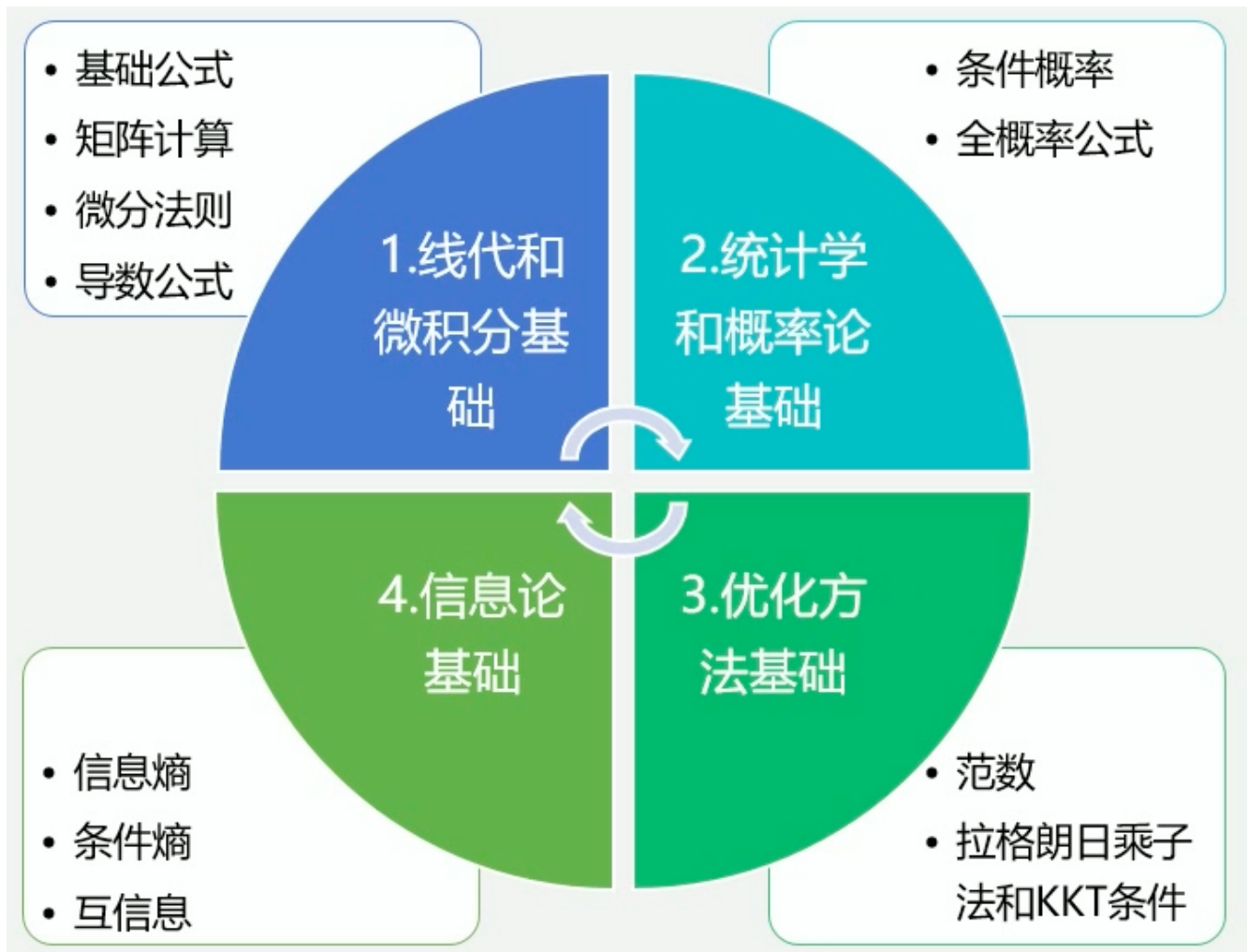


## 线性代数和微积分基础

### 机器学习基础



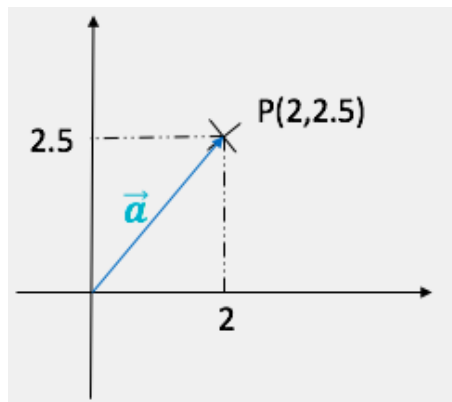
### 向量基础

标量定义：

标量是一个单独的数，一般用小写字母或希腊字母表示，如  $a, \alpha$  等

向量定义：

一个同时具有大小和方向的几何对象  $[a_1 \quad \dots \quad a_N]$



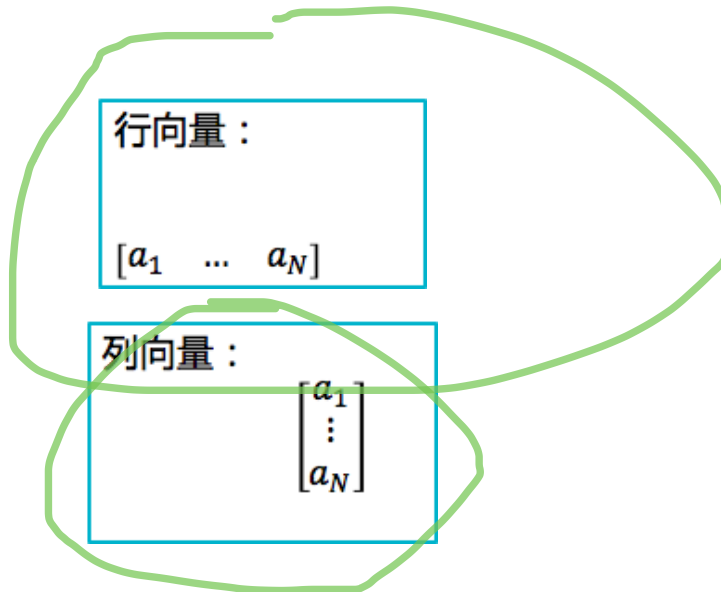
通俗来说，把数排成一行或一列就是向量，比如：

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, [1 \quad 3 \quad 4], [x_1 \quad \cdots \quad x_n]$$

向量的表示：

向量一般用粗体小写字母或粗体希腊字母表示，如  $\mathbf{x}$ （有时也会用箭头来标识，如  $\vec{\mathbf{x}}$ ），其元素记作  $x_i$

向量的分类：



向量的模：

$$|\vec{\mathbf{a}}| = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$$

向量的范数：

$$\|\vec{a}\|_1 = \sum |x_i|$$

$$\|\vec{a}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$$

$$\|\vec{a}\|_\infty = \max |x_i|$$

向量的加法:

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

$$[x_1 \quad \cdots \quad x_n] + [y_1 \quad \cdots \quad y_n] = [x_1 + y_1 \quad \cdots \quad x_n + y_n]$$

向量的数乘:

$$c \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c \cdot x_1 \\ \vdots \\ c \cdot x_n \end{bmatrix}$$

$$c \cdot [x_1 \quad \cdots \quad x_n] = [c \cdot x_1 \quad \cdots \quad c \cdot x_n]$$

向量的乘积—点积（代数定义）:

两个向量  $\vec{a} = [a_1, a_2, \cdots, a_n]$  和  $\vec{b} = [b_1, b_2, \cdots, b_n]$  的点积定义:

$$\vec{a} \cdot \vec{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

点积还可以写为： $\vec{a} \cdot \vec{b} = |\vec{a} \cdot \vec{b}^T|$

这里， $\vec{b}^T$  是行向量  $\vec{b}$  的转置，而  $|\vec{a} \cdot \vec{b}^T|$  是  $\vec{a} \cdot \vec{b}^T$  的行列式

向量的乘积—点积（几何定义）：

在欧几里得空间中，点积可以直观地定义为：

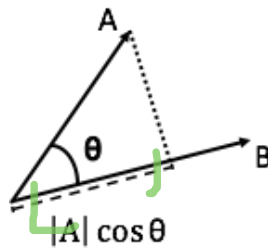
$$\vec{a} \cdot \vec{b} = |\vec{a}| |\vec{b}| \cos \theta$$

这里的  $|\vec{x}|$  表示  $\vec{x}$  的模（长度）， $\theta$  表示两个向量的角度。



欧式空间中向量A在向量B上的标量投影是指：

$$A_B = |A| \cos \theta$$



向量的乘积—点积（高维空间定义方式）

$$\langle \vec{a}, \vec{b} \rangle = \sum_{i=1}^n a_i b_i$$

机器学习基础公式

$$y = f(x) = \begin{matrix} \uparrow \\ \boxed{x} \end{matrix} \begin{matrix} \boxed{w} \\ \downarrow \end{matrix}^T + b$$

$[x_1, x_2, \dots, x_N]$   
 $[w_1, w_2, \dots, w_N]$

## 矩阵基础

矩阵定义：

由MxN个数排列成M行，N列的表

$$\begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix}$$

矩阵加法：

$$A + B = C \Rightarrow a_{ij} + b_{ij} = c_{ij}$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{M1} & \cdots & b_{MN} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & \cdots & a_{1N} + b_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} + b_{M1} & \cdots & a_{MN} + b_{MN} \end{bmatrix}$$

矩阵加法等于对应元素相加

矩阵加法满足：

结合律：(A+B)+C = A+(B+C)

交换律：A+B = B+A

python代码展示：

```
1 import numpy as np
2
3 # Matrix Add
4 x = numpy.mat([[1, 2], [3, 4]])
5 y = numpy.mat([[10, 20], [30, 40]])
6 print("Matrix Add:")
7 print(x + y)
```

in :

```

9      ''' Output:
10     Matrix Add:
11     [[11 22]
12     [[33 44]]
13     '''

```

out :

矩阵乘法—点积

假定两矩阵A, B:

A是由MxN个数排列成M行, N列的表:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix}$$

B是由NxK个数排列成N行, K列的表:

$$B = \begin{bmatrix} b_{11} & \cdots & b_{1K} \\ \vdots & \ddots & \vdots \\ b_{N1} & \cdots & b_{NK} \end{bmatrix}$$

AxB的表达式为:

$$A \cdot B = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \cdots & b_{1K} \\ \vdots & \ddots & \vdots \\ b_{N1} & \cdots & b_{NK} \end{bmatrix}$$

$$= \begin{bmatrix} d_{11} & \cdots & d_{1K} \\ \vdots & \ddots & \vdots \\ d_{M1} & \cdots & d_{MK} \end{bmatrix} \in \mathbb{R}^{M \times K}$$

$$= d_{ij}$$

$d_{ij}$  的表示式:

$$d_{ij} = \sum_{k=1}^N a_{ik} \cdot b_{kj}$$

矩阵相乘约束：乘数A的列数和乘数B的行数相等

注意：矩阵乘法不满足交换律（有时是不能乘，有时是乘积不相等）

$$A \cdot B \neq B \cdot A$$

python代码展示：

```

1  # Matrix Multiplication
2  import numpy as np
3
4  print("Matrix Multiplication")
5  a = np.mat([10, 15])
6  b = np.mat([[3], [2]])
7  print("a*b:\n")
8  print(a * b)
9  print("\n")
10 print("b*a:\n")
11 print(b * a)
12
13 a = np.mat([[1, 2], [3, 4]])
14 b = np.mat([[10, 20], [30, 40]])
15 print(a * b)

```

in :

```

17  ''' Output:
18  a*b:
19  [[60]]
20
21  b*a:
22  [[30 45]
23   [20 30]]
24
25  [[70 100]
26   [150 220]]
27  '''

```

out :

矩阵乘法—元素积：

$A \cdot B$ 的表达式为:

$$A \cdot B = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \cdots & b_{1N} \\ \vdots & \ddots & \vdots \\ b_{M1} & \cdots & b_{MN} \end{bmatrix}$$

$$= \begin{bmatrix} e_{11} & \cdots & e_{1N} \\ \vdots & \ddots & \vdots \\ e_{M1} & \cdots & e_{MN} \end{bmatrix} \in R^{M \times N}$$

$$= e_{ij}$$

$e_{ij}$  的表示式:

$$e_{ij} = a_{ij} \cdot b_{ij}$$

即对应位置相乘

矩阵元素积约束: 乘数 $A$ 的(行数, 列数)和乘数 $B$ 的(行数, 列数)相等

矩阵元素积满足交换律

python代码展示:

```
1 # Matrix Multiplication multiply
2 import numpy as np
3
4 print("Matrix Multiplication multiply")
5 a = np.mat([[1, 2], [3, 4]])
6 b = np.mat([[10, 20], [30, 40]])
7 print("a.*b:\n")
8 print(multiply(a,b))
9 print("\n")
10
11 print("b.*a:\n")
12 print(multiply(b,a))
```

in :

```
13 ''' Output:
14 a.*b:
15 [[10 40]
16  [90 160]]
17
18 b.*a
19 [[10 40]
20  [90 160]]
```

out :

矩阵的转置:

将矩阵中的数按照对角线交换

矩阵转置的数学形式:  $a_{ij} \Rightarrow a_{ji}$



$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

python代码展示:

```

1 # Matrix Transpose
2 m = numpy.mat([[1, 2], [3, 4]])
3 print("Matrix:\n")
4 print(m)
5 m_T = m.T
6 print("Matrix.Transpose:\n")
7 print(m_T)

```

in :

```

8 ''' Output:
9 Matrix:
10 m=
11 [[1 2]
12  [3 4]]
13 Matrix.Transpose:
14 m_T =
15 [[1 3]
16  [2 4]]
17 '''

```

out :

数学中的符号与运算

求最大化参数:

$$\operatorname{argmax}_c P(c)$$

意义: 可以用于返回一个可能性最大的分类, 返回当 $P(c)$ 值最大时对应的 $C$ 的值。

例如: 求最大化参数

当 $P(c = 1) = 0.2$ ,  $P(c = 2) = 0.8$ 时

$$\operatorname{argmax}_c P(c) = 2$$

求最值:

$$\max y$$

约束:

$$y = 2x + 1$$

$$s.t \ x \leq 0$$

结果:

$$\text{则} \max y(x) = 1$$

意义: 在所有  $x \in X$  的计算中, 返回最大值y

### 范数和微分

范数定义:

范数是具有“长度”概念的函数, 在线性代数, 泛函分析及相关的数学领域中是一个函数, 其为向量空间内的所有向量赋予非零的正长度或大小。

例:

$$L = [1, 2, 5]$$

- **L1范数**:  $\|L\|_1 = \sum_{i=1}^3 |L| = 8$

- **L2范数**:  $\|L\|_1 = \sqrt{\sum_{i=1}^3 |L|^2} = \sqrt{30}$

- **L无穷范数**:  $\|L\|_\infty = \max |L| = 5$

最大单一方向距离

微分定义:

在数学中，微分是对函数的局部变化率的一种线性描述

单变量微积分：

$$f' = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

导数：

$$df = f' dx$$

微分定义为：

常见函数与其导数：

$f(x)$	$f'(x)$		$f(x)$	$f'(x)$
$a$	$0$		$\ln x$	$\frac{1}{x}$
$ax$	$a$		$\sin x$	$\cos x$
$a^x$	$a^x \ln a$		$\cos x$	$-\sin x$
$e^x$	$e^x$		$\tan x$	$1/\cos^2 x$
$x^a$	$ax^{a-1}$		$\cot x$	$1/\sin^2 x$

微分的基本法则：

和法则：

$$(f + g)'(x) = f'(x) + g'(x)$$

积法则：

$$(fg)'(x) = f'(x)g(x) + f(x)g'(x)$$

复合函数的法则：

$$(g \circ f)'(x) = g'(f(x)) \cdot f'(x)$$

$$\text{令 } u = f(x)$$

$$\text{则 : } F(x) = g(u), u = f(x)$$

$$F'(x) = dF/dx = dF/du \cdot du/dx = g'(u) \cdot u'(x)$$