Data Structures in Kernel Density Estimation

LUC DEVROYE AND FRED MACHELL

Abstract—We analyze and compare several data structures and algorithms for evaluating the kernel density estimate. Frequent evaluations of this estimate are for example needed for plotting, error estimation, Monte Carlo estimation of probabilities and functionals, and pattern classification. An experimental comparison is included.

Index Terms—Algorithms, complexity, data structure, density estimation, expected time, pattern recognition.

Manuscript received March 12, 1984; revised January 15, 1985. Recommended for acceptance by Josef V. Kittler.

L. Devroye is with the School of Computer Science, McGill University, Montreal, P.Q., Canada.

F. Machell is with Applied Research Laboratories, University of Texas, Austin, TX 78713.

I. Introduction

In kernel density estimation, R^d-valued *data* $X_1, \cdots, X_n$ are stored in one way or another to enable future evaluations of the kernel density estimate

$$f_n(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K((x - X_i)/h), \qquad x \in R^d,$$

where K is a given density and $h > 0$ is a positive number. These estimates play a crucial role in statistical pattern recognition, and most statisticians who have implemented them have found to their distress that simulations involving repeated evaluations of f_n require a lot of CPU time even for $d = 1$ and n as small as 2000. Repeated evaluations of f_n are needed, for example, in

a) *Error estimation* (where $f_n - f$ must be computed for many values of x; here f is the density of $X_1, \cdots, X_n$).

b) *Plotting* f_n.

c) *Estimation of probabilities of sets* A (by $\int_A f_n$).

d) *Computation of functionals* $\int g(f_n)$ such as $\int f_n^2$, etc.

e) *Classification of many new observations* by the kernel method (this involves computing various class densities for many values of x).

To summarize this, we can say that the object is to evaluate $f_n(x_i)$, $1 \leq i \leq l$ at l given points. This will be called the *deterministic model*. In applications a), c), d), and e), the given points x_i can sometimes be thought of as a sample $Y_1, \cdots, Y_l$ of independent identically distributed random vectors with common density f and independent of $X_1, \cdots, X_n$. The data $X_1, \cdots, X_n$ also form an i.i.d. sequence with common density f. This will be called the *random model*.

In this note, we will compare various data structures and algorithms for carrying out these evaluations, both from a theoretical and experimental point of view. Because there are many variable factors (n, l, K, h, d, and f), we have a gigantic task ahead of us. We will present the main principles for $d = 1$, and limit ourselves to general outlines for $d > 1$. Our theoretical results hold for all f thanks to the Lebesgue density theorem. In our choice of K and h, we will only consider the optimal form of K and sequences h depending upon n in such a way that the density estimate f_n converges to f in some sense. Finally, to be able to make a clear distinction between the different data structures for several values of n and l, we will consider two stages.

1) The Setup (Preprocessing Stage or Initialization Stage): In this stage, a data structure is constructed starting from an array containing $X_1, \cdots, X_n$. If the data structure uses knowledge about K and/or h, it will be called a *specific data structure*. In general, these data structures will be less efficient or even useless for other K and h. In contrast, data structures that are useful for all reasonable choices of K and h are called *open data structures*.

2) The Evaluation: The evaluation of $f_n(x)$ at *one* point x.

We will find, not unexpectedly, that although they can be used in fewer situations, specific data structures lead to faster evaluations than open data structures. In our analysis and comparisons, we will use the symbols $\sim$ and $\simeq$ in the following senses:

$$a_n \sim b_n \text{ means } \lim_{n \to \infty} a_n/b_n = 1;$$

$$a_n \simeq b_n \text{ means } \lim_{n \to \infty} a_n/b_n = c$$

$$\text{for some constant } c \in (0, \infty).$$

To compute the time taken by a certain algorithm, we assume that most common operations (compare, move, truncate,

$\times$, /, +, −) take a constant time independent of the operand(s) involved. The *setup time* $T_s = T_s(X_1, \cdots, X_n)$ and the *evaluation time* $T_e(x) = T_e(x; X_1, \cdots, X_n)$ are both assumed to be Borel measurable functions of their arguments (for purely theoretical reasons). In the random model, the two quantities that are of primary importance are

$$\bar{T}_s = E(T_s) \qquad \text{(the } \textit{expected setup time}\text{)}$$

and

$$\bar{T}_e = \int E(T_e(x))\, f(x)\, dx \qquad \text{(the } \textit{expected evaluation time}\text{)}.$$

In the deterministic model, we still assume that $X_1, \cdots, X_n$ are i.i.d. with density f, but the x_i's can take any value, including the value that makes $T_e(x)$ largest. Thus, in this model, we are more interested in $\bar{T}_s$ and

$$T_e^* = E(\sup_x T_e(x; X_1, \cdots, X_n))$$

(the *expected worst-case evaluation time*).

We will conclude our preparations by commenting on the choices of K and h. When $X_1, \cdots, X_n$ are i.i.d. with density f, and h is a sequence of positive numbers depending upon n such that $h \to 0$, $nh^d \to \infty$, then $\int |f_n - f| \to 0$ almost surely and in the mean for all f and K [7], and $f_n \to f$ in probability for almost all x, for all f and all bounded K with compact support [9]. For earlier but slightly more restrictive versions of these consistency theorems, see Rosenblatt [16], Parzen [14], and Cacoullos [3]. From a careful analysis of the rate of convergence of the expected L_2 error, Epanechnikov [10] for $d = 1$ and Deheuvels [6] for $d > 1$ derived the optimal form of K: $K(x) = c(1 - \|x\|^2)_+$. Here $(\ )_+$ denotes the positive part of $(\ )$, and c is a normalization constant, equal to $\frac{3}{4}$ for $d = 1$. This happens also to be the optimal form for the expected L_1 error, so that we will assume this form of K throughout the note. Although we will not use it anywhere, it is worth mentioning that for $d = 1$, h is often chosen as $c/n^{1/5}$ where c is a constant depending upon K and f only. This will give an idea of the order of magnitude of h, and can help in the design of a data structure that is good for most common choices of h. In practice, h is chosen as a function of the data (these are called automatic density estimates), and in some cases, we need to evaluate f_n for different values of x *and* h. This calls of course for an open data structure. Once h is fixed, we can set up a specific data structure for future evaluations.

In this paper, we will not consider a computational method based upon fast Fourier transforms—we refer the reader to Silverman [17] and Jones and Lotwick [12] for more details.

II. Open Data Structures

A. An Array

When $X_1, \cdots, X_n$ are simply stored in array form, we have, deterministically, $T_e \simeq n$ and $T_s = 0$. To reduce the proportionality constant in T_s, one could compute $\sum_{i=1}^{n} (1 - ((x - X_i)/h)^2)_+$ in a loop, and multiply this sum with $3/(4nh)$ afterwards.

B. A Sorted Array

In the setup stage, $X_1, \cdots, X_n$ can be sorted by a comparison based sorting method, preferably quicksort, heapsort, or modifications thereof (see [13] for descriptions), so that either $T_s \simeq n \log n$ deterministically, or at least $\bar{T}_s \simeq n \log n$, and that in both cases the distribution of T_s does not depend upon f.

The evaluation of $f_n(x)$ can be done by first finding the index i such that $x \in [X_i, X_{i+1})$ by binary search (where, by convention, $X_0 = -\infty$ and $X_{n+1} = +\infty$), and then traveling up and down the array, starting from index i, and visiting all the X_j's for which $|x - X_j|/h < 1$, to form the sum

$$\frac{3}{4nh} \sum_{j:\, |(x - X_j)/h| < 1} (1 - ((x - X_j)/h)^2).$$

Thus, we need only consider adjacent indexes. The time taken by the binary search is $\bar{T}_e \simeq \log n$, $T_e^* \simeq \log n$. The time needed to form the sum must be added to this: it is f-dependent and can be written as $a + bN(x)$ where a, b are positive constants, and $N(x)$ is the number of X_j's in $(x - h, x + h)$. We need to know $E(N(x))$ and $E(\sup_x N(x))$ for our analysis. In the Appendix, we will prove the following:

Theorem 2.1

For all densities f on R^1, we have

a) $\dfrac{E(N(x))}{2nh} \to f(x)$, almost all x;

b) $\dfrac{\int E(N(x))\, f(x)\, dx}{2nh} \to \int f^2$

(even if the right-hand side is ∞);

c) $\dfrac{E(\sup N(x))}{2nh} \to \operatorname{ess\,sup}_x f(x)$

when also $nh/\log n \to \infty$;

where "ess sup" denotes the essential supremum (i.e., the unique number y with the property that for all $\epsilon > 0$, $\{x: f(x) > y + \epsilon\}$ and $\{x: f(x) > y - \epsilon\}$ have Lebesgue measure 0 and > 0, respectively). We recall the standing conditions on h: $\lim_{n\to\infty} h = 0$, $\lim_{n\to\infty} nh = \infty$, and on K: $K(x) = \frac{3}{4}(1 - x^2)$. The result remains valid for the uniform kernel $K(x) = \frac{1}{2} I_{[-1,1]}(x)$.

First, we note that $\int f^2$ and ess sup f are measures of the peakedness of f. For very peaked densities, there are many x's for which $N(x)$ is quite large, namely for all the x's near the peak(s) of f. This has a negative influence on the evaluation time. The "best" density for evaluation is the uniform density. Theorem 2.1 does not give us any information about the rate of increase of $\int E(N(x))\, f(x)\, dx$ and $E(\sup N(x))$ when $\int f^2 = \infty$ and ess sup $f(x) = \infty$, respectively. But when these measures of peakedness are finite, we can conclude that

$$\bar{T}_e = (\simeq \log n) + (\simeq nh)$$

and

$$T_e^* = (\simeq \log n) + (\simeq nh).$$

The notation is self-explanatory. The first terms are not dependent upon f. The second terms have proportionality factors $2\int f^2$ and 2 ess sup f, respectively. For open data structures, $N(x)$ comes very close to being a lower bound for the time needed to compute $f_n(x)$, so that little improvement over the present method is possible from an asymptotic point of view.

C. A Bucket Structure

In an attempt to reduce the setup time over the previous method, we could apply a standard order-preserving hashing technique with chaining: find $m = \min X_j$, $M = \max X_j$ in one pass of the data; imagine that $[m, M)$ is divided into $n - 1$ equal intervals (buckets)

$$\left[m + \frac{i-1}{n-1}(M - m),\ m + \frac{i}{n-1}(M - m) \right), \qquad 1 \leq i \leq n - 1.$$

Add an nth bucket $[M, M + [1/(n-1)])$. Keep for each bucket a linked list of all X_j's belonging to the bucket (this requires a second and last pass of the data). Note that the index $i = i(x)$ of the bucket of x is obtained by simple truncation

$$i = i(x) = 1 + \underbrace{\frac{x - m}{M - m}}\ (n-1).$$

The time taken by the setup is $T_s \simeq n$. The storage is usually $3n$, i.e., n words for the original array, n words for the pointers to next elements (linked elements) in the buckets, and n words for pointers to bucket headers.

We pay very slightly in evaluation time for the fact that the data are not sorted. For fixed x, we determine $i_l = i(x-h)$ and $i_h = i(x+h)$, and replace i_l and i_h by $\max(i_l, 1)$ and $\min(i_h, n)$, respectively. Then $f_n(x)$ is computed as

$$\left(\frac{3}{4nh}\right) \sum_{i_l \le i \le i_h} \sum_{\substack{j: X_j \text{ in} \\ \text{bucket } i}} (1 - ((x - X_j)/h)^2)_+.$$

Obviously, we still have $T_e \ge a + bN(x)$ as for method 2.2. Also, $T_e \le c + bN^*(x)$ for some different constant $c > 0$, where $N^*(x)$ is the number of X_j's in $[x - h - (M-m)/(n-1), x + h + (M-m)/(n-1)]$. We note here that the constants a, b, c do not depend upon n, h, f or $X_1, \cdots, X_n$. Also, the $\log n$ component in the expression for T_e has disappeared, because we are able now to locate a bucket in time proportional to a constant. By proving that $N^*(x)$ is approximately of the size of $N(x)$, we can show the following theorem (see Appendix).

Theorem 2.2

All of Theorem 2.1 remains valid for $N^*(x)$ when f satisfies the following condition:

$$\lim_{n \to \infty} \frac{1}{nh} E\left(\max_{1 \le j \le n} |X_j| \right) = 0.$$

Let us first comment on the condition imposed upon f: it is needed to insure that the intervals around x that are considered when the sum is formed do not contain too many points, i.e., to insure that $M - m$ is not too large. The verification of the condition is sometimes a problem. However, we have the following property.

Property 2.1

If f is a density for which

$$\int e^{t|x|} f(x)\, dx < \infty \qquad \text{for some} \quad t > 0$$

(this is called Cramer's condition), and $\lim_{n \to \infty} nh/\log n = \infty$, then

$$\lim_{n \to \infty} \frac{1}{nh} E\left(\max_{1 \le j \le n} |X_j| \right) = 0.$$

Cramer's condition is satisfied when $f(x) \le ae^{-b|x|}$, $|x| \le c$, for some $a, b, c > 0$.

From Theorems 2.1 and 2.2, we immediately deduce that $\bar{T}_e$ and T_e^* are both $\simeq nh$. In the former case, the proportionality constant is $2\int f^2$, and in the second case, it is 2 ess sup f. Clearly, for those densities that are covered by Theorem 2.2, we expect similar evaluation times, with perhaps a slight edge for data structure II-B.

III. Specific Data Structures

Observe that f_n is a spline with knots at $X_j - h$, $X_j + h$, $1 \le j \le n$, and that within each interval between knots f_n takes the form $a + bx + cx^2$, where a, b, and c are coefficients depending upon the X_j's only. There are $2n$ knots and thus $2n-1$ intervals on which $f_n \ne 0$. To evaluate f_n, it suffices to store

1) The ordered sequence of $2n$ knots.
2) The coefficients a, b, and c for each of the $2n-1$ intervals.

Thus, about $8n$ real numbers have to be stored, well up from the storage requirements for the open data structures, but still linear in n. For evaluating $f_n(x)$, it suffices to locate the interval to which x belongs by binary search, and evaluate $a + x(b + cx)$. The time is deterministically bounded by $\log n$, independent of f.

In Sections III-A and III-B, we will describe two methods for setting up the data structure described above.

A. Setup by Brute Force

We proceed as follows:

Step 1) Order $X_1, \cdots, X_n$ by a comparison-based sorting algorithm in expected time $\simeq n \log n$.

Step 2) In one pass of $X_1, \cdots, X_n$ with two moving pointers, create an ordered sequence of knots and labels $(Y_1, Z_1), \cdots, (Y_{2n}, Z_{2n})$, where $Y_1 \le Y_2 \le \cdots$ is an ordered version of $X_1 - h, \cdots, X_n - h, X_1 + h, \cdots, X_n + h$, and the labels Z_i are defined by

$$Z_i = \begin{cases} +j & \text{if} \quad Y_i = X_j + h, \\ -j & \text{if} \quad Y_i = X_j - h. \end{cases}$$

The time taken in Step 2 is deterministically proportional to n.

Step 3) Compute for each of $2n-1$ intervals i (interval $i = [Y_i, Y_{i+1})$) coefficients a_i, b_i, and c_i as follows: let i_l be the largest index $\le i$ for which $Z_{i_l} < 0$ (this can be found by counting down from i); and let i_h be the smallest index $\ge i+1$ for which $Z_{i_h} > 0$. Set $j_l \leftarrow |Z_{i_l}|$, $j_h \leftarrow |Z_{i_h}|$. If $j_h \le j_l$ do

$$(a_i, b_i, c_i) = \frac{3}{4nh} \sum_{j=j_h}^{j_l} \left(\left(1 - \left(\frac{X_j}{h}\right)^2\right), \frac{2X_j}{h^2}, -\frac{1}{h^2} \right).$$

We have

$$\begin{aligned} \bar{T}_s &= (\simeq n \log n) + \left(\simeq E\left(\sum_j N(X_j) \right) \right) \\ &= (\simeq n \log n) + (\simeq nE(N(X_1))) \\ &= (\simeq n \log n) + \left(\simeq n \cdot 2nh \int f^2 \right) \end{aligned}$$

when $\int f^2 < \infty$ (by Theorem 2.1). It is worth noting that the second term, the contribution of Step 3, usually dominates: it is superlinear but subquadratic.

B. Setup by Recursive Computations

It is not hard to see that $(a_{i+1}, b_{i+1}, c_{i+1})$ can be computed from (a_i, b_i, c_i), and that we can thus compute all our coefficients in one pass of the intervals. For example, Step 3 could be replaced by

Step 3') $(A, B, C) \leftarrow (0, 0, 0)$.
For $i := 1$ *to* $2n-1$ *do*

$$j \leftarrow |Z_i|,\ s \leftarrow \operatorname{sign} Z_i;$$

$$(A, B, C) \leftarrow (A, B, C) - s \left(\left(1 - \left(\frac{X_j}{h}\right)^2\right), \frac{2X_j}{h^2}, -\frac{1}{h^2} \right);$$

$$(a_i, b_i, c_i) \leftarrow (A, B, C).$$

Since Step 3' takes the proportional to n, it is clear that Step 1 dominates and that thus $\bar{T}_s \simeq n \log n$.

Important Remark: The procedure in Section III-B is numerically inaccurate when n is large, and could lead to serious errors on limited precision machines. Also, the procedures in both Sections III-A and B have built-in cancellation errors in

the evaluation stage: a_i, b_i, and c_i become increasingly large as $h \downarrow 0$; yet, $f_n(x)$, their weighted sum, tends to f if h is well chosen. Fortunately, for h to be so small that this poses a problem, n must be astronomically large.

The numerical danger of the procedure in Section III-B can be circumvented by taking the uniform kernel $K(x) = \frac{1}{2}$, $|x| \leqslant 1$. It is known that the replacement of the optimal kernel by the uniform kernel does not increase $E(\int |f_n - f|)$ very much. For the uniform kernel, we need only remember a_i's because

$$f_n(x) = \frac{1}{2nh} \sum_{j:|X_j - x| \leqslant h} 1.$$

The numerical accuracy is achieved by doing the recursive computations on integers. For example, Step 3' becomes simply

Step 3') $A \leftarrow 0$ (A is an integer.)
For $i := 1$ *to* $2n - 1$ *do*

$j \leftarrow |Z_i|$, $s \leftarrow$ sign Z_i;

$A \leftarrow A - s$;

$a_i \leftarrow A/(2nh)$.

Also, the evaluation is now very simple: for $x \in [Y_i, Y_{i+1})$, we have $f_n(x) = a_i$.

C. Summary of Complexities

We will now summarize our findings for $\bar{T}_s$ and $\bar{T}_e$ for densities with $\int f^2 < \infty$, and sequences h satisfying $h \to 0$, $nh/\log n \to \infty$:

	Data structure and algorithm	$\bar{T}_s$	$\bar{T}_e$
Open data structures	II-A	0	$\simeq n$
	II-B	$\simeq n \log n$	$\simeq nh$
	II-C	$\simeq n$	$\simeq nh$
Specific data structures	III-A	$\simeq n^2 h$	$\simeq \log n$
	III-B	$\simeq n \log n$	$\simeq \log n$

For the $\bar{T}_e$ result in row II-C there is a weak additional condition on f (see Theorem 2.2). For what happens in the uninteresting case when $nh/\log n \not\to \infty$, we refer to Sections II-B (for the expression for $\bar{T}_e$) and III-A (for the expression for $\bar{T}_s$). Finally, it is worth pointing out that the proportionality factors do not depend upon f except in the following cases: the entries for $\bar{T}_e$ in rows II-B and II-C and the entry for $\bar{T}_s$ in row III-A. In those cases, f influences the time in proportion to $\int f^2$. We will see in our experiments, described in Section IV, that the theoretical rankings predicted by this table are preserved.

D. The Uniform Kernel

When $K = \frac{1}{2}$, $|x| \leqslant 1$ (the uniform kernel), there is an open data structure that allows $\simeq \log n$ evaluations: for, if we consider the sorted array of Section II-B, then $f_n(x)$ can be evaluated as follows: locate the intervals i and j such that $x - h \in [X_i, X_{i+1})$, $x + h \in [X_j, X_{j+1})$, by binary search of the sorted array. Recall the convention $X_0 = -\infty$ and $X_{n+1} = +\infty$. Then, $f_n(x) = (j - i)/(2nh)$. This method combines the fast setup of II-B with the simple evaluations of the specific methods. Of course, we are now using the fact that K is the uniform kernel, and thus, the comparison with open data structures is not fair.

E. Setup by Computing Distribution Functions

The method of Section III-D has an evaluation time that is about twice that of Sections III-A and III-B, because two lookups are needed instead of one, and the lookups are the main contributing factor to $\bar{T}_e$. On the other hand, the storage costs are drastically reduced, and $\bar{T}_s$ remains roughly the same. We can generalize it now to the optimal kernel as follows.

In the setup stage, we order $X_1, \cdots, X_n$, and compute the following auxiliary arrays, cumulative distribution functions of sorts:

$$S_i = X_1 + \cdots + X_i, \qquad 1 \leqslant i \leqslant n,$$

$$T_i = X_1^2 + \cdots + X_i^2, \qquad 1 \leqslant i \leqslant n.$$

The storage is $3n$, the time taken is $(\simeq n \log n) + (\simeq n)$, and the data structure does not use information about h, an important point in some applications.

The evaluation is done as in Section III-D: first locate the intervals i and j such that $x - h \in [X_i, X_{i+1})$, $x + h \in [X_j, X_{j+1})$ (this takes time $\simeq \log n$). Then, compute $f_n(x)$ by the formula

$$\frac{3}{4nh}\left((j - i)\left(1 - \frac{x^2}{h^2}\right) + (S_j - S_i)\frac{2x}{h^2} - (T_j - T_i)\frac{1}{h^2}\right)$$

(this takes time $\simeq 1$). $\bar{T}_e \simeq \log n$. Because of its similarity with III-B, and its inferior proportionality constant in $\bar{T}_e$, it will not be tested. Its lower storage requirements and h-independence could make it useful in some situations, but, as for III-B, numerical accuracy remains a problem.

TABLE I
MONTE CARLO ESTIMATES OF $\bar{T}_s$ AND $\bar{T}_e$ (IN MILLISECONDS) FOR THE NORMAL DENSITY

	n	100	200	400	800	1600
TIME	METHOD					
$\bar{T}_s$	II-A	0	0	0	0	0
	II-B	15	29	69	151	347
	II-C	6	13	22	49	89
	III-A	431	1552	5446	18421	
	III-B	35	76	164	341	
	III-E	15	76	78	168	381
$\bar{T}_e$	II-A	3.7	7.1	14.1	27.5	55.0
	II-B	2.0	3.3	5.8	9.2	16.9
	II-C	2.5	4.4	7.0	11.66	20.0
	III-A	0.36	0.39	0.43	0.45	
	III-B	0.37	0.39	0.42	0.45	
	III-E	0.61	0.71	0.75	0.81	0.92

IV. AN EXPERIMENTAL COMPARISON

We compared methods II-A, II-B, II-C, III-A, and III-B for the normal density and sample sizes n = 100, 200, 400, 800, and 1600. All the experiments were carried out on a CYBER 171 computer at Applied Research Laboratories at the University of Texas. The smoothing factor h was taken in the (asymptotically) optimal way with respect to $E(\int |f_n - f|)$: for the normal density, this is known to be

$$h = \left(\frac{225\pi e^2}{32}\right)^{1/10} \Big/ n^{1/5} = 1.6644745 \cdots / n^{1/5}$$

(see, e.g., [8]). For each method, Monte Carlo estimates of $\bar{T}_s$ and $\bar{T}_e$ were obtained (see Table I).

The variations of the expected times with respect to n are exactly as predicted in the summary of Section III-C. For example, $\bar{T}_e$ increases linearly with n for method II-A. For methods II-B and II-C, the rate of increase is $n^{4/5}$, and the ratio

TABLE II
MONTE CARLO ESTIMATES (IN MILLISECONDS) OF $(\bar{T}_s + l\bar{T}_e)/l$ WHEN f IS NORMAL, $n = 800$

l	1	10	100	1000	10000	∞
Method						
II-A	27.5	27.5	27.5	27.5	27.5	27.5
II-B	160.2	24.3	10.7	9.35	9.22	9.20
II-C	60.7	16.6	12.2	11.7	11.7	11.7
III-A	18421.5	1842.6	184.7	18.9	2.29	0.450
III-B	341.5	34.6	3.86	0.391	0.484	0.450
III-E	168.8	17.61	2.49	0.978	0.827	0.810

TABLE III
SPEEDUP RELATIVE TO METHOD II-A FOR THE NORMAL DENSITY, $n = 800$

l	Best open data structure	Best specific or open data structure
1	1.00	1.00
10	1.66	1.66
100	2.57	11.04
1000	2.94	34.77
10000	2.99	56.80
∞	2.99	61.12

between $\bar{T}_e$ for method II-B and $\bar{T}_e$ for method II-C remains constaht, with a slight edge for method II-B. The figures for methods III-A and III-E increase logarithmically with n, and are all but identical. The improvement over the open data structures is impressive.

In real applications, we first set up a data structure and carry out l evaluations. Thus, the criterion should be $(\bar{T}_s + l\bar{T}_e)/l$. For $n = 800$, we computed Monte Carlo estimates of this. All the sorting in the setup stages was done by the celebrated quicksort algorithm ([11], [18], and [1]). Table II shows the estimates. For very small $l (l \leqslant 30)$, the best open data structure is better than the best specific data structure because of smaller setup times. The range $0 \leqslant l \leqslant 30$ is unrealistically small in most cases. For standard values of l used in Monte Carlo computations or plotting subprograms, method III-B is by far the fastest method. We should stress that for large n, method III-B is numerically inaccurate, and should be used with care. Of course, for large n, it can be replaced by method III-A which, despite its gigantic setup time, manages to be the fastest among the more accurate methods II-A–III-A for l greater than approximately 2000. As pointed out in Section III-B, the numerical problem can be sidestepped by using a uniform kernel instead of Epanechnikov's kernel.

In Table III, we show the speedup (in terms of $\bar{T}_s + l\bar{T}_e$) that is obtainable if the standard algorithm II-A is replaced by either the best open data structure, or the best data structure overall. It does not really matter what value of l is chosen for such a table: for $n = 800$, the value chosen for Table III, the speedups are enormous. In general, the speedups will be better for larger n and l, and worse for smaller n and l.

In our experiment, h was rather large because the density was smooth. For estimation of unsmooth or multimodal densities, h is usually much smaller. This should work in favor of $\bar{T}_e$ for methods II-B, II-C, and of $\bar{T}_s$ for method III-A. Also, if we were to replace the optimal kernel by the uniform kernel, most times will decrease slightly. The performance of the distribution function method of Section III-E is comparable to that of Section III-B, with slightly smaller values for $\bar{T}_s$, but moderately larger values for $\bar{T}_e$.

We conclude this section by recommending that for open data structures methods II-B or II-C be used for all n, l. Specific data structures should be used whenever possible provided that l is not too small and that one takes care of the numerical inaccuracies.

V. EXTENSIONS TO R^d

Open data structures in R^d create special problems because there is no simple extension of the notion of a sorted array. There are many multidimensional data structures that qualify as generalizations, such as trees, quadtrees, k-d-trees, grids, etc., and that could help us search for a neighborhood of x, and identify the X_j's that are close to x. Regardless of what data structure is chosen to generalize II-B, the evaluation time is bounded from below by $E(N(x))$. Now, Theorems 2.1 and 2.2 have straightforward extensions to R^d, and in particular, for the optimal K, $E(N(x)) \sim$ volume unit sphere in $R^d \cdot nh^d \cdot \int f^2$. If the data structure is sophisticated enough so that this lower bound is achieved, then we will have made great progress in our search for a fast algorithm: indeed, the optimal h is often $\simeq n^{-1/(d+4)}$, and thus $nh^d \simeq n^{4/(d+4)}$: thus, *the savings in computation time become more outspoken as d grows larger.*

It seems natural, for example, to extend the bucket structure of II-C by dividing each axis into $n^{1/d}$ equal pieces. The storage and $\bar{T}_s$ are both $\simeq n$, while $\bar{T}_e \simeq nh^d$: for each x, we need only locate its bucket, and then travel to all buckets intersecting the sphere centered at x with radius h (and there are of the order of nh^d such buckets). This seems one of the most promising structures at this point.

To extend the ideas of Section III, we could proceed as follows. Let us start, for the sake of simplicity, with the bucket structure described in the previous paragraph. If $S_{x,r}$ is the sphere of radius r centered at x, and B is a given bucket, then we could partition all the buckets into three sets: B_1, all the buckets completely covered by $\cap_{x \in B} S_{x,h}$; B_2, all the buckets not in B_1 but intersecting $\cup_{x \in B} S_{x,h}$; and B_3, all leftover buckets. B_1 forms a core of buckets, B_2 is a shell (doughnut) of buckets, and B_3 groups all faraway buckets. Store the partial coefficients of the paraboloid for all the X_j's in B_1 for bucket B, and repeat this for all buckets B. To evaluate $f_n(x)$, we must form the sum of $K((x - X_j)/h)$. For those X_j's in B_1, this sum takes time $0(1)$; the contribution of the X_j's in B_3 is of course 0; and for the contribution to the sum coming from the X_j's in B_2, we need only travel to the buckets in the thin shell B_2 (there are about $(nh^d)^{(d-1)/d}$ such buckets, with proportionally many points). Thus, $\bar{T}_e$ seems to grow as $(nh^d)^{(d-1)/d}$. The improvement over the open data structures is no longer as spectacular as it was for $d = 1$, but it should be noticeable for $d = 2$, 3 too. Because the shell becomes more voluminous as d grows (which is reflected in the fact that $(d - 1)/d \to 1$ as $d \to \infty$), the improvement over the open data structures seems hardly worth the trouble for large d. The setup time can be kept small by a careful extension of the recursive method for $d = 1$.

We finally note the attempt by Postaire and Vasseur [15] to reduce the computation time in R^d when f must be evaluated at a mesh only, and K is the product of d univariate densities.

APPENDIX WITH PROOFS

Proof of Theorem 2.1

Proof of A: We note that

$$E(N(x)) = n \int_{x-h}^{x+h} f = 2nh \cdot \left(\frac{1}{2h} \int_{x-h}^{x+h} f \right) \sim 2nhf(x)$$

for almost all x, by the Lebesgue density theorem (see, e.g., [19, ch. 7 and 9] or [5, ch. 1–3]).

Proof of B: Let us define $f_h(x) = 1/2h \int_{x-h}^{x+h} f$. Thus,

$$\int E(N(x))\, f(x)\, dx/(2nh) = \int f_h f.$$

By Fatou's lemma, $\liminf_{n\to\infty} \int f_h f \geq \int \liminf_{n\to\infty} f_h f = \int f^2$. In the last step we used part A of this theorem. This shows that we can assume that $\int f^2 < \infty$. But $f_h(x) \leq f^*(x) = \sup_{r>0} (2r)^{-1} \int_{x-r}^{x+r} f$, the maximal function corresponding to f. It is known that $\int f^{*2} < \infty$ when $\int f^2 < \infty$ [5]. Thus, by the Lebesgue dominated convergence theorem, $\int f_h f \to \int f^2$, which concludes the proof of B.

Proof of C: Let us introduce $y = \operatorname{ess\,sup} f(x)$. We have, by part A,

$$\frac{E(\sup N(x))}{2nh} \geq \frac{\operatorname{ess\,sup} E(N(x))}{2nh} \geq y + o(1).$$

Thus, we need only worry about upper bounds. For every $\epsilon > 0$, it is true that

$$\frac{E(\sup N(x))}{2nh} \leq y + \epsilon + \frac{n}{2nh} P\left(\frac{\sup N(x)}{2nh} > y + \epsilon\right),$$

and we are done if we can show that the last term is $o(1)$ for all $\epsilon > 0$. Using the fact that $N(x)$ is a histogram with jumps at $X_i - h$ and $X_i + h$, we see that

$$\begin{aligned} P\left(\frac{\sup N(x)}{2nh} > y + \epsilon\right) &\leq \sum_{i=1}^{n} P(N(X_i - h) > 2nh(y + \epsilon)) \\ &\quad + P(N(X_i + h) > 2nh(y + \epsilon)) \\ &\leq n \int f(x)\,(P(A_{nx}) + P(B_{nx}))\, dx \end{aligned}$$

where $A_{nx} = \{[x - 2h, x]$ contains at least $2nh(y + \epsilon) - 1$ of the X_j's$\}$, and B_{nx} is defined similarly for the interval $[x, x + 2h]$. Take a new ϵ equal to the old ϵ minus $1/(2nh)$. We have obviously asymptotic equality. Let Z be a binomial $(n, \int_x^{x+2h} f)$ random variable. Then

$$\begin{aligned} P(A_{nx}) \leq P(Z > 2nh(y + \epsilon)) &\leq P(Z - E(Z) > 2nh\epsilon) \\ &\leq \exp(-n(2h\epsilon)^2 / (2(\int_x^{x+2h} f + 2h\epsilon))) \\ &\leq \exp\left(-nh \frac{\epsilon^2}{\epsilon + y}\right) \end{aligned}$$

where we used Bennett's version of Bernstein's inequality [2]. The same bound is valid for $P(B_{nx})$. Collecting bounds gives us

$$\frac{E(\sup N(x))}{2nh} \leq y + \epsilon + \frac{n}{h} \exp\left(-nh \frac{\epsilon^2}{\epsilon + y}\right).$$

The last term is $o(1)$ when $nh/\log(n/h) \to \infty$. But this follows from $1/h = o(n)$ and $nh/\log n \to \infty$.

Proof of Theorem 2.2

Because $N^*(x) \geq N(x)$ for all x, we can apply Theorem 2.1 to obtain lower bounds. In the remainder, we only consider upper bounds.

Proof of A: Throughout the proofs, we will set

$$p = \int_m^M f,\ q = q(x) = \int_{x-h-(M-m)/(n-1)}^{x+h+(M-m)/(n-1)} f.$$

For fixed $\epsilon > 0$, let A be the vent $\{(M - m)/(n - 1) < \epsilon h\}$, with complement A^c. Then, for $\delta > 0$, we can find $\epsilon > 0$ small enough such that

$$\begin{aligned} E(N^*(x)) &\leq E\left(E\left(\text{binomial}\left(n, \frac{q}{p}\right) \middle|\ m, M\right)\right) = nE\left(\frac{q}{p}\right) \\ &\leq (f(x) + \delta) E\left(\frac{n}{p} \cdot 2\left(h + \frac{M - m}{n - 1}\right) I_A\right) \\ &\quad + f^*(x) E\left(\frac{n}{p} \cdot 2\left(h + \frac{M - m}{n - 1}\right) I_{A^c}\right) \\ &\leq (f(x) + \delta)\, 2nh\, E\left(\frac{1}{p} + \frac{M - m}{h(n - 1)p}\right) \\ &\quad + f^*(x)\, 2nh \left(1 + \frac{1}{\epsilon}\right) E\left(\frac{M - m}{h(n - 1)p}\right). \end{aligned}$$

These inequalities are only valid at Lebesgue points x, i.e., at points where the Lebesgue density theorem is applicable. We are done if we can show that $E(1/p) \to 1$, and $E((M - m)/hnp) \to 0$. We can use a trick for the latter expression: let Q be the distance between the $\frac{1}{4}$ and $\frac{3}{4}$ quantiles of f. Then,

$$E\left(\frac{M - m}{hnp}\right) \leq 2E\left(\frac{M - m}{hn}\right) + \frac{Q}{hn} E\left(\frac{1}{p}\right),$$

and this is $o(1) + o(E(1/p))$ by our assumptions. Thus, we are left only with the proof of $E(1/p) \to 1$. Note that $1 - p$ is distributed as the range of n i.i.d. uniform $[0, 1]$ random variables. It has density $n(n - 1)x(1 - x)^{n-2}$, $0 < x < 1$ (see, e.g., [4, pp. 8–11]). Thus,

$$\begin{aligned} E\left(\frac{1}{p}\right) &= \int_0^1 (1 - x) x^{n-2} \frac{n(n - 1)}{x}\, dx \\ &= \frac{n(n - 1)}{(n - 1)(n - 2)} = \frac{n}{n - 2} \to 1, \end{aligned}$$

which was to be shown.

Proof of B: We will mimick the proof of B of Theorem 2.1. First, we note that

$$\frac{E(N^*(x))}{2nh} \leq f^*(x) E\left(\frac{n}{p} \cdot \frac{1}{2nh} \cdot 2\left(h + \frac{M - m}{n - 1}\right)\right).$$

This top bound is integrable with respect to $f(x)\, dx$ for all n, at least when $\int f^2 < \infty$. And its integral has a limit

$$\int f^* f \cdot \lim_{n\to\infty} E\left(\frac{1}{p}\left(1 + \frac{M - m}{h(n - 1)}\right)\right) = \int f^* f < \int f^{*2} < \infty.$$

Thus, by an extended version of the Lebesgue dominated convergence theorem, and part A of this Theorem,

$$\frac{1}{2nh} \int E(N^*(x))\, f(x)\, dx \to \int f^2.$$

This concludes the proof of B.

Proof of C: Let $H = (M - m)/(n - 1)$, and define $N''(x)$ and $N'(x)$ as $N(x)$ with δh and H instead of h, respectively, where $\delta > 0$ is a constant to be specified further on. Obviously, $\sup N^*(x) \leq \sup N(x) + \sup N'(x)$. Since $E(\sup N(x))/(2nh) \to y$ by Theorem 2.1, we need only show that $E(\sup N'(x))/(2nh) \to 0$. But we observe that for $0 < \delta < 1$,

$$\frac{E(\sup N'(x))}{2nh} \leqslant \delta \frac{E(\sup N''(x))}{2n\delta h} + E\left(\frac{\sup N'(x)}{2nh} I_{[H>\delta h]}\right)$$

$$\leqslant \delta(y + o(1)) + E(\sqrt{H/(\delta h)} \sup N''(x)/(2nh))$$

$$\leqslant \delta(y + o(1)) + \frac{E(\sup N''(x))}{2nh} + E\left(\frac{H}{h} \cdot \frac{\sup N''(x)}{2n\delta h}\right)$$

$$\leqslant 2\delta(y + o(1)) + \frac{1}{\delta} \frac{E(\sup N(x))}{2nh},$$

where I denotes the indicator function. Notice that given m, M, the remaining $n - 2X_j$'s are i.i.d. random variables with density f/p restricted to $[m, M]$. Its essential supremum cannot exceed y/p. Define $\epsilon = (y/p) - (1/2nh)$, and let n be so large that, $1/2nh < y/2$ (so that $\epsilon > y/(2p)$). We condition first on m, M and use the exponential bounding technique of the proof of Theorem 2.1.C:

$$E\left(\frac{\sup N(x)}{2nh} \,\middle|\, m, M\right)$$

$$\leqslant 2\frac{y}{p} + \frac{n}{2nh} P\left(\frac{\sup N(x)}{2nh} \geqslant 2\frac{y}{p} \,\middle|\, m, M\right)$$

$$\leqslant 2\frac{y}{p} + \frac{2n}{2h} \exp\left(-(n-2)\frac{\epsilon^2}{\epsilon + y/p}\right)$$

$$\leqslant 2\frac{y}{p} + \frac{n}{h} \exp\left(-(n-2)\frac{hy}{6p}\right)$$

$$\leqslant 2\frac{y}{p} + \frac{n}{h} \exp\left(-(n-2)\frac{hy}{6}\right).$$

Thus,

$$\frac{E(\sup N'(x))}{2nh} \leqslant 2\delta(y + o(1)) + \frac{2y}{\delta} E\left(\frac{H}{ph}\right) + E\left(\frac{H}{h}\right)\frac{n}{h\delta} \exp\left(-(n-2)\frac{hy}{6}\right).$$

The first term on the right-hand side can be made small by choosing δ small. The second term is $o(1)$ since $E(H/(ph)) = o(1)$ (see proof of Theorem 2.1.A). Both factors in the third term are $o(1)$ by our assumptions about H (i.e., $E(H) = o(h)$) and h (i.e., $nh/\log n \to \infty$). This concludes the proof of C.

Proof of Property 2.1: By Jensen's inequality, $E(e^{t|X|}) < \infty$ and $\log n = o(nh)$,

$$E(\max |X_j|) \leqslant \frac{1}{t} \log(E(e^{t \max |X_j|})) \leqslant \frac{1}{t} \log\left(\sum_{j=1}^{n} E(e^{t|X_j|})\right)$$

$$= \frac{1}{t} \log(nE(e^{t|X|})) = \frac{1}{t} \log n + \frac{1}{t} \log(E(e^{t|X|}))$$

$$= 0(\log n) = o(nh).$$

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading, MA: Addison-Wesley, 1976.

[2] G. Bennett, "Probability inequalities for the sum of independent random variables," *J. Amer. Statist. Ass.*, vol. 57, pp. 33–45, 1962.

[3] T. Cacoullos, "Estimation of a multivariate density," *Annals Inst. Statist. Math.*, vol. 18, pp. 179–189, 1965.

[4] H. A. David, *Order Statistics*. New York: Wiley, 1970.

[5] M. de Guzman, *Differentiation of Integrals in R^n*. Lecture Notes in Mathematics #481. Berlin, West Germany: Springer-Verlag, 1975.

[6] P. Deheuvels, "Estimation non paramétrique de la densité par histogrammes generalisés," *Revue de Statist. Appl.*, vol. 25, pp. 5–42, 1977.

[7] L. Devroye, "The equivalence of weak, strong and complete convergence in $L1$ for kernel density estimates," *Annals Statist.*, vol. 11, to be published.

[8] L. Devroye and C. S. Penrod, "Distribution-free lower bounds in density estimation," Appl. Res. Lab., Univ. Texas, Austin, Tech. Rep., 1982.

[9] L. Devroye and T. J. Wagner, "The $L1$ convergence of kernel density estimates," *Annals Statist.*, vol. 7, pp. 1136–1139, 1979.

[10] V. A. Epanechikov, "Nonparametric estimates of a multivariate probability density," *Theory Prob. Appl.*, vol. 14, pp. 153–158, 1969.

[11] C. A. R. Hoare, "Quicksort," *Comput. J.*, vol. 5, pp. 10–15, 1962.

[12] M. C. Jones and H. W. Lotwick, "A remark on Algorithm AS176. Kernel density estimation using the fast Fourier transform," *Appl. Statist.*, vol. 33, pp. 120–122, 1984.

[13] D. E. Knuth, *The Art of Computer Programming, vol. 3: Sorting and Searching*. Reading, MA: Addison-Wesley, 1973.

[14] E. Parzen, "On estimation of a probability density function and the mode," *Annals Math. Statist.*, vol. 33, pp. 1065–1076, 1962.

[15] J. G. Postaire and C. Vasseur, "A fast algorithm for nonparametric probability density estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, pp. 663–666, 1982.

[16] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function," *Annals Math. Statist.*, vol. 27, pp. 832–837, 1956.

[17] B. W. Silverman, "Algorithm AS176, Kernel density estimation using the fast Fourier transform," *Appl. Statist.*, vol. 31, pp. 93–99, 1982.

[18] R. C. Singleton, "Algorithm 347: An algorithm for sorting with minimal storage," *Commun. ACM*, vol. 12, pp. 185–187, 1969.

[19] R. L. Wheeden and A. Zygmund, *Measure and Integral*. New York: Marcel Dekker, 1977.