

## 1 定义

### 标量

标量是一个数字, 无方向。

### 向量

一行或者一列数字, 用加粗的小写字母表示。默认为列向量。  
实空间  $\mathbb{R}^n$  中的向量是  $n$  个实数的有序集合

### 矩阵

$m \times n$  的矩阵有  $m$  行  $n$  列, 每个元素都为一个小数  
矩阵通常表示为:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

方阵是行数与列数相等的矩阵。

### 特殊矩阵

对角矩阵

$$\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$$

上三角阵

$$\begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

下三角阵

$$\begin{bmatrix} a & 0 & 0 \\ b & c & 0 \\ d & e & f \end{bmatrix}$$

单位矩阵

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

对称矩阵

$$\begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

### 矩阵的行表示与列表示

$\mathbf{A}$  的第  $j$  列表示为  $a_j$  或  $A_{:,j}$ :

$$\mathbf{A} = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}$$

$\mathbf{A}$  的第  $i$  行表示为  $a_i^T$  或  $A_{i,:}$ :

$$\mathbf{A} = \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_n^T & - \end{bmatrix}$$

## 编程实现

本讲义主要介绍使用 Python 和 numpy 库来实现对矩阵的操作。首先是如何定义一个矩阵, 以及如何直接定义特殊的矩阵。

```
1 import numpy as np
2 # 定义一个向量
3 # np.array() 返回一个 ndarray 类型
4 a = np.array([1, 2, 3])
5 print(a)
6 # Output:
7 # [1 2 3]
8
9 # 定义一个普通矩阵
10 A = np.array([
11     [1, 2, 3],
12     [4, 5, 6],
13     [7, 8, 9]
14 ])
15 print(A)
16 # Output:
17 # [[1 2 3]
18 #  [4 5 6]
19 #  [7 8 9]]
20
21 # 定义一个单位矩阵
22 I = np.identity(2, dtype=int)
23 print(I)
24 # Output:
25 # [[1 0]
26 #  [0 1]]
27
28 # 定义一个对角矩阵
29 D = np.diag([1, 2, 3])
30 print(D)
31 # Output:
32 # [[1 0 0]
33 #  [0 2 0]
34 #  [0 0 3]]
35
36 # 定义一个全为0的矩阵
37 Z = np.zeros((2, 2))
38 print(Z)
39 # Output:
40 # [[0. 0.]
41 #  [0. 0.]]
42
43 # 定义一个全为1的矩阵
44 O = np.ones((2, 2))
45 print(O)
46 # Output:
47 # [[1. 1.]
48 #  [1. 1.]]
```

## 2 矩阵运算

### 2.1 转置

将矩阵的行与列翻转

例如:

$$\begin{bmatrix} a \\ b \end{bmatrix}^T = \begin{bmatrix} a & b \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

矩阵转置的性质

- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A + B)^T = A^T + B^T$

编程实现

```
1 import numpy as np
2 A = np.array([
3     [1, 2, 3],
4     [4, 5, 6],
5     [7, 8, 9]
6 ])
7 B = A.T
8 print(B)
9 # Output:
10 # [[1 4 7]
11 #   [2 5 8]
12 #   [3 6 9]]
```

## 2.2 加减

矩阵必须是相同大小，即相同行数相同列数

矩阵进行逐元素加减运算

例如：

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} + \begin{bmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix} = \begin{bmatrix} 8 & 12 \\ 11 & 15 \\ 14 & 18 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} - \begin{bmatrix} 7 & 10 \\ 8 & 11 \\ 9 & 12 \end{bmatrix} = \begin{bmatrix} -6 & -8 \\ -5 & -7 \\ -4 & -6 \end{bmatrix}$$

编程实现

```
1 import numpy as np
2 A = np.array([
3     [1, 2],
4     [3, 4],
5     [5, 6]
6 ])
7 B = np.array([
8     [7, 10],
9     [8, 11],
10    [9, 12]
11 ])
12 # 加法
13 C = A + B
14 print(C)
15 # Output:
16 # [[ 8 12]
17 #   [11 15]
18 #   [14 18]]
19
20 # 减法
21 C = A - B
22 print(C)
23 # Output:
24 # [[-6 -8]
25 #   [-5 -7]
26 #   [-4 -6]]
```

## 2.3 乘法

矩阵  $A \in \mathbb{R}^{m \times n}$  和矩阵  $B \in \mathbb{R}^{n \times p}$  的乘积为

$$C = AB \in \mathbb{R}^{m \times p}$$

其中

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$$

例如：

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \times \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} 30 & 66 \\ 36 & 81 \\ 42 & 96 \end{bmatrix}$$

矩阵能够相乘的前提条件是矩阵 A 的列数必须等于矩阵 B 的行数

矩阵乘法的性质：

- 矩阵乘法不具有交换律， $AB$  不一定等于  $BA$
- 矩阵乘法在满足前提条件下可以连乘
- 矩阵乘法具有结合律，即  $A(BC) = (AB)C$
- $(AB)^T = B^T A^T$
- $AI = IA = A$ ， $I$  为单位矩阵

矩阵乘法的特殊情况

- 标量与矩阵相乘

乘积为每个元素乘以标量的矩阵。例如：

$$3.5 \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 3.5 & 7.0 \\ 10.5 & 14.0 \\ 17.5 & 21.0 \end{bmatrix}$$

- 向量点乘（内积）

两个向量相同位置的元素乘积之和

假设

$$\mathbf{b} = \begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix} \quad \mathbf{a}^T = \begin{bmatrix} 3 & 4 & 6 \end{bmatrix}$$

它们的内积为

$$\mathbf{a}^T \mathbf{b} = \mathbf{a} \cdot \mathbf{b} = \begin{bmatrix} 3 & 4 & 6 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix} = 71 = \mathbf{b}^T \mathbf{a}$$

- 张量积（外积）

列向量乘以行向量，结果为包含两个向量中每对元素乘积的矩阵

假设

$$\mathbf{a} = \begin{bmatrix} 3 \\ 4 \\ 6 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 5 \\ 2 \\ 8 \end{bmatrix}$$

则

$$\mathbf{a} \mathbf{b}^T = \begin{bmatrix} 3 \\ 4 \\ 6 \end{bmatrix} \begin{bmatrix} 5 & 2 & 8 \end{bmatrix} = \begin{bmatrix} 15 & 6 & 24 \\ 20 & 8 & 32 \\ 30 & 12 & 48 \end{bmatrix}$$

### ▪ 矩阵与向量相乘 (1)

矩阵  $A \in \mathbb{R}^{m \times n}$  乘向量  $x \in \mathbb{R}^n$ , 乘积为  $y = Ax \in \mathbb{R}^m$   
若  $A$  用行向量表示, 则

$$y = Ax = \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_m^T & - \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_m^T x \end{bmatrix}$$

### ▪ 矩阵与向量相乘 (2)

若  $A$  用列向量表示, 则

$$y = Ax = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \begin{bmatrix} | \\ a_1 \\ | \end{bmatrix} x_1 + \begin{bmatrix} | \\ a_2 \\ | \end{bmatrix} x_2 + \cdots + \begin{bmatrix} | \\ a_n \\ | \end{bmatrix} x_n$$

可以看出,  $y$  是矩阵  $A$  的列向量的线性组合。

### ▪ 矩阵与向量相乘 (3)

向量  $x \in \mathbb{R}^m$  乘矩阵  $A \in \mathbb{R}^{m \times n}$ , 乘积为  $y = Ax \in \mathbb{R}^n$   
若  $A$  用列向量表示, 则

$$y^T = x^T A = x^T \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \cdots & a_n \\ | & | & & | \end{bmatrix}$$

$$= \begin{bmatrix} x^T a_1 & x^T a_2 & \cdots & x^T a_n \end{bmatrix}$$

### ▪ 矩阵与向量相乘 (4)

若  $A$  用行向量表示, 则

$$y^T = x^T A = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \begin{bmatrix} - & a_1^T & - \\ - & a_2^T & - \\ & \vdots & \\ - & a_m^T & - \end{bmatrix}$$

$$= x_1 \begin{bmatrix} - & a_1^T & - \end{bmatrix} + x_2 \begin{bmatrix} - & a_2^T & - \end{bmatrix} + \cdots$$

$$+ x_m \begin{bmatrix} - & a_m^T & - \end{bmatrix}$$

### 编程实现

```
1 import numpy as np
2 # 矩阵乘法
3 A = np.array([
4     [1, 4, 7],
5     [2, 5, 8],
6     [3, 6, 9]
7 ])
8 B = np.array([
9     [1, 4],
10    [2, 5],
11    [3, 6]
12 ])
13 C = A @ B
14 print(C)
15 # Output:
16 # [[30 66]
17 #  [36 81]
18 #  [42 96]]
19
```

```
20 # 标量乘矩阵
21 C = 2 * B
22 print(C)
23 # Output:
24 # [[ 2  8]
25 #  [ 4 10]
26 #  [ 6 12]]
27
28 # 向量点乘
29 a = np.array([3, 4, 6])
30 b = np.array([5, 2, 8])
31 c = a @ b
32 print(c)
33 # Output:
34 # 71
35
36 # 张量积
37 # 需要将一维ndarray转化成一维ndarray
38 c = a.reshape(3, 1) @ b.reshape(1, 3)
39 print(c)
40 # Output:
41 # [[15  6 24]
42 #  [20  8 32]
43 #  [30 12 48]]
44
45 # 矩阵乘以向量
46 c = A @ a
47 print(c)
48 # Output:
49 # [61 74 87]
```

## 2.4 范数

### 2.4.1 向量范数

向量范数是用来衡量一个向量的“长度”最常用的范数有  $L_1$  范数,  $L_2$  范数和  $L_\infty$  范数

#### ▪ $L_1$ 范数

$$\|x\|_1 = \sum_{i=1}^n |x_i|$$

#### ▪ $L_2$ 范数

$$\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{x^T x}$$

#### ▪ $L_\infty$ 范数

$$\|x\|_\infty = \max_i |x_i|$$

#### ▪ $L_p$ 范数

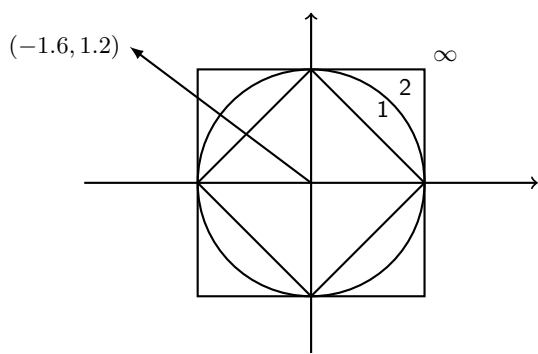
$$\|x\|_p = \left( \sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

例如有向量  $x = (-1.6, 1.2)$ , 则

$$\|x\|_1 = 2.8 \quad \|x\|_2 = 2.0 \quad \|x\|_\infty = 1.6$$

通常向量范数满足

$$\|x\|_1 \geq \|x\|_2 \geq \|x\|_\infty$$



更广义的范数是指将向量映射到实数的任何函数  $g()$ ，并满足以下条件：

- 非负性：  $\forall x \in \mathbb{R}^n, g(x) \geq 0$
- 严格正定性：  $\forall x \in \mathbb{R}^n$ ，若  $g(x) = 0$ ，则  $x = 0$
- 齐次性：  $\forall x \in \mathbb{R}^n, a \in \mathbb{R}, g(ax) = |a|g(x)$
- 三角不等式：  $\forall x, y \in \mathbb{R}^n, g(x+y) \leq g(x) + g(y)$

### 编程实现

`np.linalg.norm()` 的第一个参数为一维向量时，第二个参数便是  $L_p$  范数的  $p$

```
1 import numpy as np
2 a = np.array([-1.6, 1.2])
3 # L1 范数
4 np.linalg.norm(a, 1)
5 # Output:
6 # 2.8
7
8 # L2 范数 (默认)
9 np.linalg.norm(a)
10 # Output:
11 # 2.0
12
13 # L-infinity 范数
14 np.linalg.norm(a, np.inf)
15 # Output:
16 # 1.6
```

### 2.4.2 正交与单位正交

#### 定义：向量正交

对于向量  $u, v \in \mathbb{R}^n$ ，如果  $u \cdot v = 0$ ，且  $\|u\|_2 \neq 0, \|v\|_2 \neq 0$ ，那么称向量  $u$  和  $v$  是正交的。

#### 定义：向量单位正交

对于向量  $u, v \in \mathbb{R}^n$ ，如果  $u \cdot v = 0$ ，且  $\|u\|_2 = 1, \|v\|_2 = 1$ ，那么称向量  $u$  和  $v$  是单位正交的。

#### 定义：正交矩阵

正交矩阵是一个方阵，其元素为实数，且行向量与列向量皆为正交的单位向量。

正交矩阵具有如下性质：

- $AA^T = A^T A = I$ ，即  $A^{-1} = A^T$
- $\|Av\| = \|v\|$ ，单位矩阵乘一个向量不改变结果向量的长度，可以理解为旋转操作

- $A^{-1}$  和  $A^T$  也为正交矩阵

- $|A| = \pm 1$

### 2.4.3 矩阵范数

此部分仅介绍一些常见的矩阵范数。

- $p$ -范数诱导的矩阵范数  
也称为诱导  $p$ -范数

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

当  $p = 1$  时：

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|$$

可以看出矩阵的 1-范数就等于每列元素绝对值之和的最大值。

当  $p = \infty$  时：

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|$$

可以看出矩阵的  $\infty$ -范数就等于每行元素绝对值之和的最大值。

当  $p = 2$  (欧几里得范数) 时，诱导的矩阵范数就是谱范数。矩阵  $A$  的谱范数是  $A^T A$  的最大特征值的平方根，特征值会在后面的部分介绍

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$$

- F-范数

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$$

F-范数通常比诱导范数容易计算，因此也较为常用。

### 编程实现

`np.linalg.norm()` 的第一个参数为二维矩阵时，第二个参数是诱导  $p$ -范数的  $p$

```
1 import numpy as np
2 a = np.array([
3     [0, 3, 4],
4     [1, 6, 4]
5 ])
6 # 诱导1范数
7 np.linalg.norm(a, 1)
8 # Output:
9 # 9.0
10
11 # 诱导infinity范数
12 np.linalg.norm(a, np.inf)
13 # Output:
14 # 11.0
15
16 # 诱导2范数
17 np.linalg.norm(a, 2)
18 # Output:
19 # 8.704570789056772
20
21 # F-范数 (默认)
22 np.linalg.norm(a)
23 # Output:
24 # 8.831760866327848
```

## 2.5 行列式

矩阵  $A$  的行列式记为  $|A|$  或  $\det(A)$

### 定义：行列式的直接定义

一个  $n$  阶方阵  $A$  的行列式为

$$\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^n a_{i, \sigma(i)}$$

其中  $S_n$  是  $(1, 2, \dots, n)$  的全排列的集合， $\text{sgn}(\sigma)$  表示排列  $\sigma$  的逆序符号，即逆序为偶数时为  $+1$ ，逆序为奇数时为  $-1$ 。

例如 2 阶方阵  $A$  的行列式为

$$|A| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$$

### 定义：余子式

对一个  $n$  阶行列式  $M$ ，划去  $M$  的第  $i$  行和第  $j$  列元素后形成的  $n-1$  阶行列式叫作  $M$  关于元素  $m_{ij}$  的余子式，记作  $M_{ij}$ 。

### 定义：代数余子式

$M$  关于元素  $m_{ij}$  的代数余子式记作  $C_{ij}$ 。

$$C_{ij} = (-1)^{(i+j)} \cdot M_{ij}$$

由此我们可以得到行列式的递归定义

### 定义：行列式的递归定义

一个  $n$  阶行列式  $M$ ，

当  $n=1$  时， $M = m_{11}$ ；

当  $n \geq 2$  时， $M = \sum_{i=1}^n m_{ik} C_{ik}$ ， $1 \leq k \leq n$

行列式的性质：

- $\det(I_n) = 1$
- $\det(A^T) = \det(A)$
- $\det(A^{-1}) = [\det(A)]^{-1}$
- $\det(AB) = \det(A) \times \det(B)$

```
1 import numpy as np
2 # 求行列式
3 a = np.array([
4     [1, 2],
5     [3, 4]
6 ])
7 b = np.linalg.det(a)
8 print(b)
9 # Output:
10 # -2.0000000000000004
```

## 2.6 逆矩阵

### 定义：逆矩阵

给定一个  $n$  阶方阵  $A$ ，若存在一  $n$  阶方阵  $B$ ，使得  $AB = BA = I_n$ ，则称  $A$  是可逆的，且  $B$  是  $A$  的逆矩阵，记作  $A^{-1}$ 。

### 2.6.1 可逆矩阵的性质

- 若方阵  $A$  可逆，则称  $A$  为非奇异方阵，反之则为奇异方阵
- $A$  可通过初等行变换转化为单位矩阵
- $A$  可通过初等列变换转化为单位矩阵
- $|A| \neq 0$
- $A$  满秩（秩会在后面部分介绍），即  $\text{rank}(A) = n$
- $(A^{-1})^{-1} = A$
- $k$  是不为 0 的标量， $(kA)^{-1} = k^{-1}A^{-1}$
- $(A^T)^{-1} = (A^{-1})^T$
- $A$  和  $B$  都是  $n \times n$  的可逆矩阵，则  $(AB)^{-1} = B^{-1}A^{-1}$
- $|A^{-1}| = |A|^{-1}$

### 2.6.2 逆矩阵求法

#### 伴随矩阵法

##### 定义：伴随矩阵

一个  $n$  阶方阵  $A$  第  $i$  行第  $j$  列的代数余子式为  $C_{ij}$ ，则  $A$  的伴随矩阵的第  $j$  行第  $i$  列的元素值等于  $C_{ij}$ ，记作  $A^*$ 。

如果矩阵  $A$  可逆，则

$$A^{-1} = \frac{A^*}{|A|}$$

通过计算伴随矩阵和行列式，可以求得逆矩阵。例如

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

则

$$A^* = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \quad |A| = ad - bc$$

$$A^{-1} = \begin{bmatrix} \frac{d}{ad-bc} & \frac{-b}{ad-bc} \\ \frac{-c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix}$$

#### 初等变换法

如果矩阵  $A$  和  $B$  互逆，则有  $AB = BA = I$ ，根据可逆矩阵的性质， $A$  和  $B$  均可通过初等行变换或初等列变换变为单位矩阵。由于对矩阵  $A$  施以初等行变换（初等列变换）就相当于在  $A$  的左边（右边）乘以相应的初等矩阵，所以我们可以同时对  $A$  和  $I$  施以相同的初等行变换（初等列变换）。这样当  $A$  被变为  $I$  时， $I$  就被变为  $A$  的逆矩阵  $B$ 。

例如

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

对  $A$  和  $I$  一起进行初等行变换

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{r_2 - 3r_1} \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -3 & 1 \end{bmatrix} \xrightarrow{-\frac{1}{2}r_2} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} \xrightarrow{r_1 - 2r_2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

$$\text{则 } A^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

### 2.6.3 逆矩阵用于求线性方程组

解线性方程组

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

可以表示成矩阵乘法

$$Ax = b$$

其中

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

若矩阵  $A$  可逆，则等式两边同左乘一个  $A^{-1}$

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

由此得到

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

例如求

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

那么解为：

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} = \begin{bmatrix} -4 \\ \frac{9}{2} \end{bmatrix}$$

### 2.6.4 伪逆矩阵

对于非方阵的矩阵，存在广义逆矩阵，也成为伪逆。此部分仅介绍单边逆矩阵。对于一个  $m \times n$  的矩阵  $A$  且为满秩矩阵，若  $m > n$  则用左逆矩阵，若  $m < n$  则用右逆矩阵。

- 左逆矩阵为  $A_l^{-1} = (A^T A)^{-1} A^T$ ，也就是  $A_l^{-1} A = I_n$
- 右逆矩阵为  $A_r^{-1} = A^T (A A^T)^{-1}$ ，也就是  $A A_r^{-1} = I_m$

编程实现

```
1 import numpy as np
2
3 # 求逆矩阵
4 a = np.array([
5     [1, 2],
6     [3, 4]
7 ])
8 b = np.linalg.inv(a)
9 print(b)
```

```
10 # Output:
11 # [[-2.   1. ]
12 #  [ 1.5 -0.5]]
13
14 # 求伪逆矩阵
15 a = np.array([
16     [1, 2],
17     [3, 4],
18     [5, 6]
19 ])
20 b = np.linalg.pinv(a)
21 print(b)
22 # Output:
23 # [[-1.33333333 -0.33333333  0.66666667]
24 #  [ 1.08333333  0.33333333 -0.41666667]]
```

## 3 矩阵的秩

### 3.1 向量线性相关性

定义：向量线性相关与线性不相关

如果等式

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n = 0 \quad (1)$$

只有在  $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$  时才成立，那么这一组向量  $\{x_1, x_2, \dots, x_n\}$  之间线性不相关。反之，如果  $\exists i, \alpha_i \neq 0$  使得等式 (1) 成立，那么这组向量之间线性相关。

向量的线性相关性可以通过对公式 (1) 进行行化简来验证。假设我们有两个向量

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} -3 \\ 2 \end{bmatrix}$$

和两个对应的线性变换系数  $\alpha_1, \alpha_2$ 。我们解线性方程组

$$\begin{bmatrix} 1 & -3 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

化简为

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

所以， $\alpha_1 = \alpha_2 = 0$ 。那么我们就验证得到  $x_1$  与  $x_2$  线性不相关。

### 3.2 矩阵的秩

定义：矩阵的秩

矩阵的秩就是矩阵中线性不相关的列（行）向量的个数。

根据选择行向量或列向量的不同，矩阵的秩也分为列秩和行秩两种。以上一节中的两个向量为例，假设  $x_1$  和  $x_2$  作为组成一个矩阵的两个列

$$X = \begin{bmatrix} 1 & -3 \\ 1 & 2 \end{bmatrix}$$

由于矩阵  $X$  的两个列线性不相关，那么  $X$  的列秩就为 2。行秩与列秩类似，需要考虑线性不相关的行向量的个数。

一般来说，对于任意大小的一个矩阵，它的行秩与列秩是相等的。我们通常通过行化简来获得列向量的线性不相关性，也就是用列秩来代表矩阵的秩。矩阵  $X$  的秩表示为  $\text{rank}(X)$ 。特别地，对于一个  $m \times n$  大小的矩阵  $X \in \mathbb{R}^{m \times n}$ ，如果  $\text{rank}(X) = \min(m, n)$ ，那么我们就说矩阵  $X$  是满秩的。

一个矩阵的秩还有另外一种定义，即行列式非 0 的最大子方阵的行数。假设我们有一个矩阵

$$A = \begin{bmatrix} 4 & 5 & 2 & 14 \\ 3 & 9 & 6 & 21 \\ 8 & 10 & 7 & 28 \\ 1 & 2 & 9 & 5 \end{bmatrix}$$

因为  $\det(A) = 0$  而

$$\det \begin{pmatrix} \begin{bmatrix} 4 & 5 & 2 \\ 3 & 9 & 6 \\ 8 & 10 & 7 \end{bmatrix} \end{pmatrix} = 63 \neq 0$$

所以  $\text{rank}(A) = 3$ 。

### 3.3 矩阵的秩的相关性质

对于矩阵  $A \in \mathbb{R}^{m \times n}$  和方阵  $B \in \mathbb{R}^{n \times n}$ ,

- $\text{rank}(A) \leq \min(m, n)$ 。
- $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$
- $\text{rank}(B) = n$  当且仅当  $B$  不是奇异矩阵，即  $B$  可逆。
- $\text{rank}(B) = n$  当且仅当  $\det(B) \neq 0$ ，即  $B$  满秩。

### 3.4 用线性方程组解释矩阵的秩

矩阵的秩表示的是它对应的多元方程组中真正有用的方程个数。我们考虑这样一个有  $n$  个变量， $n$  个方程的方程组

$$\begin{cases} \alpha_{11}x_1 + \alpha_{12}x_2 + \cdots + \alpha_{1n}x_n = b_1 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \\ \alpha_{n1}x_1 + \alpha_{n2}x_2 + \cdots + \alpha_{nn}x_n = b_n \end{cases} \quad (2)$$

众所周知，要解出  $n$  个变量  $x_1, \cdots, x_n$ ，我们至少需要  $n$  个方程。然而有时候这  $n$  个方程并不是都有用。考虑如下这个方程组

$$\begin{cases} x_1 - 1x_2 - 3x_3 - x_4 = 1 \\ x_1 - x_2 + 2x_3 - x_4 = 3 \\ 4x_1 - 4x_2 + 3x_3 - 2x_4 = 10 \\ 2x_1 - 2x_2 - 11x_3 + 4x_4 = 0 \end{cases} \quad (3)$$

这个方程组通过变量替换可以化简为

$$\begin{cases} x_1 - 1x_2 - 3x_3 - x_4 = 1 \\ 5x_3 - 2x_4 = 2 \\ 0 = 0 \\ 0 = 0 \end{cases} \quad (4)$$

因此这个线性方程组中虽然有四个方程，但其中只有两个方程有用。因为我们所知道的有用的方程数少于未知的变量数，这个方程组有无穷多个解。回顾我们之前所说的验证向量线性不相关的过程，其实就是在排除“无用”的方程。具体来说，考虑方程组 (3) 中前两个方程式，并将它们表示成向量

$$v_1 = \begin{bmatrix} 1 \\ -1 \\ -3 \\ -1 \end{bmatrix} \quad \text{和} \quad v_2 = \begin{bmatrix} 1 \\ -1 \\ 2 \\ -1 \end{bmatrix}$$

解  $k_1v_1 + k_2v_2 = 0$  可得  $k_1 = k_2 = 0$ ，这说明这两个列向量之间线性不相关。而如果我们考虑前三个或前四个方程式对应的列向量，可以证明  $-v_1 - 3v_2 + v_3 = 0$  与  $-3v_1 + v_2 + 0v_3 + v_4 = 0$ ，这说明前三个列向量或前四个列向量组成的向量组是线性相关的。因此通过验证线性不相关性，我们证明了方程组 (3) 中只有两个方程包含了不同的信息，也就是只有两个方程式有用的。这与我们用变量替换解方程组得到的结果相同。

在之前我们就已经说过，矩阵的秩就是它线性不相关的列向量的个数（在这个例子里是 2）。这说明一个矩阵的秩体现的是它对应的方程组中有用的方程数量。因此矩阵的秩说明了线性方程组的可解性以及解的唯一性。

### 3.5 编程实现

在 Python 中，numpy 库提供了相应的函数：

```
1 import numpy as np
2 A = np.array([
3     [4,5,2,14],
4     [3,9,6,21],
5     [8,10,7,28],
6     [1,2,9,5]
7 ])
8 print('rank(A):', np.linalg.matrix_rank(A))
9 # Output: rank(A): 3
10
```

## 4 特征值以及特征向量

### 4.1 特征值，特征向量

#### 定义：矩阵特征值以及特征向量

对于一个方阵  $A \in \mathbb{R}^{n \times n}$ ，存在一个向量  $x \in \mathbb{R}^n$  以及一个标量  $\lambda$ ，使得

$$Ax = \lambda x \quad (5)$$

其中  $x$  称为特征向量， $\lambda$  称为特征值。

一个矩阵的特征值需要通过解  $\det(A - \lambda I) = 0$  来求得。

假设我们有一个矩阵

$$A = \begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}$$



则

$$\det \begin{pmatrix} 0.8 - \lambda & 0.3 \\ 0.2 & 0.7 - \lambda \end{pmatrix} = \lambda^2 - \frac{3}{2}\lambda + \frac{1}{2} = (\lambda - 1)(\lambda - \frac{1}{2})$$

因此，矩阵  $A$  有两个特征值  $\lambda_1 = 1, \lambda_2 = \frac{1}{2}$ 。然后通过公式 (5) 可以解得对应的特征向量。

$$(A - \mathbf{I})x_1 = \begin{bmatrix} 0.8 - 1 & 0.3 \\ 0.2 & 0.7 - 1 \end{bmatrix} x_1 = 0$$
$$(A - \frac{1}{2}\mathbf{I})x_2 = \begin{bmatrix} 0.8 - 0.5 & 0.3 \\ 0.2 & 0.7 - 0.5 \end{bmatrix} x_2 = 0$$

解得  $x_1 = [0.6, 0.4]^\top, x_2 = [1, -1]^\top$ 。

## 4.2 特征值及特征向量的性质

对于一个方阵  $A \in \mathbb{R}^{n \times n}$  和它的特征值  $\lambda$  以及对应的特征向量  $x$ ，有如下几个性质：

- $A^k x = \lambda^k x$ 。也就是说  $A$  的  $k$  次方 ( $k > 0$ ) 与  $A$  有相同的特征向量，但是特征值是  $A$  的特征值的  $k$  次方。
- 如果  $A$  的特征值都非 0，则  $A^{-1}$  的特征值为  $\frac{1}{\lambda}$ 。
- 如果  $Ax = \lambda x$ ，那么  $(A + c\mathbf{I})x = (\lambda + c)x$ 。
- $A$  可逆当且仅当它的所有特征值都非 0。

## 4.3 特征值及特征向量的简单解释

对于一个向量  $x$ ，乘以矩阵  $A$  就是对向量  $x$  在空间中做变换。一般来说，对向量进行空间变换都会使得向量的长度和方向发生改变。但是如  $Ax = \lambda x$  所表示的，有一些  $x$  做空间变换之后方向并没有改变，只是长度发生了变化，其中  $\lambda$  就是长度变化的系数。 $A$  的特征向量体现的是  $A$  的变换效果最强的那几个方向。因此我们可以认为这些特征向量代表了变换矩阵  $A$ ，所以称它们为“特征”向量。一般来说，我们可能会选择特征值较大几个特征向量来进行研究。

## 4.4 编程实现

numpy 同样提供了求矩阵特征值的函数：

```
1 import numpy as np
2 A = np.array([
3     [0.8, 0.3],
4     [0.2, 0.7]
5 ])
6 eigenval = np.linalg.eigvals(A)
7 print('Eigenvalues:', eigenval)
8 # Output: Eigenvalues: [1.  0.5]
9
```

## 5 矩阵的迹

### 5.1 矩阵的迹

#### 定义：矩阵的迹

对于一个方阵  $A \in \mathbb{R}^{n \times n}$ ，它的迹定义为对角线上所有元素之和，表示为

$$\text{tr}(A) = \sum_{i=1}^n A_{ii} \quad (6)$$

比如，对于矩阵

$$A = \begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}$$

有  $\text{tr}(A) = 0.8 + 0.7 = 1.5$ 。除了上述的定义之外，一个方阵的迹还等于这个方阵的特征值之和。即假设方阵  $A$  有  $N$  个特征值  $\lambda_1, \dots, \lambda_N$ ，则

$$\text{tr}(A) = \sum_{k=1}^N \lambda_k \quad (7)$$

对于矩阵  $A = \begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}$ ，我们已经证明了它有两个特征值  $\lambda_1 = 1, \lambda_2 = 0.5$ ，它的迹为  $\text{tr}(A) = 1 + 0.5 = 1.5$ 。

### 5.2 矩阵的迹的性质

对于方阵  $A, B, C, D \in \mathbb{R}^{n \times n}$ ，矩阵的迹有如下几个性质：

- $\text{tr}(A + B) = \text{tr}(A) + \text{tr}(B)$ 。
- $\text{tr}(cA) = c\text{tr}(A)$ 。
- 一个矩阵和它的转置的迹相等： $\text{tr}(A) = \text{tr}(A^T)$ 。
- $\text{tr}(AB) = \text{tr}(BA)$ ，但是  $\text{tr}(AB) \neq \text{tr}(A)\text{tr}(B)$ 。
- $\text{tr}(A^T B) = \text{tr}(AB^T) = \text{tr}(B^T A) = \text{tr}(BA^T)$ 。
- 循环性质： $\text{tr}(ABCD) = \text{tr}(BCDA) = \text{tr}(CDAB) = \text{tr}(DABC)$ 。

### 5.3 编程实现

利用 numpy 提供的函数计算一个方阵的迹的代码实现如下：

```
1 import numpy as np
2 A = np.array([
3     [0.8, 0.3],
4     [0.2, 0.7]
5 ])
6 print('tr(A):', np.trace(A))
7 # Output: tr(A): 1.5
8
```



## 6 矩阵对角化

### 6.1 可对角矩阵

如果一个矩阵是对角矩阵（即除了对角线上的元素，其他元素都为0），那么与这个矩阵的相关运算会变得非常简单。因此，在线性代数中，为了后续运算的方便，我们可以先将一个矩阵转化为对角矩阵。然而并不是所有矩阵都可以转化为对角矩阵。我们先定义什么样的矩阵可以转化为对角形式。

#### 定义：可对角矩阵

对于一个方阵  $A \in \mathbb{R}^{n \times n}$ ，如果存在一个可逆矩阵  $P$  使得  $P^{-1}AP$  是对角矩阵，那么就称矩阵  $A$  是可对角化的。

### 6.2 矩阵对角化

矩阵的对角化需要用到特征值及特征向量。假设我们有一个  $n$  阶方阵，并且这个方阵有  $n$  个不同的特征值  $\lambda_1, \lambda_2, \dots, \lambda_n$  以及对应的特征向量  $x_1, x_2, \dots, x_n$ 。那么矩阵  $A$  对角化时对应的变换矩阵  $P$  的形式为以特征向量为列组成的矩阵

$$P = [x_1 | x_2 | \dots | x_n]$$

我们以矩阵

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

为例。矩阵  $A$  具有三个不同的特征值  $\lambda_1 = 3, \lambda_2 = 2$  和  $\lambda_3 = 1$ 。相应的特征向量为

$$x_1 = \begin{bmatrix} -1 \\ -1 \\ 2 \end{bmatrix}, \quad x_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad x_3 = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}$$

现在设  $P$  是以这些特征向量作为列的矩阵，则  $P$  对角化了  $A$ 。通过简单的计算我们可以验证

$$P^{-1}AP = \begin{bmatrix} 0 & -1 & 0 \\ 2 & 0 & 1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 \\ 0 & 3 & 0 \\ 2 & -4 & 2 \end{bmatrix} \begin{bmatrix} -1 & 0 & -1 \\ -1 & 0 & 0 \\ 2 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

### 6.3 矩阵对角化的本质

假设矩阵  $A \in \mathbb{R}^{n \times n}$  对角化后的结果为  $\tilde{A} = P^{-1}AP$ 。矩阵  $A$  是在一组标准正交基  $(1, 0, \dots, 0), (0, 1, 0, \dots, 0), \dots, (0, \dots, 0, 1)$  上定义的。而经过变换后， $\tilde{A}$  是在由特征向量组成的基  $x_1, x_2, \dots, x_n$  上所定义的。矩阵  $P$  就是基变换矩阵。因此，原矩阵  $A$  和对角化的矩阵  $\tilde{A}$  是同一种线性变化在不同基上的描述。在用后一种基描述线性变换时，这个线性变换就只是伸缩变换。当  $A$  没有  $n$  个线性无关的特征向量时，这些特征向量就不能作为  $\mathbb{R}^n$  空间中的一组基。

## 6.4 编程实现

numpy 同样没有提供对矩阵直接进行对角化的函数，但是利用特征值以及特征向量我们可以对矩阵进行对角化。注意在代码的第 8 行，numpy 计算得到的特征向量并非普通意义上的特征向量，所以利用这些特征向量来对角化得到的是上三角矩阵。因此在第 10 行中对于  $P^{-1}AP$  计算得到的结果，需要提取对角线上的元素来完成对角化的过程。

```
1 import numpy as np
2
3 A = np.array([
4     [1, 2, 0],
5     [0, 3, 0],
6     [2, -4, 2]
7 ])
8 eigval, eigvec = np.linalg.eig(A)
9
10 diag_A = np.diag(
11     np.linalg.inv(eigvec) @ np.diag(eigval) @ eigvec
12 )
13 print('对角化后的矩阵为: \n', np.diag(diag_A))
14 # Output: 对角化后的矩阵为:
15 #         [[3.  0.  0.]
16 #          [0.  2.  0.]
17 #          [0.  0.  1.]
18
```

## 7 矩阵的正定性

### 7.1 正定矩阵

#### 定义：正定矩阵

一个矩阵  $A \in \mathbb{R}^{n \times n}$ ，如果对于任何非 0 向量  $x$ ，都满足  $x^T Ax > 0$ ，则称矩阵  $A$  为正定矩阵。

一个正定矩阵的特征值都为正，因此可以通过计算特征值来判断一个矩阵是否正定。还是以矩阵  $A = \begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}$  为例，由于它的特征值为 1 和 0.5，则矩阵  $A$  是一个正定矩阵。

另外还有一种方法，可以通过计算矩阵的各阶顺序主子式的行列式来判断正定性。如果一个矩阵的各阶顺序主子式都大于 0，则矩阵正定。以上述矩阵  $A$  为例，它的一阶顺序主子式为

$$\det([0.8]) = 0.8 > 0$$

，它的二阶顺序主子式为

$$\det\left(\begin{bmatrix} 0.8 & 0.3 \\ 0.2 & 0.7 \end{bmatrix}\right) = 0.5 > 0$$

这种正定判别方法也可以说明一个正定矩阵的行列式一定大于 0。

### 7.2 半正定矩阵

半正定矩阵的定义与正定矩阵类似。

#### 定义：半正定矩阵

一个矩阵  $A \in \mathbb{R}^{n \times n}$ ，如果对于任何非 0 向量  $x$ ，都满足  $x^T Ax \geq 0$ ，则称矩阵  $A$  为半正定矩阵。

一个半正定矩阵的特征值都非负。

### 7.3 正定/半正定矩阵性质

对于正定矩阵，有以下性质：

- 如果矩阵  $A, B$  都为正定矩阵，那么矩阵  $A + B$  也是正定矩阵。
- 如果  $A$  是正定矩阵，则存在一个矩阵  $C$ ，满足  $A = C^2$ 。
- 如果  $A$  是正定矩阵，则矩阵的  $k$  次幂  $A^k$  也是正定矩阵。
- 如果  $A$  是正定矩阵，则存在一个可逆矩阵  $C$ ，使得  $B = C^T A C$  也是正定矩阵。

类似的，对于半正定矩阵，有如下性质：

- 如果矩阵  $A, B$  都为半正定矩阵，那么矩阵  $A + B$  也是半正定矩阵。
- 如果  $A$  是半正定矩阵，则存在一个非负实数  $h$ ，使得  $B = hA$  也为半正定矩阵。

### 7.4 矩阵正定性的直观理解

正定矩阵可以类于是实数空间中的正数，而半正定矩阵则可类比为实数空间中的非负数。假设有一个正定矩阵  $A$ ，向量经过  $A$  的空间变化后成为  $Y = AX$ 。我们可以算出变化后的向量  $Y$  与原向量之间的夹角的余弦值为  $\cos(\theta) = \frac{X^T Y}{\|X\|_2 \cdot \|Y\|_2} = \frac{X^T A X}{\|X\|_2 \cdot \|AX\|_2} > 0$ 。这说明任意一个向量经过  $A$  的变换之后与原向量的夹角小于  $90^\circ$ 。

在实数空间中，正数乘以任何一个数会产生“正向”的扩大效应。相对应地，一个正定矩阵会对向量产生“正向”的变换。半正定矩阵与非负数的关系也类似。

### 7.5 编程实现

numpy 并未直接提供判断矩阵正定性的函数，但是根据相关性质，我们可以通过判断特征值的正负性来判断正定性：

```
1 import numpy as np
2
3 # 判断矩阵是否为正定矩阵
4 def isPositiveDefinite(mat):
5     return np.all(np.linalg.eigvals(mat) > 0)
6
7 # 判断矩阵是否为半正定矩阵
8 def isPositiveSemidefinite(mat):
9     return np.all(np.linalg.eigvals(mat) >= 0)
10
11 A = np.array([
12     [0.8, 0.3],
13     [0.2, 0.7]
14 ])
15 B = np.array([
16     [0, 0],
17     [0, 2]
18 ])
19
20 print('A是正定矩阵:', isPositiveDefinite(A))
21 print('A是半正定矩阵:', isPositiveSemidefinite(A))
22 print('B是正定矩阵:', isPositiveDefinite(B))
23 print('B是半正定矩阵:', isPositiveSemidefinite(B))
24 # Output: A是正定矩阵: True
25 #         B是半正定矩阵: True
```

```
26 # B是正定矩阵: False
27 # B是半正定矩阵: True
28
```

## 8 格拉姆矩阵 (Gram matrix)

### 定义：格拉姆矩阵

对于一组向量  $x_1, x_2, \dots, x_n$ ，它们对应的格拉姆矩阵为

$$G(x_1, \dots, x_n) = \begin{bmatrix} \langle x_1, x_1 \rangle & \langle x_1, x_2 \rangle & \cdots & \langle x_1, x_n \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle x_n, x_1 \rangle & \langle x_n, x_2 \rangle & \cdots & \langle x_n, x_n \rangle \end{bmatrix} \quad (8)$$

其中  $\langle \cdot, \cdot \rangle$  是求两个向量的内积。

格拉姆矩阵  $G$  是半正定的。只有向量组  $x_1, \dots, x_n$  之间是线性不相关时，矩阵  $G$  才是正定的，即  $\det(G) > 0$ 。因此，我们可以用格拉姆矩阵的行列式来判断向量组的线性不相关性。

## 9 向量/矩阵求导

### 9.1 标量关于标量的导数

我们先回顾一下求一个标量函数关于标量的导数。

### 定义：标量关于标量的导数

假设有函数  $f(x) : \mathbb{R} \mapsto \mathbb{R}$ ，则函数  $f(x)$  关于  $x$  的导数可以表示为

$$f'(x) = \frac{\partial f(x)}{\partial x} \quad (9)$$

以  $f(x) = x^2$  为例，我们有  $f'(x) = \frac{\partial f(x)}{\partial x} = 2x$ 。 $f'(x)$  有许多性质，常用的有如下几个：

- $\forall \alpha \in \mathbb{R}, \frac{\partial \alpha f(x)}{\partial x} = \alpha \frac{\partial f(x)}{\partial x}$ 。
- 如果有另一个函数  $g(x) : \mathbb{R} \mapsto \mathbb{R}$ ，则  $\frac{\partial f(x) + g(x)}{\partial x} = \frac{\partial f(x)}{\partial x} + \frac{\partial g(x)}{\partial x}$ 。
- $\frac{\partial x^k}{\partial x} = kx^{k-1}$ 。
- 如果有另一个函数  $g(x) : \mathbb{R} \mapsto \mathbb{R}$ ，则  $\frac{\partial f(x)g(x)}{\partial x} = g(x)\frac{\partial f(x)}{\partial x} + f(x)\frac{\partial g(x)}{\partial x}$ 。
- $\frac{\partial e^x}{\partial x} = e^x$ ,  $\frac{\partial a^x}{\partial x} = \log(a)e^x$  和  $\frac{\partial \log x}{\partial x} = \frac{1}{x}$ 。

## 9.2 标量关于向量的导数

### 定义：标量关于向量的导数

假设有向量  $x = [x_1, \dots, x_n]^\top$ ，函数  $f(x) : \mathbb{R}^n \mapsto \mathbb{R}$ ，则函数  $f(x)$  关于  $x$  的导数表示为

$$\nabla_x f(x) = \frac{\partial f(x)}{\partial x} = \left[ \frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n} \right]^\top \quad (10)$$

以  $x = [x_1, x_2]$  且  $f(x) = x_1^2 + x_2^2$  为例， $\nabla_x f(x) = [2x_1, 2x_2]^\top$ 。

## 9.3 标量关于矩阵的导数

标量关于矩阵的导数与标量关于向量的导数类似。

### 定义：标量关于矩阵的导数

假设有矩阵  $X = \begin{bmatrix} x_{11} & \dots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \dots & x_{mn} \end{bmatrix}$ ，函数  $f(X) : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$ ，则函数  $f(X)$  关于  $X$  的导数表示为

$$\nabla_X f(X) = \frac{\partial f(X)}{\partial X} = \begin{bmatrix} \frac{\partial f(X)}{\partial x_{11}} & \dots & \frac{\partial f(X)}{\partial x_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(X)}{\partial x_{m1}} & \dots & \frac{\partial f(X)}{\partial x_{mn}} \end{bmatrix} \quad (11)$$

以  $X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$  且  $f(X) = x_{11}^2 + x_{12}^2 + x_{21}^2 + x_{22}^2$  为例，  
则  $\nabla_X f(X) = \begin{bmatrix} 2x_{11} & 2x_{12} \\ 2x_{21} & 2x_{22} \end{bmatrix}$ 。

## 9.4 向量关于标量的导数

### 定义：向量关于标量的导数

假设有标量  $x$ ，函数  $f(x) : \mathbb{R} \mapsto \mathbb{R}^n$ ，则函数  $f(x)$  关于  $x$  的导数表示为

$$\nabla_x f(x) = \frac{\partial f(x)}{\partial x} = \left[ \frac{\partial f(x)_1}{\partial x}, \frac{\partial f(x)_2}{\partial x}, \dots, \frac{\partial f(x)_n}{\partial x} \right]^\top \quad (12)$$

以  $f(x) = [x^2, x^3]^\top$  为例， $\nabla_x f(x) = \left[ \frac{\partial x^2}{\partial x}, \frac{\partial x^3}{\partial x} \right]^\top = [2x, 3x^2]^\top$ 。

## 9.5 向量关于向量的导数

### 定义：向量关于向量的导数

假设有向量  $x = [x_1, x_2, \dots, x_n]^\top$ ，函数  $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^m$ ，则函数  $f(x)$  关于  $x$  的导数表示为

$$\nabla_x f(x) = \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)_1}{\partial x_1} & \dots & \frac{\partial f(x)_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(x)_m}{\partial x_1} & \dots & \frac{\partial f(x)_m}{\partial x_n} \end{bmatrix} \quad (13)$$

以  $x = [x_1, x_2, x_3]^\top$ ， $f(x) = [x_1^2, x_2^2, x_3^2]^\top$  为例，则

$$\nabla_x f(x) = \begin{bmatrix} 2x_1 & 0 & 0 \\ 0 & 2x_2 & 0 \\ 0 & 0 & 2x_3 \end{bmatrix} \quad (14)$$

## 9.6 矩阵关于标量的导数

### 定义：矩阵关于标量的导数

假设有标量  $x$ ，函数  $f(x) : \mathbb{R} \mapsto \mathbb{R}^{m \times n}$ ，则函数  $f(x)$  关于  $x$  的导数表示为

$$\nabla_x f(x) = \frac{\partial f(x)}{\partial x} = \begin{bmatrix} \frac{\partial f(x)_{11}}{\partial x} & \dots & \frac{\partial f(x)_{1n}}{\partial x} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(x)_{m1}}{\partial x} & \dots & \frac{\partial f(x)_{mn}}{\partial x} \end{bmatrix} \quad (15)$$

以  $f(x) = \begin{bmatrix} x & x^2 \\ x^3 & x^4 \end{bmatrix}$  为例，则  $\nabla_x f(x) = \begin{bmatrix} 1 & 2x \\ 3x^2 & 4x^3 \end{bmatrix}$ 。

## 9.7 编程实现

在编程实现向量/矩阵的求导时，需要先自己手动计算出所需的导数表达形式，然后根据具体的变量值来计算导数值。这里以第9.6节提到的矩阵关于标量的导数为例，用 Python 实现为：

```
1 import numpy as np
2
3 # (1,1) 位置对应的导数函数: 1
4 def f_11(x):
5     return 1
6
7 # (1,2) 位置对应的导数函数: 2x
8 def f_12(x):
9     return 2*x
10
11 # (2,1) 位置对应的导数函数: 3x^2
12 def f_21(x):
13     return 3*x**2
14
15 # (2,2) 位置对应的导数函数: 4x^3
16 def f_22(x):
17     return 4*x**3
```

```
18
19 # 构建导数函数矩阵
20 deri_funcs = [[f_11, f_12],[f_21,f_22]]
21 row_num = 2
22 col_num = 2
23
24 #计算x=2时对应的导数值
25 x = 2
26 deri_values = np.array([
27     [deri_funcs[i][j](x) for j in range(col_num)]
28     for i in range(row_num)
29 ])
30 print('x=2时, f(x)关于x的导数为: \n', deri_values)
31
32 # x=2时, f(x)关于x的导数为:
33 #     [[ 1  4]
34 #      [12 32]]
35
```

## 引用

- [1 ] Strang, G., Strang, G., Strang, G., & Strang, G. (1993). Introduction to linear algebra (Vol. 3). Wellesley, MA: Wellesley-Cambridge Press.
- [2 ] Oliphant, T. E. (2006). A guide to NumPy (Vol. 1). Trelgol Publishing USA.

## 版权声明

本课件仅限东南大学机器学习课程使用，严禁传播