# Toán học cho khoa học dữ liệu

## Mathematics for Data Science

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Nội dung

- Một số vấn đề nền tảng toán học được sử dụng
- Toán học cho học máy
- Nền tảng cơ bản của mạng lưới thần kinh nhân tạo

# Đại số tuyến tính

## Linear Algebra

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Vector

- Đại số tuyến tính nhằm nghiên cứu về vector và các thao tác trên vector, trong lĩnh vực Khoa học máy tính cũng như Khoa học dữ liệu, vector là một mảng một chiều của các đại lượng vô hướng có giá trị thực sự.
- Còn trong Toán học, vector là một đại lượng có cả độ lớn và hướng, được biểu thị bằng một mũi tên biểu thị hướng và độ dài (length) của nó tỷ lệ thuận với độ lớn (magnitude). Vector ở đây xuất phát từ vector hình học qua hình tượng có dấu mũi tên $\rightarrow$ bên trên, chẳng hạn, vector $\overrightarrow{AB}$

- Trong thực tế có rất nhiều khái niệm có thể diễn giải thành vector; chẳng hạn
  - **Vector hình học**: Cho 2 vector loại hình học $\overrightarrow{x}, \overrightarrow{y}$ và đại lượng vô hướng $\lambda \in \mathbb{R}$, khi đó ta có thể tạo ra
    - vector tổng $\overrightarrow{z} = \overrightarrow{x} + \overrightarrow{y}$
    - vector gấp $\lambda$ lần $\overrightarrow{u} = \lambda \overrightarrow{x}$
      - Khi đó $\overrightarrow{z}, \overrightarrow{u}$ cũng đều là vector loại hình học

- **Đa thức vector**: Cho đa thức bậc $n$ với một biến độc lập $x$. Ký hiệu
  $P(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{n-1}x^{n-1} + a_nx^n$

  - Đa thức này có thể được coi là *đa thức vector* khi ta biểu diễn $P(x)$ dưới dạng $n+1$ thành phần $a_0, a_1, \ldots, a_{n-1}, a_n$
  - Cho thêm đa thức $Q(x)$ có các thành phần là $b_0, b_1, \ldots, b_{n-1}, b_n$. Từ đó, ta có thể tạo ra đa thức vector $R(x) = P(x) + Q(x)$ mới bằng cách cộng 2 đa thức vector theo quy ước là các thành phần được cộng với nhau là
    $a_0 + b_0, a_1 + b_1, \ldots, a_{n-1} + b_{n-1}, a_n + b_n$
  - Tương tự với việc nhân với một vô hướng $\lambda \in \mathbb{R}$ để có đa thức $T(x) = \lambda P(x)$ với các thành phần $\lambda a_0, \lambda a_1, \ldots, \lambda a_{n-1}, \lambda a_n$

- **Vector tín hiệu âm thanh**: Một tín hiệu âm thanh có thể biểu diễn bởi một chuỗi các con số.
  - Khi đó ta có thể cộng 2 tín hiệu, nhân tín hiệu lên một đại lượng nào đó
  - Kết quả tín hiệu cộng hay tín hiệu nhân cũng là vector tín hiệu âm thanh
- **Phần tử của $\mathbb{R}^n$ là vector có $n$ số thực** $\mathbb{R}$: Cho $n$ số thực $x_1, x_2, \ldots, x_n \in \mathbb{R}$. Ký hiệu vector trong trường hợp này là $x = (x_1, x_2, \ldots, x_n)$, hoặc $x = [x_1 \ x_2 \ \ldots \ x_n]^T$
- Trong tin học, khi xử lý dữ liệu số, chúng ta sử dụng vector các số thực như trên, nên đại lượng vô hướng ở đây là số thực.

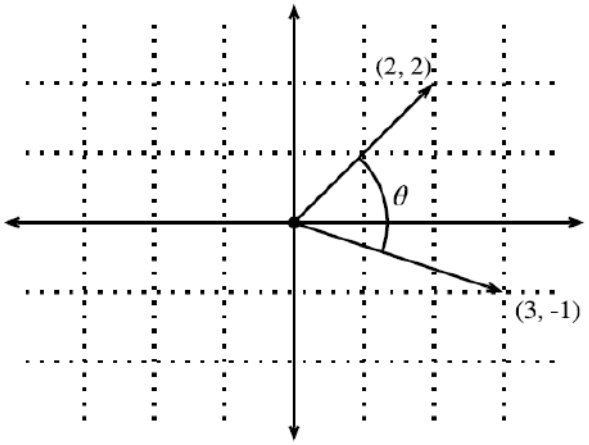
# Tích vô hướng

- **Tích vô hướng**: Cho 2 vector $u = (u_1, u_2, \ldots, u_n)$ và $v = (v_1, v_2, \ldots, v_n)$ được ký hiệu là

$$u \,.\, v = u^T v = \langle u, v \rangle = \sum_{i=1}^{n} u_i v_i$$

- Ý nghĩa về mặt hình học của tích vô hướng như hình
- Khi đó $\langle u, v \rangle = \|u\| \|v\| \cos\theta$, với $\|.\|$ là chiều dài của vector.
  - Khi $\theta = 90^0$, nghĩa là 2 vector $u, v$ vuông góc với nhau thì $\cos\theta = 0$, nên $\langle u, v \rangle = 0$

```
# Minh hoạ bằng Python
import numpy as np

u = np.array([1, 2, 3])
v = np.array([4, 5, 6])
np.dot(u, v)
```



# Ma trận (Matrix)

- Ma trận đóng vai trò trọng tâm của Đại số tuyến tính. Chúng ta có thể dùng để biểu diễn một hệ phương trình (system of linear equations) hoặc một hàm hay một ánh xạ tuyến tính (linear function/linear mapping)
- **Định nghĩa ma trận**: Ma trận $A$ với $m, n$ phần tử là một dãy gồm $m \times n$ phần tử $a_{ij}, i = 1,...,m, j = 1,...,n$ để tạo thành một hình chữ nhật gồm $m$ dòng và $n$ cột như sau:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, a_{ij} \in \mathbb{R}$$

- Như vậy, thực chất ma trận $A$ là một vector thuộc $\mathbb{R}^{m\times n}$.
- Nên cũng có thể cộng 2 ma trận và nhân ma trận với một con số với quy ước

$$A+B=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{bmatrix}$$

và $$\lambda A=\begin{bmatrix} \lambda a_{11} & \lambda a_{12} & \dots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & \lambda a_{2n} \\ \vdots & \vdots & & \vdots \\ \lambda a_{m1} & \lambda a_{m2} & \dots & \lambda a_{mn} \end{bmatrix}$$

- Ngoài ra, cũng có thể định nghĩa thêm phép nhân 2 ma trận như sau:
- Cho $A$ là ma trận $m, n$ phần tử; $B$ là ma trận có $n, k$ phần tử. Phần tử $c_{ij}$ của ma trận tích $C = AB$ được tính bởi: $c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj},\ i = \overline{1,m},\ j = \overline{i,k}$
- Ví dụ: Cho $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 3}, B = \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 3 \end{bmatrix} \in \mathbb{R}^{3\times 2}$
- Suy ra $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 5 & 7 \\ 3 & 5 \end{bmatrix} \in \mathbb{R}^{2\times 2}$

# Python

- Bằng Python, dùng thư viện numpy có thể viết như sau để tính $C = AB$

```
import numpy as np
# Cho 2 ma trận A, B
A = np.array( [[1,2,3],[3,2,1]] )
B = np.array( [[0,1],[1,0],[1,2]] )

# Tích C = AB
C = A.dot(B)

# Xuất các ma trận
print( "A =", A )
print( "B =", B )
print( "C =", C )
```

- **Định nghĩa ma trận đơn vị**: Trong $\mathbb{R}^{n\times n}$, ma trận đơn vị (Identity matrix) hay ma trận đồng nhất $\mathbf{I}_n$ là ma trận có dạng $\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}$

- Trong numpy của Python, ma trận đơn vị cấp $n$ có thể tạo bằng lệnh np.eye(n)

- **Định nghĩa ma trận nghịch đảo**: Cho ma trận vuông $A \in \mathbb{R}^{n\times n}$. Giả sử ma trận vuông $B \in \mathbb{R}^{n\times n}$ có tính chất $AB = \mathbf{I}_n = BA$. Thì $B$ được gọi mà ma trận nghịch đảo (Inverse matrix) của ma trận $A$. Khi đó ký hiệu là $A^{-1}$.

# Python

- Bằng Python, dùng thư viện numpy có thể viết như sau để tìm nghịch đảo của ma trận

```
import numpy as np

# Cho ma trận A
A = np.array( [[1,2,1],[4,4,5],[6,7,7]] )

# B là nghịch đảo của A
B = np.linalg.inv(A)

print( "Inverse Matrix of A = ", B )
```

- **Định nghĩa ma trận chuyển vị**: Cho ma trận $A \in \mathbb{R}^{m \times n}$. Ma trận $B \in \mathbb{R}^{n \times m}$ với $b_{ij} = a_{ji}, \forall i = \overline{1,n},\ j = \overline{i,m}$ được gọi là ma trận chuyển vị (Transpose Matrix) của ma trận $A$ . Ký hiệu $B = A^T$.
- **Tính chất**:
  - $AA^{-1} = \mathbf{I} = A^{-1}A$
  - $(AB)^{-1} = B^{-1}A^{-1}$
  - $(A^T)^T = A$
  - $(A + B)^T = A^T + B^T$
  - $(AB)^T = B^T A^T$

- **Định nghĩa ma trận đối xứng**: Ma trận $A \in \mathbb{R}^{n \times n}$ được gọi là ma trận đối xứng (Symmetric Matrix) nếu như $A = A^T$

# Không gian vector (Vector Space)

- Để hiểu về Không gian vector (*Vector Space*) trước hết ta cần biết về khái niệm ***nhóm***
- **Định nghĩa nhóm (*Group*):** Cho tập hợp $\mathcal{G}$ và phép toán $\oplus : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ được định nghĩa trên $\mathcal{G}$. Thì $G := (\mathcal{G}, \oplus)$ được gọi là ***nhóm*** nếu thoả các điều kiện sau:
  - $\forall x, y \in \mathcal{G} : x \oplus y \in \mathcal{G}$
  - $\forall x, y, z \in \mathcal{G} : (x \oplus y) \oplus z = x \oplus (y \oplus z)$
  - $\exists e \in \mathcal{G}, \forall x \in \mathcal{G} : x \oplus e = x$ và $e \oplus x = x$ ($e$ gọi là phần tử trung tính)
  - $\forall x \in \mathcal{G} : \exists y \in \mathcal{G} \big/ x \oplus y = e$ và $y \oplus x = e$ (với $e$ là phần tử trung tính và $y$ thường ký hiệu là $x^{-1}$)

- **Định nghĩa nhóm Abel (*Abelian Group*)**: Là nhóm mà có tính giao hoán, nghĩa là:
  $\forall x, y \in \mathcal{G} : x \oplus y = y \oplus x$

- Phép toán "cộng" $\oplus$ trong ánh xạ $\oplus : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ như trên gọi là phép toán trong (*inner operator*) của Nhóm $G := (\mathcal{G}, \oplus)$; ngoài ra có thể bổ sung thêm phép toán "nhân" $\odot$ gọi là phép toán ngoài (*outer operator*) để có thể tác động với một phần tử không thuộc tập hợp $\mathcal{G}$ (chẳng hạn là phần tử thuộc tập $\mathbb{K}$) để
  $\forall \lambda \in \mathbb{K}, \forall x \in \mathcal{G} : \lambda \odot x \in \mathcal{G}$

- **Định nghĩa không gian vector (Vector Space)**: Không gian vector $V$ là một nhóm $V := (\mathscr{V}, \oplus)$ với tập hợp $\mathscr{V}$ với 2 phép toán
  - $\oplus : \mathscr{V} \times \mathscr{V} \to \mathscr{V}$
  - $\odot : \mathbb{K} \times \mathscr{G} \to \mathscr{G}$
- Trong đó
  - $V := (\mathscr{V}, \oplus)$ và $(\mathbb{K}, \odot)$ là nhóm Abel
  - $\forall \lambda \in \mathbb{K}, \forall x, y \in \mathscr{V} : \lambda \odot (x \oplus y) = (\lambda \odot x) \oplus (\lambda \odot y)$
  - $\forall \lambda, \psi \in \mathbb{K}, \forall x \in \mathscr{V} : (\lambda \odot \psi) \oplus x = (\lambda \odot x) \oplus (\psi \odot y)$
  - $\forall \lambda, \psi \in \mathbb{K}, x \in \mathscr{V} : \lambda \odot (\psi \odot x) = (\lambda \odot \psi) \odot x$

- **Ví dụ**: Tập các số thực với phép toán cộng $\mathbb{R}^n, n \in \mathbb{N}$ là ***không gian vector*** với phép toán cộng và nhân được định nghĩa như sau:
  - Với $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ và $y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$, thì
    $x + y = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$
  - Và $\lambda \in \mathbb{R}, x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, thì $\lambda x = (\lambda x_1, \lambda x_2, \ldots, \lambda x_n)$

- **Ví dụ**: Tập hợp ma trận gồm $m$ dòng và $n$ cột các số thực $\mathbb{R}^{m\times n}, m, n \in \mathbb{N}$ là một không gian vector với phép toán cộng 2 ma trận:

- Với $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix}$, thì

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b12 & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}, \forall A, B \in \mathbb{R}^{m\times n}$$

- Và phép nhân một số thực với ma trận được định nghĩa như sau:

$$\lambda A = \begin{bmatrix} \lambda a_{11} & \lambda a_{12} & \dots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & \lambda a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \lambda a_{m1} & \lambda a_{m2} & \dots & \lambda a_{mn} \end{bmatrix}, \forall \lambda \in \mathbb{R}, A \in \mathbb{R}^{m \times n}$$

- Như vậy, không gian vector là một nhóm Abel với phép toán bên trong nhóm và phép toán bên ngoài nhóm thoả mãn 2 đặc điểm:
  - Có thể cộng chúng lại với nhau để được một vector có cùng loại
  - Có thể nhân với một đại lượng vô hướng để có cùng loại

# Sử dụng ma trận

- **Biểu diễn hệ phương trình**: có thể biểu diễn một hệ phương trình bằng các ký hiệu ma trận và vector như sau:
  - Cho hệ gồm $n$ phương trình và $n$ ẩn số $\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ & \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{cases}$
  - Tập hợp các hệ số $a_{ij}$ thành vector, ta được: $\begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} x_2 + \cdots + \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix} x_n = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$

- Khi đó tiếp tục gom các vector này thành vector mà mỗi thành phần là vector (là ma trận)

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

- Ký hiệu lại, ta có hệ $Ax = b$ với $A$ ký hiệu ma trận như định nghĩa, còn $x = (x_1, x_2, \dots, x_n)$ và $b = (b_1, b_2, \dots, b_n)$

- Lưu ý, do vector $n$ thành phần là ma trận $n \times 1$, nên ta có thể viết $x = [x_1\ x_2\ \dots x_n]^T$ thay cho cách viết ở trên thể hiện đó là vector với $n$ thành phần. Khi đó, có thể lý giải hệ ma trận trên theo định nghĩa nhân ma trận.

# Hệ phương trình

- Hệ phương trình đóng vai trò như một phần trung tâm của đại số tuyến tính, nhiều bài toán thực tế có thể biểu diễn như một hệ phương trình tuyến tính.
- **Ví dụ**: Một công ty sản xuất $n$ sản phẩm sử dụng $m$ nguồn tài nguyên $b_1, b_2, \ldots, b_m$. Với giả thiết rằng để sản xuất được một đơn vị $x_j$ của sản phẩm $N_j, j = \overline{1,n}$ thì công ty cần $a_{ij}$ đơn vị từ tài nguyên $b_i, i = \overline{1,m}$.
- Như vậy, với tài nguyên $b_i$, ta có $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i$
- Bài toán đưa về tìm các $x_j, j = \overline{1,n}$ từ các phương trình (hệ phương trình)

  $a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i, \forall i = \overline{1,m}$

- Như vậy một kế hoạch sản xuất tối ưu $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ phải thoả mãn hệ phương trình

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

  - Trong đó, $a_{ij}, b_i \in \mathbb{R}$

- Phương trình trên gọi là dạng tổng quát của hệ phương trình tuyến tính với $x_1, x_2, \ldots, x_n$ là ẩn số của hệ, còn vector $(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ thoả phương trình gọi là nghiệm hay lời giải của hệ phương trình đại số tuyến tính.

- Dùng Python để giải hệ phương trình
- **Ví dụ**: cho hệ phương trình $\begin{cases} x_1 + 8x_3 - 4x_4 = 42 \\ x_2 + 2x_3 + 12x_4 = 8 \end{cases}$
- Đưa về dạng ma trận và vector là $\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 42 \\ 8 \end{bmatrix}$
- Hệ phương trình chỉ có nghiệm duy nhất khi số phương trình bằng số ẩn số.

- **Ví dụ**: Giải hệ phương trình: $\begin{cases} 3x_1 + x_2 + 4x_3 - 5x_4 = -10 \\ x_1 - 2x_2 + 3x_3 + x_4 = -2 \\ 2x_1 + 4x_2 - x_3 + x_4 = 9 \\ x_1 + 2x_3 + 3x_4 = 5 \end{cases}$

- Chuyển về dạng ma trận: $\begin{bmatrix} 3 & 1 & 4 & -5 \\ 1 & -2 & 3 & 1 \\ 2 & 4 & -1 & 1 \\ 1 & 0 & 2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -10 \\ -2 \\ 9 \\ 5 \end{bmatrix}$ sau đó dùng thư viện Python để giải

- Dùng thư viện Numpy

```
import numpy as np

A = np.array([[3, 1, 4, -5],
              [1, -2, 3, 1],
              [2, 4, -1, 1],
              [1, 0, 2, 3]])

b = np.array([-10, -2, 9, 5])

x = np.linalg.solve(A, b)

print("Nghiệm của hệ phương trình:", x)
```

# Hình học giải tích

## Analytic Geometry

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Khái niệm chuẩn

- **Định nghĩa chuẩn trong một không gian vector *V*:** Chuẩn (Norm) là khái niệm để đo độ dài của một vector, đó là hàm

$$\|.\| : V \to \mathbb{R}$$
$$x \mapsto \|x\|$$

- Sao cho $\forall \lambda \in \mathbb{R}$ và $\forall x, y \in V$ có các tính chất sau:

  - $\|\lambda x\| = |\lambda| \|x\|$
  - $\|x + y\| \leq \|x\| + \|y\|$
  - $\|x\| = 0 \iff x = 0$

- $\|x\| \in \mathbb{R}$ còn được gọi độ dài (chiều dài) của vector $x$

- **Định nghĩa chuẩn Manhattan**: $\forall x \in \mathbb{R}^n$, chuẩn Manhattan của $x$ là $\|x\|_1 := \sum_{i=1}^{n} |x_i|$.

  Chuẩn Manhattan còn được gọi là chuẩn $l_1$ $(l_1 - norm)$

- **Chuẩn Euclid (Euclidean Norm)**:

  $\|x\|_2 := \sqrt{\sum_{i=1}^{n} x_i^2} = \sqrt{x^T x}$. Chuẩn Euclide còn gọi là chuẩn $l_2$ $(l_2 - norm)$

- Ngoài ra còn có:
  - Chuẩn $l_p$:

    $\|u\|_p = \left(\sum_{i=1}^{n} |u_i|^p\right)^{\frac{1}{p}}$

  - Chuẩn $l_\infty$:

    $\|u\|_\infty = \max_{1 \le i \le n} |u_i|$

```
# Chuẩn L2
v = np.array([1, 2, 3])
np.linalg.norm(v)

# Tổng quát với Lp
np.linalg.norm(v,p)

# Chuẩn vô cùng
np.max( np.abs(v) )
```

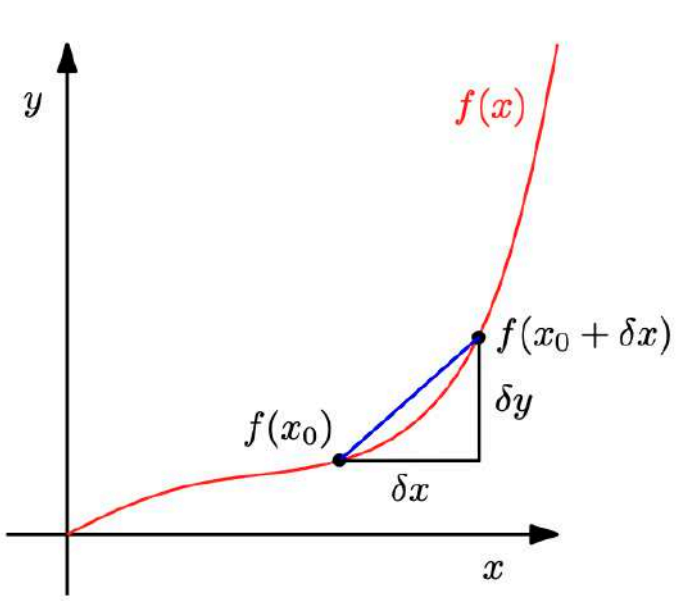
# Giải tích vector

## Vector Calculus

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Phép tính vi phân của hàm một biến

- Phép tính vi phân của hàm một biến (*Differentiation of Univariate Functions*) được tính như sau với hàm $f : \mathbb{R} \to \mathbb{R}$ để biến đổi giá trị $x \in \mathbb{R}$ qua hàm (hay ánh xạ) $f$ là $y = f(x)$.
- **Định nghĩa tỷ số sai phân (Difference Quotient):** Tỷ số sai phân của hàm số $y = f(x)$ là đại lượng
$$\frac{\delta y}{\delta x} := \frac{f(x + \delta x) - f(x)}{\delta x}$$
- **Định nghĩa đạo hàm (Derivative)**: Cho $h > 0$, đạo hàm của hàm số $y = f(x)$ là giới hạn $\frac{df}{dx} := \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}$



- **Ví dụ**: Tính đạo hàm của hàm số $f(x) = x^n, n \in \mathbb{N}$.
- Theo khai triển nhị thức Newton, $(a+b)^n = \sum_{i=0}^{n} C_i^n a^{n-i} b^i$, với $C_i^n = \dfrac{n!}{i!(n-i)!}$,

  $C_0^n = \dfrac{n!}{0!(n-0)!} = 1$, nên

$$\frac{df}{dx} = \lim_{h\to 0} \frac{(x+h)^n - x^n}{h} = \lim_{h\to 0} \frac{\sum_{i=0}^{n} C_i^n x^{n-i} h^i - x^n}{h} = \lim_{h\to 0} \frac{\sum_{i=0}^{n} C_i^n x^{n-i} h^i - C_0^n x^{n-0} h^0}{h}$$

$$= \lim_{h\to 0} \frac{\sum_{i=1}^{n} C_i^n x^{n-i} h^i}{h} = \lim_{h\to 0} \sum_{i=1}^{n} C_i^n x^{n-i} h^{i-1} = C_1^n x^{n-1} + \lim_{h\to 0} \sum_{i=2}^{n} C_i^n x^{n-i} h^{i-1}$$

- Mà $\lim_{h\to 0}\sum_{i=2}^{n} C_i^n x^{n-i}h^{i-1} = 0$, nên $\frac{df}{dx} = \frac{n!}{(n-1)!}x^{n-1} = nx^{n-1}$
- **Định nghĩa Chuỗi Taylor (Taylor Series)**: Cho hàm số $f$ khả vi vô hạn, Chuỗi Taylor của hàm số này trong lân cận điểm $x_0$ là $T_n(x) := \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$

  Trong đó $f^{(k)}(x_0)$ là đạo hàm bậc $k$ của hàm $f$ tại điểm $x_0$ và $\frac{f^{(k)}(x_0)}{k!}$ là hệ số thứ $k$ của đa thức $T_n(x)$.
- **Lưu ý: Chuỗi Maclaurin** là chuỗi $T_\infty(x)$

- **Định nghĩa quy tắc tính đạo hàm**: Ký hiệu, $f'$ là đạo hàm của hàm số $f$. Ta có:
  - $\Big(f(x)\,.\,g(x)\Big)' = f'(x)g(x) + f(x)g'(x)$
  - $\Big(f(x) + g(x)\Big)' = f'(x) + g'(x)$
  - $\left(\dfrac{f(x)}{g(x)}\right)' = \dfrac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$
  - $\Big(g(f(x)\Big)' = (g \circ f)'(x) = g'(f(x))f'(x)$
  - Trong đó, $g \circ f : x \mapsto f(x) :\mapsto g(f(x))$

# Phép tính vi phân hàm nhiều biến

- **Định nghĩa đạo hàm riêng (Partial Derivative)**: Cho hàm $f : \mathbb{R}^n \to \mathbb{R}$, $\forall x \in \mathbb{R}^n, x = (x_1, x_2, \ldots, x_n) :\mapsto f(x)$, đạo hàm riêng của $f$ theo các biến $x_i, i = \overline{1,n}$ là

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f(x_1, x_2, \ldots, x_i + h, \ldots, x_n) - f(x_1, x_2, \ldots, x_n)}{h}, \forall i = \overline{1,n}$$

- **Định nghĩa gradient**: Gradient của hàm $f : \mathbb{R}^n \to \mathbb{R}$ là vector

$$\nabla_x f = \text{grad}\, f = \frac{df}{dx} = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n}\right] \in \mathbb{R}^{1 \times n}$$

- **Ví dụ**: $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3 \in \mathbb{R}$, tính đạo hàm riêng và gradient của $f$
- Ta có:

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2$$

- Và gradient là

$$\text{grad}\, f = \left[2x_1 x_2 + x_2^3, x_1^2 + 3x_1 x_2^2\right] \in \mathbb{R}^{1\times 2}$$

- **Định nghĩa đạo hàm hàm hợp (Chain Rule)**: Cho $f : \mathbb{R}^n \mapsto \mathbb{R}$, với $x = x(t) = \Big(x_1(t), x_2(t), \ldots, x_n(t)\Big)$,

$$\text{Thì } \frac{df}{dt} = \left[\frac{\partial f}{\partial x_1}\frac{\partial f}{\partial x_2}\ldots\frac{\partial f}{\partial x_n}\right]\begin{bmatrix}\frac{\partial x_1(t)}{\partial t}\\ \frac{\partial x_2(t)}{\partial t}\\ \vdots \\ \frac{\partial x_n(t)}{\partial t}\end{bmatrix} = \left[\frac{\partial f}{\partial x_1}\frac{\partial x_1(t)}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2(t)}{\partial t} + \ldots + \frac{\partial f}{\partial x_n}\frac{\partial x_n(t)}{\partial t}\right]$$

- **Định nghĩa đạo hàm của hàm hợp khi $x = x(s,t)$:** Gradient của hàm $f$ như sau:

$$\frac{df}{d(s,t)} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial (s,t)} = \left[\frac{\partial f}{\partial x_1}\frac{\partial f}{\partial x_2}\cdots\frac{\partial f}{\partial x_n}\right]\begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t}\\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\\ \vdots & \\ \frac{\partial x_n}{\partial s} & \frac{\partial x_n}{\partial t}\end{bmatrix}$$

- **Gradient của hàm vector (Gradient of Vector-Values Function)**: Cho $f: \mathbb{R}^n \to \mathbb{R}^m, n \geq 1, m > 1$ và $f = [f_1, f_2, \ldots, f_m]^T$, với $f_i: \mathbb{R}^n \to \mathbb{R}, \forall i = \overline{1,m}$

- Thì

$$\frac{df(x)}{dx} = \left[\frac{\partial f(x)}{\partial x_1} \cdots \frac{\partial f(x)}{\partial x_n}\right]$$

$$= \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ & \vdots & \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}$$

- **Định nghĩa Jacobian**: Tập hợp tất cả các đạo hàm bậc nhất của hàm vector của hàm $f : \mathbb{R}^n \to \mathbb{R}^m$ gọi là Jacobian $J$ với $J_{ij} = \frac{\partial f_i}{\partial x_j}$:

$$J = \nabla_x f = \frac{dfx)}{dx} = \left[\frac{\partial f(x)}{\partial x_1} \cdots \frac{\partial f(x)}{\partial x_n}\right] = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ & \vdots & \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix}$$

là một ma trận $m \times n$

- **Ví dụ**: Về gradient của hàm mất mát bình phương tối thiểu (*Least-Squares Loss Function*) trong mô hình tuyến tính.
- Cho mô hình tuyến tính: $y = \Phi w$, trong đó
  - $w \in \mathbb{R}^K$ là vector tham số của $K$ đặc trưng (feature)
  - $\Phi \in \mathbb{R}^{N \times K}$ là $N$ giá trị đầu vào của mỗi bộ gồm $K$ đặc trưng
  - $y \in \mathbb{R}^N$ là $N$ quan sát (observation) tương ứng của $N$ giá trị đầu vào.
- Vấn đề đặt ra là xác định hàm $L\big(e(w)\big) := \|e(w)\|^2 = \|y - \Phi w\|^2$, sao cho hàm này có giá trị bé nhất. Hàm này gọi là hàm mất mát bình phương tối thiểu.

- Ta có: $L(e) = \|e\|^2 = e^T e, e = y - \Phi w.$
  - Suy ra $\frac{\partial L}{\partial e} = 2e^T$, và $\frac{\partial e}{\partial w} = -\Phi$
  - Nên $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial e}\frac{\partial e}{\partial w} = -2e^T\Phi = -2(y - \Phi w)^T\Phi = -2(y^T - w^T\Phi^T)\Phi$
- Do $y^T \in \mathbb{R}^{1\times N}, w^T \in \mathbb{R}^{1\times K}, \Phi^T \in \mathbb{R}^{K\times N}.$
  - Nên $-2(y^T - w^T\Phi^T) \in \mathbb{R}^{1\times N}$
- Và $\Phi \in \mathbb{R}^{N\times K}$. Nên $\frac{\partial L}{\partial w} \in \mathbb{R}^{1\times K}$

# Tìm cực trị của hàm số

- Một bài toán hầu như xuyên suốt trong tin học đó là bài toán tìm kiếm với một ràng buộc nào đó.
- Chẳng hạn, tìm những bài viết về mạng lưới thần kinh nhân tạo (ANN) có trên internet sao cho đọc dễ hiểu nhất.
- Như vậy, một vấn đề đặt ra là làm sao biểu diễn được hàm số $\mathbf{L}(x)$ để đo mức độ hiểu, có thể là $\mathbf{L}: \mathbf{D} \rightarrow \mathbb{R}^{+}$
- Từ đó đưa ra khái niệm thế nào là "hiểu". Chẳng hạn, khi $\mathbf{L}(x)$ càng lớn thì việc hiểu càng dễ.

- Khi đó ta có phát biểu bài toán dưới dạng ngôn ngữ của Toán học:
  - Cho $\mathbf{D} = \{x \in \text{Internet}/x \text{ bài viết về ANN}\}$, và hàm số $y = \mathbf{L}(x)$ có miền giá trị là số thực dương $\mathbf{L}: \mathbf{D} \to \mathbb{R}^+$ là độ hiểu của $x \in \mathbf{D}$
  - Tìm $x^*$ sao cho $x^* = \arg\max_{x \in \mathbf{D}} \mathbf{L}(x)$
- $x^*$ chính là bài viết về ANN mà dễ hiểu nhất trong số các bài viết về ANN trên internet.
- Để giải quyết bài toán này, một vấn đề đặt ra là làm sao tính được đạo hàm $\mathbf{L}'(x)$ của hàm số $y = \mathbf{L}(x)$ này.
- Từ đó giải phương trình $\mathbf{L}'(x) = 0$ để tìm $x$ sao cho $\mathbf{L}(x)$ đạt cực đại.

- Như vậy, việc tính đạo hàm của hàm số được đặt ra không phải cho Toán học mà là Tin học - lĩnh vực nghiên cứu về *những đối tượng trên tập hữu hạn đếm được*.
- Với những hàm số được biểu diễn dạng công thức tường minh thể hiện qua mối liên hệ giữa biến số và hằng số một cách rõ ràng dạng $y = f(x)$ thì hầu như đều có thể tính đạo hàm được (đương nhiên với ràng buộc là hàm số khả vi - có khả năng tính vi phân).
- Sau khi tính xong, ta thay giá trị số cụ thể của $x_i$ nào đó để có giá trị tương ứng $y_i = f'(x_i)$.
- Trong thực tế, mối qua hệ giữa $x$ và $y$ không biểu diễn được dưới dạng công thức tường minh, mà dưới dạng các giá trị rời rạc gồm các cặp $(x_i, y_i), i = \overline{1,n}$

- Ngoài ra, với những hàm số khó để tính đạo hàm, hay những hàm số có nhiều biến số, việc tính đạo hàm cũng phức tạp.
- Thêm nữa, nhờ việc khảo sát đạo hàm ta có thể nhận biết dáng điệu của hàm số.
  - Từ đạo hàm cấp một, chúng ta nhận ra đoạn trên miền xác định từ đó biết được sự tăng giảm của hàm số. Qua đó cho thấy được sự tồn tại của cực trị.
  - Để biết rõ hơn nữa hàm số chỉ có cực trị duy nhất trong một đoạn nào đó, thì hàm số không có điểm uốn (chỉ lồi hoặc lõm). Như vậy phải cần đến đạo hàm bậc hai (*là đạo hàm của đạo hàm bậc một*).
  - Ngoài ra, tính đóng của hàm số hay của tập hợp, thực chất là tính lồi của hàm số để sao cho hàm số chỉ có một giá trị đạt cực tiểu

# Gradient Descent

- Đạo hàm của hàm số $y = f(x)$ là sự biến biến thiên của hàm số đó theo $x$. Đạo hàm bằng không tại một điểm cục bộ nào đó thì hàm đạt cực trị (*ở đây xét cho trường hợp đạt cực tiểu*)
- Nếu nhìn thêm ở góc độ âm dương, với $x^*$ là điểm làm cho hàm số ***đạt cực tiểu***, $f'(x)$ sẽ chuyển từ âm sang dương khi đi qua điểm $x^*$.
  - Nói cách khác càng đi xa về phía trái của $x^*$ đạo hàm càng âm, và xa về phía phải đạo hàm càng dương.
- Đạo hàm chính là độ dốc (*gradient*) của hàm số, nên việc giảm (*descent*) độ dốc của hàm số đến bằng không, thì điểm đó là cực trị

- Thuật toán trong trường hợp này là, cho một giá trị ban đầu $x^{(0)}$, lần lượt tìm các giá trị tại các bước tính thứ $t$: $x^{(t)}$, $t > 0$ sao cho độ dốc của hàm số (*chính là đạo hàm*) giảm đến không.
- Để rời rạc hoá từng bước, gọi giá trị mới là $x^{(t+1)}$ và $\Delta x$ là độ chênh lệch giữa giá trị mới này với giá trị tính ở bước trước đó, thì $x^{(t+1)}$ chính là $x^{(t)} + \Delta x$ (nghĩa là $x^{(t+1)} = x^{(t)} + \Delta x$).
- Có 2 trường hợp xảy ra với $\Delta x$
  - Nếu $f'(x^{(t)}) < 0$, thì $x^{(t)}$ nằm bên trái của $x^*$, nên để $x^{(t+1)}$ tiến dần hơn đến $x^*$ thì $\Delta x > 0$
  - Ngược lại, nếu $f'(x^{(t)}) > 0$ thì $x^{(t)}$ nằm bên phải của $x^*$, nên để $x^{(t+1)}$ tiến dần đến $x^*$ thì $\Delta x < 0$

- Từ lập luận trên, ta thấy: **Thứ nhất**: $\Delta x$ ngược dấu với $f'(x^{(t)})$
- Ngoài ra, nếu khoảng cách giữa $x^{(t+1)}$ và $x^*$ càng xa thì $|f'(x^{(t)})| \gg 0$ (i.e. đồ thị càng dốc), nên có thể suy ra: **Thứ hai**: $|\Delta x|$ tỷ lệ thuận với $|f'(X^{(t)})|$. Như vậy, ta có thể biểu diễn $\Delta x = -\lambda f'(x^{(t)})$, với $\lambda$ là một hằng số dương như là một step-size (trong Machine Learning gọi là learning rate), nên $x^{(t+1)} = x^{(t)} + \Delta x = x^{(t))} - \lambda f'(x^{(t)})$
- **Phương pháp Gradient Descent**: Cho $x^{(0)} \in \mathbb{R}^n$, và ***cực trị địa phương*** của hàm $f$: $\mathbb{R}^n \mapsto \mathbb{R}$ này là giá trị $x^{(t+1)}$ với $t$ khá lớn được tính từ biểu thức lặp $x^{(t+1)} = x^{(t)} - \lambda(\nabla_x f)(x^{(t)}), \forall t \geq 0$, trong đó $\lambda$ là hằng số dương.

- Cũng có thể dùng khai triển Taylor đến đạo hàm cấp một trong lân cận điểm $x^{(t)}$ để biểu diễn đạo hàm qua giá trị của hàm số.
- Cho trước hằng số $\lambda > 0$ khá bé, thì $f(x^{(t)} + \lambda) = f(x^{(t)}) + \lambda f'(x^{(t)}) + O(\lambda^2)$
  - Suy ra $\lambda f'(x^{(t)}) \approx f(x^{(t)} + \lambda) - f(x^{(t)})$
- **Phương pháp sai phân**: Cho $x^{(0)} \in \mathbb{R}^n$ và hàm $y = f(x)$ ***cực trị địa phương*** là giá trị $x^{(t+1)}$ với $t$ khá lớn thoả mãn $x^{(t+1)} = x^{(t)} - [f(x^{(t)} + \lambda) - f(x^{(t)})], \forall t \geq 0$, trong đó $\lambda$ là hằng số dương khá bé cho trước.

# Ví dụ

- Tìm cực trị của hàm $f(x) = x^2 - 2x + 1$, với $x \in \mathbb{R}$ bằng phương pháp Gradient Descent.
  - Ta có $f'(x) = 2x - 2$
  - Từ đây với $x^{(0)} \in \mathbb{R}$ ban đầu và cho trước một hằng số $\lambda$ khá bé, và một sai số $\varepsilon$
  - Lần lượt tính các giá trị của $x^{(t)}$ cho đến khi 2 giá trị tính liên tiếp nhau $|x^{(t+1)} - x^{(t)}| < \varepsilon$.
    - $x^{(t+1)} = x^{(t)} - \lambda f'(x^{(t)}), t = 0,1,2,...$

- Tìm cực trị của hàm số $f(x,y) = x^2 + y^2, \forall x, y \in \mathbb{R}$
  - Đặt $x_1 = x, x_2 = y$, hàm số được viết lại là $f(x_1, x_2) = x_1^2 + x_2^2$
  - Hay

$$f(x_1, x_2) = [x_1 - x_2, x_1 + x_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= [x_1, x_2] \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Suy ra $\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 2\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$

- Cho $x^{(0)} = [x_1^{(0)}, x_2^{(0)}]$, cực tiểu là giá trị $[x_1^{(t+1)}, x_2^{(t+1)}]$ với $t$ khá lớn được tính theo công thức lặp

- $\begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix} - 2\lambda \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, t = 0,1,...,N$

- Cho $f(x,y) = x^2 + xy + 10y^2 - 5x - 3y, \forall x, y \in \mathbb{R}$. Hãy tìm cực tiểu địa phương của hàm này với $(x^{(0)}, y^{(0)})$ ban đầu cho trước.

  - Đặt $x_1 = x, x_2 = y$, hàm $f$ được viết lại như sau:

$$f(x_1,x_2) = \frac{1}{2}\left[(2x_1 + x_2)x_1 + (x_1 + 20x_2)x_2\right] - (5x_1 + 3x_2)$$

$$= \frac{1}{2}\begin{bmatrix}2x_1 + x_2 & x_1 + 20x_2\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - \begin{bmatrix}5 & 3\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}2 & 1\\1 & 20\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} - \begin{bmatrix}5 & 3\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Dưới dạng ma trận, viết lại $f(x_1, x_2) = \frac{1}{2}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

- Suy ra $\nabla f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} - \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T$

- Cho $x^{(0)} = [x_1^{(0)}, x_2^{(0)}] = [-3, -1]^T$, lặp lại các bước tính sau với $N$ khá lớn

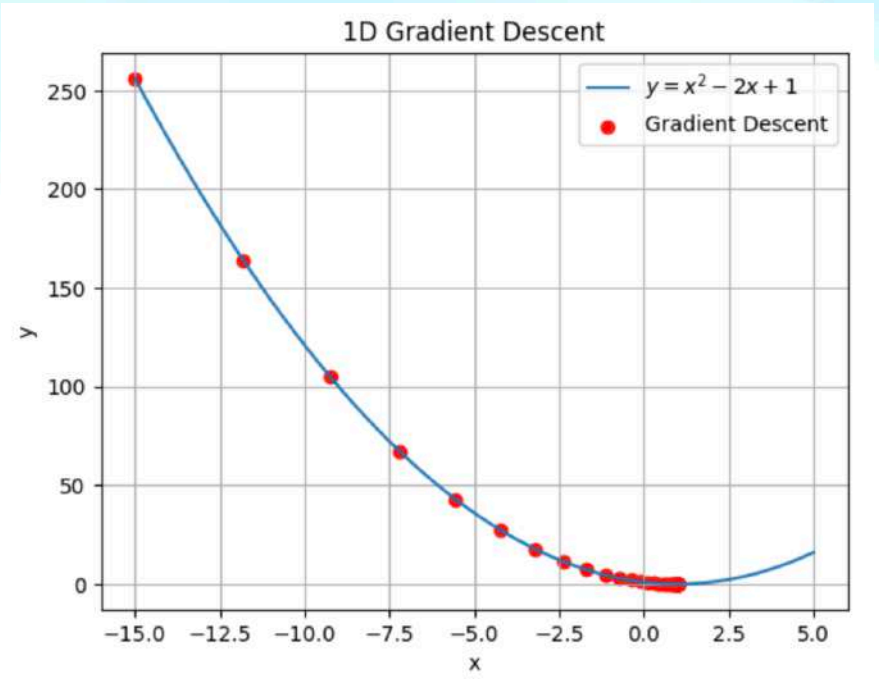
$$\begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix} - \lambda \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} + \lambda \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T, t = 0,1,...,N, \text{ và } \lambda = 0.085$$

# Thuật giải

- **Input**: $x^{(t+1)} = x^{(0)}, \lambda, \varepsilon, \nabla f(x)$
- **Output**: $x^* = x^{(t+1)}$
- **Begin**
  - **Repeat**
    - $x^{(t)} = x^{(t+1)}$
    - $x^{(t+1)} = x^{(t)} - \lambda \nabla f(x^{(t)})$
  - **Until** $\|x^{(t+1)} - x^{(t)}\| < \varepsilon,$
- **End.**

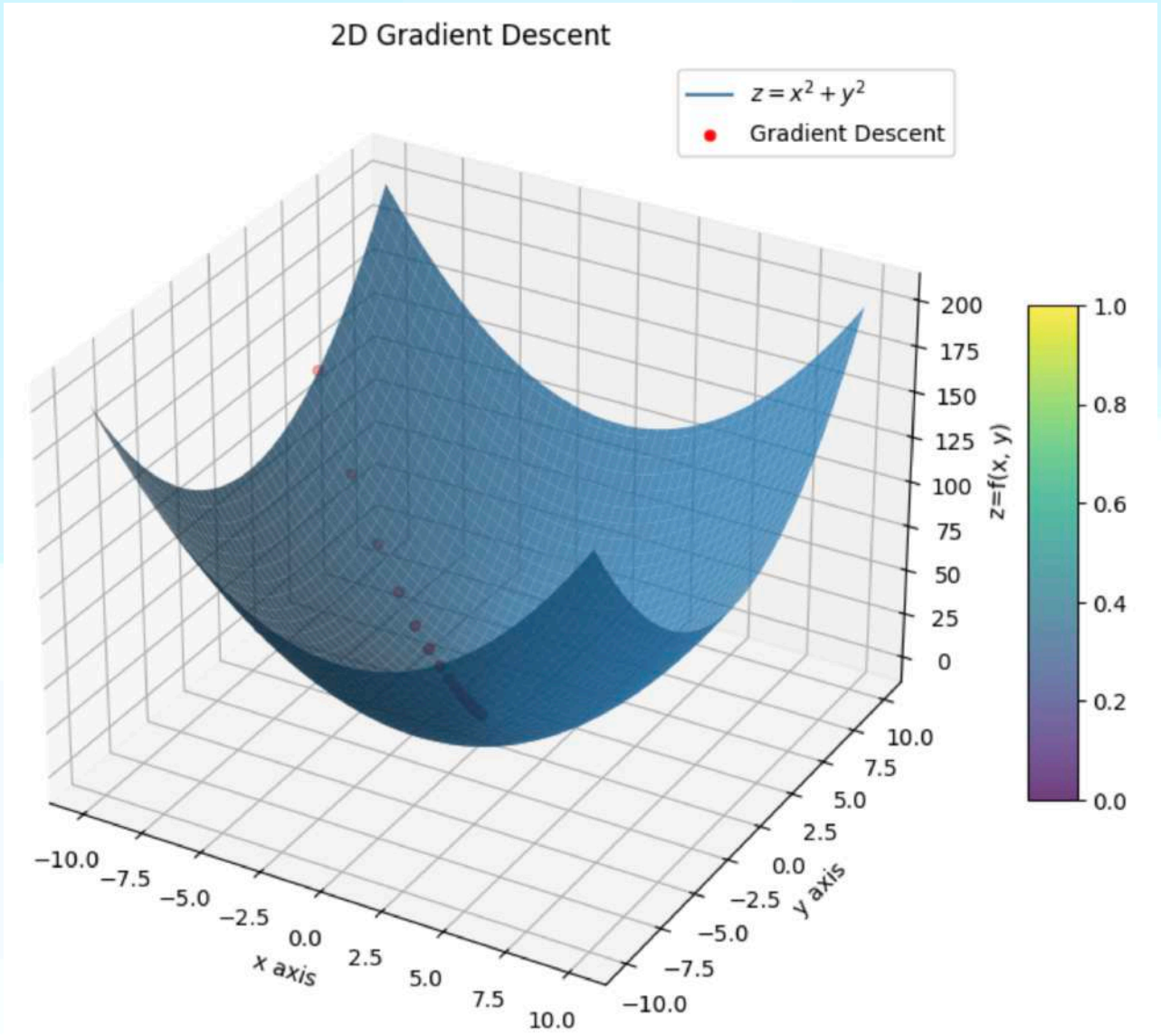
```
### f(x) = x^2 - 2x + 1

import numpy as np
from sympy import *

lamda, eps = 0.1, 0.00001
x = symbols('x')
f = x**2 - 2*x + 1
F = lambdify(x,f) # SymPy expressions into functions

def GD():
    xt, flag, iter = -15, True, 0
    trajectory = [xt]

    while flag:
        xt1 = xt - lamda*diff(f,x).subs(x,xt)
        iter = iter + 1
        if abs(xt1 - xt) < eps:
            flag = False
        else:
            xt = xt1
        trajectory.append(xt)
    return xt1, iter, trajectory

xmin, iter, trajectory = GD()
print("Cực trị: %.6f với %d lần lặp" % (xmin,iter))
```

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

x = np.linspace(-15, 5, 100)
y = F(x)
plt.plot(x, y, label=r'$y=x^2 - 2x + 1$')

plt.scatter(trajectory,[F(xt) for xt in
trajectory],color='red',label='Gradient Descent')

plt.xlabel('x')
plt.ylabel('y')
plt.title('1D Gradient Descent')
plt.legend()
plt.grid(True)
plt.show()
```

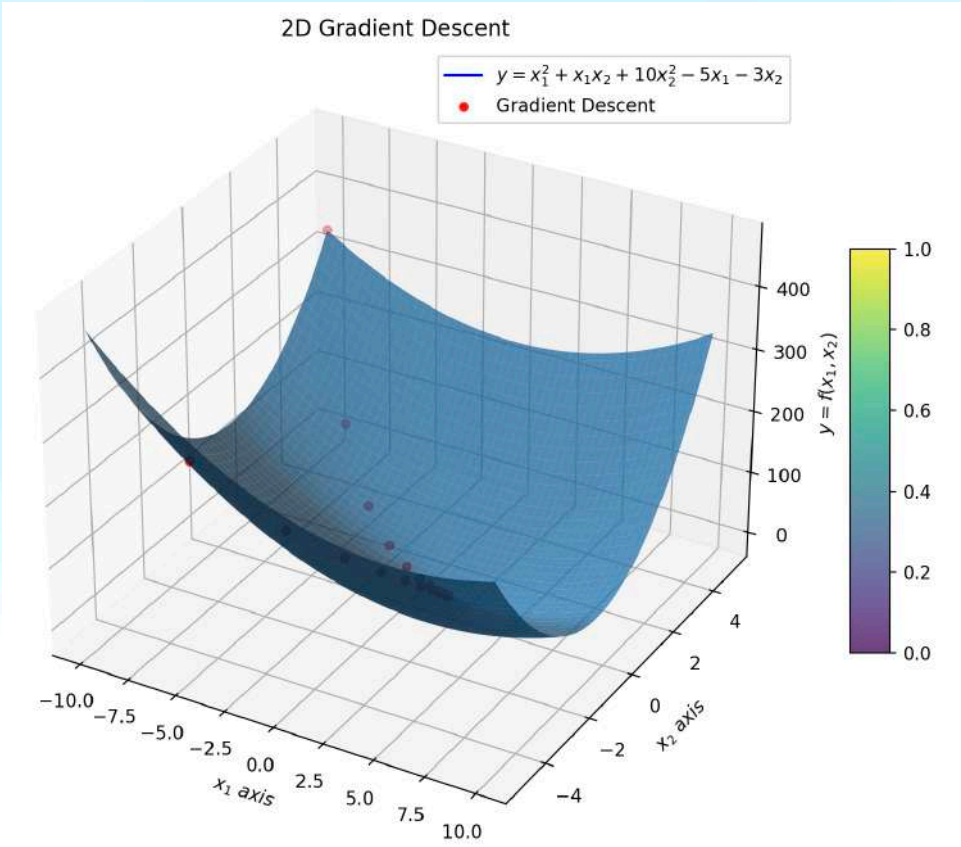


```
### f(x,y) = x^2 + y^2

import numpy as np
from sympy import *

x, y = symbols( 'x y' )
f = x**2 + y**2
F = lambdify((x, y),f)

lamda, n_iters = 0.1, 20
x0, y0 = -10, 5
def GD2D():
    X = x0
    Y = y0
    trajectory = [(X,Y)]

    for _ in range(n_iters):
        dfx, dfy = diff(f,x).subs({x:X,y:Y}),diff(f,y).subs({x:X,y:Y})
        X = X - lamda * dfx
        Y = Y - lamda * dfy
        trajectory.append((X,Y))
    return trajectory

trajectory = GD2D()

x_vals = np.linspace(-10, 10, 100)
y_vals = np.linspace(-10, 10, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = F(X, Y)
```

```
from matplotlib.lines import Line2D
import matplotlib.pyplot as plt

fig = plt.figure( figsize=(10,8))
ax = plt.axes(projection ='3d')

surf = ax.plot_surface(X,Y,Z,alpha=0.75)
surf_proxy = Line2D([0],[0],color='blue')

scat = ax.scatter([x[0] for x in trajectory],[x[1] for x in trajectory],
           [F(x[0],x[1]) for x in trajectory],color='r')

ax.legend([surf_proxy,scat], ['$z = x^2+y^2$', 'Gradient Descent'])
fig.colorbar(surf, shrink=0.5, aspect=10)

ax.set_xlabel('x axis')
ax.set_ylabel('y axis')
ax.set_zlabel('z=f(x, y)')
ax.set_title('2D Gradient Descent')

plt.grid(True)
plt.show()
```


2D Gradient Descent
$z = x^2 + y^2$
Gradient Descent
x axis
y axis
z=f(x, y)
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
200
175
150
125
100
75
50
25
0
1.0
0.8
0.6
0.4
0.2
0.0

```
### f(x,y) = x^2 + xy+ 10y^2 -5x - 3y

import numpy as np
from sympy import *

x1, x2 = symbols( 'x1 x2' )
f = x1**2 + 10*x2**2 + x1*x2 - 5*x1 - 3*x2
F = lambdify((x1,x2),f)

lamda, n_iters = 0.085, 20
x10, x20 = -10, 5
def GD2D_():
    X1 = x10
    X2 = x20
    trajectory = [(X1,X2)]
    for _ in range(n_iters):
        dfx1, dfx2 = diff(f,x1).subs({x1:X1,x2:X2}),diff(f,x2).subs({x1:X1,x2:X2})
        X1 = X1 - lamda * dfx1
        X2 = X2 - lamda * dfx2
        trajectory.append((X1,X2))
    return trajectory
```



# Giải hệ phương trình

- Có thể sử dụng phương pháp Gradient Descent để giải hệ phương trình
- Cho hệ phương trình tuyến tính $Ax = b$, cần tìm $x \in \mathbb{R}^n$ thoả hệ này.
- Hệ phương trình trên có thể đưa về phương trình tương đương là $Ax - b = 0$
- Chính vì vậy, việc giải hệ phương trình thực chất là tìm $x^*$ làm cho sai số bình phương (squared error) đạt cực tiểu.
- Điều đó có nghĩa là tìm cực tiểu của $f(x) = ||Ax - b||^2 = (Ax - b)^T(Ax - b)$
- Ta có gradient của $f$ tương ứng với biến $x$ là $(\nabla_x f)(x) = 2(Ax - b)^T A$. Khi đó $x^*$ là giá trị của lần tính thứ $t + 1$: $x^{(t+1)} = x^{(t)} - \lambda((\nabla_x f)(x^{(t)})), t = 0,1,2,...$ với $x^{(0)}$ cho trước.

# Vấn đề chính của học máy

## The Pillars of Machine Learning:

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Giới thiệu

- Có 4 vấn đề mang tính trụ cột (pillar) của học máy:
  - Hồi quy (Regression)
  - Rút gọn chiều (Dimensionality Reduction)
  - Ước lượng mật độ (Density Estimation)
  - Phân nhóm: phân lớp (Classification), gom cụm (Clustering)
- Có 3 vấn đề quan tâm chính của học máy, đó là dữ liệu (data), mô hình (model), và học (learning).

# Data

- Đơn giản để cho máy tính có thể thực hiện được thì Data như là Vector với các thành phần là con số.
- Chẳng hạn, với bảng dữ liệu như sau:

| Tên | Giới tính | Học vị | Plus Code | Tuổi | Lương |
|---|---|---|---|---|---|
| Đinh Tiên Hùng | Nam | Tiến sĩ | 84W6+H2R | 38 | 25000000 |
| Lê Đại Hưng | Nam | Thạc sĩ | 849V+22W | 28 | 18000000 |
| Lý Thái Hạnh | Nữ | Thạc sĩ | 938R+P5 | 36 | 20000000 |
| Trần Hưng Quốc | Nam | Tiến sĩ | PMCW+HG | 65 | 30000000 |
| Nguyễn Gia Hoa | Nữ | Tiến sĩ | FPQW+GQ | 42 | 25000000 |

- Quy ước: Giới tính Nam là 0, Nữ là 1; Học vị từ cao xuống thấp với Tiến sĩ; và Plus Code có thể gồm 2 thành phần là kinh tuyến và vị tuyến.
- Từ đó có bảng

| Tên | Giới tính | Học vị | Vĩ độ | Kinh độ | Tuổi | Lương |
|---|---|---|---|---|---|---|
| Đinh Tiên Hùng | 0 | 6 | 21.01 | 105.84 | 38 | 25000000 |
| Lê Đại Hưng | 0 | 5 | 23.15 | 160.36 | 28 | 18000000 |
| Lý Thái Hạnh | 1 | 5 | 51.81 | 5.82 | 36 | 20000000 |
| Trần Hưng Quốc | 0 | 6 | 10.72 | 106.69 | 65 | 30000000 |
| Nguyễn Gia Hoa | 1 | 6 | 53.49 | -2.53 | 42 | 25000000 |

- Trong ví dụ trên, cột Tên mang tính định danh, không có giá trị khi huấn luyện nên có thể chọn bộ Data gồm $N = 5$ dòng và $K = 6$ cột như sau:

| Giới tính | Học vị | Vĩ độ | Kinh độ | Tuổi | Lương |
|---|---|---|---|---|---|
| 0 | 6 | 21.01 | 105.84 | 38 | 25000000 |
| 0 | 5 | 23.15 | 160.36 | 28 | 18000000 |
| 1 | 5 | 51.81 | 5.82 | 36 | 20000000 |
| 0 | 6 | 10.72 | 106.69 | 65 | 30000000 |
| 1 | 6 | 53.49 | -2.53 | 42 | 25000000 |

- Một cách tổng quát, gọi số mẫu (*example*) là $N$. Khi đó trong bộ dữ liệu (*Dataset*) là một mảng bao gồm mảng gồm $N$ vector $x_n, n = 1,...,N$. Mỗi vector này được gọi là một ***data point*** trong Học máy (ML - Machine Learning).
- Mỗi cột trong bộ dữ liệu này được gọi là một đặc trưng (*feature*) của *data point*. Khi đó một vector có số thành phần là $K$ tương ứng với $K$ đặc trưng.
- Chẳng hạn, với bảng dữ liệu như trên, coi nhãn (*label*) là lương tháng, ký hiệu là $y_n, n = 1,...,N$ và ta muốn tạo ra một Dataset để học có giám sát (*Supervised Learning*) từ các đặc trưng là *Giới tính*, *Học vị*, *Tuổi* để nhận biết *Lương*; khi đó số mẫu là $N = 5$ và số đặc trưng $K = 3$.
- Bộ dữ liệu là $(X, y) \in \mathbb{R}^{N \times K} \times \mathbb{R}$, trong đó $X$ là ma trận $N$ dòng và $K$ cột, $y$ là vector $N$ phần tử.

# Models

- Có thể coi *Model* như là một Hàm (*Function*).
  - Gọi $x$ là một vector có các thành phần là các đặc trưng (*feature*), hãy tìm một hàm $f$ để làm sao biết được nhãn (*label*) của các đặc trưng này dạng $f$: $\mathbb{R}^K \rightarrow \mathbb{R}$
  - Hàm $f$ được gọi là hàm dự đoán (*predictive function*), trong Học máy hàm này được biết như là **yếu tố dự đoán** (*predictor*) có thể biểu diễn dạng $f(x) = w^T x + w_0$ với các ẩn số là $w = (w_1, w_2, \ldots, w_K)$ và $w_0$.
- Ngoài ra, thay vì coi một *predictor* là một hàm duy nhất, chúng ta có thể coi các **yếu tố dự đoán** là các mô hình xác suất, tức là các mô hình mô tả phân phối của các hàm có thể có (*model as probability distributions*).

# Learning

- Thực chất của việc học (*learning*) là tìm một mô hình và các tham số (*parameter*) tương ứng sao cho predictor kết quả sẽ hoạt động tốt trên dữ liệu chưa có (*unseen data*).
- Trong việc học này, có ba công việc cần phải quan tâm
  - Dự đoán (khi model là function) hoặc suy luận (khi có mô hình là các phân phối xác suất)
  - Huấn luyện hoặc ước lượng tham số
  - Điều chỉnh siêu tham số (*hyperparameter*) hoặc lựa chọn mô hình

# Độ đo trong bài toán phân lớp

## Metrics in Classification Problem

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Minh hoạ

- Cần phân lớp về ung thư vú là lành tính hay ác tính.
- Sử dụng dữ liệu tại https://www.kaggle.com/datasets/sahilnbajaj/cancer-classification
- Bộ dữ liệu gồm 30 thuộc tính, và nhãn gồm 2 lớp: ác tính (1 - malignant) và lành tính (0 - benign) lưu ở cột cuối cùng của tập tin cancer_classification.csv
- Sử dụng Python để giải quyết bài toán dự đoán với các bước:
  - Load dữ liệu vào biến trong bộ nhớ
  - Chia dữ liệu thành tập huấn luyện (sử dụng để huấn luyện mô hình) và tập kiểm tra (sử dụng để đánh giá hiệu suất mô hình)
  - Tạo mô hình bằng cách sử dụng một thuật toán phân lớp nào để huấn luyện

# Bằng Python với các thư viện cần thiết

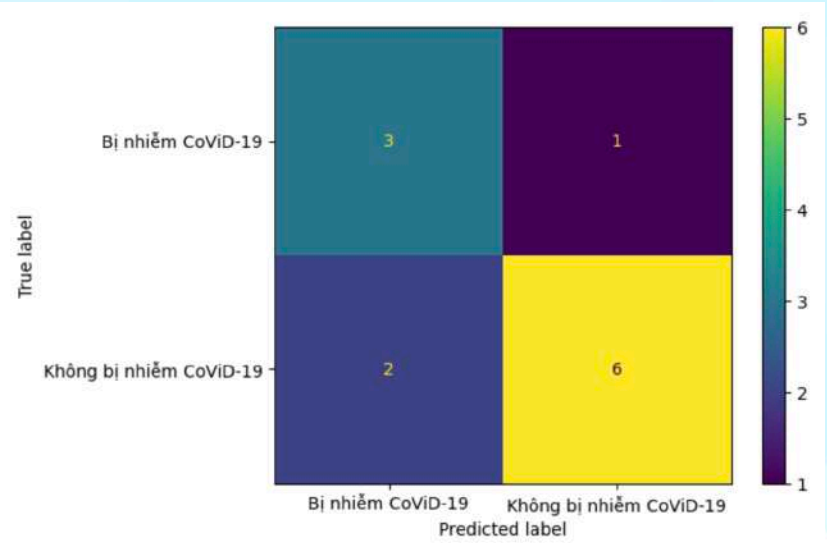
```
### Chuẩn bị dữ liệu
import polars as pl
file = '/Users/lang/Documents/Works/Training/Mathematics for Machine Learning/cancer_classification.csv'
df = pl.read_csv( file )

import numpy as np
X = np.array(df.drop('benign_0__mal_1'))
y = np.array(df.select('benign_0__mal_1')).ravel()

### Các bước tạo mô hình dự đoán
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5).fit(X_train, y_train)

### Sử dụng để dự đoán
y_pred = model.predict(X_test)
```

# Khái niệm Positive và Negative

- Trong việc phân lớp cần xác định trước về mặt ý nghĩa thực tế đâu là lớp Positive (dương tính) và đâu là Negative (âm tính).
  - Giá trị của Positive phụ thuộc vào cách mà định nghĩa lớp trong bài toán phân loại: lớp Positive là lớp quan tâm hoặc muốn dự đoán chính xác hơn, còn Negative là lớp còn lại.
- Chẳng hạn, chẩn đoán bệnh thì Positive là lớp bị bệnh, còn Negative là lớp không bị bệnh; nhưng khi khám bệnh để tuyển người làm việc thì Positive là lớp không bị bệnh, còn Negative là lớp bị bệnh.

- Từ đó có các khái niệm:
  - **True Positive** (TP): Mẫu thuộc lớp Positive được dự đoán chính xác là Positive
  - **False Positive** (FP): Mẫu thuộc lớp Negative được dự đoán thành Positive (dự đoán sai hay nhầm là Positive)
  - **True Negative** (TN): Mẫu thuộc lớp Negative được dự đoán chính xác là Negative
  - **False Negative** (FN): Mẫu thuộc lớp Positive nhưng dự đoán thành Negative (dự đoán sai hay nhầm là Negative)

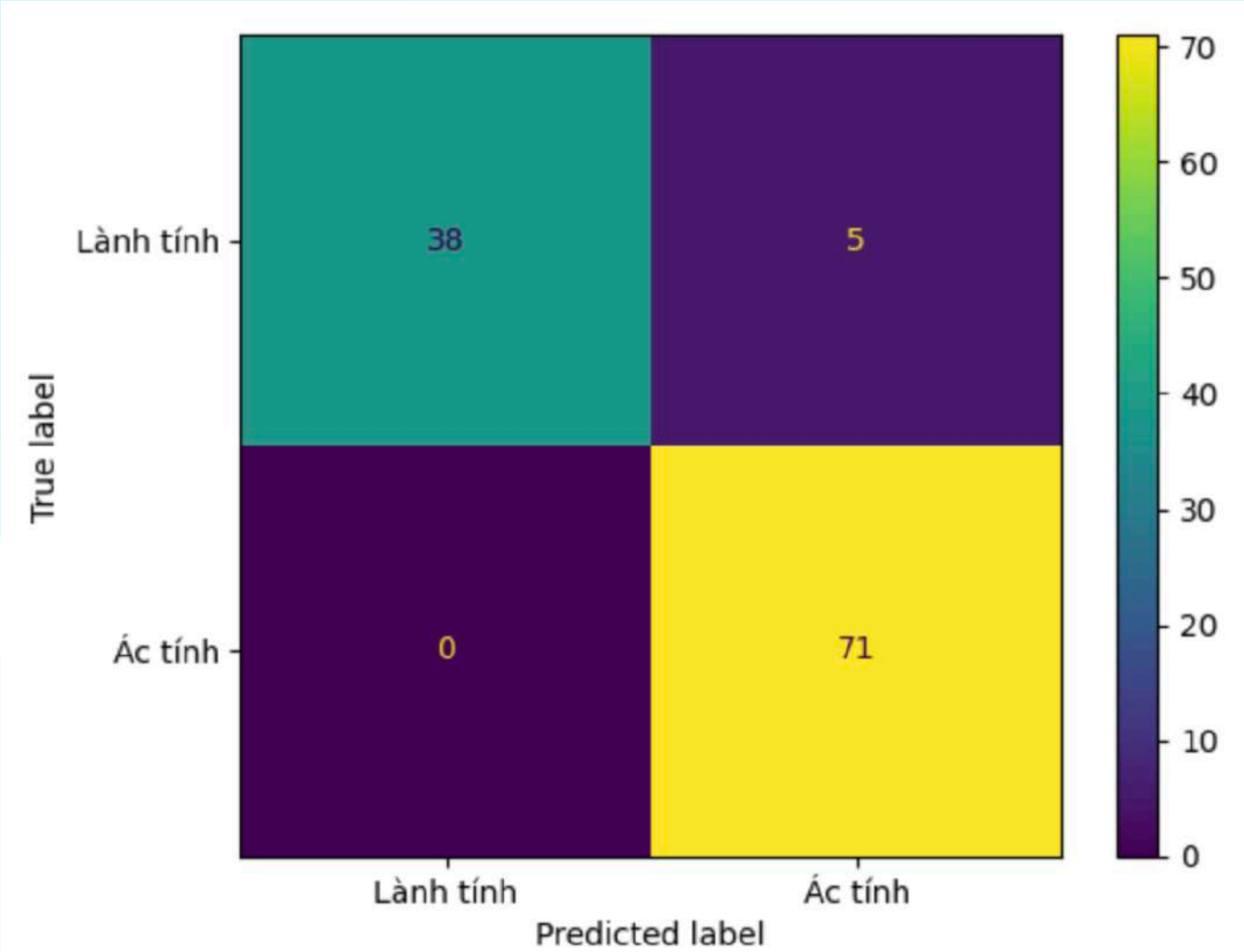
- Giả sử có 12 mẫu xét nghiệm CoViD-19, trong đó có 8 mẫu không bị nhiễm (Negative), còn 4 mẫu bị nhiễm (Positive)
- Nhưng kết quả dự đoán (predict) của một mô hình học máy là trong số 8 mẫu Negative chỉ dự báo đúng 6 mẫu, và trong số 4 mẫu Positive dự đoán đúng 3 mẫu.
- Ma trận phân loại (Confusion Matrix) biểu diễn bằng hình ảnh bên cạnh.




```
from sklearn.metrics import *

y_true = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
y_pred = [0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

cm = confusion_matrix(y_true, y_pred)
ConfusionMatrixDisplay( confusion_matrix=cm,display_
labels=["Bị nhiễm CoViD-19","Không bị nhiễm
CoViD-19"] ).plot()
```

- Hoặc với dữ liệu ung thứ vú như phần trên.
- Bộ dữ liệu có 569 dữ liệu (data point), dành 20% cho thử nghiệm (test data): 114 dữ liệu
- Trong bộ dữ liệu này, giả sử việc tạo train data và test data luôn luôn giống nhau ở các lần thực thi chương trình khác nhau.
- Khi đó, trong số 114 dữ liệu dùng để thử nghiệm này có 71 trường hợp ác tính (giá trị 1) và 43 trường hợp lành tính (giá trị 0)

```
from sklearn.metrics import *

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay( confusion_matrix=cm,display_labels=["Lành tính","Ác tính"] ).plot()
```


Lành tính
Ác tính
True label
38
5
0
71
Lành tính
Ác tính
Predicted label
70
60
50
40
30
20
10
0

# Recall hay Sensitivity (Độ nhạy)

- Còn gọi là tỷ số thu hồi, hay tỷ số TP (**True Position Rate - TPR**)
- Nhằm xác định trong tất cả các trường hợp thực tế (True case) là Positive, thì có bao nhiêu dự đoán Positive là đúng.
- Đây là một độ đo quan trọng khi chúng ta muốn đảm bảo không bỏ sót các mẫu của lớp thiểu số, nên còn được gọi là tỷ lệ phát hiện.
- Công thức tính: $Recall = \frac{TP}{TP + FN}$
- Ý nghĩa: tỷ số phần trăm mẫu thuộc lớp Positive được dự đoán chính xác.

# Accuracy

- Là độ chính xác tổng thể
- Nhằm xác định trong tổng số cần dự đoán (cả Positive và Negative), thì có bao nhiêu phần là dự đoán đúng (là $TP + TN$).
- Công thức tính: $Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$
- Ngoài ra còn có **Balanced Accuracy** (BA) - **độ chính xác cân bằng** là một chỉ số được sử dụng để đánh giá hiệu suất của mô hình phân lớp trong trường hợp dữ liệu có sự mất cân bằng giữa các lớp.
- BA là trung bình cộng của sensitivity và specificity.

- Ý nghĩa: Tỷ số phần trăm đo độ phù hợp khi dữ liệu mất cân bằng và một độ đo chính xác tổng quát hơn so với accuracy truyền thống vì phối hợp cả tỷ số Positive thật - TPR (hay độ nhạy sensitivity) và tỷ số Negative thật - TNR (hay độ đặc hiệu specificity).
- Công thức tính: $BA = \frac{\frac{TP}{TP+FN} + \frac{TN}{TN+FP}}{2}$
- Trong Python có phương thức sklearn.metrics.balanced_accuracy_score()

# Precision hay Positive Predictive Value

- Còn gọi là độ chính xác, hay trị số tiên lượng (PPV - Positive Predictive Value)
- Nhằm xác định trong tất cả các dự đoán về Positive (cả true và false), thì có bao nhiêu phần dự đoán Positive là đúng.
- Đây là một độ đo quan trọng khi chúng ta quan tâm đến việc tránh dự đoán sai cho lớp thiểu số.
- Công thức tính: $Precision = \frac{TP}{TP + FP}$
- Ý nghĩa: Tỷ số phần trăm mẫu được dự đoán là Positive thực sự thuộc lớp Positive

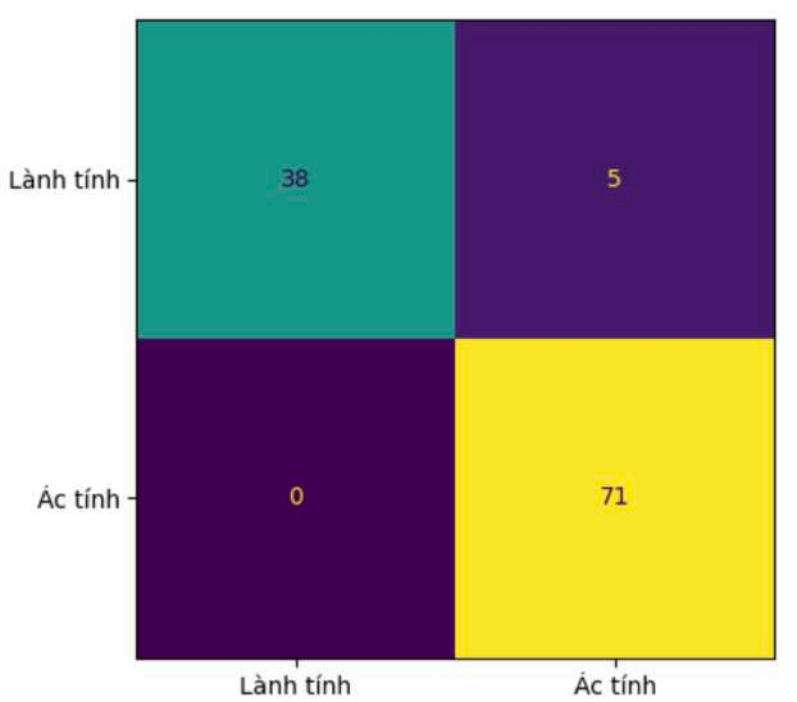
# F1-Score

- Với 2 độ đo Precision và Recall được dùng để khắc phục hạn chế của độ do Accuracy đó là chỉ tập trung cho việc đánh giá mô hình qua việc phân loại theo lớp Positive.
- Cả hai có tử số giống nhau, chỉ khác về mẫu số.
  - Precision quan tâm đến việc dự đoán về Positive (cả dự đoán đúng Positive và cả dự đoán sai Positive).
  - Trong khi đó Recall quan tâm đến những gì là đúng với thực tế là Positive (cả dự đoán đúng Positive và cả dự đoán sai Negative - nghĩa là đúng Positive).
- Từ đó, có F1-Score là trung bình điều hoà (harmonic mean) của Precision và Recall, nhằm quan tâm nhiều đến độ đo có giá trị nhỏ.

- Công thức tính: $F_1 - Score = 2\frac{Precision \times Recall}{Precision + Recall}$
- Ý nghĩa: Là tỷ số phần trăm, giúp cân bằng giữa Precision và Recall
- Lưu ý: Trung bình điều hoà là nghịch đảo của trung bình cộng của nghịch đảo các gía trị.
  - Trung bình điều hoà của $n$ giá trị $x_1, x_2, \ldots, x_n$ được tính là $\frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$
  - Nên $F_1 - Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$

```
from sklearn.metrics import *

y_true = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
y_pred = [0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

print( recall_score(y_true,y_pred) )
print( precision_score(y_true,y_pred) )
print( f1_score(y_true,y_pred) )
print( accuracy_score(y_true,y_pred) )
```

```
from sklearn.metrics import *

print( "Recall: %4.2f" %recall_score(y_test,y_pred) )
print( "Precision: %4.2f" %precision_score(y_test,y_pred) )
print( "F1: %4.2f" %f1_score(y_test,y_pred) )
print( "Accuracy: %4.2f" %accuracy_score(y_test,y_pred) )
print( "Balanced Accuracy: %4.2f" %balanced_accuracy_score(y_test,y_pred) )
```



# Trung bình điều hoà

- Trung bình cộng và trung bình điều hòa là hai phép toán thống kê thường được sử dụng để tính toán giá trị trung tâm của một tập dữ liệu.
- Tuy nhiên, trong một số trường hợp, trung bình điều hòa có thể là lựa chọn phù hợp hơn so với trung bình cộng. Dưới đây là một số lý do:
- **Khi dữ liệu có nhiều giá trị cực đoan**:
  - Trung bình cộng rất nhạy cảm với các giá trị cực đoan (outlier) trong dữ liệu. Nếu có một hoặc nhiều giá trị rất cao hoặc rất thấp trong tập dữ liệu, trung bình cộng sẽ bị ảnh hưởng đáng kể, dẫn đến kết quả không chính xác hoặc sai lệch.
  - Mặt khác, trung bình điều hòa ít bị ảnh hưởng bởi các giá trị cực đoan hơn. Do sử dụng phép nghịch đảo của mỗi giá trị trong tập dữ liệu trước khi tính toán trung bình, giúp giảm thiểu tác động của các giá trị cao hoặc thấp.

- Chẳng hạn, tập dữ liệu gồm các số: 2, 4, 4, 4, 100.
  - Trung bình cộng: $(2 + 4 + 4 + 4 + 100) / 5 = 22$
  - Trung bình điều hòa: $5 / (1/2 + 1/4 + 1/4 + 1/4 + 1/100) = 4$
- Như ta thấy, trung bình điều hòa là 4, gần với giá trị trung tâm thực tế của tập dữ liệu hơn (4) so với trung bình cộng (là 22) bị ảnh hưởng bởi giá trị cực đoan 100.
- **Khi ta muốn tính trung bình của các đại lượng có tỷ lệ:**
  - Trung bình điều hòa thường được sử dụng để tính trung bình của các đại lượng có tỷ lệ,
  - Như tốc độ, tỷ lệ, tỷ lệ phần trăm.

- Ví dụ, khi tính trung bình tốc độ của hai xe đi với tốc độ 40 km/h và 60 km/h, ta nên sử dụng trung bình điều hòa thay vì trung bình cộng:
  - Trung bình cộng: $(40 + 60) / 2 = 50$ km/h
  - Trung bình điều hòa: $2 / (1/40 + 1/60) = 48$ km/h
- Trung bình điều hòa (48 km/h) gần với tốc độ trung bình thực tế của hai xe hơn so với trung bình cộng (50 km/h).
- **Khi ta muốn tính trung bình của các đại lượng có đơn vị đo khác nhau:**
  - Trung bình cộng không thể được sử dụng để tính trung bình của các đại lượng có đơn vị đo khác nhau. Trung bình điều hòa có thể được sử dụng bằng cách chuyển đổi các đại lượng sang cùng đơn vị trước khi tính trung bình.

- Giả sử ta có tập dữ liệu gồm giá tiền của ba chiếc xe: 10 triệu đồng, 500 USD, và 800 EUR.
  - Ta có thể chuyển đổi tất cả các giá trị sang đơn vị triệu đồng trước khi tính trung bình điều hòa:
    - 1 USD ≈ 23.000 VND
    - 1 EUR ≈ 26.000 VND
  - Giá tiền trung bình của ba chiếc xe: 3 / (1/10 + 1/(500 * 23.000) + 1/(800 * 26.000)) ≈ 12.4 triệu đồng

- Lưu ý thêm: Còn có trung bình nhân,
  - Thường được sử dụng khi các giá trị cần tính trung bình là tỷ lệ phần trăm
  - Chẳng hạn trung bình nhân của $n$ đại lượng $x_1, x_2, \ldots, x_n$ là $\sqrt[n]{x_1 \times x_2 \times \ldots \times x_n}$

# Độ đo trong bài toán hồi quy

## Metrics in Regression Problem

**Dr. Tran Van Lang, A.Prof. in Computer Science**

# Bài toán hồi quy

- Cho một dãy các cặp riêng biệt $(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)$ gọi là bộ dữ liệu huấn luyện (training dataset) gồm có $N$ quan sát (data point),
  - Trong đó mỗi quan sát chứa $K$ thành phần $X_i = (x_1, x_2, \ldots, x_K) \in \mathbb{R}^K$ gọi là $K$ đặc trưng (feature) của bộ dữ liệu
  - Còn các $y_i, \forall i = \overline{1,N}$ gọi là các biến phụ thuộc, đôi khi đó là nhãn (label) hay giá trị thực (truth value), đó chí là dữ liệu thực tế.
- Vấn đề đặt ra là tìm hàm $f$ sao cho $f(X_i)$ xấp xỉ tốt nhất $y_i, \forall i = \overline{1,N}$.

- Trong học máy (Machine Learning), giá trị $f(X_i)$ được ký hiệu là $\hat{y}_i = f(X_i)$ được gọi là giá trị dự đoán hay giá trị mong đợi (outcome),
- Còn hàm $f$ được gọi là mô hình huấn luyện (training model)
- Giá trị mong đợi này phải xấp xỉ tốt nhất các giá thực có trong bộ dữ liệu.
- Chính vì vậy, cần có các các thước đo để đánh giá mô hình huấn luyện này.

- Có nhiều cách để đánh giá mô hình hồi quy, tuỳ thuộc vào mục đích nghiên cứu và loại mô hình hồi quy để dùng chỉ số đánh giá tương ứng.
  - Nếu mục đích nghiên cứu là dự đoán giá trị, dùng các độ đo về sai số
  - Còn nếu mục đích nghiên cứu là kiểm định giả thuyết, tính chất của mô hình, dùng các chỉ số kiểm định liên quan đến dữ liệu, tỷ lệ dự đoán đúng.

# Mean Absolute Error

- Mean Absolute Error - MAE (Sai số tuyệt đối trung bình) được dùng để đo lường sai số trung bình của mô hình so với dữ liệu thực tế.
- Công thức tính: $\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|$
- Ưu điểm: MAE tính trung bình cộng giá trị tuyệt đối của sai số giữa giá trị dự đoán và giá trị thực tế. Do đó, nó có thể được hiểu một cách dễ dàng và trực quan bởi người dùng không chuyên về kỹ thuật.
- Hạn chế: Do MAE không bình phương các phần dư, nên không nhạy với các điểm dữ liệu ngoại lai. Điều này có thể dẫn đến việc mô hình không đánh giá chính xác các điểm dữ liệu hiếm gặp hoặc ngoại lai. Hơn nữa, một mô hình có thể có nhiều sai số nhỏ và một sai số lớn, nhưng MAE chỉ tính trung bình giá trị tuyệt đối của chúng. Điều này có thể làm mất đi thông tin quan trọng về sự biến động của sai số.

# Lưu ý

- Khi dùng MAE, ... giá trị có thể lớn hơn 1, trong trường hợp này có thể do một trong những nguyên nhân sau:
  - Dữ liệu có nhiễu nhiều
  - Mô hình hồi quy không phù hợp với dữ liệu
  - Dữ liệu có nhiều giá trị ngoại lai.
- Khi đó,
  - Thử sử dụng các kỹ thuật tiền xử lý dữ liệu như chuẩn hóa dữ liệu, loại bỏ nhiễu, xử lý giá trị ngoại lai, hoặc
  - Hoặc thử các mô hình hồi quy khác nhau để tìm mô hình phù hợp nhất với dữ liệu, hoặc
  - Có thể điều chỉnh các tham số của mô hình hồi quy để tối ưu hóa hiệu suất

# Mean Square Error

- Mean Square Error - MSE (Sai số bình phương trung bình) là một chỉ số thống kê đo mức độ chênh lệch giữa giá trị dự đoán và giá trị thực tế trong một mô hình thống kê.
- Công thức tính: $\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$
- Ưu điểm: Dễ dàng tính toán và hiểu, có ý nghĩa thống kê rõ ràng; qua đó đo lường sai số bình phương trung bình giữa giá trị dự đoán và giá trị thực tế. Điều này giúp nắm bắt được mức độ chính xác của mô hình. Đặc biệt nhạy cảm với sai số lớn (MSE tăng lên nhanh chóng nếu có sai số lớn, giúp chúng ta phát hiện các dự đoán không chính xác).
- Hạn chế: MSE bị ảnh hưởng bởi các giá trị ngoại lệ (outliers), điều đó cho thấy nếu có nhiều ngoại lệ. Đặc biệt MSE có thể không phản ánh chính xác hiệu suất của mô hình.

# Root Mean Square Error

- Root Mean Square Error - RMSE đo lường sai số trung bình của mô hình so với dữ liệu thực tế. Giá trị RMSE càng nhỏ thì mô hình càng tốt.
- Công thức tính: $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}$
- Ưu điểm: Dễ hiểu, có đơn vị giống với đơn vị của biến tính toán sai số làm cho việc so sánh giữa các mô hình dễ dàng hơn.
- Hạn chế: Nhạy cảm với ngoại lệ thường bị ảnh hưởng bởi các giá trị ngoại lệ. Nếu có nhiều ngoại lệ, RMSE có thể không phản ánh chính xác hiệu suất của mô hình. RMSE không thể phát hiện được sai số lớn hơn, vì nó đã được chuẩn hóa bằng căn bậc hai.

# R-square

- R-square ($R^2$) là chỉ số đo lường mức độ phù hợp giữa giá trị dự đoán và giá trị thực. Giá trị của R-square dao động từ $\infty$ đến 1, trong đó giá trị càng gần 1 thì mô hình càng tốt.
- Công thức tính: $R^2 = 1 - \dfrac{\text{MSE(giá trị dự đoán, giá trị thực)}}{\text{MSE(giá trị trung bình, giá trị thực)}}$
- $R^2 = 1 - \dfrac{\sum_{i=1}^{N} (\hat{y} - y_i)^2}{\sum_{i=1}^{N} (\bar{y} - y_i)^2}$
- Giá trị âm của $R^2$ chứng tỏ mô hình hồi quy còn tệ hơn so với mô hình hồi quy đơn giản (là mô hình lấy giá trị trung bình làm giá trị dự đoán)

- Ưu điểm: Đơn giản và dễ hiểu, thể hiện tỷ lệ phần trăm biến thiên của giá trị dự đoán được giải thích bởi các biến độc lập trong mô hình hồi quy.
  - Có giá trị thông tin, cho biết mô hình có thể giải thích được bao nhiêu phần trăm sự biến thiên của dữ liệu.
  - R-Square có thể được sử dụng để so sánh các mô hình hồi quy khác nhau, giúp lựa chọn mô hình phù hợp nhất để giải thích dữ liệu.
- Hạn chế: Dễ bị ảnh hưởng bởi số lượng biến vì R-Square có xu hướng tăng khi số lượng biến độc lập trong mô hình tăng, bất kể ý nghĩa thống kê của các biến.
  - Không phản ánh độ chính xác của dự đoán vì R-Square chỉ đo lường mức độ phù hợp của mô hình với dữ liệu hiện có, không cho biết mô hình có thể dự đoán chính xác dữ liệu mới hay không.
  - Bị ảnh hưởng bởi giá trị ngoại lai vì giá trị ngoại lai có thể làm tăng hoặc giảm R-Square một cách đáng kể, ảnh hưởng đến độ tin cậy của chỉ số này.

# Thử nghiệm

- Với dataset từ https://www.kaggle.com/datasets/camnugent/california-housing-prices về giá nhà ở California
- Dữ liệu chứa thông tin từ cuộc điều tra dân số California năm 1990. Có thể không giúp bạn dự đoán giá nhà.
- Có các thành phần (các đặc trưng):
  - **longitude**: Giá trị cao hơn là xa hơn về phía Tây
  - **latitude**: Giá trị cao hơn là xa hơn về phía Bắc
  - **housing_median_age**: con số thấp hơn là một tòa nhà mới hơn

- **total_rooms**: về số phòng
- **total_bedrooms**: về số phòng ngủ
- **population: total_bedrooms**: về người cư trú
- **households**: về số hộ gia đình - một nhóm người cư trú trong một đơn vị gia đình
- **median_income**: thu nhập trung bình cho các hộ gia đình (*được đo bằng chục nghìn đô la Mỹ*)
- **median_house_value**: Giá trị trung bình cho các hộ gia đình trong nhà (*được đo bằng đô la Mỹ*)
- **ocean_proximity**: Vị trí của ngôi nhà

# Bằng Python với các thư viện cần thiết

```
### Chuẩn bị dữ liệu
import polars as pl
file = '/Users/lang/Documents/Works/Training/Mathematics for Machine Learning/housing.csv'
df = pl.read_csv( file )

# Bỏ những dòng thiếu dữ liệu
df = df.drop_nulls()

import numpy as np
X = np.array(df.drop('median_house_value','ocean_proximity'))
y = np.array(df.select('median_house_value')).ravel()

### Các bước tạo mô hình hồi quy
from sklearn.linear_model import LinearRegression
model = LinearRegression().fit(X_train, y_train)

### Sử dụng để dự đoán
y_pred = model.predict(X_test)
```

# Thử một số độ đo để đánh giá mô hình

```
print( "Mean Squared Error: %4.2f" %mean_squared_error(y_test, y_pred) )
print( "Mean Absolute Error: %4.2f" %mean_absolute_error(y_test, y_pred) )
print( "Root Mean Squared Error: %4.2f" %root_mean_squared_error(y_test, y_pred) )
print( "R2-Square: %4.2f" %r2_score(y_test, y_pred) )
```

```
Mean Squared Error: 4921881237.63
Mean Absolute Error: 51372.67
Root Mean Squared Error: 70156.12
R2-Square: 0.64
```

# Độ đo trong bài toán gom cụm

## Metrics in Clustering Problem

**Dr. Tran Van Lang, A.Prof. in Computer Science**

- Việc đánh giá chất lượng mô hình gom cụm là một bước quan trọng để đảm bảo hiệu quả và tính hữu dụng của nó. Có nhiều phương pháp khác nhau để thực hiện đánh giá này, các phương pháp có thể được chia thành hai nhóm chính như sau, hai nhóm này phân biệt qua thông tin sử dụng để đánh giá và mục tiêu đánh giá:
  - Đánh giá nội tại hay nội sinh (*Internal Metrics*): Đặc điểm chính là
    - Được sử dụng khi không có nhãn đúng (*ground truth*).
    - Đánh giá chất lượng của các cụm dựa trên cấu trúc nội tại của dữ liệu.
    - Tập trung vào các đặc điểm bên trong của mô hình gom cụm như độ chặt chẽ và độ tách biệt của các cụm.

- Đánh giá ngoại sinh (*External Metrics*): đặc điểm chính là:
  - Được sử dụng khi có nhãn đúng (*ground truth*).
  - Đánh giá chất lượng của các cụm dựa trên sự so sánh với phân cụm thật.
  - Tập trung vào mức độ tương đồng giữa phân cụm dự đoán và phân cụm thật.

- **Internal Metrics** rất hữu ích khi không có nhãn đúng và cần đánh giá mô hình dựa trên cấu trúc dữ liệu nội tại. Qua đó giúp hiểu rõ hơn về cách các cụm được hình thành và mức độ tách biệt giữa chúng.
- **External Metrics** cần thiết khi có nhãn đúng và muốn so sánh trực tiếp phân cụm dự đoán với phân cụm thật. Cung cấp cái nhìn về mức độ chính xác của mô hình so với thực tế.

- Bảng so sánh sự khác nhau

| Tiêu chí | Internal Metrics | External Metrics |
|---|---|---|
| **Thông tin sử dụng** | Chỉ dựa trên dữ liệu và phân cụm hiện tại | Sử dụng cả nhãn đúng (ground truth) |
| **Mục tiêu đánh giá** | Đánh giá độ chặt chẽ và độ tách biệt | Đánh giá mức độ tương đồng với phân cụm thật |
| **Ứng dụng chính** | Khi không có nhãn đúng | Khi có nhãn đúng |
| **Ví dụ các chỉ số** | Silhouette Score, Davies-Bouldin Index, Dunn Index | Adjusted Rand Index, Mutual Information, Homogeneity, Completeness, V-Measure |

- Internal Metrics, có các độ đo thông dụng:
  - Silhouette Index /ˌsɪl.əˈwet/
  - Davies-Bouldin Index
  - Calinski-Harabasz Index
- Còn External Metrics có:
  - Adjusted Rand Index
  - Adjusted Mutual Information
  - Homogeneity, Completeness, V-measure

# Silhouette Score

- **Silhouette Score** là chỉ số để đánh giá nội sinh, để đánh giá mức độ tương đồng của một điểm dữ liệu với cụm của nó so với các cụm khác.
- Sử dụng độ nhỏ gọn của các cụm riêng lẻ (**khoảng cách trong cụm** - *intra cluster distance*) và sự tách biệt giữa các cụm (**khoảng cách giữa các cụm** - *inter cluster distance*) để đo lường điểm đại diện tổng thể về mức độ động tác động của việc phân cụm.
- Giả sử bộ dữ liệu $D = \{D_1, D_2, \ldots, D_N\}$ gồm $N$ điểm được gom lại thành tập $U = \{U_1, U_2, \ldots, U_{N_U}\}$ gồm $N_U$ cụm ($N_U$ tập con).

# Silhouette Score

- Khoảng cách trung bình của data point thứ $D_i$ (thuộc cụm $U_I$) đến tất cả các data point còn lại trong cụm (**khoảng cách trong cụm**) là $a_i = \frac{1}{|U_I| - 1} \sum_{j \in U_I, i \neq j} d(i,j)$
  - Trong đó $|U_I|$ là số data point của cụm $U_I$, và $|U_I| - 1$ để không tính khoảng cách $d(i,i)$ trong tổng.
- $b_i$ là là ***khoảng cách giữa các cụm*** được định nghĩa là khoảng cách trung bình đến cụm gần nhất của điểm dữ liệu $i$ ngoại trừ cụm mà nó thuộc về:

$$b_i = \min_{J \neq I} \frac{1}{|U_J|} \sum_{j \in U_J} d(i,j)$$

# Silhouette Score

- Khi đó Silhouette Score /ˌsɪl.əˈwet/ của data point thứ $i$ được tính như sau:

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}, \text{ nếu } |C_I| > 1, \text{ và } s_i = 0, \text{ nếu } |C_I| = 1$$

- Hoặc có thể viết:

- $$s_i = \begin{cases} 1 - a_i/b_i, & \text{nếu } a_i < b_i \\ 0, & \text{nếu } a_i = .\, b_i \\ b_i/a_i - 1, & \text{nếu } a_i > b_i \end{cases}$$

- Silhouette Score tổng thể cho toàn bộ tập dữ liệu có thể được tính bằng cách lấy trung bình của điểm Silhouette cho tất cả các điểm dữ liệu trong tập dữ liệu.
- Như có thể thấy từ công thức trên, Silhouette Score luôn nằm trong khoảng từ -1 đến 1. Giá trị 1 biểu thị việc gom cụm tốt hơn.
- Cụ thể,
  - -1: điểm dữ liệu nằm giữa hai cụm
  - 0: điểm dữ liệu nằm trên ranh giới cụm
  - 1: điểm dữ liệu nằm gần tâm cụm của nó

# Davies-Bouldin Index

- **Davies-Bouldin Index - *DBI***: là một metric dùng để đánh giá chất lượng của các cụm trong thuật toán gom cụm.
- Chỉ số này được sử dụng để đo lường mức độ chặt chẽ (*cohesion*) trong cụm và mức độ tách biệt (*separation*) giữa các cụm.
- Giá trị của ***DBI*** thuộc $[0,\infty)$. ***DBI*** càng thấp thì chất lượng của cụm càng cao (khi DBI bằng 0 thì các cụm tách biệt hoàn toàn; giá trị cao hơn, thì các cụm chồng chéo lên nhau), nghĩa là các cụm vừa chặt chẽ trong nội bộ, vừa tách biệt rõ ràng với nhau.
- Bộ dữ liệu được phân thành $N_U$ cụm, ***DBI*** được tính như sau:

$$DBI = \frac{1}{N_U}\sum_{i=1}^{N_U} R_i, \text{ với } R_i = \max_{j\neq i} R_{ij}, \text{ và } R_{ij} = \frac{S_i + S_j}{d(i,j)}$$

- Trong đó,
  - $S_i$ là độ chặt chẽ của cụm $U_i$, thường được tính bằng độ lệch chuẩn hoặc bán kính của cụm.
    - $S_i = \sqrt[q]{\frac{1}{|U_i|}\sum_{j=1}^{|U_i|}\|D_j - C_i\|^q}$, với $C_i$ là điểm trung tâm (centroid) của cụm $U_i$
    - Chính là trung bình khoảng cách giữa các điểm dữ liệu trong cụm và tâm cụm. Một cụm có độ chặt chẽ cao (giá trị $S_i$ nhỏ) nếu các điểm dữ liệu gần với tâm cụm.

- $d(i,j)$ là khoảng cách giữa 2 cụm $U_i$, $U_j$ (là khoảng cách giữa các tâm của cụm $i$ và cụm $j$). Khoảng cách này càng lớn thì các cụm càng tách biệt.
- $R_i$ đại diện cho "độ giống nhau" lớn nhất giữa cụm $U_i$ và các cụm khác, đại diện mức độ mà cụm $U_i$ bị "gộp" với cụm gần nhất (không phải là chính nó)

- Lưu ý rằng:
  - $DBI < 1$: các cụm tách biệt tốt, không có chồng chéo đáng kể.
  - $1 \leq DBI \leq 2$: các cụm tách biệt vừa đủ.
  - $DBI > 2$: các cụm bắt đầu chồng chéo lên nhau, chất lượng mô hình phân cụm giảm.

# Calinski-Harabasz Index

- Chỉ số **Calinski-Harabasz Index - *CH*** để đo lường sự phân tán nội cụm so với sự phân tán liên cụm. Điều đó có nghĩa là Chỉ số này đo lường tỷ lệ giữa tổng phương sai giữa các cụm (*inter-cluster dispersion*) và tổng phương sai trong các cụm (*intra-cluster dispersion*)

- Với bộ dữ liệu gồm $N$ điểm $D = \{D_1, D_2, \ldots, D_N\}$ được phân thành $N_U$ cụm $\{U_1, U_2, \ldots, U_{N_U}\}$, ***CH*** được định nghĩa là tỷ lệ ***phân tách giữa các cụm*** (*between-cluster separation* - $B_U$) với sự ***phân tán trong cụm*** (*within-cluster dispersion* - $W_U$):

$$CH = \frac{\frac{B_U}{N_U - 1}}{\frac{W_U}{N - N_U}} = \frac{B_U}{W_U}\frac{N - N_U}{N_U - 1}$$

- Trong đó,
  - $B_U$: tổng số phương sai giữa các cụm (*between-cluster variance*), chính là tổng có trọng số của các khoảng cách tâm cụm ($C_i$) và tâm dữ liệu ($\mathbf{C}$): $B_U = \sum_{i=1}^{N_U} \|C_i - \mathbf{C}\|^2$
  - $W(k)$: tổng số phương sai bên trong cụm (*within-cluster variance*), là tổng của khoảng cách giữa các điểm dữ liệu và các tâm cụm tương ứng của chúng:

    $$W_U = \sum_{i=1}^{N_U} \sum_{D_k \in U_i} \|D_k - C_i\|^2$$

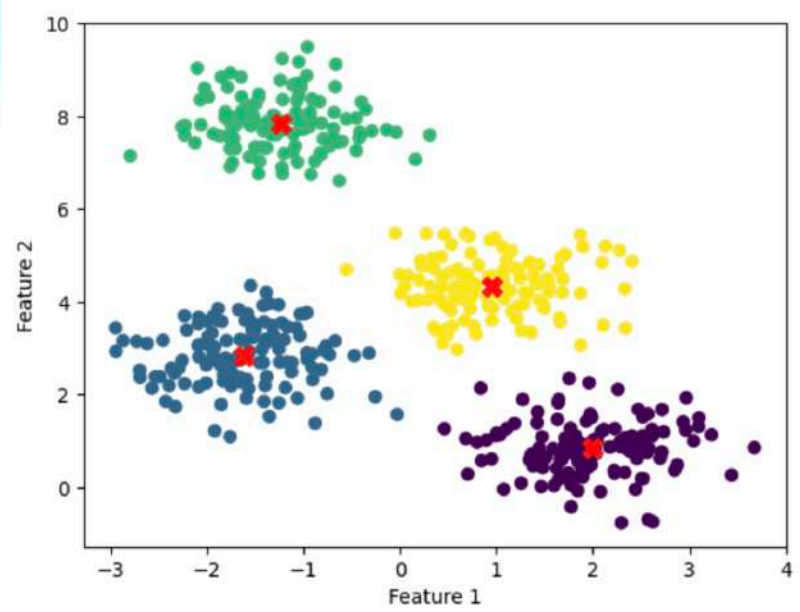
- Chỉ số ***CH*** còn được gọi là Variance Ratio Criterion.
- Giá trị của ***CH*** càng cao thì chất lượng của cụm càng tốt, tức là các cụm vừa chặt chẽ trong nội bộ, vừa tách biệt rõ ràng với nhau.
  - $CH > 1$: các cụm tách biệt tốt, không có chồng chéo đáng kể.
  - $CH \geq 0$: các cụm tách biệt ở mức độ nào đó.
  - $CH < 0$: các cụm không tách biệt, có thể cần xem xét lại số lượng cụm.

# Minh hoạ bằng Python

```
### Tạo dữ liệu mẫu
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=500, centers=4, random_state=0)

### Áp dụng KMeans để gom cụm
from sklearn.cluster import KMeans
model = KMeans(n_clusters=4, random_state=0)
y_pred = model.fit_predict(X)

### Đánh giá mô hình gom cụmg
from sklearn.metrics import *
print( "Silhouette Score: %4.2f" %silhouette_score(X, y_pred) )
print( "Davies Bouldin Score: %4.2f" %davies_bouldin_score(X, y_pred) )
print( "Calinski Harabasz Score: %4.2f" %silhouette_score(X, y_pred) )
```

### **Vẽ biểu đồ các cụm**

```
import matplotlib.pyplot as plt

plt.scatter( X[:,0],X[:,1],c=labels )
centers = model.cluster_centers_
plt.scatter( centers[:,0],centers[:,1],c='red',s=100,marker='X')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```



# Adjusted Rand Index

- **Adjusted Rand Index (*ARI*)**: So sánh độ phân bố nhãn lớp dự đoán bởi mô hình gom cụm với độ phân bố nhãn lớp thực tế. Giá trị cao hơn cho thấy sự phân cụm phù hợp với nhãn lớp ban đầu.
- ***ARI*** được tính dựa trên công thức **Rand Index** (***RI***), nhưng có điều chỉnh để loại bỏ sự ảnh hưởng của phân cụm ngẫu nhiên. Công thức tổng quát của ***ARI*** như sau:

$$ARI = \frac{RI - \mathbf{E}[RI]}{\max\{RI\} - \mathbf{E}[RI]}$$

- Trong đó,
  - $RI$ là Rand Index
  - $\mathbf{E}[RI]$ là giá trị kỳ vọng của Rand Index đối với phân cụm ngẫu nhiên.

# Rand Index

- **Rand Index** (***RI***) được sử dụng trong thống kê, và đặc biệt là trong phân cụm dữ liệu để đo sự tương đồng giữa hai phân cụm. ***RI*** được định nghĩa như sau:
  - Cho một Dataset $D = \{D_1, D_2, \ldots, D_N\}$ gồm $N$ phần tử (gọi là $N$ điểm dữ liệu) và 2 phân vùng $U$, $V$ của D gồm lần lượt có $N_U, N_V$ tập con (cụm dữ liệu), với

    $U = \{U_1, U_2, \ldots, U_{N_U}\}$ và $V = \{V_1, V_2, \ldots, V_{N_V}\}$ thì $RI = \dfrac{a+b}{a+b+c+d}$

  - Trong đó, $a$ là số cặp điểm trong $D$ mà thuộc cùng một tập con trong của cả $U$ và $V$ (là những cặp điểm mà trong cả hai kết quả phân cụm đều nằm trong cùng một cụm)

    $a = \left| \left\{ (D_i, D_j) \middle/ D_i \in U_k, D_j \in V_l, 1 \le i, j \le N, 1 \le k \le N_U, 1 \le l \le N_V \right\} \right|$

- $b$ là số cặp điểm trong $D$ mà không thuộc cùng một tập con trong cả $U$ và $V$,

  $$b = \left| \left\{ (D_i, D_j) \middle/ D_i \in U_{k_1}, D_j \in U_{k_2}, D_i \in V_{l_1}, D_j \in V_{l_2}, 1 \le i, j \le N, 1 \le k_1, k_2 \le N_U, k_1 \ne k_2, 1 \le l_1, l_2 \le N_V, l_1 \ne l_2 \right\} \right|$$

- $c$ là số cặp điểm trong $D$ mà thuộc cùng một tập con trong $U$ nhưng thuộc tập con khác nhau trong $V$,

  $$c = \left| \left\{ (D_i, D_j) \middle/ D_i, D_j \in U_k, D_i \in V_{l_1}, D_j \in V_{l_2}, 1 \le i, j \le N, 1 \le k \le N_U, 1 \le l_1, l_2 \le N_V, l_1 \ne l_2 \right\} \right|$$

- $d$ là số cặp điểm trong $D$ mà thuộc cùng một tập con trong $V$ nhưng thuộc tập con khác nhau trong $U$,

  $$d = \left| \left\{ (D_i, D_j) \middle/ D_i \in U_{k_1}, D_j \in U_{k_2}, D_i, D_j \in V_l, 1 \le i, j \le N, 1 \le k_1, k_2 \le N_U, 1 \le l \le N_V, k_1 \ne k_2 \right\} \right|$$

- $a + b$ có thể được coi là số lượng đồng thuận giữa $U$ và $V$; còn $c + d$ là số lượng không đồng thuận giữa $U$ và $V$.
- Nên mẫu số chính là tổng số cặp, ***RI*** đại diện cho tần suất xuất hiện của các đồng thuận trên tổng số cặp hoặc xác suất X và Y sẽ đồng ý về một cặp được chọn ngẫu nhiên.
- Chính vì vậy, mẫu số của $RI$ chính là $\binom{N}{2}$ hay $C_2^N$.

- Hay

$$\binom{N}{2} = \frac{N!}{2!(N-2)!} = \frac{1.2.3...(N-2)(N-1)N}{(1.2) \times (1.2.3....(N-2)}$$

$$= \frac{(N-1)N}{2} = 1 + 2 + 3 + \ldots + (N-1)$$

- Suy ra $RI = \dfrac{a+b}{\binom{N}{2}}$
- ***RI*** có giá trị từ 0 đến 1, với 0 chỉ ra rằng hai cụm dữ liệu không đồng thuận về bất kỳ cặp điểm nào và 1 chỉ ra rằng các cụm dữ liệu hoàn toàn giống nhau.

- ***RI*** được sử dụng để đo lường sự tương đồng giữa hai kết quả phân cụm $U$ với $V$ là kết quả phân cụm tham chiếu - đã biết (*ground truth*). Nên có thể đồng nhất
  - $a$ với $TP$ (số lượng cặp được dán nhãn chính xác là thuộc cùng một tập hợp con), $b$ với $TN$ (số lượng các cặp được dán nhãn chính xác là thuộc về các tập hợp con khác nhau)
  - $c$ với $FP$ và $d$ với $FN$.
- Nên $RI = \dfrac{TP + TN}{TP + TN + FP + FN}$
- Giá trị này chính độ chính xác tổng thể (***Accuracy***) trong phân loại nhị phân trên các cặp phần tử trong $D$.

# Adjusted Rand Index

- ***Chỉ số Rand điều chỉnh*** (**Adjusted Rand Index - *ARI***) là một phiên bản được điều chỉnh của chỉ số Rand, có tính đến khả năng nhóm các phần tử ngẫu nhiên, nhằm cung cấp một thước đo chính xác hơn về độ tương đồng giữa hai cụm dữ liệu.
- ARI được tính bằng cách bổ sung thêm số phần tử $n_{ij}$ nằm trong giao của cụm $i$ của $U$ và cụm $j$ của $V$ như bảng bên cạnh.

$$n_{ij} = \left| U_i \cap V_j \right|, \forall i = \overline{1,N_U}, j = \overline{1,N_V}$$

- Khi đó *ARI* được tính như sau

| | $V_1$ | $V_2$ | ... | $V_{NV}$ | Sums |
|---|---|---|---|---|---|
| $U_1$ | $n_{11}$ | $n_{12}$ | ... | $n_{1NV}$ | $a_1$ |
| $U_2$ | $n_{21}$ | $n_{22}$ | ... | $n_{2NV}$ | $a_2$ |
| ... | ... | ... | ... | ... | ... |
| $U_{NU}$ | $n_{NU1}$ | $n_{NU2}$ | ... | $n_{NUNV}$ | $a_{NU}$ |
| **Sums** | $b_1$ | $b_2$ | ... | $b_{NV}$ | |

- $$ARI = \frac{\sum_{i=1}^{N_U}\sum_{j=1}^{N_V}\binom{n_{ij}}{2} - \frac{\sum_{i=1}^{N_U}\binom{a_i}{2}\sum_{j=1}^{N_V}\binom{b_j}{2}}{\binom{N}{2}}}{\frac{1}{2}\left[\sum_{i=1}^{N_U}\binom{a_i}{2} + \sum_{j=1}^{N_V}\binom{b_j}{2}\right] - \frac{\sum_{i=1}^{N_U}\binom{a_i}{2}\sum_{j=1}^{N_V}\binom{b_j}{2}}{\binom{N}{2}}}$$

- Trong đó,
  - $a_i$ là số phần tử trong cụm $i$ của $U$
  - $b_j$ là số phần tử trong cụm $j$ của $V$

- Ví dụ minh hoạ bằng Python sử dụng cả **silhouette_score()** và **adjusted_rand_score()** trên cùng một tập dữ liệu với cùng một mô hình huấn luyện để đánh giá chất lượng của kết quả phân cụm theo các khía cạnh khác nhau.

  - Phương thức adjusted_rand_score () đo lường sự tương đồng giữa các nhãn thực sự (y_true) và các nhãn dự đoán (y_pred). Giá trị ARI dao động từ -1 đến 1, trong đó 1 biểu thị sự hoàn hảo, 0 biểu thị phân cụm ngẫu nhiên và giá trị âm biểu thị sự phân cụm tệ hơn ngẫu nhiên.

  - Phương thức silhouette_score(): Đo lường mức độ tương đồng của các điểm trong cùng một cụm so với các cụm khác. Giá trị silhouette dao động từ -1 đến 1, trong đó giá trị cao hơn biểu thị sự phân cụm tốt hơn. Silhouette score không yêu cầu nhãn thực sự (y_true).

# Minh hoạ bằng Python

```
### Áp dụng KMeans để gom cụm
from sklearn.cluster import KMeans
model = KMeans(n_clusters=4, random_state=0)
y_pred = model.fit_predict(X)

### Đánh giá nội sinh và ngoại sinh
from sklearn.metrics import *
print( "Silhouette Score: %4.2f" %silhouette_score(X,y_pred) )
print( "ARI: %4.2f" %adjusted_rand_score(y_true,y_pred) )

### Đánh giá so với mô hình gom cụm khác
from sklearn.cluster import AgglomerativeClustering, DBSCAN
agglom = AgglomerativeClustering(n_clusters=4).fit_predict(X)
dbscan = DBSCAN(eps=0.5, min_samples=5).fit_predict(X)

print( "ARI: %4.2f" %adjusted_rand_score(y_pred,agglom) )
print( "ARI: %4.2f" %adjusted_rand_score(y_pred,dbscan) )
```

# Adjusted Mutual Information

- **Adjusted Mutual Information -** ***AMI*** được hiệu chỉnh từ chỉ số ***Mutual Information*** **(*MI*)**.
- Tương tự như ***ARI*** với dataset gồm $N$ phần tử $D = \{d_1, d_2, \ldots, d_N\}$ có 2 phân cụm được dùng để so sánh là $U = \{U_1, U_2, \ldots, U_{N_U}\}$ có $N_U$ cụm, và $V = \{V_1, V_2, \ldots, V_{N_V}\}$ có $N_V$ cụm.
- Ở đây giả thiết việc phân vùng được gọi là phân cụm cứng (hard cluster), nghĩa là việc phân vùng được tách rời theo cặp: $U_i \cap U_j = \emptyset = V_i \cap V_j, \forall i, j \in [1,N]$ và

  $\cup_{i=1}^{N_U} U_i = \cup_{j=1}^{N_V} V_j = D$

- Một đối tượng được chọn ngẫu nhiên từ $D$ thì xác suất mà đối tượng rơi vào cụm $U_i$ là

$$P_{U_i} = \frac{|U_i|}{N}$$

- Khi đó Entropy liên kết với phân cụm $U$ là:

$$H(U) = -\sum_{i=1}^{N_U} P_{U_i} \log P_{U_i} = -\sum_{i=1}^{N_U} \frac{|U_i|}{N} \log \frac{|U_i|}{N}$$

  - ***Entropy*** là một khái niệm từ lý thuyết thông tin được sử dụng để đo lường mức độ không chắc chắn hoặc sự hỗn loạn của một phân phối xác suất. Trong ngữ cảnh phân cụm, entropy của một tập hợp nhãn U đo lường sự không chắc chắn liên quan đến phân phối các nhãn trong U.

- Tương tự như vậy, cho $V$ có entropy là: $H(V) = -\sum_{j=1}^{N_V} \frac{|V_j|}{N} \log \frac{|V_j|}{N}$
- Khi đó độ đo ***MI*** của 2 phân cụm $U$ và $V$ là: $MI = \sum_{i=1}^{N_U} \sum_{j=1}^{N_V} P_{U_i V_j} \log \frac{P_{U_i V_j}}{P_{U_i} P_{V_j}}$
- Trong đó $P_{U_i V_j}$ biểu thị xác suất mà một điểm thuộc về cả hai cụm là cụm $U_i$ trong $U$ và cụm $V_j$ trong $V$. Với $P_{U_i V_j} = \frac{|U_i \cap V_j|}{N}$

- Khi đó,

$$MI = \sum_{i=1}^{N_U} \sum_{j=1}^{N_V} \frac{|U_i \cap V_j|}{N} \log \frac{\frac{|U_i \cap V_j|}{N}}{\frac{|U_i|}{N} \frac{|V_j|}{N}} = \sum_{i=1}^{N_U} \sum_{j=1}^{N_V} \frac{n_{ij}}{N} \log \frac{n_{ij} N}{|U_i||V_j|}$$

- ***MI*** là một đại lượng không âm chặn trên bởi entropy $H(U), H(V)$. Từ đây ***AMI*** được tính theo kiểu ***ARI*** là $AMI = \dfrac{MI - \mathbf{E}[MI]}{\max\{H(U), H(V)\} - \mathbf{E}[MI]}$

- Trong đó

$$\mathbf{E}\{MI(U,V)\} = \sum_{i=1}^{N_U}\sum_{j=1}^{N_V}\sum_{n_{ij}=\max\{0,a_i+b_j-N\}}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N\times n_{ij}}{a_i b_j}\right)\times$$

- $$\frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!}$$
- Với
  - $a_i = \sum_{j=1}^{N_V} n_{ij}, b_j = \sum_{i=1}^{N_U} n_{ij}$

- ***AMI*** cũng như ***ARI*** cho biết mức độ tương đồng giữa các cụm thực sự và các cụm được dự đoán
  - Giá trị cao (gần 1): phân cụm dự đoán gần như giống với phân cụm thực sự.
  - Giá trị bằng 0: phân cụm dự đoán không tốt hơn phân cụm ngẫu nhiên.
  - Giá trị âm: phân cụm dự đoán tệ hơn phân cụm ngẫu nhiên.

# Homogeneity

- Homogeneity là một thước đo đánh giá mức độ mà các cụm chỉ chứa các điểm dữ liệu thuộc về một lớp duy nhất.
- Khi có hai phân cụm $U$ và $V$, có thể sử dụng công thức Homogeneity để đánh giá mức độ mà các cụm trong $U$ đồng nhất với các lớp trong $V$. Homogeneity $H$ của phân cụm $U$ so với phân cụm $V$ được định nghĩa như sau:

$$H = 1 - \frac{H(U \mid V)}{H(U)}; \text{ nếu } H(U) = 0 \text{ thì } H = 1$$

- Trong đó,
  - Entropy có điều kiện $H(U \mid V)$ là một thước đo đánh giá mức độ không chắc chắn hoặc hỗn loạn của một phân cụm $U$ khi biết phân cụm $V$.

- Còn trong ngữ cảnh phân cụm, $H(U|V)$ đo lường sự không chắc chắn của việc gán các điểm dữ liệu vào các cụm của $U$ khi biết chúng đã được gán vào các cụm của $V$.

- Entropy có điều kiện của U cho trước V được định nghĩa là:

$$H(U|V) = -\sum_{j=1}^{N_V} \frac{|V_j|}{N} \sum_{i=1}^{N_U} \frac{n_{ij}}{|V_j|} \log \frac{n_{ij}}{|V_j|}, \text{ với } n_{ij} = |U_i \cap V_j|$$

- Còn $H(U)$ là entropy của cụm $U$: $H(U) = -\sum_{i=1}^{N_U} \frac{|U_i|}{N} \log \frac{|U_i|}{N}$

# Completeness

- Homogeneity đo lường mức độ mà các cụm dự đoán chứa các điểm dữ liệu của các cụm thực sự. Còn Completeness ***C*** đo lường mức độ mà các cụm thực sự được phân bố trong các cụm dự đoán.
- Hai thước đo này giúp đánh giá tính đúng đắn của phân cụm theo hai hướng khác nhau, đảm bảo rằng cả việc gom nhóm các điểm dữ liệu trong cùng một cụm và phân phối các cụm dự đoán đều được xem xét.
- Nên $C = 1 - \frac{H(V|U)}{H(V)}$, với $H(V)$ là entropy phân cụm thực sự $V$; còn $H(V|U)$ là entropy có điều kiện của phân cụm thực sự $V$ cho phân cụm dự đoán $U$.

# V-measure

- Độ đo V-measure ***V*** là trung bình điều hòa của Homogeneity và Completeness, cung cấp một thước đo tổng hợp về chất lượng phân cụm
- Khi đó $V = \frac{2}{\frac{1}{H} + \frac{1}{C}} = 2\frac{H \times C}{H + C} = 2\frac{\left(1 - \frac{H(U|V)}{H(U)}\right)\left(1 - \frac{H(V|U)}{H(V)}\right)}{2 - \frac{H(U|V)}{H(U)} - \frac{H(V|U)}{H(V)}}$
- $V \in [0,1]$, càng gần 1 càng tốt.

# Minh hoạ bằng Python

```
### Bài toán gom cụm
from sklearn.datasets import make_blobs
from sklearn.cluster import *
from sklearn.metrics import *

# Tạo dữ liệu mẫu
X, y_true = make_blobs(n_samples=500, centers=4, random_state=0)

# Áp dụng KMeans để gom cụm
model = KMeans(n_clusters=4, random_state=0)
y_pred = model.fit_predict(X)

### Đánh giá ngoại sinh
print( "ARI: %4.2f" %adjusted_rand_score(y_true,y_pred) )
print( "AMI: %4.2f" %adjusted_mutual_info_score(y_true, y_pred) )
print( "Homogeneity: %4.2f" %homogeneity_score(y_true, y_pred) )
print( "Completeness: %4.2f" %completeness_score(y_true, y_pred) )
print( "V-measure: %4.2f" %v_measure_score(y_true, y_pred) )
```

# Nhận định

- Việc gom cụm tốt là việc phân chia tập dữ liệu thành các nhóm (cụm) sao cho các điểm dữ liệu trong cùng một cụm có độ tương đồng cao và các điểm dữ liệu thuộc các cụm khác nhau có độ khác biệt cao.
- Một mô hình gom cụm tốt sẽ đáp ứng các tiêu chí sau:
  - **Độ tương đồng nội cụm cao**: Các điểm dữ liệu trong cùng một cụm nên có độ tương đồng cao về các thuộc tính hoặc đặc điểm. Ví dụ, trong trường hợp phân cụm khách hàng, các khách hàng trong cùng một cụm có thể có hành vi mua hàng tương đồng, sở thích chung hoặc thuộc cùng một phân khúc thu nhập.
  - **Độ khác biệt liên cụm cao**: Các điểm dữ liệu thuộc các cụm khác nhau nên có độ khác biệt cao về các thuộc tính hoặc đặc điểm. Điều này giúp đảm bảo rằng các cụm được phân biệt rõ ràng và không chồng chéo lên nhau.

- **Số lượng cụm hợp lý**: Số lượng cụm nên phù hợp với bản chất dữ liệu và mục tiêu phân cụm. Nếu có quá ít cụm,có thể bỏ qua các cấu trúc quan trọng trong dữ liệu. Ngược lại, nếu có quá nhiều cụm, có thể dẫn đến việc phân chia dữ liệu quá mức và khó giải thích kết quả.
- **Sự ổn định**: Mô hình gom cụm nên tạo ra kết quả tương tự khi áp dụng cho cùng một tập dữ liệu với các giá trị khởi tạo khác nhau. Điều này đảm bảo rằng mô hình không phụ thuộc vào các yếu tố ngẫu nhiên và có thể cung cấp kết quả nhất quán.
- **Khả năng giải thích**: Mô hình gom cụm nên dễ hiểu và giải thích. Điều này giúp người dùng hiểu được cơ sở cho việc phân chia dữ liệu và có thể sử dụng kết quả phân cụm cho các mục đích tiếp theo.
- **Hiệu quả tính toán**: Mô hình gom cụm nên có thể tính toán nhanh chóng và hiệu quả, đặc biệt khi làm việc với tập dữ liệu lớn.

- Ngoài ra, việc gom cụm tốt còn phụ thuộc vào mục tiêu cụ thể của ứng dụng.
  - Ví dụ, trong trường hợp phân cụm khách hàng, có thể tập trung vào việc phân chia khách hàng thành các nhóm có hành vi mua hàng tương đồng để thực hiện các chiến dịch tiếp thị hiệu quả hơn.
  - Hoặc trong trường hợp phân cụm gen, có thể tập trung vào việc xác định các nhóm gen có chức năng hoặc liên quan đến một bệnh cụ thể.

- Ngoài ra cũng có thể xem xét các đặc điểm và tiêu chí cụ thể như sau:
  - **Độ chặt chẽ** (Cohesion)
    - Các điểm trong cùng một cụm nên gần nhau.
    - Được đo lường bằng các metric như Silhouette Score và khoảng cách trung bình giữa các điểm trong cùng cụm.
    - Cụm có độ chặt chẽ cao thường có khoảng cách nhỏ giữa các điểm trong cụm.

- **Độ tách biệt** (Separation)
  - Các cụm nên khác biệt và cách xa nhau.
  - Được đo lường bằng khoảng cách giữa các cụm hoặc các metric như Davies-Bouldin Index.
  - Cụm có độ tách biệt tốt có khoảng cách lớn giữa các điểm của các cụm khác nhau.
- **Số lượng cụm hợp lý**
  - Số lượng cụm nên phù hợp với bản chất của dữ liệu.
  - Quá nhiều cụm có thể dẫn đến việc phân mảnh (fragmentation), trong khi quá ít cụm có thể dẫn đến việc gộp nhóm các điểm không tương đồng.

- **Độ ổn định** (Stability)
  - Kết quả gom cụm nên ổn định khi áp dụng trên các mẫu dữ liệu khác nhau hoặc khi có các biến đổi nhỏ trong dữ liệu.
  - Thường được đánh giá bằng cách chạy lại thuật toán trên các bộ dữ liệu khác nhau hoặc với các khởi tạo khác nhau và kiểm tra sự đồng nhất của các kết quả.
- **Ý nghĩa thực tế** (Interpretability)
  - Các cụm nên có ý nghĩa và dễ hiểu trong ngữ cảnh của bài toán.
  - Điều này thường yêu cầu sự kiểm tra bằng mắt hoặc phân tích chủ quan bởi chuyên gia trong lĩnh vực cụ thể.