Non-parametric Density Estimation

I have implemented three non-parametric density estimation algorithm, namely histogram method, kernel density estimation method and the nearest neighbor method, which will be compared and discussed in the following contents.

0. GuiLines of running codes

1. Run the origin histogram. You need to give the --bin_num param.
   ```
   python source.py --num_data=200 --estimation_method=histogram --bin_num=25
   ```
2. Run the different bin rules to do histogram estimation. Bin number will automatically generated by the rules.
   ```
   python source.py --num_data=200 --estimation_method=different_rule_histogram
   ```
3. Run the origin kernel method. You need to give the h by hand.
   ```
   python source.py --num_data=200 --estimation_method=kernel --kernel_h=0.29
   ```
4. Run the improved kernel method, which can automatically optimize the parameter h by maximum likelihood tech.
   ```
   python source.py --num_data=200 --estimation_method=auto_kernel
   ```
5. Run the KNN estimation method. You need to give the k by hand.
   ```
   python source.py --num_data=200 --estimation_method=knn --knn_k=50
   ```

1. Histogram Method

1.1 How does the data number inferences the performance

The principle of histogram is to approximate the probability of a certain cell by means of statistics. Just as the formula reffers

$$(\int_{x_0}^{x_0+\Delta} f(x)dx) * N = n$$

From this point of view, the smaller Δ is, the more accurate estimation will we have. **However, we need to consider the limitation of data number. If we choose a too small Δ (which equals to larger bin number), points in each bin will become so small that we can not achieve enough statistical information to do a proper estimation.**

On another side, if we choose a too large Δ(which equals to less bin number), obviously we can neither do density estimation properly.

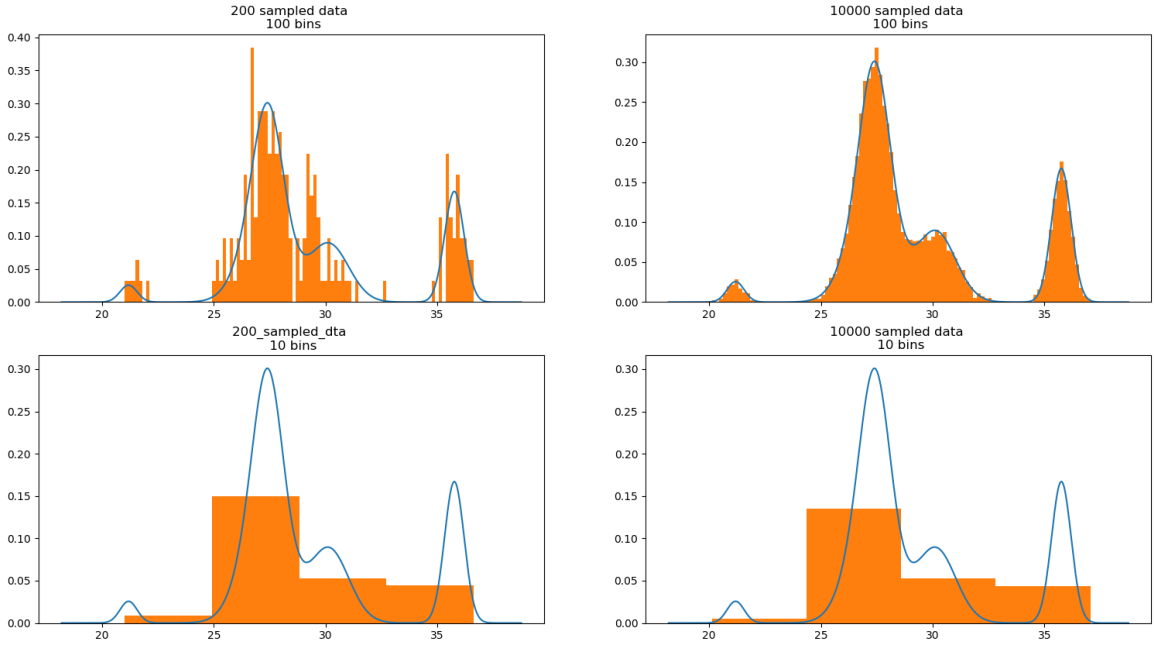

Figure 1. The graph shows that too many bins will bring some noise when the data number is limited. By the way, too few bins will also make the histogram too smooth, lossing a lot of information.

So We can have the following conclusions:

- **More training data helps, because we can set a smaller bin width and gain enough statistic information in each bin. So in the light of a same distribution, the more sample data we have, the larger bin number(smaller bin width) should be chosen.**
- **In the light of limited data points, too many or too few bins will harm the estimation performance. How many bins should we choose is an important problem of histogram estimation method. This will be discussed in $1.2**

In the following experiments, we just simply fix num_data = 200, 1000, 10000 as the "Report Requirements" asks.

1.2 Strategy of choosing bin number

In **$ 1.1**, we have figured out that the number of bins is related to the number of data. So we believe there exists some mapping relation between them. θ is added reffering some other related information.

$$bin\_num = f(data\_number, \theta)$$

I have found different kinds of rules of calculating bin number. **Some of them are rules of thumb and others are trying to minimize estimated risk functions.** In this part of the report, I will show some and do experiments on them.

First we suppose h is the width of each bin and k is the number of bins. h can map to k in such a way:

$$k = \lceil (max\_x - min\_x)/h \rceil$$

and n is the number of all the data.

Square-root choice

$$k = \lceil \sqrt{n} \rceil$$

Sturges' formula

$$k = \lceil \log_2 n + 1 \rceil$$

Doane's formula

$$k = 1 + \log_2(n) + \log_2(1 + \frac{|g1|}{\sigma_g})$$

$$g1 = \frac{\Sigma_i^n x_i^3}{(\Sigma_i^n x_i^2)^{2/3}}$$

$$\sigma_g = \sqrt{\frac{6 * (n-2)}{(n+1)(n+3)}}$$

Rice Rule

$$k = \lceil 2n^{1/3} \rceil$$

Scott's normal reference rule

$$h = \frac{3.5\hat{\sigma}}{n^{1/3}}$$

$$k = \lceil (max\_x - min\_x)/h \rceil$$

where $\hat{\sigma}$ is the sample standard deviation.

Freedman–Diaconis' choice

$$h = 2\frac{\mathrm{IQR}(x)}{n^{1/3}}$$

$$k = \lceil (max\_x - min\_x)/h \rceil$$

where IQR reffers to interquatile range

Shimazaki and Shinomoto's choice

$$\arg\min_h \frac{2\bar{m} - v}{h^2}$$

$$k = \lceil (max\_x - min\_x)/h \rceil$$

$$\bar{m} = \frac{1}{k}\Sigma_{i=1}^{k} m_i$$

$$v = \frac{1}{k}\Sigma_{i=1}^{k} (m_i - \bar{m})^2$$

1.3 Result

Rule	bin_num (sample data 200)	bin_num (sample data 1000)	bin_num(sample_data 10000)
Square-root choice	15	32	100
Sturges' formula	9	11	15
Doane's formula	10	12	15
Rice Rule	11	19	43
Scott's normal reference rule	8	13	30
Freedman–Diaconis' choice	10	16	37
Shimazaki and Shinomoto's choice	26	44	69

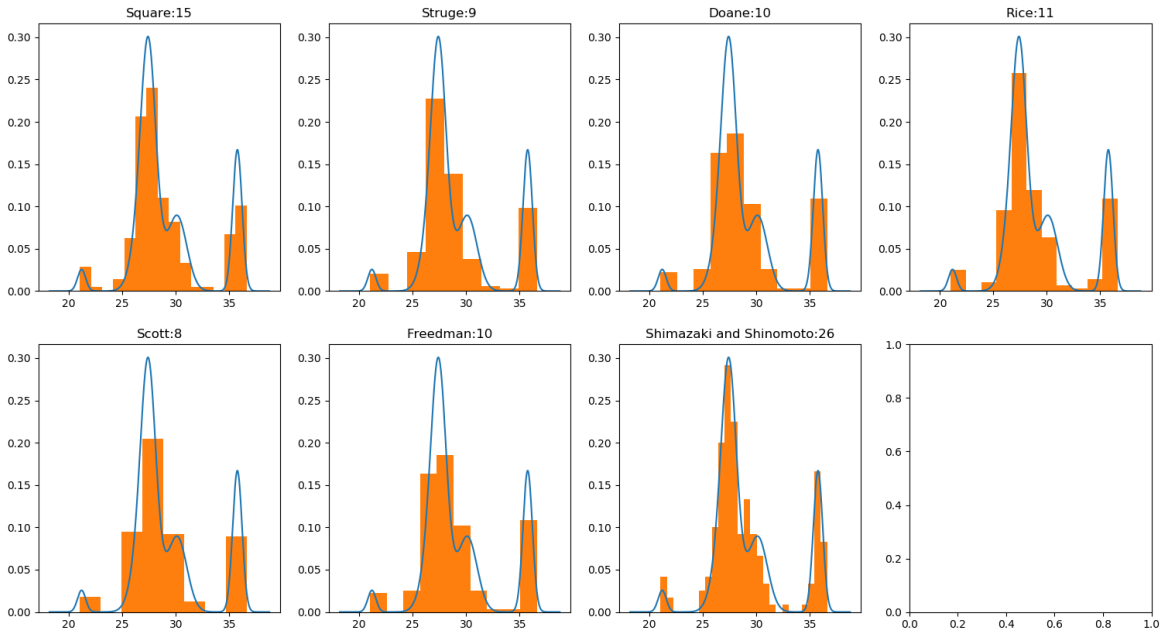

Figure 2. The graph shows us the result of different types of choosing ways. The data number is fixed as 200.

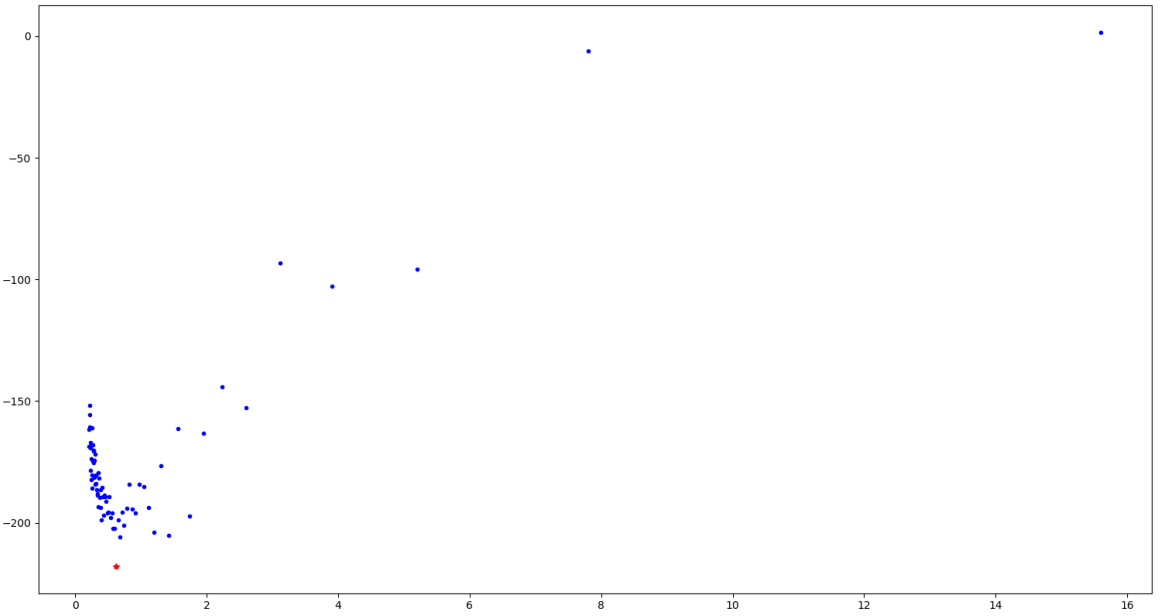

Figure 3. The graph shows us the process of minimizing the estimaed risk function of Shimazaki and Shinomoto's choice method.

1.4 Analyse and Conclusion

As you can see, different rules give different bin numbers. **It is hard to judge which rule is actually the best since different bin sizes may reveal different features of the data.**

As for myself, I prefer Shimazaki and Shinomoto's choice. And if the data spreads in a higher dimension space and the number of the data is larger, genetic algorithm can be considered to calculate a not bad result.

2. Kernel Density Estimation Method

Gaussian function is chosen to be the kernel function in our experiments, which gives a softer result than the hard cube kernel performs.

2.1 How does the data number inferences the model

The experiments' results (reffered in $2.3) show that more example data points really do some good for density estimation, as their estimated density curve is closer to the real curve.

2.2 Strategy of choosing h

In [1], the author includes many strategies to determine h. One of the choosing ways is using **Maximum likelihood cross validation**.

For each points, we can estimate its value by such formula:

$$\hat{f}_{h,i}(X_i) = \frac{1}{(n-1)h}\Sigma_{j\neq i}K(\frac{X_j - X_i}{h})$$

Then by using maximum likelihood tech, we can transform the problem into an optization problem.

$$h_{mlcv} = argmax_{h>0}MLCV(h)$$

$$st. MLCV(h) = (n^{-1}\Sigma_{i=1}n\lg(\Sigma_{j\neq i}K(\frac{X_j - X_i}{h}) - \lg((n-1)h))$$

In my experiments, I used **skipy.optimize.minimize** function to help me optimize the parameters **(minimize (-1)*MLCV)**. **By the way, changing the loop operations into matrix operations in python will actually fasten the speed of optimization process a lot.**

2.3 Result

example data numbers	best_h
100	0.35037849
200	0.37750606

example data numbers	best_h
2000	0.14217995
10000	0.16274119

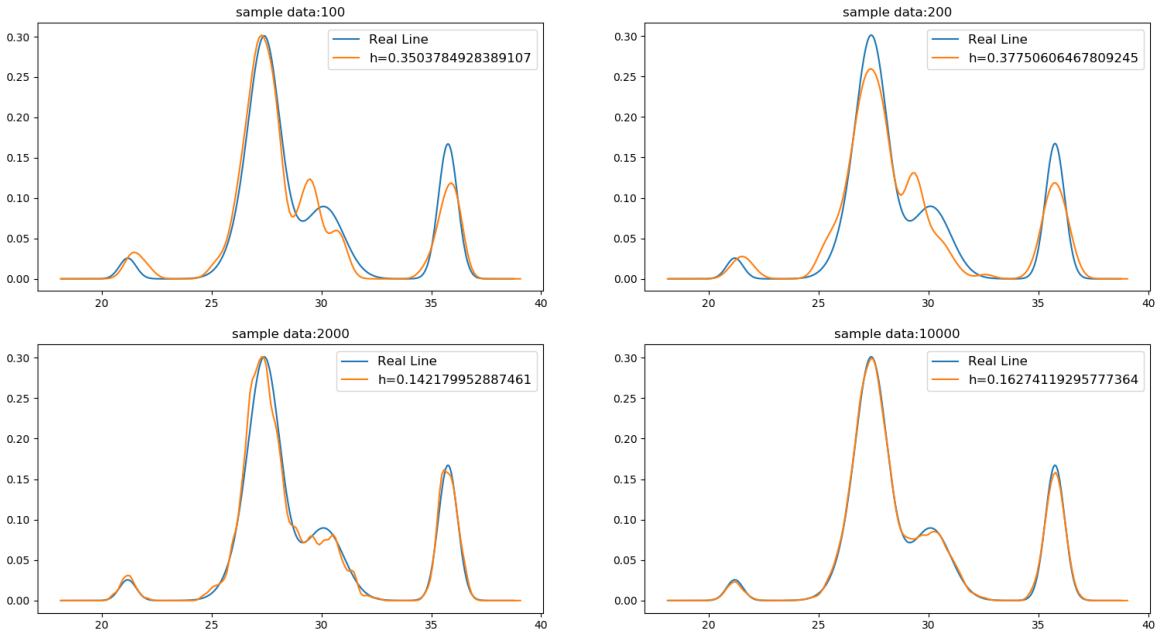

Figure 4. The graph shows different kernel estimation results under different sample sizes. Parameter h is determined by the method mentioned in $2.2.

3. K-Nearest Neighbors Method

3.1 How does the data number inferences the model

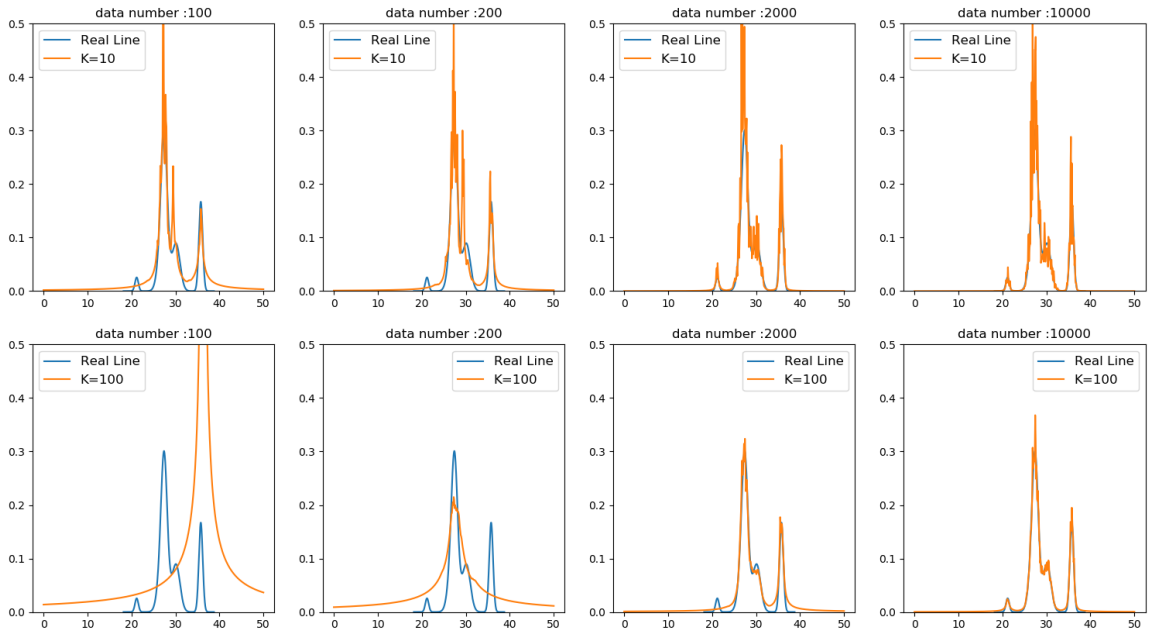

Figure 5. The graph shows the inference of example data number. In order to mine the rules better, K is fixed to 10 and 100.

As we can see in Figure 5, **more data will help us grasp more details of the density, but more noise is introduced at the same time. On the contrary, if the data size is small, although the prediction line will be smoother, we will lose lots of information about the origin distribution.**

3.2 Exploration of Convergence

Obviously **KNN method will not converge to 1 if we sum the probability mass over all the space**. Here I will demonstrate it.

First, we estimate each position density by such formula:

$$f(x) = (\frac{k}{N}) * \frac{1}{V}$$

In 1-D space, we can split the data set A into B and C, where A = B+C and C contains all the points who are larger than any of the sample data points. As $p(x_i) \geq 0$. We can ignore the distribution of B directly. So we can derive the following formula:

$$\Sigma p(x_i) = \Sigma_B p(x_i) + \Sigma_C p(x_j) \geq \Sigma_C p(x_j) = (\frac{k}{N}) \Sigma_C \frac{1}{V_i} \equiv \infty$$

hints: Focused on Set C, the K neariest points are always the same. So the speed of V increasement can be seen as linear growth.

As is demonstrated above, the KNN method won't converge to 1 if we try to sum the probability mass over all the space

3.3 Effection of k

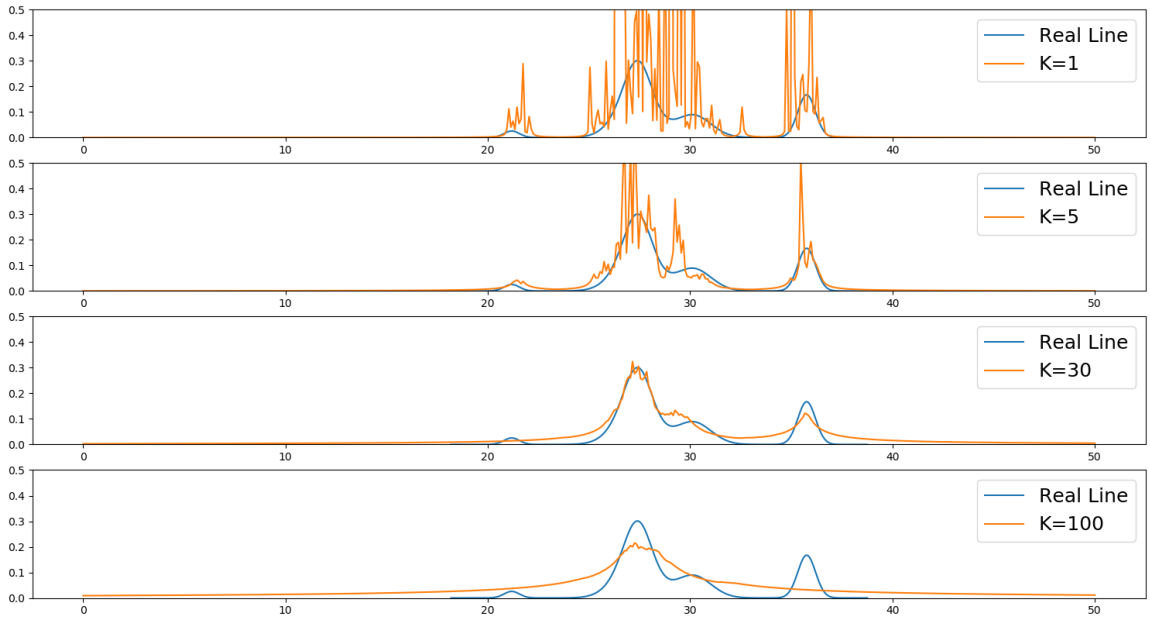

Figure 6. The graph shows performances of different k. The sample data number is kept the same: 200.

Figure 6 shows us the effects of k. We can firstly focus on the left peak of the real line. When we set k as a small number, the left peak can be detected by our model, but more noise will be taken in at the same time. On the contrary, the larger k we set, the smoother but less sensitive line will we get.

3.4 Algorithm Implelmentation

If we run through all the data points when estimating each position, the complexity of the algorithm will be $\Theta(n^2)$. But in this project all the data points are 1-D, so we sort each data point first, and then ultilize the order information to reduce the complexity of the algorithm to $\Theta(n \lg(n))$.

But we should also consider high-dimensional cases. When the problem is facing to high-dimentional data, order information won't be so easy to be ultilized. There are two algorithms that should be noticed. **KD Tree** and **Locality-Sensitive Hashing**.

I have implemented an enhensed version of **locality-sensitve hashing** in my source code, which needs $\Theta(n)$ time to build hash tables and can only needs $\Theta(\lg(n))$ to find the neariest points because the searching space will be reduce to $\Theta(\lg(n))$.

Reference

[1] Guidoum A C. Kernel estimator and bandwidth selection for density and its derivatives[J]. The kedd package, version, 2013, 1(3).
[2] Datar M, Immorlica N, Indyk P, et al. Locality-sensitive hashing scheme based on p-stable distributions[C]//Proceedings of the twentieth annual symposium on Computational geometry. ACM, 2004: 253-262.
[3] Birgé L, Rozenholc Y. How many bins should be put in a regular histogram[J]. ESAIM: Probability and Statistics, 2006, 10: 24-45.
[4] http://www.neuralengine.org//res/histogram.html
[5] https://en.wikipedia.org/wiki/Histogram