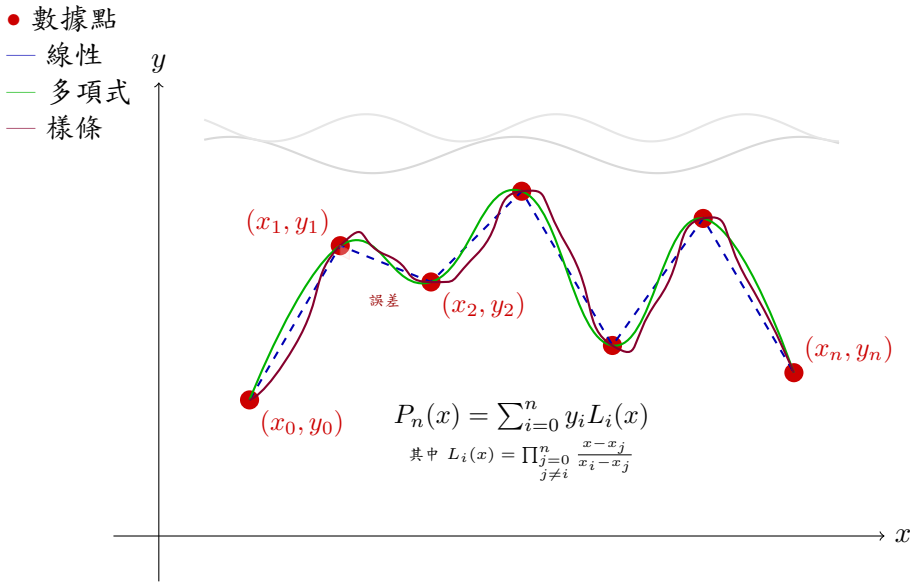
數學建模補充教材

插值與曲線擬合

理論與應用



程淑鍵 Kenneth

最後更新：July 7, 2025

連接離散數據與連續函數

Contents

1 插值 **4**

1.1 引言 . . . 4
1.1.1 數學先備知識 . . . 4
1.1.2 理解插值 . . . 4
1.1.3 歷史發展 . . . 4
1.1.4 插值在數學建模中的作用 . . . 4
1.1.5 激勵性例子 . . . 5
1.2 插值的基本定理 . . . 5
1.2.1 存在性和唯一性 . . . 5
1.2.2 拉格朗日插值 . . . 6
1.2.3 牛頓前向差分法 . . . 6
1.2.4 誤差分析和邊界 . . . 6
1.2.5 龍格現象 . . . 7
1.2.6 切比雪夫插值 . . . 7
1.2.7 樣條插值 . . . 11
1.3 實際實現和應用 . . . 15
1.4 高級插值技術 . . . 16
1.4.1 埃爾米特插值 . . . 16
1.4.2 三角插值 . . . 16
1.4.3 多維插值 . . . 17

2 曲線擬合 **18**

2.1 曲線擬合簡介 . . . 18
2.1.1 區分曲線擬合與插值 . . . 18
2.1.2 曲線擬合的哲學 . . . 18
2.1.3 應用和動機 . . . 18
2.2 線性最小二乘 . . . 19
2.2.1 最小二乘法 . . . 19
2.2.2 多項式曲線擬合 . . . 19
2.2.3 最小二乘的統計性質 . . . 20
2.3 非線性曲線擬合 . . . 20
2.3.1 非線性最小二乘問題 . . . 20
2.3.2 高斯-牛頓算法 . . . 20
2.3.3 模型選擇和驗證 . . . 20
2.4 穩健曲線擬合 . . . 21
2.4.1 最小二乘的限制 . . . 21
2.4.2 穩健損失函數 . . . 21

2.4.3 迭代重加權最小二乘 21
2.5 正則化回歸 22
2.5.1 偏差-方差權衡 22
2.5.2 嶺回歸 22
2.5.3 套索回歸 22
2.6 計算方法和實現 22
2.7 模型選擇和驗證 24
2.7.1 交叉驗證 24
2.7.2 信息準則 24
2.7.3 殘差分析 25
2.8 高級主題和擴展 25
2.8.1 貝葉斯曲線擬合 25
2.8.2 高斯過程回歸 25
2.8.3 機器學習方法 25

Chapter 1

插值

1.1 引言

1.1.1 數學先備知識

開始學習插值理論的學生應該在幾個數學領域具有堅實的基礎。對微積分的深入理解是必要的，特別是導數、泰勒級數展開和收斂概念的知識。線性代數構成另一個重要支柱，包括向量空間、基函數和線性無關的概念。此外，熟悉實分析非常有價值，特別是關於連續性、一致收斂和函數空間。最後，對數值方法的基本掌握，包括近似理論和誤差分析，將增進對插值實際方面的理解。

1.1.2 理解插值

定義 **1.1** (插值)**.** 給定一組數據點 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$，其中 $x_i \neq x_j$ 對於 $i \neq j$，插值是找到一個函數 $f(x)$ 的過程，使得對於所有 $i = 0, 1, \ldots, n$ 都有 $f(x_i) = y_i$。

插值代表數值分析和數學建模中最基本的技術之一。其核心是解決從離散數據點重建連續函數的挑戰。這個過程假設潛在現象可以用平滑、連續的函數來描述，我們的目標是找到一個數學表示，既能捕捉這種連續性，又能精確地符合已知的數據點。

1.1.3 歷史發展

插值理論的演進跨越千年，反映了人類從可觀察數據預測和估計未知值的持續需求。插值的最早表現可以追溯到公元前 2000 年左右的古巴比倫天文學家，他們開發了初步技術來從表格天文觀測預測天體事件。這些早期實踐者認識到自然現象經常遵循可預測的模式，通過理解這些模式，他們可以超越直接觀測進行推斷。

中世紀期間，伊斯蘭數學家顯著推進了插值方法。其中值得注意的是 Al-Biruni (973-1048)，他專門為天文計算開發了系統的插值方法。他的工作為更嚴格的數學處理奠定了重要基礎。文藝復興時期見證了約翰內斯·開普勒創新地應用插值方法分析第谷·布拉赫的精密行星觀測，最終導致他革命性的行星運動定律。

17 和 18 世紀標誌著插值理論的正式數學基礎。艾薩克·牛頓和約瑟夫-路易·拉格朗日獨立開發了以他們命名的多項式插值方法。他們的工作建立了理論框架，至今仍是現代插值理論的核心，提供了實用算法和插值程序的嚴格數學證明。

1.1.4 插值在數學建模中的作用

插值在數學建模和科學計算中發揮多重關鍵功能。也許最根本的是，它通過允許我們估計測量點之間的值來實現數據重建。考慮這樣一個場景：全天每小時記錄溫度測量。插值提供了一種系統方法來估計任何中間時間的溫度，例如在 2:00 PM 和 3:00 PM 進行測量時的 2:30 PM。

實際應用

氣候科學應用：氣象學家經常使用插值從離散氣象站數據創建連續天氣地圖。通過在地理區域間插值溫度、壓力和濕度測量，科學家可以以卓越的準確性建模天氣模式和預測大氣行為。
計算機圖形和動畫：現代計算機圖形嚴重依賴插值算法從控制點創建平滑曲線和表面。貝塞爾曲線、B 樣條和其他插值技術構成計算機輔助設計軟件和動畫系統的數學基礎。
金融建模：利率曲線，對衍生品定價和風險評估至關重要，使用複雜的插值技術構建。這些曲線必須準確反映市場條件，同時提供適合數學分析的平滑、連續函數。
醫學成像：先進的醫學成像系統，包括 MRI 和 CT 掃描儀，在圖像重建過程中廣泛使用插值。離散傳感器測量被插值以創建詳細、連續的圖像，供醫生分析診斷。

除了數據重建，插值還通過提供複雜現象的簡單數學表示來促進函數近似。許多真實世界過程由在數學上難以處理或計算昂貴的方程控制。插值為用多項式或其他易於操作的數學形式近似這些函數提供了途徑。

1.1.5 激勵性例子

為了說明插值的實際重要性，考慮一位研究反應動力學的化學工程師。工程師定期測量反應物的濃度，獲得如表 1.1 所示的數據。

時間（分鐘）	濃度（mol/L）
0	1.00
5	0.78
10	0.61
15	0.47
20	0.37

Table 1.1: 化學反應濃度數據

工程師需要確定在 $t = 7.5$ 分鐘時的濃度以驗證理論模型。沒有插值，這將需要額外昂貴的實驗。然而，插值提供了一種系統的數學方法來估計這個中間值，可能節省大量時間和資源，同時為模型驗證提供必要的數據。

1.2 插值的基本定理

1.2.1 存在性和唯一性

多項式插值的數學基礎建立在一個基本定理上，該定理保證在特定條件下插值多項式的存在性和唯一性。

定理

定理 1.1 (多項式插值的存在性和唯一性)**.** 給定 $n+1$ 個不同的點 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$，其中 $x_i \neq x_j$ 對於 $i \neq j$，存在唯一的最多 n 次多項式 $P_n(x)$ 使得：

$$P_n(x_i) = y_i \quad \text{對於} i = 0, 1, \ldots, n$$

證明大綱：證明依賴於產生的線性系統的結構。當我們尋求多項式 $P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ 時，插值條件創建了一個關於 $n+1$ 個未知數（係數 $a_0, a_1, \ldots, a_n$）的 $n+1$ 個線性方程組。係數矩陣是著名的範德蒙德矩陣，當所有 x_i 值都不同時，它是非奇異的。這保證了系統有唯一解，建立了插值多項式的存在性和唯一性。

1.2.2 拉格朗日插值

雖然存在性和唯一性定理保證插值多項式存在，但它沒有提供找到它的構造性方法。拉格朗日插值公式通過提供插值多項式的顯式表示來填補這個空白。

定理

定理 1.2 (拉格朗日插值公式)**.** 通過點 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ 的最多 n 次唯一插值多項式由下式給出：

$$P_n(x) = \sum_{i=0}^{n} y_i L_i(x)$$

其中拉格朗日基多項式為：

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

拉格朗日基多項式具有幾個令人矚目的性質，使它們非常適合插值。最重要的性質是 $L_i(x_k) = \delta_{ik}$，其中 δ_{ik} 是克羅內克符號函數。這意味著 $L_i(x)$ 在 x_i 處等於 1，在所有其他插值點處等於 0。此外，對於所有 x 值，和 $\sum_{i=0}^{n} L_i(x) = 1$，這反映了拉格朗日基形成單位分割的事實。每個基多項式 $L_i(x)$ 本身是 n 次多項式，儘管最終的插值多項式 $P_n(x)$ 的次數最多為 n。

1.2.3 牛頓前向差分法

對於等間距的數據點，牛頓前向差分法提供了一種替代插值方法，通常比拉格朗日插值計算效率更高。

定理

定理 1.3 (牛頓前向差分插值)**.** 對於等間距點 $x_i = x_0 + ih$，其中 h 是步長，插值多項式可以寫成：

$$P_n(x) = f[x_0] + \binom{s}{1}\Delta f[x_0] + \binom{s}{2}\Delta^2 f[x_0] + \cdots + \binom{s}{n}\Delta^n f[x_0]$$

其中 $s = \frac{x - x_0}{h}$，$\Delta^k f[x_0]$ 表示 f 在 x_0 處的 k 階前向差分。

前向差分算子 Δ 遞歸定義：一階差分為 $\Delta f[x_i] = f[x_{i+1}] - f[x_i]$，高階差分為 $\Delta^k f[x_i] = \Delta^{k-1} f[x_{i+1}] - \Delta^{k-1} f[x_i]$。這種方法在處理規則間隔的表格數據時特別有價值，因為它減少了計算複雜性，並通過連續差分的行為提供對底層函數平滑性的洞察。

1.2.4 誤差分析和邊界

理解插值的準確性對實際應用至關重要。插值誤差定理提供理論洞察和近似質量的實際邊界。

定理

定理 1.4 (插值誤差邊界). 設 $f(x)$ 是在包含所有插值點 $x_0, x_1, \ldots, x_n$ 的區間 $[a, b]$ 上 $(n+1)$ 次連續可微的函數。如果 $P_n(x)$ 是插值多項式，那麼對於任何 $x \in [a, b]$：

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

對某個 $\xi \in (a, b)$。

這個基本結果揭示了關於插值準確性的幾個重要洞察。誤差大小直接取決於被插值函數的 $(n+1)$ 階導數，這意味著具有大高階導數的函數將更難以準確插值。誤差在所有插值點處恆等於零，確認插值多項式精確再現給定數據。最重要的是，誤差可以通過策略性選擇插值點來最小化，如乘積項 $\prod_{i=0}^{n}(x - x_i)$ 所證明的。

1.2.5 龍格現象

當使用等間距插值點的高次多項式時，多項式插值出現一個關鍵限制，這個問題稱為龍格現象。

定理

定理 1.5 (龍格現象). 對於區間 $[-1, 1]$ 上的函數 $f(x) = \frac{1}{1+25x^2}$，使用等間距點的多項式插值在邊界附近表現出振盪行為，隨著多項式次數的增加而加劇。

龍格現象表明高次多項式插值並不總是優於低次方法。這個反直觀的結果對實際插值有深遠影響。振盪通常在插值區間的邊界附近表現，插值多項式儘管完全符合數據點，卻可能表現出劇烈波動。這種現象促使開發替代方法，包括樣條等分段插值方法、用於最優點選擇的切比雪夫節點的使用，以及考慮避免高次多項式的替代插值技術。

1.2.6 切比雪夫插值

多項式插值的最優點選擇問題在切比雪夫插值理論中找到解決方案。這種方法解決了多項式插值的基本限制之一：當使用等間距插值點時可能出現的不良行為，特別是對於高次多項式。

第一類切比雪夫多項式

在討論最優插值點之前，我們必須介紹切比雪夫多項式，它們構成理解為什麼某些點分佈優於其他分佈的理論基礎。

定義 **1.2** (切比雪夫多項式). 第一類切比雪夫多項式 $T_n(x)$ 在區間 $[-1, 1]$ 上定義為：

$$T_n(x) = \cos(n \arccos(x)), \quad x \in [-1, 1]$$

對於 $n = 0, 1, 2, \ldots$

前幾個切比雪夫多項式是：

$$T_0(x) = 1 \tag{1.1}$$
$$T_1(x) = x \tag{1.2}$$
$$T_2(x) = 2x^2 - 1 \tag{1.3}$$
$$T_3(x) = 4x^3 - 3x \tag{1.4}$$
$$T_4(x) = 8x^4 - 8x^2 + 1 \tag{1.5}$$
$$T_5(x) = 16x^5 - 20x^3 + 5x \tag{1.6}$$

這些多項式滿足顯著的遞歸關係：

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

對於 $n \geq 1$，初始條件為 $T_0(x) = 1$ 和 $T_1(x) = x$。

切比雪夫多項式的性質

切比雪夫多項式具有幾個非凡的性質，使它們成為近似理論的基礎：

正交性：切比雪夫多項式在 $[-1, 1]$ 上關於權函數 $w(x) = (1 - x^2)^{-1/2}$ 形成正交系統：

$$\int_{-1}^{1} T_m(x)T_n(x)\frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0 & \text{如果} m \neq n \\ \pi & \text{如果} m = n = 0 \\ \frac{\pi}{2} & \text{如果} m = n > 0 \end{cases}$$

極值性質：在所有 n 次首項係數為 1 的多項式中，多項式 $2^{1-n}T_n(x)$ 在 $[-1, 1]$ 上具有最小的最大絕對值。具體地：

$$\max_{x\in[-1,1]} |2^{1-n}T_n(x)| = 2^{1-n}$$

零點和極值：$T_n(x)$ 的 n 個零點由下式給出：

$$x_k = \cos\left(\frac{(2k+1)\pi}{2n}\right), \quad k = 0, 1, \ldots, n-1$$

極值（ $|T_n(x)| = 1$ 的點 ）出現在：

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0, 1, \ldots, n$$

最優插值點

定理

定理 1.6 (切比雪夫插值點)**.** 最小化最大插值誤差的最優插值點選擇由切比雪夫節點給出：

$$x_k = \cos\left(\frac{(2k+1)\pi}{2(n+1)}\right), \quad k = 0, 1, \ldots, n$$

這些點最小化量 $\max_{x\in[-1,1]} |\prod_{i=0}^{n}(x - x_i)|$。

證明概要：乘積 $\prod_{i=0}^{n}(x - x_i)$ 是 $n+1$ 次首項係數為 1 的多項式。根據切比雪夫多項式的極值性質，當它等於 $2^{-n}T_{n+1}(x)$ 時這個乘積被最小化，這恰好發生在 x_i 是 $T_{n+1}(x)$ 的零點時。

切比雪夫插值的誤差分析

當使用切比雪夫節點時，插值誤差邊界變得比等間距點顯著更有利。

定理

定理 1.7 (切比雪夫插值誤差)**.** 對於在 $[-1,1]$ 上 $(n+1)$ 次連續可微的函數 f，切比雪夫插值的誤差被界定為：

$$|f(x) - P_n(x)| \leq \frac{2^{-n}}{(n+1)!} \max_{\xi \in [-1,1]} |f^{(n+1)}(\xi)|$$

對所有 $x \in [-1,1]$。

這個邊界比等間距點的相應邊界顯著更好，特別是對於大的 n。因子 2^{-n} 代表誤差邊界的指數改進。

轉換到一般區間

對於一般區間 $[a,b]$ 上的插值，切比雪夫節點使用線性映射進行轉換：

$$x_k = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2k+1)\pi}{2(n+1)}\right)$$

這種轉換在轉換區間上保持切比雪夫節點的最優性質。

收斂性質

切比雪夫插值最顯著的特徵之一是其對解析函數的收斂行為。

定理

定理 1.8 (切比雪夫插值的收斂性)**.** 如果 f 在包含 $[-1,1]$ 的區域內是解析的，那麼當 $n \to \infty$ 時，切比雪夫插值多項式序列在 $[-1,1]$ 上一致收斂到 f。

這與等間距點的插值形成鮮明對比，即使對於解析函數，龍格現象也可能導致發散。

實際實現

Python 程式碼

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange
from numpy.polynomial.chebyshev import Chebyshev

def chebyshev_nodes(n, interval=(-1, 1)):
    """生成切比雪夫插值節點。"""
    a, b = interval
    k = np.arange(n + 1)
    nodes = np.cos((2*k + 1) * np.pi / (2*(n + 1)))
    # 轉換到 [a, b]
    return (a + b)/2 + (b - a)/2 * nodes
```

```
def equally_spaced_nodes(n, interval=(-1, 1)):
    """生成等間距節點。"""
    a, b = interval
    return np.linspace(a, b, n + 1)

# 測試函數 - 龍格例子
def runge_function(x):
    return 1 / (1 + 25 * x**2)

# 比較不同節點分佈的插值
degrees = [5, 10, 15, 20]
x_fine = np.linspace(-1, 1, 1000)
y_true = runge_function(x_fine)

fig, axes = plt.subplots(2, 2, figsize=(15, 12))
axes = axes.flatten()

for i, n in enumerate(degrees):
    ax = axes[i]

    # 等間距插值
    x_equal = equally_spaced_nodes(n)
    y_equal = runge_function(x_equal)
    p_equal = lagrange(x_equal, y_equal)
    y_equal_interp = p_equal(x_fine)

    # 切比雪夫插值
    x_cheb = chebyshev_nodes(n)
    y_cheb = runge_function(x_cheb)
    p_cheb = lagrange(x_cheb, y_cheb)
    y_cheb_interp = p_cheb(x_fine)

    # 繪製結果
    ax.plot(x_fine, y_true, 'k-', linewidth=2, label='真實函數')
    ax.plot(x_fine, y_equal_interp, 'r--', linewidth=1.5,
            label='等間距', alpha=0.8)
    ax.plot(x_fine, y_cheb_interp, 'b-', linewidth=1.5,
            label='切比雪夫節點', alpha=0.8)
    ax.plot(x_equal, y_equal, 'ro', markersize=4, label='等間距節點')
    ax.plot(x_cheb, y_cheb, 'bo', markersize=4, label='切比雪夫節點')

    ax.set_title(f'{n} 次插值')
    ax.set_ylim(-2, 2)
    ax.grid(True, alpha=0.3)
    ax.legend()

plt.tight_layout()
plt.show()

# 誤差分析
print("最大插值誤差：")
print("次數\t等間距\t\t切比雪夫")
```

```
print("-" * 40)
for n in degrees:
    x_equal = equally_spaced_nodes(n)
    y_equal = runge_function(x_equal)
    p_equal = lagrange(x_equal, y_equal)

    x_cheb = chebyshev_nodes(n)
    y_cheb = runge_function(x_cheb)
    p_cheb = lagrange(x_cheb, y_cheb)

    error_equal = np.max(np.abs(y_true - p_equal(x_fine)))
    error_cheb = np.max(np.abs(y_true - p_cheb(x_fine)))

    print(f"{n:2d}\t{error_equal:.2e}\t{error_cheb:.2e}")
```

對切比雪夫插值的增強理解揭示了為什麼它代表數值近似中最重要的技術之一，為廣泛的應用提供理論最優性和實際穩健性。

1.2.7 樣條插值

高次多項式插值的局限性導致樣條插值的發展，它使用分段多項式來實現更好的整體近似性質。”樣條”一詞源於造船師和製圖員用來通過一組點繪製平滑曲線的柔性木條或金屬條。

三次樣條的數學基礎

定理

定理 1.9 (三次樣條插值). 給定 $n+1$ 個數據點，存在唯一的三次樣條 $S(x)$ 滿足以下條件：

1. $S(x)$ 插值所有數據點，即 $S(x_i) = y_i$，
2. $S(x)$ 在整個定義域上二次連續可微，即 $S(x) \in C^2[x_0, x_n]$，
3. $S(x)$ 在每個區間 $[x_i, x_{i+1}]$ 上由三次多項式組成，以及
4. $S(x)$ 滿足指定的邊界條件。

三次樣條通過使用在連接點保持平滑性的分段三次多項式來克服與高次多項式插值相關的振盪行為。這種方法提供優秀的近似性質，同時避免可能困擾全局多項式插值方法的不穩定性。

三次樣條的構造

對於每個區間 $[x_i, x_{i+1}]$，三次樣條可以表示為：

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

在內部節點 $x_1, x_2, \ldots, x_{n-1}$ 處的連續性條件要求：
函數連續性：$S_i(x_{i+1}) = S_{i+1}(x_{i+1})$ 對於 $i = 0, 1, \ldots, n-2$
一階導數連續性：$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ 對於 $i = 0, 1, \ldots, n-2$
二階導數連續性：$S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$ 對於 $i = 0, 1, \ldots, n-2$
這些條件，結合插值要求 $S_i(x_i) = y_i$，為 $4n$ 個未知係數提供了 $4n$ 個方程。

邊界條件

為了確保唯一性，三次樣條需要額外的邊界條件。最常見的類型是：

定義 **1.3** (自然樣條邊界條件)**.** 自然樣條滿足：

$$S''(x_0) = 0 \quad \text{且} \quad S''(x_n) = 0$$

這對應於端點處的零曲率，模擬兩端自由的物理樣條的行為。

定義 **1.4** (固定樣條邊界條件)**.** 固定樣條滿足：

$$S'(x_0) = f'(x_0) \quad \text{且} \quad S'(x_n) = f'(x_n)$$

其中端點處的導數被指定，通常基於已知或估計值。

定義 **1.5** (非節點邊界條件)**.** 這個條件強制 $S'''(x_1) = S_0'''(x_1)$ 和 $S'''(x_{n-1}) = S_{n-1}'''(x_{n-1})$，有效地使樣條在前兩個和後兩個區間上成為單一三次多項式。

三次樣條構造算法

構造三次樣條最有效的方法使用二階導數作為主要變量。設 $M_i = S''(x_i)$ 對於 $i = 0, 1, \ldots, n$。

定理 **1.10** (三次樣條算法)**.** 給定每個節點處的二階導數 M_i，區間 $[x_i, x_{i+1}]$ 上的三次樣條為：

$$S_i(x) = \frac{M_i}{6h_i}(x_{i+1} - x)^3 + \frac{M_{i+1}}{6h_i}(x - x_i)^3 + \left(\frac{y_i}{h_i} - \frac{M_i h_i}{6}\right)(x_{i+1} - x) + \left(\frac{y_{i+1}}{h_i} - \frac{M_{i+1} h_i}{6}\right)(x - x_i)$$

其中 $h_i = x_{i+1} - x_i$。

二階導數滿足三對角系統：

$$h_{i-1} M_{i-1} + 2(h_{i-1} + h_i) M_i + h_i M_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$

對於 $i = 1, 2, \ldots, n-1$。

性質和優勢

三次樣條具有幾個顯著性質，使它們優於高次多項式插值：

最小曲率性質：在所有插值給定數據點的二次連續可微函數中，自然三次樣條最小化二階導數平方的積分：

$$\int_{x_0}^{x_n} [f''(x)]^2 dx$$

這個性質反映了柔性梁的物理行為，它自然地假設最小化彎曲能量的形狀。

收斂性質：對於足夠平滑的函數，三次樣條插值表現出優秀的收斂行為。如果 $f \in C^4[a, b]$ 且 $h = \max_i h_i$，那麼：

$$\|f - S\|_\infty = O(h^4)$$
$$\|f' - S'\|_\infty = O(h^3)$$
$$\|f'' - S''\|_\infty = O(h^2)$$

形狀保持：與高次多項式不同，三次樣條保持數據的一般形狀，不會引入虛假振盪。

計算實現

Python 程式碼

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline, interp1d
from scipy import sparse
from scipy.sparse.linalg import spsolve

def natural_cubic_spline(x, y):
    """
    構造自然三次樣條插值。
    """
    n = len(x) - 1
    h = np.diff(x)

    # 為二階導數構建三對角系統
    A = np.zeros((n-1, n-1))
    b = np.zeros(n-1)

    # 填充三對角矩陣
    for i in range(n-1):
        if i > 0:
            A[i, i-1] = h[i-1]
        A[i, i] = 2 * (h[i-1] + h[i]) if i > 0 else 2 * (h[0] + h[1])
        if i < n-2:
            A[i, i+1] = h[i+1]

        # 右端項
        if i == 0:
            b[i] = 6 * ((y[2] - y[1])/h[1] - (y[1] - y[0])/h[0])
        else:
            b[i] = 6 * ((y[i+2] - y[i+1])/h[i+1] - (y[i+1] - y[i])/h[i])

    # 求解內部二階導數
    M_interior = np.linalg.solve(A, b)

    # 自然邊界條件：M[0] = M[n] = 0
    M = np.zeros(n+1)
    M[1:n] = M_interior

    return M

def evaluate_spline(x_data, y_data, M, x_eval):
    """
    在給定點評估三次樣條。
    """
    n = len(x_data) - 1
    result = np.zeros_like(x_eval)

    for i in range(n):
        # 找到當前區間的點
        mask = (x_eval >= x_data[i]) & (x_eval <= x_data[i+1])
        x_seg = x_eval[mask]
```

```
        if len(x_seg) > 0:
            h = x_data[i+1] - x_data[i]

            # 三次樣條公式
            t1 = M[i] * (x_data[i+1] - x_seg)**3 / (6*h)
            t2 = M[i+1] * (x_seg - x_data[i])**3 / (6*h)
            t3 = (y_data[i]/h - M[i]*h/6) * (x_data[i+1] - x_seg)
            t4 = (y_data[i+1]/h - M[i+1]*h/6) * (x_seg - x_data[i])

            result[mask] = t1 + t2 + t3 + t4

    return result

# 演示：比較不同插值方法
np.random.seed(42)
x_data = np.array([0, 1, 2, 3, 4, 5, 6])
y_data = np.array([0, 1, 4, 1, 2, 3, 0]) + 0.1 * np.random.randn(7)

x_fine = np.linspace(0, 6, 200)

# 不同插值方法
# 1. 線性插值
linear_interp = interp1d(x_data, y_data, kind='linear')
y_linear = linear_interp(x_fine)

# 2. 拉格朗日多項式（高次）
from scipy.interpolate import lagrange
poly = lagrange(x_data, y_data)
y_poly = poly(x_fine)

# 3. 自然三次樣條
spline_natural = CubicSpline(x_data, y_data, bc_type='natural')
y_spline_nat = spline_natural(x_fine)

# 4. 固定三次樣條
spline_clamped = CubicSpline(x_data, y_data, bc_type='clamped')
y_spline_clamp = spline_clamped(x_fine)

# 繪圖比較
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(15, 12))

# 線性插值
ax1.plot(x_data, y_data, 'ro', markersize=8, label='數據點')
ax1.plot(x_fine, y_linear, 'b-', linewidth=2, label='線性插值')
ax1.set_title('線性插值')
ax1.grid(True, alpha=0.3)
ax1.legend()

# 多項式插值
ax2.plot(x_data, y_data, 'ro', markersize=8, label='數據點')
ax2.plot(x_fine, y_poly, 'r-', linewidth=2, label='多項式插值')
ax2.set_title('多項式插值（拉格朗日）')
```

```
ax2.set_ylim(-5, 8)
ax2.grid(True, alpha=0.3)
ax2.legend()

# 自然三次樣條
ax3.plot(x_data, y_data, 'ro', markersize=8, label='數據點')
ax3.plot(x_fine, y_spline_nat, 'g-', linewidth=2, label='自然三次樣條')
ax3.set_title('自然三次樣條')
ax3.grid(True, alpha=0.3)
ax3.legend()

# 固定三次樣條
ax4.plot(x_data, y_data, 'ro', markersize=8, label='數據點')
ax4.plot(x_fine, y_spline_clamp, 'm-', linewidth=2, label='固定三次樣條')
ax4.set_title('固定三次樣條')
ax4.grid(True, alpha=0.3)
ax4.legend()

plt.tight_layout()
plt.show()

print("樣條性質分析：")
print(f"自然樣條 - 端點零曲率：{spline_natural.derivative(2)(x_data[0]):.6f},
    ↪ {spline_natural.derivative(2)(x_data[-1]):.6f}")
print(f"最大曲率幅度（自然）：{np.max(np.abs(spline_natural.derivative(2)(x_fine))):.3f}")
print(f"最大曲率幅度（固定）：{np.max(np.abs(spline_clamped.derivative(2)(x_fine))):.3f}")
```

樣條理論的發展代表了由實際工程問題啟發的數學最成功的例子之一，最終導致了從計算機圖形到數值分析等應用的豐富理論框架。

1.3 實際實現和應用

例子 **1.1** (線性插值). 考慮兩點 (x_0, y_0) 和 (x_1, y_1) 的最簡單多項式插值情況。線性插值函數的形式為：

$$P_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

對於特定點 $(1, 2)$ 和 $(3, 8)$，我們可以求出 $x = 2$ 處的值：

$$P_1(2) = 2 + \frac{8 - 2}{3 - 1}(2 - 1) = 2 + \frac{6}{2} \cdot 1 = 2 + 3 = 5$$

例子 **1.2** (使用拉格朗日方法的二次插值). 對於三個點 $(0, 1)$、$(1, 4)$ 和 $(2, 9)$，我們可以構造拉格朗日插值多項式：

$$P_2(x) = 1 \cdot \frac{(x-1)(x-2)}{(0-1)(0-2)} + 4 \cdot \frac{(x-0)(x-2)}{(1-0)(1-2)} + 9 \cdot \frac{(x-0)(x-1)}{(2-0)(2-1)} \tag{1.7}$$

$$= \frac{(x-1)(x-2)}{2} - 4x(x-2) + \frac{9x(x-1)}{2} \tag{1.8}$$

$$= \frac{x^2 - 3x + 2}{2} - 4x^2 + 8x + \frac{9x^2 - 9x}{2} \tag{1.9}$$

$$= x^2 + 2x + 1 \tag{1.10}$$

Python 程式碼

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import lagrange, CubicSpline

# 例子：比較不同插值方法
x_data = np.array([0, 1, 2, 3, 4, 5])
y_data = np.array([1, 0.5, 0.25, 0.125, 0.0625, 0.03125])

# 拉格朗日多項式插值
poly_lagrange = lagrange(x_data, y_data)

# 三次樣條插值
spline = CubicSpline(x_data, y_data)

# 生成細緻網格用於繪圖
x_plot = np.linspace(0, 5, 200)
y_lagrange = poly_lagrange(x_plot)
y_spline = spline(x_plot)

# 創建比較圖
plt.figure(figsize=(12, 8))
plt.plot(x_data, y_data, 'ro', markersize=8, label='數據點')
plt.plot(x_plot, y_lagrange, 'b-', linewidth=2, label='拉格朗日多項式')
plt.plot(x_plot, y_spline, 'g-', linewidth=2, label='三次樣條')
plt.xlabel('x')
plt.ylabel('y')
plt.title('插值方法比較')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()

# 分析插值精度
print(f"拉格朗日多項式次數：{poly_lagrange.order}")
print(f"樣條平滑性：C^2 連續性")
```

1.4 高級插值技術

1.4.1 埃爾米特插值

傳統多項式插值僅使用插值點處的函數值。埃爾米特插值通過結合導數信息來擴展這個概念，當函數值和導數都已知時，允許更複雜的近似。

當導數信息可用時，埃爾米特插值可以用較少的插值點達到更高的精度。該方法構造不僅匹配函數值，還匹配插值點處的一階（以及可能更高階）導數的多項式。這種方法在導數信息容易獲得或維持平滑性質至關重要的應用中特別有價值。

1.4.2 三角插值

對於表現出週期性行為的數據，三角插值通常比多項式方法提供更優越的結果。三角插值使用正弦和餘弦來構造插值函數，而不是使用多項式基函數。

這種方法自然處理週期數據，並構成離散傅里葉變換（DFT）和快速傅里葉變換（FFT）的數學基礎。在信號處理應用中，三角插值能夠從離散樣本重建連續信號，促進濾波、重採樣和頻譜分析等操作。

1.4.3 多維插值

實際應用經常需要多維插值。二維插值可能需要從散點數據估計表面上的值，而三維插值可能需要建模空間中的現象。

常見方法包括規則網格數據的雙線性和雙三次插值、散點數據的徑向基函數，以及地質統計應用的克里金方法。每種方法根據數據結構和應用的精度要求都有特定優勢。

Chapter 2

曲線擬合

2.1 曲線擬合簡介

2.1.1 區分曲線擬合與插值

雖然插值尋求構造一個精確通過所有給定數據點的函數，但曲線擬合採用根本不同的方法。曲線擬合承認真實世界數據經常包含測量誤差、噪音和其他不完美之處，使得精確插值既不理想也無意義。相反，曲線擬合旨在找到最好代表數據中潛在趨勢或模式的函數，即使它不精確通過每個數據點。

定義 **2.1** (曲線擬合)**.** 給定一組數據點 $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ 和一類函數 $\mathcal{F}$，曲線擬合尋求找到函數 $f^* \in \mathcal{F}$ 最小化 f 和數據之間的某種差異度量，通常表達為：

$$f^* = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{m} \rho(y_i - f(x_i))$$

其中 ρ 是量化擬合誤差的損失函數。

損失函數 ρ 的選擇取決於具體應用和對數據的假設。最常見的選擇是平方誤差損失 $\rho(r) = r^2$，這導致最小二乘法。然而，當數據包含離群值或需要不同誤差特徵時，其他損失函數如絕對誤差 ($\rho(r) = |r|$) 或穩健損失函數有時更可取。

2.1.2 曲線擬合的哲學

曲線擬合體現了與插值根本不同的哲學。雖然插值將每個數據點視為精確和不可侵犯的，但曲線擬合認識到數據點通常是潛在現象的觀測值，被測量誤差、噪音或其他不確定性來源所污染。目標從完美再現數據轉向識別潛在模式或趨勢。

這種範式轉變對我們處理數據分析的方式有深遠影響。曲線擬合通常偏愛較簡單的模型，這些模型捕捉基本行為同時對數據不完美保持穩健，而不是尋求足夠複雜以匹配每個數據點的複雜性。這個原則在科學語境中被稱為奧卡姆剃刀，建議在具有相似解釋力的競爭模型中，更簡單的模型通常更可取。

2.1.3 應用和動機

曲線擬合在幾乎每個科學和工程領域都有應用。在實驗物理中，研究人員將理論模型擬合到實驗數據以提取物理參數和測試理論預測。在經濟學中，分析師將回歸模型擬合到市場數據以理解變量間關係並進行預測。在工程中，曲線擬合能夠從測量數據建模系統行為，促進設計優化和性能預測。

實際應用

藥物開發：製藥研究人員使用曲線擬合建模藥物濃度與時間數據，能夠確定重要的藥代動力學參數，如半衰期、清除率和生物利用度。這些參數直接影響劑量方案和治療協議。
氣候科學：氣候科學家將數學模型擬合到溫度、降水和大氣數據以理解長期趨勢並預測未來氣候條件。這些模型必須在複雜性與可解釋性之間平衡，同時考慮自然變異性。
質量控制：製造工程師使用曲線擬合建模工藝參數與產品質量指標之間的關係。這能夠優化製造工藝並從工藝條件預測質量結果。
天文學：天文學家將模型擬合到觀測數據以確定恆星性質、軌道參數和宇宙學常數。這些擬合的精度往往決定基本物理常數和宇宙學模型的準確性。

2.2 線性最小二乘

2.2.1 最小二乘法

最小二乘法由卡爾·弗里德里希·高斯和阿德里安-馬里·勒讓德在 19 世紀初開發，為現代曲線擬合的大部分提供了數學基礎。該方法最小化平方殘差之和，其中殘差是觀測值與預測值之間的差異。

定理

定理 2.1 (線性最小二乘解)**.** 對於線性系統 $\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon}$，其中 $\mathbf{A} \in \mathbb{R}^{m\times n}$ 且 $m \geq n$，最小化 $\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$ 的最小二乘解由下式給出：

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}$$

條件是 $\mathbf{A}^T\mathbf{A}$ 是可逆的。

這個結果的推導來自多變數微積分。我們尋求最小化目標函數 $J(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$。對 $\mathbf{x}$ 取梯度並設為零，得到法方程：$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{y}$。當 $\mathbf{A}^T\mathbf{A}$ 可逆時，該系統有上述唯一解。

2.2.2 多項式曲線擬合

最小二乘最常見的應用之一是多項式曲線擬合，我們尋求將 n 次多項式擬合到數據點 (x_i, y_i)，其中 $i = 1, 2, \ldots, m$。

例子 **2.1** (二次曲線擬合)**.** 考慮將二次多項式 $f(x) = a_0 + a_1x + a_2x^2$ 擬合到數據點。設計矩陣變為：

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 \end{pmatrix}$$

參數向量為 $\mathbf{x} = (a_0, a_1, a_2)^T$。

多項式次數的選擇涉及基本權衡。高次多項式可以更緊密地擬合數據，但可能對噪音過擬合並表現出差的泛化性。低次多項式提供更平滑的擬合，但可能欠擬合潛在模式。這種權衡是統計學習理論中偏差-方差分解的核心。

2.2.3 最小二乘的統計性質

在關於誤差結構的某些假設下，最小二乘估計器具有理想的統計性質。

定理

定理 2.2 (高斯-馬爾可夫定理)**.** 在線性性、零均值誤差、常數方差（同方差性）和不相關誤差的假設下，最小二乘估計器是最佳線性無偏估計器（*BLUE*）。也就是說，在所有線性無偏估計器中，它具有最小方差。

這個基本結果建立了最小二乘在其假設下的最優性。然而，當這些假設被違反時，替代方法可能更可取。例如，當誤差不是同方差時，加權最小二乘提供更好的結果。當誤差相關時，廣義最小二乘變得合適。

2.3 非線性曲線擬合

2.3.1 非線性最小二乘問題

當參數與觀測之間的關係是非線性時，最小二乘問題變得顯著更複雜。我們面對的不是簡單的線性代數問題，而是非線性優化挑戰。

定義 **2.2** (非線性最小二乘)**.** 給定數據點 (x_i, y_i) 和由 $\boldsymbol{\theta}$ 參數化的非線性模型 $f(x; \boldsymbol{\theta})$，非線性最小二乘問題尋求找到：

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{m} [y_i - f(x_i; \boldsymbol{\theta})]^2$$

與線性最小二乘不同，該問題通常缺乏封閉形式解，需要迭代數值方法。算法的選擇取決於模型函數的具體特徵以及計算效率與穩健性之間的期望平衡。

2.3.2 高斯-牛頓算法

高斯-牛頓算法代表非線性最小二乘問題最重要的方法之一。它在每次迭代中線性化問題並應用線性最小二乘解。

定理 **2.3** (高斯-牛頓方法)**.** 從初始參數猜測 $\boldsymbol{\theta}_0$ 開始，高斯-牛頓算法根據以下公式迭代更新參數估計：

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - (\mathbf{J}_k^T \mathbf{J}_k)^{-1} \mathbf{J}_k^T \mathbf{r}_k$$

其中 $\mathbf{J}_k$ 是殘差的雅可比矩陣，$\mathbf{r}_k$ 是第 k 次迭代的殘差向量。

當殘差較小且模型不太偏離線性時，高斯-牛頓方法通常快速收斂。然而，當這些條件不滿足時，它可能收斂失敗或收斂到局部最小值。列文伯格-馬夸特算法通過結合高斯-牛頓與梯度下降來解決這些限制中的一些。

2.3.3 模型選擇和驗證

選擇適當的模型結構代表曲線擬合最具挑戰性的方面之一。模型必須足夠複雜以捕捉數據的基本特徵，同時保持足夠簡單以避免過擬合。

實際應用

交叉驗證：將數據分為訓練集和驗證集。將不同複雜性的模型擬合到訓練數據，並在驗證集上評估其性能。獲得最佳驗證性能的模型提供偏差和方差之間的最優平衡。
信息準則：使用懲罰模型複雜性的準則，如赤池信息準則（AIC）或貝葉斯信息準則（BIC）。這些準則提供在同一數據集上比較不同複雜性模型的原則性方法。
正則化：在目標函數中添加懲罰項，阻止過於複雜的模型。嶺回歸（L2 正則化）和套索回歸（L1 正則化）是常見例子，有助於防止過擬合同時保持擬合精度。

2.4 穩健曲線擬合

2.4.1 最小二乘的限制

雖然最小二乘在理想條件下提供最優解，但真實世界數據經常違反潛在假設。離群值、重尾誤差分佈或異方差噪音的存在可能顯著降低最小二乘估計器的性能。

最小二乘使用的平方誤差損失函數對大殘差給予不成比例的權重，使該方法對離群值敏感。單個離群值可能戲劇性地影響擬合曲線，將其從數據的主要趨勢拉開。這種敏感性促使開發穩健回歸方法，即使在離群值存在時也能保持良好性能。

2.4.2 穩健損失函數

替代損失函數可以為離群值提供更好的穩健性，同時在乾淨數據上保持良好性能。

定義 **2.3** (胡伯損失函數)**.** 胡伯損失函數結合了小殘差的平方誤差和大殘差的絕對誤差的好處：

$$\rho_\delta(r) = \begin{cases} \frac{1}{2}r^2 & \text{如果} |r| \le \delta \\ \delta(|r| - \frac{1}{2}\delta) & \text{如果} |r| > \delta \end{cases}$$

其中 δ 是控制二次和線性行為之間轉換的調節參數。

胡伯損失函數為乾淨數據的最小二乘效率和污染數據的最小絕對偏差穩健性之間提供妥協。參數 δ 可以基於預期噪音水平選擇或從數據估計。

2.4.3 迭代重加權最小二乘

許多穩健回歸方法可以使用迭代重加權最小二乘（IRLS）實現，它在基於當前殘差計算權重和求解加權最小二乘問題之間交替。

定理 **2.4** (IRLS 算法)**.** 對於具有權重函數 $w(r) = \rho'(r)/r$ 的穩健損失函數 $\rho(r)$，*IRLS* 算法迭代：

1. 計算殘差：$r_i^{(k)} = y_i - f(x_i; \boldsymbol{\theta}^{(k)})$
2. 更新權重：$w_i^{(k)} = w(r_i^{(k)})$
3. 求解加權最小二乘：$\boldsymbol{\theta}^{(k+1)} = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^m w_i^{(k)} [y_i - f(x_i; \boldsymbol{\theta})]^2$

該算法在溫和的正則性條件下收斂到穩健估計。權重自動降低離群值的權重同時為內點保持完全權重，在不需要顯式離群值檢測的情況下實現穩健性。

2.5 正則化回歸

2.5.1 偏差-方差權衡

在許多曲線擬合應用中，我們面對偏差和方差之間的基本權衡。具有許多參數的複雜模型可以很好地擬合訓練數據（低偏差），但可能對新數據泛化較差（高方差）。簡單模型可能欠擬合數據（高偏差），但通常泛化更好（低方差）。

正則化通過在目標函數中添加懲罰項來管理這種權衡，這些懲罰項阻止過於複雜的模型。正則化參數控制這種懲罰的強度，允許我們調節偏差-方差權衡以獲得最優性能。

2.5.2 嶺回歸

嶺回歸在最小二乘目標中添加 L2 懲罰項，懲罰大參數值。

定義 **2.4** (嶺回歸)**.** 嶺回歸目標函數是：

$$J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2$$

其中 $\lambda \geq 0$ 是正則化參數。

嶺回歸解具有封閉形式：

$$\boldsymbol{\theta}^*_{ridge} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{y}$$

正則化參數 λ 控制應用於參數估計的收縮量。當 $\lambda \to 0$ 時，我們恢復普通最小二乘解。當 $\lambda \to \infty$ 時，參數估計收縮向零。

2.5.3 套索回歸

套索（最小絕對收縮和選擇算子）回歸使用 L1 懲罰而不是 L2，這可能導致某些參數恰好為零的稀疏解。

定義 **2.5** (套索回歸)**.** 套索回歸目標函數是：

$$J(\boldsymbol{\theta}) = \|\mathbf{y} - \mathbf{A}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_1$$

其中 $\|\boldsymbol{\theta}\|_1 = \sum_{i=1}^{n} |\theta_i|$ 是 L1 範數。

與嶺回歸不同，套索沒有封閉形式解，但坐標下降等高效算法可以解決優化問題。L1 懲罰通過驅動某些參數恰好為零來鼓勵稀疏性，有效地執行自動特徵選擇。

2.6 計算方法和實現

Python 程式碼

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from sklearn.linear_model import Ridge, Lasso
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score

# 生成含噪音的合成數據
```

```
np.random.seed(42)
x = np.linspace(0, 10, 50)
y_true = 2 * np.sin(x) + 0.5 * x
y_noisy = y_true + np.random.normal(0, 0.5, len(x))

# 添加一些離群值
outlier_idx = [10, 25, 40]
y_noisy[outlier_idx] += np.random.normal(0, 3, len(outlier_idx))

# 線性最小二乘多項式擬合
def polynomial_features(x, degree):
    return np.vander(x, degree + 1, increasing=True)

# 擬合不同次數的多項式
degrees = [1, 3, 5, 10]
x_plot = np.linspace(0, 10, 200)

plt.figure(figsize=(15, 10))
for i, degree in enumerate(degrees):
    plt.subplot(2, 2, i + 1)

    # 普通最小二乘
    A = polynomial_features(x, degree)
    coeffs = np.linalg.lstsq(A, y_noisy, rcond=None)[0]
    y_pred = polynomial_features(x_plot, degree) @ coeffs

    plt.plot(x, y_noisy, 'ro', alpha=0.7, label='含噪數據')
    plt.plot(x_plot, y_true, 'g-', linewidth=2, label='真實函數')
    plt.plot(x_plot, y_pred, 'b-', linewidth=2, label=f'{degree}次擬合')
    plt.title(f'多項式次數 {degree}')
    plt.legend()
    plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# 正則化回歸比較
X = polynomial_features(x, 10)
X_plot = polynomial_features(x_plot, 10)

# 嶺回歸
ridge = Ridge(alpha=1.0)
ridge.fit(X, y_noisy)
y_ridge = ridge.predict(X_plot)

# 套索回歸
lasso = Lasso(alpha=0.1)
lasso.fit(X, y_noisy)
y_lasso = lasso.predict(X_plot)

# 繪圖比較
plt.figure(figsize=(12, 8))
plt.plot(x, y_noisy, 'ro', alpha=0.7, label='含噪數據')
```

```
plt.plot(x_plot, y_true, 'g-', linewidth=3, label='真實函數')
plt.plot(x_plot, y_ridge, 'b-', linewidth=2, label='嶺回歸')
plt.plot(x_plot, y_lasso, 'r-', linewidth=2, label='套索回歸')
plt.title('正則化回歸比較')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()

# 非線性曲線擬合例子
def exponential_model(x, a, b, c):
    return a * np.exp(-b * x) + c

# 生成指數數據
x_exp = np.linspace(0, 5, 30)
y_exp_true = exponential_model(x_exp, 2, 0.5, 0.1)
y_exp_noisy = y_exp_true + np.random.normal(0, 0.1, len(x_exp))

# 擬合非線性模型
popt, pcov = curve_fit(exponential_model, x_exp, y_exp_noisy)
x_exp_plot = np.linspace(0, 5, 200)
y_exp_pred = exponential_model(x_exp_plot, *popt)

plt.figure(figsize=(10, 6))
plt.plot(x_exp, y_exp_noisy, 'ro', label='含噪數據')
plt.plot(x_exp_plot, y_exp_true, 'g-', linewidth=2, label='真實函數')
plt.plot(x_exp_plot, y_exp_pred, 'b-', linewidth=2,
         label=f'擬合：{popt[0]:.2f}*exp(-{popt[1]:.2f}*x)+{popt[2]:.2f}')
plt.title('非線性曲線擬合')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()

print(f"擬合參數：a={popt[0]:.3f}, b={popt[1]:.3f}, c={popt[2]:.3f}")
print(f"真實參數：a=2.000, b=0.500, c=0.100")
```

2.7 模型選擇和驗證

2.7.1 交叉驗證

交叉驗證提供評估模型性能和選擇最優模型複雜性的穩健方法。通過系統地將數據分割為訓練集和驗證集，我們可以估計模型對未見數據的泛化能力。

K 折交叉驗證將數據分為 k 個大致相等的子集。模型在 k-1 個子集上訓練，在剩餘子集上驗證，這個過程重複 k 次。所有折的平均驗證性能提供模型泛化能力的估計。

2.7.2 信息準則

信息準則提供平衡擬合優度與模型複雜性的模型選擇替代方法。赤池信息準則（AIC）和貝葉斯信息準則（BIC）在實踐中廣泛使用。

定義 **2.6** (信息準則)**.** 對於具有 k 個參數和對數似然 ℓ 的模型：

$$\mathrm{AIC} = 2k - 2\ell \tag{2.1}$$
$$\mathrm{BIC} = k\log(n) - 2\ell \tag{2.2}$$

其中 n 是數據點數量。

兩個準則都懲罰模型複雜性，但 BIC 對大數據集應用更強的懲罰。具有較小 AIC 或 BIC 值的模型更受青睞，因為它們在保持簡潔性的同時實現更好的性能。

2.7.3 殘差分析

檢查殘差（觀測值與預測值之間的差異）為模型充分性和潛在假設違反提供有價值的洞察。

殘差圖可以揭示建議模型誤設定、異方差性或離群值存在的模式。良好擬合的模型應該產生在零附近隨機散佈且在擬合值範圍內具有大致常數方差的殘差。

2.8 高級主題和擴展

2.8.1 貝葉斯曲線擬合

貝葉斯曲線擬合方法提供結合先驗知識和量化參數估計不確定性的原則性框架。貝葉斯方法產生參數的完整概率分佈，而不是點估計。

貝葉斯範式將參數視為具有先驗分佈的隨機變量，這些分佈編碼我們在觀察數據之前的信念。後驗分佈結合先驗與從數據導出的似然函數，提供關於參數的更新信念。

2.8.2 高斯過程回歸

高斯過程提供可以自動適應數據複雜性同時提供不確定性估計的非參數回歸方法。高斯過程不假設特定函數形式，而是在函數上放置先驗分佈。

當潛在函數未知或需要量化預測不確定性時，這種方法特別有價值。高斯過程可以捕捉複雜的非線性關係，同時避免可能困擾高維參數模型的過擬合問題。

2.8.3 機器學習方法

現代機器學習技術為傳統曲線擬合方法提供強大的替代方案。神經網絡、支持向量機和集成方法可以捕捉可能難以用參數方法建模的複雜非線性關係。

然而，這些方法經常為預測性能犧牲可解釋性。傳統曲線擬合和機器學習方法之間的選擇取決於具體應用要求，包括對可解釋性的需求、可用數據量和潛在關係的複雜性。