# 02 Multiplication and Division

# 乘除

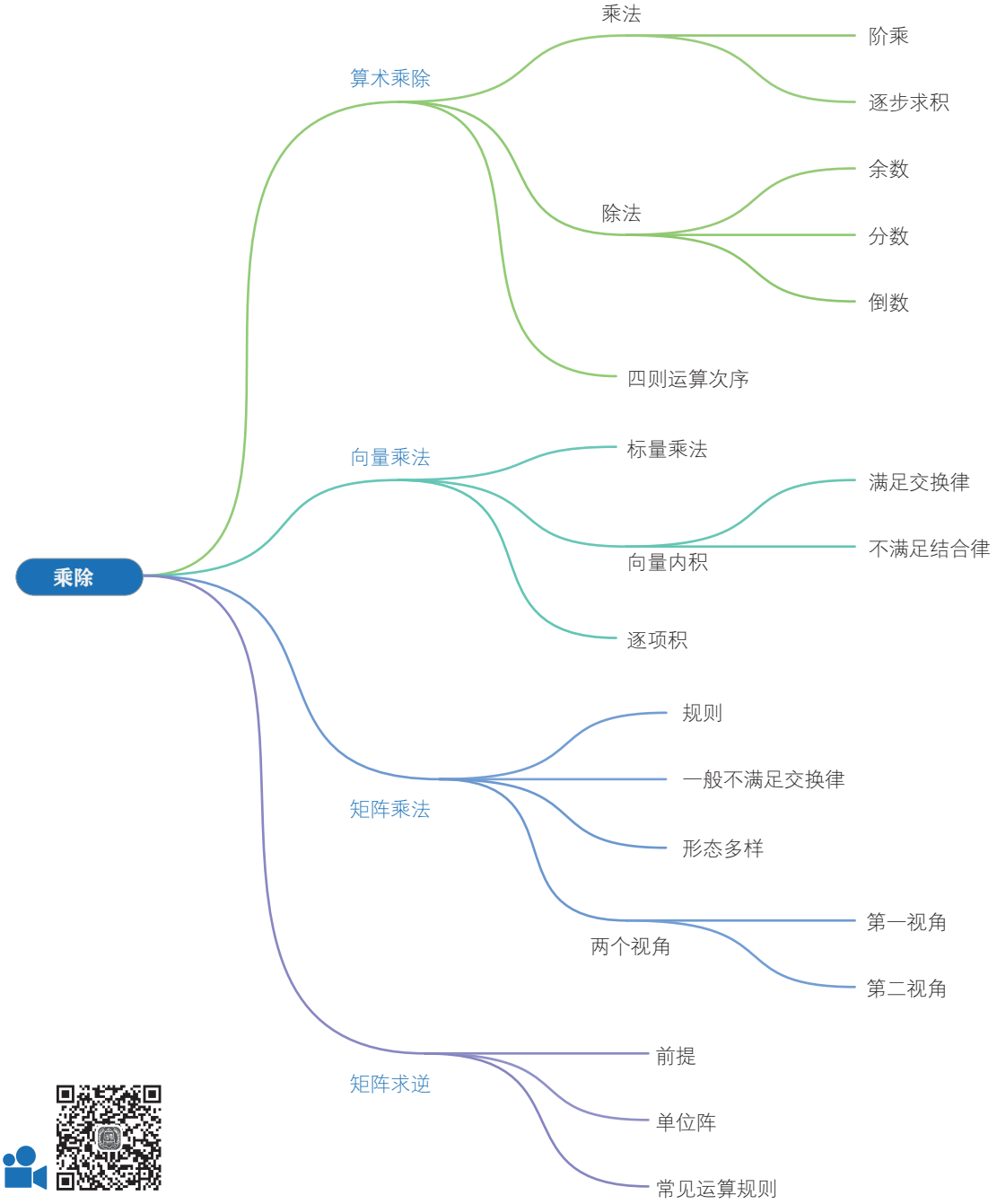
## 从九九乘法到矩阵乘法

大自然只使用最长的线来编织她的图景；因此，每根织线都能洞见整个大自然的锦绣图景。

***Nature uses only the longest threads to weave her patterns, so that each small piece of her fabric reveals the organization of the entire tapestry.***

—— 理查德 • 费曼 (Richard P. Feynman) | 美国理论物理学家 | 1918 — 1988

- `input()` 函数接受一个标准输入数据，返回为字符串 `str` 类型
- `int()` 将输入转化为整数
- `math.factorial()` 计算阶乘
- `numpy.cumprod()` 计算累计乘积
- `numpy.inner()` 计算行向量的内积，函数输入为两个列向量时得到的结果为张量积
- `numpy.linalg.inv()` 计算方阵的逆
- `numpy.linspace()` 在指定的间隔内，返回固定步长的数组
- `numpy.math.factorial()` 计算阶乘
- `numpy.random.seed()` 固定随机数发生器种子
- `numpy.random.uniform()` 产生满足连续均匀分布的随机数
- `numpy.sum()` 求和
- `scipy.special.factorial()` 计算阶乘
- `seaborn.heatmap()` 绘制热图


乘除
算术乘除
乘法
阶乘
逐步求积
除法
余数
分数
倒数
四则运算次序
向量乘法
标量乘法
向量内积
满足交换律
不满足结合律
逐项积
矩阵乘法
规则
一般不满足交换律
形态多样
两个视角
第一视角
第二视角
矩阵求逆
前提
单位阵
常见运算规则

# 2.1 算术乘除：先乘除，后加减，括号内先算

## 乘法

**乘法** (multiplication) 算式等号左端是**被乘数** (multiplicand) 和**乘数** (multiplier)，右端是**乘积** (product)，如图2.1所示。乘法运算符读作**乘** (times或multiplied by)。**乘法表** (multiplication table或times table) 是数字乘法运算的基础。


图2.1　乘法运算

图2.2所示为在数轴上可视化 $2 \times 3 = 6$。

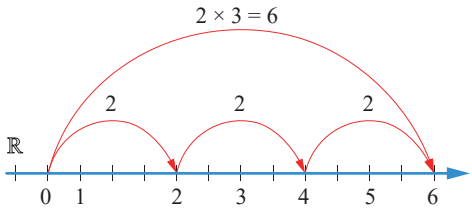



图2.2　$2 \times 3 = 6$ 在数轴上的可视化

介绍几个常用乘法符号。乘法符号 × 用于数字相乘，一般不用于两个变量相乘。而在线性代数中，× 表示**叉乘** (cross product或vector product)，完全是另外一回事。

在代数中，两个变量$a$和$b$相乘，可以写成$ab$；这种记法被称做**隐含乘法** (implied multiplication)。$ab$也可以写成$a \cdot b$。

通常，圆点 · 不用在数字相乘，因为它容易和**小数点** (decimal point) 混淆。线性代数中，$\boldsymbol{a} \cdot \boldsymbol{b}$表示$\boldsymbol{a}$和$\boldsymbol{b}$两个向量的**标量积** (scalar product)，这是本章后续要介绍的内容。

多提一嘴，乘法计算时，请大家多留意数值单位。举个例子，正方形的边长为1 m，其面积数值可以通过乘法运算1 × 1 = 1获得，而结果单位为平方米(m²)。有一些数值本身**无单位** (unitless)，如个数、Z分数。Z分数也叫**标准分数** (standard score)，是概率统计中的一个概念，Z分数是一个数与平均数的差再除以标准差的结果。

> 鸢尾花书会在《统计至简》一册详细介绍Z分数这个概念。

与乘法相关的常用英文表达见表2.1。

表2.1　乘法相关英文表达

| 数学表达 | 英文表达 |
| --- | --- |
| $2 \times 3 = 6$ | Two times three equals six.<br>Two multiplied by three equals six.<br>Two cross three equals six.<br>The product of two and three is six.<br>If you multiply two by three, you get six. |
| $a \cdot b = c$ | $a$ times $b$ equals $c$.<br>$a$ multiplied by $b$ equals $c$.<br>$a$ dot $b$ equals $c$.<br>The product of $a$ and $b$ is $c$. |

Bk3_Ch2_01.py完成两个数乘法。Python中两个数字相乘用 * (asterisk或star)。

## 阶乘

某个正整数的**阶乘** (factorial) 是所有小于及等于该数的正整数的积。比如，5的阶乘记作5!，对应的运算为

$$5! = 5 \times 4 \times 3 \times 2 \times 1 \tag{2.1}$$

特别地，定义0的阶乘为0! = 1。本书有两个重要的数学概念需要用到阶乘——排列组合和泰勒展开。

Python中可以用math.factorial()、scipy.special.factorial()、numpy.math.factorial() 计算阶乘。为了帮助大家理解，Bk3_Ch2_02.py自定义函数求解阶乘。

## 累计乘积

对于一组数字，**累计乘积** (cumulative product) 也叫**累积乘积**，得到的结果不仅仅是一个乘积，而是从左向右每乘一个数值得到的分步结果。比如，自然数1到10求累计乘积结果为

$$1,\quad 2,\quad 6,\quad 24,\quad 120,\quad 720,\quad 5040,\quad 40320,\quad 362880,\quad 3628800 \tag{2.2}$$

对应的累计乘积过程为

$$\begin{matrix} \overbrace{\overbrace{\overbrace{\overbrace{\overbrace{\overbrace{\overbrace{\overbrace{\overbrace{1}^{1}\times 2}^{2}\times 3}^{6}\times 4}^{24}\times 5}^{120}\times 6}^{720}}^{5040}\times 7}^{40320}\times 8}^{362880}\times 9}^{3628800}\times 10 \\ 1,\quad 2,\quad 3,\quad 4,\quad 5,\quad 6,\quad 7,\quad 8,\quad 9,\quad 10 \end{matrix} \tag{2.3}$$

Bk3_Ch2_03.py利用numpy.linspace(1, 10, 10) 产生1 ~ 10这十个自然数，然后利用numpy.cumprod()函数来求累计乘积。请大家自行研究如何使用numpy.arange()，并用这个函数生成1 ~ 10。

## 除法

**除法** (division) 是**乘法的逆运算** (reverse operation of multiplication)。**被除数** (dividend或numerator) **除以** (over或divided by) **除数** (divisor或denominator) 得到**商** (quotient)，如图2.3所示。

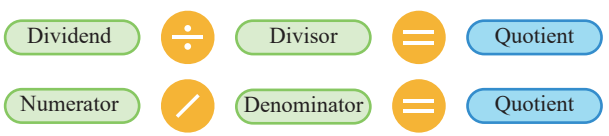



图2.3　除法运算

除法运算有时可以**除尽** (divisible)，如**6可以被3除尽** (six is divisible by three)。除法有时得到**余数** (remainder)，如7除2余1。除法的结果一般用**分数** (fraction) 或**小数** (decimal) 来表达，详见表2.2。

表2.2　除法英文表达

| 数学表达 | 英文表达 |
| --- | --- |
| $6 \div 3 = 2$ | Six divided by three equals two.<br>If you divide six by three you get two. |
| $7 \div 2 = 3R1$ | Seven over two is three and the remainder is one.<br>Seven divided by two equals three with a remainder of one.<br>Seven divided by two equals three and the remainder is one. |

Bk3_Ch2_04.py完成两个数的除法运算，除法运算符为正斜杠 /。
Bk3_Ch2_05.py介绍如何求余，求余数的运算符为 %。

## 分数

最常见的**分数** (fraction) 是**普通分数** (common fraction或simple fraction)，由**分母** (denominator) 和**分子** (numerator) 组成，分隔两者的是**分数线** (fraction bar) 或**正斜杠** (forward slash) /。

**非零整数** (nonzero integer) $a$的**倒数** (reciprocal) 是$1/a$。分数 $b/a$ 的倒数是$a/b$。$a$、$b$均不为0。

表2.3中总结了常用分数英文表达。

表2.3　分数相关英文表达

| 数学表达 | 英文表达 |
| --- | --- |
| $\frac{1}{2}$，　$1/2$ | One half<br>A half<br>One over two |
| $1:2$ | One to two |
| $-\frac{3}{2}$ | Minus three-halves<br>Negative three-halves |
| $\frac{1}{3}$，　$1/3$ | One over three<br>One third |
| $\frac{1}{4}$，　$1/4$ | One over four<br>One fourth<br>One quarter<br>One divided by four |
| $1\frac{1}{4}$ | One and one fourth |
| $1/5$ | One fifth |
| $3/5$ | Three fifths |
| $\frac{1}{n}$，　$1/n$ | One over $n$ |
| $\frac{a}{b}$，　$a/b$ | $a$ over $b$<br>$a$ divided by $b$<br>The ratio of $a$ to $b$<br>The numerator is $a$ while the denominator is $b$ |

# 2.2 向量乘法：标量乘法、向量内积、逐项积

这一节介绍三种重要的向量乘法：① **标量乘法** (scalar multiplication)；② **向量内积** (inner product)；③ **逐项积** (piecewise product)。

## 标量乘法

标量乘法运算中，标量乘向量的结果还是向量，相当于缩放。

标量乘法运算规则很简单，向量$\boldsymbol{a}$乘以$k$，$\boldsymbol{a}$的每一个元素均与$k$相乘，如下例标量2乘行向量[1, 2, 3]

$$2\times\begin{bmatrix}1 & 2 & 3\end{bmatrix}=\begin{bmatrix}2\times1 & 2\times2 & 2\times3\end{bmatrix}=\begin{bmatrix}2 & 4 & 6\end{bmatrix} \tag{2.4}$$

再如，标量乘列向量如

$$2\times\begin{bmatrix}1\\2\\3\end{bmatrix}=\begin{bmatrix}2\times1\\2\times2\\2\times3\end{bmatrix}=\begin{bmatrix}2\\4\\6\end{bmatrix} \tag{2.5}$$

图2.4所示为标量乘法示意图。

图2.4　标量乘法

同理，标量$k$乘矩阵$\boldsymbol{A}$的结果是$k$与矩阵$\boldsymbol{A}$每一个元素相乘，比如

$$2\times\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}_{2\times3}=\begin{bmatrix}2\times1 & 2\times2 & 2\times3\\2\times4 & 2\times5 & 2\times6\end{bmatrix}_{2\times3}=\begin{bmatrix}2 & 4 & 6\\8 & 10 & 12\end{bmatrix}_{2\times3} \tag{2.6}$$

Bk3_Ch2_06.py完成向量和矩阵标量乘法。

## 向量内积

**向量内积** (inner product) 的结果为标量。向量内积又叫**标量积** (scalar product) 或**点积** (dot product)。

向量内积的运算规则是：两个形状相同的向量，对应位置元素一一相乘后再求和。比如，下例计算两个行向量内积

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix}\cdot\begin{bmatrix}4 & 3 & 2\end{bmatrix}=1\times4+2\times3+3\times2=4+6+6=16 \tag{2.7}$$

计算两个列向量内积，比如：

$$\begin{bmatrix}1\\2\\3\end{bmatrix}\cdot\begin{bmatrix}-1\\0\\1\end{bmatrix}=1\times(-1)+2\times0+3\times1=-1+0+3=2 \tag{2.8}$$

图2.5所示为向量内积规则的示意图。


图2.5　向量内积示意图

显然，向量内积满足**交换律** (commutative)，即

$$\boldsymbol{a}\cdot\boldsymbol{b}=\boldsymbol{b}\cdot\boldsymbol{a} \tag{2.9}$$

向量内积**对向量加法满足分配律** (distributive over vector addition)，即

$$\boldsymbol{a}\cdot(\boldsymbol{b}+\boldsymbol{c})=\boldsymbol{a}\cdot\boldsymbol{b}+\boldsymbol{a}\cdot\boldsymbol{c} \tag{2.10}$$

显然，向量内积不满足**结合律** (associative)，即

$$(\boldsymbol{a}\cdot\boldsymbol{b})\cdot\boldsymbol{c}\neq\boldsymbol{a}\cdot(\boldsymbol{b}\cdot\boldsymbol{c}) \tag{2.11}$$

Bk3_Ch2_07.py代码用numpy.inner() 计算行向量的内积；但是，numpy.inner() 函数输入为两个列向量时得到的结果为**张量积** (tensor product)。

机器学习和深度学习中，张量积是非常重要的向量运算，鸢尾花书将在《矩阵力量》一册中进行详细介绍。

下面举几个例子，让大家管窥标量积的用途。

给定以下五个数字，即

$$1,\quad 2,\quad 3,\quad 4,\quad 5 \tag{2.12}$$

这五个数字求和，可以用标量积计算得到，即

$$\begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix}\cdot\begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}=1\times1+2\times1+3\times1+4\times1+5\times1=15 \tag{2.13}$$

前文提过，$[1, 1, 1, 1, 1]^T$ 叫作全1向量。

这五个数字的平均值，也可以通过标量积得到，即

$$\begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix}\cdot\begin{bmatrix}1/5 & 1/5 & 1/5 & 1/5 & 1/5\end{bmatrix}=1\times1/5+2\times1/5+3\times1/5+4\times1/5+5\times1/5=3 \tag{2.14}$$

计算五个数字的平方和，有

$$\begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix}\cdot\begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix}=1\times1+2\times2+3\times3+4\times4+5\times5=55 \tag{2.15}$$

此外，标量积还有重要的几何意义。本书后续将介绍这方面内容。

## 逐项积

**逐项积** (piecewise product)，也叫**阿达玛乘积** (hadamard product)。两个相同形状向量的逐项积为对应位置元素分别相乘，结果为相同形状的向量。

逐项积的运算符为 ⊙。逐项积相当于算术乘法的批量运算。

举个例子，两个行向量逐项积如

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix}\odot\begin{bmatrix}4 & 5 & 6\end{bmatrix}=\begin{bmatrix}1\times4 & 2\times5 & 3\times6\end{bmatrix}=\begin{bmatrix}4 & 10 & 18\end{bmatrix} \tag{2.16}$$

图2.6所示为向量逐项积运算示意图。

图2.6　向量逐项积

同理，两个矩阵逐项积的运算前提是——矩阵形状相同。矩阵逐项积运算规则为对应元素相乘，结果形状不变，如

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}_{2\times3}\odot\begin{bmatrix}1 & 2 & 3\\ -1 & 0 & 1\end{bmatrix}_{2\times3}=\begin{bmatrix}1\times1 & 2\times2 & 3\times3\\ 4\times(-1) & 5\times0 & 6\times1\end{bmatrix}_{2\times3}=\begin{bmatrix}1 & 4 & 9\\ -4 & 0 & 6\end{bmatrix}_{2\times3} \tag{2.17}$$

Python中，对于numpy.array() 定义的形状相同的向量或矩阵，逐项积可以通过 * 计算得到。请大家参考Bk3_Ch2_08.py。更多有关NumPy用法，请参考《编程不难》。

# 2.3 矩阵乘法：最重要的线性代数运算规则

矩阵乘法是最重要线性代数运算，没有之一——这句话并不夸张。

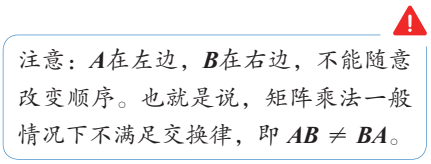
矩阵乘法规则可以视作算术“九九乘法表”的进阶版。

## 矩阵乘法规则

$\boldsymbol{A}$和$\boldsymbol{B}$两个矩阵相乘的前提是矩阵$\boldsymbol{A}$的列数和矩阵$\boldsymbol{B}$的行数相同。$\boldsymbol{A}$和$\boldsymbol{B}$的乘积一般写作$\boldsymbol{AB}$。

$\boldsymbol{A}$和$\boldsymbol{B}$两个矩阵相乘$\boldsymbol{AB}$读作“matrix boldface capital $\boldsymbol{A}$ times matrix boldface capital $\boldsymbol{B}$”或“the matrix product boldface capital $\boldsymbol{A}$ and boldface capital $\boldsymbol{B}$”。

NumPy中，两个矩阵相乘的运算符为 @，鸢尾花书一部分矩阵乘法也会采用 @。比如，$\boldsymbol{AB}$也记做$\boldsymbol{A}@\boldsymbol{B}$：



注意：$\boldsymbol{A}$在左边，$\boldsymbol{B}$在右边，不能随意改变顺序。也就是说，矩阵乘法一般情况下不满足交换律，即 $\boldsymbol{AB} \neq \boldsymbol{BA}$。

$$\boldsymbol{C}_{m\times n} = \boldsymbol{A}_{m\times p}\boldsymbol{B}_{p\times n} = \boldsymbol{A}_{m\times p} @ \boldsymbol{B}_{p\times n} \tag{2.18}$$

如图2.7所示，矩阵$\boldsymbol{A}$的形状为$m$行、$p$列，矩阵$\boldsymbol{B}$的形状为$p$行、$n$列。$\boldsymbol{A}$和$\boldsymbol{B}$相乘得到矩阵$\boldsymbol{C}$，$\boldsymbol{C}$的形状为$m$行、$n$列，相当于消去了$p$。

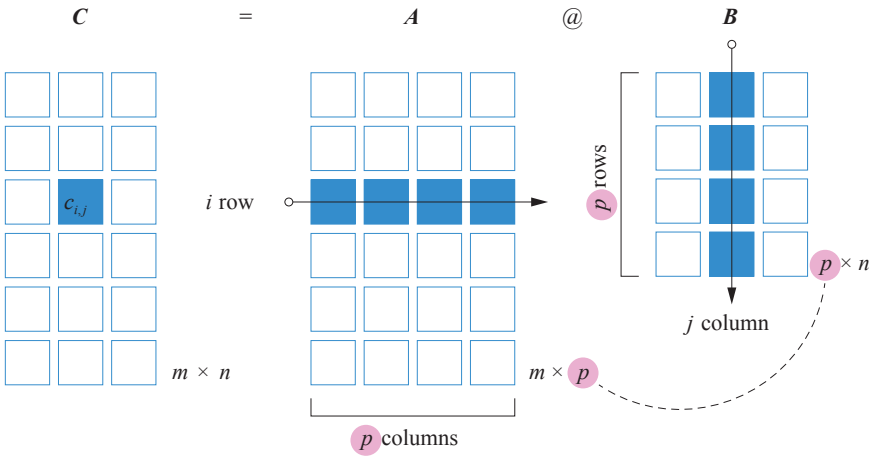



图2.7　矩阵乘法规则

再次强调，矩阵乘法不满足交换律。也就是说，一般情况下下式不成立，即

$$\boldsymbol{A}_{m\times p}\boldsymbol{B}_{p\times n} \neq \boldsymbol{B}_{p\times n}\boldsymbol{A}_{m\times p} \tag{2.19}$$

首先，$\boldsymbol{B}$的列数和$\boldsymbol{A}$的行数很可能不匹配。即便$m = n$，也就是$\boldsymbol{B}$的列数等于$\boldsymbol{A}$的行数，$\boldsymbol{BA}$结果也很可能不等于$\boldsymbol{AB}$。

## 两个 2×2 矩阵相乘

下面，用两个 2 × 2 矩阵相乘讲解矩阵乘法运算规则。

设矩阵$\boldsymbol{A}$和$\boldsymbol{B}$相乘结果为矩阵$\boldsymbol{C}$，有

$$\boldsymbol{C} = \boldsymbol{AB} = \underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_{\boldsymbol{A}}\underbrace{\begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}}_{\boldsymbol{B}} = \underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_{\boldsymbol{A}} @ \underbrace{\begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}}_{\boldsymbol{B}} \tag{2.20}$$

图2.8所示为两个2 × 2矩阵相乘如何得到矩阵$\boldsymbol{C}$的每一个元素。

矩阵$\boldsymbol{A}$的第一行元素和矩阵$\boldsymbol{B}$第一列对应元素分别相乘，再相加，结果为矩阵$\boldsymbol{C}$的第一行、第一列元素$c_{1,1}$。

矩阵$\boldsymbol{A}$的第一行元素和矩阵$\boldsymbol{B}$第二列对应元素分别相乘，再相加，得到$c_{1,2}$。

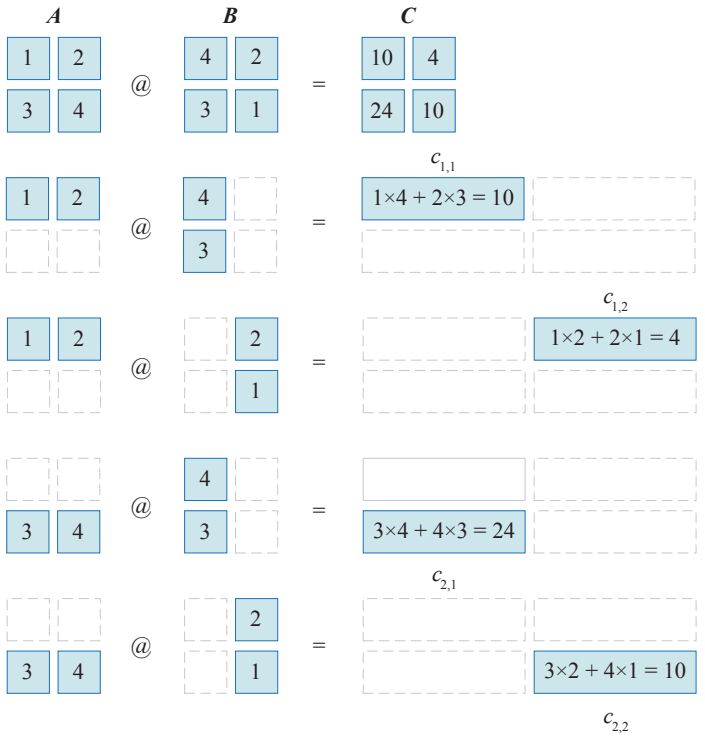



图2.8　矩阵乘法规则，两个2 × 2矩阵相乘为例

同理，依次获得矩阵$\boldsymbol{C}$的$c_{2,1}$和$c_{2,2}$两个元素。

总结来说，$\boldsymbol{A}$和$\boldsymbol{B}$乘积$\boldsymbol{C}$的第$i$行第$j$列的元素$c_{i,j}$等于矩阵$\boldsymbol{A}$的第$i$行的元素与矩阵$\boldsymbol{B}$的第$j$列对应元素乘积再求和。

注意：这个矩阵运算规则既是一种发明创造，也是一种约定成俗。也就是说，这种乘法规则在被法国数学家雅克·菲利普·玛丽·比内 (Jacques Philippe Marie Binet, 1786 — 1856) 提出之后，在长期的数学实践中被广为接受。矩阵乘法可谓“成人版九九乘法表”。就像大家儿时背诵九九乘法表时一样，这里建议大家先把矩阵乘法规则背下来，熟能生巧，慢慢地大家就会通过不断学习认识到这个乘法规则的精妙之处。

Bk3_Ch2_09.py展示如何完成矩阵乘法运算。

## 矩阵乘法形态

图2.9所示给出了常见的多种矩阵乘法形态，每一种形态对应一类线性代数问题。图2.9中特别高亮显示出矩阵乘法中左侧矩阵的“列”和右侧矩阵的“行”。鸢尾花书《矩阵力量》一册将会详细介绍图2.9每一种乘法形态。

这里特别提醒大家，初学者对矩阵乘法会产生一种错误印象，认为这些千奇百怪的矩阵乘法形态就是“奇技淫巧”。这是极其错误的想法！在不断学习中，大家会逐渐领略到每种矩阵乘法形态的力量所在。

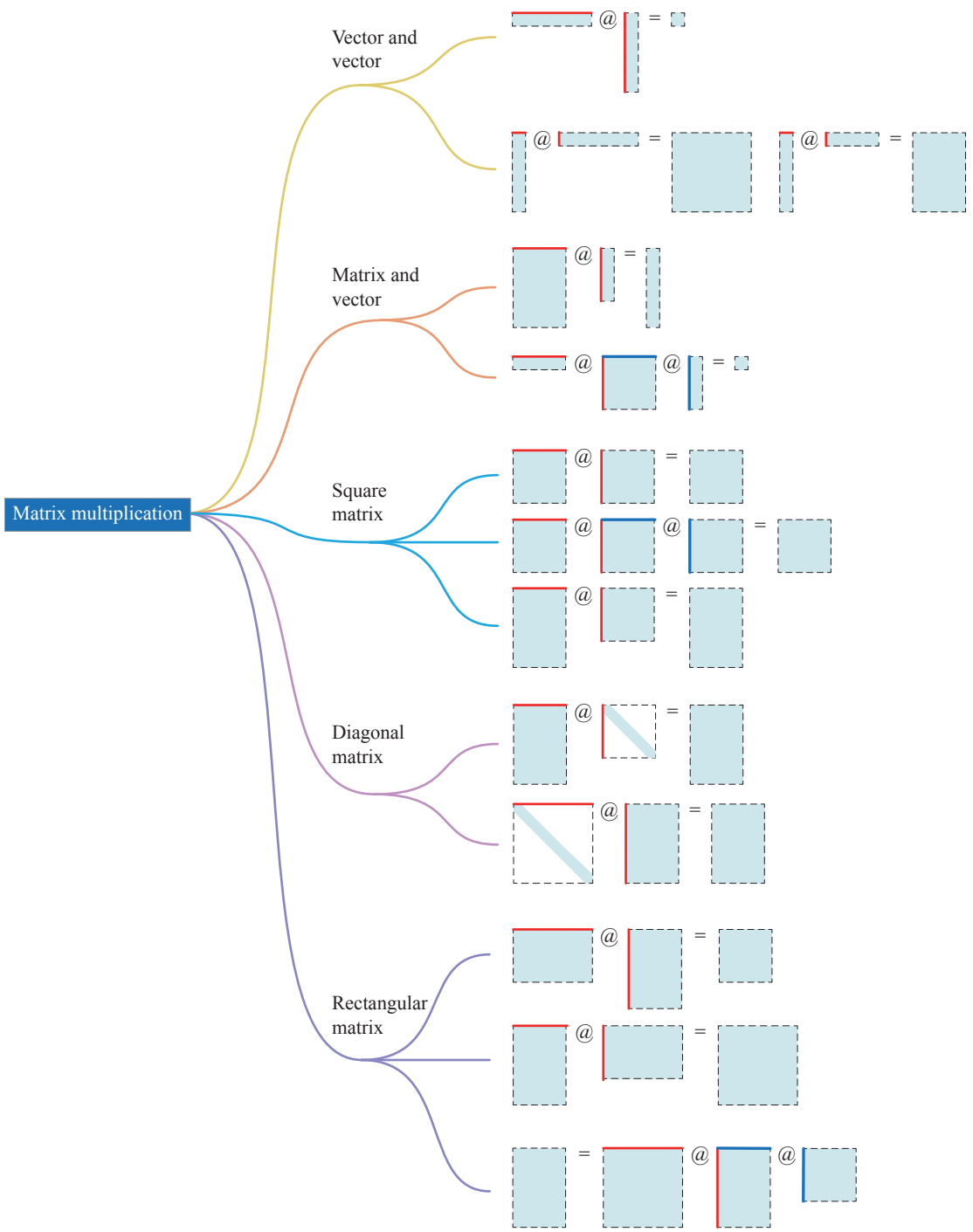



图2.9　矩阵乘法形态多样性

## 两个向量相乘

本节最后着重讲一下图2.9最上面两种向量的乘积。这两种特殊形态的矩阵乘法正是理解矩阵乘法规则的两个重要视角。

向量$\boldsymbol{a}$和$\boldsymbol{b}$为等长列向量，$\boldsymbol{a}$转置 ($\boldsymbol{a}^{\mathrm{T}}$) 乘$\boldsymbol{b}$为标量，等价于$\boldsymbol{a}$和$\boldsymbol{b}$的标量积，即

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}=\boldsymbol{a}\cdot\boldsymbol{b} \tag{2.21}$$

举个例子：

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}=\begin{bmatrix}1\\2\\3\end{bmatrix}_{3\times1}^{\mathrm{T}}@\begin{bmatrix}4\\3\\2\end{bmatrix}_{3\times1}=\begin{bmatrix}1&2&3\end{bmatrix}_{1\times3}@\begin{bmatrix}4\\3\\2\end{bmatrix}_{3\times1}=16 \tag{2.22}$$

列向量$\boldsymbol{a}$乘$\boldsymbol{b}$转置 ($\boldsymbol{b}^{\mathrm{T}}$)，乘积结果$\boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}$为方阵，也就是行数和列数相同的矩阵，即

$$\boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}=\begin{bmatrix}1\\2\\3\end{bmatrix}_{3\times1}@\begin{bmatrix}4\\3\\2\end{bmatrix}_{3\times1}^{\mathrm{T}}=\begin{bmatrix}1\\2\\3\end{bmatrix}_{3\times1}@\begin{bmatrix}4&3&2\end{bmatrix}_{1\times3}=\begin{bmatrix}4&3&2\\8&6&4\\12&9&6\end{bmatrix}_{3\times3} \tag{2.23}$$

如果$\boldsymbol{a}$和$\boldsymbol{b}$分别为不等长列向量，请大家自行计算$\boldsymbol{a}\boldsymbol{b}^{\mathrm{T}}$的结果：

$$\boldsymbol{a}=\begin{bmatrix}1\\2\end{bmatrix}_{2\times1},\quad \boldsymbol{b}=\begin{bmatrix}4\\3\\2\end{bmatrix}_{3\times1} \tag{2.24}$$

再次强调：使用numpy.array() 构造向量时，np.array([1,2]) 构造的是一维数组，不能算是矩阵。而np.array([[1,2]]) 构造得到的相当于1 × 2行向量，是一个特殊矩阵。注意，《编程不难》专门区分数组、向量、矩阵等概念。

np.array([[1],[2]]) 构造的是一个2 × 1列向量，也是个矩阵。鸢尾花书会在《矩阵力量》一册介绍更多构造行向量和列向量的方法。

# 2.4 矩阵乘法第一视角

这一节探讨矩阵乘法的第一视角。

## 两个 2×2 矩阵相乘

上一节最后介绍，$\boldsymbol{a}$和$\boldsymbol{b}$均是形状为$n$ × 1的列向量，$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$结果为标量，相当于标量积$\boldsymbol{a}\cdot\boldsymbol{b}$。我们可以把式 (2.20) 中$\boldsymbol{A}$写成两个行向量$\boldsymbol{a}^{(1)}$ 和 $\boldsymbol{a}^{(2)}$，把$\boldsymbol{B}$写成两个列向量 $\boldsymbol{b}_1$和$\boldsymbol{b}_2$，即

$$\boldsymbol{A}=\begin{bmatrix}\underbrace{\begin{bmatrix}1&2\end{bmatrix}}_{\boldsymbol{a}^{(1)}}\\\underbrace{\begin{bmatrix}3&4\end{bmatrix}}_{\boldsymbol{a}^{(2)}}\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}\underbrace{\begin{bmatrix}4\\3\end{bmatrix}}_{\boldsymbol{b}_1}&\underbrace{\begin{bmatrix}2\\1\end{bmatrix}}_{\boldsymbol{b}_2}\end{bmatrix} \tag{2.25}$$

这样$\boldsymbol{AB}$矩阵乘积可以写成

$$\boldsymbol{A}\text{@}\boldsymbol{B}=\underbrace{\begin{bmatrix}\underbrace{\begin{bmatrix}1 & 2\end{bmatrix}}_{\boldsymbol{a}^{(1)}}\\ \underbrace{\begin{bmatrix}3 & 4\end{bmatrix}}_{\boldsymbol{a}^{(2)}}\end{bmatrix}}_{\boldsymbol{A}}\text{@}\underbrace{\begin{bmatrix}\underbrace{\begin{bmatrix}4\\3\end{bmatrix}}_{\boldsymbol{b}_1} & \underbrace{\begin{bmatrix}2\\1\end{bmatrix}}_{\boldsymbol{b}_2}\end{bmatrix}}_{\boldsymbol{B}}=\begin{bmatrix}\begin{bmatrix}1 & 2\end{bmatrix}\text{@}\begin{bmatrix}4\\3\end{bmatrix} & \begin{bmatrix}1 & 2\end{bmatrix}\text{@}\begin{bmatrix}2\\1\end{bmatrix}\\ \begin{bmatrix}3 & 4\end{bmatrix}\text{@}\begin{bmatrix}4\\3\end{bmatrix} & \begin{bmatrix}3 & 4\end{bmatrix}\text{@}\begin{bmatrix}2\\1\end{bmatrix}\end{bmatrix}=\begin{bmatrix}10 & 4\\24 & 10\end{bmatrix} \tag{2.26}$$

也就是说，将位于矩阵乘法左侧的$\boldsymbol{A}$写成行向量，右侧的$\boldsymbol{B}$写成列向量。然后，行向量和列向量逐步相乘，得到乘积每个位置的元素。

用符号代替具体数字， 可以写成

$$\begin{aligned}\boldsymbol{A}\text{@}\boldsymbol{B}&=\begin{bmatrix}\begin{bmatrix}a_{1,1} & a_{1,2}\end{bmatrix}_{1\times 2}\\ \begin{bmatrix}a_{2,1} & a_{2,2}\end{bmatrix}_{1\times 2}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}b_{1,1}\\b_{2,1}\end{bmatrix}_{2\times 1} & \begin{bmatrix}b_{1,2}\\b_{2,2}\end{bmatrix}_{2\times 1}\end{bmatrix}\\ &=\begin{bmatrix}\boldsymbol{a}^{(1)}\\ \boldsymbol{a}^{(2)}\end{bmatrix}_{2\times 1}\begin{bmatrix}\boldsymbol{b}_1 & \boldsymbol{b}_2\end{bmatrix}_{1\times 2}=\begin{bmatrix}\boldsymbol{a}^{(1)}\boldsymbol{b}_1 & \boldsymbol{a}^{(1)}\boldsymbol{b}_2\\ \boldsymbol{a}^{(2)}\boldsymbol{b}_1 & \boldsymbol{a}^{(2)}\boldsymbol{b}_2\end{bmatrix}_{2\times 2}=\begin{bmatrix}\left(\boldsymbol{a}^{(1)}\right)^{\mathrm{T}}\cdot\boldsymbol{b}_1 & \left(\boldsymbol{a}^{(1)}\right)^{\mathrm{T}}\cdot\boldsymbol{b}_2\\ \left(\boldsymbol{a}^{(2)}\right)^{\mathrm{T}}\cdot\boldsymbol{b}_1 & \left(\boldsymbol{a}^{(2)}\right)^{\mathrm{T}}\cdot\boldsymbol{b}_2\end{bmatrix}_{2\times 2}\end{aligned} \tag{2.27}$$

式 (2.27) 展示的是矩阵乘法的基本视角，它直接体现出来的是矩阵乘法规则。

再次强调：$\boldsymbol{a}^{(1)}$ 是行向量，$\boldsymbol{b}_1$ 是列向量。

## 更一般情况

矩阵乘积$\boldsymbol{AB}$中，左侧矩阵$\boldsymbol{A}$的形状为$m \times p$，将矩阵$\boldsymbol{A}$写成一组上下叠放的行向量$\boldsymbol{a}^{(i)}$，即

$$\boldsymbol{A}_{m\times p}=\begin{bmatrix}\boldsymbol{a}^{(1)}\\ \boldsymbol{a}^{(2)}\\ \vdots\\ \boldsymbol{a}^{(m)}\end{bmatrix}_{m\times 1} \tag{2.28}$$

其中：行向量$\boldsymbol{a}^{(i)}$ 列数为$p$，即有$p$个元素。

矩阵乘积$\boldsymbol{AB}$中，右侧矩阵$\boldsymbol{B}$的形状为$p \times n$列，将矩阵$\boldsymbol{B}$写成左右排列的列向量，即

$$\boldsymbol{B}_{p\times n}=\begin{bmatrix}\boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_n\end{bmatrix}_{1\times n} \tag{2.29}$$

其中：列向量$\boldsymbol{b}_j$ 行数为$p$，也有$p$个元素。

$\boldsymbol{A}$和$\boldsymbol{B}$相乘，可以展开写成

$$\boldsymbol{A}_{m\times p}\text{@}\boldsymbol{B}_{p\times n}=\begin{bmatrix}\boldsymbol{a}^{(1)}\\ \boldsymbol{a}^{(2)}\\ \vdots\\ \boldsymbol{a}^{(m)}\end{bmatrix}_{m\times 1}\begin{bmatrix}\boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_n\end{bmatrix}_{1\times n}=\begin{bmatrix}\boldsymbol{a}^{(1)}\boldsymbol{b}_1 & \boldsymbol{a}^{(1)}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}^{(1)}\boldsymbol{b}_n\\ \boldsymbol{a}^{(2)}\boldsymbol{b}_1 & \boldsymbol{a}^{(2)}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}^{(2)}\boldsymbol{b}_n\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{a}^{(m)}\boldsymbol{b}_1 & \boldsymbol{a}^{(m)}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}^{(m)}\boldsymbol{b}_n\end{bmatrix}_{m\times n}=\boldsymbol{C}_{m\times n} \tag{2.30}$$

## 热图

图2.10所示为**热图**(heatmap) 可视化矩阵乘法。

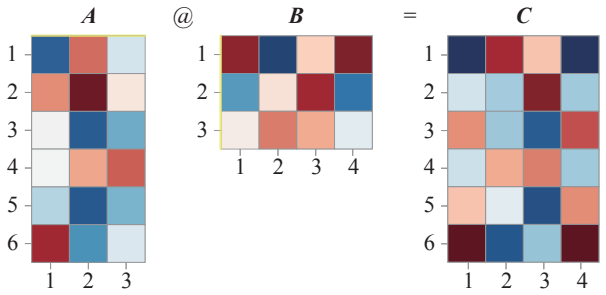



图2.10　矩阵乘法热图展示

具体如图2.11所示，$\boldsymbol{A}$中的第$i$行向量 $\boldsymbol{a}^{(i)}$ 乘以$\boldsymbol{B}$中的第$j$列向量 $\boldsymbol{b}_j$，得到标量$\boldsymbol{a}^{(i)}\boldsymbol{b}_j$，对应乘积矩阵$\boldsymbol{C}$中第$i$行、第$j$列元素$c_{i,j}$，即

$$c_{i,j} = \boldsymbol{a}^{(i)}\boldsymbol{b}_j \tag{2.31}$$

这就是矩阵乘法的第一视角。

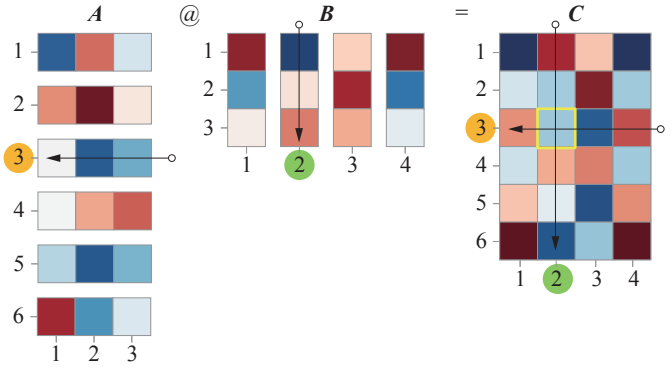



图2.11　矩阵乘法第一视角

代码文件Bk3_Ch2_10.py中Bk3_Ch2_10_A部分代码用于绘制图2.10。

代码用numpy.random.uniform() 函数产生满足连续均匀分布的随机数，并用seaborn.heatmap() 绘制热图。热图采用的colormap为'RdBu_r'，'Rd'是红色的意思，'Bu'是蓝色，'_r'代表“翻转”。

此外，我们还用Streamlit制作了展示矩阵乘法运算规则的App，请大家参考代码文件Streamlit_Bk3_Ch2_10.py。文件中还展示了如何使用try-except。

# 2.5 矩阵乘法第二视角

下面，我们聊一聊矩阵乘法的第二视角。

## 两个 2×2 矩阵相乘

还是以式 (2.20) 为例，$\boldsymbol{A}$和$\boldsymbol{B}$相乘，把左侧矩阵$\boldsymbol{A}$写成两个列向量$\boldsymbol{a}_1$ 和 $\boldsymbol{a}_2$，把右侧矩阵$\boldsymbol{B}$写成两个行向量 $\boldsymbol{b}^{(1)}$和$\boldsymbol{b}^{(2)}$，即

$$\boldsymbol{A}=\begin{bmatrix}\underbrace{\begin{bmatrix}1\\3\end{bmatrix}}_{\boldsymbol{a}_1} & \underbrace{\begin{bmatrix}2\\4\end{bmatrix}}_{\boldsymbol{a}_2}\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}\underbrace{\begin{bmatrix}4 & 2\end{bmatrix}}_{\boldsymbol{b}^{(1)}}\\ \underbrace{\begin{bmatrix}3 & 1\end{bmatrix}}_{\boldsymbol{b}^{(2)}}\end{bmatrix} \tag{2.32}$$

这样$\boldsymbol{AB}$乘积可以展开写成

$$\boldsymbol{A}\text{@}\boldsymbol{B}=\underbrace{\begin{bmatrix}\underbrace{\begin{bmatrix}1\\3\end{bmatrix}}_{\boldsymbol{a}_1} & \underbrace{\begin{bmatrix}2\\4\end{bmatrix}}_{\boldsymbol{a}_2}\end{bmatrix}}_{\boldsymbol{A}}\text{@}\underbrace{\begin{bmatrix}\underbrace{\begin{bmatrix}4 & 2\end{bmatrix}}_{\boldsymbol{b}^{(1)}}\\ \underbrace{\begin{bmatrix}3 & 1\end{bmatrix}}_{\boldsymbol{b}^{(2)}}\end{bmatrix}}_{\boldsymbol{B}}=\underbrace{\begin{bmatrix}1\\3\end{bmatrix}}_{\boldsymbol{a}_1}\text{@}\underbrace{\begin{bmatrix}4 & 2\end{bmatrix}}_{\boldsymbol{b}^{(1)}}+\underbrace{\begin{bmatrix}2\\4\end{bmatrix}}_{\boldsymbol{a}_2}\text{@}\underbrace{\begin{bmatrix}3 & 1\end{bmatrix}}_{\boldsymbol{b}^{(2)}}=\begin{bmatrix}4 & 2\\12 & 6\end{bmatrix}+\begin{bmatrix}6 & 2\\12 & 4\end{bmatrix}=\begin{bmatrix}10 & 4\\24 & 10\end{bmatrix} \tag{2.33}$$

在这个视角下，我们惊奇地发现矩阵乘法竟然变成了“加法”！

用符号代替数字， 可以写成

$$\begin{aligned}\boldsymbol{A}\text{@}\boldsymbol{B}&=\begin{bmatrix}\begin{bmatrix}a_{1,1}\\a_{2,1}\end{bmatrix}_{2\times1} & \begin{bmatrix}a_{1,2}\\a_{2,2}\end{bmatrix}_{2\times1}\end{bmatrix}\begin{bmatrix}\begin{bmatrix}b_{1,1} & b_{1,2}\end{bmatrix}_{1\times2}\\ \begin{bmatrix}b_{2,1} & b_{2,2}\end{bmatrix}_{1\times2}\end{bmatrix}\\&=\begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{a}_2\end{bmatrix}_{1\times2}\begin{bmatrix}\boldsymbol{b}^{(1)}\\ \boldsymbol{b}^{(2)}\end{bmatrix}_{2\times1}=\boldsymbol{a}_1\boldsymbol{b}^{(1)}+\boldsymbol{a}_2\boldsymbol{b}^{(2)}\end{aligned} \tag{2.34}$$

## 更一般情况

将矩阵$\boldsymbol{A}_{m\times p}$写成一系列左右排列的列向量，即

$$\boldsymbol{A}_{m\times p}=\begin{bmatrix}\boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_p\end{bmatrix}_{1\times p} \tag{2.35}$$

其中：列向量$\boldsymbol{a}_i$元素数量为$m$，即行数为$m$。

将矩阵$\boldsymbol{B}_{p\times n}$写成上下叠放的行向量，即

$$\boldsymbol{B}_{p\times n}=\begin{bmatrix}\boldsymbol{b}^{(1)}\\ \boldsymbol{b}^{(2)}\\ \vdots\\ \boldsymbol{b}^{(p)}\end{bmatrix}_{p\times1} \tag{2.36}$$

其中：行向量$\boldsymbol{b}^{(i)}$元素数量为$n$，即列数为$n$。

矩阵$\boldsymbol{A}$和矩阵$\boldsymbol{B}$相乘，可以展开写成$p$个 $m \times n$矩阵相加，即

$$\boldsymbol{A}_{m\times p} @ \boldsymbol{B}_{p\times n} = \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_p \end{bmatrix}_{1\times p} \begin{bmatrix} \boldsymbol{b}^{(1)} \\ \boldsymbol{b}^{(2)} \\ \vdots \\ \boldsymbol{b}^{(p)} \end{bmatrix}_{p\times 1} = \underbrace{\boldsymbol{a}_1\boldsymbol{b}^{(1)} + \boldsymbol{a}_2\boldsymbol{b}^{(2)} + \cdots \boldsymbol{a}_p\boldsymbol{b}^{(p)}}_{p \text{ matrices with shape of } m\times n} = \boldsymbol{C}_{m\times n} \tag{2.37}$$

我们可以把$\boldsymbol{a}_k\boldsymbol{b}^{(k)}$的结果矩阵写成$\boldsymbol{C}_k$，这样$\boldsymbol{A}$和$\boldsymbol{B}$的乘积$\boldsymbol{C}$可以写成$\boldsymbol{C}_k$ $(k = 1, 2, \cdots, p)$ 之和，即

$$\boldsymbol{a}_1\boldsymbol{b}^{(1)} + \boldsymbol{a}_2\boldsymbol{b}^{(2)} + \cdots \boldsymbol{a}_p\boldsymbol{b}^{(p)} = \boldsymbol{C}_1 + \boldsymbol{C}_2 + \cdots + \boldsymbol{C}_p = \boldsymbol{C} \tag{2.38}$$

在这个视角下，矩阵的乘法变成了若干矩阵的叠加。这是一个非常重要的视角，数据科学和机器学习很多算法都离不开它。

## 热图

图2.12所示给出的是图2.11所示矩阵乘法第二视角的热图。图2.12中三个形状相同矩阵$\boldsymbol{C}_1$、$\boldsymbol{C}_2$、$\boldsymbol{C}_3$相加得到$\boldsymbol{C}$。

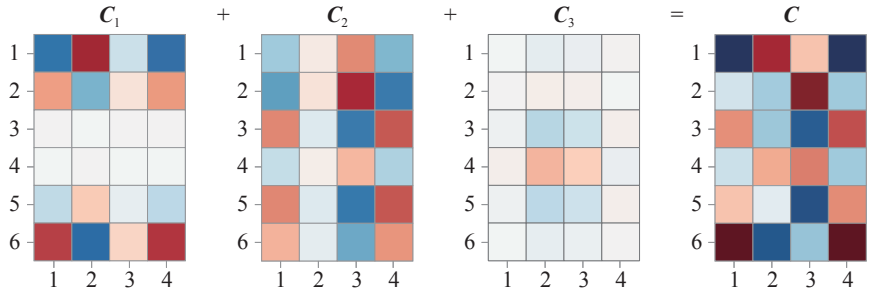



图2.12　矩阵乘法第二视角

如图2.13所示，从图像角度来看， 好比若干形状相同的图片，经过层层叠加，最后获得了一幅完整的热图。式 (2.38) 中的 $p$决定了参与叠加的矩阵层数。矩阵乘法中，$p$所在维度被“消去”，这也相当于一种“压缩”。

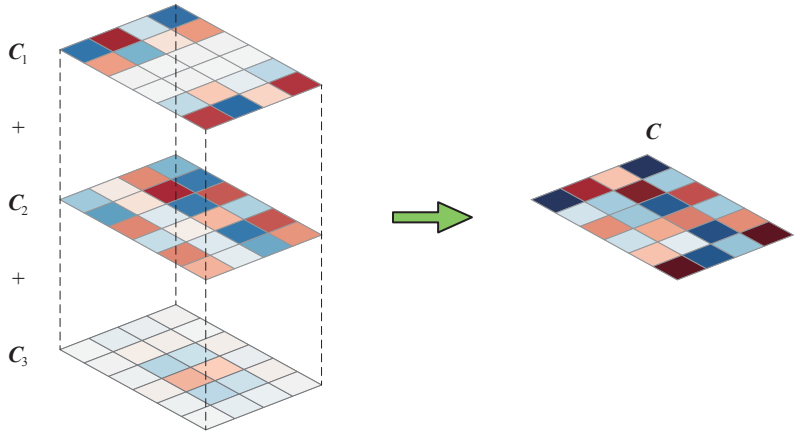



图2.13　三幅图像叠加得到矩阵$\boldsymbol{C}$热图

图2.14、图2.15、图2.16所示分别展示了如何获得图2.12中矩阵$\boldsymbol{C}_1$、$\boldsymbol{C}_2$、$\boldsymbol{C}_3$的热图。

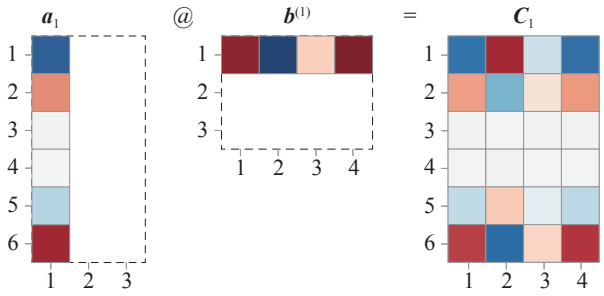



图2.14　获得$\boldsymbol{C}_1$

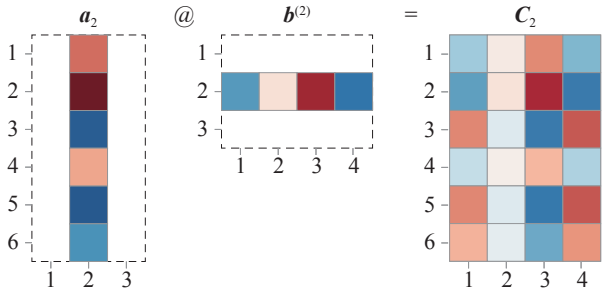



图2.15　获得$\boldsymbol{C}_2$

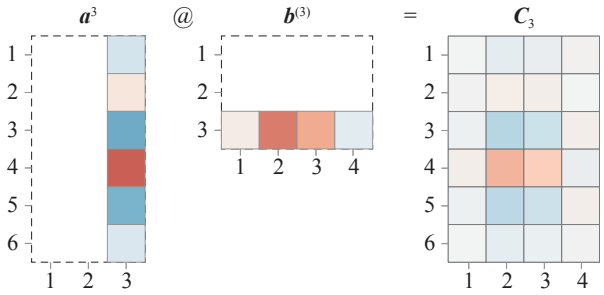



图2.16　获得$\boldsymbol{C}_3$

观察热图可以发现一个有意思的现象，列向量乘行向量好像张起了一幅平面。张量积用的就是类似于图2.14、图2.15、图2.16的运算思路。

鸢尾花书《矩阵力量》一册会讲解张量积。

代码文件Bk3_Ch2_10.py中Bk3_Ch2_10_B部分为绘制图2.12。

**主成分分析** (Principal Component Analysis, PCA) 是机器学习中重要的降维算法。这种算法可以把可能存在线性相关的数据转化成线性不相关的数据，并提取数据中的主要特征。

图2.17中，$\boldsymbol{X}$为原始数据，$\boldsymbol{X}_1$、$\boldsymbol{X}_2$、$\boldsymbol{X}_3$分别为第一、第二、第三主成分。根据热图颜色色差可以看出：第一主成分解释了原始数据中最大的差异；第二成分则进一步解释剩余数据中最大的差异，以此类推。

图2.17实际上就是本节介绍的矩阵乘法第二视角。鸢尾花书会在《矩阵力量》《统计至简》《数据有道》三本书中以不同视角介绍主成分分析。

《矩阵力量》将从矩阵分解、空间、优化等视角讲解PCA；《统计至简》将从数据、中心化数据、Z分数、协方差矩阵、相关性系数矩阵这些统计视角来讨论PCA不同技术路线之间的差异；《数据有道》则侧重讲解如何在实践中使用PCA分析数据，并使用PCA结果进行回归分析。

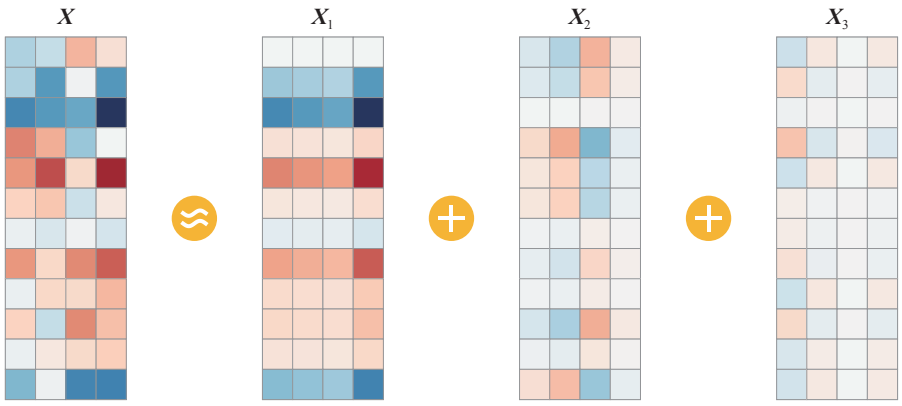



图2.17　用数据热图叠加看主成分分析

## 2.6 矩阵除法：计算逆矩阵

实际上，并不存在所谓的矩阵除法。所谓矩阵$\boldsymbol{B}$除以矩阵$\boldsymbol{A}$，实际上是将矩阵$\boldsymbol{A}$先转化逆矩阵$\boldsymbol{A}^{-1}$，然后计算$\boldsymbol{B}$和逆矩阵$\boldsymbol{A}^{-1}$乘积，即

$$\boldsymbol{B}\boldsymbol{A}^{-1} = \boldsymbol{B} @ \boldsymbol{A}^{-1} \tag{2.39}$$

$\boldsymbol{A}$如果**可逆** (invertible)，则仅当$\boldsymbol{A}$为方阵且存在矩阵$\boldsymbol{A}^{-1}$使得下式成立

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I} \tag{2.40}$$

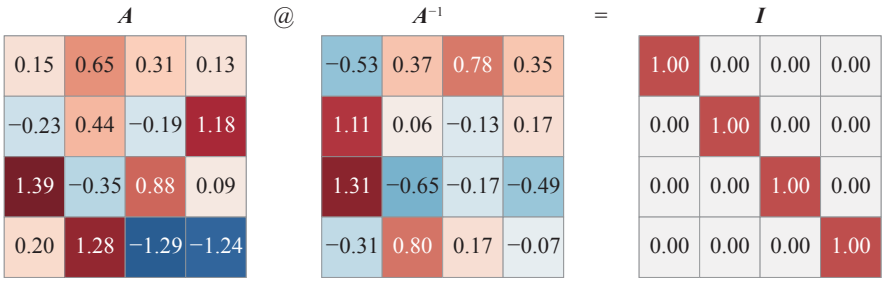
$\boldsymbol{A}^{-1}$叫作**矩阵$\boldsymbol{A}$的逆矩阵** (the inverse of matrix $\boldsymbol{A}$)。

式 (2.40) 中的$\boldsymbol{I}$就是前文介绍过的**单位矩阵** (identity matrix)。**$n$阶单位矩阵** ($n$-square identity matrix或$n$-square unit matrix) 的特点是对角线上的元素为1，其他为0，即

$$\boldsymbol{I}_{n\times n} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{2.41}$$

我们可以用numpy.linalg.inv() 计算方阵的逆。

图2.18所示为方阵$\boldsymbol{A}$和逆矩阵$\boldsymbol{A}^{-1}$相乘得到单位矩阵$\boldsymbol{I}$的热图。




注意图中数值仅保留小数点后两位，按图中数值相乘不能准确得到单位矩阵。

图2.18　方阵$\boldsymbol{A}$和逆矩阵$\boldsymbol{A}^{-1}$相乘

一般情况，有

$$\left(\boldsymbol{A}+\boldsymbol{B}\right)^{-1} \neq \boldsymbol{A}^{-1}+\boldsymbol{B}^{-1} \tag{2.42}$$

请大家注意以下和矩阵逆有关的运算规则：

$$\begin{aligned}
&\left(\boldsymbol{A}^{\mathrm{T}}\right)^{-1}=\left(\boldsymbol{A}^{-1}\right)^{\mathrm{T}} \\
&\left(\boldsymbol{AB}\right)^{-1}=\boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \\
&\left(\boldsymbol{ABC}\right)^{-1}=\boldsymbol{C}^{-1}\boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \\
&\left(k\boldsymbol{A}\right)^{-1}=\frac{1}{k}\boldsymbol{A}^{-1}
\end{aligned} \tag{2.43}$$

其中：假设$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$、$\boldsymbol{AB}$和$\boldsymbol{ABC}$逆运算存在，且$k$不等于0。

表2.4总结常见矩阵逆相关的英文表达。

表2.4　和矩阵逆相关的英文表达

| 数学表达 | 英文表达 |
|---|---|
| $\boldsymbol{A}^{-1}$ | Inverse of the matrix boldface capital ***A***.<br>Matrix boldface capital ***A*** inverse. |
| $(\boldsymbol{A}+\boldsymbol{B})^{-1}$ | Left parenthesis bold face capital ***A*** plus boldface capital ***B*** right parenthesis superscript minus one.<br>Inverse of the matrix sum boldface capital ***A*** plus boldface capital ***B***. |
| $(\boldsymbol{AB})^{-1}$ | Left parenthesis boldface capital ***A*** times boldface capital ***B*** right parenthesis superscript minus one.<br>Inverse of the matrix product boldface capital ***A*** and boldface capital ***B***. |
| $\boldsymbol{ABC}^{-1}$ | The product boldface capital ***A*** boldface capital ***B*** and boldface capital ***C*** inverse. |

Bk3_Ch2_11.py计算并绘制图2.18。

如果学完这一章，大家对矩阵乘法规则还是一头雾水，我只有一个建议——死记硬背！

先别问为什么。就像背诵九九乘法口诀表一样，把矩阵乘法规则背下来。此外，再次强调矩阵乘法等运算不是“奇技淫巧”。后面，大家会逐步意识到矩阵乘法的洪荒伟力。

# 03 Geometry 几何

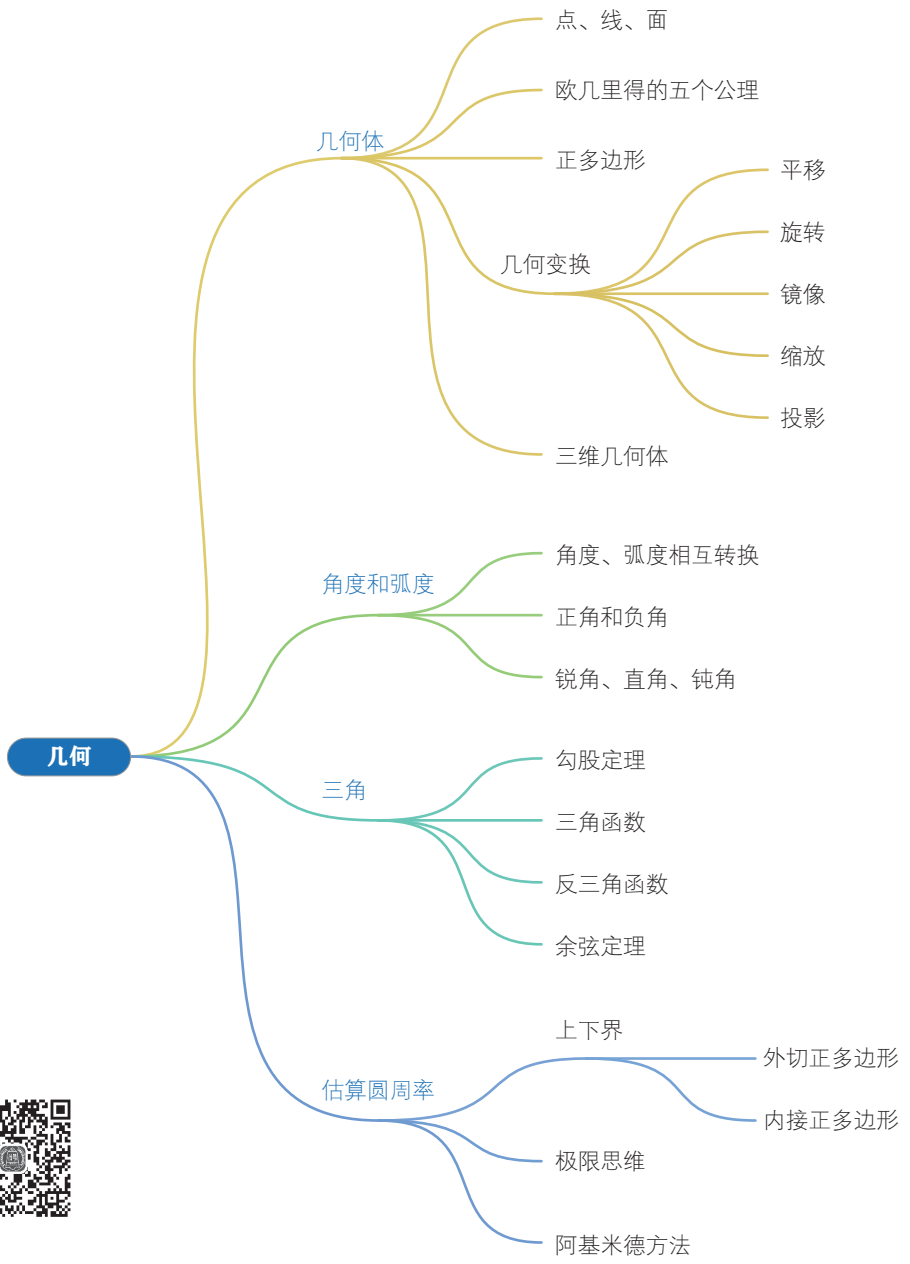
音乐之美由耳朵来感受，几何之美让眼睛去欣赏

不懂几何，勿入斯门。

***Let no one destitute of geometry enter my doors.***

—— 柏拉图 (Plato) | 古希腊哲学家 | 424/423B.C.— 348/347 B.C.

- `ax.add_patch()` 绘制图形
- `math.degrees()` 将弧度转换为角度
- `math.radians()` 将角度转换成弧度
- `matplotlib.patches.Circle()` 创建正圆图形
- `matplotlib.patches.RegularPolygon()` 创建正多边形图形
- `numpy.arccos()` 计算反余弦
- `numpy.arcsin()` 计算反正弦
- `numpy.arctan()` 计算反正切
- `numpy.cos()` 计算余弦值
- `numpy.deg2rad()` 将角度转化为弧度
- `numpy.rad2deg()` 将弧度转化为角度
- `numpy.sin()` 计算正弦值
- `numpy.tan()` 计算正切值


几何
几何体
点、线、面
欧几里得的五个公理
正多边形
几何变换
平移
旋转
镜像
缩放
投影
三维几何体
角度和弧度
角度、弧度相互转换
正角和负角
锐角、直角、钝角
三角
勾股定理
三角函数
反三角函数
余弦定理
估算圆周率
上下界
外切正多边形
内接正多边形
极限思维
阿基米德方法

# 3.1 几何缘起：根植大地，求索星空

毫不夸张地说，几何思维印刻在人类基因中。生而为人，时时刻刻看到的、接触到的都是各种各样的几何形体。大家现在不妨停下来看看、摸摸周围环境中的物体，相信你一定会惊奇地发现整个物质世界就是几何的世界。宏观如天体，微观至原子，几何无处不在。正如**约翰内斯·开普勒** (Johannes Kepler, 1571 — 1630) 所言：“但凡有物质的地方，就有几何。”

哪怕在遥远的古代文明，人类活动也离不开几何知识，丈量距离、测绘地形、估算面积、计算体积、营造房屋、设计工具、制作车轮、工艺美术……无处不需要几何这种数学工具。

如图3.1所示，几何滥觞于田间地头。在古埃及，尼罗河每年都要淹没河两岸。当洪水退去，留下的肥沃土壤让河流两岸平原的农作物生长，但是洪水同样冲走了标示不同耕地的界桩。

法老每年都要派大量测量员重新丈量土地。测绘员们用打结的绳子去丈量土地和角度，以便重置这些界桩。计算矩形、三角形农田的面积当然简单。而对于复杂的几何形体，测绘员经常将土地分割成矩形和三角形来估算土地面积。古埃及的几何知识则随着测量精度提高而不断累积精进。

无独有偶，中国古代重要的数学典籍之一《九章算术》的第一章名为“方田”。这一章多数题目以丈量土地为例，讲解如何计算长方形、三角形、梯形、圆形等各式几何形状的面积，如图3.2所示。

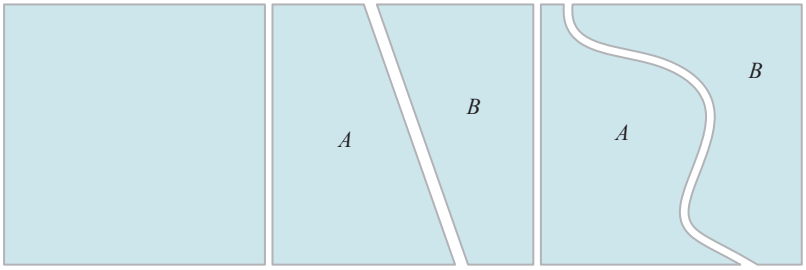



图3.1　各种形状田地地块

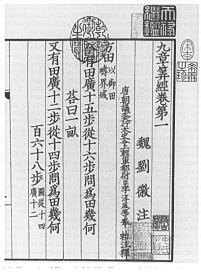



图3.2　《九章算术》第一章开篇

几何学的重大飞跃来自于古希腊。古希腊人创造了几何geometron (英文单词为geometry) 这个词；“geo”在希腊语里是“大地”的意思，“metron”的意思是“测量”。

在古希腊，几何学受到高度重视。几何是博雅教育七艺的重要一门课程。据传说，柏拉图学院门口刻着这句话：“不懂几何者，不许入内。”图3.3所示为古希腊几何发展时间轴上重要的数学家，以及同时代的其他伟大思想家。值得注意的是，中国春秋时代的孔子和苏格拉底、柏拉图、亚里士多德，竟然是同属一个时代。东西方两条历史轴线给人以平行时空的错觉。

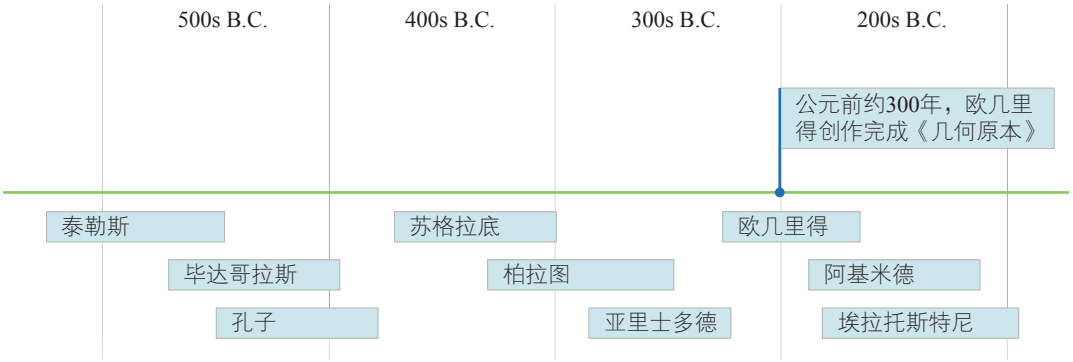



图3.3　古希腊几何发展历史时间轴

**欧几里得 (Euclid)**
**古希腊数学家 | 约公元前330 — 公元前275**
**被称为几何之父，他的《几何原本》堪称西方现代数学的开山之作**

古希腊数学家中关键人物是**欧几里得** (Euclid)，他的巨著**《几何原本》**(The *Elements*) 首次尝试将几何归纳成一个系统。

不夸张地说，欧几里得《几何原本》是整个人类历史上最成功、影响最深刻的数学教科书，没有之一。《几何原本》不是习题集，它引入严谨的推理，使得数学变得体系化。

古希腊的几何学发展要远远领先于其他数学门类，可以说古希腊的算术和代数知识也都是建立在几何学基础之上。而代数的大发展要归功于一位波斯数学家——花拉子密，这是下一章要介绍的人物。

中文“几何”一词源自于《几何原本》的翻译，如图3.4所示。1607年，明末科学家徐光启和意大利传教士**利玛窦** (Matteo Ricci) 共同翻译完成了《几何原本》前六章，如图3.5所示。

他们确定了包括“几何”“点”“直线”“角”等大量中文译名。“几何”一词的翻译特别精妙，发音取自geo，而“几何”二字的中文又有“大小如何”的含义。《九章算术》几乎所有的题目都以“几何”这一提问结束，比如：“问：为田几何？”

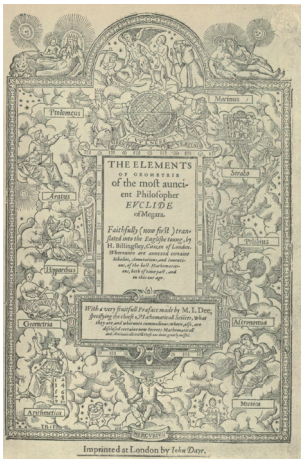



图3.4 《几何原本》1570年首次被翻译为英文版

图3.5 《中国图说》(*China Illustrata*) 中插图描绘利玛窦和徐光启

在估算圆周率的竞赛中，**阿基米德** (Archimedes) 写下了浓墨重彩的一笔。如图3.6所示，阿基米德利用圆内接正多边形和圆外切正多边形，估算圆周率在223/71和22/7之间，即3.140845和3.142857之间。

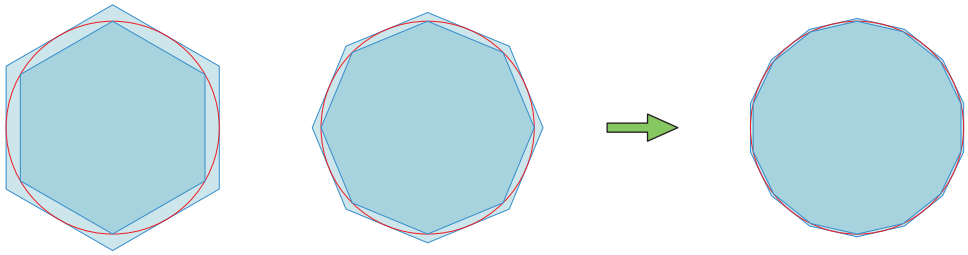
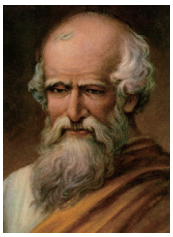


图3.6 圆形内接和外切正四、正八、正十六边形



阿基米德 (Archimedes)
古希腊数学家、物理学家 | 公元前287 — 公元前212
常被称作力学之父，估算圆周率

公元前212年，阿基米德的家乡被罗马军队攻陷时，他还在潜心研究几何问题。罗马士兵闯入他的家，阿基米德大声训斥这些不速之客，“别弄乱我的圆”。但是，罗马士兵还是踩坏了画在沙盘上的几何图形，并杀死了阿基米德。

几何学有纬地经天之功。比如，利用相似三角形原理，古希腊数学家**埃拉托斯特尼** (Eratosthenes) 估算出了地球直径：正午时分，在点$A$ (阿斯旺) 太阳光垂直射入深井中，井底可见太阳倒影。此时，在点$B$ (亚历山大港)，埃拉托斯特尼找人测量了一个石塔影子的长度。利用石塔的高度和影子的长度，埃拉托斯特尼计算得到图3.7中所示的$\theta = 7°$，也就是$A$和$B$两点的距离为整个地球圆周的7/360。埃拉托斯特尼恰好知道$AB$距离，从而估算出了地球的周长，进而计算得到地球周长在39,690 km到46,620 km之间，误差约为2%。

**托勒密** (Claudius Ptolemy) 在约150年创作出了《天文学大成》 (*Almagest*)。这本书可以说是代表了古希腊天文学的最高水平，它也是古希腊几何思维在天文学领域的结晶。托勒密总结前人成果，在书中明确提出**地心说** (geocentric model)——地球位于宇宙中心，固定不动，星体绕地球运动，如图3.8所示。此外，《天文学大成》中给出了人类历史上第一个系统建立的三角函数表。

托勒密 (Claudius Ptolemy)
希腊数学家、天文学 | 公元前100 — 公元前170
创作《天文学大成》，系统提出地心说

然而，托勒密的地心说被宗教思想奉为圭臬，牢牢禁锢人类长达一千两百多年，直到**哥白尼** (Nicolaus Copernicus, 1473 — 1543) 唤醒人类沉睡的思想世界。正是利用古希腊发展的圆锥曲线知识，开普勒提出了行星运动的三定律。

圆锥曲线是本书第8、9章要介绍的内容。

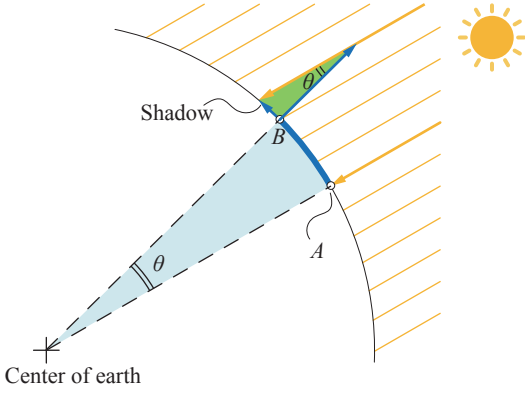



图3.7　埃拉托斯特尼计算地球直径用到的几何知识


图3.8　后人绘制的托勒密地心说模型

# 3.2 点动成线，线动成面，面动成体

点动成线，线动成面，面动成体——相信大家对这句话耳熟能详。点没有维度，线是一维，面是二维，体是三维，如图3.9所示。当然，在数学的世界中，四维乃至多维都是存在的。

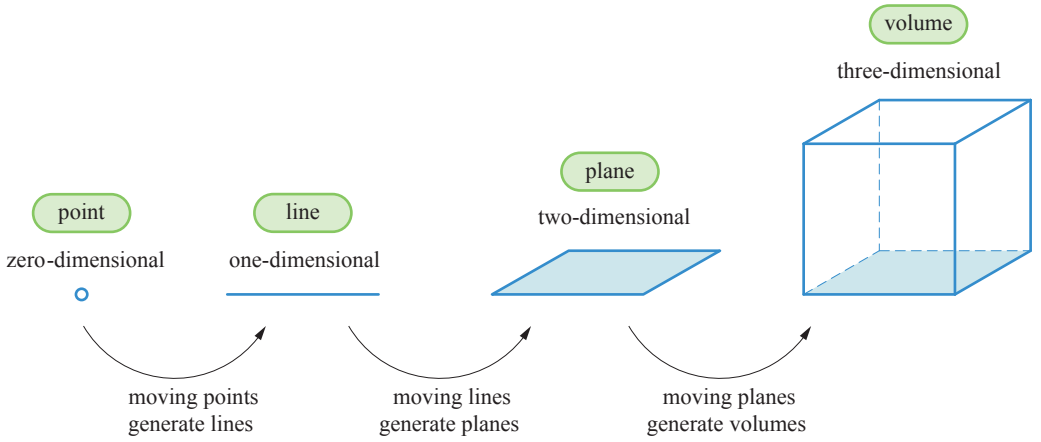



图3.9　点动成线，线动成面，面动成体

## 点

点确定空间的一个位置，点本身没有长度、面积等几何属性。

所有几何图形都离不开点，图3.10所示为常见的几种点——**端点** (endpoint)、**中点** (midpoint)、**起点** (initial point)、**终点** (terminal point)、**圆心** (center)、**切点** (point of tangency)、**顶点** (vertex)、**交点** (point of intersection)。点和点之间的线段长度叫**距离** (distance)。

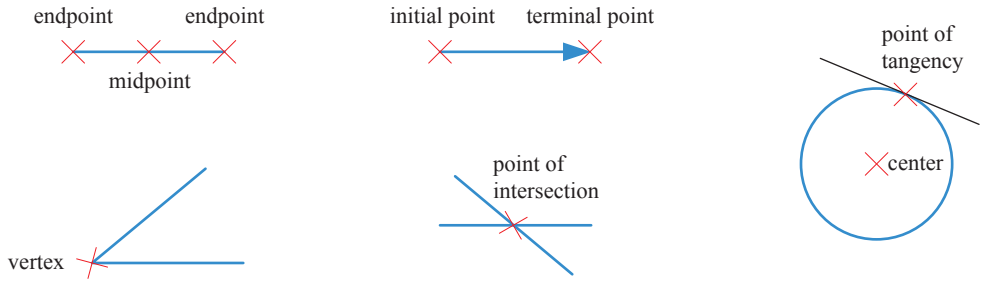



图3.10　几种点

## 线

如图3.11所示，**直线** (line) **沿两个方向无限延伸** (extends in both directions without end)，**没有端点** (has no endpoints)。

**射线** (ray或half-line) 开始于一端点，仅沿一个方向无限延伸。

**线段** (line segment) 有两个**端点** (endpoint)。

**向量** (vector) 则是有方向的线段。

线段具有**长度** (length) 这种几何性质，但是没有面积这种性质。

给定参考系，线又可以分为**水平线** (horizontal line)、**斜线** (oblique line) 和**竖直线** (vertical line)。

图3.11还给出了其他几种线：**边** (edge)、**曲线** (curve或curved line)、**等高线** (contour line)、**法线** (normal line)、**切线** (tangent line)、**割线** (secant line) 等。

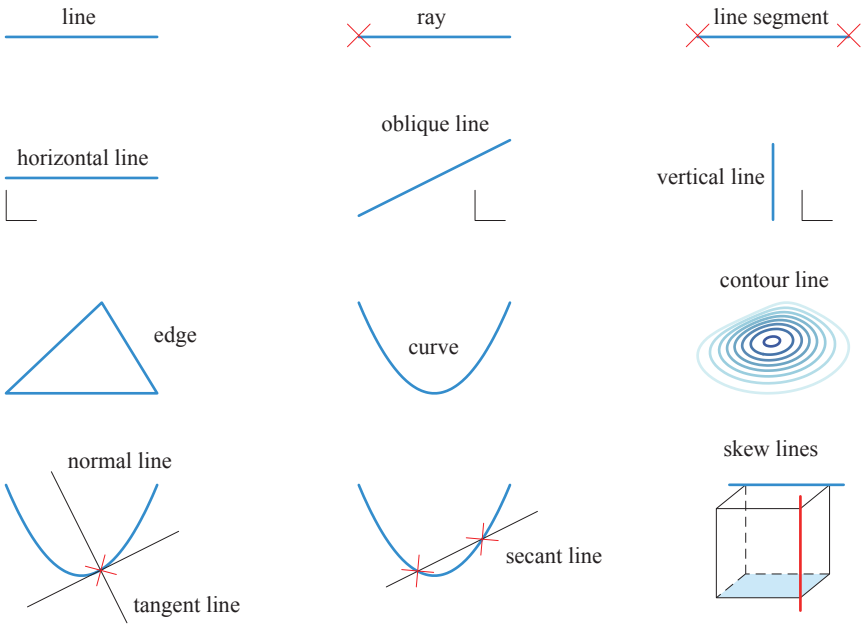



图3.11　几种线

在平面上，线与线之间有四种常见的关系：**平行** (parallel)、**相交** (intersecting)、**垂直** (perpendicular) 和**重合** (coinciding)，如图3.12所示。

两条线平行可以记作 $l_1 /\!/ l_2$ (读作line $l$ sub one is parallel to the line $l$ sub two)。$l_1$与$l_2$相交于点$P$可以读作“line $l$ sub one intersects the line $l$ sub two at point capital $P$”。两条线垂直可以记作 $l_1 \perp l_2$ (读作line $l$ sub one is perpendicular to the line $l$ sub two)。三维空间中，两条直线还可以互为**异面线** (skew line)。

如图3.13所示，可视化时还会用到不同样式的线型，如**实线** (solid line或continuous line)、**粗实线** (heavy solid line或continuous thick line)、**点虚线** (dotted line)、**短画线** (dashed line)、**点画线** (dash-dotted line) 等。

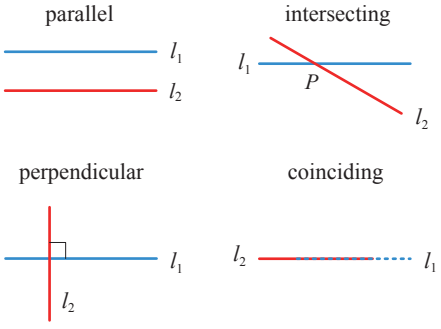



图3.12　平面上两条线的关系

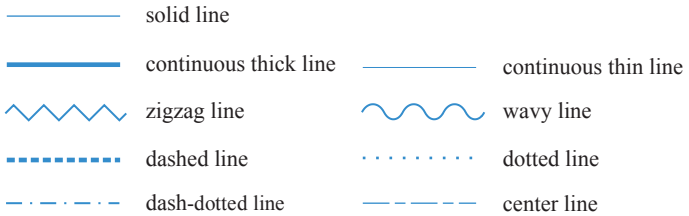



图3.13　几种线的样式

## 欧几里得的五个公理

在《几何原本》中，欧几里得提出以下五个公理，如图3.14所示。

- 任意两点可以画一条直线；
- 任意线段都可以无限延伸成一条直线；
- 给定任意线段，以该线段为半径、一个端点为圆心，可以画一个圆；
- 所有直角都全等；
- 两直线被第三条直线所截，如果同侧两内角之和小于两个直角之和，则两条直线则会在该侧相交。

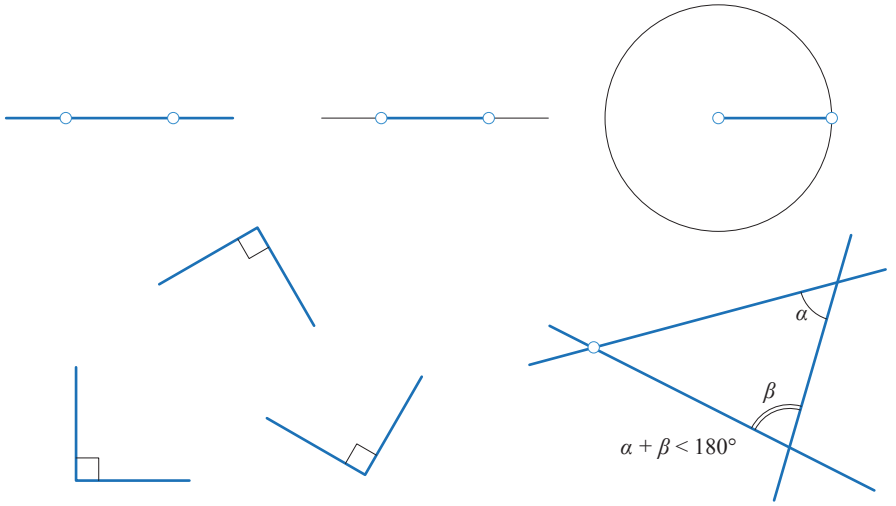
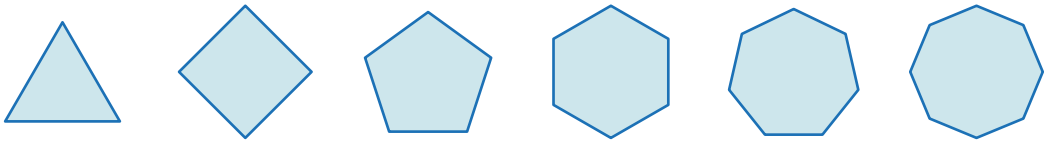



图3.14　欧几里得提出的五个几何公理

以五个公理为基础，欧几里得一步步建立起了几何学大厦。坚持第五条定理，我们在欧几里得几何体系之内。而去掉第五条公理，则进入非欧几里得几何体系。值得一提的是，非欧几里得几何中的黎曼几何为爱因斯坦的广义相对论提供了数学工具。

## 正多边形

**正多边形** (regular polygons) 是边长相等的多边形，正多边形内角相等，如图3.15所示。我们将在圆周率估算中用到正多边形的相关知识。




(a) equilateral triangle；(b) square；(c) pentagon；(d) hexagon；(e) heptagon；(f) octagon

图3.15　六个正多边形

Bk3_Ch3_01.py绘制图3.15中的六个正多边形。

## 三维几何体

图3.16所示为常见三维几何体，它们依次是：**正球形** (sphere)、**圆柱体** (cylinder)、**圆锥** (cone)、**锥台** (cone frustum)、**正方体/正六面体** (cube)、**长方体** (cuboid)、**平行六面体** (parallelepiped)、**四棱台** (square pyramid frustum)、**四棱锥** (square-based pyramid)、**三棱锥** (triangle-based pyramid)、**三棱柱** (triangular prism)、**四面体** (tetrahedron)、**八面体** (octahedron)、**五棱柱** (pentagonal prism)、**六棱柱** (hexagonal prism) 和**五棱锥** (pentagonal pyramid)。

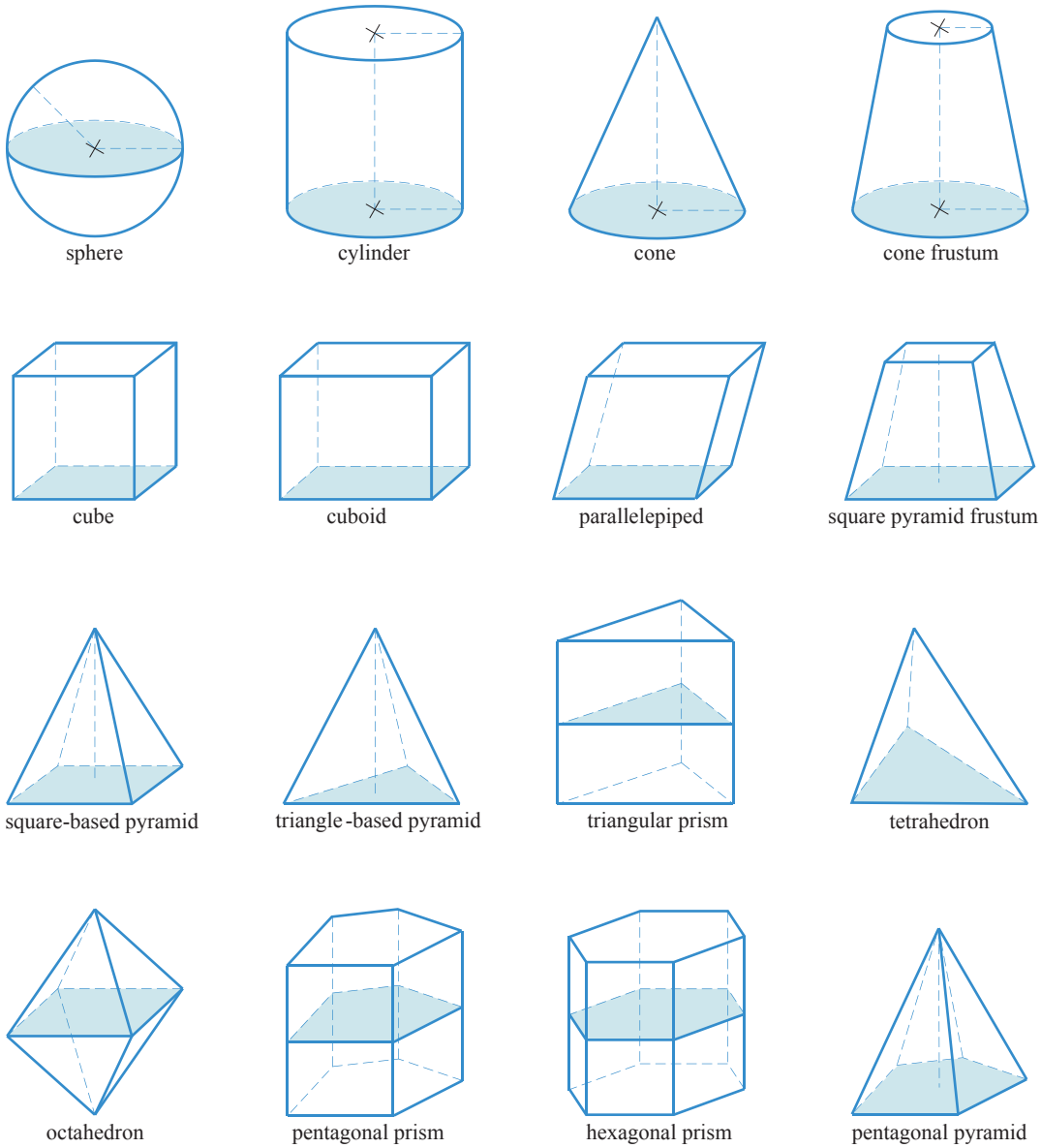



图3.16　常见三维几何体

**柏拉图立体** (Platonic solid)，又称正多面体。图3.17所示为五个正多面体，包括**正四面体** (tetrahedron)、**正方体、正六面体** (cube)、**正八面体** (octahedron)、**正十二面体** (dodecahedron) 和**正二十面体** (icosahedron)。

正多面体的每个面全等，均为**正多边形** (regular polygons)。图3.18所示为五个正多面体展开得到的平面图形。表3.1总结了五个正多面体的结构特征。

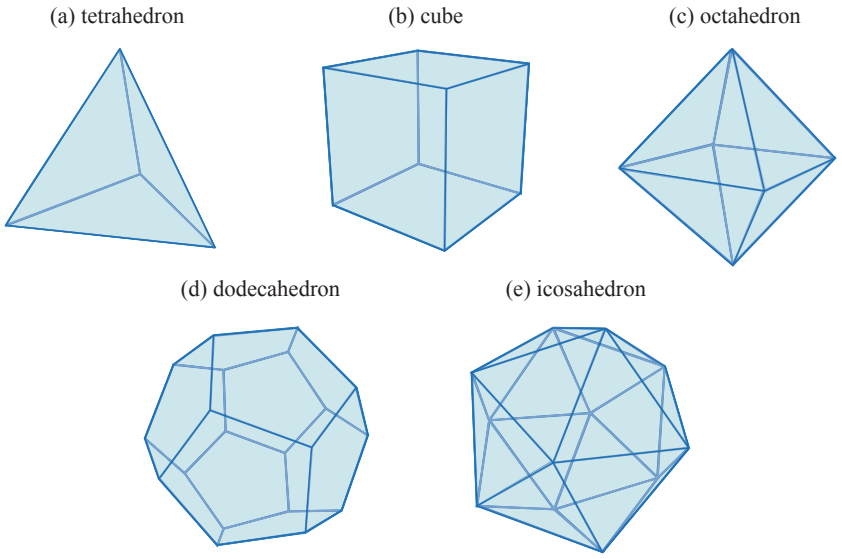



图3.17　五个正多面体

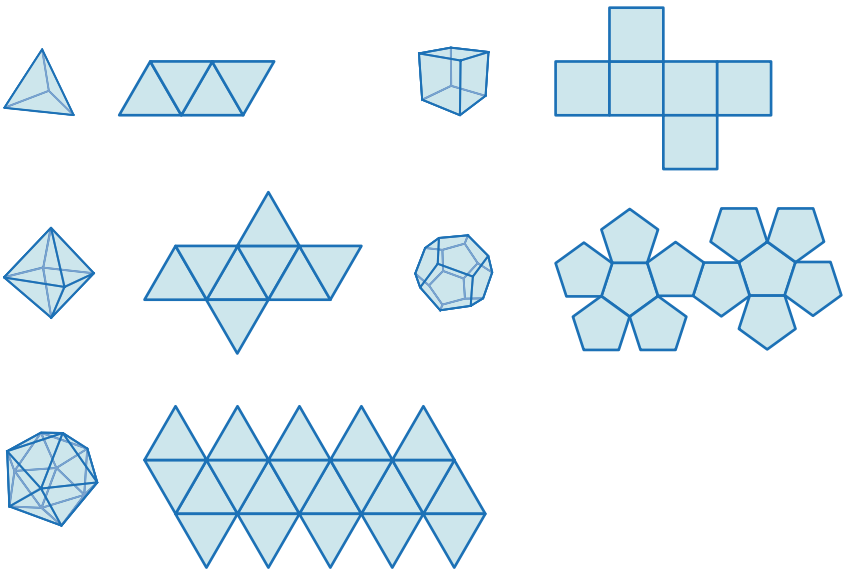


图3.18　五个正多面体展开得到的平面图形

**表3.1　正多面体的特征**

| 正多面体 | 顶点数 | 边数 | 面数 | 面形状 |
|---|---|---|---|---|
| Tetrahedron | 4 | 6 | 4 | Equilateral triangle |
| Cube | 8 | 12 | 6 | Square |
| Octahedron | 6 | 12 | 8 | Equilateral triangle |
| Dodecahedron | 20 | 30 | 12 | Pentagon |
| Icosahedron | 12 | 30 | 20 | Equilateral triangle |