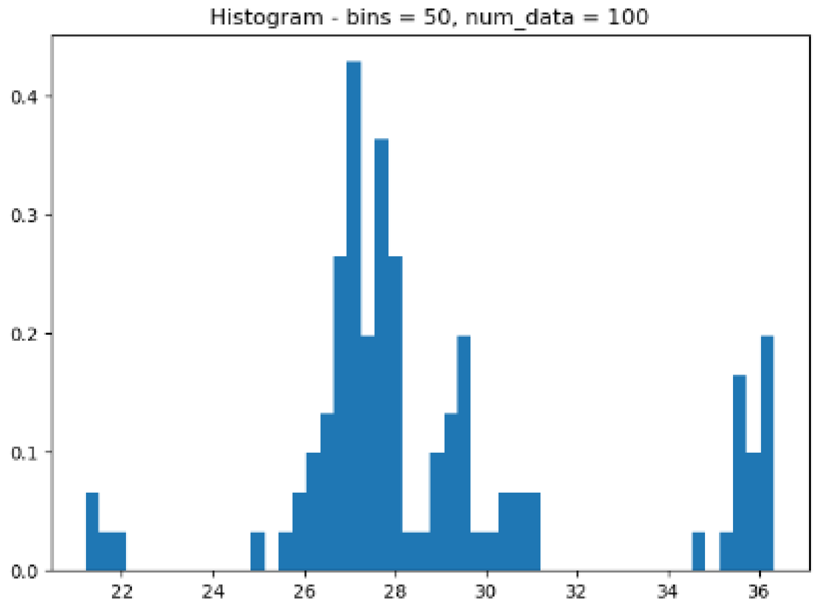
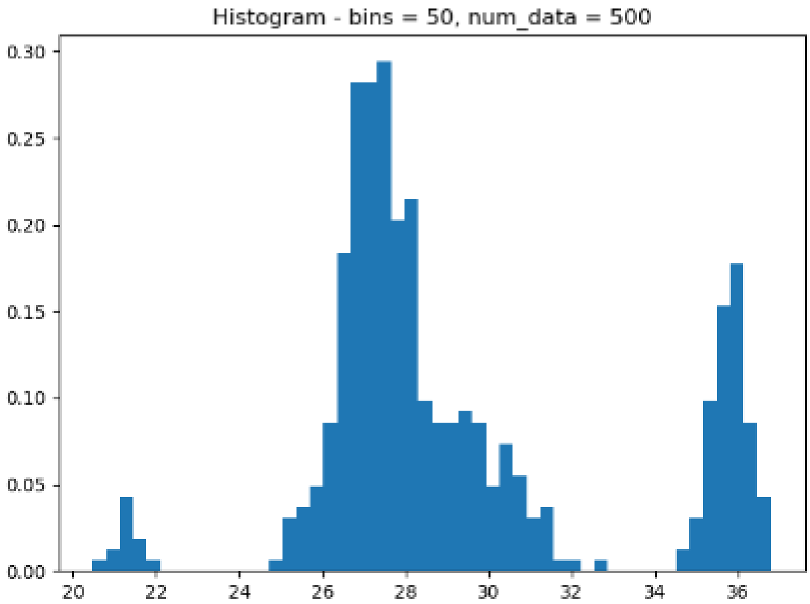
Assignment 1 of Pattern Recognition

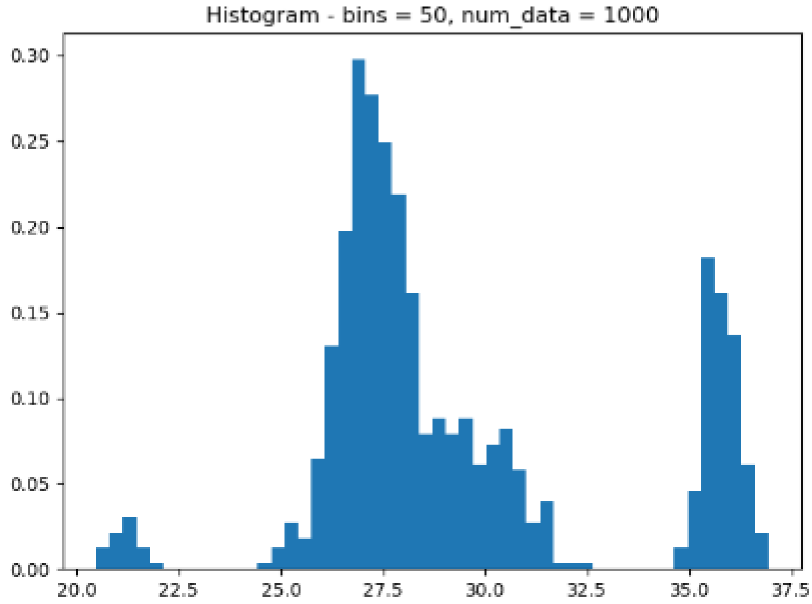
Task 1 - 观察三种方法对于num_data变化的响应

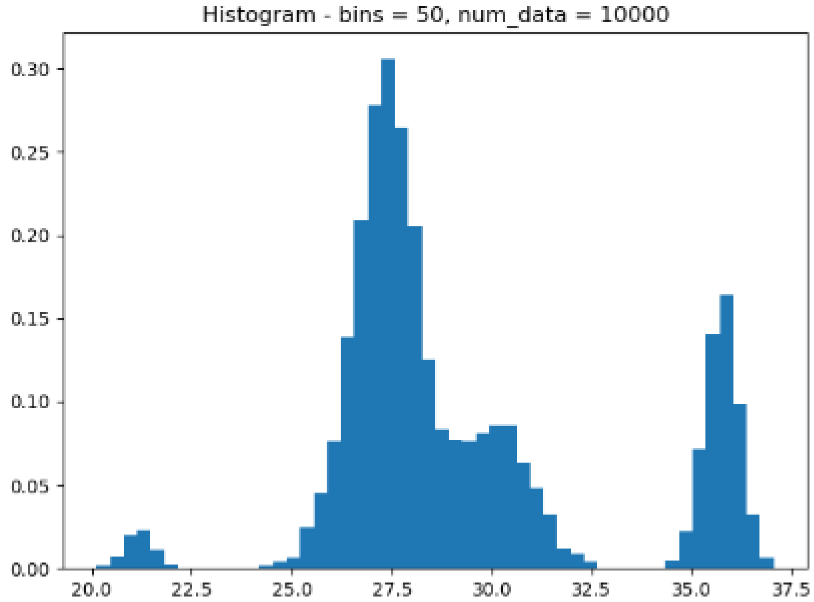
- ### 直方图方法

  以`bins = 50`为例，依次更改`num_data`的值，得到以下4张图。可以看出，当`num_data`的数量增加时，相邻柱体的高度差减少，即整个图像呈现出**更好的连续性**，这说明数据点的增加可以帮助我们更准确地描述概率密度函数。



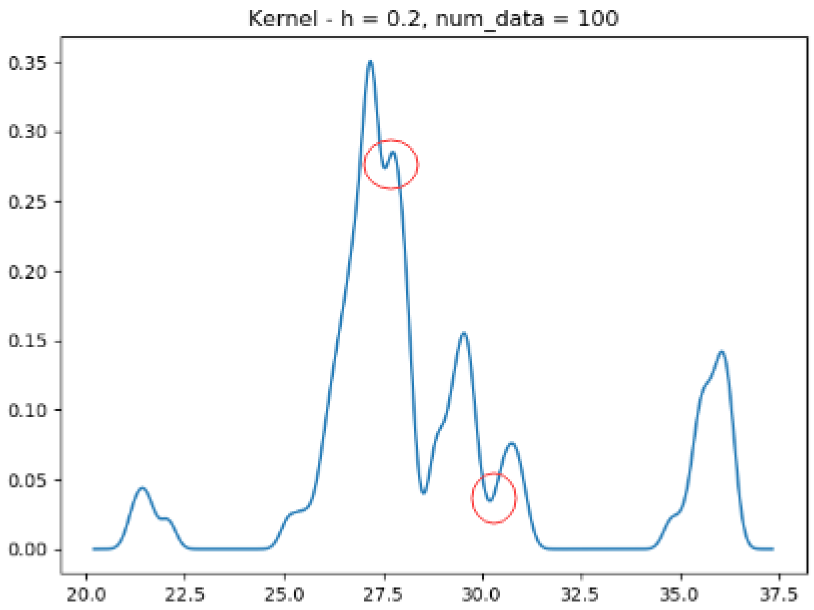


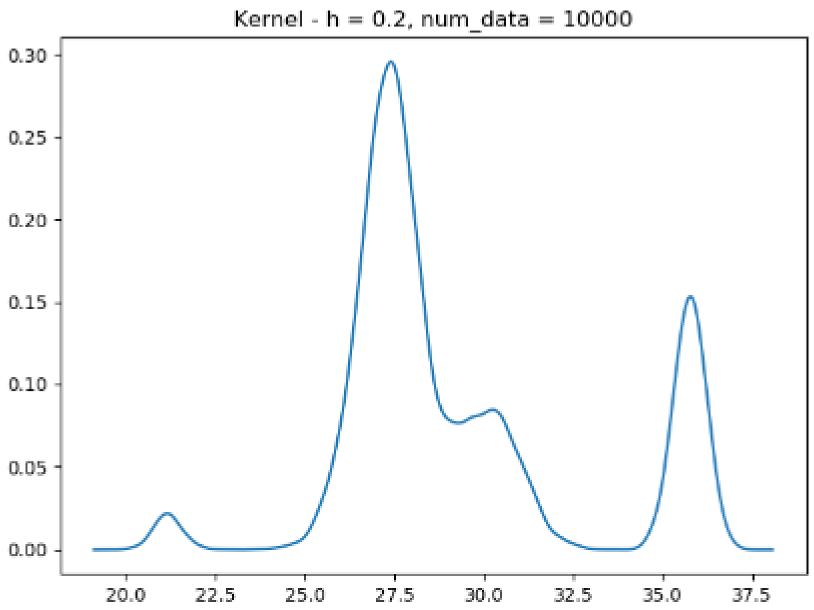

Histogram - bins = 50, num_data = 1000
0.30
0.25
0.20
0.15
0.10
0.05
0.00
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5


Histogram - bins = 50, num_data = 10000
0.30
0.25
0.20
0.15
0.10
0.05
0.00
20.0
22.5
25.0
27.5
30.0
32.5
35.0
37.5

- ## 核密度估计

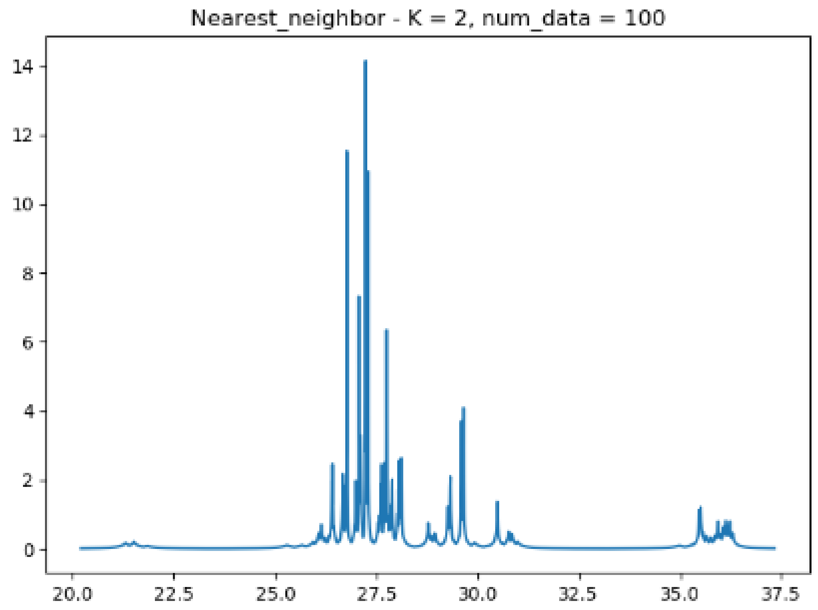
以`h = 0.2`为例，依次更改`num_data`的值进行观察，当它为100和10000时，可以看出图像有明显的区别。`num_data`为100时，图线有很多凹凸不平的地方，而值为10000时，图线就变得很平滑且有更好的连续性，可以帮助我们更准确地估计其概率密度；

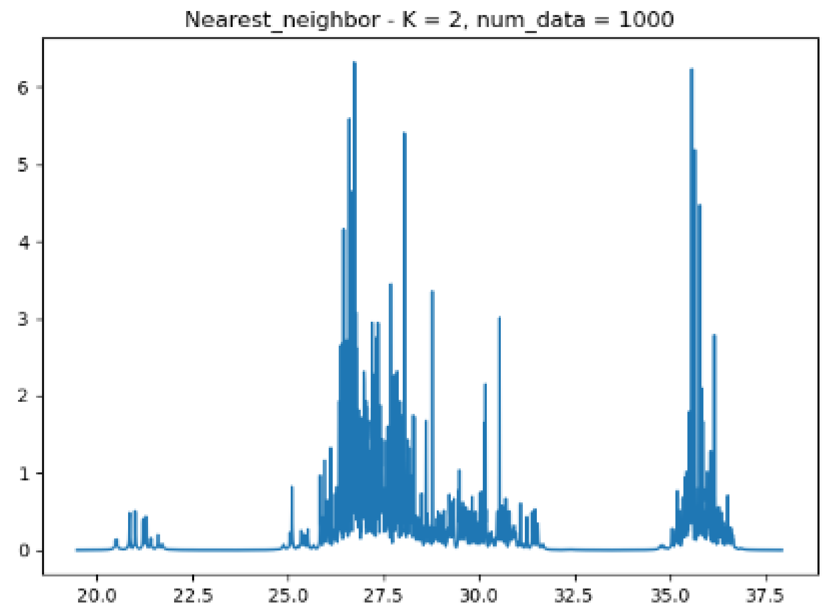


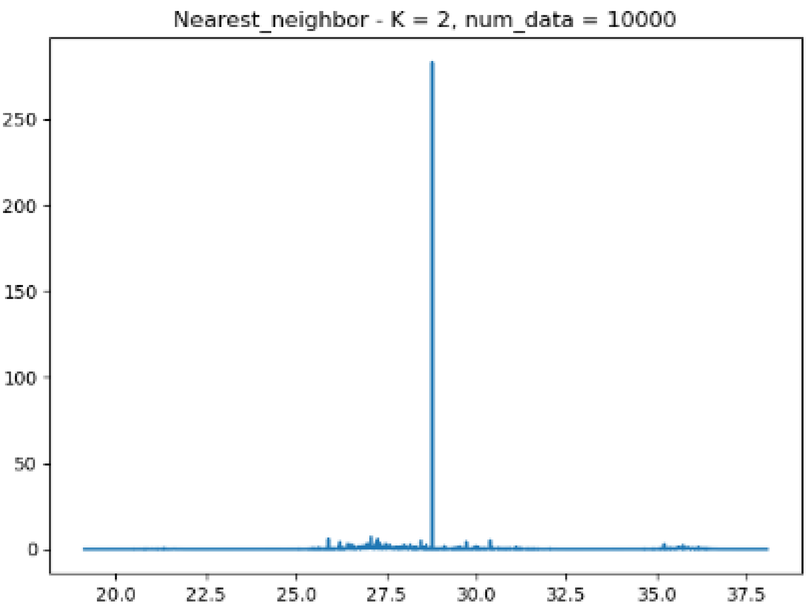


- ## K近邻方法

以`K = 2`为例，依次更改`num_data`的值进行观察。显然，数据点个数增多时，可以帮助我们更准确地估计概率密度，但是当数据点个数增多至非常大（例如 1000）时，个别数据点的噪声就会被放大，会呈现出不适合观察的图案，这是我们估计时需要注意的。


Nearest_neighbor - K = 2, num_data = 100

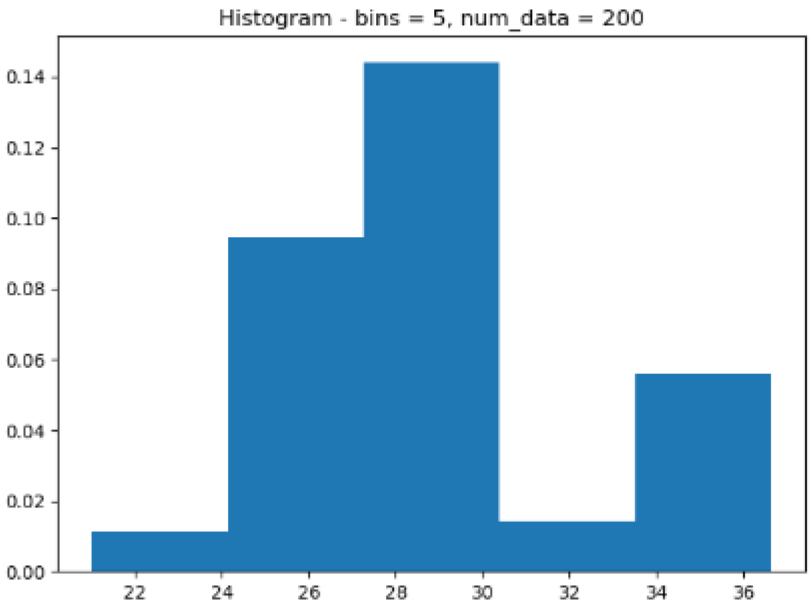

Nearest_neighbor - K = 2, num_data = 1000

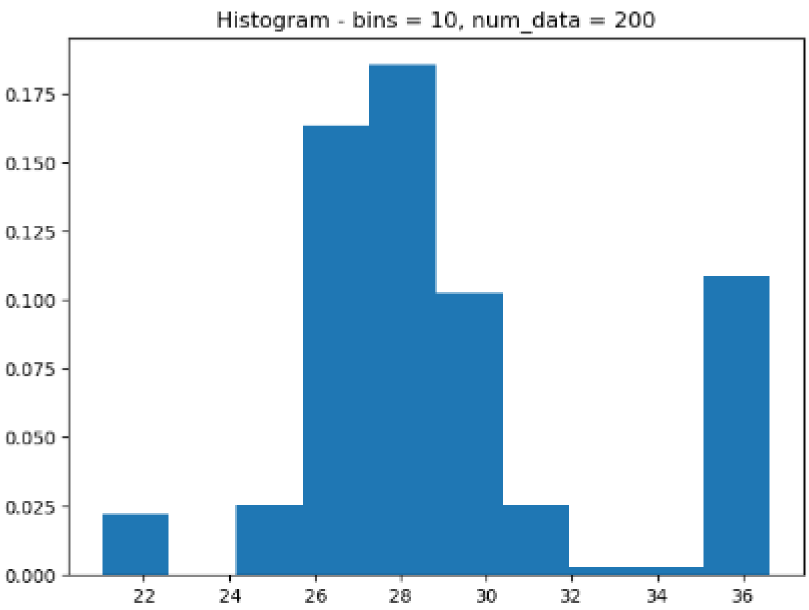

Nearest_neighbor - K = 2, num_data = 10000

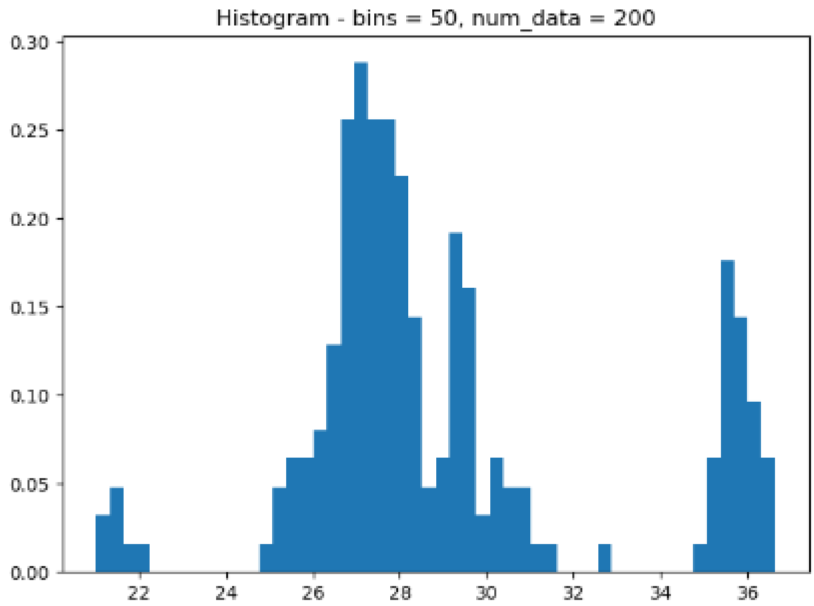
Task 2 - 为直方图法挑选合适的箱数量

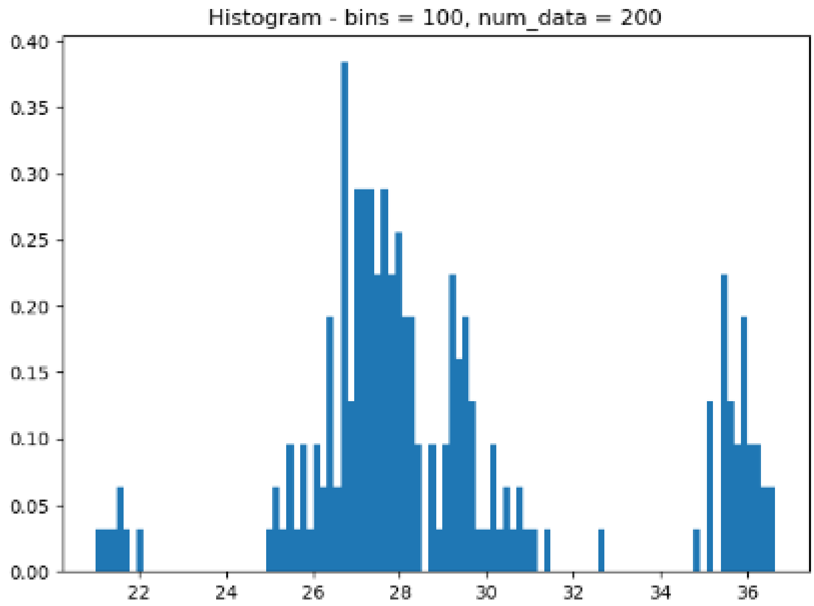
控制变量`num_data`为200，改变箱数量为5、10、50、100、200

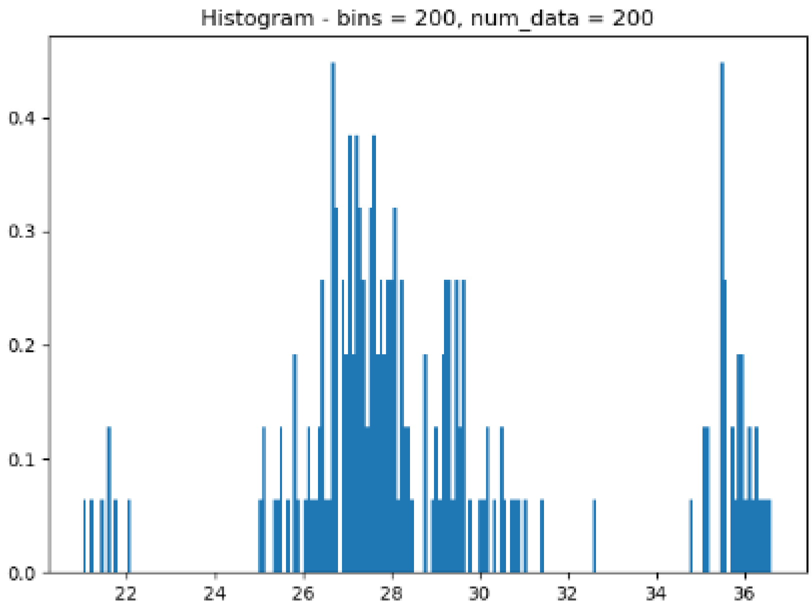
```
if __name__ == '__main__':
    for i in [5, 10, 50, 100, 200]:
        histogram_estimation_graph(200, i)
```



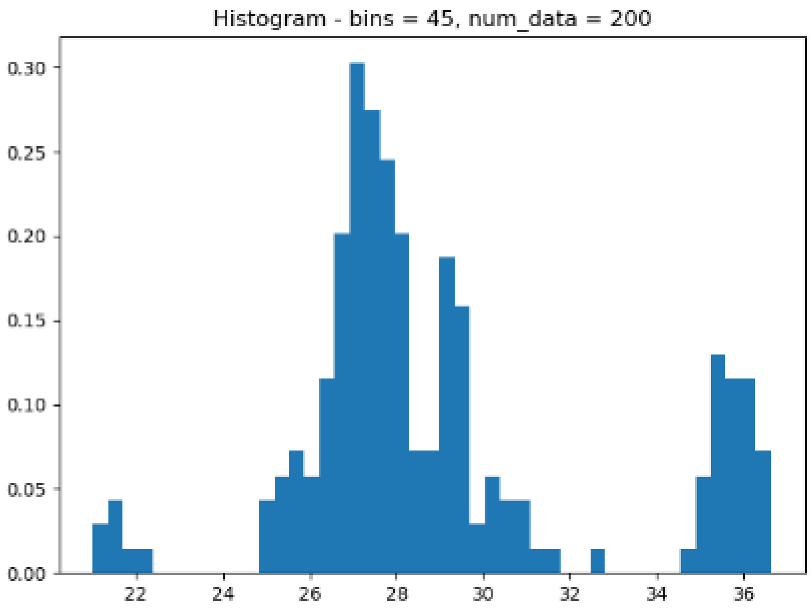



Histogram - bins = 50, num_data = 200
0.30
0.25
0.20
0.15
0.10
0.05
0.00
22
24
26
28
30
32
34
36


Histogram - bins = 100, num_data = 200
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
22
24
26
28
30
32
34
36


Histogram - bins = 200, num_data = 200
0.4
0.3
0.2
0.1
0.0
22
24
26
28
30
32
34
36

- 可以看出，随着`bins`的增大：图像最开始过于平滑无法提供任何信息，之后呈现出一定的变化趋势；当箱数量为50时，图像连续性较好，且尖刺和低坑数量较少；但箱数量达到100和200时，明显看出图像变得尖锐，连续性很差。经尝试，`bins`**在30到50之间，会有较好的估计效果**：



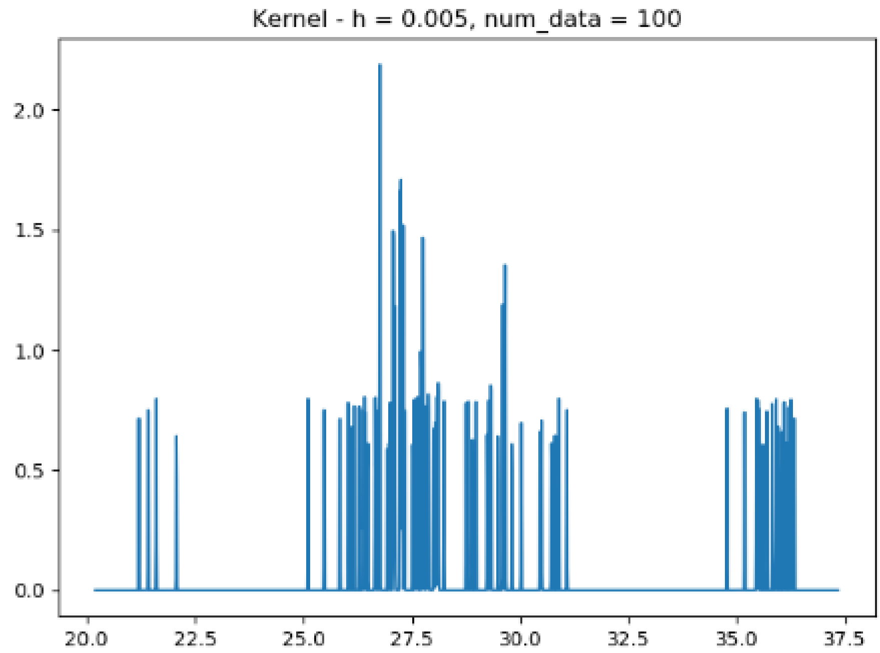
- **总结挑选合适`bins`的方法**：
    - 当图像过于平滑、几乎没有变化趋势时，选择增大`bins`的数量；
    - 在增大箱数量的过程中，注意尖刺和低坑的出现，如果图像连续性受到严重影响，立刻减少`bins`的值；
    - 经试验，当大部分相邻箱子(除趋势变化外)的高度差**小于纵轴的1/3**时，可以达到较好的估计效果。

Task 3 - 为核密度法选择合适的 h

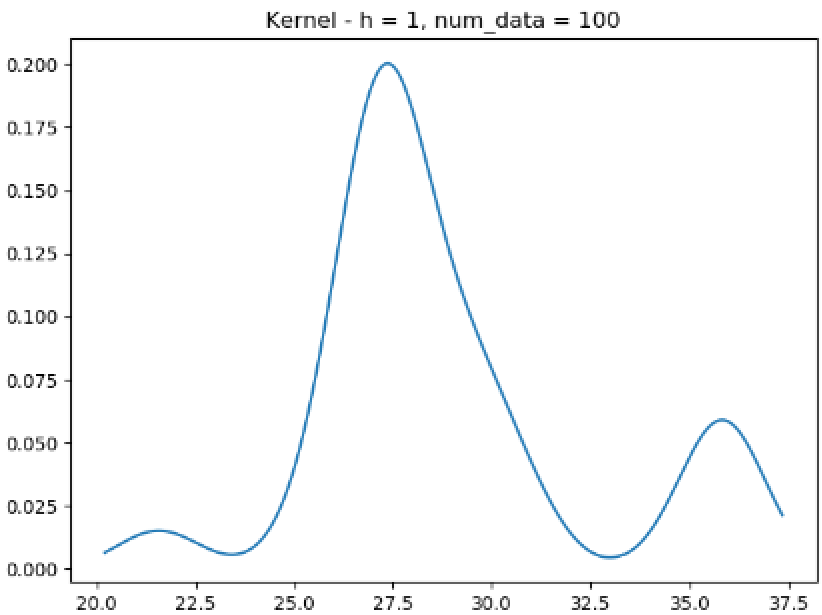
控制变量`num_data = 100`，使 h 分布于不同的数量级，先初步感受一下图像随着h的变化：

```python
if __name__ == '__main__':
    for h in [0.005, 0.07, 0.2, 0.5, 1.0]:
        kernel_estimation_graph(100, h)
```

▼ h = 0.005 以及0.07时，图像很尖锐，而且很难看出规律，这说明h过小将导致噪声放大；



▼ h = 1.0 时，图像虽然很平滑，但是平滑度过高难以看出概率变化的详尽趋势；



为了得到最佳图像，可以用肉眼估计得到较为合理的图像，但不妨使用均平方积分误差，来找到最佳的 h

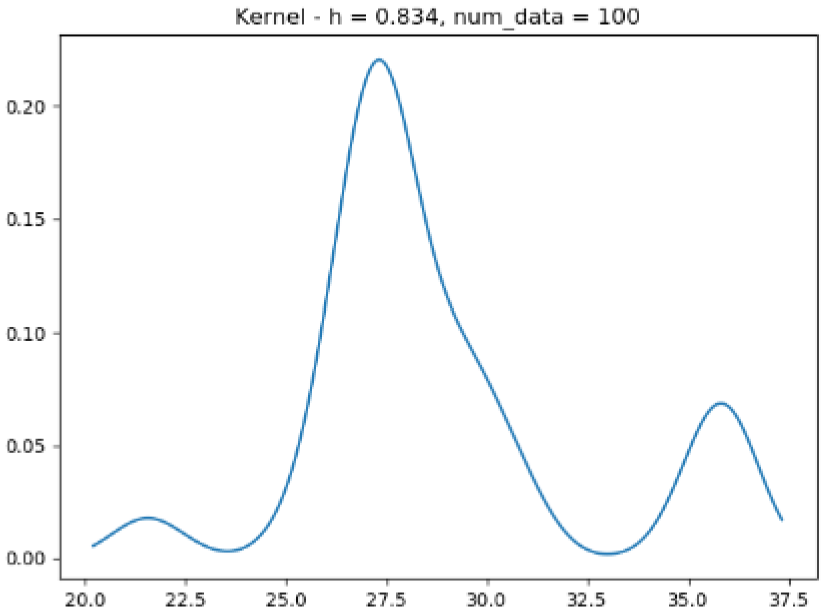
均平方积分误差：

$$MISE(h) = \int (F(x) - f(x))^2 dx$$

由于是高斯核函数，由已有结论：

$$h = (\frac{4\sigma^5}{3n})^{\frac{1}{5}}$$

由于取2000个坐标刻度，且σ = 3.60， 可以计算出最佳的 **h = 0.834**



总结选择合适 h 的方法：

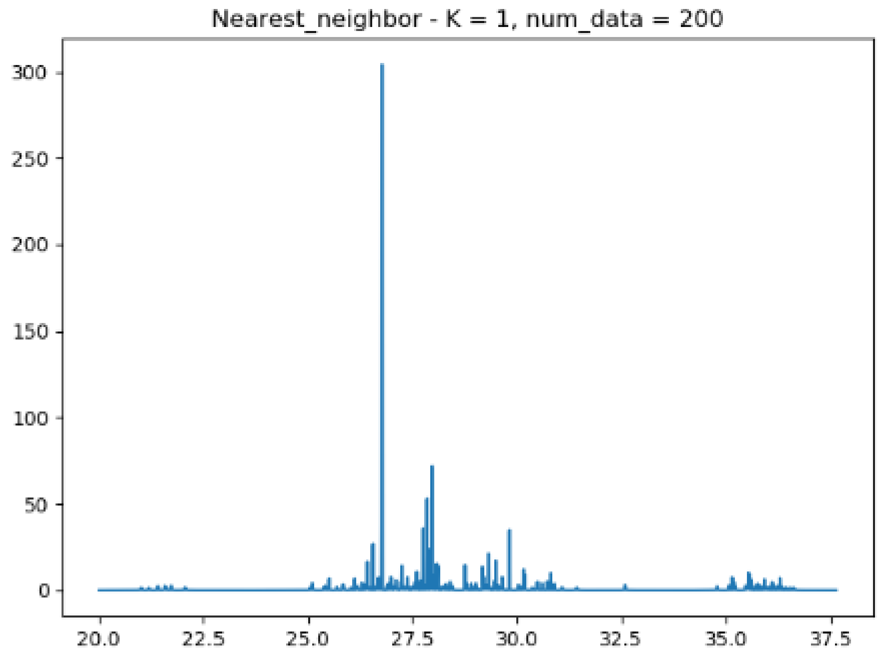
- 首先让h分布在不同数量级（如 0.007 0.05 0.3），剔除图像尖锐和过于平滑的两个极端后，取得一个大致的参数区间，使得图像连续性较好、没有较大落差；
- 在区间内调整h的值，本次实验中，也**结合前两种估计法**，来整合出大致的趋势，避免因为追求图像的平滑度而摒弃正确的概率变化趋势（例如上图中左起第4个山峰），同时也避免了噪声使得图像尖锐的情况；
- 最好**使用误差分析**，来得到最佳图像；
- 注意图像区间的上限和下限，观察handout中的代码，其区间左右各添加了3个标准差的扩展长度，经实验发现并不需要那么长，各**扩展1**即可。总之在画图时，仍需要注意区间边缘的问题。

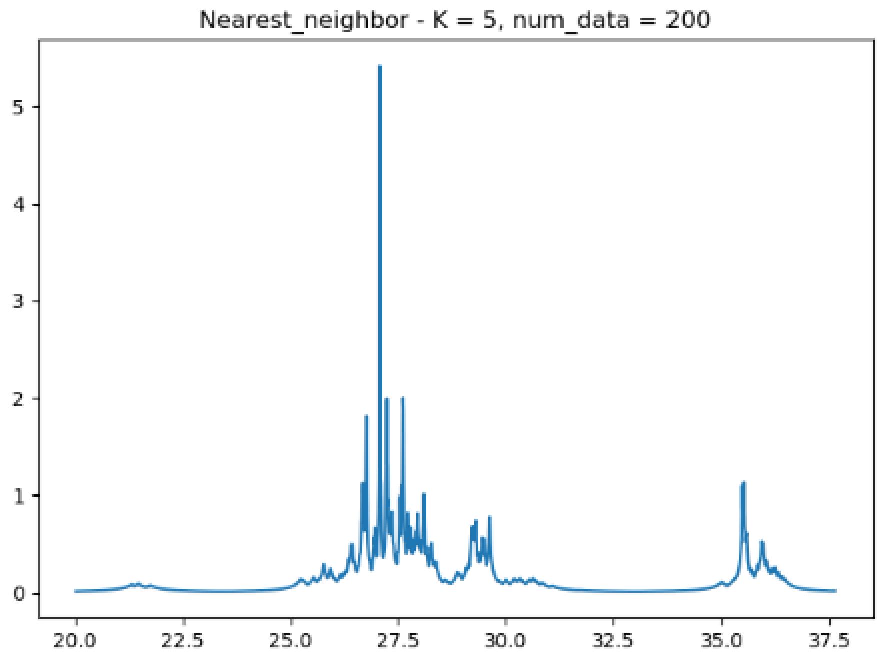
Task 4 - 近邻法作图、非归1解释

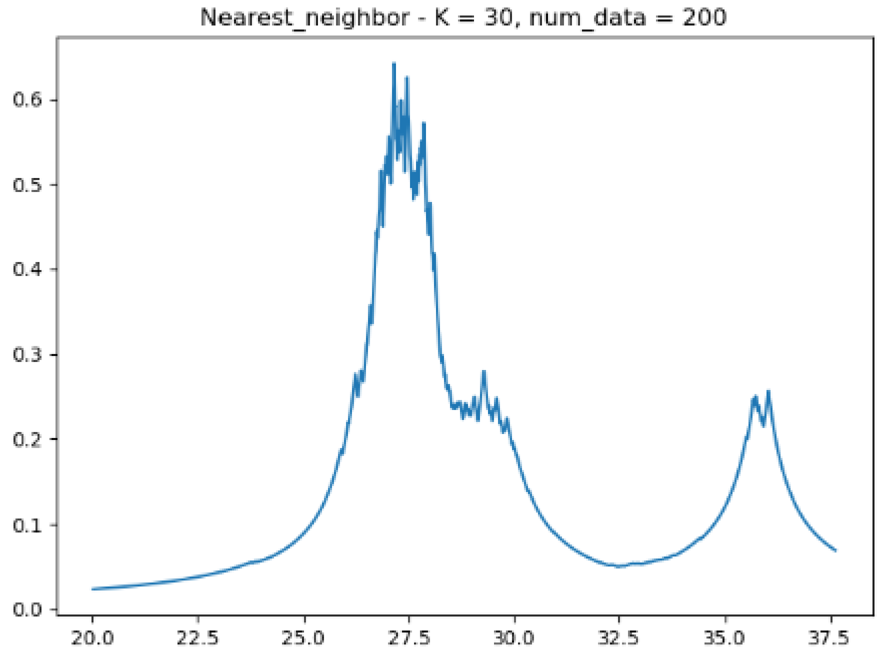
控制变量`num_data = 200`，改变K的值为1、5、30、100：

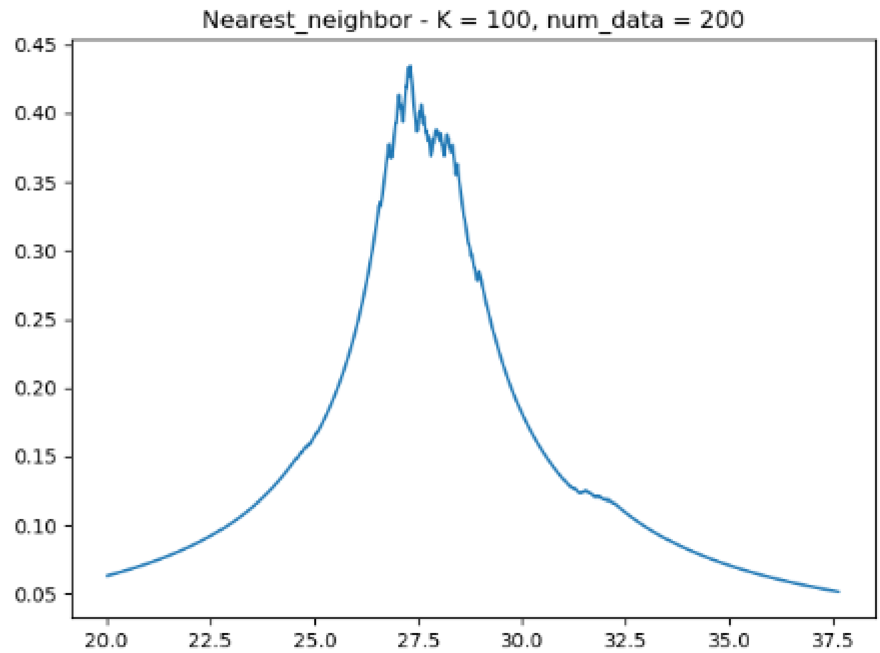
```python
if __name__ == '__main__':
    for K in [1, 5, 30, 100]:
        nearest_neighbor_method_graph(200, K)
```

作图如下：









- 前两张图（K = 1，5）图像激越过高，使得我们很难估计出概率的变化；
- 最后一张图（K = 100）直接抹杀掉了左右两端的变化趋势，显然是不可取的；
- 当 K = 30 时，图像相对正常，但是结合前几种方法的结果，可以知道图像最左边应有一个峰值，并没有表现出来。于是再次调整K的值，发现只有当 K<5 时，左边的峰值才可以展现出来，但这时候图像已经变得十分尖锐了，这也进一步说明了近邻法的弊端：虽然刻画了数据的边缘性，但是忽略了等距情况下图像的趋势变化；

> 一点感想：想要得到较好的估计，需要结合多种方法，综合得到的结果才能弥补各个方法的缺陷，做出正确的估计，仅一种方法做出来的图往往容易出现疏漏

非归1性的解释：

概率密度：

$p(x) = \frac{K}{N*V}$

积分求和：

$\int_{-\infty}^{+\infty} p(x)dx = \int \frac{K}{N*V} dv = \frac{K}{N} \int \frac{1}{V} dv$

显然由基础的积分知识可以得知，$\frac{1}{x}$ 的积分是不收敛的，因此近邻法估计的结果不是真正的概率密度，也不具有归1性，它在整个区间上的积分是发散的。