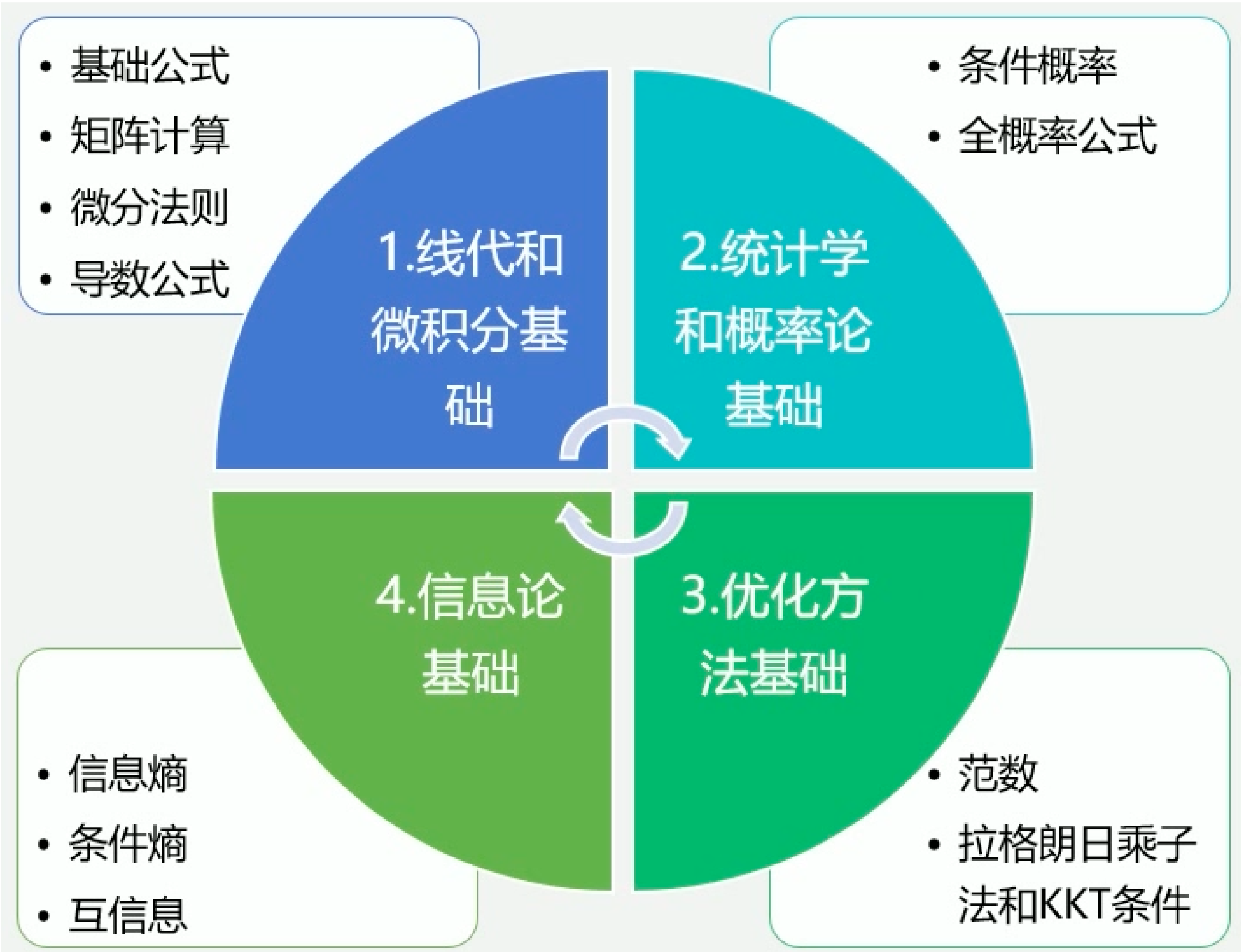
线性代数和微积分基础

机器学习基础



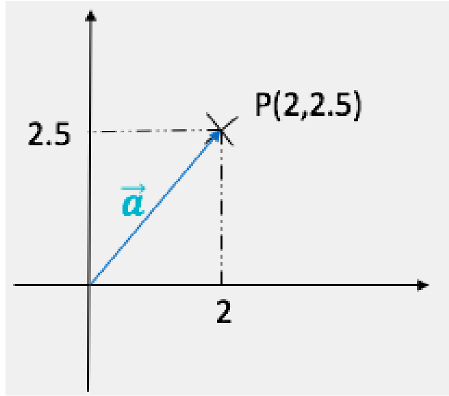
向量基础

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, α 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：

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

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$\boldsymbol{X}$（有时也会用箭头来标识，如$\vec{\boldsymbol{X}}$），其元素记作x_i

向量的分类：

行向量：

$$\begin{bmatrix}a_1 & \dots & a_N\end{bmatrix}$$

列向量：

$$\begin{bmatrix}a_1\\ \vdots \\ a_N\end{bmatrix}$$

向量的模：

$$|\vec{\mathbf{a}}| = \sqrt{\mathbf{x}_1^2 + \mathbf{x}_2^2 + \cdots + \mathbf{x}_N^2}$$

向量的范数：

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

$$\|\vec{\mathbf{a}}\|_2 = \sqrt{\mathbf{x}_1^2 + \mathbf{x}_2^2 + \cdots + \mathbf{x}_N^2}$$

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

向量的加法：

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

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

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: $\vec{a} \cdot \vec{b} = \left|\vec{a} \cdot \vec{b}^{\mathrm{T}}\right|$

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

向量的乘积－点积（几何定义）:

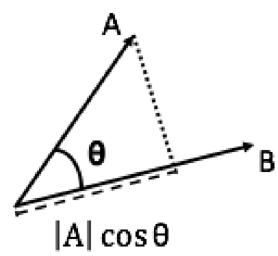
在欧几里得空间中，点积可以直观地定义为:

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

这里的$|\vec{x}|$表示$\vec{x}$的模（长度）, θ 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:

$$A_{\mathrm{B}}=|A|\ \cos\theta$$



向量的乘积－点积（高维空间定义方式）

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

机器学习基础公式

$$y = f(x) = \overset{[x_1, x_2, \cdots, x_N]}{\boxed{x}}\ \underset{[w_1, w_2, \cdots, w_N]}{\boxed{w}}^T + b$$

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:

```python
import numpy as np

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

in :

```
''' Output:
Matrix Add:
[[11 22]
 [33 44]]
'''
```

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：

$$\mathrm{A} \cdot \mathrm{B} = \begin{bmatrix} \mathrm{a}_{11} & \cdots & \mathrm{a}_{1\mathrm{N}} \\ \vdots & \ddots & \vdots \\ \mathrm{a}_{\mathrm{M}1} & \cdots & \mathrm{a}_{\mathrm{MN}} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{b}_{11} & \cdots & \mathrm{b}_{1\mathrm{K}} \\ \vdots & \ddots & \vdots \\ \mathrm{b}_{\mathrm{N}1} & \cdots & \mathrm{b}_{\mathrm{NK}} \end{bmatrix}$$

$$= \begin{bmatrix} \mathrm{d}_{11} & \cdots & \mathrm{d}_{1\mathrm{K}} \\ \vdots & \ddots & \vdots \\ \mathrm{d}_{\mathrm{M}1} & \cdots & \mathrm{d}_{\mathrm{MK}} \end{bmatrix} \in \mathrm{R}^{\mathrm{M} \times \mathrm{K}}$$

$$= \mathrm{d}_{\mathrm{ij}}$$

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：

```python
# Matrix Multiplication
import numpy as np

print("Matrix Multiplication")
a = np.mat([10, 15])
b = np.mat([[3], [2]])
print("a*b:\n")
print(a * b)
print("\n")
print("b*a:\n")
print(b * a)

a = np.mat([[1, 2], [3, 4]])
b = np.mat([[10, 20], [30, 40]])
print(a * b)
```

in :

```
''' Output:
a*b:
[[60]]

b*a:
[[30 45]
 [20 30]]

[[70 100]
 [150 220]]
'''
```

out :

矩阵乘法－元素积：

$A•B$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：

```
# Matrix Multiplication multiply
import numpy as np

print("Matrix Multiplication multiply")
a = np.mat([[1, 2], [3, 4]])
b = np.mat([[10, 20], [30, 40]])
print("a.*b:\n")
print(multiply(a,b))
print("\n")

print("b.*a:\n")
print(multiply(b,a))
```

in :

```
''' Output:
a.*b:
[[10 40]
 [90 160]]

b.*a
[[10 40]
 [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：

in :

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

out :

```
''' Output:
Matrix:
m=
[[1 2]
 [3 4]]
Matrix.Transpose:
m_T =
[[1 3]
 [2 4]]
'''
```

数学中的符号与运算

求最大化参数：

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

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

$$当\ P(c=1)=0.2\ ,\ P(c=2)=0.8时$$

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

求最值：

$$\max y$$

约束：

$$y = 2x + 1$$

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

结果：

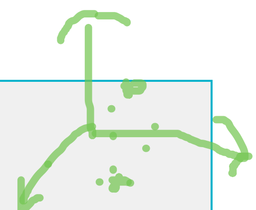
$$则 \max y(x) = 1$$

意义：在所有 $x \in \chi$ 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：

导数：

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

微分定义为：

$$\mathbf{df=f'\,dx}$$

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：

$(g \cdot f)^\wedge{}' (x)=g^\wedge{}' (f(x))\cdot f^\wedge{}' (x)$

令$u=f(x)$

则：$F(x)=g(u),u=f(x)$

$F^\wedge{}' (x)=dF/dx=dF/du\cdot du/dx=g' (u)\cdot u' (x)$