HỆ PHƯƠNG TRÌNH TUYẾN TÍNH

Nguyễn Mạnh Hùng

AI Academy Vietnam

September, 2021

Nội dung

1. Hệ phương trình tuyến tính
2. Các phương pháp giải
3. Thực hành trên Python
4. Giới thiệu bài toán hồi quy tuyến tính

Khái niệm

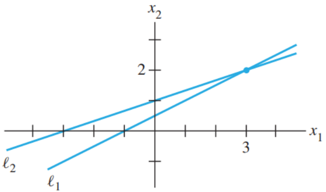
Hệ phương trình tuyến tính (system of linear equations)

Hệ m phương trình tuyến tính n ẩn là hệ có dạng:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots\dots\dots\dots\dots\dots\dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases} \qquad (1)$$

Kí hiệu $\{a_{ij}\}$ và $\{b_i\}$ là các hệ số cho trước và $\{x_j\}$ là các ẩn số cần tìm.

Ví dụ:



$$\begin{aligned} x_1 - 2x_2 &= -1 \\ -x_1 + 3x_2 &= 3 \end{aligned}$$

Khái niệm

Biểu diễn ma trận

Hệ phương trình tuyến tính (1) có thể được viết lại dưới dạng:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Kí hiệu các ma trận

$$\begin{aligned} \mathrm{A} = [a_{ij}]_{m\times n} &: \quad \text{ma trận hệ số của ẩn} \\ \mathrm{x} = [x_j]_{n\times 1} &: \quad \text{ma trận ẩn} \\ \mathrm{b} = [b_i]_{m\times 1} &: \quad \text{ma trận hệ số tự do} \end{aligned}$$

Hệ phương trình (1) có dạng ma trận như sau: $\mathrm{Ax} = \mathrm{b}$.

Khái niệm

Ma trận hệ số mở rộng (augmented matrix)

Ma trận

$$\overline{A} = [A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

được gọi là ma trận hệ số mở rộng của hệ phương trình (1).

Định lý Kronecker-Capelli

Hệ phương trình tuyến tính dạng ma trận: $Ax = b$ có nghiệm khi và chỉ khi:

$$\operatorname{rank} A = \operatorname{rank} \overline{A}$$

Khái niệm

Ví dụ: Xét hệ phương trình $\begin{cases} x_1 + 4x_2 - 2x_3 + 3x_4 = 6 \\ 2x_1 + 7x_2 + x_3 - 4x_4 = 2 \\ 3x_1 + 14x_2 - 4x_3 + 5x_4 = 11 \\ -x_1 - 6x_2 \qquad\quad + x_4 = 5 \end{cases}$. Biến đổi ma trận hệ số mở rộng về dạng bậc thang:

$$\left(\begin{array}{cccc|c} 1 & 4 & -2 & 3 & 6 \\ 2 & 7 & 1 & -4 & 2 \\ 3 & 14 & -4 & 5 & 11 \\ -1 & -6 & 0 & 1 & 5 \end{array}\right) \underset{\substack{R_3 - 3\cdot R_1 \to R_3 \\ R_4 - (-1)\cdot R_1 \to R_4}}{\overset{R_2 - 2\cdot R_1 \to R_2}{\sim}} \left(\begin{array}{cccc|c} 1 & 4 & -2 & 3 & 6 \\ 0 & -1 & 5 & -10 & -10 \\ 0 & 2 & 2 & -4 & -7 \\ 0 & -2 & -2 & 4 & 11 \end{array}\right) \underset{R_4 - 2\cdot R_2 \to R_4}{\overset{R_3 - (-2)\cdot R_2 \to R_3}{\sim}}$$

$$\left(\begin{array}{cccc|c} 1 & 4 & -2 & 3 & 6 \\ 0 & -1 & 5 & -10 & -10 \\ 0 & 0 & 12 & -24 & -27 \\ 0 & 0 & -12 & 24 & 31 \end{array}\right) \underset{R_4 - (-1)\cdot R_3 \to R_4}{\sim} \left(\begin{array}{cccc|c} 1 & 4 & -2 & 3 & 6 \\ 0 & -1 & 5 & -10 & -10 \\ 0 & 0 & 12 & -24 & -27 \\ 0 & 0 & 0 & 0 & 4 \end{array}\right)$$

Vì $\operatorname{rank} A = 3 \neq \operatorname{rank} \overline{A} = 4 \Rightarrow$ hệ phương trình đã cho vô nghiệm.

Phương pháp ma trận nghịch đảo

Ma trận nghịch đảo (matrix inverse)

Ma trận vuông A được gọi là ma trận khả nghịch nếu tồn tại ma trận B sao cho

$$AB = BA = I \quad (\text{I là ma trận đơn vị})$$

Khi đó, B gọi là ma trận nghịch đảo của ma trận A.

- Ma trận nghịch đảo của A nếu tồn tại thì duy nhất, do đó được kí hiệu là A^{-1}.
- Ma trận vuông A là khả nghịch khi và chỉ khi $\det A \neq 0$.
- **Tính chất:**
  1. $(A^{-1})^{-1} = A$
  2. $(AB)^{-1} = B^{-1}A^{-1}$
  3. $(A^{-1})^T = (A^T)^{-1}$
  4. $(cA)^{-1} = \frac{1}{c}A^{-1}$

Phương pháp ma trận nghịch đảo

Biểu diễn của ma trận nghịch đảo

Cho ma trận vuông A là ma trận khả nghịch. Khi đó, ma trận nghịch đảo A^{-1} được xác định bởi:

$$\mathrm{A}^{-1} = \frac{1}{\det \mathrm{A}} \begin{bmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \vdots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{bmatrix}$$

trong đó A_{ij} là phần bù đại số của phần tử a_{ij} của ma trận A.

Nhắc lại: $A_{ij} = (-1)^{i+j} M_{ij}$, với M_{ij} là định thức của ma trận thu được từ A sau khi bỏ đi hàng và cột chứa phần tử a_{ij}.

Phương pháp ma trận nghịch đảo

Ví dụ: Tìm ma trận nghịch đảo của ma trận $A = \begin{pmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{pmatrix}$

$$C_{1,1} = (-1)^{(1+1)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 21 \quad C_{2,1} = (-1)^{(2+1)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 9 \quad C_{3,1} = (-1)^{(3+1)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = -1$$

$$C_{1,2} = (-1)^{(1+2)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = -8 \quad C_{2,2} = (-1)^{(2+2)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 5 \quad C_{3,2} = (-1)^{(3+2)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 6$$

$$C_{1,3} = (-1)^{(1+3)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 22 \quad C_{2,3} = (-1)^{(2+3)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 1 \quad C_{3,3} = (-1)^{(3+3)} \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 13$$

$$|A| = \begin{vmatrix} 1 & -2 & 1 \\ 4 & 5 & -2 \\ -2 & 3 & 3 \end{vmatrix} = 59 \quad \Rightarrow \quad A^{(-1)} = \frac{1}{|A|} \cdot C^T = \frac{1}{59} \cdot \begin{pmatrix} 21 & 9 & -1 \\ -8 & 5 & 6 \\ 22 & 1 & 13 \end{pmatrix}$$

Phương pháp ma trận nghịch đảo

Giải hệ phương trình tuyến tính

Cho hệ phương trình tuyến tính viết dưới dạng ma trận:

$$Ax = b$$

trong đó A, b là các ma trận cho trước, x là ma trận ẩn cần tìm. Nếu ma trận A khả nghịch thì phương trình có nghiệm

$$x = A^{-1}b$$

Ví dụ: Giải hệ phương trình $\begin{cases} 2x_1 + x_2 - 3x_3 = 5 \\ -3x_1 + 2x_2 + 7x_3 = 4 \\ x_1 + 4x_2 + 2x_3 = 6 \end{cases}$

Phương pháp ma trận nghịch đảo

Chuyển hệ phương trình tuyến tính về dạng ma trận

$$\begin{pmatrix} 2 & 1 & -3 \\ -3 & 2 & 7 \\ 1 & 4 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 6 \end{pmatrix}$$

Ma trận hệ số A khả nghịch: $A^{-1} = \begin{pmatrix} \frac{-24}{7} & -2 & \frac{13}{7} \\ \frac{13}{7} & 1 & \frac{-5}{7} \\ -2 & -1 & 1 \end{pmatrix}$

Khi đó, nghiệm của hệ thu được là:

$$x = A^{-1}b = \begin{pmatrix} \frac{-24}{7} & -2 & \frac{13}{7} \\ \frac{13}{7} & 1 & \frac{-5}{7} \\ -2 & -1 & 1 \end{pmatrix} \begin{pmatrix} 5 \\ 4 \\ 6 \end{pmatrix} = \begin{pmatrix} -14 \\ 9 \\ -8 \end{pmatrix}$$

Phương pháp khử Gauss

Phương pháp khử Gauss

Các bước giải hệ phương trình tuyến tính $Ax = b$:

1. Lập ma trận hệ số mở rộng $\overline{A}$
2. Biến đổi $\overline{A}$ thành ma trận bậc thang bởi các phép biến đổi sơ cấp theo hàng.
3. Lập hệ phương trình mới từ ma trận bậc thang.
4. Tìm nghiệm theo hệ bậc thang.

Ví dụ: Giải hệ phương trình $\begin{cases} 2x_1 + x_2 - 3x_3 = 5 \\ -3x_1 + 2x_2 + 7x_3 = 4 \\ x_1 + 4x_2 + 2x_3 = 6 \end{cases}$

Phương pháp khử Gauss

$$\left(\begin{array}{ccc|c} 2 & 1 & -3 & 5 \\ -3 & 2 & 7 & 4 \\ 1 & 4 & 2 & 6 \end{array}\right) \cdot\left(\frac{3}{2}\right) \underset{R_2-\left(\frac{-3}{2}\right)\cdot R_1 \to R_2}{\overset{?}{\sim}} \left(\begin{array}{ccc|c} 2 & 1 & -3 & 5 \\ 0 & \frac{7}{2} & \frac{5}{2} & \frac{23}{2} \\ 1 & 4 & 2 & 6 \end{array}\right) \cdot\left(\frac{-1}{2}\right) \underset{R_3-\left(\frac{1}{2}\right)\cdot R_1 \to R_3}{\overset{?}{\sim}}$$

$$\left(\begin{array}{ccc|c} 2 & 1 & -3 & 5 \\ 0 & \frac{7}{2} & \frac{5}{2} & \frac{23}{2} \\ 0 & \frac{7}{2} & \frac{7}{2} & \frac{7}{2} \end{array}\right) \cdot(-1) \underset{R_3-1\cdot R_2 \to R_3}{\overset{?}{\sim}} \left(\begin{array}{ccc|c} 2 & 1 & -3 & 5 \\ 0 & \frac{7}{2} & \frac{5}{2} & \frac{23}{2} \\ 0 & 0 & 1 & -8 \end{array}\right)$$

$$\begin{cases} 2\cdot x_1 + x_2 - 3\cdot x_3 = 5 \\ \frac{7}{2}\cdot x_2 + \frac{5}{2}\cdot x_3 = \frac{23}{2} \\ x_3 = -8 \end{cases}$$

Hệ phương trình có nghiệm: $X = \begin{pmatrix} -14 \\ 9 \\ -8 \end{pmatrix}$

Phương pháp phân tích LU

Phân tích LU (LU decomposition)

Giả sử A làm một ma trận kích thước $m \times n$ có thể biến đổi về dạng bậc thang mà không cần đổi hàng. Khi đó ma trận A có thể viết dưới dạng $A = LU$, ở đó L là ma trận tam giác dưới cỡ $m \times m$ với các phần tử đường chéo bằng 1 và U là ma trận dạng bậc thang của ma trận A.

$$A = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} \blacksquare & * & * & * & * \\ 0 & \blacksquare & * & * & * \\ 0 & 0 & 0 & \blacksquare & * \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{U}$$

Figure 1: Minh họa phân tích LU một ma trận

Phương pháp phân tích LU

Thuật toán phân tích LU

1. Biến đổi A thành dạng bậc thang U bằng biến đổi sơ cấp theo hàng.
2. Đặt L sao cho các biến đổi trên biến L thành ma trận đơn vị I.

Ví dụ: Tìm phân tích LU của ma trận $A = \begin{pmatrix} 2 & 4 & -1 & 5 & -2 \\ -4 & -5 & 3 & -8 & 1 \\ 2 & -5 & -4 & 1 & 8 \\ -6 & 0 & 7 & -3 & 1 \end{pmatrix}$

$$A = \begin{bmatrix} 2 & 4 & -1 & 5 & -2 \\ -4 & -5 & 3 & -8 & 1 \\ 2 & -5 & -4 & 1 & 8 \\ -6 & 0 & 7 & -3 & 1 \end{bmatrix} \sim \begin{bmatrix} 2 & 4 & -1 & 5 & -2 \\ 0 & 3 & 1 & 2 & -3 \\ 0 & -9 & -3 & -4 & 10 \\ 0 & 12 & 4 & 12 & -5 \end{bmatrix} = A_1$$

$$\sim A_2 = \begin{bmatrix} 2 & 4 & -1 & 5 & -2 \\ 0 & 3 & 1 & 2 & -3 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 4 & 7 \end{bmatrix} \sim \begin{bmatrix} 2 & 4 & -1 & 5 & -2 \\ 0 & 3 & 1 & 2 & -3 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix} = U$$

Phương pháp phân tích LU

Ma trận L được tạo thành từ các cột chứa phần tử trụ trong quá trình biến đổi của A (các cột tô màu xanh), sau khi đã chia các phần tử trên cột cho phần tử trụ tương ứng:

$$\begin{bmatrix} 2 \\ -4 \\ 2 \\ -6 \end{bmatrix} \quad \begin{bmatrix} 3 \\ -9 \\ 12 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 4 \end{bmatrix} \quad \begin{bmatrix} 5 \end{bmatrix}$$

$$\div 2 \downarrow \qquad \div 3 \downarrow \qquad \div 2 \downarrow \qquad \div 5 \downarrow$$

$$\begin{bmatrix} 1 & & & \\ -2 & 1 & & \\ 1 & -3 & 1 & \\ -3 & 4 & 2 & 1 \end{bmatrix} \Rightarrow L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ 1 & -3 & 1 & 0 \\ -3 & 4 & 2 & 1 \end{bmatrix}$$

Phương pháp phân tích LU

Giải hệ phương trình bằng phân tích LU

Xét hệ phương trình tuyến tình dạng ma trận $\mathrm{A}\mathrm{x} = \mathrm{b}$. Nếu có phân tích $\mathrm{A} = LU$, hệ phương trình được viết lại thành $L(U\mathrm{x}) = \mathrm{b}$. Đặt $\mathrm{y} = U\mathrm{x}$, ta có thể tìm x bằng cách giải cặp hệ phương trình:

$$L\mathrm{y} = \mathrm{b}$$
$$U\mathrm{x} = \mathrm{y}$$

Ví dụ: Giải hệ phương trình $\begin{cases} 2x_1 + x_2 - 3x_3 = 5 \\ -3x_1 + 2x_2 + 7x_3 = 4 \\ x_1 + 4x_2 + 2x_3 = 6 \end{cases}$

Phương pháp phân tích LU

Đầu tiên, tìm phân tích LU của ma trận hệ số:

$$\begin{pmatrix} 2 & 1 & -3 \\ -3 & 2 & 7 \\ 1 & 4 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{-3}{2} & 1 & 0 \\ \frac{1}{2} & 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & -3 \\ 0 & \frac{7}{2} & \frac{5}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

Tiếp theo, giải hệ tam giác dưới:

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{-3}{2} & 1 & 0 \\ \frac{1}{2} & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 5 \\ 4 \\ 6 \end{pmatrix} \Rightarrow \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 5 \\ \frac{23}{2} \\ -8 \end{pmatrix}$$

Cuối cùng, giải hệ tam giác trên:

$$\begin{pmatrix} 2 & 1 & -3 \\ 0 & \frac{7}{2} & \frac{5}{2} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 5 \\ \frac{23}{2} \\ -8 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -14 \\ 9 \\ -8 \end{pmatrix}$$

Phương pháp lặp giải hệ phương trình

Xét hệ phương trình tuyến tính: $Ax = b$

Phân tích ma trận: $A = L + D + U$ (lower, diagonal, upper)

Biến đổi hệ phương trình về dạng: $Dx = -(L + U)x + b$

Một số phép lặp

- **Jacobi**: $x^{(k+1)} = D^{-1}[-(L + U)x^{(k)} + b]$
- **Gauss-Seidel**: $x^{(k+1)} = D^{-1}[-Lx^{(k+1)} - Ux^{(k)} + b]$
- **Successive-Over-Relaxation**:

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}[-Lx^{(k+1)} - (D + U)x^{(k)} + b]$$

Ví dụ giải hệ bằng phương pháp lặp Jacobi

$$\begin{pmatrix} 5 & 1 & -3 \\ 1 & 4 & 2 \\ -3 & 2 & 7 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -37 \\ 6 \\ 4 \end{pmatrix}$$

```
Xuất phát: [1 1 1]
Lần lặp 10: [-13.52363217   8.54455019  -7.55137783]
Lần lặp 20: [-13.97267975   8.9739238   -7.97448395]
Lần lặp 30: [-13.99843898   8.99851015  -7.99854251]
Lần lặp 40: [-13.99991082   8.99991489  -7.99991673]
Lần lặp 50: [-13.99999491   8.99999514  -7.99999524]
Lần lặp 60: [-13.99999971   8.99999972  -7.99999973]
Lần lặp 70: [-13.99999998   8.99999998  -7.99999998]
Lần lặp 80: [-14.   9.  -8.]
Lần lặp 90: [-14.   9.  -8.]
```

Thực hành giải hệ phương trình tuyến tính

- **np.linalg.inv**(A) : trả về ma trận nghịch đảo của ma trận vuông A

```
In [1]: import numpy as np
        A=np.array([[1,-2,3],[2,-5,12],[0,2,-10]])
        print("detA=",np.linalg.det(A))
        print("inv(A)=",np.linalg.inv(A))

        detA= -2.0
        inv(A)= [[-13.    7.    4.5]
         [-10.    5.    3. ]
         [ -2.    1.    0.5]]
```

- **np.linalg.solve**(A, b) : trả về nghiệm của phương trình $Ax = b$

```
In [2]: import numpy as np
        A=np.array([[1,-2,3],[2,-5,12],[0,2,-10]])
        b=np.array([4,5,6])
        np.linalg.solve(A,b)

Out[2]: array([10.,  3., -0.])
```

Thực hành giải hệ phương trình tuyến tính

- **scipy.linalg.lu**(A) : trả về các ma trận P, L, U trong phân tích LU của ma trận A

```
In [3]: import numpy as np
        from scipy import linalg
        A=np.array([[1,-2,3],[2,-5,12],[0,2,-10]])
        P,L,U=linalg.lu(A)
        B=np.dot(P,np.dot(L,U))
        print(P,L,U,sep="\n")
        print("P*L*U=",B)
        np.allclose(A,B)

        [[0. 0. 1.]    [[1.   0.   0.  ]
         [1. 0. 0.]     [0.   1.   0.  ]
         [0. 1. 0.]]    [0.5  0.25 1.  ]]
        [[  2.   -5.   12. ]
         [  0.    2.  -10. ]
         [  0.    0.   -0.5]]
        P*L*U= [[  1.  -2.   3.]
         [  2.  -5.  12.]
         [  0.   2. -10.]]

Out[3]: True
```

Thực hành giải hệ phương trình tuyến tính

- **scipy.linalg.lu_factor**(A) : trả về ma trận lu (ghép 2 ma trận L và U, bỏ qua đường chéo L), piv (mô tả hoán vị hàng) trong phân tích LU của ma trận A
- **scipy.linalg.lu_solve**((lu,piv),b) : giải hệ phương trình bằng phân tích LU

```
In [4]: import numpy as np
        from scipy import linalg
        A=np.array([[1,-2,3],[2,-5,12],[0,2,-10]])
        b=np.array([4,5,6])
        lu,piv=linalg.lu_factor(A)
        x=linalg.lu_solve((lu,piv),b)
        print(lu,piv,x,sep="\n")

[[  2.    -5.    12.  ]
 [  0.     2.   -10.  ]
 [  0.5    0.25  -0.5 ]]
[1 2 2]
[10.  3. -0.]
```

Bài toán hồi quy (Regression)

- Nhiều bài toán trong các lĩnh vực khoa học, kĩ thuật liên quan đến việc xác định quan hệ giữa các biến, chẳng hạn:
    1. Thu nhập y của một kĩ sư phụ thuộc vào trình độ học vấn x_1 và số năm kinh nghiệm x_2.
    2. Giá nhà y phụ thuộc vào diện tích x_1, số phòng ngủ x_2, và khoảng cách đến khu vực trung tâm x_3.
    3. Cường độ chịu nén của đất y phụ thuộc vào địa điểm khảo sát (x_1, x_2) và độ sâu x_3 tính từ mặt đất.
- Xây dựng một mô hình mô tả mối quan hệ giữa biến phụ thuộc y và một hay nhiều biến độc lập $x_1, x_2, \ldots, x_k$:

$$y_n = f(\mathrm{x_n}; \beta) + \epsilon_n$$

với $\mathrm{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nk})^T$, y_n là giá trị của các biến tại quan sát thứ n, $\epsilon_n \sim N(0, \sigma^2)$ mô tả sai số ngẫu nhiên, và β là véc tơ tham số của mô hình cần xác định.

Mô hình hồi quy tuyến tính (Linear Regression Model)

- Mô hình đơn giản nhất là mô hình hồi quy tuyến tính:

$$y_n = \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_k x_{nk} + \epsilon_n = \mathrm{x}_n^T \beta + \epsilon_n$$

- Tập dữ liệu quan sát: $\{y_n, x_{n1}, x_{n2}, \ldots, x_{nk}\}$ với $n = 1, 2, \ldots, N$
- Mô hình cho n quan sát có thể được viết lại dưới dạng sau:

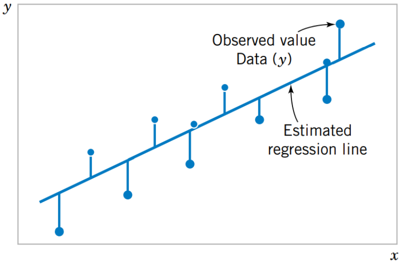
$$\mathrm{y} = \mathrm{X}\beta + \epsilon$$

trong đó $\mathrm{y} = (y_1, y_2, \ldots, y_N)^T$, $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_N)^T$ và

$$\mathrm{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Nk} \end{pmatrix}$$

Mô hình hồi quy tuyến tính

- Rõ ràng ta không thể xác định chính xác véc tơ tham số β của mô hình. Do đó, ta tìm một ước lượng B của nó. Khi đó, giá trị quan sát y_n có thể được xấp xỉ bởi mô hình:
$$y_n \approx \mathrm{x}_n^T \mathrm{B}$$
- Sai số dự báo của mô hình: $e_n = y_n - \mathrm{x}_n^T \mathrm{B}$.



Mô hình hồi quy tuyến tính

- Ước lượng B làm cực tiểu hóa hàm tổng bình phương sai số:

$$SSE = \|\mathrm{y} - \mathrm{X}\,\mathrm{B}\|^2 = \sum_{n=1}^{N} \left(y_n - \mathrm{x}_n^T\,\mathrm{B} \right)^2$$

- Khi đó, ước lượng B là nghiệm của hệ phương trình:

$$(\mathrm{X}^T\mathrm{X})\,\mathrm{B} = \mathrm{X}^T\mathrm{y}$$

Nếu ma trận $(\mathrm{X}^T\mathrm{X})$ khả nghịch thì $\mathrm{B} = (\mathrm{X}^T\mathrm{X})^{-1}\,\mathrm{X}^T\mathrm{y}$.

Mô hình hồi quy tuyến tính

Ví dụ: Xây dựng mô hình hồi quy tuyến tính đơn cho dữ liệu về số năm kinh nghiệm và thu nhập dưới đây:

Số năm KN	1.1	1.3	1.5	2.0	2.2
Thu nhập	39343	46205	37731	43525	39891

$$X = \begin{pmatrix} 1 & 1.1 \\ 1 & 1.3 \\ 1 & 1.5 \\ 1 & 2.0 \\ 1 & 2.2 \end{pmatrix}, \quad y = \begin{pmatrix} 39343 \\ 46205 \\ 37731 \\ 43525 \\ 39891 \end{pmatrix} \Rightarrow \begin{pmatrix} 5 & 8.1 \\ 8.1 & 13.99 \end{pmatrix} B = \begin{pmatrix} 206695 \\ 334750.5 \end{pmatrix}$$

$$\Rightarrow B = \begin{pmatrix} 41517.05 \\ -109.91 \end{pmatrix}$$

Đường hồi quy tuyến tính có dạng: $y = 41517.05 - 109.91\,x$

VINBIGDATA VINGROUP AI Academy Vietnam

Thực hành giải bài toán hồi quy tuyến tính

Nghiệm bình phương tối thiểu (least squares solution)

Trong một số trường hợp, hệ phương trình tuyến tính $Ax = b$ vô nghiệm. Ta muốn tìm một véc tơ x sao cho Ax "gần b nhất" có thể. Véc tơ x được gọi là nghiệm bình phương tối thiểu của hệ phương trình nếu bình phương sai số $\|Ax - b\|^2$ đạt giá trị nhỏ nhất. Khi đó nghiệm bình phương tối thiểu thỏa mãn phương trình:

$$(A^T A)x = A^T b$$

Nếu $(A^T A)$ khả nghịch thì công thức nghiệm bình phương tối thiểu là:

$$x = (A^T A)^{-1} A^T b$$

Thực hành giải bài toán hồi quy tuyến tính

- **np.linalg.lstsq**(A, b) : trả về nghiệm bình phương tối thiểu của hệ phương trình $Ax = b$, hạng và các giá trị kì dị của ma trận A

```
In [5]: import numpy as np
        A=np.array([[1,-2,3,1],[2,-5,12,4],[0,2,-10,1]])
        b=np.array([4,5,6])
        np.linalg.lstsq(A,b,rcond=None)

Out[5]: (array([ 1.80053908, -2.46630728, -1.05121294,  0.42048518]),
         array([], dtype=float64),
         3,
         array([17.12877864,  3.88620317,  0.70877874]))
```

- Hàm **lstsq**() khi áp dụng để giải hệ phương trình có vô số nghiệm sẽ trả về kết quả là nghiệm bình phương tối thiểu có độ lớn nhỏ nhất.

Thực hành giải bài toán hồi quy tuyến tính

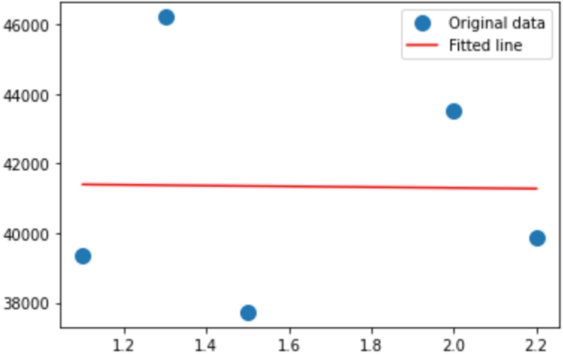
Xét bài toán hồi quy cho dữ liệu dưới đây:

Số năm KN	1.1	1.3	1.5	2.0	2.2
Thu nhập	39343	46205	37731	43525	39891

```
In [6]: import numpy as np
        import matplotlib.pyplot as plt

        # Dữ liệu
        x=np.array([1.1,1.3,1.5,2.0,2.2])
        y=np.array([39343,46205,37731,43525,39891])

        # Nghiệm bình phương tối thiểu
        A=np.vstack([np.ones(len(x)),x]).T
        b0,b1=np.linalg.lstsq(A,y,rcond=None)[0]
        print("y=%f+%f x"%(b0,b1))

        # Vẽ đồ thị
        _=plt.plot(x,y,'o',label='Original data',markersize=10)
        _=plt.plot(x,b0+b1*x,'r',label='Fitted line')
        _=plt.legend()
        plt.show()
```

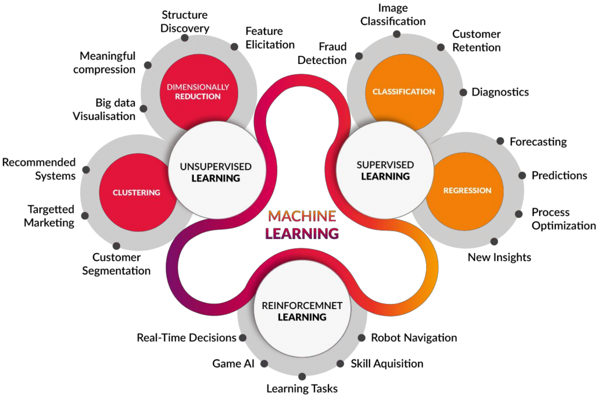
Thực hành giải bài toán hồi quy tuyến tính

```
y=41517.050691+-109.907834 x
```



Giới thiệu về **scikit-learn**

- **Scikit-Learn** là một thư viện của Python cung cấp các công cụ hoàn chỉnh để thực thi các thuật toán thông dụng trong học máy.

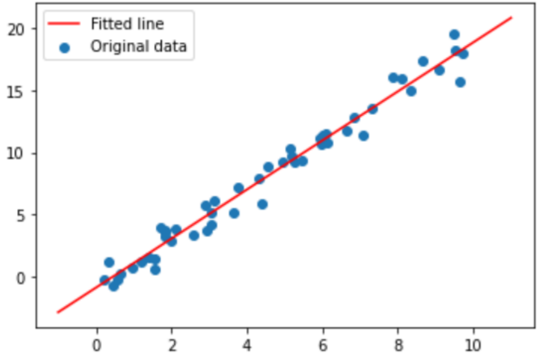


Hồi quy tuyến tính với **scikit-learn**

In [7]:
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression

# Tạo dữ liệu
rng = np.random.RandomState(42)
x = 10*rng.rand(50)
y = 2*x-1+rng.randn(50)

# Xây dựng mô hình
model = LinearRegression(fit_intercept=True)
X = x[:, np.newaxis]
model.fit(X, y)
print("Mô hình: y=%f+%fx"%(model.intercept_,model.coef_[0]))

# Vẽ đồ thị
xfit = np.linspace(-1, 11)
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
_=plt.scatter(x,y,label='Original data')
_=plt.plot(xfit,yfit,'r',label='Fitted line')
_=plt.legend()
plt.show()
```

Hồi quy tuyến tính với **scikit-learn**

```
Mô hình: y=-0.903311+1.977657x
```



Tài liệu tham khảo

1. Charu C. Aggarwal; Linear Algebra and Optimization for Machine Learning, Springer, 2020.
2. Stephen Boyd, Lieven Vandenberghe; Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares, Cambridge University Press, 2018.
3. David C. Lay, Steven R. Lay, Judi J. McDonald; Linear Algebra and Its Applications, Fifth edition, Pearson, 2016
4. Tom Lyche; Numerical Linear Algebra and Matrix Factorizations, Springer, 2020.
5. Gilbert Strang; Linear Algebra and Learning from Data, Wellesley- Cambridge Press, 2019.