人工智能之机器学习

数学基础、Python科学计算库回顾

上海育创网络科技股份有限公司

主讲人：刘老师(GerryLiu)

课程要求

- 课上课下“九字”真言
  - 认真听，善摘录，勤思考
  - **多温故，乐实践**，再发散
- 四不原则
  - 不懒散惰性，不迟到早退
  - 不请假旷课，不拖延作业
- 一点注意事项
  - 违反“四不原则”，不包就业和推荐就业

课程内容

- 数学基础回顾
- Python科学计算库回顾

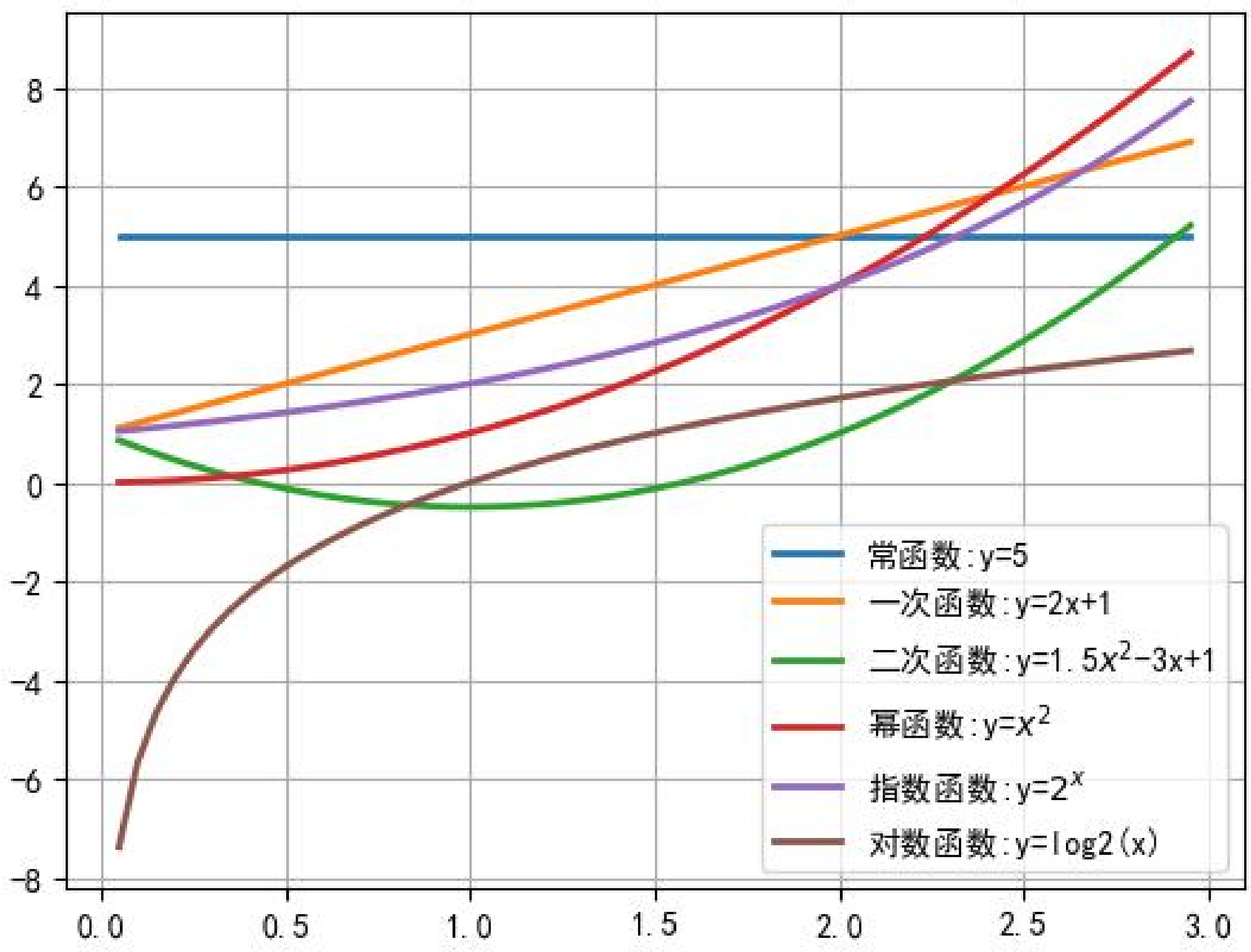
数学知识回顾

- 常见函数
- **导数、梯度《求导的方式、导数/梯度的含义/作用》**
- Taylor公式
- **联合概率、条件概率、全概率公式、贝叶斯公式**
- **期望、方差、协方差《了解这三个东西表示数据具有什么样的特性》**
- 大数定理、中心极限定理
- **最大似然估计(MLE)《最大似然估计必须掌握》**
- **向量、矩阵的运算**
- **向量、矩阵的求导**
- **SVD**、QR分解

常见函数

- 常函数: $y = C$ 一次函数: $y = ax + b$
- 二次函数: $y = ax^2 + bx + c$ 幂函数: $y = x^a$
- 指数函数: $y = a^x$, a的取值范围为: a>0&a≠1
- 对数函数: $y = \log_a(x)$, a的取值范围为: a>0&a≠1

常见函数




```
x = np.arange(0.05, 3, 0.05)
# 常函数
y1 = [5 for i in x]
plt.plot(x, y1, linewidth=2, label='常函数:y=5')
# 一次函数
y2 = [2 * i + 1 for i in x]
plt.plot(x, y2, linewidth=2, label='一次函数:y=2x+1')
# 二次函数
y3 = [1.5 * i * i - 3 * i + 1 for i in x]
plt.plot(x, y3, linewidth=2, label='二次函数:y=1.5$x^2$-3x+1')
# 幂函数
y4 = [math.pow(i, 2) for i in x]
plt.plot(x, y4, linewidth=2, label='幂函数:y=$x^2$')
# 指数函数
y5 = [math.pow(2, i) for i in x]
plt.plot(x, y5, linewidth=2, label='指数函数:y=$2^x$')
# 对数函数
y6 = [math.log(i, 1.5) for i in x]
plt.plot(x, y6, linewidth=2, label='对数函数:y=log2(x)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

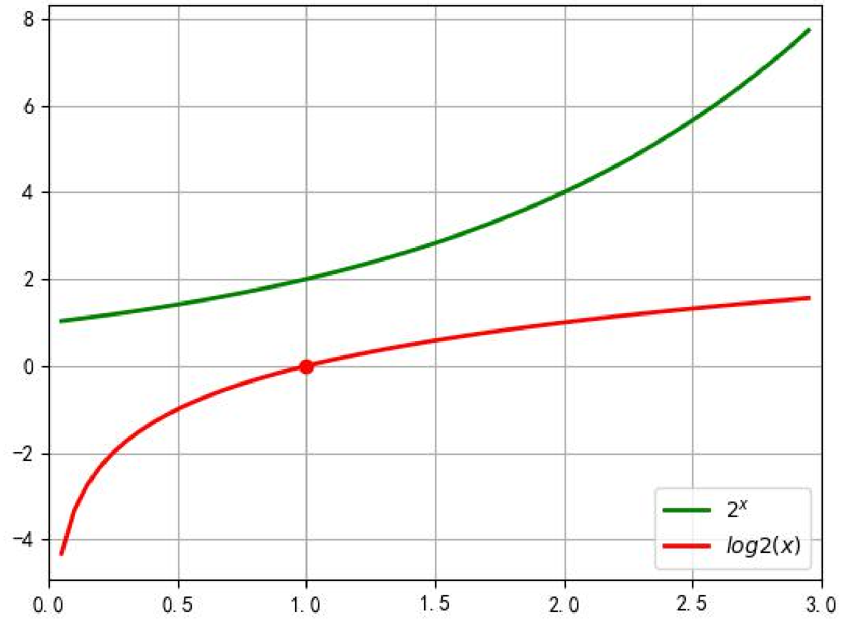
对数函数、指数函数

- 对数函数：$y=\log_a(x), a>0$且$a\neq 1$
- 指数函数：$y=a^x, a>0$且$a\neq 1$

$$\log_a x+\log_a y=\log_a(x*y)$$

$$\log_a x-\log_a y=\log_a \frac{x}{y}$$

$$\frac{a^x}{a^y}=a^{x-y} \qquad a^x\cdot a^y=a^{x+y}$$

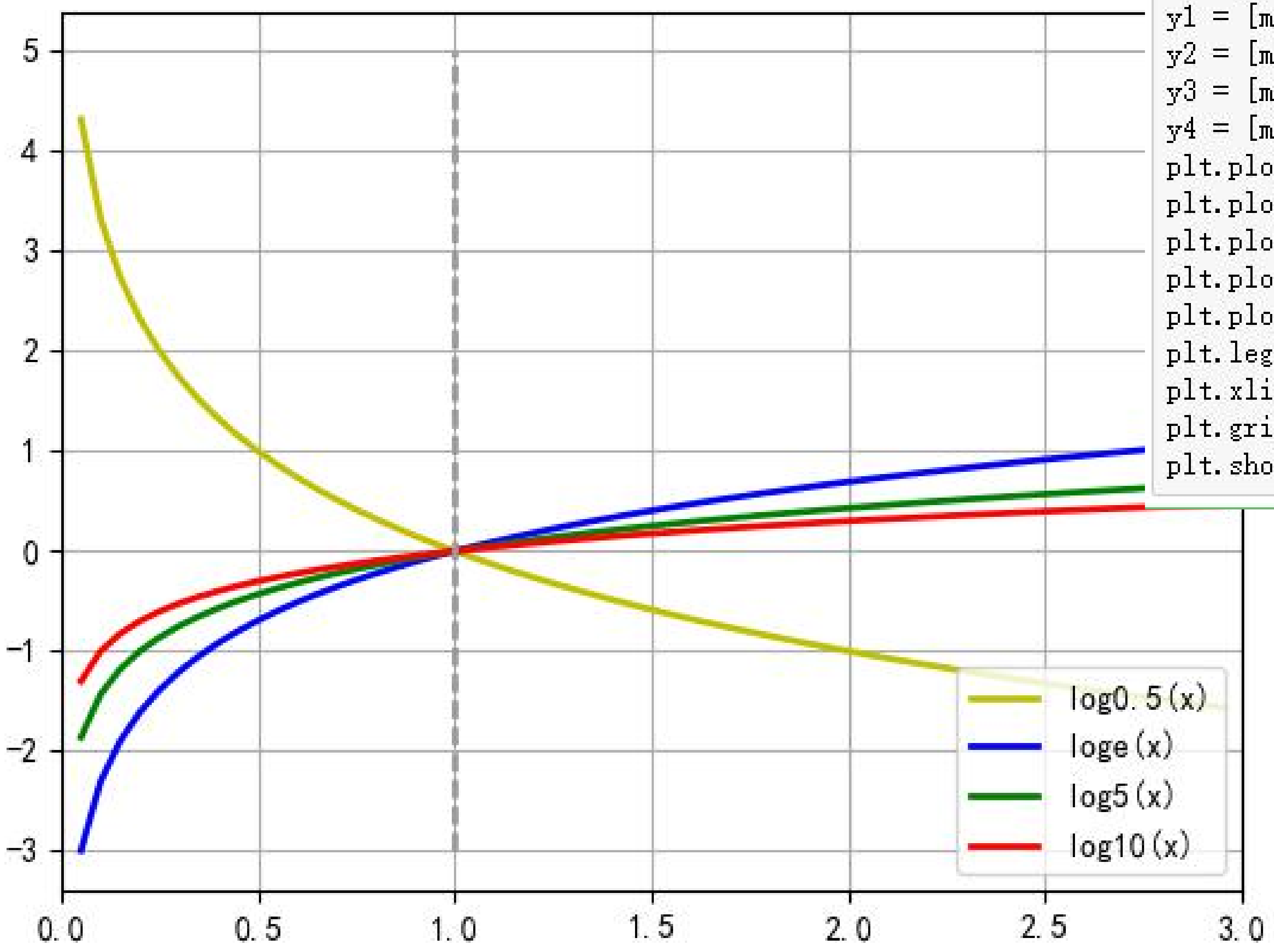


$$\log_a(1)=0$$

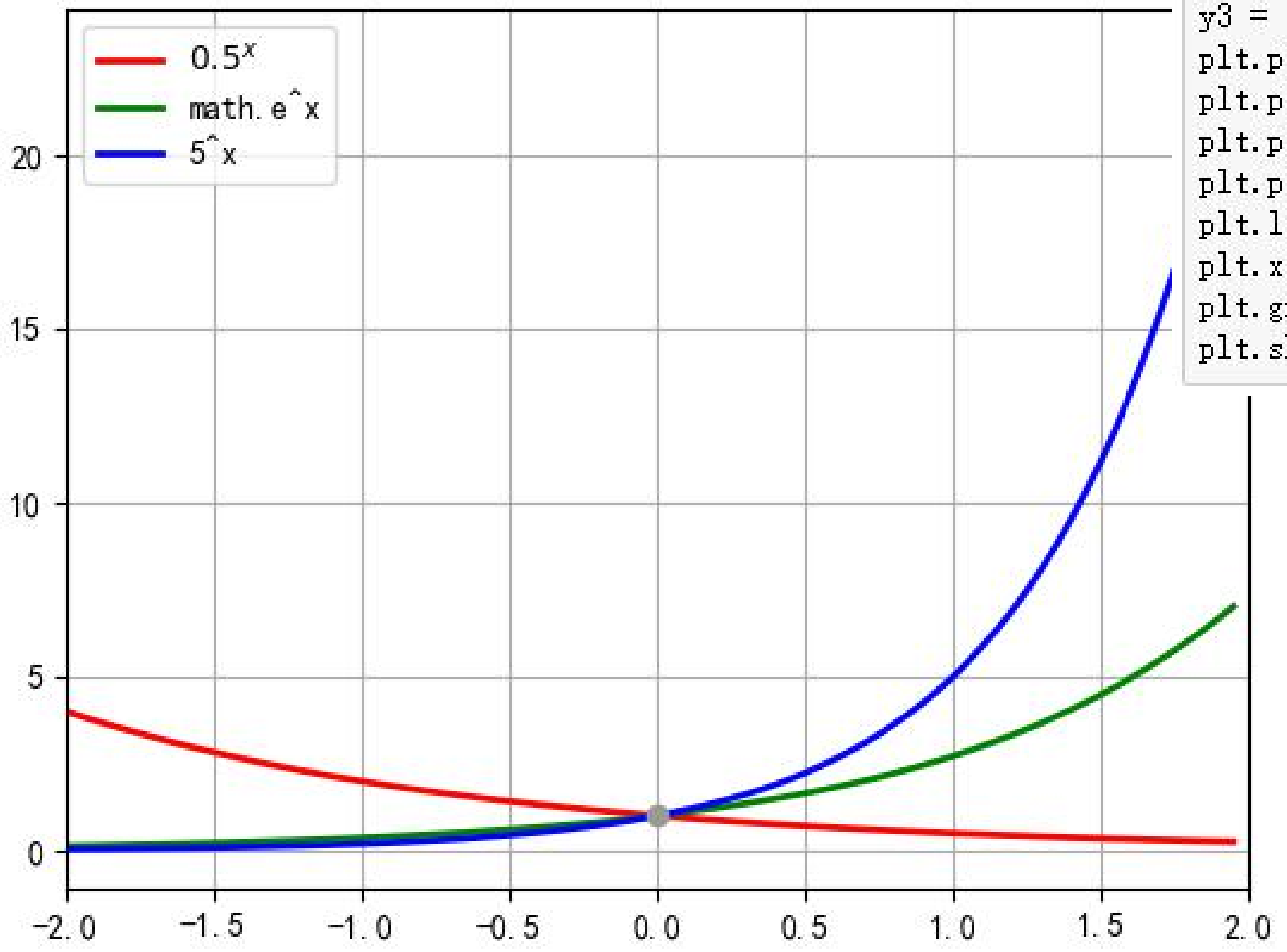
$$a^0=1$$

对数指数、指数函数

```python
x = np.arange(0.05, 3, 0.05)
y1 = [math.log(i, 0.5) for i in x]
y2 = [math.log(i, math.e) for i in x]
y3 = [math.log(i, 5) for i in x]
y4 = [math.log(i, 10) for i in x]
plt.plot(x, y1, linewidth=2, color='y', label='log0.5(x)')
plt.plot(x, y2, linewidth=2, color='b', label='loge(x)')
plt.plot(x, y3, linewidth=2, color='g', label='log5(x)')
plt.plot(x, y4, linewidth=2, color='r', label='log10(x)')
plt.plot([1,1], [-3, 5], '--', color='#999999', linewidth=2)
plt.legend(loc='lower right')
plt.xlim(0, 3)
plt.grid(True)
plt.show()
```



对数指数、指数函数

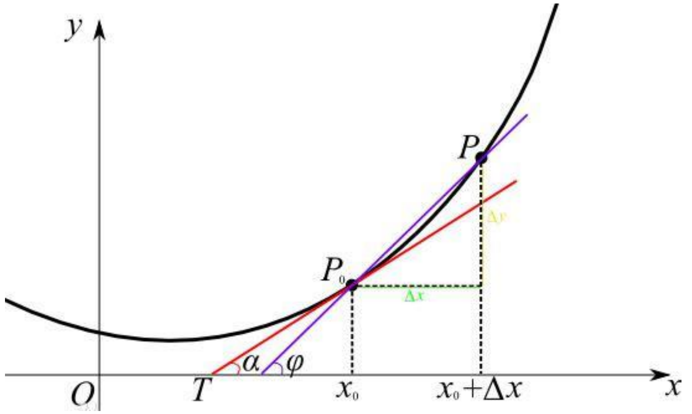



```python
x = np.arange(-2, 2, 0.05)
y1 = [math.pow(0.5, i) for i in x]
y2 = [math.pow(math.e, i) for i in x]
y3 = [math.pow(5, i) for i in x]
plt.plot(x, y1, linewidth=2, color='r', label='$0.5^x$')
plt.plot(x, y2, linewidth=2, color='g', label='math.e^x')
plt.plot(x, y3, linewidth=2, color='b', label='5^x')
plt.plot([0], [1], 'o', color='#999999', linewidth=2)
plt.legend(loc='upper left')
plt.xlim(-2, 2)
plt.grid(True)
plt.show()
```

导数

- 一个函数在某一点的导数描述了这个函数在这一点附近的**变化率**，也可以认为是函数在某一点的导数就是该函数所代表的曲线在这一点的**切线斜率**。导数值越大，表示函数在该点处的变化越大。
- 定义：当函数y=f(x)在自变量$x=x_0$上产生一个增量Δx时，函数输出值的增量Δy和自变量增量Δx之间的比值在Δx趋近与0的时候存在极限值a，那么a即为函数在x_0处的导数值。

$$\lim_{\Delta x\to 0}\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}=\lim_{\Delta x\to 0}\frac{\Delta y}{\Delta x}$$



导数

- **导数**就是曲线的斜率，是曲线变化快慢的一个反应。
- **二阶导数**是斜率变化的反应，表现曲线是**凹凸性**。

$$y = f(x)$$

$$y' = f'(x) = \frac{\partial f(x)}{\partial x} = \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}$$

常见导函数

$$C'=0, C\text{为常数} \qquad \left(x^n\right)'=n\cdot x^{n-1}$$

$$\left(a^x\right)'=a^x\cdot\ln a \qquad \left(e^x\right)'=e^x$$

$$(\log_a x)'=\frac{1}{x\ln a} \qquad (\ln x)'=\frac{1}{x}$$

$$(u\pm v)'=u'\pm v' \qquad (uv)'=u'v+uv' \qquad \left(\frac{u}{v}\right)'=\frac{u'v-uv'}{v^2}$$

$$y=f(g(x))\Rightarrow y'=\frac{df}{dg}\cdot\frac{dg}{dx}$$

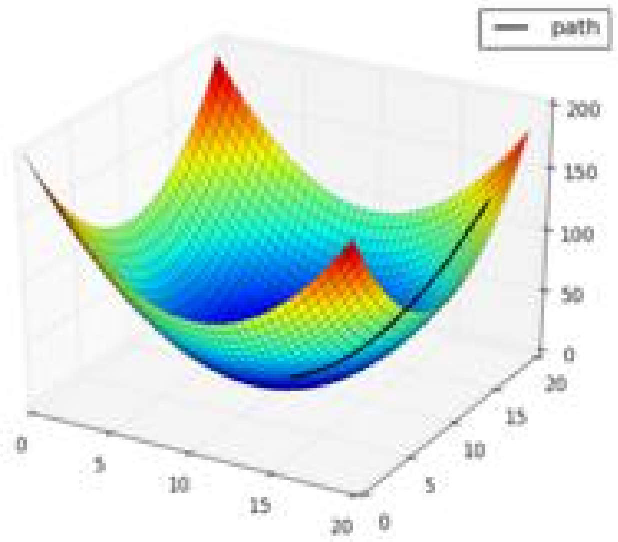
偏导数

- 在一个多变量的函数中，偏导数就是关于其中一个变量的导数而保持其它变量恒定不变。假定二元函数z=f(x,y)，点(x_0,y_0)是其定义域内的一个点，将y固定在y_0上，而x在x_0上增量Δx，相应的函数z有增量$\Delta z=f(x_0+\Delta x, y_0) - f(x_0,y_0)$；Δz和Δx的比值当Δx的值趋近于0的时候，如果极限存在，那么此极限值称为函数z=f(x,y)在处对x的偏导数(partial derivative)，记作：$f'x(x_0,y_0)$

$$\text{对x的偏导数}: \left.\frac{\partial f}{\partial x}\right|_{\substack{x=x_0\\ y=y_0}} \qquad \text{对y的偏导数}: \left.\frac{\partial f}{\partial y}\right|_{\substack{x=x_0\\ y=y_0}}$$

梯度

- 梯度：梯度是一个向量，表示某一函数在该点处的**方向导数**沿着该方向取的最大值，即函数在该点处沿着该方向变化最快，变化率最大(即该梯度向量的模)；当函数为一维函数的时候，梯度其实就是导数。



$$\nabla f(x_1, x_2) = \left(\frac{\partial f(x_1, x_2)}{\partial x_1}, \frac{\partial f(x_1, x_2)}{\partial x_2} \right)$$

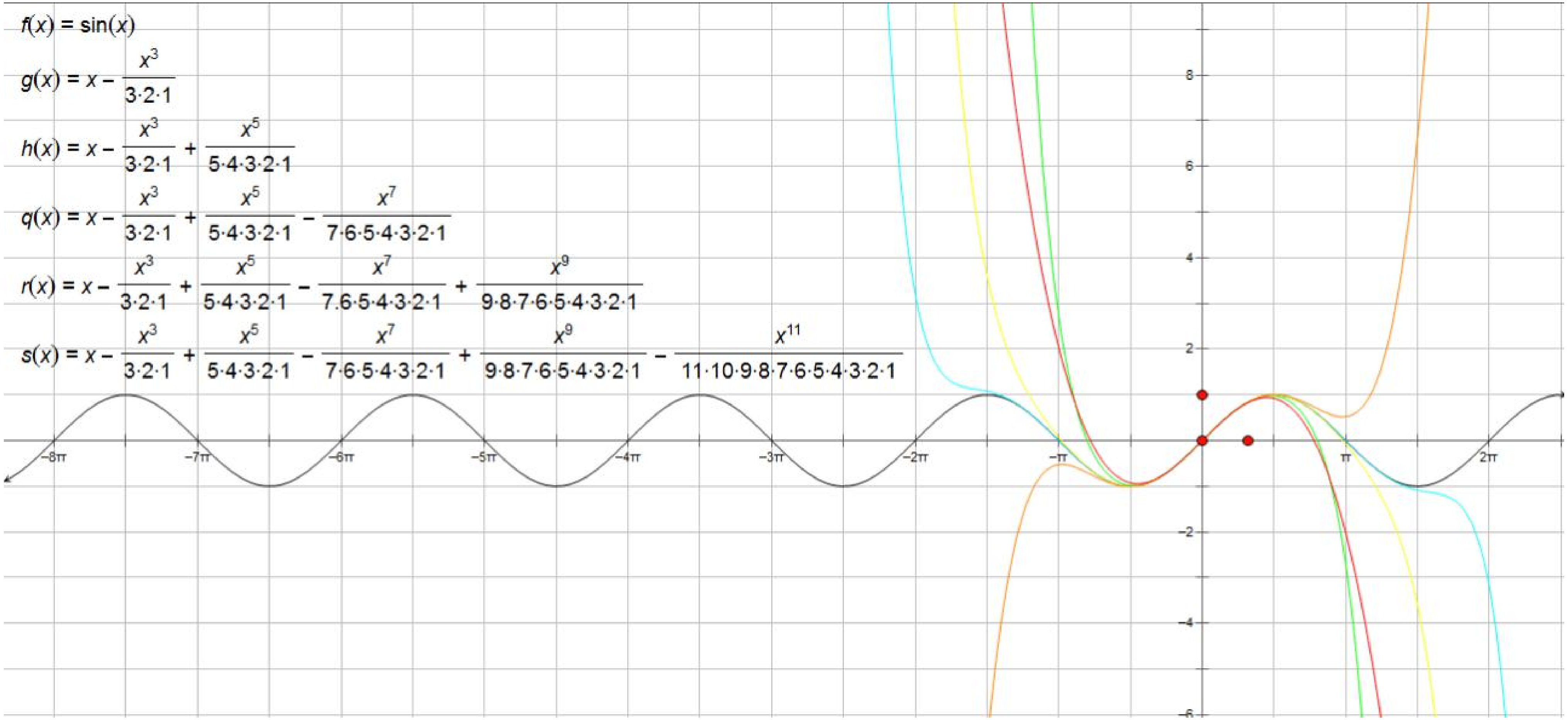
Taylor公式

- Taylor(泰勒)公式是用一个函数在某点的信息描述其附近取值的公式。如果函数足够平滑，在已知函数在某一点的各阶导数值的情况下，Taylor公式可以利用这些导数值来做系数构建一个多项式近似函数在这一点的邻域中的值。
- 若函数f(x)在包含x_0的某个闭区间[a,b]上具有n阶函数，且在开区间(a,b)上具有n+1阶函数，则对闭区间[a,b]上任意一点x，有Taylor公式如下：<$f^{(n)}(x)$表示f(x)的n阶导数，$R_n(x)$是Taylor公式的余项，是$(x-x_0)^n$的高阶无穷小>
- 备注：Taylor公式是一种多项式近似拟合的方式。用一个多项式的值去逼近某个函数。

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x)$$

Taylor公式应用

$$\sin(x)=x-\frac{x^3}{3!}+\frac{x^5}{5!}+\ldots+(-1)^{m-1}\frac{x^{2m-1}}{(2m-1)!}+R_{2m-1}(x)$$



Taylor公式应用2

$$y=e^x \Rightarrow y'=y=e^x$$

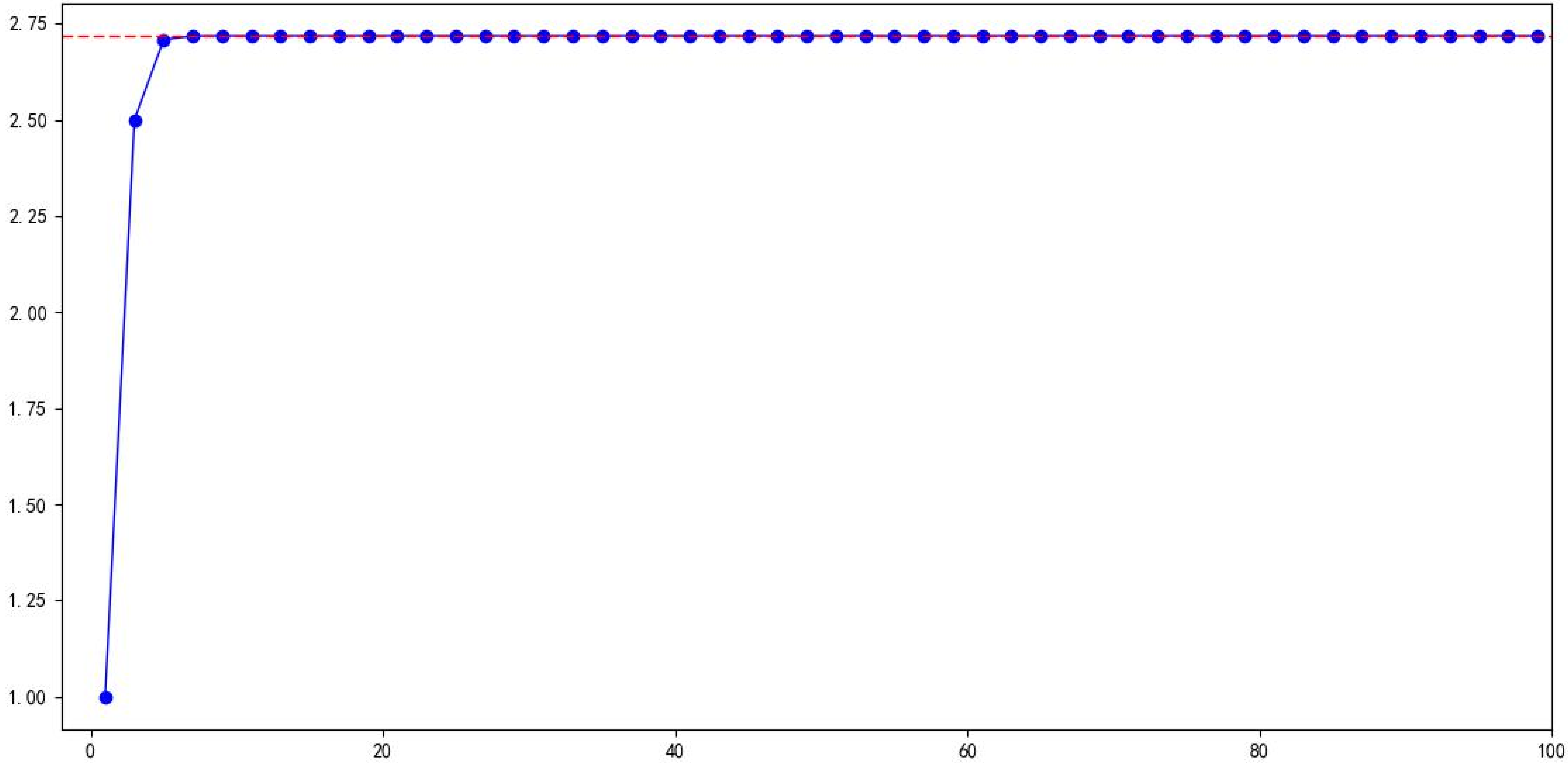
- 计算近似值 $e=\lim\limits_{x\to\infty}\left(1+\dfrac{1}{n}\right)^n$ ，并估计误差值.

$$e^x \approx \sum_{k=0}^{n}\frac{e^{x_0}}{k!}(x-x_0)^k \xRightarrow{\text{令}x_0=0} e^x \approx 1+x+\frac{x^2}{2!}+\ldots+\frac{x^n}{n!}$$

$$\xRightarrow{\text{令}x=1} e \approx 1+1+\frac{1}{2!}+\frac{1}{3!}+\ldots+\frac{1}{n!} \xRightarrow{\text{令}n=10} e \approx 2.7182815$$

$$\delta=|R_{10}|=\frac{1}{11!}+\frac{1}{12!}+\ldots=\frac{1}{11!}\left(1+\frac{1}{12}+\frac{1}{12*13}+\ldots\right)<\frac{1}{11!}\left(1+\frac{1}{12}+\frac{1}{12^2}+\ldots\right)=\frac{12}{11*11!}=2.73*10^{-8}$$

Taylor公式应用



古典概率

- 概率是以假设为基础的，即假定随机现象所发生的事件是有限的、互不相容的，而且每个基本事件发生的可能性相等。一般来讲，如果在全部可能出现的基本事件范围内构成事件A的基本事件有a个，不构成事件A的有b个，那么事件A出现的概率为：

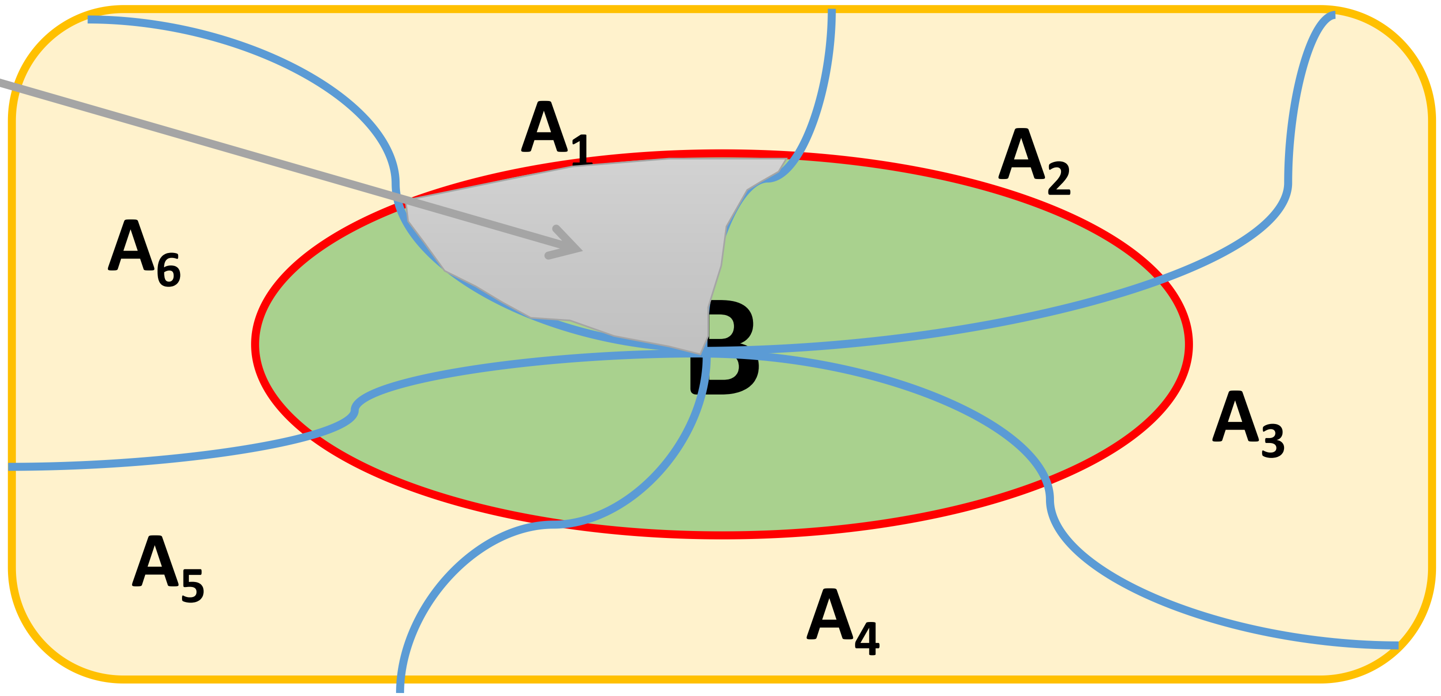
$$P(A)=\frac{a}{a+b}$$

- 概率体现的是随机事件A发生可能的**大小度量**(数值)

联合概率

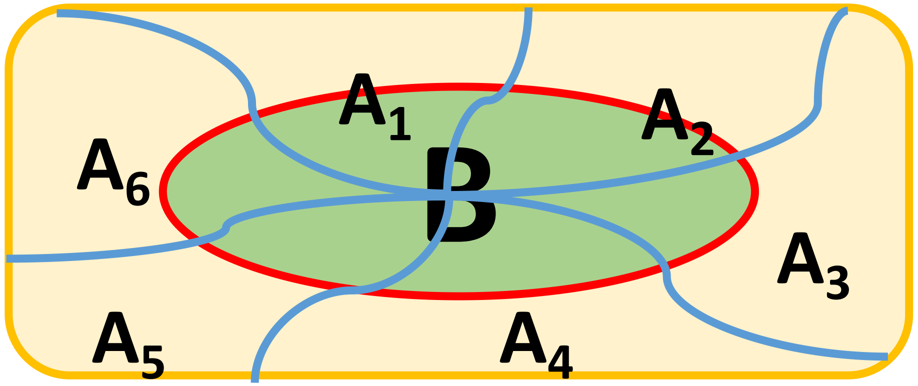
- 表示两个事件共同发生的概率，事件A和事件B的共同概率记作：P(AB)、P(A,B)或者P(A∩B)，读作“事件A和事件B同时发生的概率”

$P(A_1,B)$



条件概率

- 事件A在另外一个事件B已经发生的条件下的发生概率叫做条件概率，表示为P(A|B)，读作“在B条件下A发生的概率“，一般情况下P(A|B)≠P(A)，而且条件概率具有三个特性：
  - 非负性
  - 可列性
  - 可加性



$$P(A|B)=\frac{P(A,B)}{P(B)}$$

条件概率

- 将条件概率公式由两个事件推广到任意有穷多个事件时，可以得到如下公式，假设A_1，A_2，....，A_n为n个任意事件(n≥2)，而且$P(A_1A_2 \ldots A_n)>0$，则：

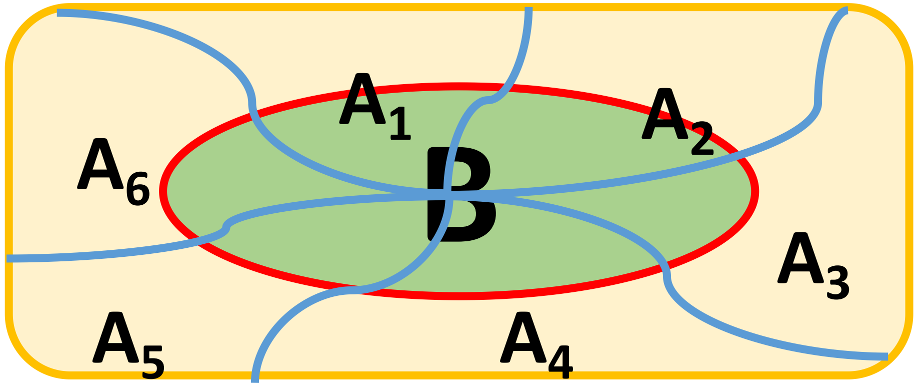
$$P\left(A_1A_2\ldots A_n\right)=P\left(A_1\right)P\left(A_2 \mid A_1\right)\ldots P\left(A_n \mid A_1A_2\ldots A_{n-1}\right)$$

全概率公式

- 样本空间Ω有一组事件A_1、A_2...A_n, 如果事件组满足下列两个条件，那么事件组称为样本空间的一个划分：

$$\forall i \neq j \in \{1,2\ldots,n\}, A_i A_j = \phi$$

$$A_1 \cup A_2 \ldots \cup A_n = \Omega$$



- 设事件{A_j}是样本空间Ω的一个划分，且$P(A_i)>0$，那么对于任意事件B，全概率公式为:

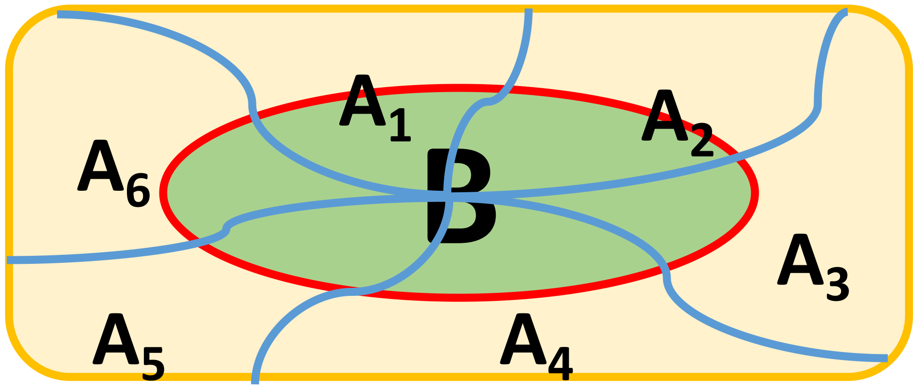
$$P(B)=\sum_{i=1}^{n} P(A_i)P(B \mid A_i)$$

贝叶斯公式

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

- 设A_1、A_2...A_n是样本空间Ω的一个划分，如果对任意事件B而言，有P(B)>0，那么：

$$P(A_i|B)=\frac{P(B,A_i)}{P(B)}=\frac{P(A_i)\bullet P(B|A_i)}{\sum_{j=1}^{n}P(A_j)\bullet P(B|A_j)}$$

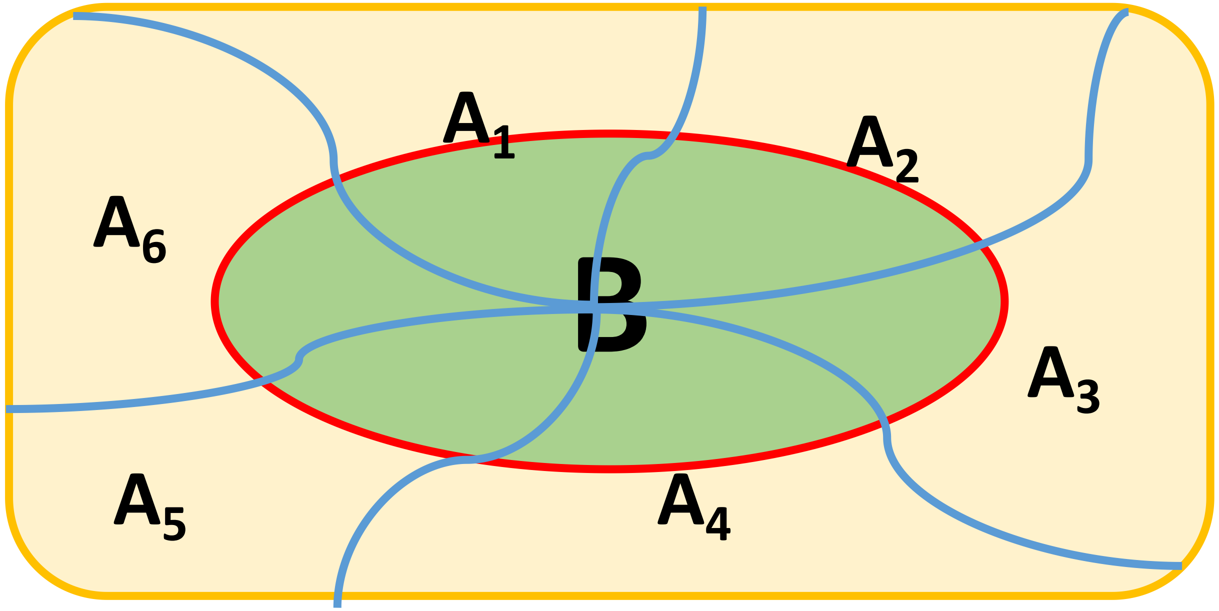


概率公式

$$P(A \mid B) = \frac{P(A, B)}{P(B)}$$

$$P(B) = \sum_{i=1}^{n} P(A_i) P(B \mid A_i)$$

$$P(A_i \mid B) = \frac{P(B \mid A_i) P(A_i)}{\sum_{j=1}^{n} P(A_j) P(B \mid A_j)}$$



期望

- 期望(mean)：也就是均值，是概率加权下的“平均值”，是每次可能结果的概率乘以其结果的总和，反映的实随机变量平均取值大小。常用符号μ表示：
- 连续性数据：

$$E(X)=\int_{-\infty}^{\infty} xf(x)dx$$

- 离散型数据：

$$E(X)=\sum_{i} x_i p_i$$

期望

X	2	4	6	8	10
P(x)	0.2	0.2	0.2	0.2	0.2

$$E(X)=\sum_i x_i p_i$$

$$=2*0.2+4*0.2+6*0.2+8*0.2+10*0.2$$

$$=6$$

期望

假设C为一个常数，X和Y实两个随机变量，那么期望有一下性质：

$$E(C)=C \qquad E(CX)=CE(X)$$

$$E(X+Y)=E(X)+E(Y)$$

如果X和Y相互独立，那么$E(XY)=E(X)E(Y)$

如果$E(XY)=E(X)E(Y)$,那么X和Y不相关

方差

- 方差(variance)是衡量随机变量或一组数据时离散程度的度量，是用来度量随机变量和其数学期望之间的偏离程度。即方差是衡量数据原数据和期望/均值相差的度量值。

$$Var(X) = D(X) = \sigma^2 = \frac{\sum (X-\mu)^2}{N}$$

$$D(X) = \sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2 \qquad D(X) = \int_a^b (x-\mu)^2 f(x) dx$$

$$D(X) = E\left((X - E(X))^2\right) = E(X^2) - (E(X))^2$$

方差

X	2	4	6	8	10
P(x)	0.2	0.2	0.2	0.2	0.2

$$E(X)=6 \quad E\left(X^2\right)=44$$

$$D(X)=E\left(X^2\right)-(E(X))^2$$

$$=44-6^2$$

$$=8$$

方差

- 假设C为一个常数，X和Y实两个随机变量，那么方差有一下性质：

$$D(C)=0 \qquad D(CX)=C^2D(X) \qquad D(C+X)=D(X)$$

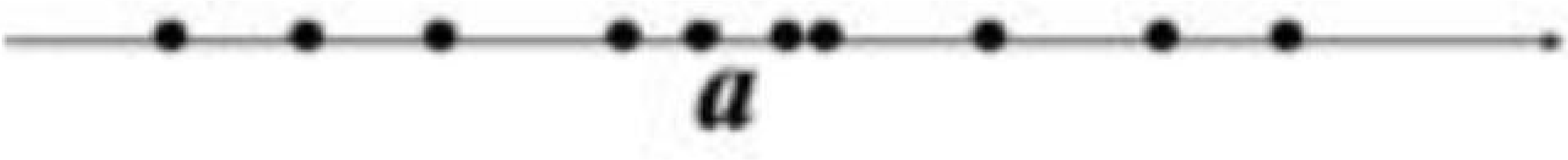
$$D(X\pm Y)=D(X)+D(Y)\pm 2Cov(X,Y)$$

协方差$Cov(X,Y)=E\{(X-E(X))\cdot(Y-E(Y))\}$

如果X和Y不相关，那么$D(X\pm Y)=D(X)+D(Y)$

方差

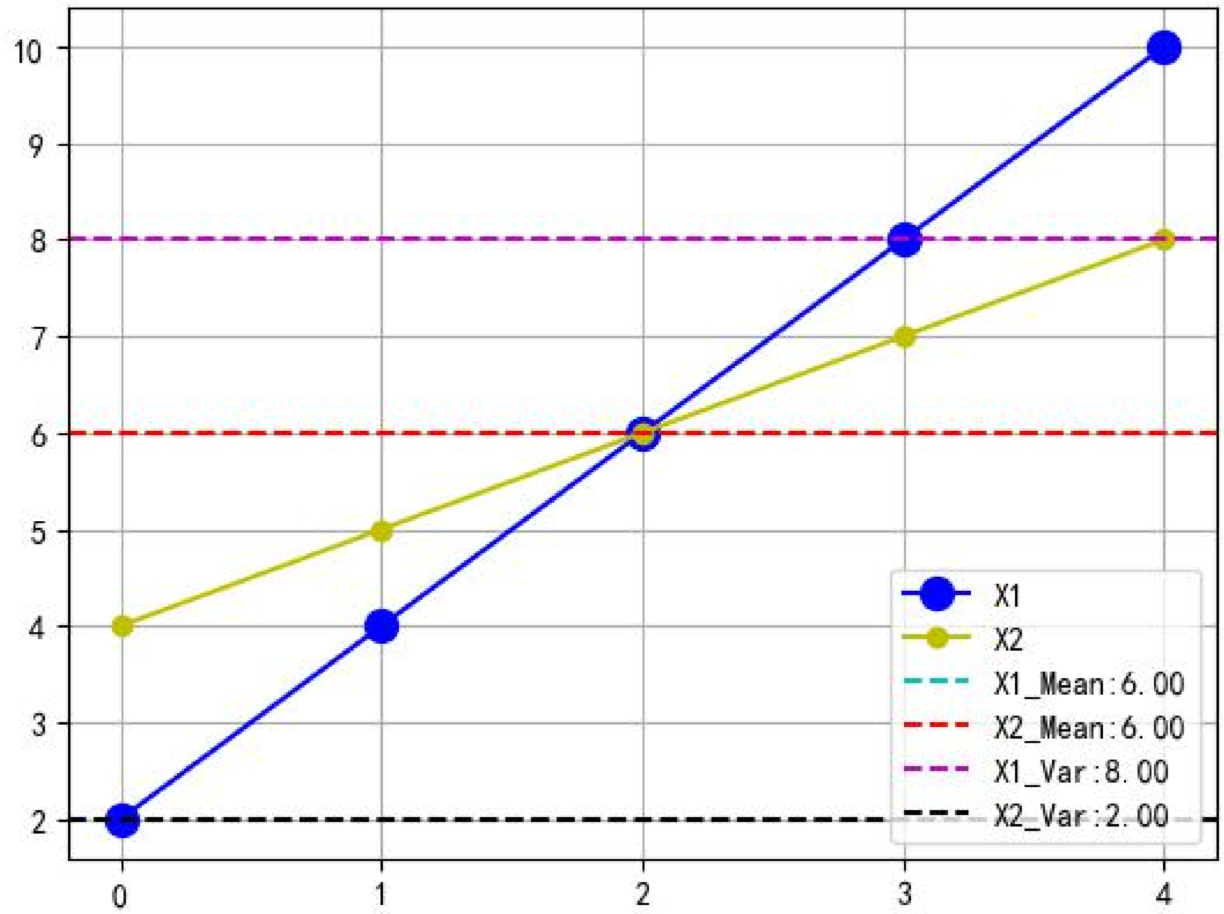
- 已知某零件的真实长度为a，现用甲、乙两台仪器各测量10次，将测量结果X用坐标轴上的点表示，并且甲仪器的测量如图中的点，乙仪器的测量结果全是a，此时两台仪器的测量均值都是a，但是我们可能会认为乙机器性能更好，因为乙机器的测量值在a附近。因此可见，研究随机变量与其均值的偏离程度是十分必要



方差

X1	2	4	6	8	10
P(x1)	0.2	0.2	0.2	0.2	0.2

X2	4	5	6	7	8
P(x2)	0.2	0.2	0.2	0.2	0.2



$$E(X_1)=6 \qquad D(X_1)=8$$

$$E(X_2)=6 \qquad D(X_2)=2$$

标准差

- 标准差(Standard Deviation)是离均值平方的算术平均数的平方根，用符号σ表示，其实标准差就是方差的算术平方根。
- 标准差和方差都是测量离散趋势的最重要、最常见的指标。标准差和方差的不同点在于，标准差和变量的计算单位是相同的，比方差清楚，因此在很多分析的时候使用的是标准差。

$$\sigma = \sqrt{D(X)} = \sqrt{\frac{\sum (X-\mu)^2}{N}}$$

标准差

X1	2	4	6	8	10
P(x1)	0.2	0.2	0.2	0.2	0.2

X2	4	5	6	7	8
P(x2)	0.2	0.2	0.2	0.2	0.2

$$D(X_1)=8 \qquad \sigma_1=\sqrt{D(X_1)}=\sqrt{8}=2.8284$$

$$D(X_2)=2 \qquad \sigma_2=\sqrt{D(X_2)}=\sqrt{2}=1.4142$$

协方差

- 协方差常用于衡量两个变量的总体误差；当两个变量相同的情况下，协方差其实就是方差。
- 如果X和Y是统计独立的，那么二者之间的协方差为零。但是如果协方差为零，那么X和Y是不相关的。

$$
\begin{aligned}
Cov(X,Y) &= E[(X-E(X))\cdot(Y-E(Y))] \\
&= E[XY - XE(Y) - YE(X) + E(X)E(Y)] \\
&= E(XY) - E(X)E(Y)
\end{aligned}
$$

协方差

- 假设C为一个常数，X和Y实两个随机变量，那么协方差有性质如下所示：

$$Cov(X,Y)=Cov(Y,X)$$

$$Cov(aX,bY)=abCov(X,Y)$$

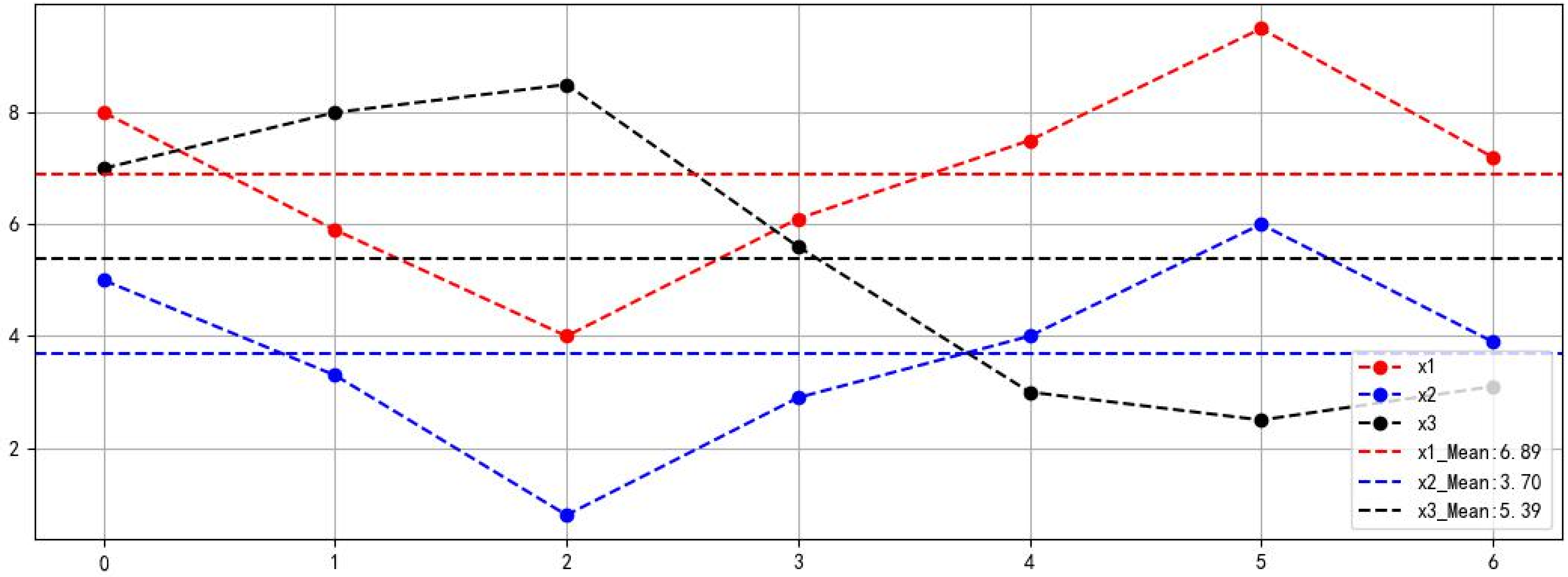
$$Cov(X_1+X_2,Y)=Cov(X_1,Y)+Cov(X_2,Y)$$

协方差

- 协方差是两个随机变量具有相同方向变化趋势的度量：
  - 若Cov(X,Y) > 0, 则X和Y的变化趋势相同；
  - 若Cov(X,Y) < 0, 则X和Y的变化趋势相反；
  - 若Cov(X,Y) = 0，则X和Y不相关，也就是变化没有什么相关性

协方差

X1	8	5.9	4	6.1	7.5	9.5	7.2
X2	5	3.3	0.8	2.9	4	6	3.9
X3	7	8.0	8.5	5.6	3.0	2.5	3.1



协方差矩阵

- 对于n个随机向量(X_1,X_2,X_3....X_n), 任意两个元素X_i和X_j都可以得到一个协方差，从而形成一个n*n的矩阵，该矩阵就叫做协方差矩阵，协方差矩阵为对称矩阵。

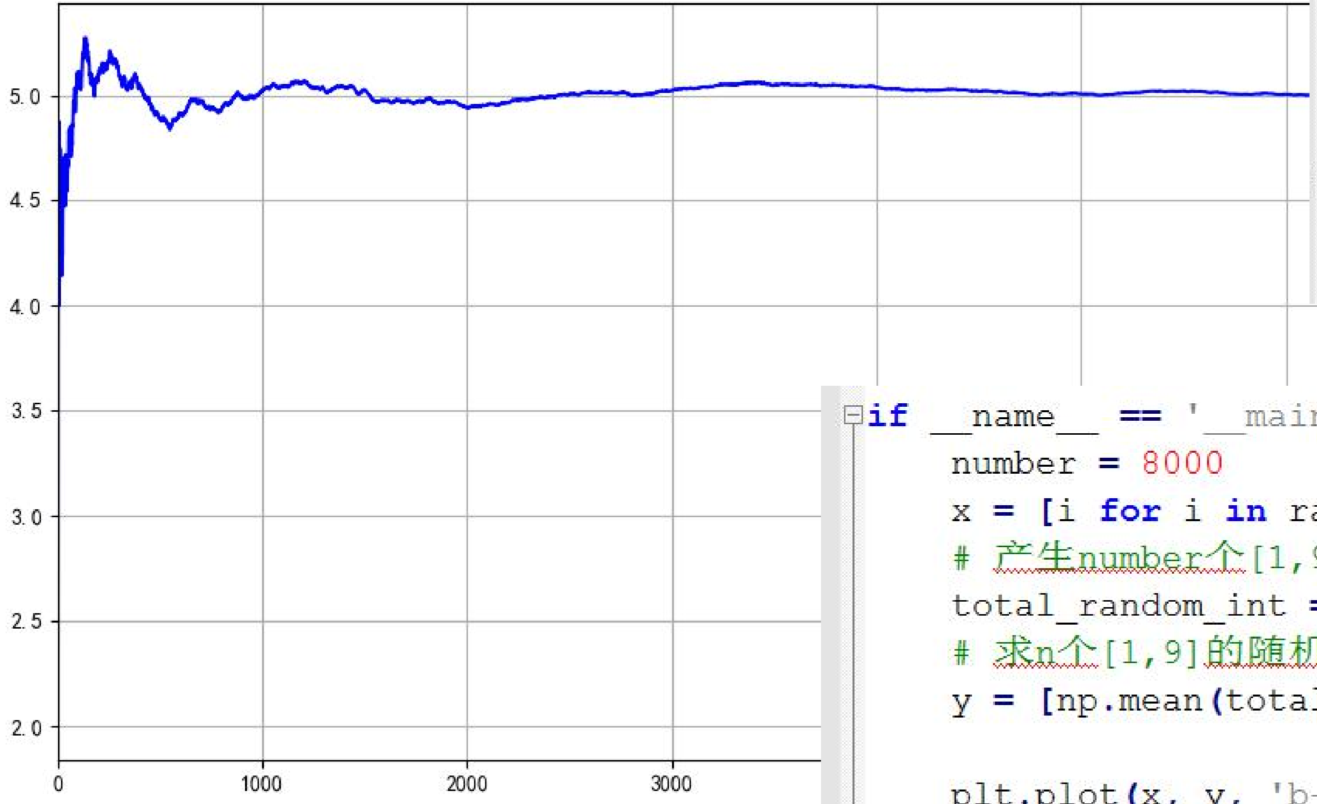
$$c_{ij} = E\{[X_i - E(X_i)][X_j - E(X_j)]\} = Cov(X_i, X_j)$$

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

大数定理

- 大数定律的意义：**随着样本容量n的增加，样本平均数将接近于总体平均数(期望μ)，所以在统计推断中，一般都会使用样本平均数估计总体平均数的值**。
- 也就是我们会使用一部分样本的平均值来代替整体样本的期望/均值，出现偏差的可能是存在的，但是当n足够大的时候，偏差的可能性是非常小的，当n无限大的时候，这种可能性的概率基本为0。
- 大数定律的主要作用就是为使用**频率**来估计**概率**提供了理论支持；为使用部分数据来近似的模拟构建全部数据的特征提供了理论支持。

大数定理




```python
# 解决中文显示问题
mpl.rcParams['font.sans-serif'] = [u'SimHei']
mpl.rcParams['axes.unicode_minus'] = False

# 给定随机数的种子
random.seed(28)

def generate_random_int(n):
    """产生n个1-9的随机数"""
    return [random.randint(1, 9) for i in range(n)]
```

```python
if __name__ == '__main__':
    number = 8000
    x = [i for i in range(number + 1) if i != 0]
    # 产生number个[1,9]的随机数
    total_random_int = generate_random_int(number)
    # 求n个[1,9]的随机数的均值, n=1,2,3,4,5.....
    y = [np.mean(total_random_int[0:i + 1]) for i in range(number)]

    plt.plot(x, y, 'b-')
    plt.xlim(0, number)
    plt.grid(True)
    plt.show()
```

中心极限定理

- 中心极限定理(**Central Limit Theorem**)；假设$\{X_n\}$为独立同分布的随机变量序列，并具有相同的期望μ和方差为σ^2，则$\{Y_n\}$服从中心极限定理，且Y_n为随机序列$\{X_n\}$的规范和：

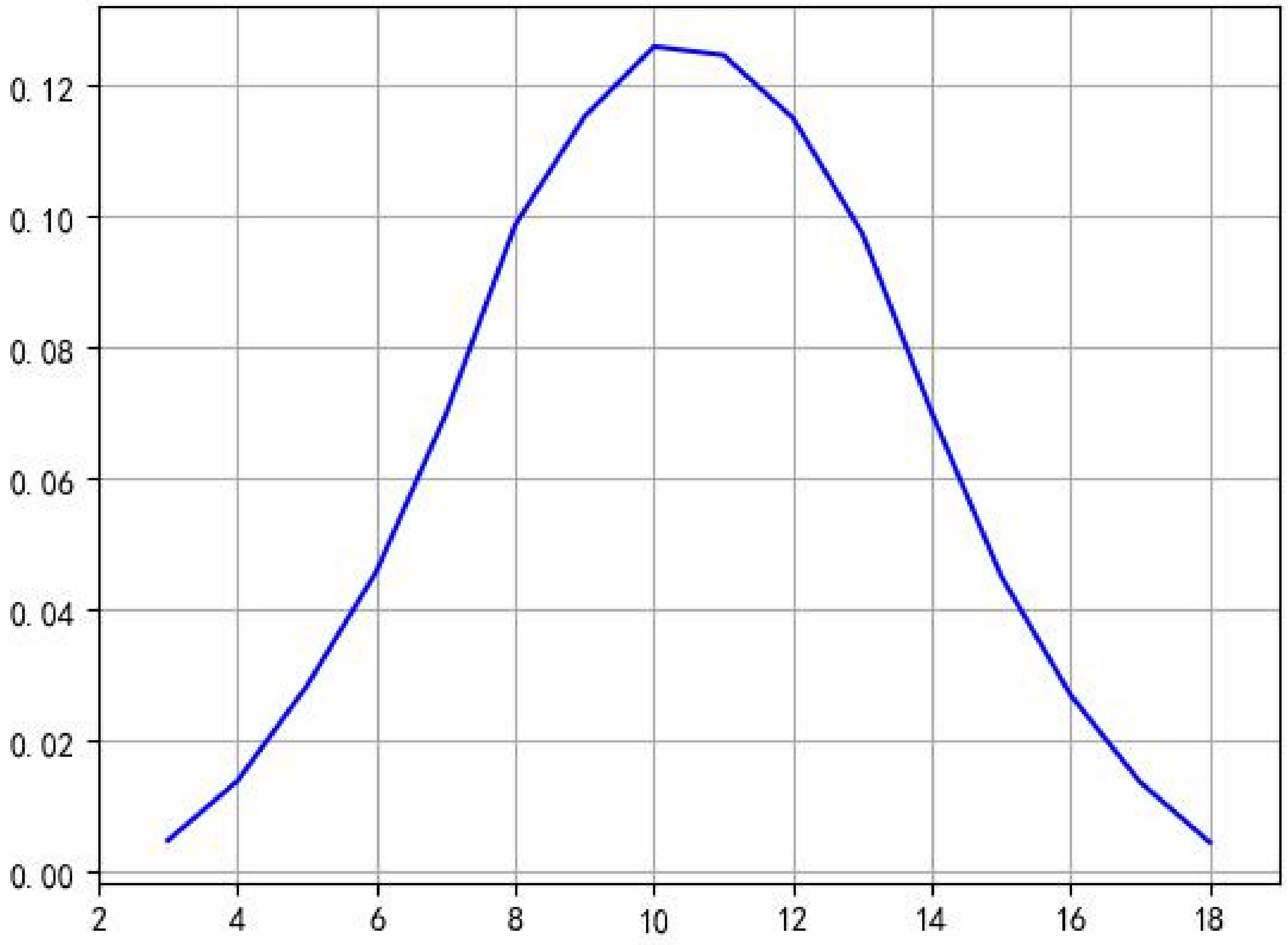
$$Y_n = X_1 + X_2 + \ldots + X_n = \sum_{i=1}^{n} X_i \to N\left(n\mu, n\sigma^2\right)$$

$$Z_n = \frac{Y_n - E\left(Y_n\right)}{\sqrt{D\left(Y_n\right)}} = \frac{Y_n - n\mu}{\sqrt{n}\sigma} \to N(0,1)$$

- 中心极限定理就是一般在同分布的情况下，抽样样本值的规范和在总体数量趋于无穷时的极限分布近似于正态分布。

中心极限定理

- 随机的抛六面的骰子，计算三次的点数的和，三次点数的和其实就是一个事件A，现在问事件A发生的概率以及事件A所属的分布是什么？



中心极限定理

```python
"""
随机的抛六面的骰子，计算三次的点数的和，三次点数的和其实就是一个事件A
==> 事件A的发生属于什么分布??
==> A = x1+x2+x3，其中x1、x2、x3是分别三次的抛骰子的点数
根据中心极限定理，由于x1、x2、x3属于独立同分布的，所以说最终的事件A属于高斯分布
"""
def generate_random_int():
    """随机产生一个[1,6]的数字，表示的是一个六面骰子的结果"""
    return random.randint(1, 6)

def generate_mean(n):
    """计算返回n次抛六面骰子的和结果"""
    return np.sum([generate_random_int() for i in range(n)])
```

```python
if __name__ == '__main__':
    # 进行A事件多少次
    number1 = 10000000
    # 表示每次A事件抛几次骰子
    number2 = 3

    # 进行number1次事件A的操作，每次事件A都进行number2次抛骰子
    keys = [generate_mean(number2) for i in range(number1)]

    # 统计每个和数字出现的次数，eg：  和为3的出现多少次、和为10出现多少次....
    result = {}
    for key in keys:
        count = 1
        if key in result:
            count += result[key]
        result[key] = count

    # 获取x和y
    x = sorted(np.unique(list(result.keys())))
    y = []
    for key in x:
        # 将出现的次数进行一个百分比的计算
        y.append(result[key] / number1)

    # 画图
    plt.plot(x, y, 'b-')
    plt.xlim(x[0] - 1, x[-1] + 1)
    plt.grid(True)
    plt.show()
```

最大似然估计

- **最大似然法(Maximum Likelihood Estimation, MLE)**也称为最大概似估计、极大似然估计，是一种具有理论性的参数估计方法。基本思想是：当从模型总体随机抽取n组样本观测值后，最合理的参数估计量应该使得从模型中抽取该n组样本观测值的概率最大；一般步骤如下：
  - 1. 写出似然函数；
  - 2. 对似然函数取对数，并整理；
  - 3. 求导数；
  - 4. 解似然方程

最大似然估计

北风网 IBEIFENG.COM

- 设总体分布为f(x,θ), $\{X_n\}$为该总体采样得到的样本。因为随机序列$\{X_n\}$独立同分布，则它们的联合密度函数为：

$$L(x_1,x_2,\ldots,x_n;\theta_1,\theta_2,\ldots,\theta_n)=\prod_{i=1}^{n} f(x_i;\theta_1,\theta_2,\ldots,\theta_n)$$

- 这里θ被看做固定但是未知的参数，反过来，因为样本已经存在，可以看做$\{X_n\}$是固定的，L(x,θ)是关于θ的函数，即似然函数；
- 求参数θ的值，使得似然函数取最大值，这种方法叫做**最大似然估计法**。

最大似然估计

- 若给定一组样本{X$_n$}，已知随机样本符合高斯分布N(μ,σ^2)，试估计σ和μ的值

$$f(x)=\frac{1}{\sqrt{2\pi\sigma}}\cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad L(x)=\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma}}\cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$l(x)=\log(L(x))=\log\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma}}\cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$=\sum_{i=1}^{n}\log\left(\frac{1}{\sqrt{2\pi\sigma}}\cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right)=\sum_{i=1}^{n}\log\left(\frac{1}{\sqrt{2\pi\sigma}}\right)-\sum_{i=1}^{n}-\frac{(x-\mu)^2}{2\sigma^2}=-\frac{n}{2}\log\left(2\pi\sigma^2\right)-\frac{1}{2\sigma^2}\sum_{i}(x-\mu)^2$$

最大似然估计

- 要求似然函数l(x)最大，即l(x)求极值即可，将似然函数对参数μ和σ分别求偏导数

$$l(x) = -\frac{n}{2}\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_i (x-\mu)^2$$

$$= -\frac{n}{2}\log(2\pi) - \left[n\log(\sigma) + \frac{1}{2\sigma^2}\sum_i (x-\mu)^2 \right] = -\frac{n}{2}\log(2\pi) - l_2(x)$$

$$\frac{l(x)}{d\mu} = -\frac{1}{\sigma^2}\sum_i (x_i - \mu) = 0 \Rightarrow \mu = \frac{1}{n}\sum_i x_i$$

$$\frac{l_2(x)}{d\sigma} = \frac{\pi}{\sigma} - \frac{\sum_i (x_i - \mu)^2}{\sigma^3} = 0 \Rightarrow \sigma^2 = \frac{1}{n}\sum_i (x_i - \mu)^2$$

向量的运算

- 设两向量为：$\vec{a}=(x_1,y_1)$，$\vec{b}=(x_2,y_2)$
- 向量的加法/减法满足平行四边形法则和三角形法则

$$\vec{a}+\vec{b}=(x_1+x_2,y_1+y_2)$$

$$\vec{a}-\vec{b}=(x_1-x_2,y_1-y_2)$$

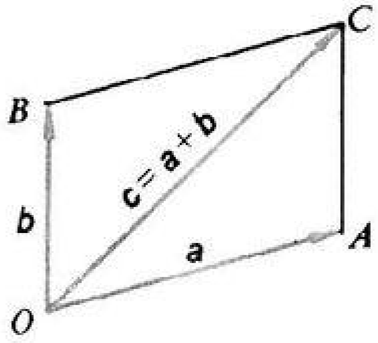


图 4　向量的加法

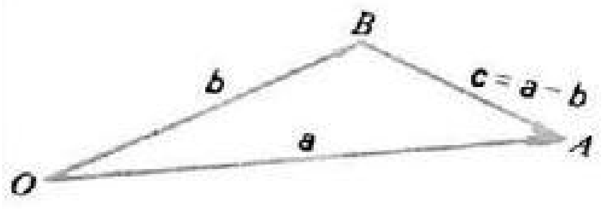


图 5　向量的减法

- 数乘：实数λ和向量a的叉乘乘积还是一个向量，记作λa，且|λa|=λ|a|；数乘的几何意义是将向量a进行伸长或者压缩操作

$$\lambda\vec{a}=(\lambda x_1,\lambda y_1)$$

向量的运算

- 设两向量为：$\vec{a}=(x_1,y_1)$，$\vec{b}=(x_2,y_2)$，并且a和b之间的夹角为:θ
- 数量积：两个向量的数量积(内积、点积)是一个数量/实数，记作 $\vec{a}\cdot\vec{b}$

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

- 向量积：两个向量的向量积(外积、叉积)是一个向量，记作 $\vec{a}\times\vec{b}$；

向量积即两个不共线非零向量所在平面的一组法向量。

$$\left|\vec{a}\times\vec{b}\right|=\left|\vec{a}\right|*\left|\vec{b}\right|*\sin\theta$$

矩阵的直观表示

- 数域F中m*n个数排成m行n列，并括以圆括弧(或方括弧)的数表示成为数域F上的矩阵，通常用大写字母记作A或者A_{m*n}，有时也记作$A=(a_{ij})_{m*n}(i=1,2...,m;j=1,2,...n)$，其中$a_{ij}$表示矩阵A的第i行的第j列元素，当F为实数域R时，A叫做实矩阵，当F为复数域C时，A叫做复矩阵。

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

矩阵的加减法

- 矩阵的加法与减法要求进行操作的两个矩阵A和B具有相同的阶，假设A为m*n阶矩阵，B为m*n阶矩阵，那么C=A ±B也是m*n阶的矩阵，并且矩阵C的元素满足：

$$c_{ij} = a_{ij} \pm b_{ij}$$

$$A = \begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Bmatrix} \quad B = \begin{Bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{Bmatrix} \quad C = A \pm B = \begin{Bmatrix} a_{11} \pm b_{11} & a_{12} \pm b_{12} & \dots & a_{1n} \pm b_{1n} \\ a_{21} \pm b_{21} & a_{22} \pm b_{22} & \dots & a_{2n} \pm b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} \pm b_{m1} & a_{m2} \pm b_{m2} & \dots & a_{mn} \pm b_{mn} \end{Bmatrix}$$

矩阵的加减法

- 记 $A_{m\times n}=(a_{ij})\quad B_{m\times n}=(b_{ij})$

加法： $A_{m\times n}+B_{m\times n}=(a_{ij}+b_{ij})$

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

减法：

$A_{m\times n}-B_{m\times n}=(a_{ij}-b_{ij})$

$$\begin{bmatrix}6&8\\11&14\end{bmatrix}-\begin{bmatrix}3&4\\7&8\end{bmatrix}=\begin{bmatrix}3&4\\4&6\end{bmatrix}$$

运算律：

交换律 $A+B=B+A$

结合律 $(A+B)+C=A+(B+C)$

矩阵与数的乘法

- 数乘：将数λ与矩阵A相乘，就是将数λ与矩阵A中的每一个元素相乘，记作λA；结果C=λA，并且C中的元素满足：

$$c_{ij} = \lambda a_{ij}$$

$$A = \left\{ \begin{array}{cccc} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{array} \right\} \qquad \lambda A = \left\{ \begin{array}{cccc} \lambda a_{11} & \lambda a_{12} & \dots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \dots & \lambda a_{2n} \\ \dots & \dots & \dots & \dots \\ \lambda a_{m1} & \lambda a_{m2} & \dots & \lambda a_{mn} \end{array} \right\}$$

矩阵与数的乘法

- 数乘： $\mathrm{k}A = (ka_{ij})$

$$3 \times \begin{bmatrix} 1 & 2 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 3\times 1 & 3\times 2 \\ 3\times 5 & 3\times 6 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 15 & 18 \end{bmatrix}$$

- 数乘运算律

结合律 $(\lambda\mu)A = \lambda(\mu A)$

分配律

$$(\lambda + \mu)A = \lambda A + \mu A$$

$$\lambda(A + B) = \lambda A + \lambda B$$

矩阵与向量的乘法

- 假设A为m*n阶矩阵，x为n*1的列向量，则Ax为m*1的列向量，

记作：$\vec{y} = A\vec{x}$

$$A = \begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Bmatrix} \qquad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \qquad \vec{y} = A\vec{x} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{pmatrix}$$

$$y_i = \sum_{j=1}^{n} a_{ij} x_j$$

矩阵与矩阵的乘法

- 矩阵的乘法仅当第一个矩阵A的列数和第二个矩阵B的行数相等时才能够定义，假设A为m*s阶矩阵，B为s*n阶矩阵，那么C=A*B是m*n阶矩阵，并且矩阵C中的元素满足:

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

$$A = \begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1s} \\ a_{21} & a_{22} & \dots & a_{2s} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{ms} \end{Bmatrix} \qquad B = \begin{Bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{s1} & b_{s2} & \dots & b_{sn} \end{Bmatrix} \qquad C = A*B = \begin{Bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{Bmatrix}$$

矩阵与矩阵的乘法

- 例：

$$A=\begin{bmatrix}1&0&-1&2\\-1&1&3&0\\0&5&-1&4\end{bmatrix}\qquad B=\begin{bmatrix}0&3&4\\1&2&1\\3&1&-1\\-1&2&1\end{bmatrix}$$

求C=AB

$$C=AB=\begin{bmatrix}1&0&-1&2\\-1&1&3&0\\0&5&-1&4\end{bmatrix}\begin{bmatrix}0&3&4\\1&2&1\\3&1&-1\\-1&2&1\end{bmatrix}=\begin{bmatrix}-5&6&7\\10&2&-6\\-2&17&10\end{bmatrix}$$

$$(AB)C=A(BC)\qquad (A+B)C=AC+BC\qquad C(A+B)=CA+CB$$

矩阵的转置

- 矩阵的转置：把矩阵A的行和列互相交换所产生的矩阵称为A的转置矩阵，这一过程叫做矩阵的转置。 使用A^T表示A的转置

$$A=\begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Bmatrix} \qquad A^T=\begin{Bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{Bmatrix}$$

矩阵的转置

- 例： $A=\begin{bmatrix}1 & 2\\ 5 & 6\end{bmatrix}$ ，A的转置为 $A^T=\begin{bmatrix}1 & 5\\ 2 & 6\end{bmatrix}$
- 转置的运算性质：

$$\left(A^T\right)^T=A$$

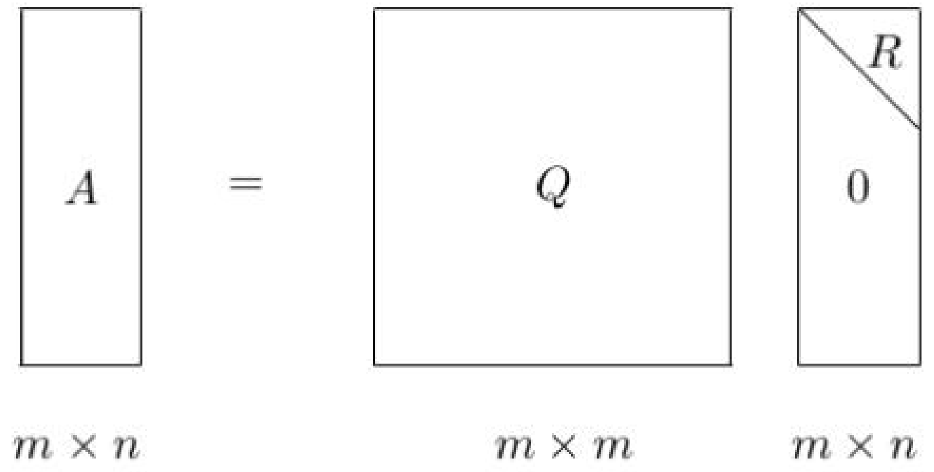
$$(\lambda A)^T=\lambda A^T$$

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

$$(A+B)^{\mathrm{T}}=A^{\mathrm{T}}+B^{\mathrm{T}}$$

QR分解

- QR分解是将矩阵分解为一个正交矩阵与上三角矩阵的乘积



- 这其中， Q为正交矩阵，$Q^TQ=I$，R为上三角矩阵。
- 实际中，QR分解经常被用来解线性最小二乘问题。

SVD分解

- **奇异值分解(Singular Value Decomposition)**是一种重要的矩阵分解方法，可以看做是对称方阵在任意矩阵上的推广。
- 假设A为一个m*n阶实矩阵，则存在一个分解使得：

$$A_{m*n} = U_{m*m} \Sigma_{m*n} V_{n*n}^{T}$$

  - 通常将奇异值由大到小排列，这样Σ便能由A唯一确定了。

向量的导数

- A为m*n的矩阵，x为n*1的列向量，则Ax为m*1的列向量，记作$\vec{y} = A \cdot \vec{x}$

$$A = \begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Bmatrix} \qquad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} \qquad \vec{y} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \\ \dots\dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \end{pmatrix}$$

$$\frac{\partial \vec{y}}{\partial \vec{x}} = \frac{\partial A \vec{x}}{\partial \vec{x}} = \begin{pmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{pmatrix} = A^T$$

向量的导数

- 向量偏导公式

$$\frac{\partial A\vec{x}}{\partial \vec{x}} = A^T$$

$$\frac{\partial A\vec{x}}{\partial \vec{x}^T} = A$$

$$\frac{\partial \left(\vec{x}^T A\right)}{\partial \vec{x}} = A$$

标量对向量的导数

- A为n*n的矩阵，x为n*1的列向量，记 $y = \vec{x}^T \cdot A \cdot \vec{x}$
- 同理可得：$$\frac{\partial y}{\partial \vec{x}} = \frac{\partial\left(\vec{x}^T \cdot A \cdot \vec{x}\right)}{\partial \vec{x}} = \left(A^T + A\right) \cdot \vec{x}$$
- 若A为对称矩阵，则有 $$\frac{\partial\left(\vec{x}^T A \vec{x}\right)}{\partial \vec{x}} = 2A\vec{x}$$

标量对向量的导数

$$A=\begin{Bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{Bmatrix} \qquad \vec{x}=\begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$$

$$\vec{x}^T \cdot A \cdot \vec{x} = (x_1, x_2 \dots x_n)\left(\sum_{j=1}^{n} a_{1j}x_j, \sum_{j=1}^{n} a_{2j}x_j, \dots \sum_{j=1}^{n} a_{nj}x_j\right)^T = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j$$

$$\frac{\partial\left(\vec{x}^T \cdot A \cdot \vec{x}\right)}{\partial \vec{x}} = \left(\sum_{j=1}^{n} a_{ij}x_j\right) + \left(\sum_{j=1}^{n} a_{ji}x_j\right) = \sum_{j=1}^{n}\left(a_{ij} + a_{ji}\right)x_j$$

标量对方阵的导数

- A为n*n的矩阵，|A|为A的行列式，计算 $\frac{\partial |A|}{\partial A}$

$$\forall 1 \le i \le n, |A| = \sum_{j=1}^{n} a_{ij} \cdot (-1)^{i+j} M_{ij}$$

$$\frac{\partial |A|}{\partial a_{ij}} = \frac{\partial \left(\sum_{j=1}^{n} a_{ij} \cdot (-1)^{i+j} M_{ij} \right)}{\partial a_{ij}} = (-1)^{i+j} M_{ij} = A^*{}_{ji}$$

$$\frac{\partial |A|}{\partial A} = \left(A^*\right)^T = |A| \cdot \left(A^{-1}\right)^T$$

数学知识回顾

- 常见函数
- 导数、**梯度**
- **Taylor公式**
- **联合概率、条件概率、全概率公式、贝叶斯公式**
- 期望、**方差、协方差**
- 大数定理、**中心极限定理**
- **最大似然估计(MLE)**
- **向量、矩阵的运算**
- 向量、矩阵的**求导**
- **SVD分解**、QR分解

Python科学计算库回顾

- Python科学计算库主要是为机器学习提供了一些便捷、封装好的API，那么在实际工作中，主要是将其应用在机器学习的特征工程阶段，主要涉及到的库有以下几个：
  - **NumPy**-数学计算基础库：N维数组、线性代数计算、傅立叶变换、随机数等；
  - **SciPy**-数值计算库：线性代数、拟合与优化、插值、数值积分、稀疏矩阵、图像处理、统计等；
  - **Pandas**-数据分析库：数据导入、整理、处理、分析等；
  - **Matplotlib**-会图库：绘制二维图形和图表。

Python科学计算库回顾

- 官网：https://scipy.org/

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

- 参考文档：
  - http://python.usyiyi.cn/
  - https://docs.scipy.org/doc/
  - http://pandas.pydata.org/pandas-docs/stable/index.html