Assignment 01

Non-parametric Methods

Anonymity

Department of Computer Science, Fudan University

March 19, 2019

1 Overview

1.1 Introduction

In Lecture 2, we have discussed non-parametric methods, which have received considerable interest from scientific community in the past decades.

Parametric Approach is a branch of statistics, which utilizes specific probability distributions and a fixed set of parameters to estimate the population of sampled data [1]. Despite the simple mathmatical form of parameters, an important limitation of this approach is that the chosen density might underfit the underlying distribution. For instance, Gaussian distribution, which is necessarily unimodal, has a poor performance when estimating mutimodal distribution. To overcome these drawbacks, **Non-parametric Methods** have been proposed, which make fewer assumptions about the form of the distribution. As will be shown in this report, non-parametric methods can capture various aspects of distributions.

1.2 Assignment Description

In this assignment, we are going to use the three non-parametric density estimation algorithms (namely histogram method, kernel density estimate and the nearest neighbor method) to estimate the distribution of the given data set.

1.3 Outline

The rest of the report is organized as follow: In Section 2, we start with the theory and implementaion of **Histogram** methods, which is trivial but fundamental among non-parametric methods. Two advanced methods, i.e., **Kernel Density Estimation** and **K-Neareast-Neighboor Method**, will be considered in Section 3 and 4, respectively. In each section, we will discuss the problem raised in assignment requirements and try to give a reasonable solution. Besides, some more efficient implementations of these methods will be proposed in Sec. 6. For your ease, you can only focus on sections answering the questions in requirements. All figures are .eps files, which means, you can enlarge them with any scale.

2 Histograms

In this section, we give a brief introduction to the basic histograms method and discuss its drawbacks for further improvements.

2.1 Theory

Histogram Method is an accurate representation of sampled date and can estimate the probability density distribution of a continuous variable, which was first introduced by Karl Pearson in 1895 [2].

A histogram is a function n_i that counts the number of observations that fall into each of the disjoint categories (known as bins), whereas the graph of a histogram is merely one way to represent a histogram. Thus, if we let N be the total number of observations and k be the total number of bins, the histogram n_i meets the following conditions:

$$N = \sum_{i=1}^{k} n_i \tag{1}$$

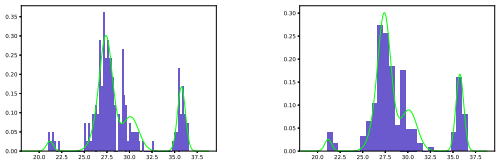
Given N sampled data of a single continuous variable x, the steps of building histograms are listed as follow:

1. Divide x into disjoint bins of width Δ_i.
2. Count the number n_i of observations of x falling in bin i.
3. Estimate the probability density for each bin using $p_i = \frac{n_i}{N\Delta_i}$.

This gives a model for the density $p(x)$ that is constant over the width of each bin, thus it is easily seen that $\int p(x) = 1$. Often, the bins are chosen to have the same width $\Delta_i = \Delta$.

2.2 Algorithm Implementation

In Python, matplotlib package has already provided a convenient tool to plot a histogram with specific bins. We can simply use the ***matplotlib.pyplot.hist*** method to calculate and plot our histograms.



(a) $k = 75$ (b) $k = 25$

(c) $k = 12$ (d) $k = 5$

Figure 1: **Histograms** with various values of k (shown by slateblue) and true distribution (shown by green)

2.3 Answer to Requirement 2

Requirement 2: *For histogram estimation, you could vary the number of bins used to locate the data samples to see how this parameter affect the estimation. Please answer: how could you pick the best (or good) choice for this number of bins?*

For this requirement, we first show some examples of histograms with various numbers of bins, which is denoted by k. As illustratred in Fig.1, with small values $k = 5$, the result graph is too smooth to capture the multimodality of underlying distribution, whereas the histogram with large values $k = 75$ becomes spiky and is sensitive to noise. Thus, moderate values of k should be chosen to better model real distributions. **Specifically, the value $k = 12$ and $k = 15$ are both appropriate for this problem.**

Our analysis above is presumably designed on aesthetic grounds: just picking the smoothing parameter leading to good-look results. There are, however, various useful guidelines and rules of thumb [3]. In practice, we can use model validation techniques, such as **cross-validation**, to pick the optimal value of k. In detail, we first divide the sampled data into training set and validation set and then draw histograms with distinct values of k. After that, we calculate the probability for validation set and select the hyper-parameter k with highest likelihood on validation set.

2.4 Drawbacks

Although it gives a rough sense of the density of underlying distribution, there are still many drawbacks of Histogram Method:

1. The estimated density is discontinuous on the bin edges.
2. The number of bins will exponetially increase when considering higher dimensional space.
3. Large number of data is neccessary for estimating local probability density in high dimensional cases.

3 Kernel Density Estimation

3.1 Theory

The intuition behind Histogram Method is to use locality to estimate probability at a particular location. With this idea, we can obtain an estimating formula in high dimension cases:

$$p(\boldsymbol{x}) = \frac{K}{NV}, \tag{2}$$

where V denotes the volume of local region $\mathcal{R}$ around $\boldsymbol{x}$ and K is the number of sampled data in this region.

The core idea in **Kernel Density Estimation(KDE)** is to fix the region $\mathcal{R}$ and count the number K of data in this area, which can be interpreted as

$$K = \sum_{n=1}^{N} k(\frac{\boldsymbol{x} - \boldsymbol{x}_n}{h}), \tag{3}$$

and

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k(\frac{\boldsymbol{x} - \boldsymbol{x}_n}{h}), \tag{4}$$

where $k(\cdot)$ is the kernel function defined to measure whether the point $\boldsymbol{x}$ is in region $\mathcal{R}$.

In order to obtain a smoother model, a common choice for the kernel function is the Gaussian, which gives rise to the following kernel density model

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{(2\pi h^2)^{1/2}}\exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{x}_n\|^2}{2h^2}\right\}, \tag{5}$$

where h represents the standard deviation of Gaussian, which is also called bandwidth for KDE.

By now, a rich basket of kernel density estimators exist and [4] is a good introduction for these functions. As shown in [5], however, the selection of bandwidth h is much more important for the behaviour of $p(\boldsymbol{x})$ than the choice of $k(\cdot)$.

3.2 Algorithm Implementation

Here we provide a pseudo-code for the naive implementation of KDE, as shown in Alg. 1.

ALGORITHM 1: Naive KDE Implementation

Input : A data set $\mathcal{S}$ with N samples from an unknown 1-D distribution

Output : An estimatation of probability density distribution

Generate a test set $\mathcal{T}$ with M points uniformly for the purpose of plotting estimating distribution
for $i = 1$ *to* M **do**
  for $j = 1$ *to* N **do**
    $p(i) \leftarrow \frac{1}{N(2\pi h^2)^{1/2}}\exp\left\{-\frac{\|\mathcal{T}(i) - \mathcal{S}(j)\|^2}{2h^2}\right\}$
  end
end
return a plot figure using test set $\mathcal{T}$ and its estimation

Since it costs $O(N)$ time for each point in test set $\mathcal{T}$, the total time complexity for this approach is $O(MN)$. For further improvement, I refer you to Sec. 6.1.

3.3 Answer to Requirement 3

Requirement 3: *For kernel density estimation, you should try the Gaussian kernel. Please also try to tune h to see what will happen, answer if you have a clue of how to choose h, plot the best estimate you could achieve with* **num_data=100**.

3.3.1 Cross Validation

Cross Validation [6, 7] is a common-used approach to test the model's ability to predict new data that was not used in estimating and give an insight on how the model will generalize to an independent dataset(i.e., an unknown dataset). In this assignment, we use the k-fold cross validation for our model selection.

First, we randomly partition our sample data into k equal sized subsamples. Among the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k-1$ subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data.

To measure the performance of estimated distribution on validation set, we invoke the *relative entropy* function [8]:

$$KL(p_h||p) = -\int p(x) \ln \left\{ \frac{p_h(x)}{p(x)} \right\} dx, \tag{6}$$

where we use $p(x)$ to denote the true distribution and $p_h(x)$ to denote the probability density of x given in Eq.(5). Note that the relative entropy satisfies $KL(p_h||p) \geq 0$ with equality if, and only if, $p_h(x) = p(x)$.

Since $\mathcal{T}_i$ is sampled from the true distribution, we can use our observations on all validation sets $\mathcal{T}_i$ to approximate the expectation of $\ln \left\{ \frac{p_h(x)}{p(x)} \right\}$:

$$KL(p_h||p) \approx -\frac{1}{N} \sum_{i=1}^{k} \sum_{x \in \mathcal{T}_i} \ln \left\{ \frac{p_h(x)}{p(x)} \right\}, \tag{7}$$

where $p_h(x)$ should be estimated by corresponding training set.

For the purpose of choosing the best estimation, we should select the bandwidth h minimizing the relative entropy $KL(p_h||p)$, that is:

$$\begin{aligned}
&\arg\max_h - KL(p_h||p) \\
=&\arg\max_h \sum_{i=1}^{k} \sum_{x \in \mathcal{T}_i} \ln \left\{ \frac{p_h(x)}{p(x)} \right\} \\
=&\arg\max_h \sum_{i=1}^{k} \sum_{x \in \mathcal{T}_i} \ln p_h(x) - \sum_x \ln p(x) \\
=&\arg\max_h \sum_{i=1}^{k} \sum_{x \in \mathcal{T}_i} \ln p_h(x)
\end{aligned}$$

Not supurisingly, this result is identical with our intuition that the optimal h will maximize the likelihood of test data.

We set $k = 5$ and evaluate the likelihood of 50 distinct values of h within range [0.1,0.6] and then depict the likelihood curve in Fig. 2. The maximum likelihood on validation set is obtained when $h = 0.3959$.

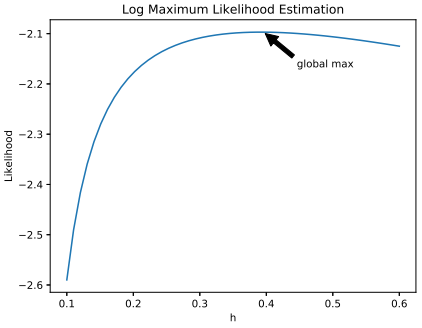


Figure 2: Log likelihood as a function of the bandwidth h

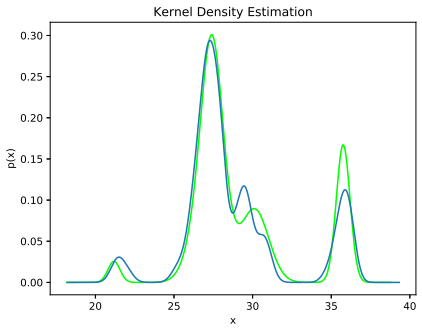


Figure 3: Best estimate with the bandwidth $h = 0.3959$

With the best bandwidth, we re-train our model with all 100 sample data and draw the estimating distribution in Fig. 3.

3.3.2 Cheat with true distribution

This is a very wrong section, since we have used the true distribution! The only purpose of this section is to compare the performance of cross-validation with the best estimation obtained with true distribution.[1] Okay, let's cheat!

Since we have already known the underlying distribution, we can measure the deviation of our estimate distribution according to it. The only thing we should do is to choose the loss function. Here we use the **Least Square Error** function:

$$L(p_h) = \int (p_h(x) - p(x))^2 \, dx, \tag{8}$$

where $p(\cdot)$, $p_h(\cdot)$ denote the true distribution and estimate distribution respectively. To avoid the troublesome integral, we only consider the square error on our test data:

$$L(p_h) \approx \sum_{i=1}^{M} (p_h(x_i) - p(x_i))^2, \tag{9}$$

[1] In fact, the main reason is that I have wrongly used true distribution for bandwidth selection before the published issue.

where M is the total number of test data and x_i is the i-th one.

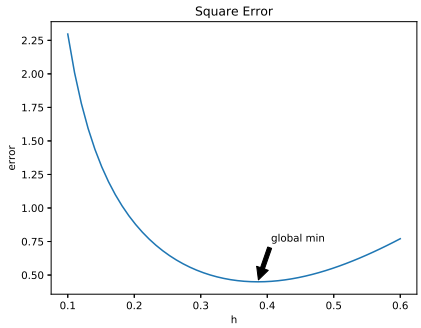


Figure 4: Square Error as a function of the bandwidth h

We have picked 50 points as the bandwidth h within the range $[0.1, 0.6]$ and calculate their loss function, whose curve is depicted in Fig. 4.

As pointed in Fig. 4, the optimal bandwidth h is chosen around 0.3857. Hence, we use hyper-parameter $h = 0.3857$ to plot our best estimate in Fig. 5. Obviously, this method managed to capture the multimodal nature despite some spikes near the modes.

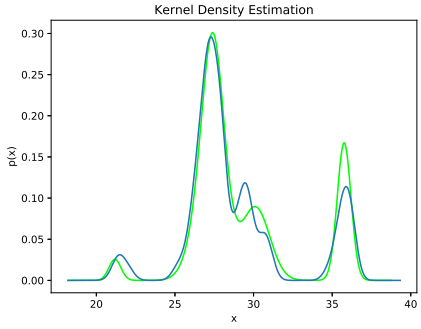


Figure 5: Best estimate with the bandwidth $h = 0.3857$

In spite of its excellent performance on test set, this selection stratagy cannot apply to cases in practice due to the unknown underlying distribution $p(x)$. It is noticed that optimal values of h obtained by two methods is close and their estimation distribution is nearly identical. Consequently, we conclude that cross-validation is a practical and efficient method for hyper-parameter selection.

4 K-Neareast-Neighboor Method

As shown in Sec. 3.3.2, the performance of KDE highly depends on the choice of bandwidth h, the optimal for which may relate to location within the data space. In this section, we address this issue by introducing a novel method called **K-Nearest-Neighboor Method(k-NN)**.

4.1 Theory

We first return to our general formula (2) for local density estimation. Instead of fixing V, we now consider a fixed number K and let sampled data determine the region volume V. For this purpose, we assume a small sphere centered on the point $\boldsymbol{x}$ at which we wish to estimate the density $p(\boldsymbol{x})$, and we allow the radius of the sphere to grow until it contains precisely K data points. In this way, the estimation of the density $p(\boldsymbol{x})$ is then given by Eq.(2) with V set to the volume of the resulting sphere.

4.2 Algorithm Implementation

Since this assignment only fouces on the 1-D case, we design our algorithm for this case. Notwithstanding, the naive implementation can be easily generalized to high dimensional space.

The naive implementation of **k-NN** algorithm is straightforward, the pseudo-code of which is given in Alg. 2.

ALGORITHM 2: Naive k-NN Implementation

Input : A data set $\mathcal{S}$ with N samples from an unknown 1-D distribution and an integer K

Output : An estimatation of probability density distribution

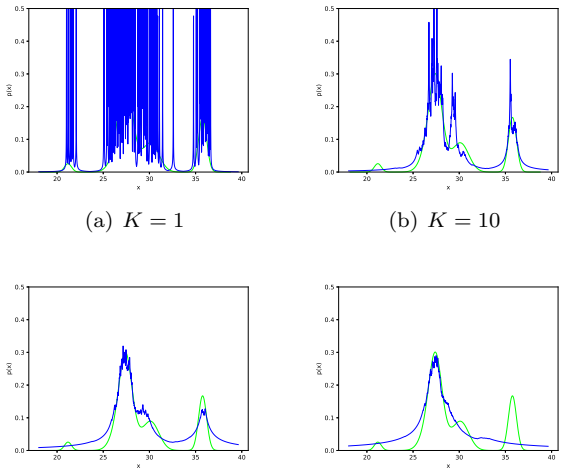
Generate a test set $\mathcal{T}$ with M points uniformly for the purpose of plotting estimating distribution
for $i = 1$ *to* M **do**
  for $j = 1$ *to* N **do**
    $dis(j) \leftarrow$ the distance between $\mathcal{T}(i)$ and $\mathcal{S}(j)$
  end
  Sort *dis* array in an ascending order
  Select $dis(K)$ as the radius of the sphere and estimate probabliby density according to Eq.(2)
end
return a plot figure using test set $\mathcal{T}$ and its estimation

In Alg. 2, for certain point $\boldsymbol{x}$ in test set $\mathcal{T}$, we calculate its distance to all points in data set $\mathcal{S}$ and sort them to pick up the k-th neareast neighboor of $\boldsymbol{x}$. This procedure leads to a prohibitive time complexity, i.e., $O(MN \log N)$. Thus, this algorithm is not scalable to large data sets.

4.3 Answer to Requirement 4

Requirement 4: *For nearest neighbor methods, you should vary K in $p(\boldsymbol{x}) = \frac{K}{NV}$ to see the difference, you are encouraged to plot an illustration as Figure 2.26 in the text book, to plot the true distribution, see* **GaussianMixture1D.plot** *in the handout for more details. Additionally, please show that the nearest neighbor method does not always yield a valid distribution, i.e. it won't converge to 1 if you sum the probability mass over all the space (in our case $(-\infty, +\infty)$), you could show this either empirically or theoretically.*

First, we plot three estimations figure with distinct K values in Fig. 6. Consistent with the



(a) $K = 1$ (b) $K = 10$

(c) $K = 30$ (d) $K = 50$

Figure 6: **k-NN** Density Estimation with various values of K (shown by blue curve) and true distribution (shown by green curve)

statement on text book [9], we observe that the parameter K governs the degree of smoothing, so that a small value of $K = 1$ leads to a very noisy density model, whereas a large value $K = 50$ smoothes out the multimodal natrue of the true distribution from which the data set was generated. Again the optimum choice for K is neither too large nor too small. In Fig. 6(c), the model not only captures the multimodal of underlying distribution, but also tolerate noise from sample data, thus $K = 30$ is a good choice in this case.

Then, we will exemplify that k-NN method does not always yield a true distribution. Empirically, just take a glance at Fig. 6(c), which suggests that our estimating probability density is almost higher than true distribtuion at every point and its area is larger. Since the integral of true distribution is equal to 1, the integral of k-NN estimation must be above 1. Consequently, it cannot be a valid distribution.

Finally, we give a simple proof for the divergence of integral on k-NN estimation in 1-D case. Let's take the case $K = 1$ as an instance. Let $p(x)$ denote the estimating distribution and $x_{\max}$ and $x_{\min}$ denote the maximum and minimum of sample data, respectively. Then, we have

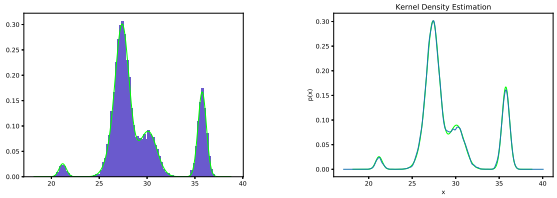
$$\int_{-\infty}^{+\infty} p(x)dx \geq \int_{-\infty}^{x_{\min}} p(x)dx + \int_{x_{\max}}^{+\infty} p(x)dx$$
$$= \frac{1}{2N}\left(\int_{-\infty}^{x_{\min}} \frac{1}{|x_{\min} - x|}dx + \int_{x_{\max}}^{+\infty} \frac{1}{|x - x_{\max}|}dx\right),$$

where we have used the fact $p(x) \geq 0$ and Eq.(2). Since both terms in the right side will diverge to positive infinity, we conclude that $\int_{-\infty}^{+\infty} p(x)dx$ will never converge to 1, which completes the proof.

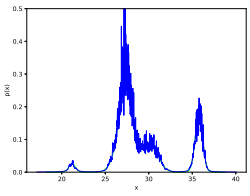
5 Answer to Requirement 1

Requirement 1: *For all three algorithms, you should vary the number of data used, let's say you could use* $100, 500, 1000, 10000$ *to see what will happen for your estimation, you don't have to report* $3 \times 4 = 12$ *plots just make an empirical assertion about how does the number of data influence the quality of the estimation. In the rest part of the requirements, if not specified, use* **num_data=200** *for exploration (or other number you think is better for your exploration, please state clear and keep consistent if so).*

To demonstrate the effect of the amount of data, we draw three estimates corresponding to three methods above in Fig. 7. We sample 10000 data points from the true distribution and performances of all these three methods are marked constrast to the original.



(a) **Histogram** with $k = 75$ (b) **KDE** with $h = 0.1389$



(c) **k-NN** with $K = 30$

Figure 7: Density Estimations using **num_data=10000**

For **Histogram**, we seperate the data into $k = 75$ bins, which obtains an overfitting case for **num_data=200**. But with growing number of data, we avoid overfitting and obtain a nearly identical estimation. As for **Kernel Density Estimation**, we use the same method in Sec. 3.3.2 to select the best bandwidth $h = 0.1389$ and re-calculate the distribution. As easily observed, the performance of this method is perfect and the estimation almost fits the true distribtuion. Finally, for **K-Neareast-Neighboor Method**, though it is still a little spiky, its behavior improves substantially and capture all modes in true distribution.

In a nutshell, the amount of data plays a significant role in machine learning. With more data, we can utilize some complex models and avoid overfitting, which contributes to the nearly perfect estimatation.

References

[1] Seymour Geisser and Wesley O Johnson. *Modes of parametric statistical inference*, volume 529. John Wiley & Sons, 2006.

[2] Karl Pearson. X. contributions to the mathematical theory of evolution.-ii. skew variation in homo-

geneous material. *Philosophical Transactions of the Royal Society of London.(A.)*, (186):343–414, 1895.

[3] BD Ripley. Modern applied statistics with s, 2002.

[4] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[5] Berwin A Turlach. Bandwidth selection in kernel density estimation: A review. In *CORE and Institut de Statistique*. Citeseer, 1993.

[6] Seymour Geisser. *Predictive inference*. Routledge, 2017.

[7] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[8] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[9] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[10] Leslie Greengard and John Strain. The fast gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[11] Leslie Greengard. *The rapid evaluation of potential fields in particle systems*. MIT press, 1988.

[12] Changjiang Yang, Ramani Duraiswami, Nail A Gumerov, and Larry Davis. Improved fast gauss transform and efficient kernel density estimation. In *null*, page 464. IEEE, 2003.

[13] Changjiang Yang, Ramani Duraiswami, and Larry S Davis. Efficient kernel machines using the improved fast gauss transform. In *Advances in neural information processing systems*, pages 1561–1568, 2005.

[14] Vikas Chandrakant Raykar and Ramani Duraiswami. Fast optimal bandwidth selection for kernel density estimation. In *Proceedings of the 2006 SIAM International Conference on Data Mining*, pages 524–528. SIAM, 2006.

6 Further Exploration

6.1 Improved Fast Gauss Transform

In Sec. 3.2, we have proposed a straightforward implementation of KDE, the complexity of which is $O(NM)$. This complexity can be a significant barrier to the use of these density estimation techniques in realistic scenarios. To address this issue, the Fast Gauss Transform(FGT) [10, 11] was proposed to improve the complexity of this evaluation to $O(N+M)$ operations. However, this method severely suffers from the curse of dimension due to the exponential growth of complexity with dimensionality. In this section, we introduce a simple and novel approach called Improved Fast Gauss Transform(IFGT) [12], which has been substantiated both scalable and practical [13, 14].

6.1.1 Theory

The fast Gauss transform was introduced for efficient computation of the weighted sum of Gaussians

$$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|y_j - x_i\|^2/h^2} \tag{10}$$

where q_i are the weight coefficients, $\{x_i\}_{i=1,\dots,N}$ are the centers of the Gaussians, h is the bandwidth parameter of the Gaussians. The sum of the Gaussians is evaluated at each of the 'target' points $\{y_j\}_{j=1,\dots,M}$

With some derivation, we have

$$G(y_j) = \sum_{i=1}^{N} q_i e^{-\|(y_j - x_*) - (x_i - x_*)\|^2/h^2}$$

$$= \sum_{i=1}^{N} q_i e^{-\|x_i - x_*\|^2/h^2} e^{-\|y_j - x_*\|^2/h^2} e^{2(y_j - x_*)(x_i - x_*)/h^2}$$

where x_* is expansion center. The crux of the IFGT algorithm is to separate this entanglement via the Taylor's series expansion of the exponentials as shown below:

$$e^{2\Delta x_i \Delta y_j/h^2} = \sum_{n=0}^{p-1} \frac{2^n}{n!} \left[\left(\frac{\Delta x_i}{h}\right) + \left(\frac{\Delta y_j}{h}\right) \right]^n + \epsilon(p)$$

$$= \sum_{n=0}^{p-1} \frac{2^n}{n!} \sum_{a+b=n} \frac{n!}{a!\,b!} \left(\frac{\Delta x_i}{h}\right)^a \left(\frac{\Delta y_j}{h}\right)^b + \epsilon(p),$$

where we have defined $\Delta x_i = x_i - x_*$ and $\Delta y_j = y_j - x_*$ and used $\epsilon(p)$ to denote the error w.r.t polynomial order p.

Substituing the result into Eq.(10), we can approximate $G(y_j)$ with

$$\tilde{G}(y_j) = \sum_{i=1}^{N} q_i e^{-{\Delta x_i}^2/h^2} e^{-{\Delta y_j}^2/h^2}$$

$$\times \sum_{n=0}^{p-1} \frac{2^n}{n!} \sum_{a+b=n} \frac{n!}{a!\,b!} \left(\frac{\Delta x_i}{h}\right)^a \left(\frac{\Delta y_i}{h}\right)^b$$

$$= \sum_{a+b \le p-1} C_{a,b}\, e^{-{\Delta y_j}^2/h^2} \left(\frac{\Delta y_i}{h}\right)^b,$$

where the coefficients $C_{a,b}$ is given by

$$C_{a,b} = \frac{2^{a+b}}{a!\,b!} \sum_{i=1}^{N} q_i e^{-{\Delta x_i}^2/h^2} \left(\frac{\Delta x_i}{h}\right)^a.$$

For moderate p, the number of terms is rather small compared with N and M. Thus, we omit it in our complexity cost. The coefficients $C_{a,b}$ can be evaluated separately in $O(N)$ and evaluations of $\tilde{G}(y_j)$ at M points is $O(M)$, hence the computational complexity has reduced from the quadratic $O(NM)$ to the linear $O(N+M)$.

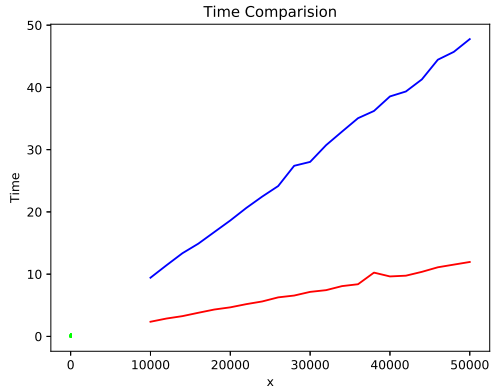


Figure 8: The running time of IFGT (shown by red curve) and Naive (shown by blue curve) with various N

6.1.2 Experiments

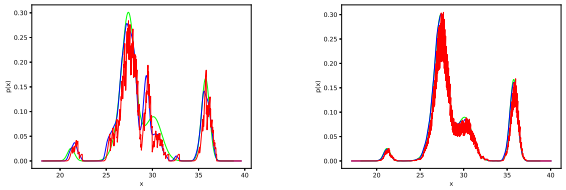
In this section, we implement the algorithm and conduct experiments on various sizes of sample set.

The key problem is to choose the center point x_*. We set K centers unifomly on the axis and approximate each source point with the closest center point. For each center, we calculate the coefficient $C_{a,b}$ for each center and estimate the target points with some nearest centers.

There are some tricks in my implementation to accelerate the algorithm. We omit them here and for details, I refer you to my source code.

To demonstrate the efficiency of IFGT algorithm, we compare the running time of IFGT with that of the naive implementation in Fig. 8. We fix the test data number $M = 10,000$ and vary the number of sample data from $10,000$ to $50,000$[2]. The results show that for moderate K, the IFGT algorithm is significantly faster than the naive implementation, especially for large N, M.

In addition to the efficiency, we also evaluate the accuarcy of the approximation algorithm. To this end, two figures are displayed in Fig. 9. As can be seen, though the IFGT is a little spiky, it approximates the direct KDE very well, especially in large case.



(a) $N = 200$ and $h = 0.2$ (b) $N = 10000$ and $h = 0.1389$

Figure 9: IFGT (shown in red), KDE (shown in blue), True (shown in green)

Thus, the approximation algorithm IFGT is both efficient and accurate.

[2]You should change the handout file a little here.

6.2 Efficient Implementation for 1-D k-NN

In this section, I will propose a method designed by myself, to further improve the complexity of the direct implementation.

The bottleneck of Alg. 2 is to calculate the distance between every pair of sources and targets. In 1-D case, we can address this issue by arranging all points on the axis and use the monotonicity of the scanning range. We propose the pseudo-code of our algorithms in Alg. 3, where we assume our sample data and test data has been sorted already.

ALGORITHM 3: Linear k-NN Implementation

Input : A sorted data set $\mathcal{S}$ with N samples from an unknown 1-D distribution and an integer K

Output : An estimatation of probability density distribution

Generate a sorted test set $\mathcal{T}$ with M points uniformly
$L \leftarrow 1, R \leftarrow K$
for $i = 1$ *to* M **do**
  while $|\mathcal{S}(R+1) - \mathcal{T}(i)| < |\mathcal{S}(L) - \mathcal{T}(i)|$ **do**
    $L \leftarrow L + 1, R \leftarrow R + 1$
  end
  $h_i \leftarrow \min\left(|\mathcal{S}(R) - \mathcal{T}(i)|, |\mathcal{S}(L) - \mathcal{T}(i)|\right)$
  Select h_i as the radius of the sphere and estimate probabliby density according to Eq.(2)
end
return a plot figure using test set $\mathcal{T}$ and its estimation

It is obvious that the K nearest neighboors is in a consecutive range of sorted sample data $\mathcal{S}$. Let L_i and R_i denote the interval of neareast neighboors of i-th test point. Then L_i and R_i is monotony with respect to i, that is, with the test point shifting right, the interval $[L_i, R_i]$ will only move right. Knowing this property, we can determine L_i and R_i for all test data with one scan of sample data. Thus, the time complexity of this method is $O(N + M)$, which substantially reduce the time cost.