25
20
15
10
5
0
−4
−2
0
2
4
6
b


40
35
30
25
20
15
10
5
z
−4
−3
−2
−1
0
1
2
3
4
x
y

Hồi quy tuyến tính đa biến: Quy trình huấn luyện

1. Khởi tạo ngẫu nhiên giá trị cho các tham số là w_1, w_2, b .

2. Lặp qua Dữ liệu: Với mỗi mẫu dữ liệu (x_i, y_i)

- Dự đoán output $\hat{y_i}$ theo công thức sau: $\hat{y_i} = w^{\mathrm{T}} x_i + b$ $\hat{y_i} = \sum_j w_j x_{ij} + b$

- Tính loss như sau: $L(\hat{y_i}, y_i) = (\hat{y_i} - y_i)^2$

- Tính đạo hàm:

$$\frac{\partial L_i}{\partial w_j} = 2x_{ij}(\hat{y_i} - y_i)$$

$$\frac{\partial L_i}{\partial b} = 2\,(\hat{y_i} - y_i)$$

- Cập nhật: w_1, w_2, b $w_j = w_j - \eta \frac{\partial L_i}{\partial w_j}, \quad b = b - \eta \frac{\partial L_i}{\partial b}$

3. Lặp lại bước 2 qua nhiều lần (epochs) cho đến khi loss đạt mức tối ưu (giá trị nhỏ nhất).

THỰC HÀNH

1. Khởi tạo giá trị cho $w_1 = 10, w_2 = 2, b = 5$ và hệ số học η = 0.01.
2. Duyệt qua từng mẫu dữ liệu:
   1. Mẫu 0: x_{01}=3, x_{02}=1, y_0=60
   a. $\widehat{y_0} = f(x_{0j})$ = $w_1 x_{01} + w_2 x_{02} + b$
   $= 10 \text{ x } 3 + 2 \text{ x } 1 + 5 = 37$
   b. $L(\widehat{y_0}, y_0)$ = (60-37)^2=529

Experience	Degree	Salary
3	1	60
4	1	55
5	2	66
6	2	93
7	?	?

THỰC HÀNH

1. Khởi tạo giá trị cho $w_1 = 10, w_2 = 2, b = 5$ và hệ số học η = 0.01.
2. Duyệt qua từng mẫu dữ liệu:
   1. Mẫu 0: x_{01}=3, x_{02}=1, y_0=60

   c. Đạo hàm (theo từng tham số)

$$\frac{\partial L_0}{\partial w_1} = 2x_{01}(\widehat{y_0} - y_0) = 2x3(60\text{-}37) = -138$$

$$\frac{\partial L_0}{\partial w_2} = 2x_{02}(\widehat{y_0} - y_0) = 2x1(60\text{-}37) = -46$$

$$\frac{\partial L_0}{\partial b} = 2\,(\widehat{y_0} - y_0) = 2(60\text{-}37) = -46$$

Experience	Degree	Salary
3	1	60
4	1	55
5	2	66
6	2	93
7	?	?

THỰC HÀNH

1. Khởi tạo giá trị cho $w_1 = 10, w_2 = 2, b = 5$ và hệ số học η = 0.01.
2. Duyệt qua từng mẫu dữ liệu:
   1. Mẫu 0: x_{01}=3, x_{02}=1, y_0=60

   d. Cập nhật

$$w_1 = w_1 - \eta \frac{\partial L_0}{\partial w_1}$$=10-0.01 x (-138)=11.38

$$w_2 = w_2 - \eta \frac{\partial L_0}{\partial w_2}$$=2-0.01 x (-46)=2.46

$$\mathrm{b} = b - \eta \frac{\partial L_0}{\partial b}$$=5-0.01 x (-46)=5.46

Experience	Degree	Salary
3	1	60
4	1	55
5	2	66
6	2	93
7	?	?

Sau mẫu 0: $w_1 = 11.38, w_2 = 2.46, b = 5.46$ và hệ số học η = 0.01

(Dùng bộ này để tính mẫu 1)

VIẾT CHƯƠNG TRÌNH

Huấn luyện mô hình theo cơ chế one-sample

```
# Initialize
w1, w2, w3, b = initialize_params()
LEARNING_RATE = 1e-2
epoch_max = 100
N = len(y_train)
for epoch in range(epoch_max):
    # Shuffle data
    indices = np.random.permutation(N)

    for i in indices:
        # Get one sample
        x1 = X_train[i][0]
        x2 = X_train[i][1]
        x3 = X_train[i][2]
        y  = y_train[i]

        # compute output
        y_pred = w1*x1 + w2*x2 + w3*x3 + b

        # Compute gradients for 1 sample
        dw1 = 2*(y_pred - y)*x1
        dw2 = 2*(y_pred - y)*x2
        dw3 = 2*(y_pred - y)*x3
        db  = 2*(y_pred - y)

        # update parameters
        w1 -= LEARNING_RATE * dw1
        w2 -= LEARNING_RATE * dw2
        w3 -= LEARNING_RATE * dw3
        b  -= LEARNING_RATE * db

    # Compute full MSE loss after epoch
    y_pred_all = w1*X_train[:,0] + w2*X_train[:,1] + w3*X_train[:,2] + b
    loss = np.mean((y_pred_all - y_train)**2)

    # Print progress
    if epoch % 10 == 0:
        print(f"Epoch {epoch}, loss = {loss:.4f}")
```

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

VIẾT CHƯƠNG TRÌNH

Huấn luyện mô hình theo cơ chế full-sample

```
w1, w2, w3, b = initialize_params()
N = len(y_train)
LEARNING_RATE = 1e-2
epoch_max = 100

for epoch in range(epoch_max):
    # compute output
    y_pred = w1*x1 + w2*x2 + w3*x3 + b

    # compute loss (MSE)
    loss = (1/N) * np.sum((y_train - y_pred)**2)

    # Show progress
    if epoch % 10 == 0:
        print(f"Epoch: {epoch}, loss: {loss:.3f}")

    # compute gradients
    dw1 = (2/N) * np.sum((y_pred - y_train) * x1)
    dw2 = (2/N) * np.sum((y_pred - y_train) * x2)
    dw3 = (2/N) * np.sum((y_pred - y_train) * x3)
    db  = (2/N) * np.sum((y_pred - y_train))

    # update parameters
    w1 -= LEARNING_RATE * dw1
    w2 -= LEARNING_RATE * dw2
    w3 -= LEARNING_RATE * dw3
    b  -= LEARNING_RATE * db
```

Đọc và hoàn thiện các đoạn code sau để có một chương trình huấn luyện mô hình hoàn chỉnh

Ridge regression và Lasso regression

1. **Ridge regression**:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha||\mathbf{w}||_2^2$$

2. **Lasso regression**:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \alpha||\mathbf{w}||_1$$

Experience	Degree	Salary
3	1	60
4	1	55
5	2	66
6	2	93
7	3	?

Experience	Degree	Salary
3	1	60
4	1	55
5	2	66
6	2	93
7	1	48
8	1	51
7	?	?