11 2D Visualizations 二维可视化

平面上的散点图、等高线、热图

文明的传播像是星星之火可以燎原；首先是星星之火，然后是闪烁的炬火，最后是燎原烈焰，排山倒海、势不可挡。

The spread of civilization may be likened to a fire; first, a feeble spark, next a flickering flame, then a mighty blaze, ever increasing in speed and power.

—— 尼古拉·特斯拉 (Nikola Tesla) | 发明家、物理学家 | 1856 ~ 1943

- XXXXX
- XXXXX
- XXXXX
- XXXXX
- XXXXX
-

11.1 二维可视化方案

散点、线图、等高线、热图是鸢尾花书最常见的四类平面可视化方案。

- 散点图 (scatter plot)：散点图用于展示两个变量之间的关系，其中每个点的位置表示两个变量的取值。可以通过设置点的颜色、大小、形状等属性来表示其他信息。
- 线图 (line plot)：线图用于展示数据随时间或其他变量而变化的趋势。线图由多个数据点连接而成，通常用于展示连续数据。
- 等高线图 (contour plot)：等高线图用于展示二维数据随着两个变量的变化而变化的趋势。每个数据点的值表示为等高线的高度，从而形成连续的轮廓线。
- 热图 (heatmap)：热图用于展示二维数据的值，其中每个值用颜色表示。热图常用于数据分析中，用于显示数据的热度、趋势等信息。建议使用 Seaborn 库绘制热图。

上一章，我们介绍了如何用 Matplotlib 和 Plotly 绘制线图，本章将主要介绍散点图、平面等高线、热图这三大类可视化方案。

11.2 散点

二维散点图是平面直角坐标系 (也叫笛卡儿坐标系) 中一种用于可视化二维数据分布的图形表示方法。它由一系列离散的数据点组成，其中每个数据点都由两个坐标值。

Matplotlib 中的 scatter() 函数可以用于创建散点图。可以使用 matplotlib.pyplot.scatter() 函数来指定数据点的坐标和其他绘图参数，例如颜色、大小等。请大家自行在 JupyterLab 中实践图 2 给出的代码。

鸢尾花书《数学要素》第 5 章专门讲解笛卡儿坐标系。

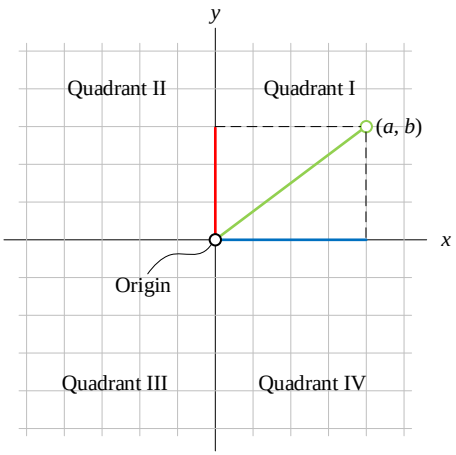

图 1. 笛卡儿坐标系，平面直角坐标系

什么是平面直角坐标系？

平面直角坐标系，也称笛卡儿坐标系，是一种二维空间中的坐标系统，由两条相互垂直的直线组成。其中一条直线称为 x 轴，另一条直线称为 y 轴。它们的交点称为原点，通常用 O 表示。平面直角坐标系可以用来描述二维空间中点的位置，其中每个点都可以由一对有序实数 (a, b) 表示，分别表示点在 a 轴和 b 轴上的距离。x 轴和 y 轴的正方向可以是任意方向，通常 x 轴向右，y 轴向上。平面直角坐标系是解析几何中重要的工具，用于研究点、直线、曲线以及它们之间的关系和性质。

Matplotlib绘制散点图

```python
# 导入包
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import numpy as np

# 加载鸢尾花数据集
iris = load_iris()

# 提取花萼长度和花萼宽度作为变量
sepal_length = iris.data[:, 0]
sepal_width = iris.data[:, 1]
target = iris.target

fig, ax = plt.subplots()

# 创建散点图
plt.scatter(sepal_length, sepal_width, c=target,
cmap='rainbow')

# 添加标题和轴标签
plt.title('Iris sepal length vs width')
plt.xlabel('Sepal length (cm)')
plt.ylabel('Sepal width (cm)')

# 设置横纵轴刻度
ax.set_xticks(np.arange(4, 8 + 1, step=1))
ax.set_yticks(np.arange(1, 5 + 1, step=1))

# 设定横纵轴尺度1:1
ax.axis('scaled')

# 增加刻度网格，颜色为浅灰
ax.grid(linestyle='--', linewidth=0.25,
color=[0.7,0.7,0.7])

# 设置横纵轴范围
ax.set_xbound(lower = 4, upper = 8)
ax.set_ybound(lower = 1, upper = 5)

# 显示图形
plt.show()
```

图 2. 用 Matplotlib 绘制散点图

特别推荐大家使用 seaborn.scatterplot() 函数来创建二维散点图，并传递数据点的坐标和其他可选参数。还可以使用 plotly.graph_objects.Scatter() 函数创建可交互的散点图，并指定数据点的坐标、样式等参数。本节下面利用 Seaborn 和 Plotly 这两个库中函数绘制散点图。

Seaborn

图 3 所示为利用 seaborn.scatterplot() 绘制鸢尾花数据集的散点图。这两幅散点图的横轴都是花萼长度，纵轴为花萼宽度。图 3 (b) 用颜色标识鸢尾花类别。

使用 seaborn.scatterplot() 函数的基本语法如下：

```
import seaborn as sns
sns.scatterplot(data=data_frame, x="x_variable", y="y_variable")
```

其中，`x_variable` 是数据集中表示 x 轴的变量列名，`y_variable` 是表示 y 轴的变量列名，`data_frame` 是包含要绘制的数据的 Pandas DataFrame 对象。

我们还可以指定 `hue` 参数，用于对数据点进行分组并在图中用不同颜色表示的列名，`size` 参数指定了数据点的大小根据 `value` 列的值进行缩放。除了 `hue` 和 `size`，还可以使用其他参数如 `style`、`palette`、`alpha` 等来进一步定制散点图的外观和风格。

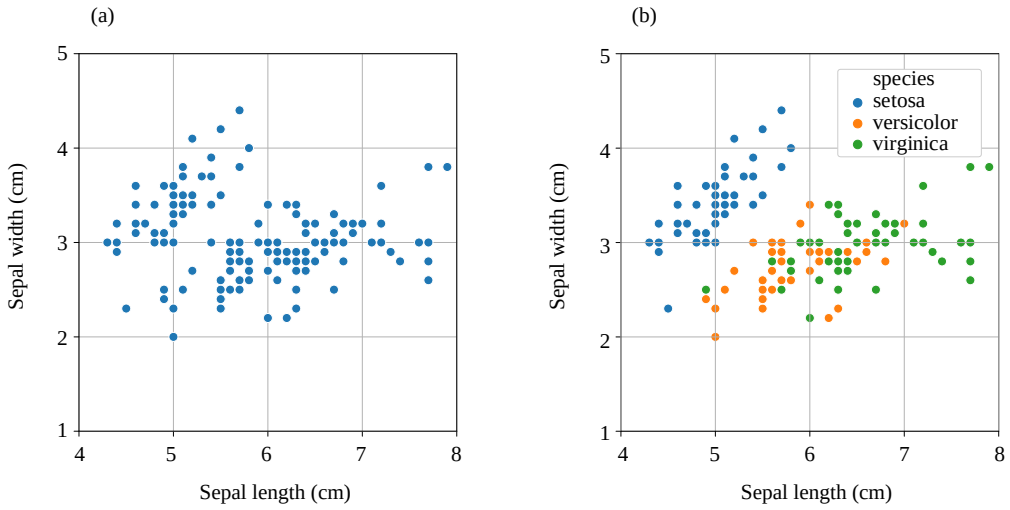

图 3. 使用 seaborn.scatterplot() 绘制鸢尾花数据集散点图

Plotly

图 4 所示为使用 `plotly.express.scatter()` 绘制鸢尾花数据集散点图。在本章配套的 Jupyter Notebook 中大家可以看到这两幅子图为可交互图像。

`plotly.express.scatter()` 用来可视化两个数值变量之间的关系，或者展示数据集中的模式和趋势。这个函数的基本语法如下：

```
import plotly.express as px
```

```
fig = px.scatter(data_frame, x="x_variable", y="y_variable")
fig.show()
```

其中，data_frame 是包含要绘制的数据的 Pandas DataFrame 对象，x_variable 是数据集中表示 *x* 轴的变量列名，y_variable 是表示 *y* 轴的变量列名。可以根据需要添加其他参数，例如 color、size、symbol 等，以进一步定制散点图的外观。

最后，通过 fig.show() 方法显示绘制好的散点图。

《可视之美》将专门讲解散点图。

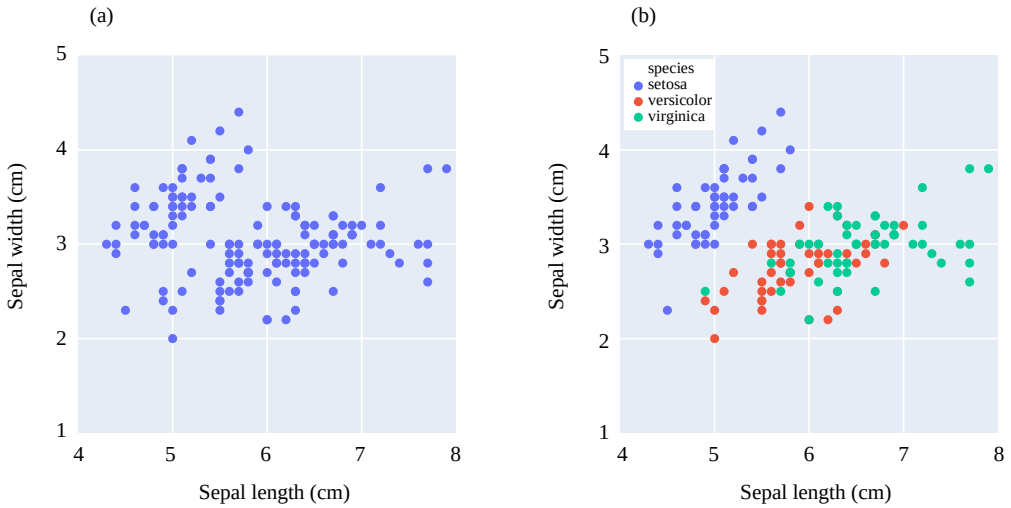

图 4. 使用 plotly.express.scatter() 绘制鸢尾花数据集散点图

11.3 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 5 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。

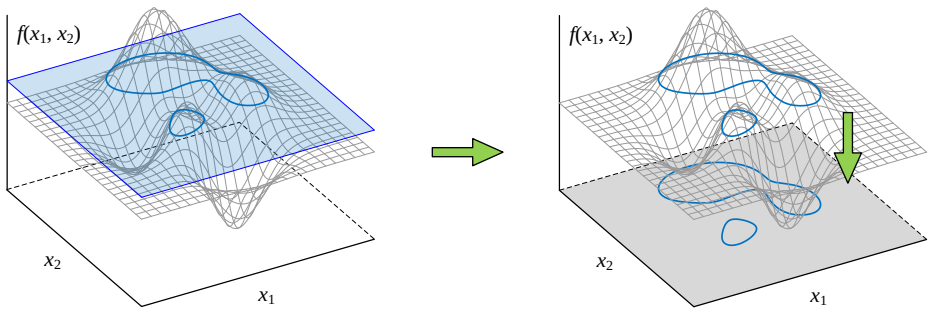

图 5. 平行 x_1x_2 平面切 $f(x_1, x_2)$ 获得等高线，然后等高线投影到 x_1x_2 平面

什么是二元函数？

二元函数是指具有两个自变量和一个因变量的函数。它接受两个输入，并返回一个输出。一般表示为 $y = f(x_1, x_2)$，其中 x_1 和 x_2 是自变量，y 是因变量。二元函数常用于描述和分析具有两个相关变量之间关系的数学模型。它可以用于表示二维空间中的曲面、表达物理或经济关系、进行数据建模和预测等。在可视化二元函数时，常使用三维图形或等高线图。三维图形以 x_1 和 x_2 作为坐标轴，将因变量 y 的值映射为曲面的高度。等高线图则使用等高线来表示 y 值的等值线，轮廓线的密集程度反映了函数值的变化。

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 6 所示，将一组不同高度的等高线投影到平面便得到右侧平面等高线。右侧子图还增加了色谱条，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。

Matplotlib 的填充等高线是在普通等高线的基础上添加填充颜色来表示不同区域的数据密度。可以使用 contourf() 函数来绘制填充等高线。

图 6 左图则是三维等高线，这是下一章要介绍的内容。

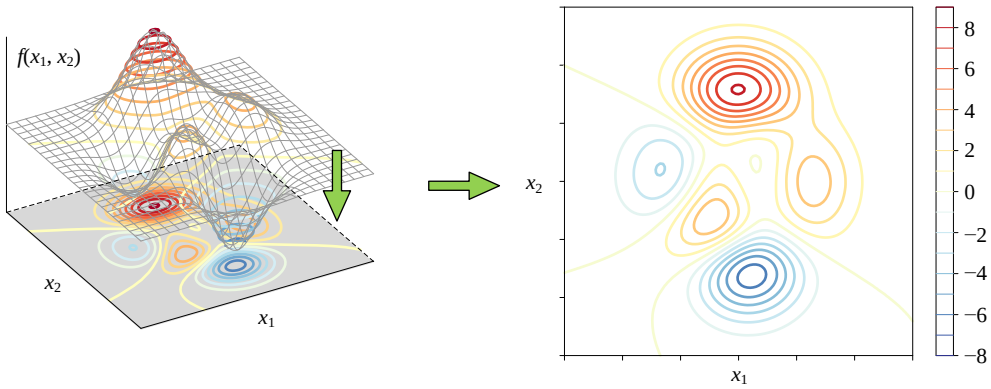

图 6. 将不同高度 (值) 对应的一组等高线投影到 x_1x_2 平面

网格数据

为了绘制平面等高线，我们需要利用 `numpy.meshgrid()` 产生网格数据。`numpy.meshgrid()` 接受一维数组作为输入，并生成二维、三维乃至多维数组来表示网格坐标。

原理上，如图 7 所示，`numpy.meshgrid()` 函数会将输入的一维数 (`x1_array` 和 `x2_array`) 组扩展为二维数组 (`xx1` 和 `xx2`)，其中一个数组的每一行都是输入数组的复制，而另一个数组的每一列都是输入数组的复制。这样，通过组合这两个二维数组的元素，就形成了一个二维网格。

`xx1, xx2 = numpy.meshgrid(x1, x2)`

提供两个一维数组 `x1` 和 `x2` 作为输入。函数将生成两个二维数组 `xx1` 和 `xx2`，用于表示一个二维网格。

请大家在 JupyterLab 中自行学习下例。

```
import numpy as np

x1 = np.arange(10)
# 第一个一维数组
x2 = np.arange(5)
# 第二个一维数组

xx1, xx2 = np.meshgrid(x1, x2)
```

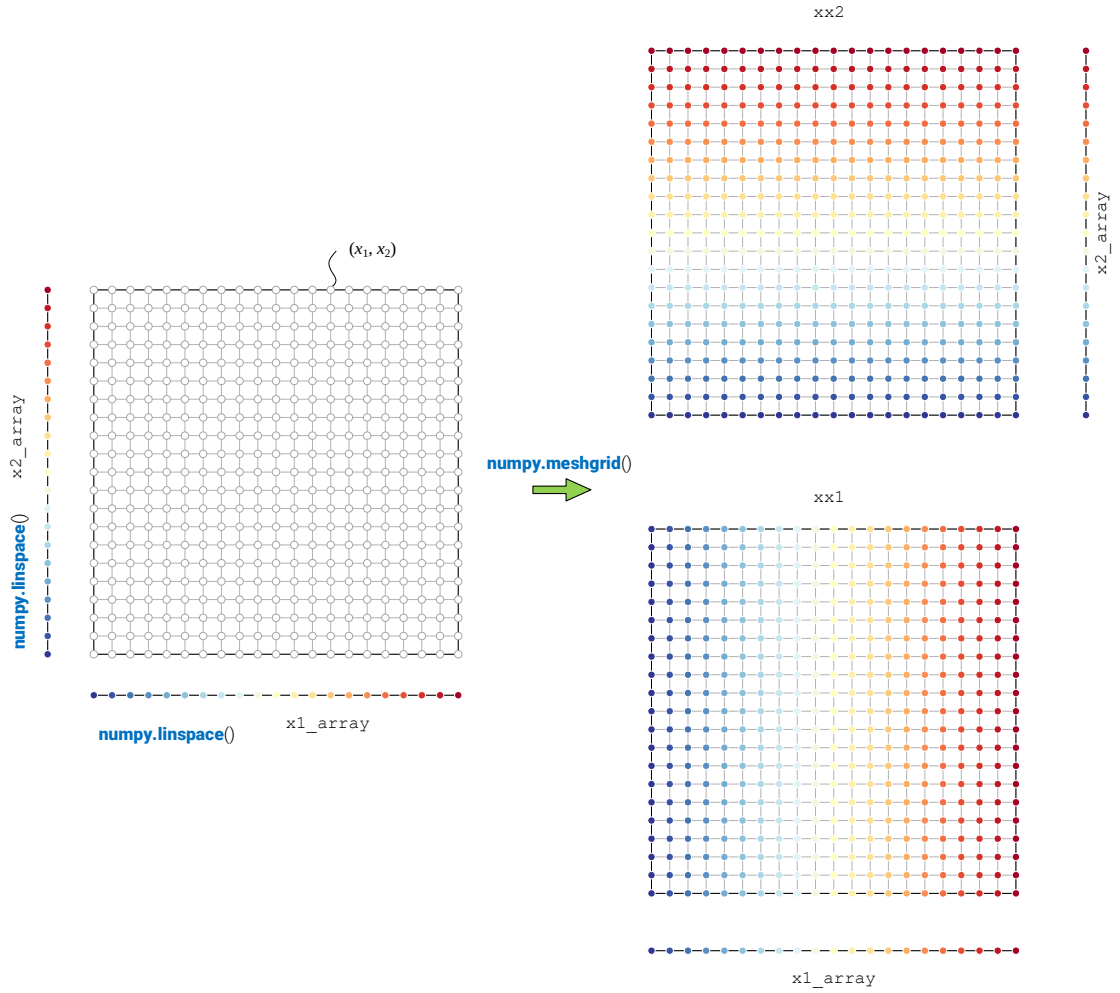

图 7. 用 numpy.meshgrid() 生成二维网络数据

Matplotlib

图 8 所示为利用 Matplotlib 中等高线可视化二元函数 $f(x_1, x_2) = x_1 \exp(-x_1^2 - x_2^2)$。填充等高线的原理是通过在等高线之间创建颜色渐变来表示不同区域的数值范围。这样可以增强等高线图的可视化效果，更直观地展示数据的分布和变化。

在 Matplotlib 中，填充等高线可以通过使用 contourf() 函数实现。该函数与 contour() 函数类似，但会填充等高线之间的区域。

`matplotlib.pyplot.contour(X,Y,Z,levels,cmap)`

下面是 `contour()` 函数的常用输入参数：

- `X`：二维数组，表示数据点的横坐标。
- `Y`：二维数组，表示数据点的纵坐标。
- `Z`：二维数组，表示数据点对应的函数值或高度。
- `levels`：用于指定绘制的等高线层级或数值列表。
- `colors`：用于指定等高线的颜色，可以是单个颜色字符串、颜色序列或 `colormap` 对象。
- `cmap`：颜色映射，用于将数值映射为颜色。可以是预定义的 `colormap` 名称或 `colormap` 对象。
- `linestyles`：用于指定等高线的线型，可以是单个线型字符串或线型序列。
- `linewidths`：用于指定等高线的线宽，可以是单个线宽值或线宽序列。
- `alpha`：用于指定等高线的透明度。

请大家在 JupyterLab 中自行学习下例。

```
import matplotlib.pyplot as plt
import numpy as np

# 创建二维数据
x = np.linspace(-2, 2, 100)
y = np.linspace(-2, 2, 100)
X, Y = np.meshgrid(x, y)
Z = X**2 + Y**2  # 示例函数，可以根据需要自定义

# 绘制等高线图
plt.contour(X, Y, Z, levels = np.linspace(0,8,16 + 1), cmap = 'RdYlBu_r')

# 添加颜色图例
plt.colorbar()

# 显示图形
plt.show()
```

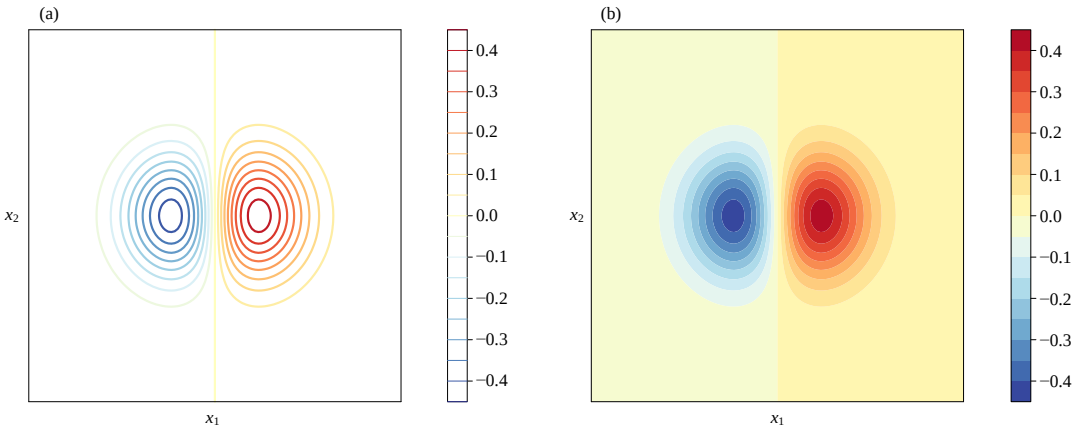

图 8. 用 Matplotlib 生成的平面 (填充) 等高线

Plotly

图 9 所示为利用 plotly.graph_objects.Contour() 绘制的 (填充) 等高线。

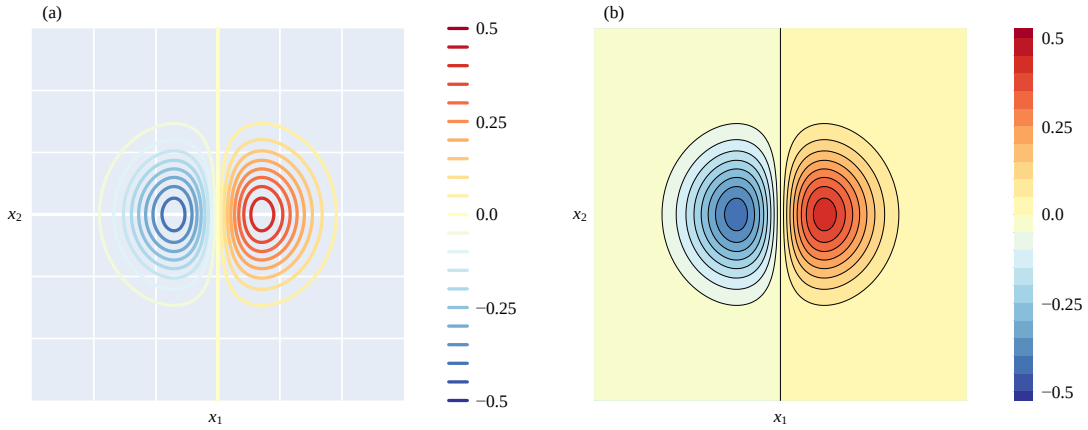

图 9. 用 Plotly 生成的平面 (填充) 等高线

11.4 热图

在 Matplotlib 中，可以使用 matplotlib.pyplot.imshow() 函数来绘制热图 (heatmap)，也叫热力图。imshow() 函数可以将二维数据矩阵的值映射为不同的颜色，从而可视化数据的密度、分布或模式。

鸢尾花书中一般会用 Seaborn 绘制静态热图，特别是在可视化矩阵运算。

```
seaborn.heatmap(data, vmin, vmax, cmap, annot)
```

下面是函数的常用输入参数：

- data：二维数据数组，要绘制的热图数据。
- vmin：可选参数，指定热图颜色映射的最小值。
- vmax：可选参数，指定热图颜色映射的最大值。
- cmap：可选参数，指定热图的颜色映射。可以是预定义的颜色映射名称或 colormap 对象。
- annot：可选参数，控制是否在热图上显示数据值。默认为 False，不显示数据值；设为 True 则显示数据值。
- xticklabels：可选参数，控制是否显示 X 轴的刻度标签。可以是布尔值或标签列表。
- yticklabels：可选参数，控制是否显示 Y 轴的刻度标签。可以是布尔值或标签列表。

请大家在 JupyterLab 中自行学习下例。

```
import seaborn as sns
import numpy as np

# 创建二维数据
data = np.random.rand(10,10)

# 绘制热图
sns.heatmap(data, vmin=0, vmax=1,
            cmap='viridis',
            annot=True,
            xticklabels=True,
            yticklabels=True)
```

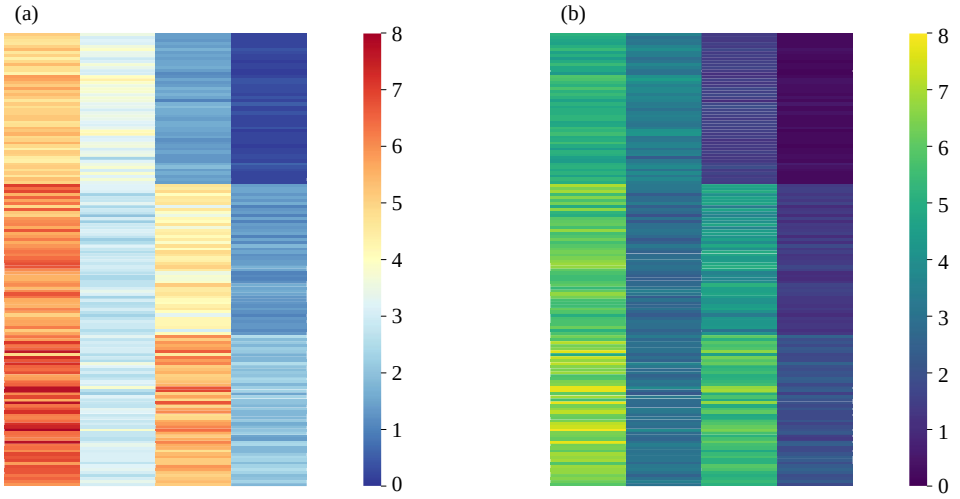

图 10. 使用 Seaborn、Plotly 热图可视化鸢尾花数据集

请大家完成下面 2 道题目。

Q1. 分别用 Matplotlib、Seaborn、Plotly 绘制鸢尾花数据集，花瓣长度、宽度散点图，并适当美化图像。

Q2. 分别用 Matplotlib 和 Plotly 绘制如下二元函数等高线图，并用语言描述图像特点 (等高线形状、疏密分布、增减、最大值、最小值等等)。

$$f(x_1,x_2)=x_1$$
$$f(x_1,x_2)=x_2$$
$$f(x_1,x_2)=x_1+x_2$$
$$f(x_1,x_2)=x_1-x_2$$
$$f(x_1,x_2)=x_1^2+x_2^2$$
$$f(x_1,x_2)=-x_1^2-x_2^2$$
$$f(x_1,x_2)=x_1^2+x_2^2+x_1x_2$$
$$f(x_1,x_2)=x_1^2-x_2^2$$
$$f(x_1,x_2)=x_1^2$$
$$f(x_1,x_2)=x_1x_2$$
$$f(x_1,x_2)=x_1^2+x_2^2+2x_1x_2$$

* 本章不提供答案。