

Python 程序设计

第三次作业：简易客户端应用

2021211306 班 | 杜抒泽 2021211110

2023 年 11 月 17 日

目录

1 作业题目	2
2 作业内容	2
3 代码说明	7
3.1 程序需求分析	7
3.2 程序基本介绍与运行方式	8
3.3 Lowess 算法	8
3.3.1 数学原理：局部加权线性回归	8
3.3.2 具体实现	9
3.4 程序组件介绍	9
3.4.1 Lowess 算法接口	10
3.4.2 GUI 界面与应用	10
3.5 程序运行结果截图与分析	10
3.6 总结	12

1 作业题目

基于 #3 作业获取的数据 (No_Smoothing, 非平滑数据), 计算出 LOWESS (局部加权回归, fraction 取前后各 5 年的数据) 结果, 该结果可以与 #2 作业中提供的 graph.txt 文件中的 Lowess 字段进行比较。

2 作业内容

由于作业要求, 代码实现为了单文件, 没有以文件的形式分离各个模块。本次作业有部分类型定义与第二、三次作业中重复。**代码冗长, 建议直接点击目录中第三节的超链接跳转代码说明开始阅读。**

程序源代码嵌入下方的 code block 中。

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from os import getenv
from tkinter.filedialog import askopenfilename
from typing import List, Optional

import matplotlib.pyplot as plt
import numpy as np
import sys
import tkinter as tk

def my_lowess(
    data_x: np.ndarray, data_y: np.ndarray, window_size: int
) → np.ndarray:
    def w(x: float) → float:
        return (1 - np.abs(x) ** 3) ** 3 if np.abs(x) ≤ 1 else 0

    def weighted_linear_regression(
        x: np.ndarray, y: np.ndarray, w: np.ndarray
    ) → np.ndarray:
        w_sum = np.sum(w)
        x_mean = np.sum(x * w) / w_sum if w_sum ≠ 0 else 0
        y_mean = np.sum(y * w) / w_sum if w_sum ≠ 0 else 0
        x_diff: np.ndarray = x - x_mean
```

```
y_diff: np.ndarray = y - y_mean
b = np.sum(w * (x_diff * y_diff)) / np.sum(w * x_diff ** 2)
a = y_mean - b * x_mean
return a + b * x

result: np.ndarray = np.zeros(len(data_y))
for i in range(len(data_y)):
    start: int = i - window_size
    end: int = i + window_size + 1
    if start < 0:
        end -= start
        start = 0
    if end > len(data_y):
        start -= end - len(data_y)
        end = len(data_y)
    start = max(start, 0)
    end = min(end, len(data_y))

    window: np.ndarray = np.arange(start, end)
    x_local: np.ndarray = data_x[window]
    y_local: np.ndarray = data_y[window]
    result[i] = weighted_linear_regression(
        x_local - i,
        y_local,
        np.array([w(x) for x in (x_local - i) / max(abs(x_local - i))]),
        )[np.where(x_local == i)[0][0]]
return result

def lowess(data_y: np.ndarray, windows_size: int) → np.ndarray:
    if getenv('SM_LOESS'):
        import statsmodels.api as sm
        return sm.nonparametric.lowess(
            data_y,
            np.arange(len(data_y)),
            frac=(2 * windows_size) / len(data_y),
            ).T[1]

    return my_lowess(np.arange(len(data_y)), data_y, windows_size)

class WeatherDataEntry:
    def __init__(
        self, year: int, temperature: float, temperature_smoothed: float
```

```
) → None:
    self.year: int = year
    self.temperature: float = temperature
    self.temperature_smoothed: float = temperature_smoothed

@classmethod
def try_from(
    cls, year: str, temperature: str, temperature_smoothed: str
) → 'WeatherDataEntry':
    try:
        return cls(
            int(year), float(temperature), float(temperature_smoothed)
        )
    except ValueError as e:
        raise Exception(f'Invalid weather data entry {e}')

def __str__(self) → str:
    return f'{self.year} {self.temperature} {self.temperature_smoothed}'


class WeatherData:
    def __init__(self, data: Optional[List[WeatherDataEntry]] = None) → None:
        self.data: List[WeatherDataEntry] = data if data else []

    def append(self, entry: WeatherDataEntry) → None:
        self.data.append(entry)

    def __str__(self) → str:
        return 'year temperature temperature_smoothed\n' + \
            '\n'.join(str(entry) for entry in self.data)

    def get_year(self) → List[int]:
        return [entry.year for entry in self.data]

    def get_temperature(self) → List[float]:
        return [entry.temperature for entry in self.data]

    def get_temperature_smoothed(self) → List[float]:
        return [entry.temperature_smoothed for entry in self.data]

class App(tk.Tk):
    def __init__(self) → None:
        super().__init__()
```

```
self.title('Simple Weather Data Analysis')
self.create_widgets()

def create_widgets(self) → None:
    self.read_from_file_button = tk.Button(
        self, text='从文件读取', command=self.read_from_file
    )
    self.read_from_file_button.grid(row=0, column=0, pady=5, sticky='E')

    self.status_label = tk.Label(self, text='未加载数据')
    self.status_label.grid(row=0, column=1, pady=5, sticky='W')

    self.plot = FigureCanvasTkAgg(
        Figure(figsize=(13, 8), dpi=120), master=self)
    self.plot.get_tk_widget().grid(
        row=1, column=0, columnspan=2, padx=5, pady=5, sticky='NSEW'
    )

@staticmethod
def validate_input(lines: List[str]) → None:
    if len(lines) < 5:
        raise Exception('数据条数过少')
    for line in lines:
        if len(line.split()) ≠ 3:
            raise Exception('文件格式错误')
    years = [int(line.split()[0]) for line in lines]
    for i in range(len(years) - 1):
        if years[i + 1] - years[i] ≠ 1:
            raise Exception('年份不连续递增')

def read_from_file(self) → None:
    file_path = askopenfilename(
        title='选择文件', filetypes=[('文本文件', '*.txt')]
    )
    if not file_path:
        return
    with open(file_path, 'r') as f:
        lines = f.readlines()[1:]
        file_name = file_path.split('/')[-1]

    try:
        self.validate_input(lines)
    except Exception as e:
```

```
        self.status_label.config(
            text='{} 加载失败: {}'.format(file_name, e)
        )
        return

    self.status_label.config(text='已加载 {}'.format(file_name))
    self.create_plot(WeatherData(
        [WeatherDataEntry.try_from(*line.split()) for line in lines]
    ))

def create_plot(self, data: WeatherData):
    years: List[int] = data.get_year()
    temperatures: List[float] = data.get_temperature()
    std_lowess: np.ndarray = np.array(data.get_temperature_smoothed())
    my_lowess: np.ndarray = lowess(np.array(temperatures), 5)
    diff: np.ndarray = std_lowess - my_lowess

    self.plot.figure.clear()

    ax = self.plot.figure.add_subplot()
    ax.plot(
        years, temperatures, marker=".", markersize=2, linewidth=0.5,
        label='原始数据', color='lightgrey'
    )
    ax.plot(
        years, std_lowess, marker="o", markersize=2, label='样例 Lowess(5)'
    )
    ax.plot(
        years, my_lowess, marker="o", markersize=2,
        label='{} Lowess(5)'.format(
            'statsmodels' if getenv('SM_LOWESS') else '手动实现'
        )
    )
    ax.plot(
        years, diff, marker="o", markersize=1, linestyle=' ', label='差值'
    )

    width = max(years) - min(years)
    step = 20 if width > 100 else 10 if width > 50 else 5
    ax.set_xticks(range(
        ((min(years) - 1) // step + 1) * step,
        max(years) // step * step + 1,
        step,
```

```
)  
    ax.set_xlabel('年份')  
    ax.set_ylabel('温度')  
    ax.set_title('年份-温度图')  
    ax.grid(True)  
    ax.legend()  
  
    self.plot.figure.tight_layout()  
    self.plot.draw()  
  
def main() -> None:  
    if getenv('SM_LOWESS'):  
        sys.stderr.write('Using statsmodels `lowess`\n')  
    else:  
        sys.stderr.write('Using manually implemented `lowess`\n')  
  
    plt.rcParams['font.sans-serif'] = ['HYZhengYuan'] # 在其他环境中可能需要修改字体  
    plt.rcParams['axes.unicode_minus'] = False  
    app = App()  
    app.mainloop()  
  
if __name__ == '__main__':  
    main()
```

3 代码说明

3.1 程序需求分析

从第三次作业实现的客户端中获取数据，并进行数据处理。具体地，实现以下功能：

- 采取 GUI 形式，提供用户界面。
- 读取第三次作业保存的数据文件，包含年份、温度、平滑温度三列。
- 提供 lowess 算法接口，既可以调用手动实现的 lowess 算法，也可以调用第三方库 statsmodels 中的 lowess 函数。
- 利用 lowess 接口和前两列数据计算 lowess(5) 结果，与提供的结果进行比较。
- 用折线图和散点图图形化地展示原始数据、提供的 lowess(5) 结果和计算得到的 lowess(5) 结果及其差值。

3.2 程序基本介绍与运行方式

这段代码实现了一个简易的天气数据处理 GUI 客户端（暂定名为 Simple Weather Data Analysis），能够对连续年份的气温数据采用 lowess 算法进行平滑处理，并将结果与提供的数据进行比较。

程序风格方面，由于作业要求，代码实现为了单文件，没有以文件的形式分离各个模块。与前三次作业一样，这次作业仍然严格遵守 Python 的 PEP8 编码规范。

开发环境与用到的第三方库如下表：

软件环境或第三方库名	版本
操作系统 / macOS	13.0 22A380 arm64
Python	3.10.6
numpy	1.26.2
matplotlib	3.8.2
statsmodels	0.14.0

在安装 Python 后，可以使用以下命令安装依赖与运行本程序（4th.py）：

```
$ python3 -m pip install numpy matplotlib statsmodels
$ python3 4th.py # 运行手写的 lowess 算法
$ SM_LOWESS=1 python3 4th.py # 运行 statsmodels 库中的 lowess 算法
```

3.3 Lowess 算法

Lowess（全称 Locally Weighted Scatterplot Smoothing），是一种非参数回归方法，用于数据点的平滑处理。它结合了多项式回归与加权最小二乘法，以在保持数据局部特性的同时，实现整体趋势的平滑化。本程序实现中 Lowess 的核心思想是局部线性回归：通过为每个数据点周围的邻近点赋予不同的权重，然后在这些邻近点上进行加权线性回归，从而得到该点的平滑值。

3.3.1 数学原理：局部加权线性回归

设有一组数据点 $(x_i, y_i) (1 \leq i \leq n)$ ，我们的目标是找到一个函数 $f(x)$ ，使得 $f(x_i) \approx y_i$ 。在局部线性回归中，我们假设 $f(x)$ 在 x_i 的邻域内可以近似为一个线性函数，即

$$f(x) \approx a_i + b_i x$$

其中 a_i 和 b_i 是待求的系数。我们可以通过最小化残差平方和来求解 a 和 b ，即求出 a 和 b ，最小化

$$\sum_{i=1}^n (y_i - (a_i + b_i x_i))^2$$

Lowess 算法中，我们采取局部加权线性回归：定义 w_i 为每个位置的权重，求 a 和 b ，最小化

$$\sum_{i=1}^n w_i (y_i - (a_i + b_i x_i))^2$$

求解 a 和 b 采用类似最小二乘法的方法。令

$$\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}, \quad \bar{y} = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}$$

则 a 和 b 的解为

$$b = \frac{\sum_{i=1}^n w_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n w_i (x_i - \bar{x})^2}, \quad a = \bar{y} - b \bar{x}$$

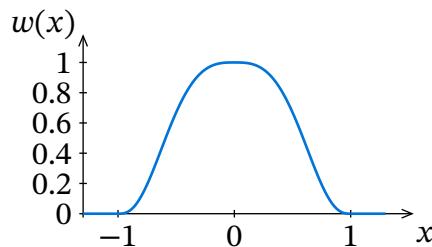
求得的近似函数为

$$f(x) = a + bx$$

3.3.2 具体实现

在本实现中，定义权重函数

$$w(x) = \begin{cases} (1 - x^3)^3, & \text{if } |x| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$



由其图像可知，函数在 $x = 0$ 的某个邻域下降较慢；当 x 的绝对值达到某个阈值时，函数值急剧下降。这样的函数可以使得 x 附近的点对于 x 的平滑值有较大的贡献，而远离 x 的点对于 x 的平滑值有较小的贡献，实现局部平滑。

对于每个数据点 x_i ，我们选取其前后各 k 个邻近点，计算其与 x_i 的距离 $d_j = |x_j - x_i|$ ，并计算其权重 $w_j = w\left(\frac{d_j}{d_{\max}}\right)$ ，其中 d_{\max} 是所有距离中的最大值。然后，我们对这些点进行加权线性回归，得到 a_i 和 b_i ，从而得到 x_i 的平滑值 $f(x_i) = a_i + b_i x_i$ 。

具体实现为始终维护一个大小为 $\mathcal{O}(k)$ 的单调滑动窗口 (i 在两侧时维护其两侧相加共 $2k$ 个点，而非每侧至多 k 个点)，计算窗口内的加权线性回归结果，作为 x_i 的平滑值。算法只进行了上述的一次迭代，没有进行多次迭代。

3.4 程序组件介绍

数据类已在第二次作业中介绍，这里不再赘述。以下是本次作业中新增的组件。

3.4.1 Lowess 算法接口

函数 `lowness` 会根据环境变量 `SM_LOESS` 的值选择调用 `statsmodels` 库中的 `lowess` 函数或手动实现的 `my_lowess` 函数。因此可以通过设置环境变量 `SM_LOESS` 的值来切换算法的实现¹。

`lowness` 函数的实现如下：

```
def lowess(data_y: np.ndarray, windows_size: int) → np.ndarray:
    if getenv('SM_LOESS'):
        import statsmodels.api as sm
        return sm.nonparametric.lowess(
            data_y,
            np.arange(len(data_y)),
            frac=(2 * windows_size) / len(data_y),
        ).T[1]

    return my_lowess(np.arange(len(data_y)), data_y, windows_size)
```

3.4.2 GUI 界面与应用

程序的 GUI 界面托管于 `tkinter` 库，由继承自 `tkinter.Tk` 的 `App` 类实现。绘图功能托管于 `matplotlib` 库，由其提供的 `tkinter` 适配器 `FigureCanvasTkAgg` 实现。

GUI 布局十分简单：



从文件读取按钮的点击事件会调用读取文件并更新状态信息。绘图区域会显示原始数据、提供的 `lowess(5)` 结果和计算得到的 `lowess(5)` 结果及其差值。

3.5 程序运行结果截图与分析

以下给出程序对两份数据在两种 `lowess` 实现下的运行指引和截图。

¹例如，可以通过[设置一次性环境变量](#)的方式指定调用 `statsmodels` 库。

打开程序后主界面如下：

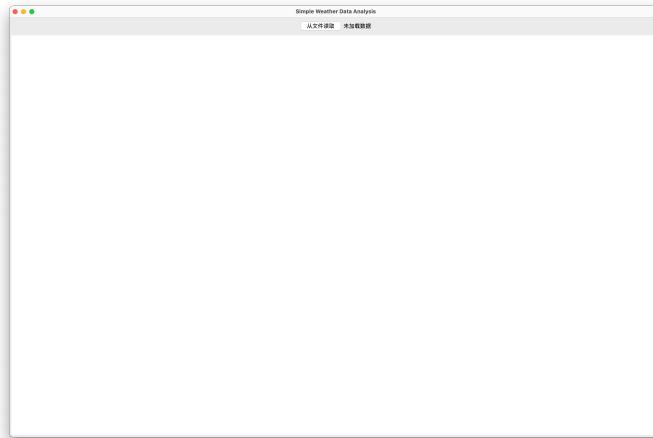


图 1：主界面（相当空荡）

点击上方的从文件读取按钮，选择数据文件，即可看到状态信息更新，绘图区域显示原始数据、提供的 lowess(5) 结果和计算得到的 lowess(5) 结果及其差值。以下是对 graph.txt 中全部数据（1880 年到 2022 年）用两个算法分别运行的截图：

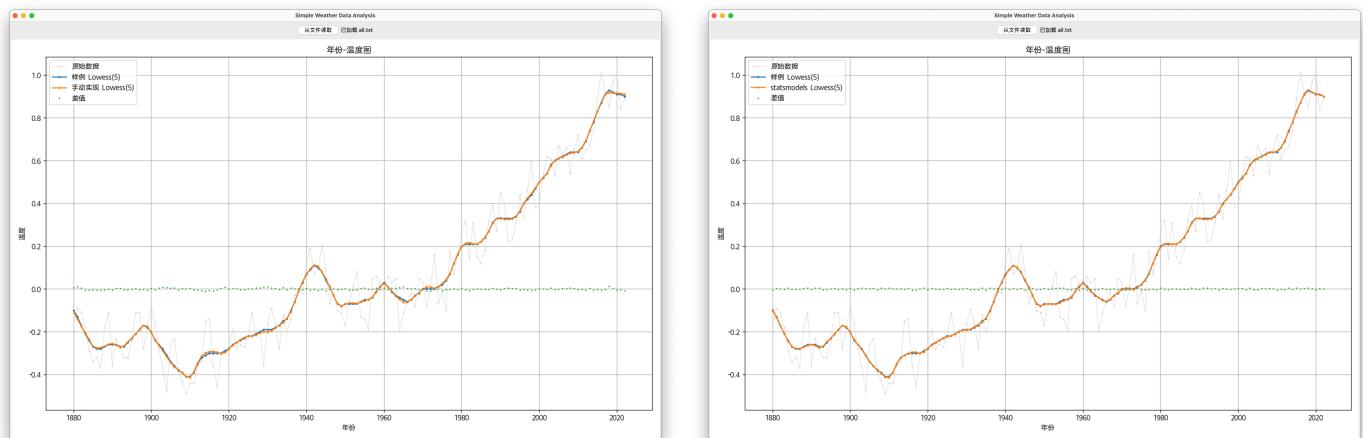


图 2：1880~2022 / 手动实现 lowness（左）/ statsmodels lowness（右）

可以看到，两个算法的结果基本一致，且都与提供的结果相差不大。statsmodels 库中的 lowess 算法的结果与提供的结果差距更小（绿色散点差值肉眼可见更接近 0 且平稳），但是手动实现的算法效果也足够令人满意。

以下是对 graph.txt 中 1949 年到 2009 年的数据用两个算法分别运行的截图：

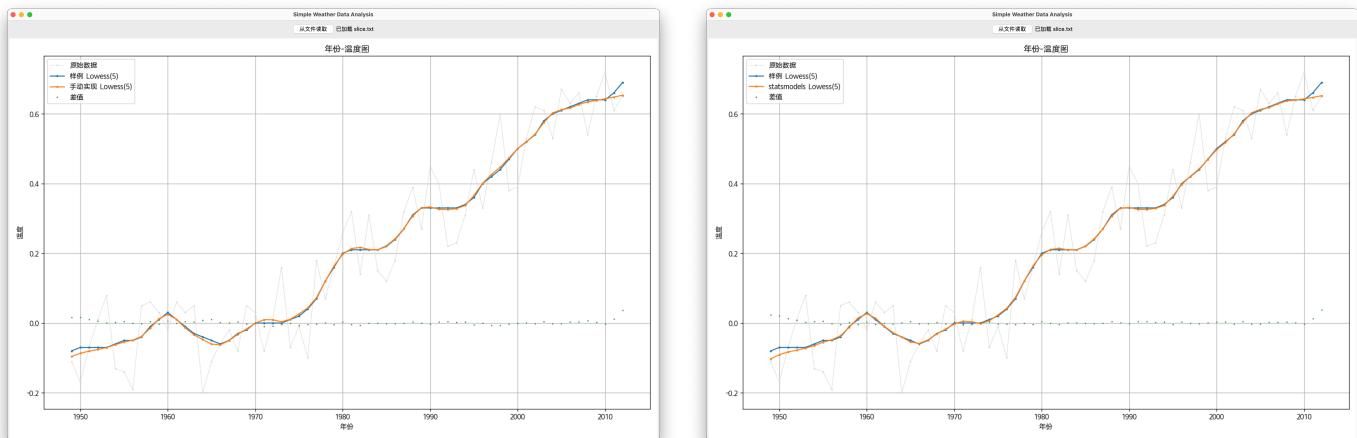


图 3: 1949~2009 / 手动实现 lowness (左) / statsmodels lowness (右)

结果与提供的结果在数据的两端有一定差异，但两个算法的结果基本一致，结果令人接受。与前一组数据相同，仍然是 statsmodels 库中的 lowess 算法的结果与提供的结果差距更小，果然调成熟的库确实是比手写厉害一点。

3.6 总结

在这次作业中，我学习了统计学中局部加权线性回归的原理，并实现了一个简易的 lowess 算法，对气温数据进行了平滑处理，并将结果与提供的样例结果和第三方库返回的结果进行了比较。与上次作业相比，本次作业新增了 matplotlib 库用于绘制图像。本次作业还有如下的改进空间：

- 与第三次作业进行耦合，在天气客户端中直接调用 lowess 算法并给出图像。
- 优化 lowess 算法的实现，测试更多加权函数；寻找多次迭代的方法，提高平滑效果。
- 增加更多的功能，例如可以选择不同的 lowess 参数（窗口大小等），可以选择不同的加权函数。