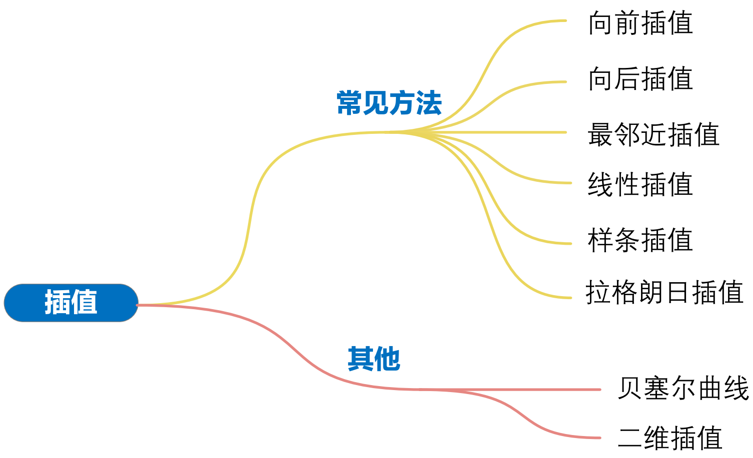
# 9 插值

Interpolation

分段插值函数，通过已知数据点

> 人们思考皆，浮皮潦草，泛泛而谈；现实世界却，盘根错节，千头万绪。
>
> ***We think in generalities, but we live in details.***
>
> —— 阿尔弗雷德·怀特海 (Alfred Whitehead) | 英国数学家、哲学家 | 1861 ~ 1947

- ◄ `scipy.interpolate.interp1d()` 一维插值
- ◄ `scipy.interpolate.lagrange()` 拉格朗日多项式插值
- ◄ `scipy.interpolate.interp2d()` 二维插值，网格化数据
- ◄ `matplotlib.pyplot.pcolormesh()` 绘制填充颜色网格数据
- ◄ `scipy.interpolate.griddata()` 二维插值，散点化数据
- ◄ `matplotlib.pyplot.imshow()` 绘制数据平面图像


插值
常见方法
向前插值
向后插值
最邻近插值
线性插值
样条插值
拉格朗日插值
其他
贝塞尔曲线
二维插值

# 9.1 插值

插值根据有限的数据点，推断其他点处的近似值。给定如图 1 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。

已知点数据范围内的插值叫做内插 (interpolation)。已知点数据外的插值叫做外插 (extrapolation)。

此外，《可视之美》介绍的贝塞尔曲线 (Bézier curve) 本质上也是一种插值。

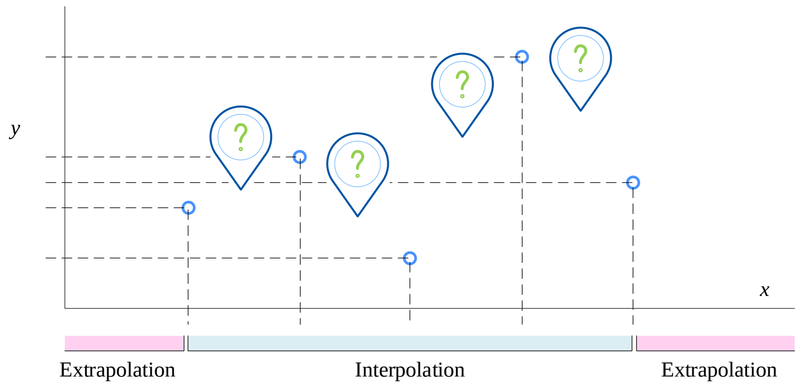



图 1. 插值的意义

## 常见插值方法

图 2 总结常用的插值的算法。本章主要介绍如下几种方法：

- **常数插值** (constant interpolation)，比如**向前** (previous 或 forward)、**向后** (next 或 backward)、**最邻近** (nearest)；
- **线性插值** (linear interpolation)；
- **二次插值** (quadratic interpolation)，本章不做介绍；
- **三次插值** (cubic interpolation)；
- **拉格朗日插值** (Lagrange polynomial interpolation)。

本章最后还要介绍**二维插值** (bivariate interpolation)，二维插值将一元插值的方法推广到二维。

此外，对于时间序列，处理缺失值或者获得颗粒度更高的数据，都可以使用插值。图 3 所示为利用线性插值插补时间序列数据中的缺失值。

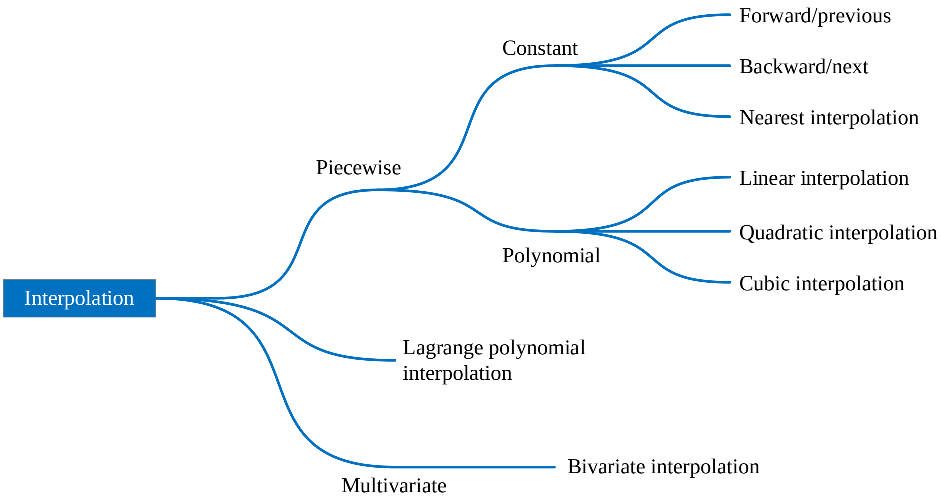



图 2. 插值的分类

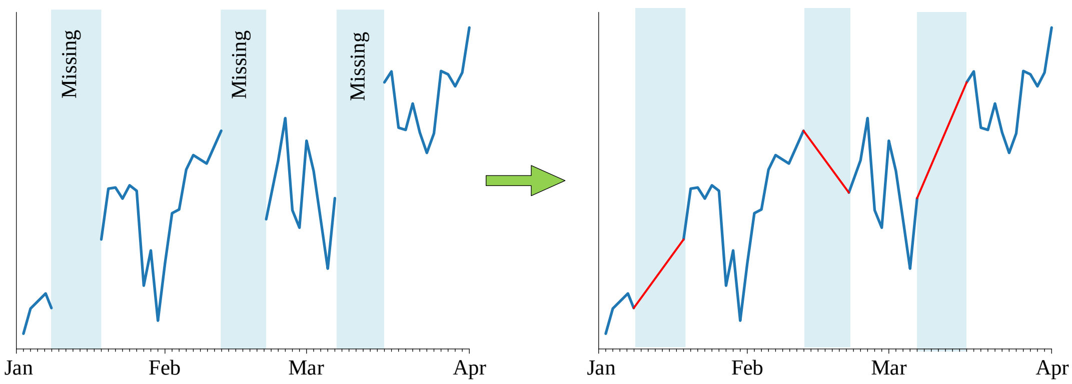



图 3. 时间序列插值

## 分段函数

虽然，一些插值分段函数构造得到的曲线整体看上去平滑。但是绝大多数情况，插值函数是分段函数，因此插值也称分段插值 (piecewise interpolation)。

《数学要素》第 11 章介绍过分段函数。对于一元函数 $f(x)$，分段函数是指自变量 $x$ 在不同取值范围对应不同解析式的函数。

每两个相邻的数据点之间便对应不同解析式：

$$f(x)=\begin{cases} f_1(x) & x^{(1)} \le x < x^{(2)} \\ f_2(x) & x^{(2)} \le x < x^{(3)} \\ \cdots & \cdots \\ f_{n-1}(x) & x^{(n-1)} \le x < x^{(n)} \end{cases} \tag{1}$$

其中，$n$ 为已知点个数。注意，上式中 $f_i(x)$ 代表一个特定解析式。分段函数虽然由一系列解析式构成，但是分段函数还是一个函数，而不是几个函数。

如图 4 所示，已知数据点一共有五个——$(x^{(1)}, y^{(1)})$、$(x^{(2)}, y^{(2)})$、$(x^{(3)}, y^{(3)})$、$(x^{(4)}, y^{(4)})$、$(x^{(5)}, y^{(5)})$。比如，分段函数 $f(x)$ 在 $[x^{(1)}, x^{(2)}]$ 区间的解析式为 $f_1(x)$。$f_1(x)$ 通过 $(x^{(1)}, y^{(1)})$、$(x^{(2)}, y^{(2)})$ 两个已知数据点。图 4 实际上就是线性插值。

(1) 还告诉我们，对于内插，$n$ 个已知点可以构成 $n-1$ 个区间，即分段函数有 $n-1$ 个解析式。

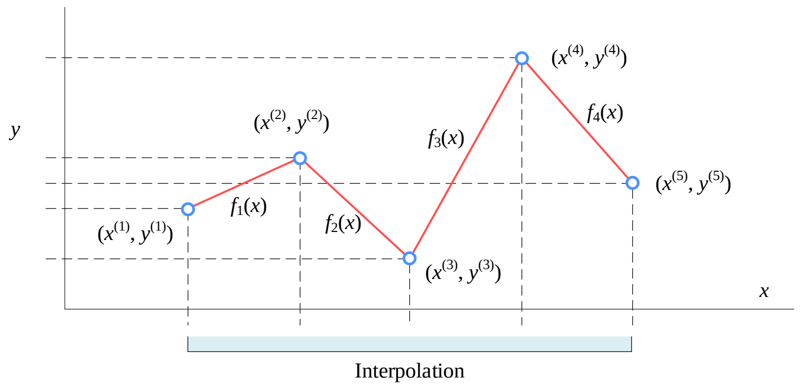



图 4. 分段函数

## 拟合、插值

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。

插值一般得到分段函数，分段函数通过所有给定的数据点，如图 5 (a)、(b) 所示。

拟合得到的函数一般只有一个解析式，这个函数尽可能靠近样本数据点，如图 5 (c)、(d) 所示。

图 6 比较二维插值和二维回归。

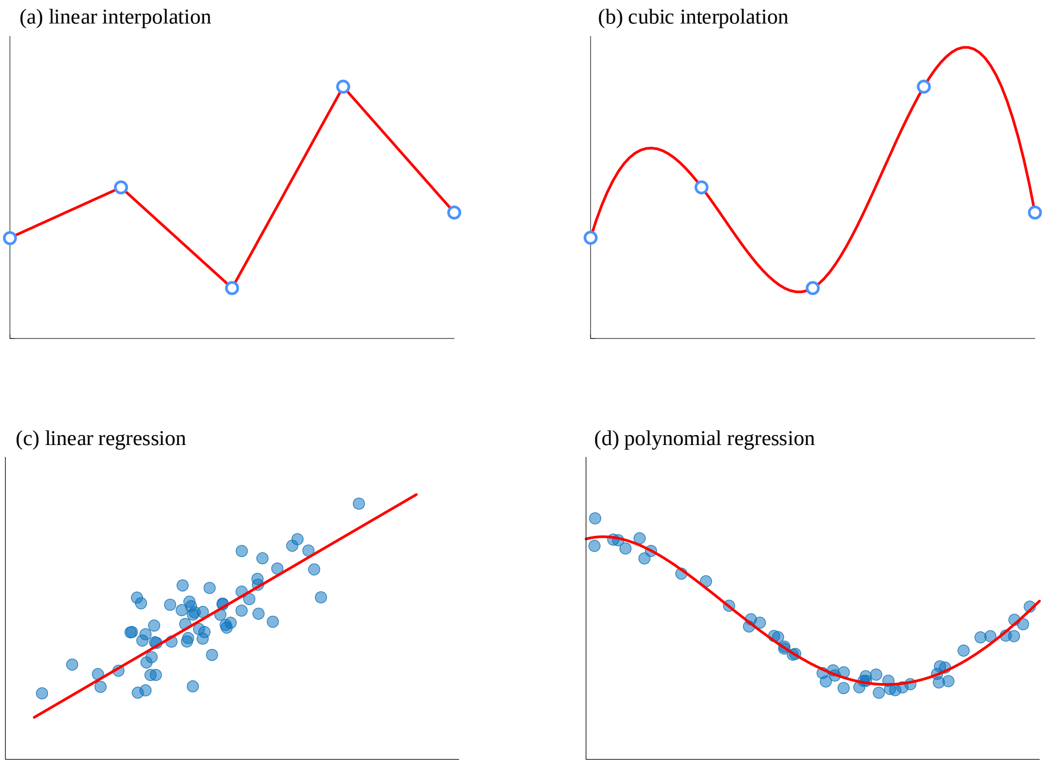



图 5. 比较一维插值和回归

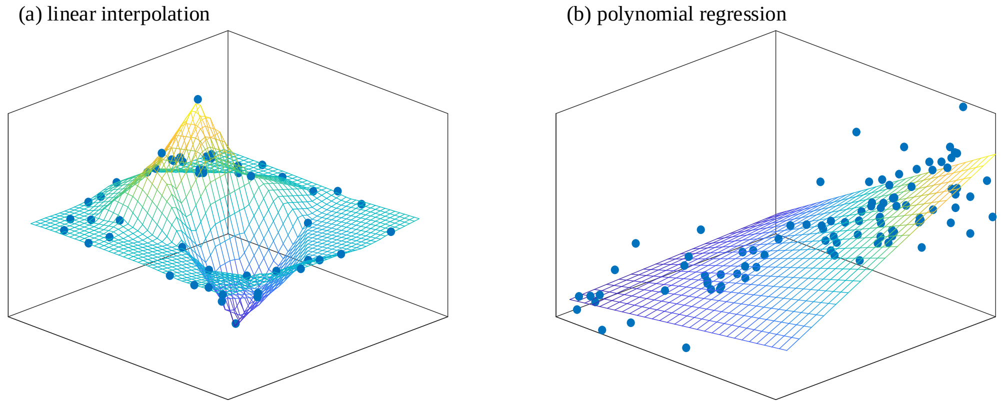



图 6. 比较二维插值和二维回归

# 9.2 常数插值：分段函数为阶梯状

本节介绍常用的三种常数插值方法。

## 向前

向前常数插值对应的分段函数为：

$$f(x)=\begin{cases} f_1(x)=y^{(1)} & x^{(1)}\le x<x^{(2)} \\ f_2(x)=y^{(2)} & x^{(2)}\le x<x^{(3)} \\ \cdots & \cdots \\ f_{n-1}(x)=y^{(n-1)} & x^{(n-1)}\le x<x^{(n)} \end{cases} \tag{2}$$

如图 7 所示，向前常数插值用区间 $[x^{(i)}, x^{(i+1)}]$ 左侧端点，即 $x^{(i)}$，对应的 $y^{(i)}$，作为常数函数的取值。图 7 中红色划线为真实函数取值。

对于数据帧 df，如果存在 NaN 的话，df.fillna(method = 'ffill') 便对应向前常数插补。

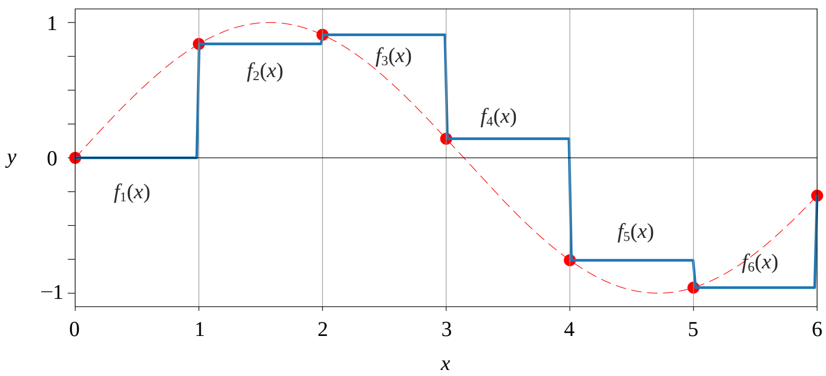



图 7. 向前常数插值

## 向后

向后常数插值对应的分段函数为：

$$f(x)=\begin{cases} f_1(x)=y^{(2)} & x^{(1)}\le x<x^{(2)} \\ f_2(x)=y^{(3)} & x^{(2)}\le x<x^{(3)} \\ \cdots & \cdots \\ f_{n-1}(x)=y^{(n)} & x^{(n-1)}\le x<x^{(n)} \end{cases} \tag{3}$$

如图 8 所示，向后常数插值和图 7 正好相反。

对于数据帧 df，如果存在 NaN 的话，df.fillna(method = 'bfill') 对应向后常数插补。

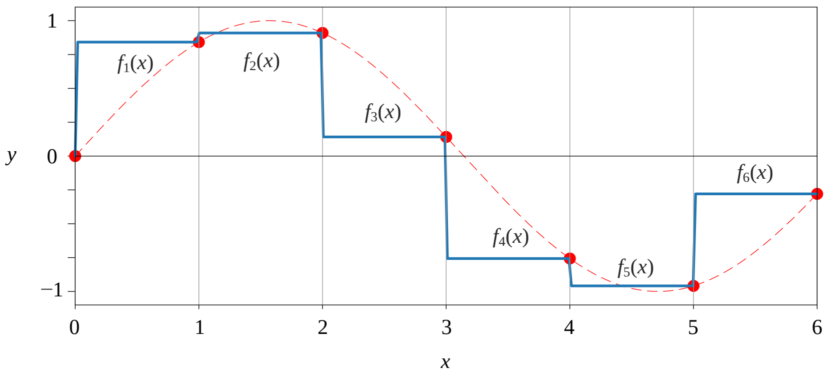



图 8. 向后常数插值

### 最邻近

最邻近插值的分段函数为：

$$f(x)=\begin{cases} f_1(x)=y^{(1)} & x^{(1)}\le x<\dfrac{x^{(1)}+x^{(2)}}{2} \\ f_2(x)=y^{(2)} & \dfrac{x^{(1)}+x^{(2)}}{2}\le x<\dfrac{x^{(2)}+x^{(3)}}{2} \\ \cdots & \cdots \\ f_n(x)=y^{(n)} & \dfrac{x^{(n-1)}+x^{(n)}}{2}\le x<x^{(n)} \end{cases} \tag{4}$$

如图 9 所示，最邻近常数插值相当于“向前”和“向后”常数插值的“折中”。分段插值函数同样是阶梯状，只不过阶梯发生在两个相邻已知点中间处。

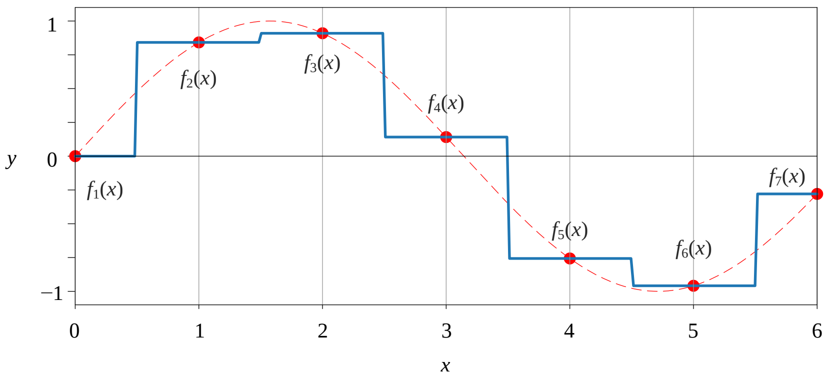



图 9. 最邻近常数插值

## 9.3 线性插值：分段函数为线段

对于线性插值，区间 $[x^{(i)}, x^{(i+1)}]$ 对应的解析式 $f_i(x)$ 为：

$$f_i(x) = \underbrace{\left(\frac{y^{(i)} - y^{(i+1)}}{x^{(i)} - x^{(i+1)}}\right)}_{\text{slope}} \left(x - x^{(i+1)}\right) + y^{(i+1)} \tag{5}$$

容易发现，上式就是《数学要素》第 11 章介绍的一元函数的点斜式。

也就是说，不考虑区间的话，上式代表通过 $(x^{(i)}, y^{(i)})$、$(x^{(i+1)}, y^{(i+1)})$ 两点的一条直线。

图 10 所示为线性插值结果。白话说，线性插值就是用任意两个相邻已知点连接成的线段来估算其他未知点的值。

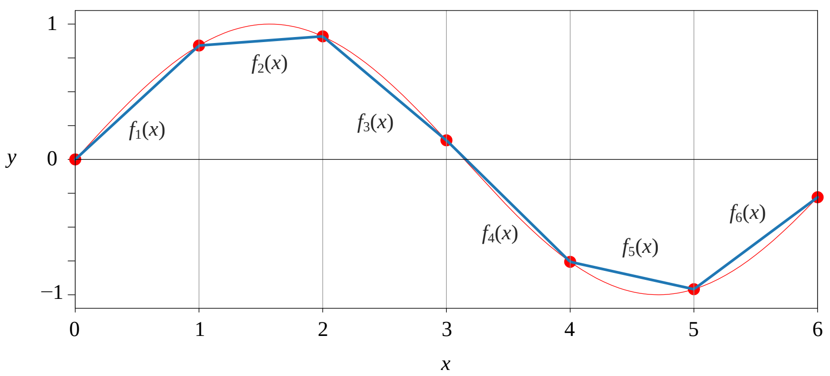



图 10. 线性插值

# 9.4 三次样条插值：光滑曲线拼接

图 11 所示为三次样条插值的结果。虽然，整条曲线看上去连续、光滑，实际上它是由四个函数拼接起来的分段函数。

对于三次样条插值，每一段的分段函数是三次多项式：

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \tag{6}$$

其中，$a_i$、$b_i$、$c_i$、$d_i$ 为需要求解的系数。

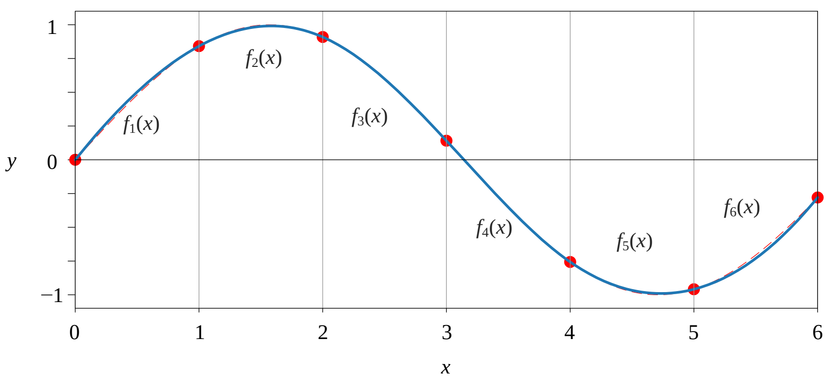


图 11. 三次样条插值

为了求解系数，我们需要构造一系列等式。类似线性插值，每一段三次函数通过区间 $[x^{(i)}, x^{(i+1)}]$ 左右两点，即：

$$\begin{cases} f_i\left(x^{(i)}\right) = y^{(i)} & i = 1, 2, \dots, n-1 \\ f_i\left(x^{(i+1)}\right) = y^{(i+1)} & i = 1, 2, \dots, n-1 \end{cases} \tag{7}$$

曲线之所以看起来很平滑是因为，除两端样本数据点以外，内部数据点处，一阶和二阶导数等值：

$$\begin{cases} f_i'\left(x^{(i+1)}\right) = f_{i+1}'\left(x^{(i+1)}\right) & i = 1, 2, \dots, n-2 \\ f_i''\left(x^{(i+1)}\right) = f_{i+1}''\left(x^{(i+1)}\right) & i = 1, 2, \dots, n-2 \end{cases} \tag{8}$$

对于三次样条插值，一般还设定两端样本数据点处二阶导数为 0：

$$\begin{cases} f_1''\left(x^{(1)}\right) = 0 \\ f_{n-1}''\left(x^{(n)}\right) = 0 \end{cases} \tag{9}$$

Bk6_Ch09_01.ipynb 完成插值并绘制图 7 ~ 图 11。Python 进行一维插值函数为 scipy.interpolate.interp1d()，二维插值的函数为 scipy.interpolate.interp2d()。下面聊聊其中关键语句。

a 从 SciPy 库中导入 interp1d 类，该类用于进行一维插值。
b 定义一个包含不同插值方法的列表。
c 使用 interp1d 类创建插值函数 f_prev，其中 kind 参数指定插值方法。

还有一句值得大家注意，plt.autoscale(enable=True, axis='x', tight=True) 自动调整 x 轴的刻度，使得数据点和曲线完全可见。

```python
# 导入包
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
import numpy as np

# 构造数据
x_known = np.linspace(0, 6, num=7, endpoint=True)
y_known = np.sin(x_known)

x_fine  = np.linspace(0, 6, num=300, endpoint=True)
y_fine  = np.sin(x_fine)

# 不同插值方法
methods = ['previous', 'next', 'nearest', 'linear', 'cubic']

for kind in methods:

    f_prev = interp1d(x_known, y_known, kind = kind)

    fig, axs = plt.subplots()
    plt.plot(x_known, y_known, 'or')
    plt.plot(x_fine,  y_fine,  'r--',  linewidth = 0.25)
    plt.plot(x_fine,  f_prev(x_fine), linewidth = 1.5)

    for xc in x_known:
        plt.axvline(x=xc, color = [0.6, 0.6, 0.6], linewidth = 0.25)

    plt.axhline(y=0, color = 'k', linewidth = 0.25)
    plt.autoscale(enable=True, axis='x', tight=True)
    plt.autoscale(enable=True, axis='y', tight=True)
    plt.xlabel('x'); plt.ylabel('y')
    plt.ylim([-1.1,1.1])
```

代码 1. 几种常见插值方法 | Bk6_Ch09_01.ipynb

# 9.5 拉格朗日插值

拉格朗日插值 (Lagrange interpolation) 不同于本章前文介绍的插值方法。前文介绍的插值方法得到的都是分段函数，而拉格朗日插值得到的是一个高次多项式函数 $f(x)$。$f(x)$ 相当由若干多项式函数叠加而成：

$$f(x)=\sum_{i=1}^{n} f_i(x) \tag{10}$$

其中，

$$f_i(x)=y^{(i)}\cdot\prod_{k=1,k\neq i}^{n}\frac{x-x^{(k)}}{x^{(i)}-x^{(k)}} \tag{11}$$

$f_i(x)$ 展开来写：

$$f_i(x) = y^{(i)} \cdot \frac{\left(x - x^{(1)}\right)\left(x - x^{(2)}\right)\dots\left(x - x^{(i-1)}\right)\left(x - x^{(i+1)}\right)\dots\left(x - x^{(n)}\right)}{\left(x^{(i)} - x^{(1)}\right)\left(x^{(i)} - x^{(2)}\right)\dots\left(x^{(i)} - x^{(i-1)}\right)\left(x^{(i)} - x^{(i+1)}\right)\dots\left(x^{(i)} - x^{(n)}\right)} \tag{12}$$

比如，$f_1(x)$ 展开来写：

$$f_1(x) = y^{(1)} \cdot \frac{\left(x - x^{(2)}\right)\left(x - x^{(3)}\right)\dots\left(x - x^{(n)}\right)}{\left(x^{(1)} - x^{(2)}\right)\left(x^{(1)} - x^{(3)}\right)\dots\left(x^{(1)} - x^{(n)}\right)} \tag{13}$$

$f_2(x)$ 展开来写：

$$f_2(x) = y^{(2)} \cdot \frac{\left(x - x^{(1)}\right)\left(x - x^{(3)}\right)\dots\left(x - x^{(n)}\right)}{\left(x^{(2)} - x^{(1)}\right)\left(x^{(2)} - x^{(3)}\right)\dots\left(x^{(2)} - x^{(n)}\right)} \tag{14}$$

## 举个例子

比如，$n = 3$，也就是有三个样本数据点 $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)})\}$ 的时候，$f(x)$ 为：

$$f(x) = \underbrace{y^{(1)} \cdot \frac{\left(x - x^{(2)}\right)\left(x - x^{(3)}\right)}{\left(x^{(1)} - x^{(2)}\right)\left(x^{(1)} - x^{(3)}\right)}}_{f_1(x)} + \underbrace{y^{(2)} \cdot \frac{\left(x - x^{(1)}\right)\left(x - x^{(3)}\right)}{\left(x^{(2)} - x^{(1)}\right)\left(x^{(2)} - x^{(3)}\right)}}_{f_2(x)} + \underbrace{y^{(3)} \cdot \frac{\left(x - x^{(1)}\right)\left(x - x^{(2)}\right)}{\left(x^{(3)} - x^{(1)}\right)\left(x^{(3)} - x^{(2)}\right)}}_{f_3(x)} \tag{15}$$

观察上式，$f(x)$ 相当于三个二次函数叠加得到。

将三个数据点 $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)})\}$，逐一代入上式，可以得到：

$$f\left(x^{(1)}\right) = y^{(1)},\quad f\left(x^{(2)}\right) = y^{(2)},\quad f\left(x^{(3)}\right) = y^{(3)} \tag{16}$$

也就是说，多项式函数 $f(x)$ 通过给定的已知点。

图 12 所示为拉格朗日插值结果。

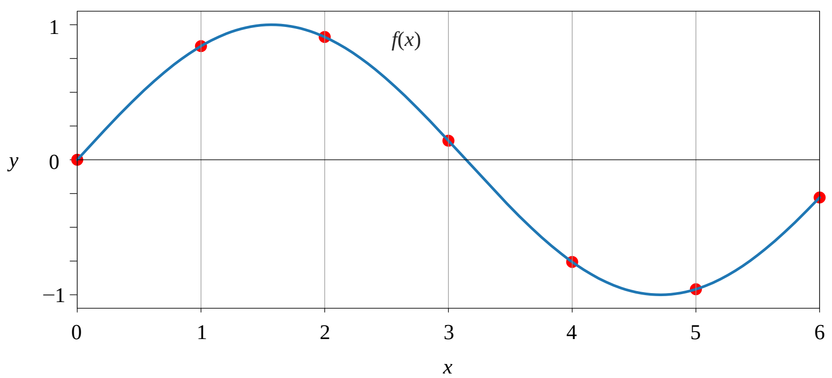



图 12. 拉格朗日插值

## 龙格现象

有一点需要大家注意的是，已知点数量 $n$ 不断增大，拉格朗日插值函数多项式函数次数不断提高，插值多项式的插值逼近效果未必好。如图 13 所示，插值多项式 (红色曲线) 区间边缘处出现振荡问题，这一现象叫做龙格现象 (Runge's phenomenon)。

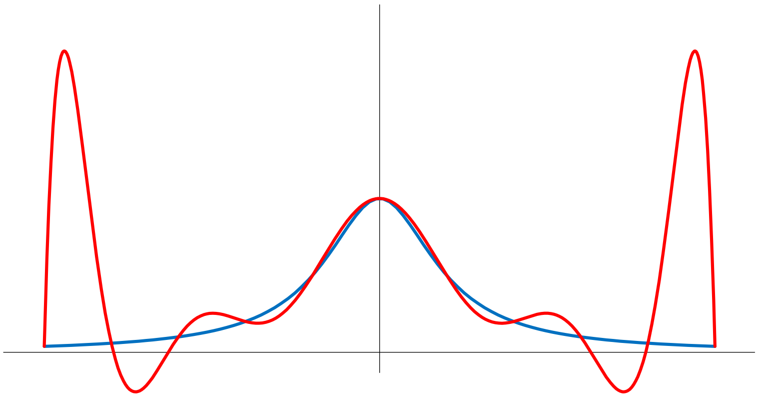


图 13. 龙格现象

| |
|---|
| Bk6_Ch09_02.ipynb 完成拉格朗日插值，并绘制图 12。 |

## 9.6 贝塞尔曲线

《可视之美》介绍过，贝塞尔曲线是一种常用于计算机图形学中的数学曲线。它由法国工程师**皮埃尔·贝塞尔** (Pierre Bézier) 在 19 世纪中叶发明。

本质上来讲，贝塞尔曲线就是一种插值方法。贝塞尔曲线可以是一阶曲线、二阶曲线、三阶曲线等，其阶数决定了曲线的平滑程度。

一阶曲线由两个控制点组成，形成一条直线。如图 14 所示，简单来说一阶贝塞尔曲线就是两点之间连线。图中 $t$ 代表权重，取值范围为 [0, 1]。$t$ 越大，点 $B(t)$ 距离 $P_0$ 越近，如图中暖色 ×，相当于 $P_0$ 对 $B(t)$影响越大。相反，$t$ 越小，点 $B(t)$ 距离 $P_1$ 越近，如图中冷色 ×，相当于 $P_1$ 对 $B(t)$ 影响大。

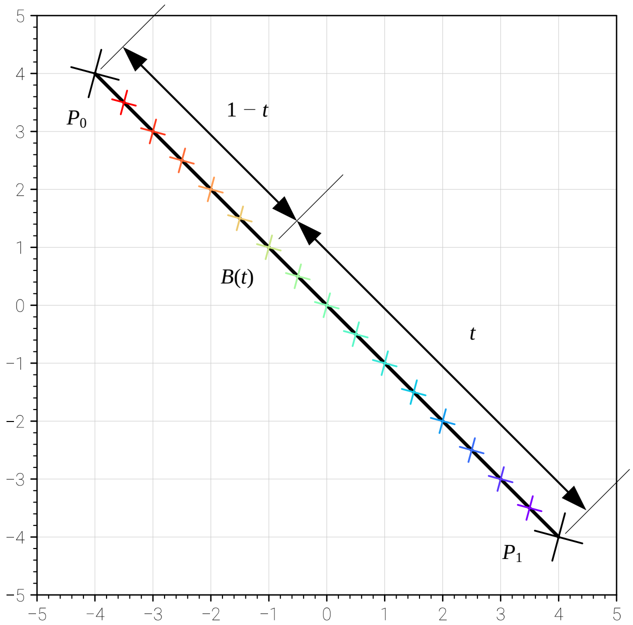


图 14. 一阶贝塞尔曲线原理，图片来自《可视之美》

二阶贝塞尔曲线由三个控制点组成，形成一条弯曲的曲线。如图 15 所示，$P_0$ 和 $P_2$ 点控制了曲线 (黑色线) 的两个端点，而 $P_1$ 则决定的曲线的弯曲行为。实际上图 15 中黑色二阶贝塞尔曲线上的每一个点都经历了两组线性插值得到。

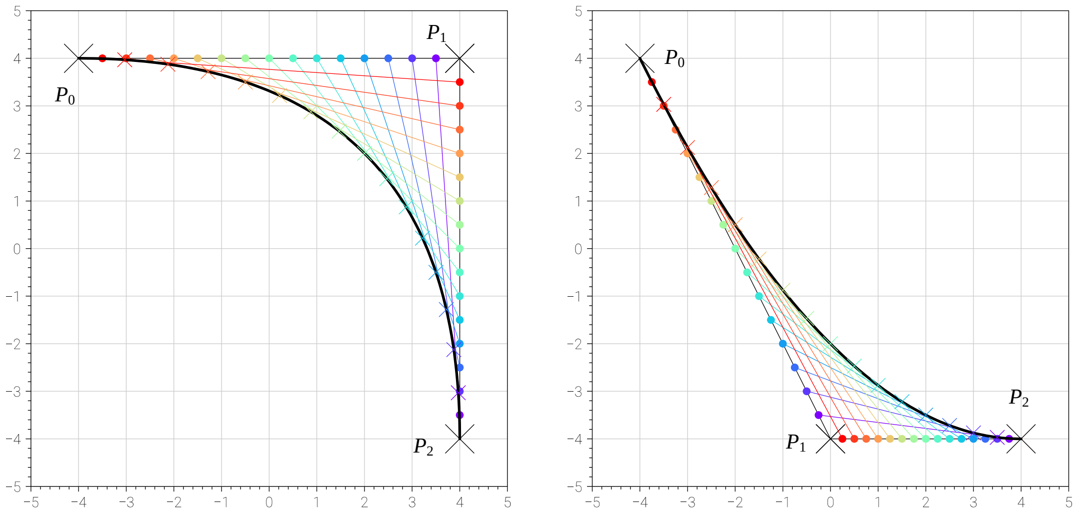


图 15. 二阶贝塞尔曲线原理，图片来自《可视之美》

如图 16 所示，设定 $t = 13/16$，通过第一组线性插值，我们分别得到了 $P_0P_1$ 线段上的 $A_0$，以及 $P_1P_2$ 线段上的 $A_1$。然后通过第二组线性插值，我们便得到 $A_0A_1$ 线段上的 $B(13/16)$。当 $t$ 在 $[0, 1]$ 之间连续取值时，我们便得到了二阶贝塞尔曲线上的一系列点。

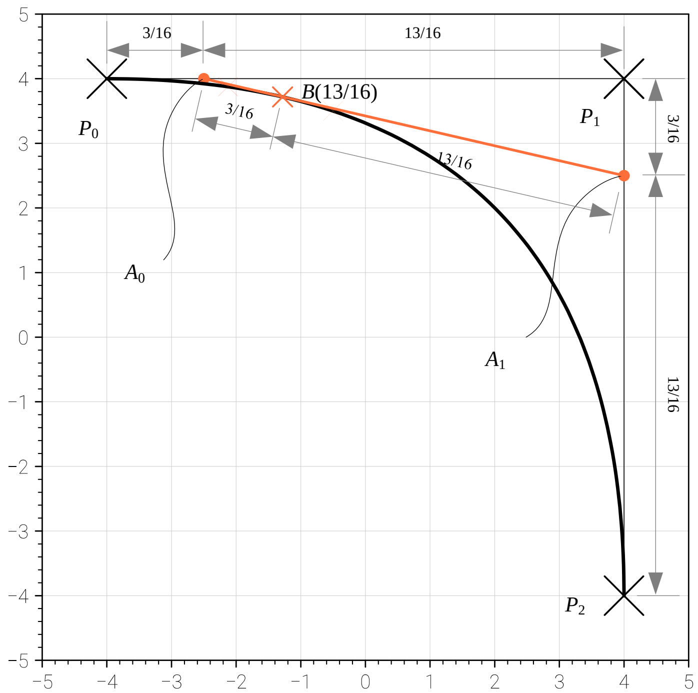



图 16. 二阶贝塞尔曲线原理，以 *B*(3/16) 为例，图片来自《可视之美》

## 9.7 二维插值

如图 17 所示，以二维线性插值为例，二维线性插值相当于处理了三个一维线性插值。

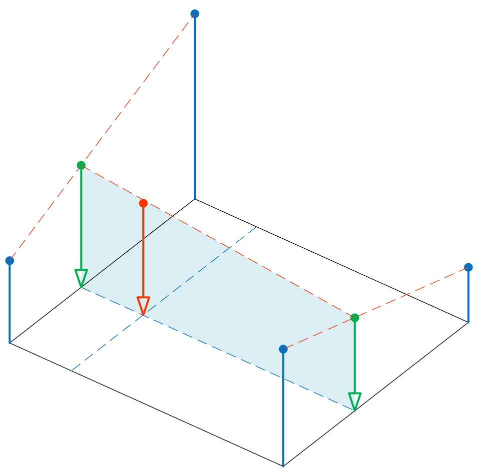


图 17. 二维线性插值原理

### 举个例子

图 18 中 × 为给定的已知数据。图 19 和图 20 所示为分别通过线性插值、三次样条插值完成的二维插值结果。二维插值用到的函数是 scipy.interpolate.interp2d()。

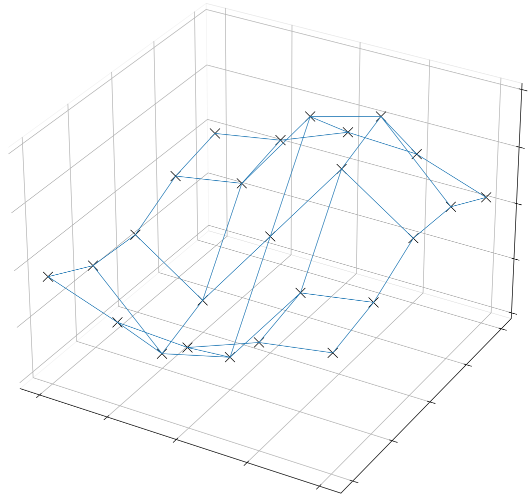


图 18. 已知数据点

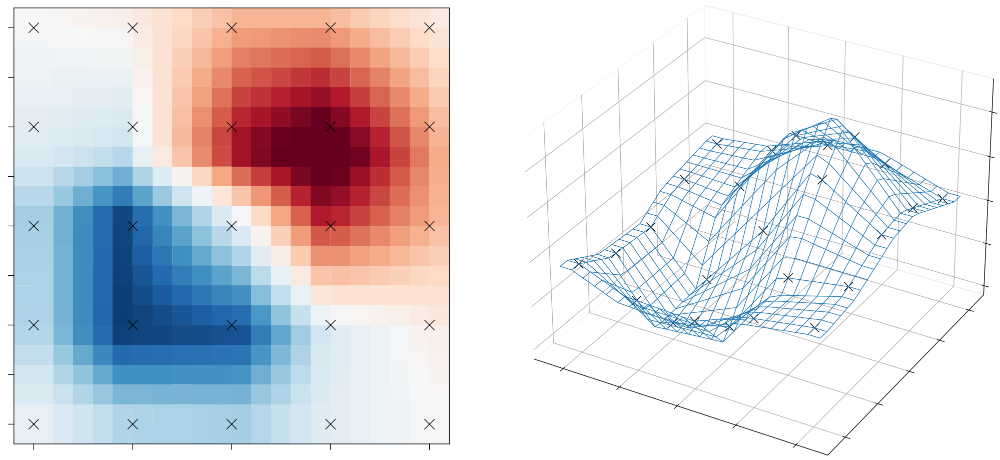


图 19. 二维插值，规则网格，线性插值

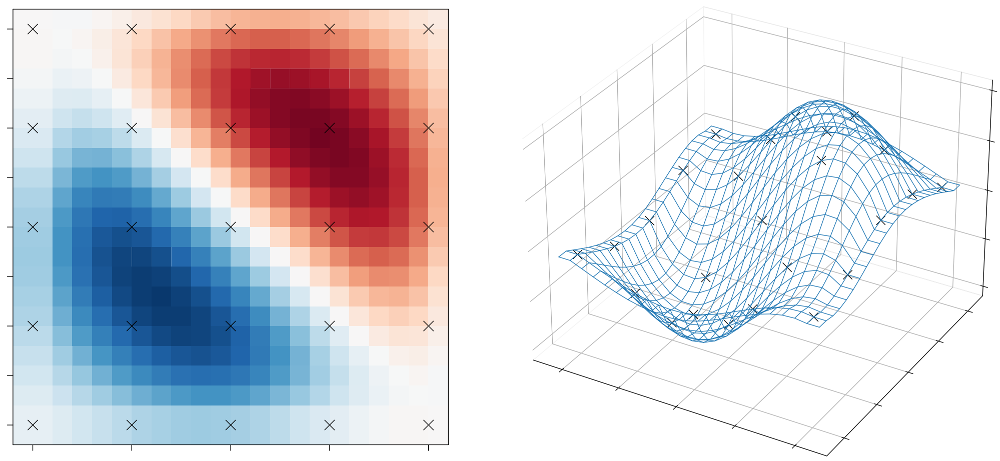


图 20. 二维插值，规则网格，三次样条

Bk6_Ch09_03.ipynb 完成二维插值，并绘制图 19 和图 20。

## 不规则散点

大家可能已经注意到，图 18 给定的已知数据是规整的网格数据。当数据并不是规整的网格数据，而是不规则的散点时，我们也可以用 scipy.interpolate.griddata() 完成二维插值。图 21、图 22、图 23 分别所示为利用最邻近、线性、三次样条方法完成不规则散点的二维插值。

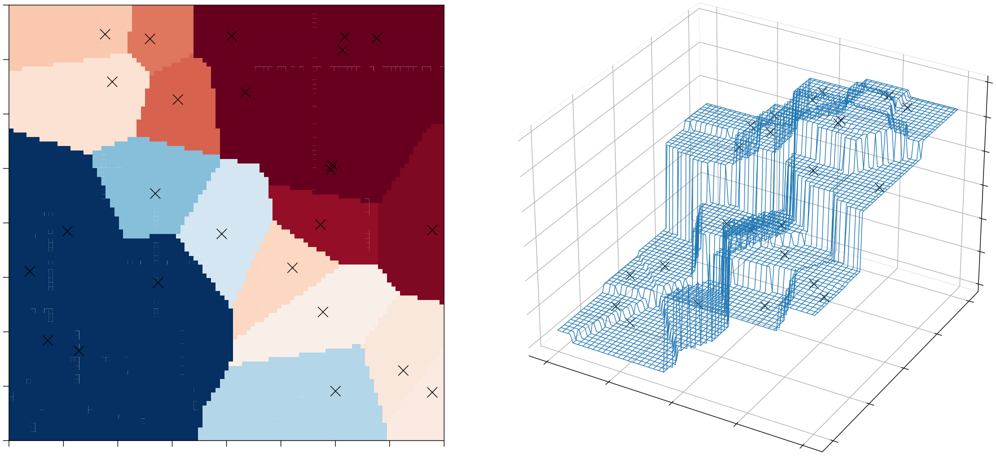


图 21. 二维插值，不规则散点，最近邻

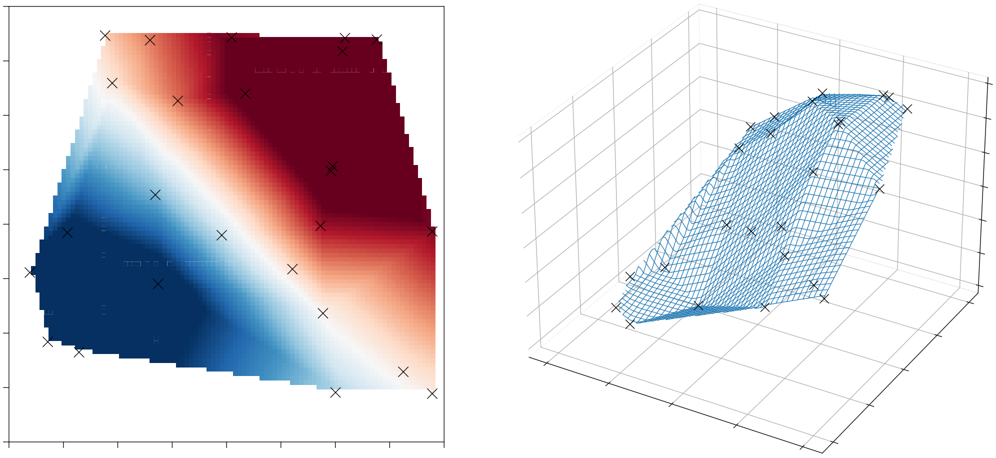


图 22. 二维插值，不规则散点，线性插值

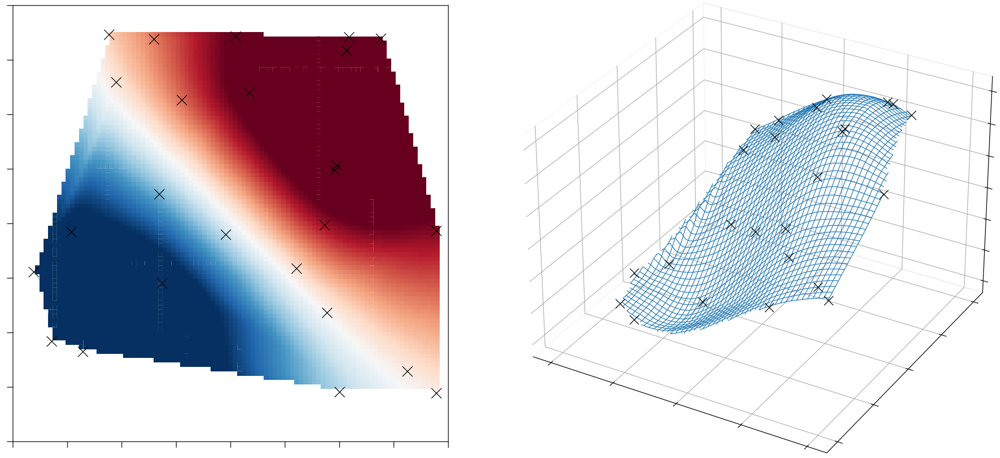


图 23. 二维插值，不规则散点，三次样条插值

Bk6_Ch09_04.ipynb 完成不规则散点插值，并绘制图 21、图 22、图 23。

## 更多插值方法

matplotlib.pyplot.imshow() 绘图函数自带大量二维插值方法，请大家参考图 24。

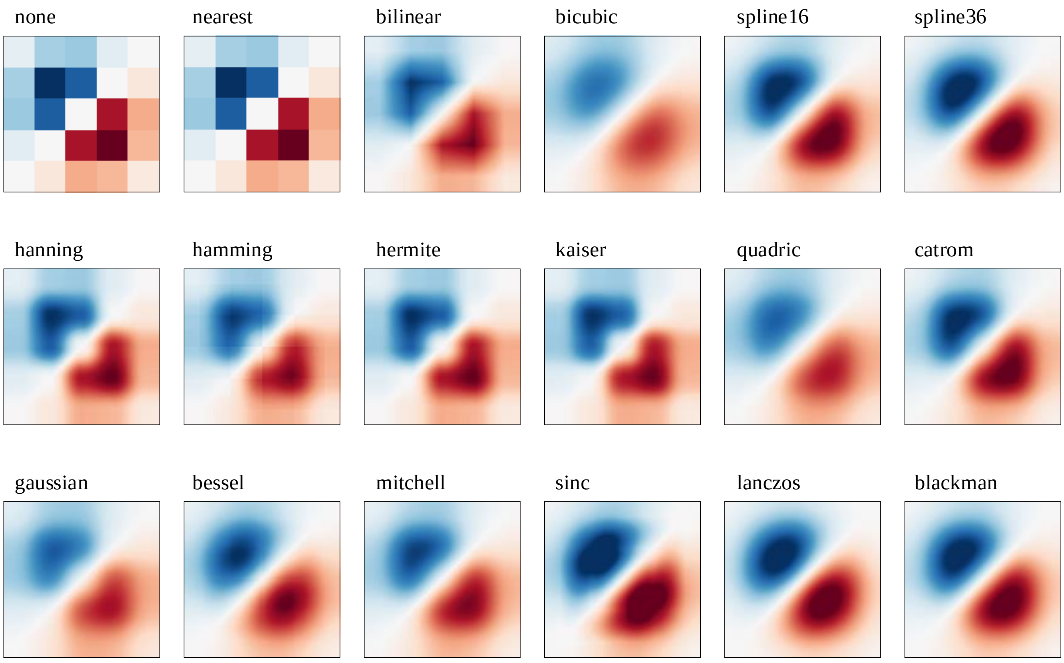



图 24. imshow() 函数插值方法

Bk6_Ch09_05.ipynb 绘制图 24。

插值是一种通过已知数据点的数值推断未知位置的数值的方法。在机器学习中，插值通常用于处理数据集中的缺失值或生成平滑曲线。

一些常用的插值方法包括线性插值、样条插值、拉格朗日插值等等。插值方法的选择取决于数据的性质、插值的目的以及对计算复杂性的要求。在实践中，线性插值通常是最简单和最常用的方法之一，但对于更复杂的情况，其他插值方法可能更适合。