

## 26

## Scientific Computation Using SciPy

## SciPy 数学运算

插值、积分、线性代数、优化、统计 ...



无限！没有其他问题能如此深刻地触动人类的精神。

*The infinite! No other question has ever moved so profoundly the spirit of man.*

—— 大卫·希尔伯特 (David Hilbert) | 德国数学家 | 1862 ~ 1943



```

▶ scipy.cluster.vq.kmeans() k 均值聚类
▶ scipy.constants.pi 圆周率
▶ scipy.constants.golden 黄金分割比
▶ scipy.constants.c 真空中光速
▶ scipy.fft.fft() 一维傅里叶变换
▶ scipy.integrate.quad() 定积分
▶ scipy.interpolate.interp1d() 一元插值
▶ scipy.interpolate.griddata() 在不规则数据点上数据进行数据插值
▶ scipy.io.loadmat() 导入 MATLAB 文件
▶ scipy.io.savemat() 保存 MATLAB 文件
▶ scipy.linalg.inv() 矩阵逆
▶ scipy.linalg.det() 行列式
▶ scipy.linalg.pinv() Moore-Penrose 伪逆
▶ scipy.linalg.eig() EVD 特征值分解
▶ scipy.linalg.cholesky() Cholesky 分解
▶ scipy.linalg.qr() QR 分解
▶ scipy.linalg.svd() SVD 奇异值分解
▶ scipy.ndimage.gaussian_filter() 高斯滤波
▶ scipy.ndimage.convolve() 多维卷积
▶ scipy.optimize.root() 求根
▶ scipy.optimize.minimize() 最小化
▶ scipy.signal.convolve() 卷积
▶ scipy.sparse.linalg.inv() 稀疏矩阵的逆
▶ scipy.sparse.linalg.norm() 稀疏矩阵范数
▶ scipy.spatial.distance.euclidean() 欧氏距离
▶ scipy.spatial.distance_matrix() 距离矩阵
▶ scipy.special.factorial() 阶乘
▶ scipy.special.gamma() Gamma 函数
▶ scipy.special.beta() Beta 函数
▶ scipy.special.erf() 误差函数
▶ scipy.special.comb() 组合数
▶ scipy.stats.norm() 一元高斯分布
▶ scipy.stats.multivariate_normal() 多元高斯分布
▶ scipy.stats.gaussian_kde() 高斯核密度估计

```



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](mailto:jiang.visualize.ml@gmail.com)

## 26.1 什么是 SciPy?

SciPy 是一个 Python 的开源科学计算库，SciPy 构建在 NumPy 之上，并提供了许多有用的功能，用于数值计算、优化、统计和信号处理等科学与工程领域。一些具体的用途包括：

- ▶ 数据预处理和特征工程：SciPy 提供了丰富的工具用于数据的插值、滤波、变换等，这些在数据预处理和特征工程中很有用。
- ▶ 优化问题：SciPy 中的 `optimize` 模块包含了各种常用的优化算法，可用于解决机器学习中的参数优化问题，例如模型训练中的参数调整。
- ▶ 数值计算：SciPy 提供了高效的数值计算工具，例如求解线性代数问题、解微分方程、积分等，在数值计算密集型的机器学习任务中很有帮助。
- ▶ 统计分析：SciPy 中的 `stats` 模块提供了许多常用的统计分析函数，例如概率分布函数、假设检验等，可以用于数据分析和模型评估。
- ▶ 信号处理：SciPy 中的 `signal` 模块提供了信号处理的工具，例如滤波、傅里叶变换等，这些在处理时间序列数据或图像数据时非常有用。
- ▶ SciPy 强大且灵活，因此在机器学习领域也有广泛的应用。在机器学习领域，SciPy 主要用于数据预处理、特征工程、优化问题、数值计算、统计分析以及信号处理等方面。

本章介绍如何使用 SciPy 中几个常见函数。

表 1. SciPy 常用模块以及示例函数

模块名称	描述	举例
<code>scipy.cluster</code>	聚类	<code>scipy.cluster.vq.kmeans()</code> k 均值聚类
<code>scipy.constants</code>	数学和物理常数	<code>scipy.constants.pi</code> 圆周率 <code>scipy.constants.golden</code> 黄金分割比 <code>scipy.constants.c</code> 真空中光速
<code>scipy.fft</code>	快速傅里叶变换	<code>scipy.fft.fft()</code> 一维傅里叶变换
<code>scipy.integrate</code>	积分	<code>scipy.integrate.quad()</code> 定积分
<code>scipy.interpolate</code>	插值和拟合	<code>scipy.interpolate.interp1d()</code> 一元插值 <code>scipy.interpolate.griddata()</code> 在不规则数据点进行数据插值
<code>scipy.io</code>	数据输入输出	<code>scipy.io.loadmat()</code> 导入 MATLAB 文件 <code>scipy.io.savemat()</code> 保存 MATLAB 文件
<code>scipy.linalg</code>	线性代数	<code>scipy.linalg.inv()</code> 矩阵逆 <code>scipy.linalg.det()</code> 行列式 <code>scipy.linalg.pinv()</code> Moore-Penrose 伪逆 <code>scipy.linalg.eig()</code> EVD 特征值分解 <code>scipy.linalg.cholesky()</code> Cholesky 分解 <code>scipy.linalg.qr()</code> QR 分解 <code>scipy.linalg.svd()</code> SVD 奇异值分解
<code>scipy.ndimage</code>	$n$ 维图像处理	<code>scipy.ndimage.gaussian_filter()</code> 高斯滤波 <code>scipy.ndimage.convolve()</code> 多维卷积
<code>scipy.odr</code>	正交回归（正交距离回归）	
<code>scipy.optimize</code>	优化算法	<code>scipy.optimize.root()</code> 求根 <code>scipy.optimize.minimize()</code> 最小化 <code>scipy.optimize.curve_fit()</code> 拟合
<code>scipy.signal</code>	信号处理	<code>scipy.signal.convolve()</code> 卷积
<code>scipy.sparse</code>	稀疏矩阵工具	<code>scipy.sparse.linalg.inv()</code> 稀疏矩阵的逆 <code>scipy.sparse.linalg.norm()</code> 稀疏矩阵范数

scipy.spatial	空间数据结构和算法	scipy.spatial.distance.euclidean() 欧氏距离 scipy.spatial.distance_matrix() 距离矩阵
scipy.special	特殊数学函数	scipy.special.factorial() 阶乘 scipy.special.gamma() Gamma 函数 scipy.special.beta() Beta 函数 scipy.special.erf() 误差函数 scipy.special.comb() 组合数
scipy.stats	统计	scipy.stats.norm() 一元高斯分布 scipy.stats.multivariate_normal() 多元高斯分布 scipy.stats.gaussian_kde() 高斯核密度估计

## 26.2 距离

图 1 所示的平面上两点，(8, 8) 和 (2, 0)，之间的欧氏距离为  $\sqrt{(8-2)^2 + (8-0)^2} = 10$ 。利用 SciPy 函数 `scipy.spatial.distance.euclidean([8,8],[2,0])`，我们可以得到同样结果。

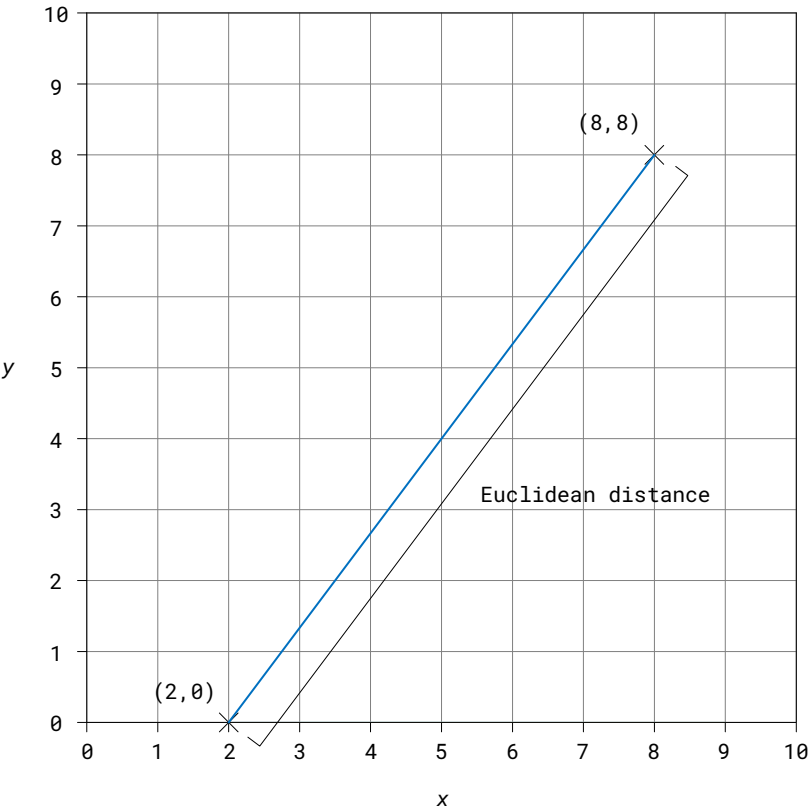


图 1. 平面上两点之间的欧氏距离

图 2 所示为利用随机数发生器生成的 26 个平面坐标，对应 26 个字母；其中 B 和 S 重叠，D 和 O 重叠。图中彩色线为两两成对坐标连线，距离远的用暖色系颜色渲染，距离近的用冷色系颜色渲染。图 3 所示为成对距离矩阵。

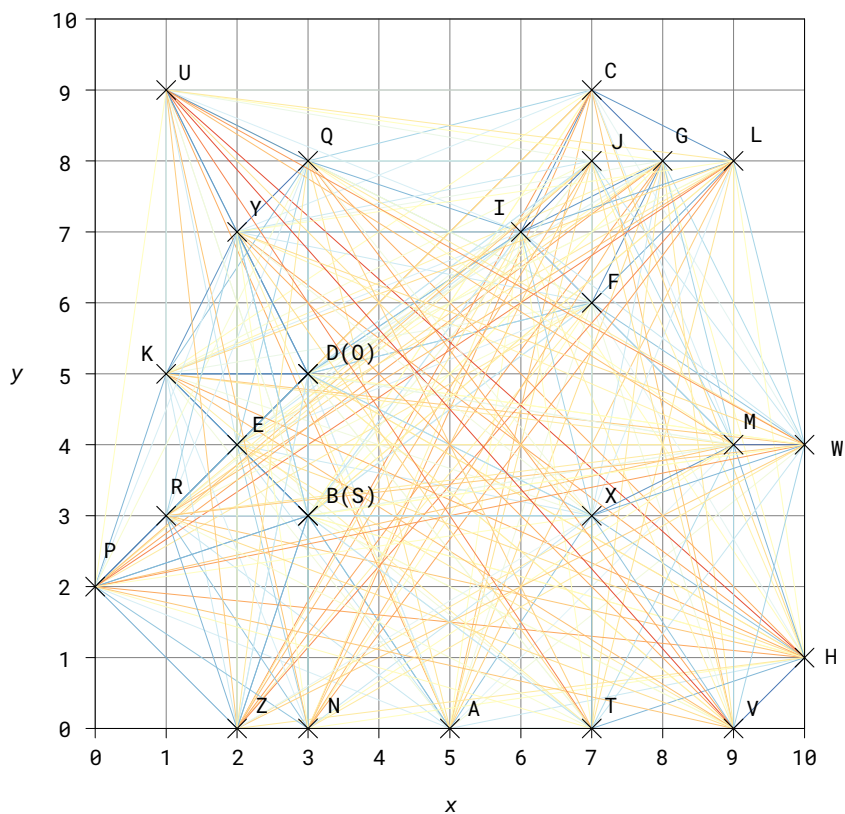


图 2. 平面上 26 个点之间的两两欧氏距离

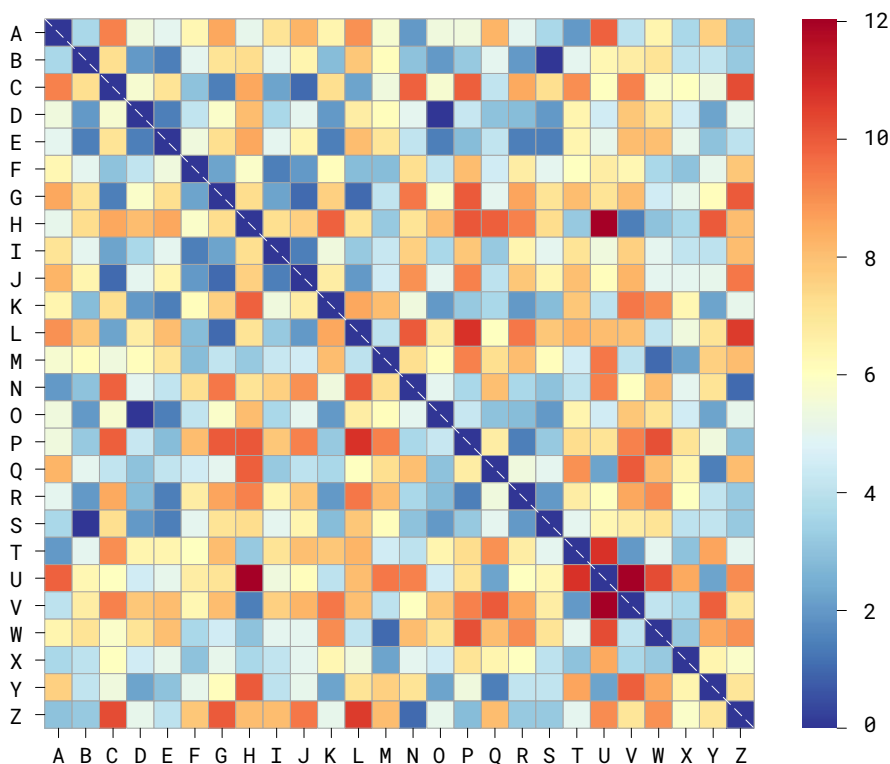


图 3. 成对欧氏距离矩阵

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](mailto:jiang.visualize.ml@gmail.com)

图 4 所示为绘制图 2 和图 3 代码。下面我们分析其中一些关键语句。

```
import matplotlib.pyplot as plt
import itertools
import numpy as np
import matplotlib as mpl
import seaborn as sns
a import string
b from scipy.spatial import distance_matrix
c from scipy.spatial.distance import euclidean
import os
# 如果文件夹不存在, 创建文件夹
if not os.path.isdir("Figures"):
    os.makedirs("Figures")
# 产生随机数
num = 26
d np.random.seed(0)
e data = np.random.randint(10 + 1, size=(num, 2))
f labels = list(string.ascii_uppercase)

g cmap = mpl.cm.get_cmap('RdYlBu_r')
fig, ax = plt.subplots()
# 绘制成对线段
h for i, d in enumerate(itertools.combinations(data, 2)):
i     d_idx = euclidean(d[0], d[1])
j     plt.plot([d[0][0], d[1][0]],
               [d[0][1], d[1][1]],
               color = cmap(d_idx/np.sqrt(2)/10), lw = 1)
k ax.scatter(data[:,0], data[:,1],
             marker = 'x', color = 'k', s = 50, zorder=100)
# 添加标签
l for i, txt in enumerate(labels):
    ax.annotate(txt, (data[i,0] + 0.2, data[i,1] + 0.2))

ax.set_xlim(0, 10); ax.set_ylim(0, 10)
ax.set_xticks(np.arange(11))
ax.set_yticks(np.arange(11))
plt.xlabel('x'); plt.ylabel('y')
ax.grid(ls='--', lw=0.25, color=[0.5, 0.5, 0.5])
ax.set_aspect('equal', adjustable='box')
fig.savefig('Figures/成对距离连线.svg', format='svg')

# 计算成对距离矩阵
m pairwise_distances = distance_matrix(data, data)
fig, ax = plt.subplots()
n sns.heatmap(pairwise_distances,
              cmap = 'RdYlBu_r', square = True,
              xticklabels = labels, yticklabels = labels,
              ax = ax)
fig.savefig('Figures/成对距离矩阵热图.svg', format='svg')
```

图 4. 计算、可视化成对距离, 代码

**a** 导入 Python 标准库中的 `string` 模块。Python 中的 `string` 模块提供了许多字符串处理相关的函数和常量, 可以方便地进行字符串操作。比如, `string.ascii_uppercase`: 包含所有大写 ASCII 字母 (A-Z) 的字符串, `string.digits`: 包含所有数字 (0-9) 的字符串。

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](mailto:jiang.visualize.ml@gmail.com)

**b** 从 `scipy.spatial` 中导入 `distance_matrix` 函数，用于计算多个点之间的成对距离矩阵。它接受点坐标的数组或列表，然后计算每两个点之间的距离，并返回一个矩阵，其中的每个元素表示两个点之间的距离。

**c** 从 `scipy.spatial.distance` 模块中导入了 `euclidean` 函数，用来计算两点欧氏距离。

**d** 设置随机数生成器的种子 `seed` 为 `0`，从而使随机数的生成具有确定性，保证实验结果可重复性。

**e** 在 `[0, 10]` 区间之内生成随机整数，形状为 26 行、2 列。

**f** 生成 A-Z 大写字母字符串，并将其转换为列表。

**g** 从 `matplotlib` 通过 `cm.get_cmap()` 函数来获取一个名为 `'RdYlBu_r'` 的颜色映射对象。`'RdYlBu_r'` 是一个预定义的颜色映射名称，它表示一种从红色 Rd 到黄色 Yl 再到蓝色 Bu 的颜色渐变，且颜色映射反向（末尾带 `_r` 表示反向）。鸢尾花书也管颜色映射叫色谱。

颜色映射对象通常被用于将数据的数值范围 `[0, 1]` 映射到一系列颜色中的某个位置。这个数值范围一般默认为 `[0, 1]`，其中 `0` 对应着颜色映射的起始位置，`1` 对应着颜色映射的结束位置。颜色映射会将 `[0, 1]` 区间内的数据值线性地映射到预定义的颜色序列上。在使用 `Matplotlib` 中的颜色映射对象时，可以使用 `matplotlib.colors.Normalize()` 函数将数据规范化到 `[0, 1]` 区间，然后再将规范化后的数据传递给颜色映射对象来获取对应的颜色。《可视之美》还会进一步介绍非线性映射，以及如何构造颜色映射。

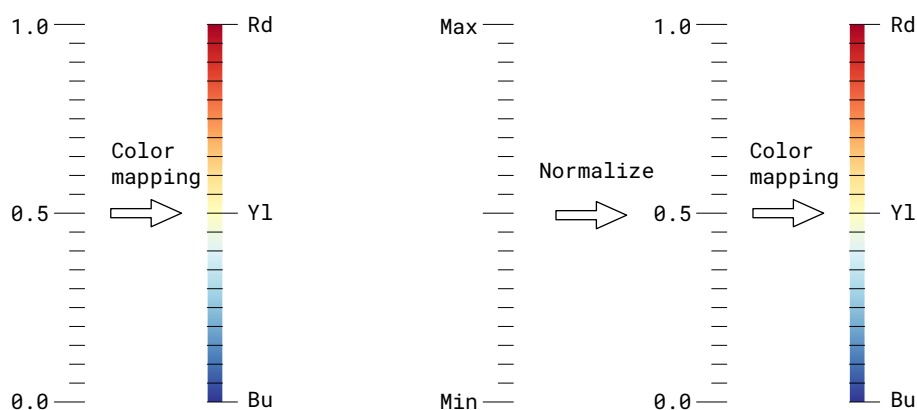


图 5. 颜色映射

**h** 使用了 Python 中的 `enumerate` 函数和 `itertools.combinations` 函数，用于在数据 `data` 的所有两两组合之间进行循环迭代，并在每次迭代中获取索引和组合数据。

**i** 利用 `scipy.spatial.distance.euclidean()` 计算两个点之间的欧氏距离。

**k** 把图 3 欧氏距离转化为 `[0, 1]` 之间的数。显然在图 3 平面上，最大的距离为  $10 \times \sqrt{2}$ 。

**l** 通过 `for` 循环利用 `annotate()` 给每个散点添加字母标签。

**m** 计算 26 个散点的成对距离矩阵，这个矩阵的大小为  $26 \times 26$ 。这个矩阵的主对角线（图 3 虚线）的元素代表某个点到自身的距离，即 0。我们容易发现，图 3 这个矩阵沿着主对角线对称；因此这个距离矩阵也叫对称矩阵（**symmetric matrix**）。换个角度来看，我们只需要这个  $26 \times 26$  矩阵中除主对角线以外，下三角（图 6）或上三角矩阵的元素信息。

**n** 利用 `seaborn.heatmap()` 绘制成对距离热图。

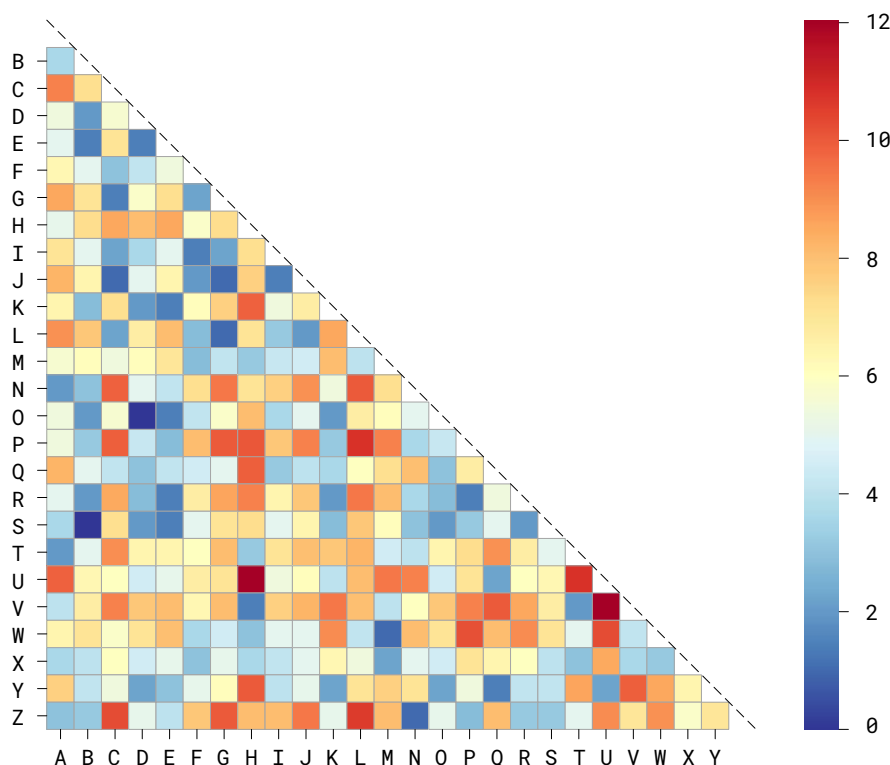


图 6. 剔除主对角线元素的下三角矩阵

图 7 代码绘制图 6。和图 4 代码不同的是，在生成成对距离矩阵之后，我们还生成了一个（剔除主对角线）下三角矩阵的面具（**mask**）。在鸢尾花书中，**mask** 一般被直译为面具，也常被翻译做蒙皮、掩码、遮罩等等。

**a** 用了 NumPy 库中的函数来创建一个面具“**mask**”，用于过滤计算得到的“**pairwise\_ds**”数组。`numpy.ones_like()` 创建了一个与“**pairwise\_ds**”数组形状相同的全为 1 的布尔类型数组。“**dtype=bool**”指定数组元素的数据类型为布尔类型（**True** 或 **False**），所有元素都被设置为 **True**。`numpy.triu()` 函数的“**triu**”代表“**triangle upper**”，它是“**numpy**”库中的函数，用于获取矩阵的上三角部分（包括对角线），而将下三角部分设置为 0。

如 **b** 所示，使用 `seaborn.heatmap` 绘制热图时，**mask** 中对应位置为 **True** 的单元格的成对距离矩阵数据将不会被显示。



```

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import string
from scipy.spatial import distance_matrix

# 产生随机数
num = 26
np.random.seed(0)
data = np.random.randint(10 + 1, size=(num, 2))
labels = list(string.ascii_uppercase)

# 计算成对距离矩阵
pairwise_ds = distance_matrix(data, data)
# 产生蒙皮/面具
a mask = np.triu(np.ones_like(pairwise_ds, dtype=bool))
fig, ax = plt.subplots()
b sns.heatmap(pairwise_distances,
               mask = mask,
               cmap = 'RdYlBu_r',
               square = True,
               xticklabels = labels,
               yticklabels = labels,
               ax = ax)
fig.savefig('下三角.svg', format='svg')

```

图 7. 可视化成对距离矩阵下三角部分（不含主对角线元素），代码

## 26.3 插值

插值（interpolation）是通过已知数据点之间的值来估计未知点的值的方法，它可以用于填补数据缺失或者进行数据平滑处理。

如图 8 所示的蓝色点为已知数据点，插值就是根据这几个离散的数据点估算其他点对应的  $y$  值。

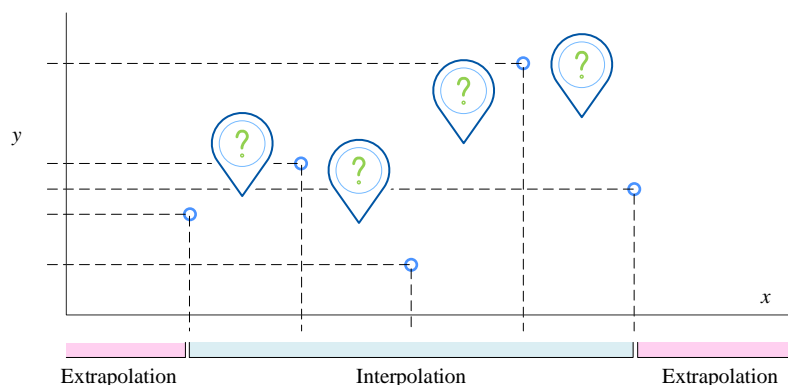


图 8. 插值的意义



插值可分为**内插** (interpolation) 和**外插** (extrapolation)。内插是在已知数据点之间进行插值，估计出未知点的值。而外插则是在已知数据点的范围之外进行插值，从而预测超出已知数据点范围的未知点的值。在进行外插时，需要考虑插值函数是否能够正确地拟合未知数据点，并且需要注意不要过度依赖插值函数来进行预测，以免导致不可靠的预测结果。

图 9 比较六种插值方法，下面结合图 10 逐一介绍。

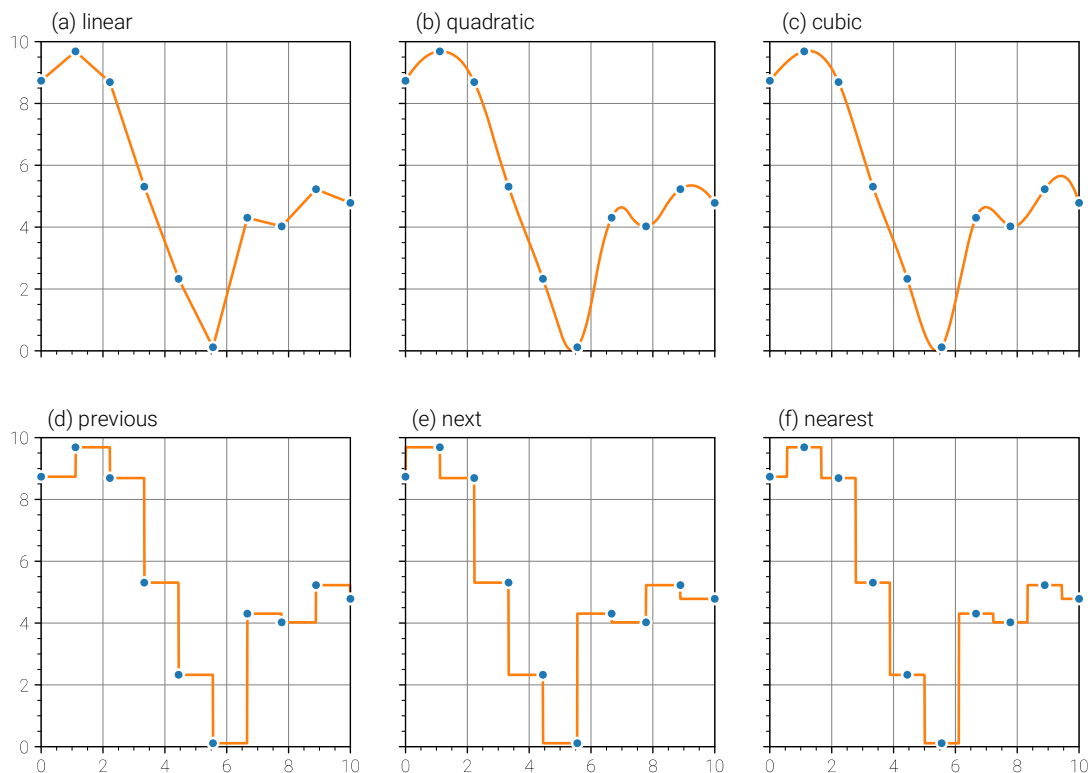


图 9. 比较六种不同插值方法

**a** 创建一个 2 行 3 列的图形子图网格，并设置图形的尺寸和共享坐标轴属性。参数 2, 3 指定了网格的行数 (2) 和列数 (3)，即总共有 6 个子图。`sharex='col'` 指定每一列子图将共享相同的 x 轴，而 `sharey='row'` 指定每一行子图将共享相同的 y 轴。这样设置可以使得网格中的子图在 x 轴和 y 轴方向上有一致的刻度和范围。

**b** 中的 `flatten()` 多维数组转换为一维数组。在这里，函数被应用于 "axes" 轴对象，将二维的子图网格数组转换成了一维数组。

**c** 列列表出 6 种插值方法。

**d** 调用 SciPy 库中的 `interp1d` 函数来进行一维插值。其中，`x` 这是一个一维数组或列表，表示原始数据点的横坐标，即自变量。`y` 也是一个一维数组或列表，表示原始数据点的纵坐标，即因变量。

参数 `kind` 用于指定插值方法。其中，`'linear'` 为线性插值。在两个相邻数据点之间进行线性插值，即使用直线来连接两个数据点。如图 9 (a) 所示，多点线性插值结果一般为折线。

`'quadratic'` 是二次插值，相邻点之间通过二次函数连接。如图 9 (b) 所示，二次插值产生的曲线较为平滑。

`'cubic'` 是三次插值，相邻点之间通过三次函数连接。如图 9 (c) 所示，三次插值产生的曲线非常平滑，能够更好地逼近数据点之间的曲线。

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](mailto:jiang.visualize.ml@gmail.com)

'nearest' 代表最近邻插值。如图 9 (d) 所示, 'nearest' 使用与插值点最近的数据点的值作为插值结果。

'previous' 代表前向插值。如图 9 (e) 所示, 使用插值点之前的数据点的值作为插值结果。

'next' 代表后向插值。如图 9 (f) 所示, 使用插值点之后的数据点的值作为插值结果。

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

# 生成随数据
np.random.seed(8)
x = np.linspace(0, 10, 10)
y = np.random.rand(10) * 10
x_fine = np.linspace(0, 10, 1001)

# 创建一个图形对象, 包含六个子图
a fig, axes = plt.subplots(2, 3, figsize=(6, 9),
                           sharex='col',
                           sharey='row')
b axes = axes.flatten()

# 六种插值方法
c methods = ['linear', 'quadratic', 'cubic',
             'previous', 'next', 'nearest']

for i, method in enumerate(methods):

    # 创建 interp1d 对象
    d f = interp1d(x, y, kind=method)

    # 生成插值后的新数据点
    e y_fine = f(x_fine)

    # 绘制子图
    axes[i].plot(x, y, 'o', label='Data',
                 markeredgewidth=1.5,
                 markeredgewidthcolor='w',
                 zorder=100)
    axes[i].plot(x_fine, y_fine, label='Interpolated')
    axes[i].set_title(f'Method: {method}')
    axes[i].legend()
    axes[i].set_xlim(0, 10)
    axes[i].set_ylim(0, 10)
    axes[i].set_aspect('equal', adjustable='box')
plt.tight_layout()
fig.savefig('不同插值方法.svg', format='svg')
```

图 10. 比较六种插值方法, 代码

大家经常混淆拟合和插值这两种方法。插值和拟合有一个相同之处，它们都是根据已知数据点，构造函数，从而推断得到更多数据点。

插值和回归都是用于对数据进行预测的方法，但两者有明显的区别。插值是用于填补已有数据点之间的空缺，预测未知点的值。回归则是预测自变量和因变量之间的关系。插值通常使用插值函数，如多项式插值；而回归则通过拟合数据点的回归方程来预测因变量的值。插值通常用于数据平滑处理、数据填补等。回归则常用于预测和建模。插值要求原始数据点之间要有一定的连续性和平滑性；而回归则对数据点的分布没有明显要求。插值得到的是精确的函数值，但在超出已有数据范围时可能不准确；而回归得到的是变量之间的大致关系，可以预测未来的趋势。

需要根据具体情况选择合适的方法。当数据缺失或需要平滑处理时，可以使用插值方法；当需要建立模型并预测未来趋势时，可以使用回归方法。

插值一般得到分段函数，分段函数通过所有给定的数据点，如图 11 (a)、(b) 所示。回归拟合得到的函数尽可能靠近样本数据点，如图 11 (c)、(d) 所示。

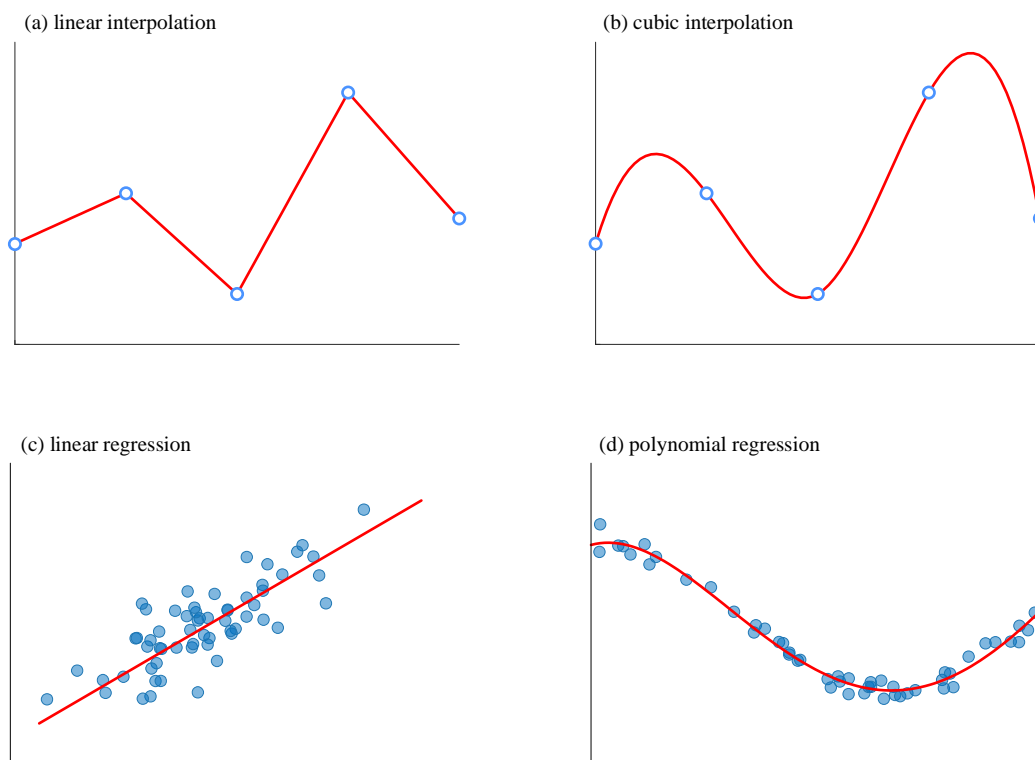


图 11. 比较一维插值和回归

## 26.4 高斯分布

高斯分布 (Gaussian Distribution)，也称为正态分布 (Normal Distribution)，是概率论和统计学中最重要且广泛应用的分布之一。高斯分布以数学家卡尔·弗里德里希·高斯 (Carl Friedrich Gauss) 的名字命名。

### 一元高斯分布

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](mailto:jiang.visualize.ml@gmail.com)

一元高斯分布概率密度函数 (Probability Density Function, PDF) 的特点是钟形曲线, 对称分布, 均值  $\mu$  和标准差  $\sigma$  决定了分布的位置和形状。均值决定了曲线的中心, 标准差决定了曲线的宽窄程度。图 12 (a) 所示均值  $\mu$  对一元高斯分布概率密度函数形状影响。图 12 (b) 所示标准差  $\sigma$  对一元高斯分布概率密度函数形状影响。



### 什么是概率密度函数?

概率密度函数是用于描述连续随机变量的概率分布的数学函数。它指定了随机变量落在不同取值范围内的概率密度, 而不是具体的概率值。一元随机变量的 PDF 在整个取值范围内的面积等于 1, 因为随机变量必然会在某个取值范围内取值。

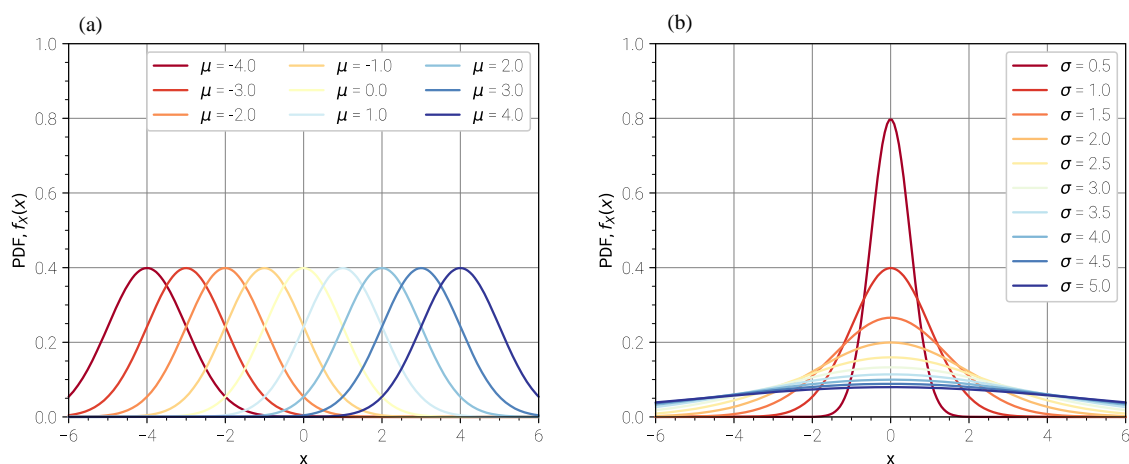


图 12. 一元高斯分布 PDF, 均值  $\mu$ 、标准差  $\sigma$  分别影响

图 13 绘制图 12 两幅子图。下面讲解图 13 中代码中最要的语句。

**a** 从 `scipy.stats` 模块导入 `norm` 子模块, 为一元正态分布对象。导入 `norm` 模块后, 可以使用其中提供的函数和方法来进行正态分布相关的操作, 比如计算概率密度函数 PDF、累积分布函数 CDF、随机样本的生成等。

**b** 中 `np.linspace(0, 1, len(mu_array))` 返回一个由 0 到 1 之间等间隔的数值构成的数组, 数组的长度与 `mu_array` 的长度相同。`mu_array` 是之前定义的不同的均值取值。这些 [0, 1] 之间的数用到颜色映射。

**c** 则利用 `scipy.stats.norm.pdf(x, loc, scale)` 函数; 其中, `x` 为需要计算概率密度的数值, 可以是一个数值或一个数组。`loc` 为正态分布的均值, `loc` 是 `location` 的简写。`scale` 代表正态分布的标准差。

**d** 设定曲线图例的字符串。其中, `'$\mu$ = '` 是一个字符串, 表示希腊字母 " $\mu$ "。

**e** 用于在绘制的图表中添加图例, `ncol` 是一个整数参数, 用于设置图例的列数。

```

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
a from scipy.stats import norm

x_array = np.linspace(-6, 6, 200)
mu_array = np.linspace(-4, 4, 9)
# 设定均值一系列取值
b colors = cm.RdYlBu(np.linspace(0,1,len(mu_array)))
# 均值对一元高斯分布PDF影响
fig, ax = plt.subplots(figsize = (5,4))
for idx, mu_idx in enumerate(mu_array):
c pdf_idx = norm.pdf(x_array, scale = 1, loc = mu_idx)
d legend_idx = '$\mu$ = ' + str(mu_idx)
plt.plot(x_array, pdf_idx,
         color=colors[idx],
         label = legend_idx)

e plt.legend(ncol=3)
ax.set_xlim(x_array.min(), x_array.max())
ax.set_ylim(0,1)
ax.set_xlabel('x')
ax.set_ylabel('PDF, $f_X(x)$')

sigma_array = np.linspace(0.5,5,10)
# 设定标准差一系列取值
colors = cm.RdYlBu(np.linspace(0,1,len(sigma_array)))
# 标准差对一元高斯分布PDF影响
fig, ax = plt.subplots(figsize = (5,4))
for idx, sigma_idx in enumerate(sigma_array):
pdf_idx = norm.pdf(x_array, scale = sigma_idx)
legend_idx = '$\sigma$ = ' + str(sigma_idx)
plt.plot(x_array, pdf_idx,
         color=colors[idx],
         label = legend_idx)

plt.legend()
ax.set_xlim(x_array.min(), x_array.max())
ax.set_ylim(0,1)
ax.set_xlabel('x')
ax.set_ylabel('PDF, $f_X(x)$')

```

图 13. 可视化一元高斯分布概率密度函数，代码

## 二元高斯分布

二元高斯分布是一个包含两个随机变量的联合概率分布。二元高斯分布的概率密度函数本质上是个二元函数。如图 14 所示，当相关性系数取不同值时，我们可以看到这个二元函数曲面形状变化。

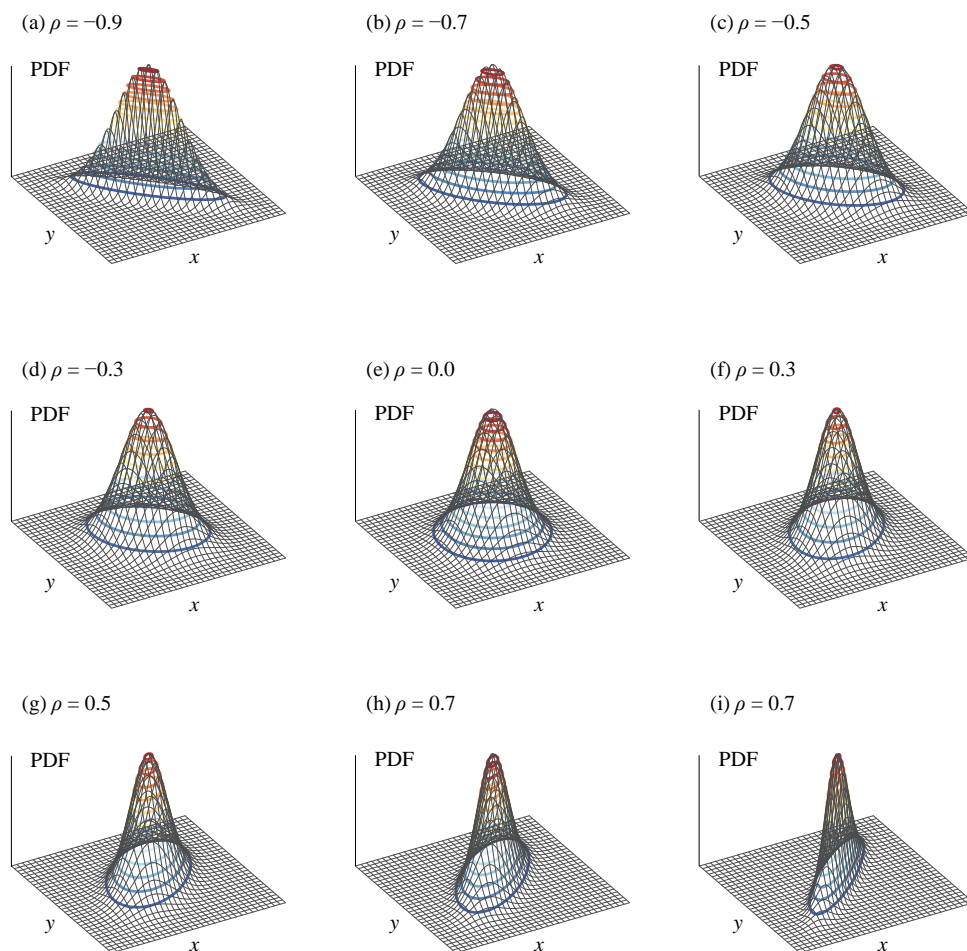


图 14. 二元高斯分布 PDF，曲面

图 15 代码绘制图 14，下面我们聊聊其中关键语句。

- a** 用列表定义了一组相关性系数取值。
- b** 定义了两个随机变量各自的标准差—— $\sigma_X$ 、 $\sigma_Y$ 。 **c** 则定义了两个随机变量的期望值—— $\mu_X$ 、 $\mu_Y$ 。
- d** 用 `numpy.dstack()` 将横纵网格坐标数组堆叠，结果为一个三维数组。
- e** 把两个期望值写成质心向量  $[\mu_X, \mu_Y]$ 。注意，这个向量是个行向量，而图 15 公式中质心向量  $\mu$  为列向量。
- f** 用相关性系数、两个标准差构造协方差矩阵 
$$\begin{bmatrix} \sigma_X^2 & \rho_{X,Y}\sigma_X\sigma_Y \\ \rho_{X,Y}\sigma_X\sigma_Y & \sigma_Y^2 \end{bmatrix}。$$
- g** 利用 `scipy.stats` 中的 `multivariate_normal` 对象，构造二元高斯分布对象实例 `bi_norm`。这个函数的输入为刚刚创建的质心向量、协方差矩阵。
- h** 利用 `bi_norm` 的 `pdf()` 方法生成概率密度函数值  $f_{X,Y}(x,y)$ 。这个方法的输入为 **d** 生成的三维数组，代表一组网格坐标点。
- i** 在三维轴对象 `ax` 上用 `plot_wireframe()` 绘制三维网格曲面。
- j** 在三维轴对象 `ax` 上用 `contour()` 再绘制一个三维等高线。



```

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import multivariate_normal

a rho_array = [-0.9, -0.7, -0.5, -0.3,
               0, 0.3, 0.5, 0.7, 0.9]
b sigma_X = 1; sigma_Y = 1 # 标准差
c mu_X = 0; mu_Y = 0 # 期望

width = 4
X = np.linspace(-width,width,321)
Y = np.linspace(-width,width,321)
XX, YY = np.meshgrid(X, Y)
d XXYY = np.dstack((XX, YY))

# 曲面
fig = plt.figure(figsize = (8,8))
for idx, rho_idx in enumerate(rho_array):
    # 质心
    e mu = [mu_X, mu_Y]
    # 协方差
    f Sigma = [[sigma_X**2, sigma_X*sigma_Y*rho_idx],
               [sigma_X*sigma_Y*rho_idx, sigma_Y**2]]
    # 二元高斯分布
    g bi_norm = multivariate_normal(mu, Sigma)
    h f_X_Y_joint = bi_norm.pdf(XXYY)

    i ax = fig.add_subplot(3,3,idx+1,projection='3d')
    ax.plot_wireframe(XX, YY, f_X_Y_joint,
                      rstride=10, cstride=10,
                      color = [0.3,0.3,0.3],
                      linewidth = 0.25)

    j ax.contour(XX,YY, f_X_Y_joint,15,
                 cmap = 'RdYlBu_r')

    ax.set_xlabel('$x$'); ax.set_ylabel('$y$')
    ax.set_zlabel('$f_{X,Y}(x,y)$')
    ax.view_init(azim=-120, elev=30)
    ax.set_proj_type('ortho')

    ax.set_xlim(-width, width); ax.set_ylim(-width, width)
    ax.set_zlim(f_X_Y_joint.min(),f_X_Y_joint.max())
    # ax.axis('off')

plt.tight_layout()
fig.savefig('二元高斯分布, 曲面.svg', format='svg')
plt.show()

```

$$f_z(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}}$$

图 15. 网格曲面可视化二元高斯分布 PDF, 代码

将图 14 投影到水平面, 如图 16 所示, 我们惊奇地发现这些曲面的等高线是椭圆! 似乎相关系数影响着椭圆的旋转角度。

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](mailto:jiang.visualize.ml@gmail.com)



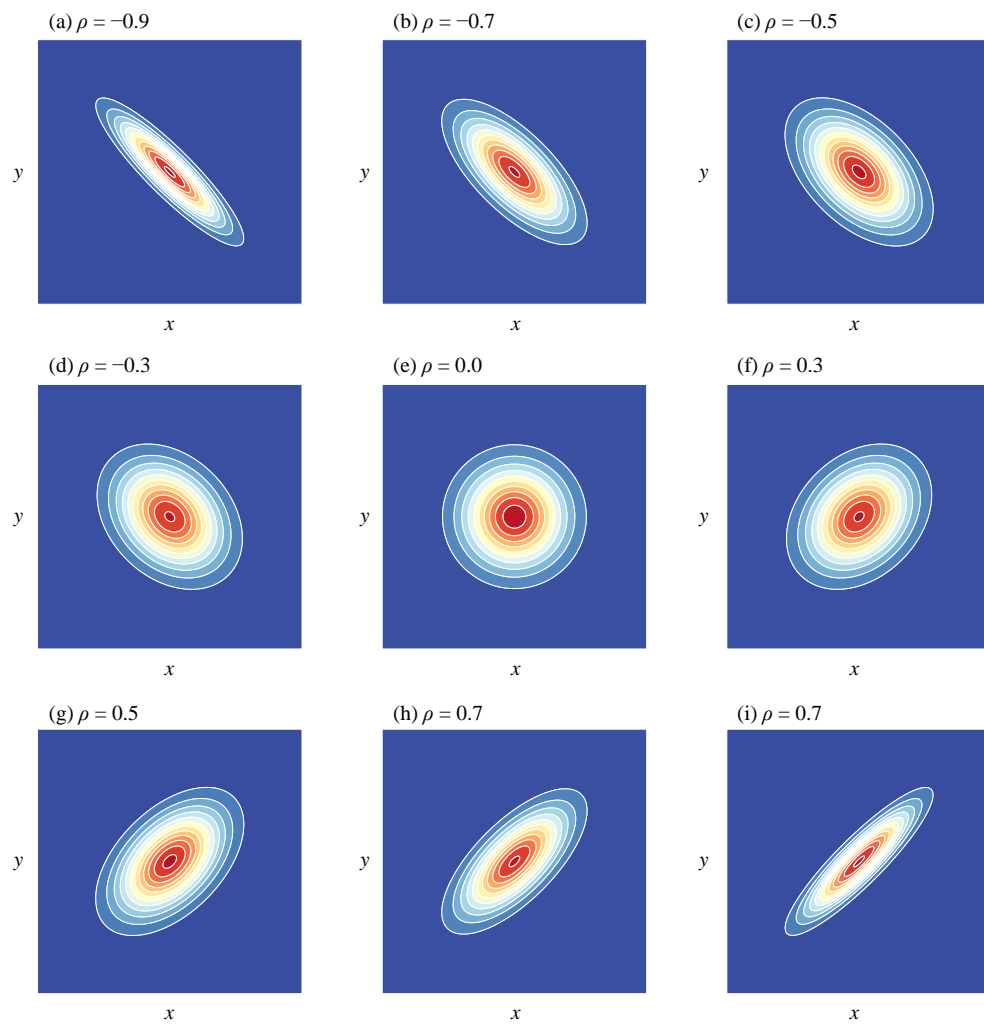


图 16. 二元高斯分布 PDF，平面填充等高线

图 17 代码绘制图 16，请大家自行分析其中关键语句，并写出图 16 中每幅子图对应写协方差矩阵的具体值。

```

# 平面填充等高线
a fig = plt.figure(figsize = (8,8))
  for idx, rho_idx in enumerate(rho_array):
      mu = [mu_X, mu_Y]
      Sigma = [[sigma_X**2, sigma_X*sigma_Y*rho_idx],
               [sigma_X*sigma_Y*rho_idx, sigma_Y**2]]

      bi_norm = multivariate_normal(mu, Sigma)
      f_X_Y_joint = bi_norm.pdf(XXYY)

b   ax = fig.add_subplot(3,3,idx+1)

c   ax.contourf(XX, YY, f_X_Y_joint,
               levels = 12, cmap='RdYlBu_r')

      ax.set_xlabel('$x$')
      ax.set_ylabel('$y$')

      ax.set_xlim(-width, width)
      ax.set_ylim(-width, width)
      ax.axis('off')

plt.tight_layout()
fig.savefig('二元高斯分布，等高线.svg', format='svg')
plt.show()

```

图 17. 平面填充等高线可视化二元高斯分布 PDF，代码



请大家完成如下题目。

Q1. 修改图 15 代码中两个随机变量质心位置，并观察概率密度函数曲面和等高线变化。

Q2. 修改图 15 代码中两个随机变量标准差的值，并观察概率密度函数曲面和等高线变化。

\* 题目很基础，本书不给答案。



本章举了三个例子(距离、插值、高斯分布)介绍如何使用 SciPy 库函数。

在计算欧氏距离这个例子中，我们要关注如何用矩阵形式保存大量散点之间的成对距离值。

在插值这个例子中，我们要理解几种常见插值算法的基本思想，以及插值和回归的本质区别。

高斯分布可能是整套鸢尾花书最重要的分布，没有之一。对于一元高斯分布，大家需要理解期望值和标准差如何影响概率密度函数形状。对于二元分布，大家一定要搞清楚协方差矩阵和椭圆形态关系。本书后续还会在机器学习算法中用到高斯分布。

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](mailto:jiang.visualize.ml@gmail.com)