

26

Symbolic Computation in SymPy

SymPy 符号运算

SymPy 是一个 Python 的符号数学计算库



等式仅仅是数学中无聊至极的那部分；我努力从几何角度观察万物。

Equations are just the boring part of mathematics. I attempt to see things in terms of geometry.

—— 斯蒂芬·霍金 (Stephen Hawking) | 英国理论物理学家和宇宙学家 | 1942 ~ 2018



- ◀ `sympy.abc import x` 定义符号变量 `x`
- ◀ `sympy.abc()` 引入符号变量
- ◀ `sympy.collect()` 合并同类项
- ◀ `sympy.cos()` 符号运算中余弦
- ◀ `sympy.diff()` 求解符号导数和偏导解析式
- ◀ `sympy.Eq()` 定义符号等式
- ◀ `sympy.evalf()` 将符号解析式中未知量替换为具体数值
- ◀ `sympy.exp()` 符号自然指数
- ◀ `sympy.expand()` 展开代数式
- ◀ `sympy.factor()` 对代数式进行因式分解
- ◀ `sympy.integrate()` 符号积分
- ◀ `sympy.is_decreasing()` 判断符号函数的单调性
- ◀ `sympy.lambdify()` 将符号表达式转化为函数
- ◀ `sympy.limit()` 求解极限
- ◀ `sympy.Matrix()` 构造符号函数矩阵
- ◀ `sympy.plot_implicit()` 绘制隐函数方程
- ◀ `sympy.plot3d()` 绘制函数的三维曲面
- ◀ `sympy.series()` 求解泰勒展开级数符号式
- ◀ `sympy.simplify()` 简化代数式
- ◀ `sympy.sin()` 符号运算中正弦
- ◀ `sympy.solve()` 求解符号方程组
- ◀ `sympy.solve_linear_system()` 求解含有符号变量的线型方程组
- ◀ `sympy.symbols()` 创建符号变量
- ◀ `sympy.sympify()` 化简符号函数表达式
- ◀ `sympy.utilities.lambdify.lambdify()` 将符号代数式转化为函数



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

26.1 什么是 SymPy?

SymPy 是一个基于 Python 的符号数学库，它可以执行代数运算、解方程、微积分、离散数学以及其他数学操作。与 NumPy、Pandas 等科学计算库不同，SymPy 主要关注的是符号计算而不是数值计算。具体来说，SymPy 可以处理未知变量和数学符号，而不仅仅是数值，这在一些数学研究和工程应用中非常有用。

本章主要介绍 SymPy 中代数、线性代数运算。此外，SymPy 还可以进行微积分运算，比如极限、导数、偏导数、泰勒展开、积分等。这部分内容需要一定的数学分析知识，我们将会在鸢尾花书《数学要素》一册展开讲解。

26.2 代数

因式分解

图 1 所示为利用 SymPy 完成因式分解。

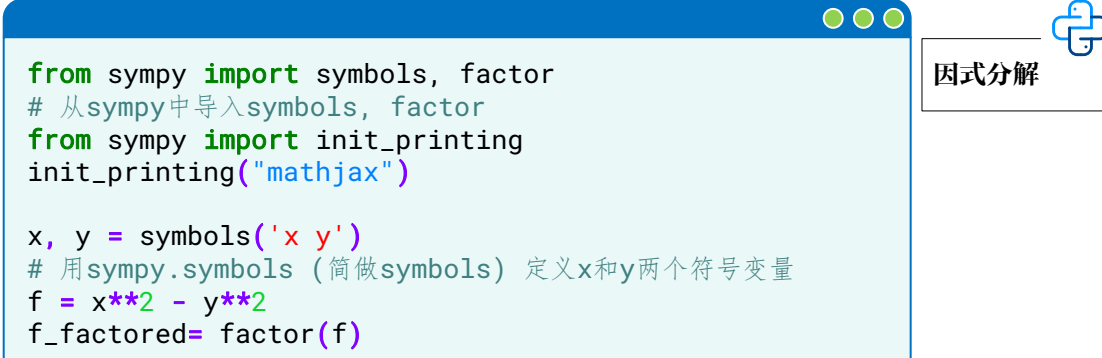
a 从 sympy 导入 symbols 和 factor，其中 symbols 用来定义符号变量，factor 用来完成因式分解。**b** 这两句的作用是将 SymPy 库中的数学符号以美观的形式打印出来。

c 定义了 x 和 y 两个符号变量。symbols 还可以定义带下角标的变量，比如 $x_1, x_2 = \text{symbols}('x_1 x_2')$ 。

也可以用 `from sympy.abc import x, y` 的形式定义符号变量。

此外，用 `sympy.symbols()` 定义变量时还可以提出符号的假设条件。比如，`k = sympy.symbols('k', integer=True)` 这一句定义符号变量 k ，并假定 k 为整数。`z = sympy.symbols('z', real=True)` 定义了符号变量 z ，并假定 z 为实数。

d 定义了 $x^2 - y^2$ 。**e** 对 $x^2 - y^2$ 进行因式分解，结果为 $(x - y)(x + y)$ 。反过来，可以用 `sympy.expand()` 展开 $(x - y)(x + y)$ ，结果为 $x^2 - y^2$ 。



```

a from sympy import symbols, factor
    # 从sympy中导入symbols, factor
b from sympy import init_printing
    init_printing("mathjax")

c x, y = symbols('x y')
    # 用sympy.symbols (简做symbols) 定义x和y两个符号变量
d f = x**2 - y**2
e f_factored= factor(f)
  
```

因式分解

图 1. 因式分解

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

替换

图 2 中 **a** 定义字符串，**b** 将字符串转化为符号表达式 $x^3 + x^2 + x + 1$ 。

c 用符号 y 替代符号 x ，符号表达式变为 $y^3 + y^2 + y + 1$ 。

d 用 0 替代 x ，结果为 1。

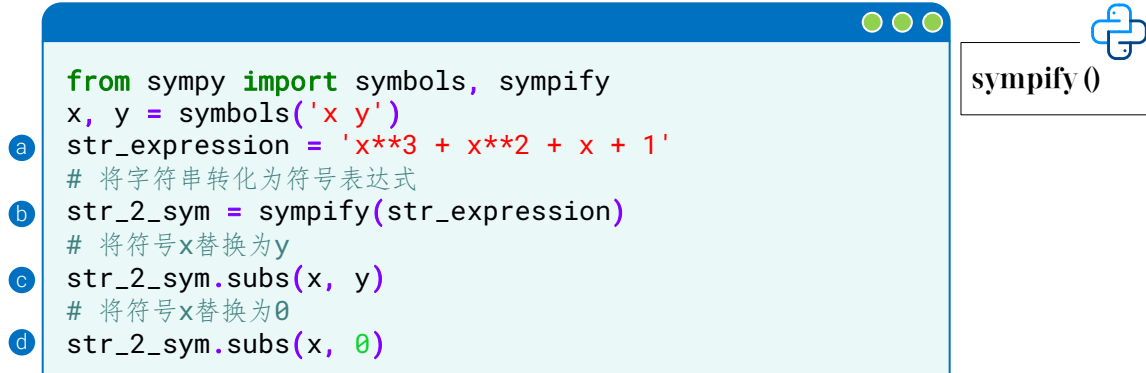


图 2. 用 sympy.sympify 将字符串转化为符号表达式

特殊符号数值

SymPy 还可以定义特殊符号数值，表 1 给出几个例子。比如，`sympy.sympify()` 将 2 转化为符号数值 2，然后进一步判断其是否为整数，是否为实数。再比如，`from sympy import Rational; Rational(1, 2)` 这两句的结果为 $\frac{1}{2}$ 。想要知道表格中结果的浮点数形式，可以用 `.evalf()`，比如 `exp(2).evalf()` 的结果为 7.38905609893065。

表 1. 用 sympy 定义特殊符号数值

代码	结果
<pre>from sympy import sympify sympify(2).is_integer sympify(2).is_real</pre>	<p>True</p> <p>True</p>
<pre>from sympy import Rational Rational(1, 2)</pre>	$\frac{1}{2}$
<pre>from sympy import sqrt 1 / (sqrt(2) + 1)</pre>	$\frac{1}{1+\sqrt{2}}$
<pre>from sympy import pi expr = pi ** 2</pre>	π^2
<pre>from sympy import exp exp(2)</pre>	e^2
<pre>from sympy import factorial factorial(5)</pre>	5!
<pre>from sympy import binomial binomial(5, 4)</pre>	$C_5^4 = 5$
<pre>from sympy import gamma gamma(5)</pre>	$\Gamma(5) = (5-1)! = 4 \times 3 \times 2 \times 1 = 24$

区间

表 2 总结如何用 `sympy.Interval()` 定义各种区间，默认区间左闭、右闭。oo (两个小写英文字母 o) 代表正无穷。注意，大家自己在同一个 Jupyter Notebook 练习时，`from sympy import Interval`, oo 只需要导入一次，不需要重复导入。

此外，用 `sympy.Interval()` 定义的区间还可以进行集合运算，比如 `Interval(0, 2) - Interval(0, 1)` 结果为 `(1, 2]`。再比如，`Interval(0, 1) + Interval(1, 2)` 的结果为 `[0, 2]`。

利用 `.has()` 还可以判断区间是否包含具体元素，比如先定义 `intvl = Interval.Lopen(0, 1)`，得到区间 `(0, 1]`。然后利用 `intvl.has(0)` 或 `intvl.contains(0)` 判断左开右闭区间是否包括元素 0，结果为 `False`。

表 2. 用 `sympy.Interval()` 定义区间

代码	结果
<code>from sympy import Interval, oo Interval(0, 1, left_open=False, right_open=False)</code>	<code>[0, 1]</code>
<code>from sympy import Interval, oo Interval(0, 1, left_open=True, right_open=True)</code>	<code>(0, 0)</code>
<code>from sympy import Interval, oo Interval(0, 1, left_open=False, right_open=True) # Interval.Ropen(0, 1)</code>	<code>[0, 1)</code>
<code>from sympy import Interval, oo Interval(0, 1, left_open=True, right_open=False) # Interval.Lopen(0, 1)</code>	<code>(0, 1]</code>
<code>from sympy import Interval, oo Interval(0, oo, left_open=False, right_open=True)</code>	<code>[0, ∞)</code>
<code>from sympy import Interval, oo Interval(-oo, 0, left_open=True, right_open=True)</code>	<code>(-∞, 0)</code>
<code>from sympy import Interval, S Interval(0, 1).complement(S.Reals)</code>	<code>(-∞, 0) ∪ (1, ∞)</code>

求解等式

图 3 代码介绍如何用 `sympy.solve()` 求解等式。a 定义等式 $x^2 = 1$ ，b 求解等式结果为 $[-1, 1]$ 。

c 定义等式 $ax^2 + bx + c = 0$ 。d 求解等式结果为 $\left[\frac{-b - \sqrt{b^2 - 4ac}}{2a}, \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right]$ 。

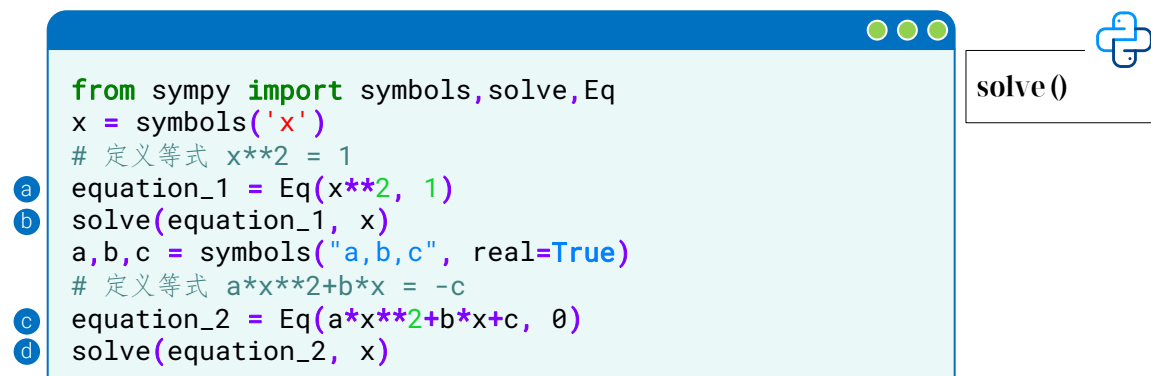


图 3. 用 `sympy.solve()` 求解等式

函数

图 4 代码用 `sympy.lambdify()` 将符号函数 $\exp(-x_1^2 - x_2^2)$ 转化为 Python 函数，从而可以进行数值运算。图 5 所示为代码绘制的二元高斯函数曲面。

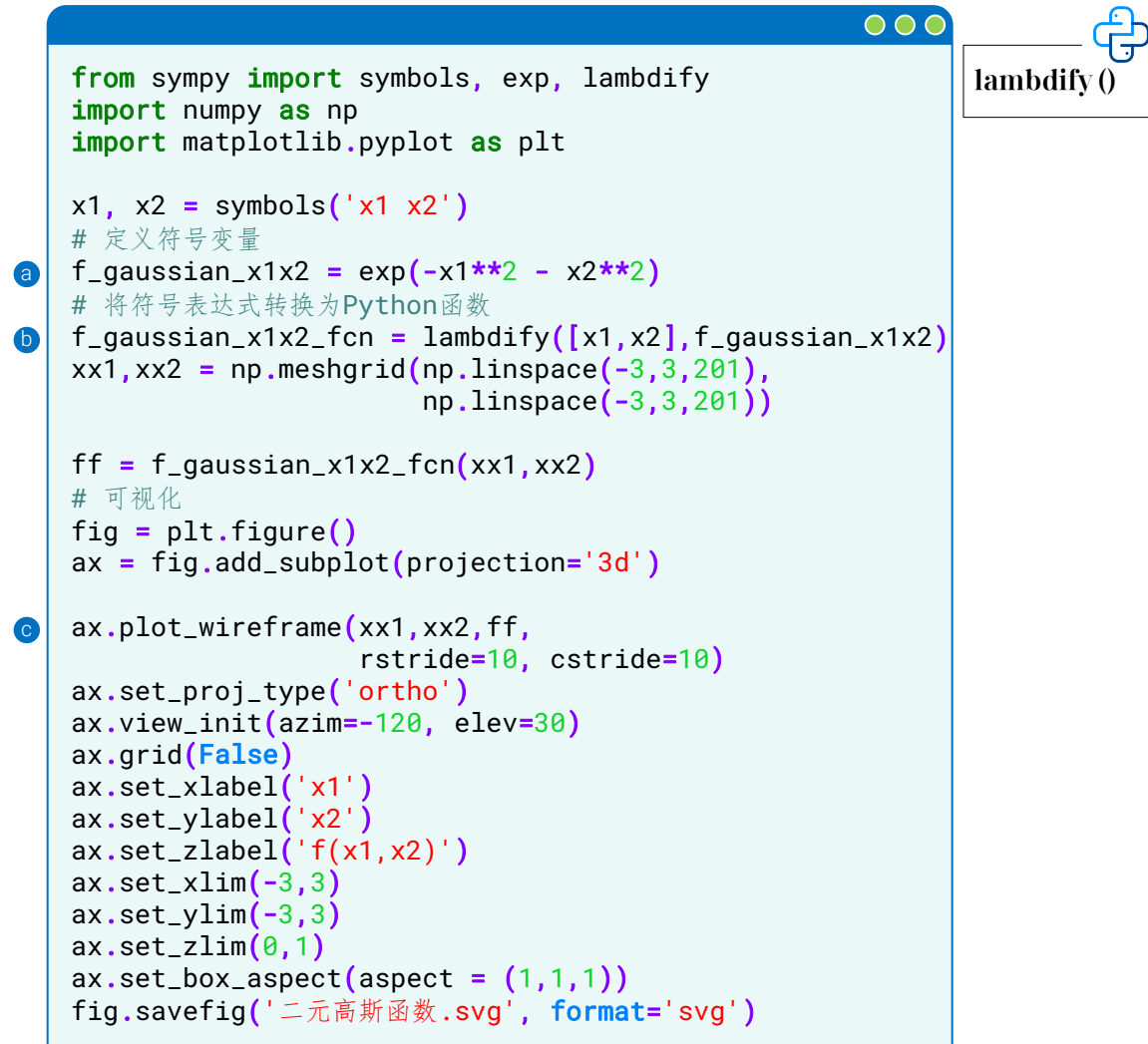


图 4. 用 `sympy.lambdify()` 将符号表达式转化为 Python 函数

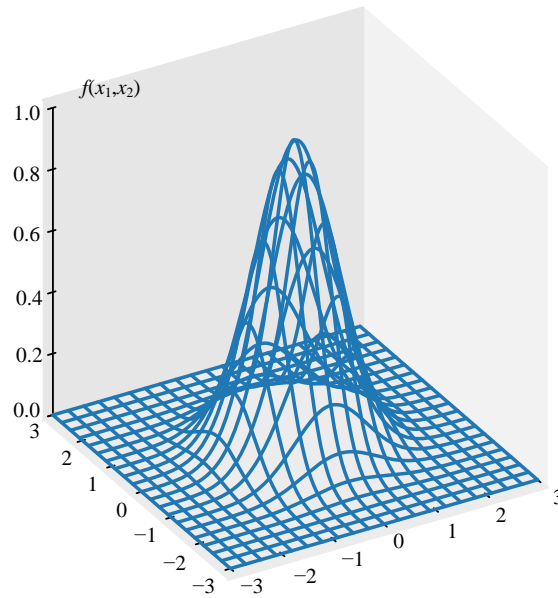


图 5. 二元高斯函数曲面

26.3 线性代数

NumPy 是 Python 科学计算中非常重要的一个库，它提供了快速、高效的多维数组对象及其操作方法，是众多其他科学计算库的基础。

矩阵

图 6 中代码用 `sympy.Matrix()` 定义矩阵、列向量。

a 从 `sympy` 导入 `Matrix` 函数。

b 定义 2 行、3 列矩阵 `A`。函数 `sympy.shape()` 可以用来获取矩阵形状。举个例子，先用 `from sympy import shape` 导入 `shape`，然后 `shape(A)` 返回元组 (2,3) 即矩阵形状。`A.T` 可以完成矩阵转置。对矩阵 `A` 的索引和切片方法和 NumPy 数组一致。比如，`A[0,0]` 提取矩阵第 1 行、第 1 列元素。`A[-1,-1]` 提取矩阵最后一行、最后一列元素。`A[0,:]` 提取矩阵第一行，`A.row(0)` 也可以用来提取矩阵第 1 行。`A[:,0]` 提取矩阵第一列，`A.col(0)` 也可以提取矩阵第一列。

此外，`A.row_del(0)` 可以用来删除第 1 行元素。`A.row_insert()` 可以用来在特定位置插入行向量。类似地，`A.col_del(0)` 可以用来删除第 1 列元素。`A.col_insert()` 可以用来在特定位置插入列向量。

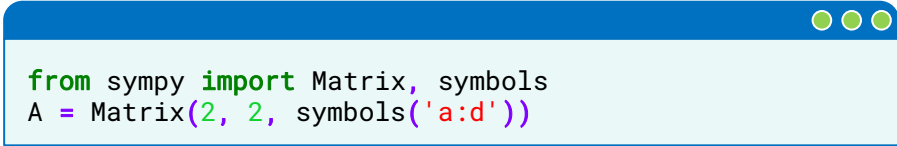
c 定义列向量 `a`。

```
a from sympy import Matrix
    # 定义矩阵
b A = Matrix([[1, 2, 3], [3, 2, 1]])
    # 定义列向量
c a = Matrix([1, 2, 3])
```

`sympy.
matrix()`

图 6. 用 `sympy.matrix()` 定义矩阵

图 7 定义的矩阵 A 为 $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 。



```
from sympy import Matrix, symbols
A = Matrix(2, 2, symbols('a:d'))
```

图 7. 用 `sympy.matrix()` 定义全符号矩阵

表 3 给出了几种产生特殊矩阵的方法。此外，`A.is_symmetric()` 判断矩阵 A 是否为对称阵，`A.is_diagonal()` 判断矩阵 A 是否为对角阵，`A.is_lower` 判断矩阵 A 是否为下三角，`A.is_upper` 判断矩阵 A 是否为上三角，`A.is_square` 判断矩阵 A 是否为方阵，`A.is_zero_matrix` 判断矩阵 A 是否为全 0 矩阵，`A.is_diagonalizable()` 判断矩阵 A 是否可以对角化，`A.is_positive_definite` 判断矩阵 A 是否为正定。

表 3. 用 `sympy` 函数产生特殊矩阵

矩阵类型	代码	结果
单位矩阵	<pre>from sympy import eye A = eye(3)</pre>	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
全 0 矩阵	<pre>from sympy import zeros A = zeros(3, 3)</pre>	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
全 1 矩阵	<pre>from sympy import ones A = ones(3, 3)</pre>	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
对角方阵	<pre>from sympy import diag A = diag(1, 2, 3)</pre>	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$
上三角矩阵	<pre>from sympy import ones A = ones(3) A.upper_triangular()</pre>	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
下三角矩阵	<pre>from sympy import ones A = ones(3) A.lower_triangular()</pre>	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$

运算

图 8 代码给出矩阵相关的常用运算。

- a** 和 **b** 给出两种矩阵乘法运算符，建议大家使用 `@`，和 NumPy 矩阵乘法符号保持一致。
- c** 和 **d** 给出两种矩阵逆运算符。
- e** 将符号矩阵转化为浮点数 NumPy 数组。
- f** 计算矩阵 Q 的逆，结果为 $\frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$ 。
- g** 计算矩阵 Q 的行列式，结果为 $ad-bc$ 。
- h** 计算矩阵 Q 的迹，结果为 $a+d$ 。

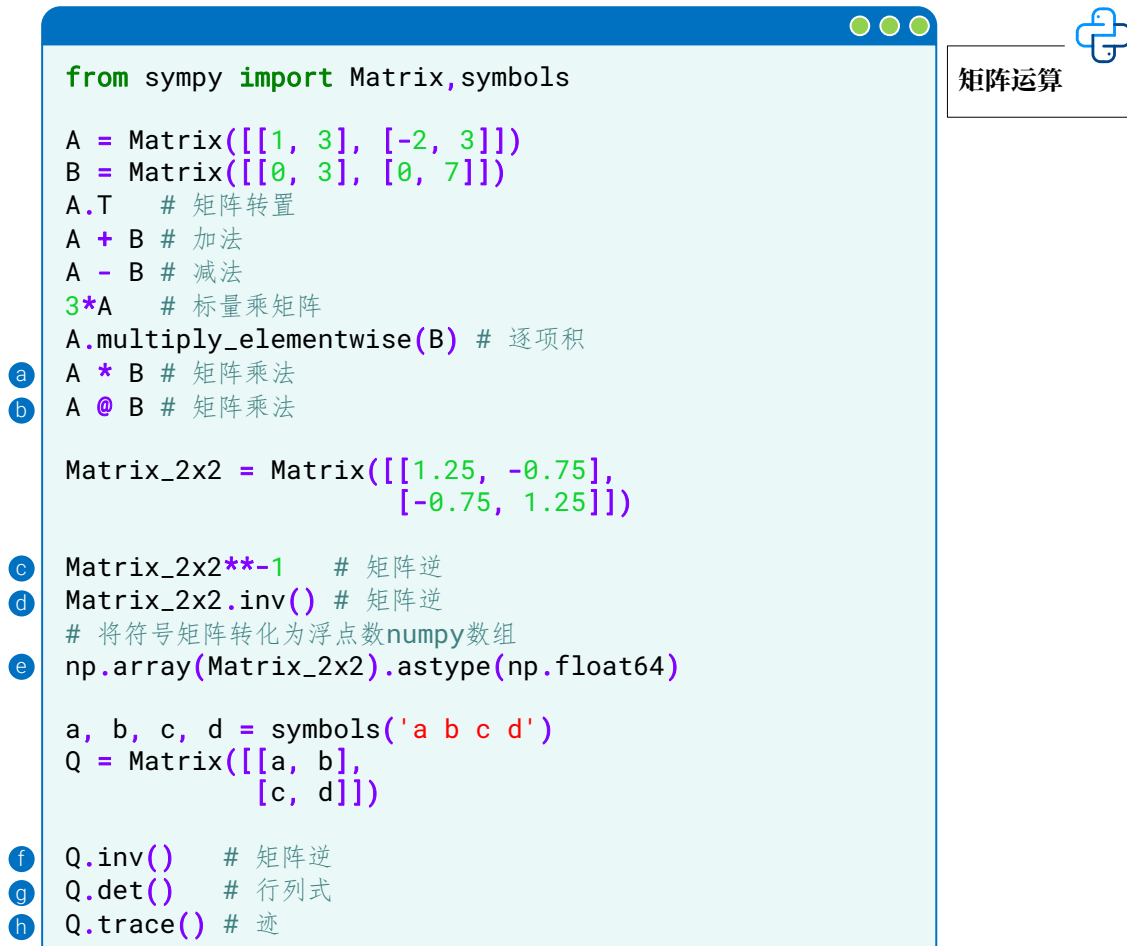


图 8. 用 sympy 中常见矩阵运算

正定性

正定性是线性代数、优化方法、机器学习重要的数学概念。下面我们用一个 2×2 矩阵 $A_{2 \times 2}$ 介绍正定性。

矩阵 $A_{2 \times 2}$ 是正定，意味着 $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 是个开口朝上的抛物面，形状像是碗。除了 $(0, 0)$ ， $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 均大于 0。 $(0, 0)$ 为最小值，图中箭头都背离 $(0, 0)$ 。

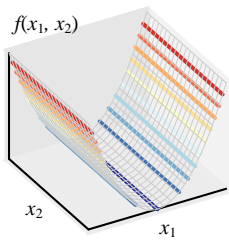
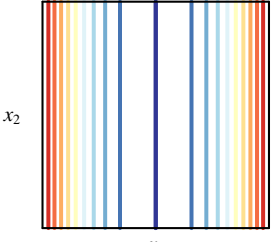
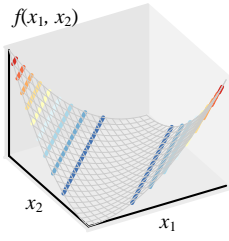
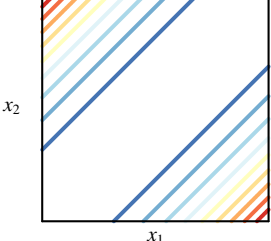
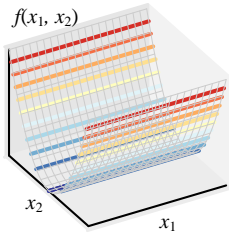
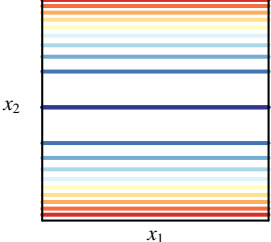
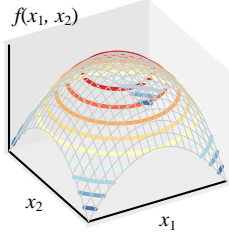
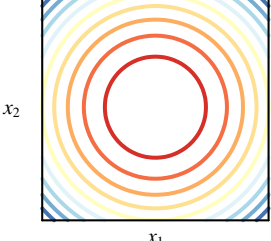
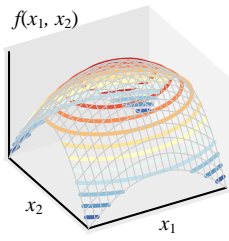
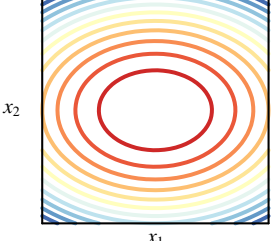
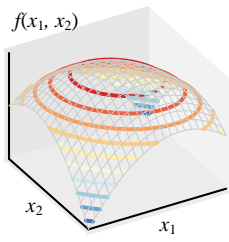
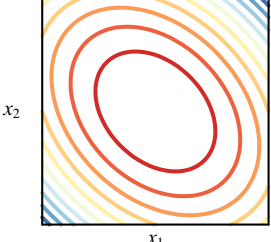
矩阵 $A_{2 \times 2}$ 是半正定，意味着 $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 是个开口朝上的山谷面。除了 $(0, 0)$ ， $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 均大于等于 0。山谷的谷底都是极小值，图中箭头都背离谷底所在直线。

矩阵 $A_{2 \times 2}$ 是负定，意味着 $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 是个开口朝下的抛物面。除了 $(0, 0)$ ， $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 均小于 0。 $(0, 0)$ 为最大值，图中箭头都指向 $(0, 0)$ 。

矩阵 $A_{2 \times 2}$ 是半负定，意味着 $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 是个开口朝下的山脊面。除了 $(0, 0)$ ， $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 均小于等于 0。山脊的顶端都是极大值，图中箭头指向山脊顶端所在直线。

矩阵 $A_{2 \times 2}$ 不定，意味着 $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 是个马鞍面， $(0, 0)$ 为鞍点。 $f(\mathbf{x}) = \mathbf{x}^T @ A_{2 \times 2} @ \mathbf{x}$ 符号不定。图中有些箭头背离 $(0, 0)$ ，有些指向 $(0, 0)$ 。

正定性	矩阵 A 和函数	三维可视化	二维可视化
正定	$A = \begin{bmatrix} 1 & \\ & 1 \end{bmatrix}$ $f(x_1, x_2) = x_1^2 + x_2^2$		
正定	$A = \begin{bmatrix} 1 & \\ & 2 \end{bmatrix}$ $f(x_1, x_2) = x_1^2 + 2x_2^2$		
正定	$A = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$ $f(x_1, x_2) = 1.5x_1^2 + x_1x_2 + 1.5x_2^2$		

半正定	$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ $f(x_1, x_2) = x_1^2$		
半正定	$A = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{bmatrix}$ $f(x_1, x_2) = 0.5x_1^2 - x_1x_2 + 0.5x_2^2$		
半正定	$A = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ $f(x_1, x_2) = x_2^2$		
负定	$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$ $f(x_1, x_2) = -x_1^2 - x_2^2$		
负定	$A = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}$ $f(x_1, x_2) = -x_1^2 - 2x_2^2$		
负定	$A = \begin{bmatrix} -1.5 & -0.5 \\ -0.5 & -1.5 \end{bmatrix}$ $f(x_1, x_2) = -1.5x_1^2 - x_1x_2 - 1.5x_2^2$		

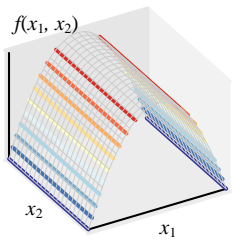
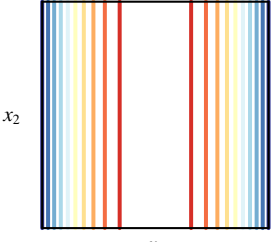
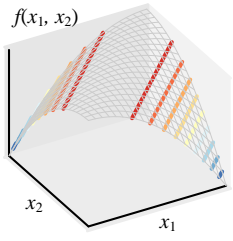
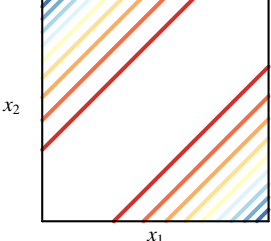
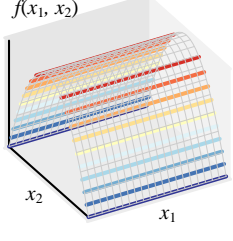
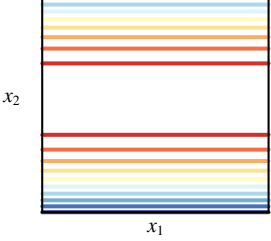
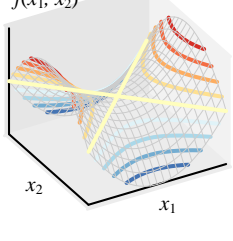
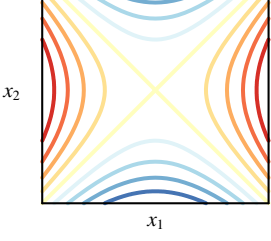
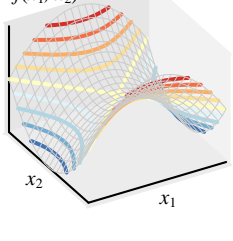
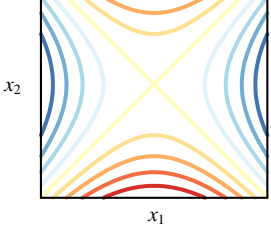
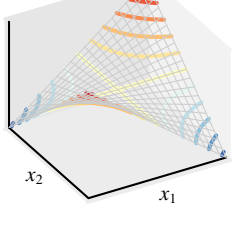
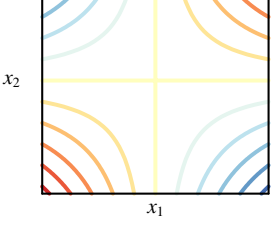
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

半负定	$A = \begin{bmatrix} -1 & \\ & 0 \end{bmatrix}$ $f(x_1, x_2) = -x_1^2$		
半负定	$A = \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$ $f(x_1, x_2) = -0.5x_1^2 + x_1x_2 - 0.5x_2^2$		
半负定	$A = \begin{bmatrix} 0 & \\ & -1 \end{bmatrix}$ $f(x_1, x_2) = -x_2^2$		
不定	$A = \begin{bmatrix} 1 & \\ & -1 \end{bmatrix}$ $f(x_1, x_2) = x_1^2 - x_2^2$		
不定	$A = \begin{bmatrix} -1 & \\ & 1 \end{bmatrix}$ $f(x_1, x_2) = -x_1^2 + x_2^2$		
不定	$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ $f(x_1, x_2) = 2x_1x_2$		

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

矩阵分解

图 9 完成符号矩阵 $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 的特征值和特征向量。

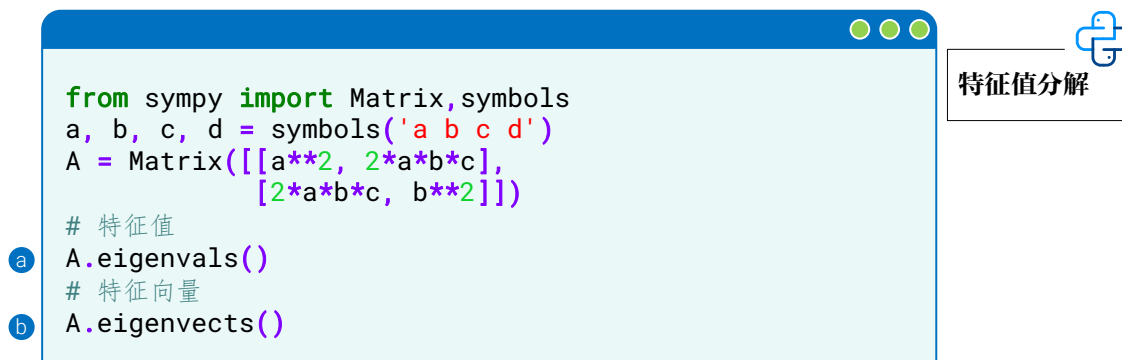


图 9. 用 sympy 完成符号矩阵的特征值分解

图 10 完成矩阵 $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$ 的奇异值分解。 U 的结果为 $U = \begin{bmatrix} \sqrt{2}/2 & \sqrt{6}/6 \\ 0 & \sqrt{6}/3 \\ -\sqrt{2}/2 & \sqrt{6}/6 \end{bmatrix}$, S 的结果为 $S = \begin{bmatrix} 1 & \\ & \sqrt{3} \end{bmatrix}$, V 的结果为 $V = \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$ 。请大家分别计算 $V.T @ V$, $V @ V.T$, $U.T @ U$ 。

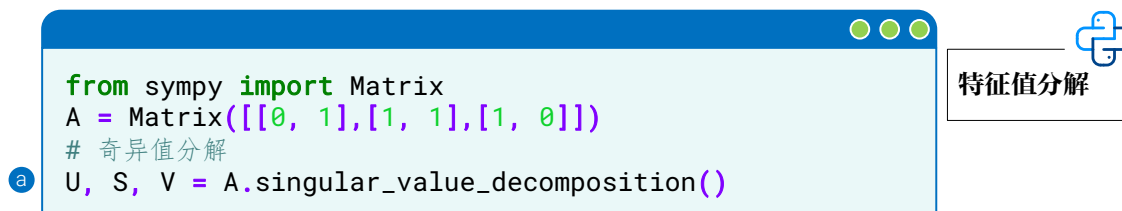


图 10. 用 sympy 完成矩阵的奇异值分解

请大家注意，SymPy 目前很多功能还不够完善。大家想要处理更为复杂的符号运算，建议使用 Mathematica 或 MATLAB Symbolic Math Toolbox。