

10

2D Contours

平面等高线

每条曲线上的不同点具有相同的数值



征服你自己，而不是世界。

Conquer yourself rather than the world.

—— 勒内·笛卡尔 (René Descartes) | 法国哲学家、数学家、物理学家 | 1596 ~ 1650



- ◀ matplotlib.pyplot.contour() 绘制等高线图
- ◀ matplotlib.pyplot.contourf() 绘制平面填充等高线
- ◀ matplotlib.pyplot.plot_trisurf() 在三角形网格上绘制平滑的三维曲面图
- ◀ matplotlib.pyplot.plot_wireframe() 绘制线框图
- ◀ matplotlib.pyplot.scatter() 绘制散点图
- ◀ matplotlib.pyplot.tricontourf() 在三角形网格上绘制填充的等高线图
- ◀ matplotlib.pyplot.triplot() 在三角形网格上绘制线条
- ◀ matplotlib.tri.Triangulation() 生成三角剖分对象
- ◀ matplotlib.tri.UniformTriRefiner() 对三角形网格进行均匀细化，生成更密集的三角形网格，以提高绘制的精细度和准确性
- ◀ numpy.asarray() 将输入数据，比如列表、元组等，转换为 NumPy 数组
- ◀ numpy.column_stack() 将两个矩阵按列合并
- ◀ numpy.concatenate() 将多个数组进行连接
- ◀ numpy.cos() 计算余弦值
- ◀ numpy.linalg.det() 计算行列式值
- ◀ numpy.linalg.inv() 矩阵求逆
- ◀ numpy.linspace() 在指定的间隔内，返回固定步长的数据
- ◀ numpy.meshgrid() 产生网格化数据
- ◀ numpy.ones_like() 用来生成和输入矩阵形状相同的全 1 矩阵
- ◀ numpy.sin() 计算正弦值
- ◀ numpy.sqrt() 计算平方根
- ◀ numpy.vstack() 返回竖直堆叠后的数组
- ◀ numpy.zeros_like() 用来生成和输入矩阵形状相同的零矩阵
- ◀ scipy.spatial.Delaunay() 生成一个点集的 Delaunay 三角剖分
- ◀ scipy.stats.dirichlet.pdf() 计算 Dirichlet 分布的概率密度函数
- ◀ sympy.diff() 求解符号导数和偏导解析式
- ◀ sympy.exp() 符号自然指数
- ◀ sympy.lambdify() 将符号表达式转化为函数

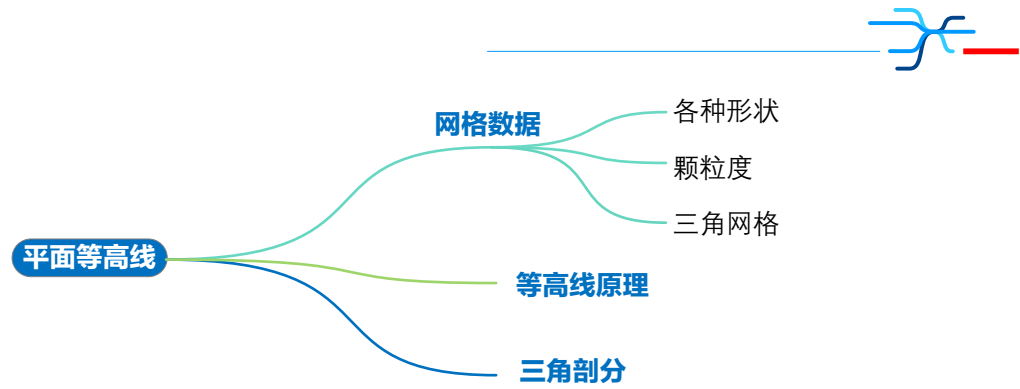
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

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



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

10.1 网格数据

在介绍平面等高线之前，我们先回顾一下网格数据。

相信大家已经对 `numpy.meshgrid()` 函数并不陌生。NumPy 中的 `meshgrid()` 函数用于生成网格状的坐标点矩阵，其作用是将两个或多个一维数组转换为多维数组。具体来说，`meshgrid()` 函数接受两个或多个一维数组作为参数，返回多维坐标矩阵。图 1 所示为生成二维网络状坐标原理。

图 2 所示为从三维空间视角看二维网络状散点。

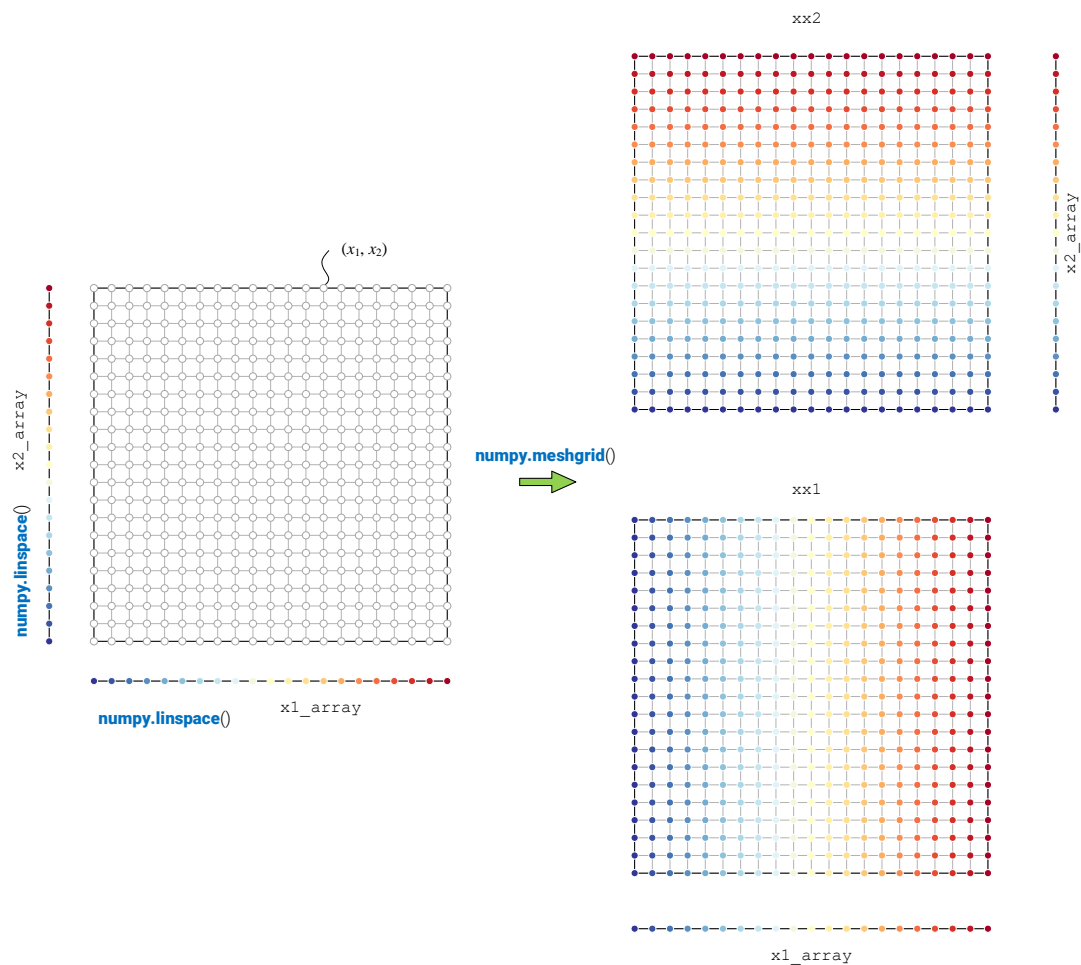
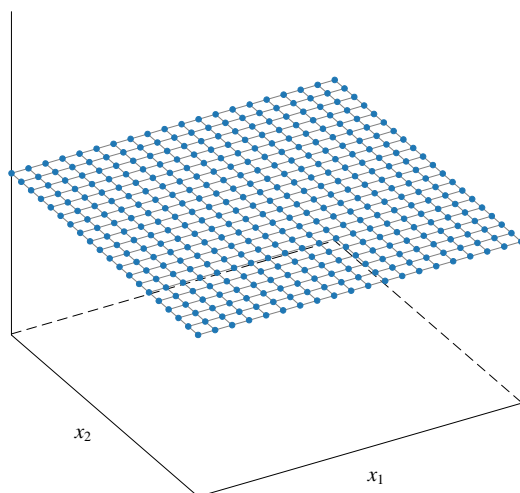


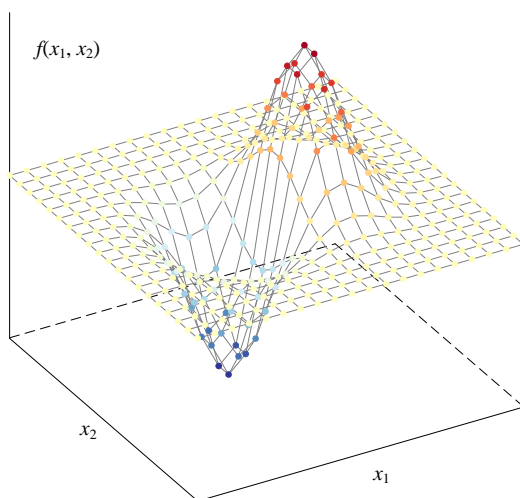
图 1. 用 `numpy.meshgrid()` 生成二维网络数据

图 2. 三维空间看二维网络状坐标 | [BK_2_Ch10_01.ipynb](#)

网格状坐标的用途

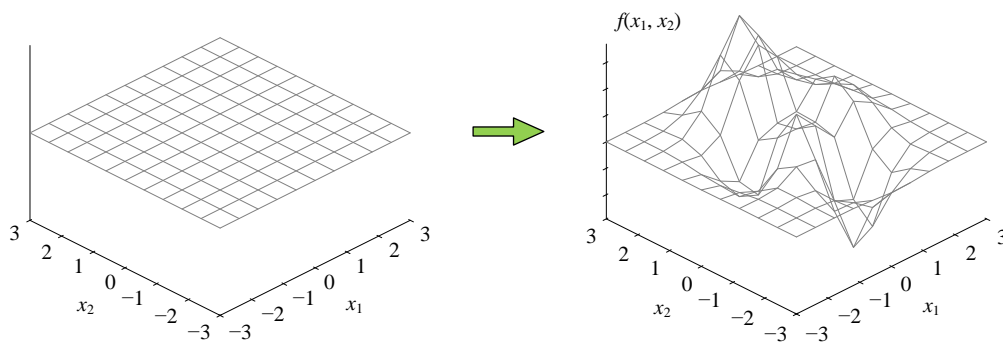
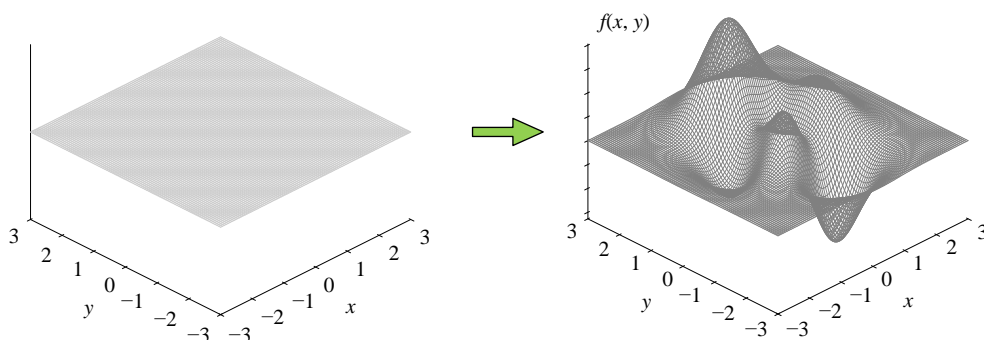
`numpy.meshgrid()` 产生的二维网络状坐标通常用于绘制网格曲面、等高线等场景。图 3 所示为用网格曲面 + 散点可视化二元函数 $f(x_1, x_2)$ 。

本书后续，大家会看到我们用网格状坐标绘制等高线。

图 3. 二维网络状坐标可视化二元函数 | [BK_2_Ch10_01.ipynb](#)

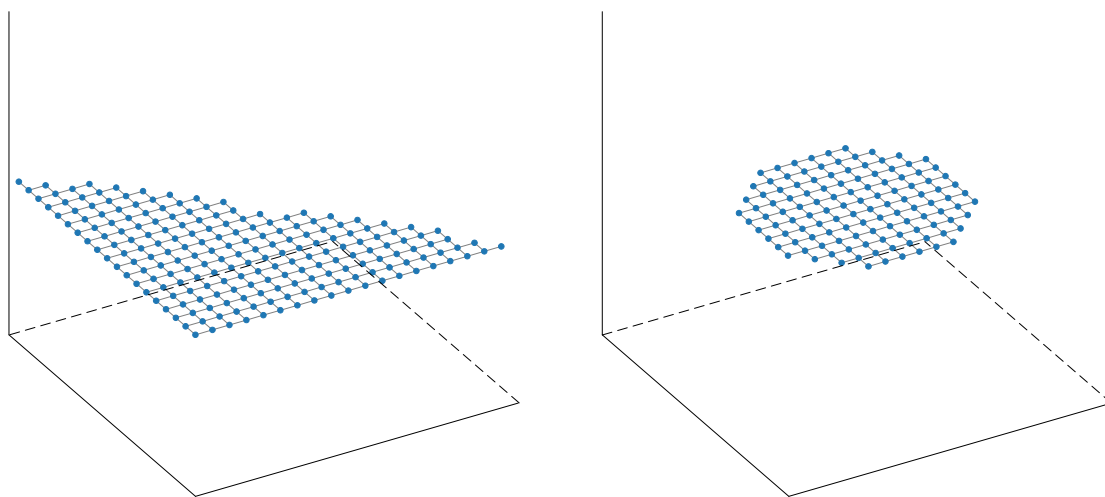
颗粒度

类似平面线图，利用网络状坐标可视化数据时，也会遇到颗粒度的问题。如图 4、图 5 所示，颗粒度过低、过高都会导致可视化效果不理想。本书后文将分别从等高线、网格曲面等几个角度继续颗粒度这个话题。

图 4. 颗粒度低 | [BK_2_Ch10_01.ipynb](#)图 5. 颗粒度过高 | [BK_2_Ch10_01.ipynb](#)

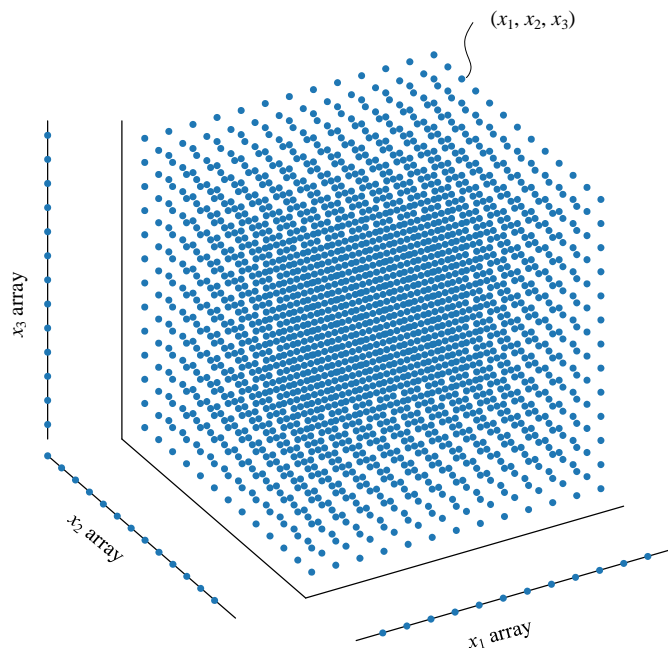
使用面具

类似前文线图，对于网格我们也可以使用**面具**（mask）。图 6 所示的两个例子为满足特定条件的的部分网格数据。

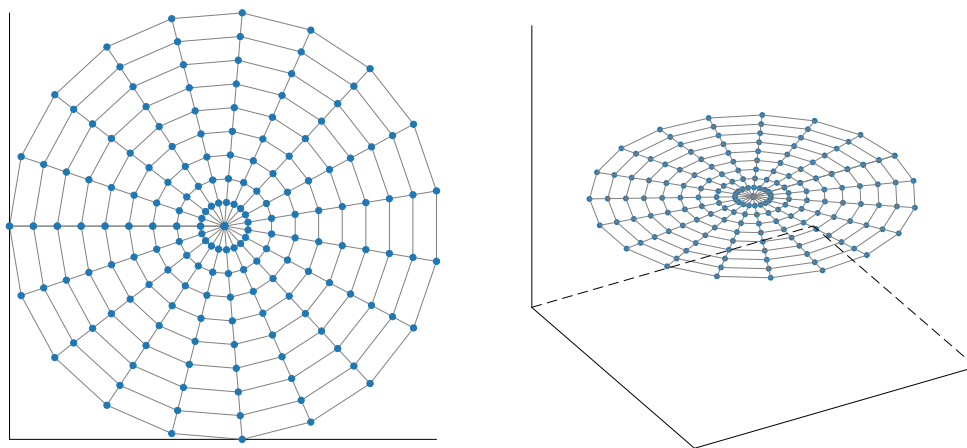
图 6. 使用面具 | [BK_2_Ch10_01.ipynb](#)

三维网格

此外，大家对图 7 所示三维网格也应该不陌生。我们在色彩模型中用过它。此外，本书后文还会继续用三维网格散点提供更为丰富的可视化方案。

图 7. 三维网格 | [BK_2_Ch10_01.ipynb](#)

除了方方正正的网格，本系列丛书还会用到极坐标网格。产生如图 8 所示的极坐标网格很容易。首先利用 `numpy.linspace()` 生成极角、极轴的数组，然后用 `numpy.meshgrid()` 生成极坐标网格坐标，最后再将其从极坐标转化为平面直角坐标系坐标。

图 8. 极坐标网格 | [BK_2_Ch10_01.ipynb](#)

本章最后还会使用三角网格完成特定的可视化方案。三角形网格是由一系列三角形所组成的网络结构。在计算机图形学和计算机模拟等领域，三角形网格常被用于表示复杂的几何体，如曲面、体细胞等，它可以通过三角形边界的拼接来逼近这些复杂的几何形状。三角形网格也常被用于数值计算中，如有限元分析等，因为三角形具有良好的性质，如易于计算、几何尺寸不变等。

三角形网格可以由多种方式生成，其中最常见的是 Delaunay 三角剖分，该方法可以将给定的点集分割成一组不重叠、不交叉的三角形。在 Delaunay 三角剖分中，对于任意三角形，其外接圆不包含其他点，这种性质可以保证三角形的质量较高，从而使得数值计算更加准确和稳定。

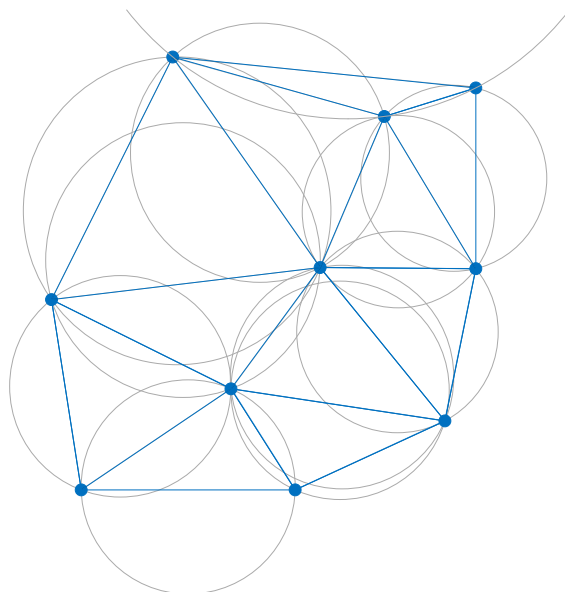


图 9. Delaunay 三角剖分法

`matplotlib.tri` 是一个 Python 库，用于创建和操作三角形网格。它提供了许多用于可视化和分析三角形网格的功能。

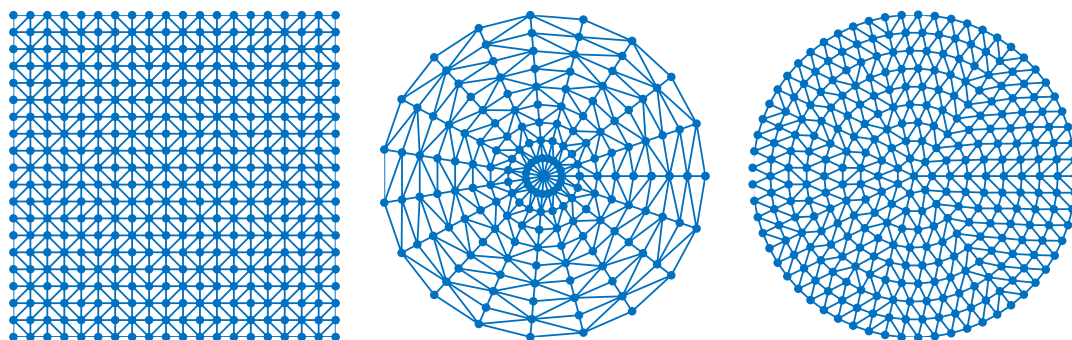
`matplotlib.tri` 可以创建三角形网格。可以使用 `Triangulation` 类从给定的点集中创建一个三角形网格，也可以使用其他函数生成各种类型的网格，如 Delaunay 三角剖分等。

`matplotlib.tri` 还可以可视化三角形网格。可以使用 `tripcolor`、`tricontour` 等函数在三角形网格上绘制颜色填充、等高线图。

`matplotlib.tri` 也可以操作三角形网格。比如，可以使用 `TriAnalyzer`、`TriInterpolator` 等类对三角形网格进行分析、插值等操作。

总的来说，`matplotlib.tri` 为处理三角形网格提供了很多方便的工具和函数，使得用户可以方便地进行可视化和分析。

图 10 所示为常见三种三角网格。本书后续还将深入介绍三角形网格及其应用场景。

图 10. 三角网格 |  BK_2_Ch10_01.ipynb

大家如果对 Delaunay 三角剖分法感兴趣的话，可以参考：

<https://mathworld.wolfram.com/DelaunayTriangulation.html>

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

10.2 等高线

平面等高线、三维等高线是“鸢尾花书”中非常重要的可视化方案。我们常用等高线可视化二元乃至多元函数、概率密度函数、机器学习中的决策边界等等。

这个话题主要介绍平面等高线，本书后续将专门介绍三维等高线。

等高线原理

等高线图是一种展示三维数据的方式，其中相同数值的数据点被连接成曲线，形成轮廓线。

形象地说，如图 11 所示，二元函数相当于一座山峰。在平行于 x_1x_2 平面在特定高度切一刀，得到的轮廓线就是一条等高线。这是一条三维空间等高线。然后，将等高线投影到 x_1x_2 平面，我们便得到一条平面等高线。

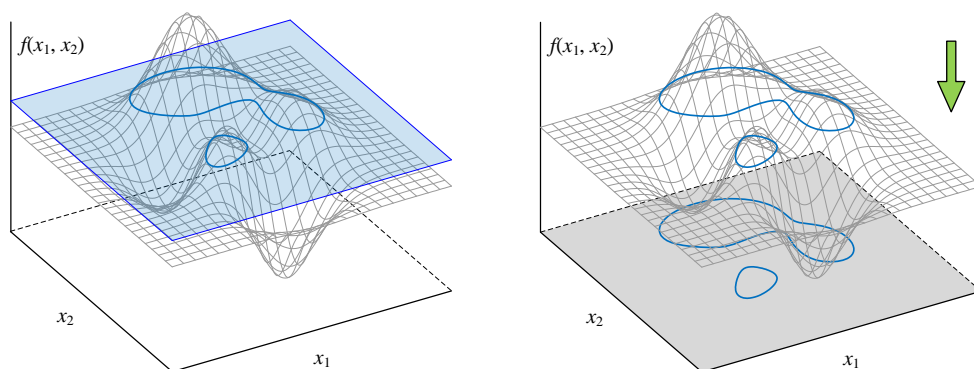


图 11. 平行 x_1x_2 平面切 $f(x_1, x_2)$ ，然后再投影到 x_1x_2 平面 | [BK_2_Ch10_02.ipynb](#)

一系列轮廓线的高度一般用不同的颜色或线型表示，使得我们可以通过视觉化方式看到数据的分布情况。如图 12 所示，将一组不同高度的等高线投影到平面便得到如图 13 所示的平面等高线。图 13 的右侧还增加了色谱条，用来展示不同等高线对应的具体高度。这一系列高度可以是一组用户输入的数值。大家可能已经发现，等高线图和海拔高度图原理完全相同。类似的图还有，等温线、等降水线、等距线等等。

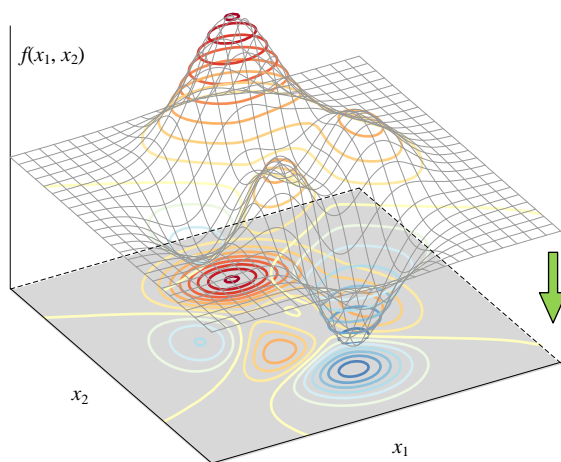


图 12. 将不同高度（值）对应的一组等高线投影到 x_1x_2 平面 | [BK_2_Ch10_02.ipynb](#)

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

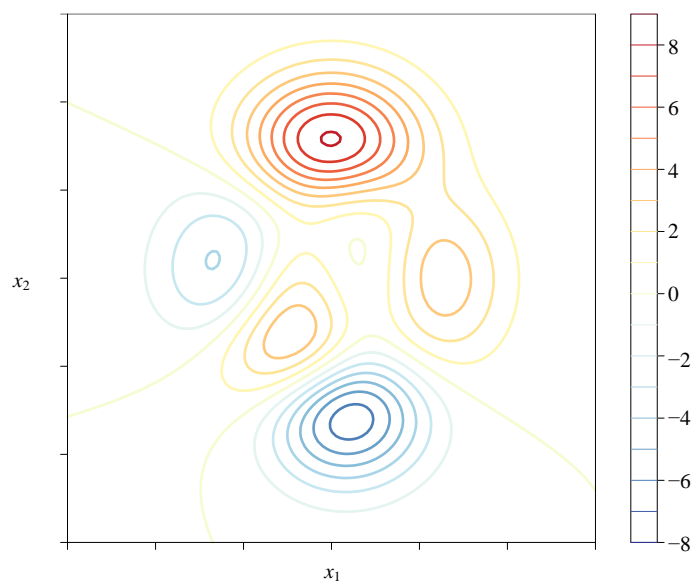


图 13. 平面等高线 | BK_2_Ch10_02.ipynb

步骤

具体来说，Matplotlib 中绘制等高线图需要以下步骤：

准备数据：等高线图需要的数据是一个二维数组，其中每个元素都表示一个点的数值。通常这个二维数组被称为“网格”。

计算轮廓线：Matplotlib 会通过对数据进行插值，计算出一组轮廓线的值，并把它们绘制在二维平面上。轮廓线的数量取决于我们指定的等高线的数量。

绘制轮廓线：Matplotlib 会根据轮廓线的高度在不同的颜色或线型中表示，使得我们可以通过视觉化方式看到数据的分布情况。通常使用 `plt.contour()` 函数进行绘制。

等高线设置

Matplotlib 还提供了许多与等高线图相关的函数和选项，例如设置轮廓线的样式、标签、标记等。这些选项可以帮助我们更好地展示数据。

图 14 所示为给每条等高线增加高度注释。图 15 所示为单色等高线，matplotlib 会用虚线代表负值。

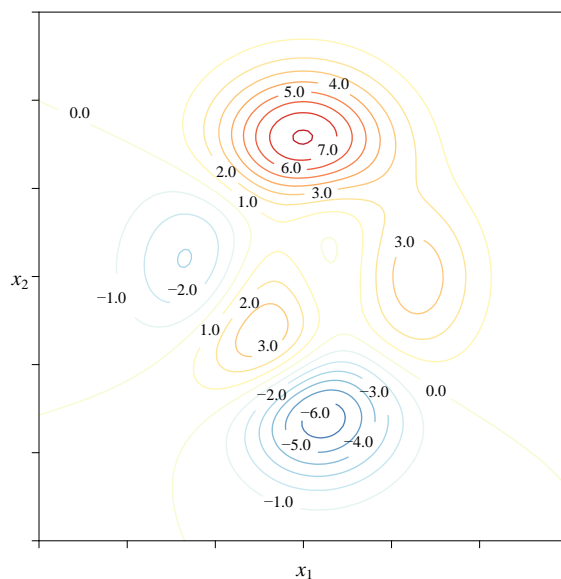


图 14. 给等高线加注释 | BK_2_Ch10_02.ipynb

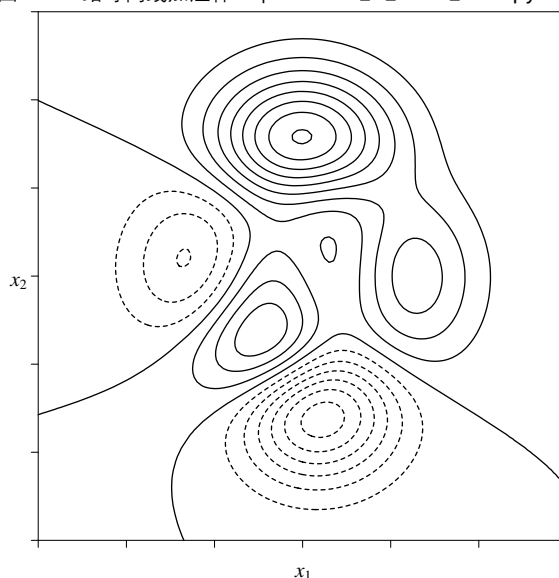
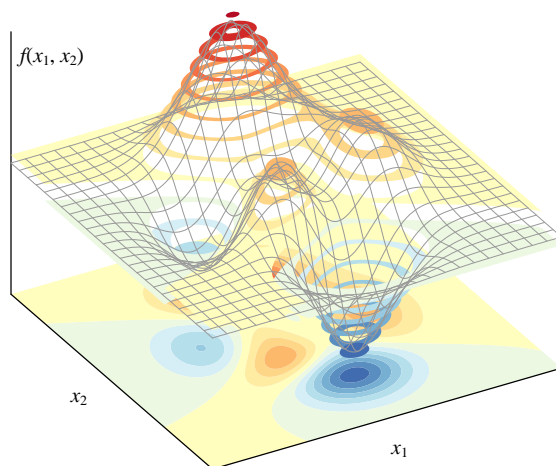
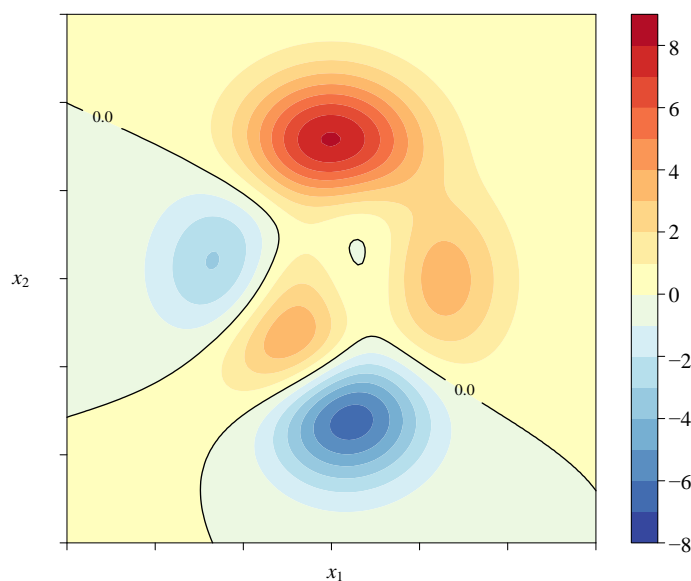


图 15. 单色等高线 | BK_2_Ch10_02.ipynb

填充等高线

`matplotlib.pyplot.contourf()`, 简作 `contourf`, 是 `Matplotlib` 库中用于绘制等高线填充图的函数。其原理是通过将数据转换为等高线线段的集合, 然后通过填充线段之间的空隙来创建颜色填充图。具体地说, `contourf` 函数首先根据数据生成一组等高线, 这些等高线可以使用 `contour` 函数绘制。然后 `contourf` 函数会根据这些等高线, 将图像中每个等高线所围成的区域填充上颜色。填充的颜色根据指定的 `colormap` 进行选择, 可以通过设置 `cmap` 参数来控制。

如图 16 所示, 在三维空间中, 我们可以把填充等高线想象成是“梯田”。每个颜色代表一定的高度区间。将图 16 填充等高线画在水平面上, 我们便得到图 17。图 17 中, 我们还绘制一条指定高度 (0.0) 的等高线。

图 16. 填充等高线原理 | [BK_2_Ch10_02.ipynb](#)图 17. 平面填充等高线 | [BK_2_Ch10_02.ipynb](#)

此外，`contourf` 还可以通过调整填充级别的数量和间隔来控制填充效果，可以通过设置 `levels` 参数来控制填充级别的数量和大小。此外，还可以通过设置 `alpha` 参数来控制填充颜色的透明度，以及通过设置 `extend` 参数来控制 `levels` 范围之外的值应如何处理。

在机器学习中，填充等高线常被用来绘制决策边界。图 23 所示为决策树、朴素贝叶斯两种方法分类鸢尾花数据集的决策边界，请大家自行学习 [BK_2_Ch10_03.ipynb](#)。

颗粒度

前文提过，绘制连续等高线基于插值，因此等高线是否“平滑”，也取决于网格数据的颗粒度是否足够细腻。

如图 18 所示，绘制等高线采用的网格数据显然过于粗糙，这导致不管是三维等高线，还是平面等高线都非常毛糙。

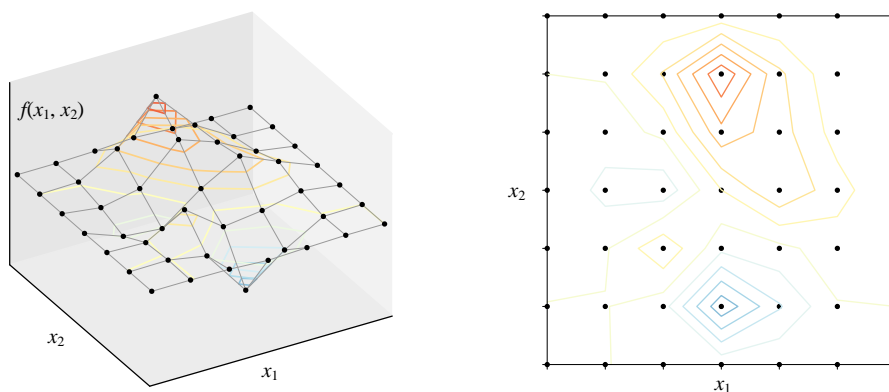
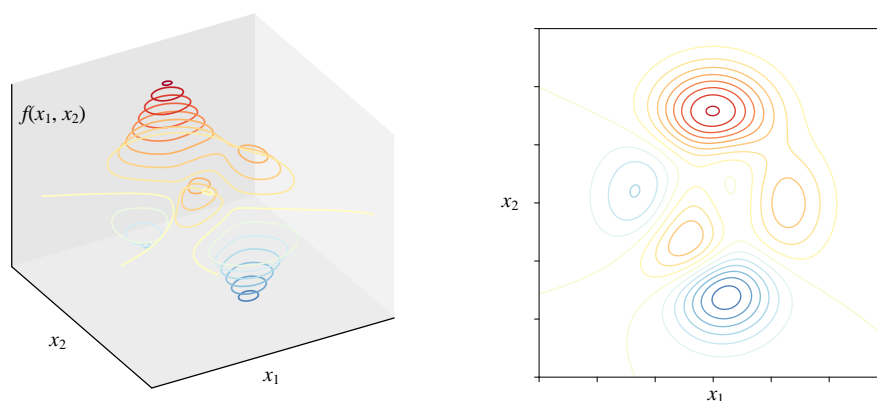
图 18. 颗粒度粗糙 | [BK_2_Ch10_04.ipynb](#)

图 19 则采用更为细腻的网格数据。显然，插值得到的三维等高线、平面等高线看上去更加平滑。网格不是越密越好。网格能够取得多密，还受到算力制约。本书后续还会介绍等高线的更广泛的用途。

图 19. 颗粒度细腻 | [BK_2_Ch10_04.ipynb](#)

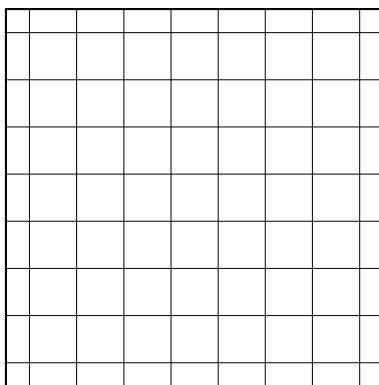
Jupyter 笔记 [BK_2_Ch10_04.ipynb](#) 绘制图 18、图 19。

可视化线性、非线性变换

如图 20 所示，`contour()` 函数还可以绘制网格。图 24、图 25 展示几种线性、非线性变换，图中网格均用 `contour()` 绘制。其中几幅图用到了复数函数，本书后续还要展示其他可视化方案呈现复数函数。此外，本书后续还会专门讲解线性变换、仿射变换。

复数 (complex number) 是数学中的一个概念，用来表示具有实部和虚部的数。复数通常表示为 $a + bi$ 的形式，其中 a 是实部， b 是虚部， i 是虚数单位。实部和虚部都可以是实数。复数的集合用 C 表示。

复数函数是定义在复数域上的函数，即将复数作为输入并产生复数作为输出的函数。复数函数可以包含各种数学操作和运算，例如加法、减法、乘法、除法、指数函数、对数函数等。

图 20. 用 `contour()` 函数绘制的方格 | [BK_2_Ch10_05.ipynb](#)

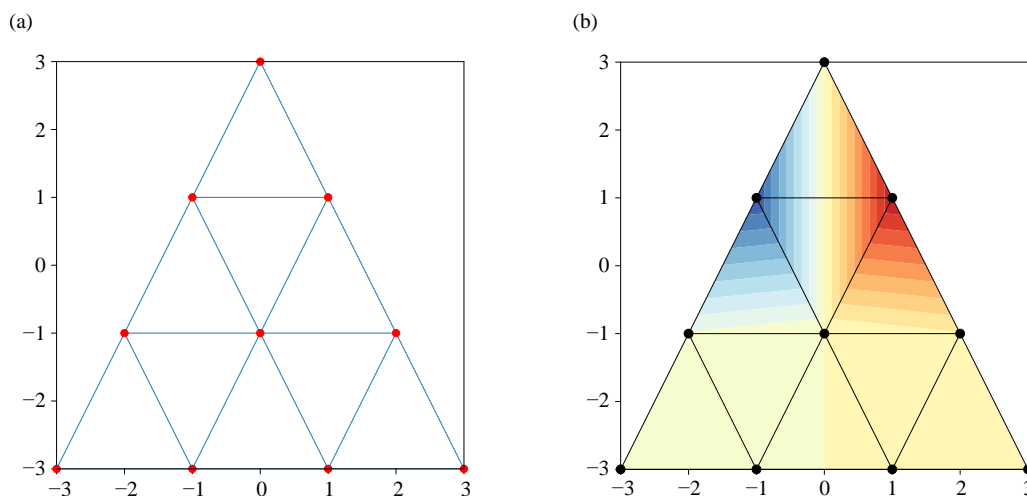
10.3 三角剖分

Delaunay 三角剖分 (Delaunay triangulation) 是计算几何学中的一个重要概念，用于将给定的几何形状划分为一组不重叠的三角形。它的名字来源于它的发明者 Boris Delaunay。

详细来说，三角剖分的目标是将一个多边形或多边形的集合分解成一组互不相交的三角形，使得这些三角形的顶点恰好是原始几何形状的顶点，并且任意两个三角形之间的交集只能是共享一个边或顶点。Delaunay 三角剖分是计算机图形学、计算几何和计算机视觉中常用的技术之一，它在三维重建、图像处理、自然语言处理、机器学习等领域都有广泛的应用。

`matplotlib.tri` 是 Matplotlib 中的一个模块，提供了三角剖分的绘图功能。`scipy.spatial` 中的 `Delaunay` 类可以帮助我们生成一个点集的 Delaunay 三角剖分，它可以用于构建三角形网格、寻找最近邻等等。

图 21 (a) 所示的网格可以手动设定，也可以自动生成。自动生成的三角网格采用 Delaunay 三角剖分。三角网格可以帮助我们绘制各种类型的三角形网格，例如等高线图、三角形色块图和三角形曲面图等。图 21 (b) 所示为三角网格等高线图。

图 21. 三角网格和等高线 | [BK_2_Ch10_06.ipynb](#)

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>

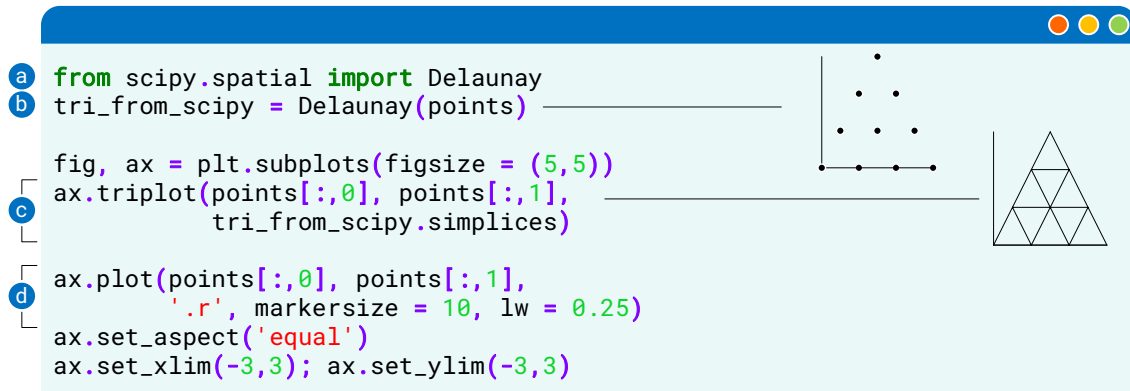
本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

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

BK_2_Ch10_06.ipynb 绘制图 21 两幅子图，下面聊聊其中关键语句。

代码 1 绘制图 21 (a)。注意，图中三角形是等腰三角形，不是等边三角形。

- a 从 `scipy.spatial` 导入 `Delaunay` 对象，`Delaunay` 用于三角剖分。
- b 创建 `Delaunay` 的一个实例。输入 `points` 是前文代码定义的一组坐标，结果是对这些点进行 `Delaunay` 三角剖分。
- c 在轴对象 `ax` 上用 `triplot()` 方法绘制三角剖分图。
`points[:,0]` 和 `points[:,1]` 是代表给定散点的 `x` 坐标和 `y` 坐标。
`tri_from_scipy.simplices` 是 `Delaunay` 对象的一个属性，它包含了三角形的顶点索引，这将被用于定义三角剖分的连接关系。
- d 绘制红色圆点代表 `points` 具体位置。



```

a from scipy.spatial import Delaunay
b tri_from_scipy = Delaunay(points)

fig, ax = plt.subplots(figsize = (5,5))
c ax.triplot(points[:,0], points[:,1],
             tri_from_scipy.simplices)

d ax.plot(points[:,0], points[:,1],
          '.r', markersize = 10, lw = 0.25)
ax.set_aspect('equal')
ax.set_xlim(-3,3); ax.set_ylim(-3,3)

```

代码 1. 绘制三角网格 | BK_2_Ch10_06.ipynb

代码 2 绘制图 21 (b)。

- a 将 `matplotlib.tri` 导入，简作 `mtri`。这个模块提供了很多处理和可视化三角剖分工具。这一句被注释掉，是因为在 BK_2_Ch10_06.ipynb 中，模块已经在前文导入。
- b 利用 `matplotlib.tri` 模块（简作 `mtri`）中的 `Triangulation` 根据给定的坐标点自动构建三角剖分。
- c 利用 `tricontourf()` 在轴对象 `ax` 上绘制一个基于三角剖分的等高线填充图。
`triang_auto` 之前创建的 `Triangulation` 对象示例，它定义了三角剖分的结构。
`f_fcn()` 是代码前文定义的一个二元函数。函数值将用于确定填充图中每个三角形的颜色。
`cmap='RdYlBu_r'` 指定了使用的颜色映射。
`levels=20` 指定了等高线层数。
- d 用 `triplot()` 在轴对象 `ax` 上绘制三角剖分线条图。
`'ko-'` 是线条图的样式设置，其中 `'k'` 表示黑色，`'o'` 表示在每个顶点处用圆圈标记，`'-'` 表示连接线样式。

```

a # import matplotlib.tri as mtri
b triang_auto = mtri.Triangulation(x_tri, y_tri)

fig, ax = plt.subplots(figsize = (5,5))

# 三角剖分网格绘制等高线
c ax.tricontourf(triang_auto, f_fcn(x_tri, y_tri),
                 cmap = 'RdYlBu_r',
                 levels = 20)
d ax.triplot(triang_auto, 'ko-')
ax.set_aspect('equal')
ax.set_xlim(-3,3); ax.set_ylim(-3,3)

```

代码 2. 绘制三角网格等高线 | BK_2_Ch10_06.ipynb

BK_2_Ch10_06.ipynb 笔记中给出更多三角剖分以及相关可视化的范例，请大家自行学习。

颗粒度

三角网格也存在颗粒度的问题。图 22 所示为给定等边三角形不同的颗粒度的三角网格剖分。颗粒度越高，三角网格越细腻，但是计算量也急剧增大。

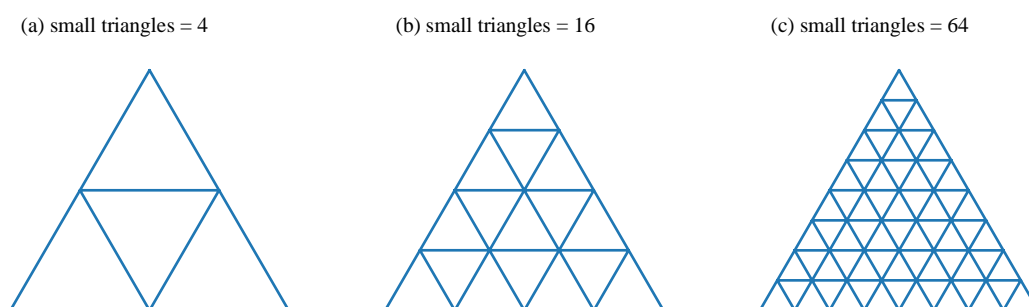


图 22. 三角网格的颗粒度 | BK_2_Ch10_07.ipynb

BK_2_Ch10_07.ipynb 绘制图 22。下面聊聊其中关键语句。

a 利用 `matplotlib.tri.Triangulation()`，简作 `tri.Triangulation()`，根据给定的坐标点自动构建三角剖分。`corners` 中有三个点，它们是等边三角形的三个顶点。

b 利用 `matplotlib.tri.UniformTriRefiner()`，简作 `tri.UniformTriRefiner()`，创建三角网格均匀细化对象实例。

c 利用 `refine_triangulation()` 方法对 **b** 中创建的 `refiner` 进行细分操作。

参数 `subdiv=subdiv_idx` 指定三角形边的细化次数，我们可以控制细化程度，以便生成更精细的三角形网格。

d 利用 `matplotlib.pyplot.subplot()`，简作 `plt.subplot()`，创建 1 行 4 列子图，并选择当前操作的子图序号 `idx`。注意，`idx` 从 1 开始编号，这是因为 `for` 循环中使用 `enumerate()` 时，加入了 1 这个参数。

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。

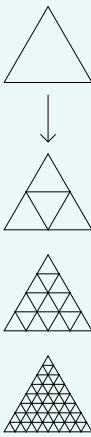
版权归清华大学出版社所有，请勿商用，引用请注明出处。

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

e 利用 `matplotlib.pyplot.triplot()`，简作 `plt.triplot()`，绘制三角剖分线条图。



```
# import matplotlib.tri as tri
corners = np.array([[0, 0], [1, 0], [0.5, 0.75**0.5]])
# 定义等边三角形的三个顶点

a triangle = tri.Triangulation(corners[:, 0], corners[:, 1])
# 构造三角形剖分对象

b refiner = tri.UniformTriRefiner(triangle)
# 对三角形网格进行均匀细化
subdiv_array = [1, 2, 3, 4]

fig, ax = plt.subplots(figsize = (12, 3))

for idx, subdiv_idx in enumerate(subdiv_array, 1):
    c trimesh_idx = refiner.refine_triangulation(subdiv=subdiv_idx)
    # 等边三角形被细化成4**subdiv 个三角形

    d plt.subplot(1, 4, idx)

    e plt.triplot(trimesh_idx)
    plt.axis('off'); plt.axis('equal')
    plt.title('Small triangles = ' + str(4**subdiv_idx))
```

代码 3. 提高三角网格颗粒度 | BK_2_Ch10_07.ipynb

本章回顾了各种网格数据，然后介绍了平面等高线原理。本章最后又聊了聊三角剖分，这部分内容对于理解本书后文要介绍的重心坐标系很重要，请大家格外注意。

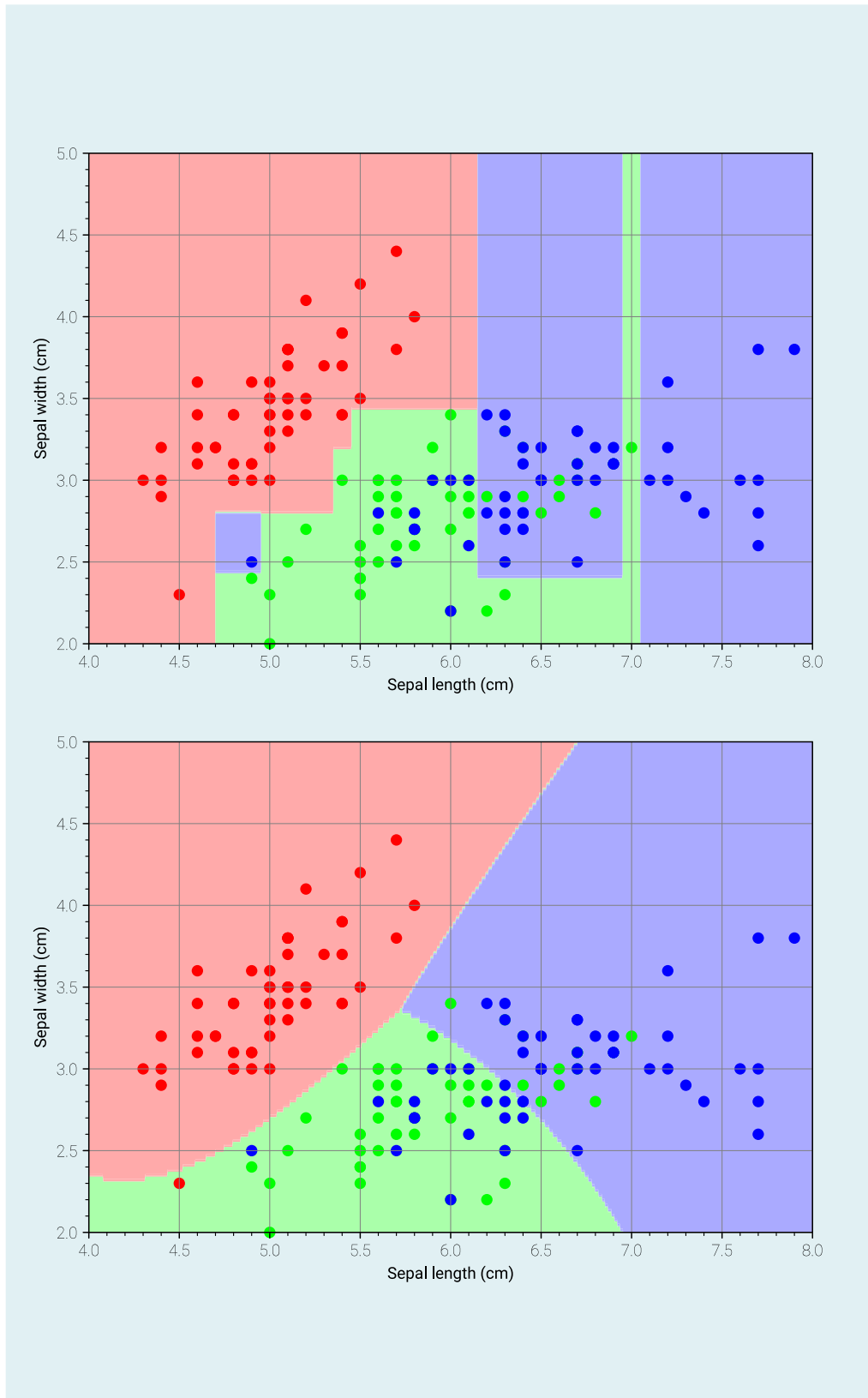



图 23. 利用 `contourf()` 绘制决策边界等高线 |  BK_2_Ch10_03.ipynb

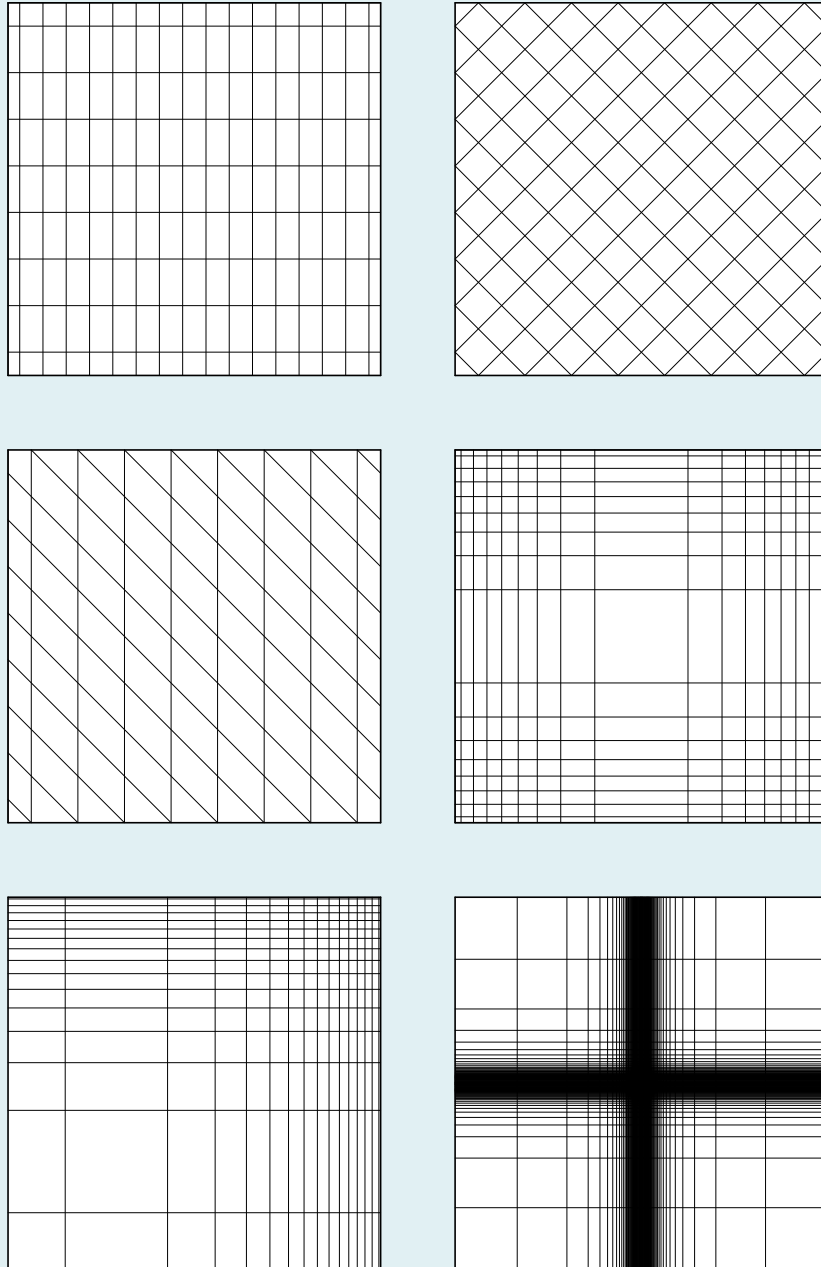



图 24. 线性、非线性变换，利用 contour 绘制，第 1 组 |  BK_2_Ch10_05.ipynb

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

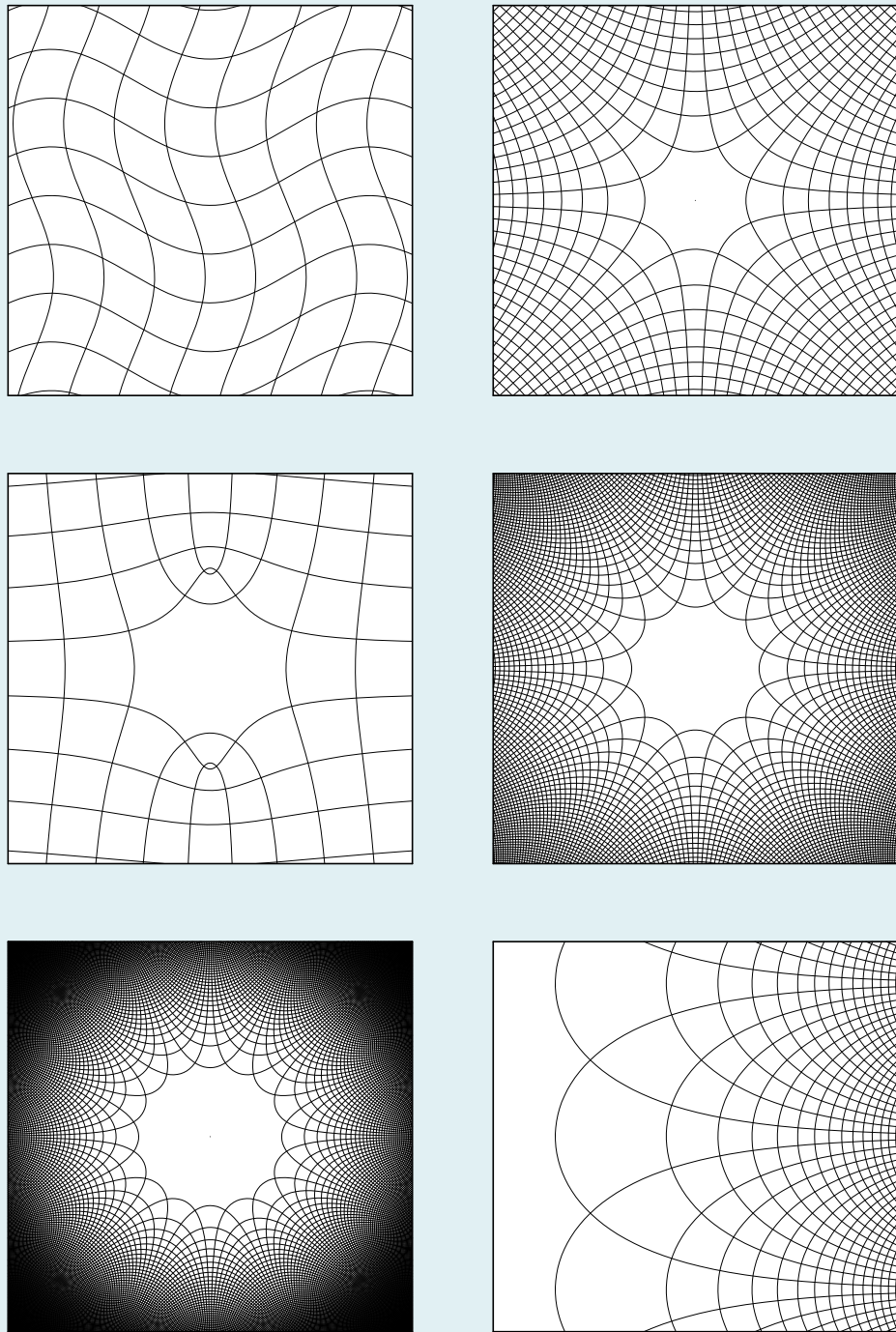



图 25. 线性、非线性变换, 利用 contour 绘制, 第 2 组 |  BK_2_Ch10_05.ipynb

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