Toán học trong khoa học dữ liệu

Mathematics for Data Science

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

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 λ 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.

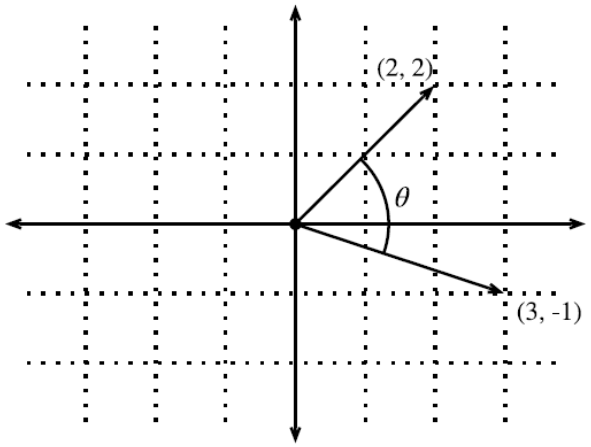
Tích vô hướng

- **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 $\lambda \in \mathbb{R}, x = (x_1, x_2, \ldots, x_n) \in R^n$, 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 giải 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

$$\begin{aligned} \|.\| : V &\to \mathbb{R} \\ x &\mapsto \|x\| \end{aligned}$$

- 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_∞:

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

```
# 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) )
```

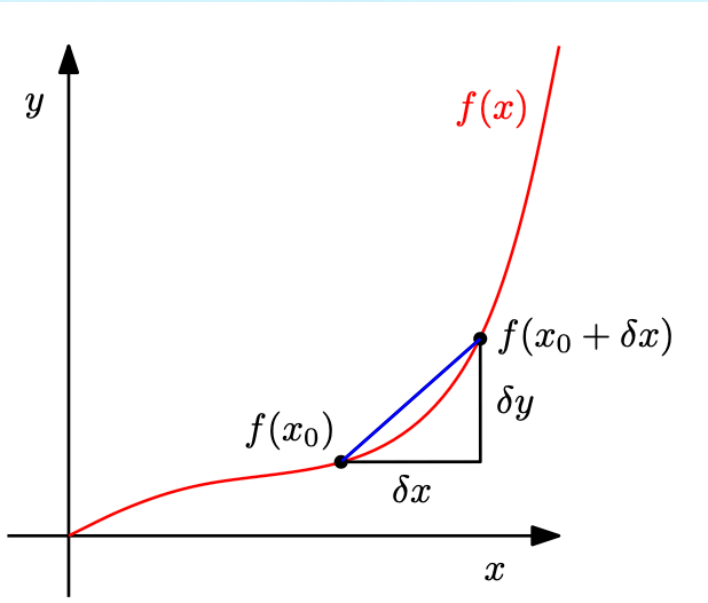
Giải tích vector

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 $f_i: \mathbb{R}^n \to \mathbb{R}, \forall i = \overline{1,n}$

- 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 Δx 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 Δx
  - 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**: Δx 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 λ 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 λ 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 λ 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 λ khá bé, và một sai số ε
  - 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

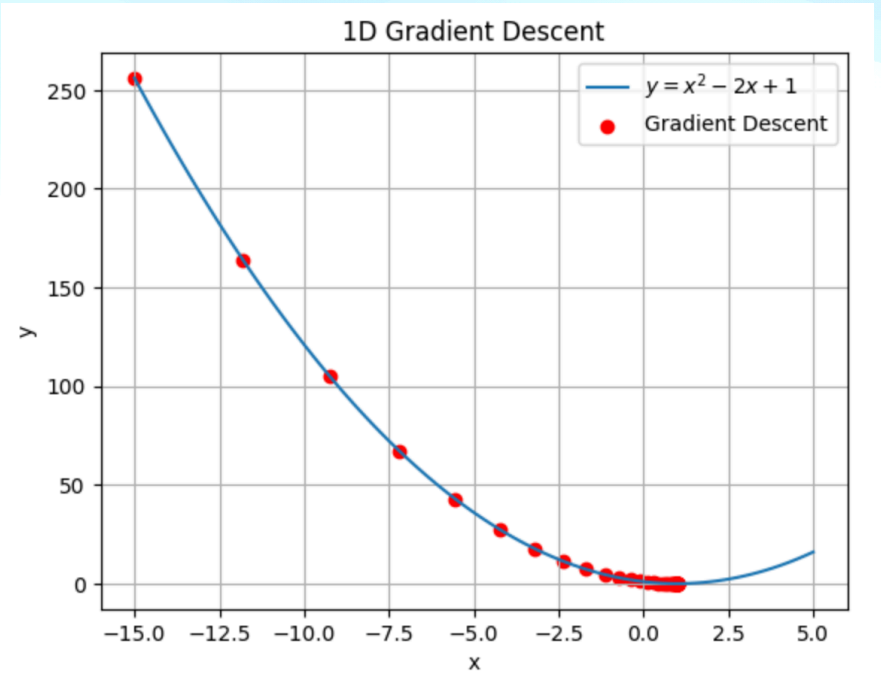
$$\begin{bmatrix} x_1^{(t+1)} \\ x_2^{(t+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix} - \lambda \begin{bmatrix} x_1^{(t)} \\ x_2^{(t)} \end{bmatrix}^T \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix} + \lambda \begin{bmatrix} 5 \\ 3 \end{bmatrix}^T, t = 0,1,...,N, \text{ và }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

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.**

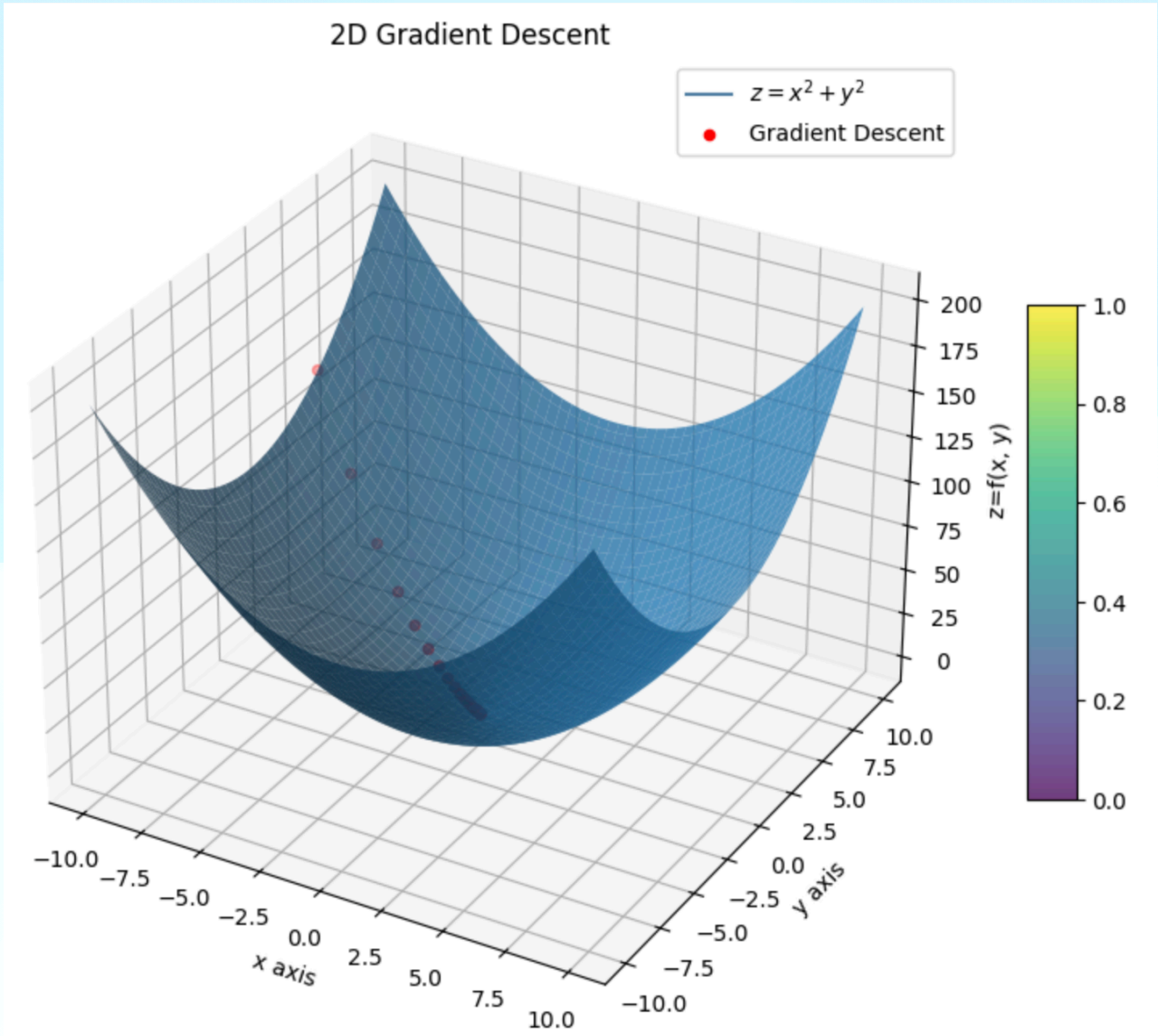
```
### f(x) = x^2 - 2x + 1

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()
```

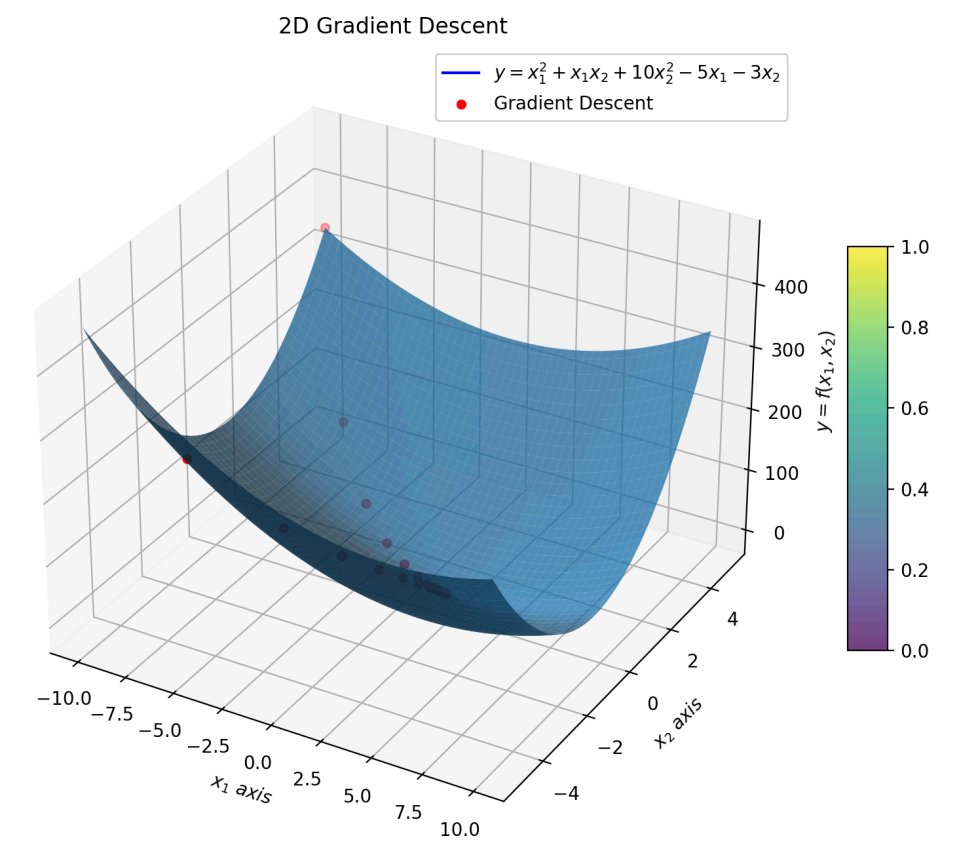


```
### f(x,y) = x^2 + y^2

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
0
25
50
75
100
125
150
175
200
0.0
0.2
0.4
0.6
0.8
1.0

```
### 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.