Knowl Inf Syst (2015) 42:285–317
DOI 10.1007/s10115-013-0712-0

REGULAR PAPER

# Fast adaptive kernel density estimator for data streams

**Arnold P. Boedihardjo · Chang-Tien Lu · Feng Chen**

Received: 29 November 2012 / Revised: 27 August 2013 / Accepted: 15 November 2013 /
Published online: 12 December 2013


**Abstract** The probability density function (PDF) is an effective data model for a variety of stream mining tasks. As such, accurate estimates of the PDF are essential to reducing the uncertainties and errors associated with mining results. The nonparametric adaptive kernel density estimator (AKDE) provides accurate, robust, and asymptotically consistent estimates of a PDF. However, due to AKDE's extensive computational requirements, it cannot be directly applied to the data stream environment. This paper describes the development of an AKDE approximation approach that heeds the constraints of the data stream environment and supports efficient processing of multiple queries. To this end, this work proposes (1) the concept of local regions to provide a partition-based variable bandwidth to capture local density structures and enhance estimation quality; (2) a suite of linear-pass methods to construct the local regions and kernel objects online; (3) an efficient multiple queries evaluation algorithm; (4) a set of approximate techniques to increase the throughput of multiple density queries processing; and (5) a fixed-size memory time-based sliding window that updates the kernel objects in linear time. Comprehensive experiments were conducted with real-world and synthetic data sets to validate the effectiveness and efficiency of the approach.

**Keywords** Data mining · Data streams · Kernel density estimation

A. P. Boedihardjo (✉)
U. S. Army Corps of Engineers, Alexandria, VA, USA
e-mail: arnold.p.boedihardjo@usace.army.mil

C. T. Lu
Computer Science Department, Virginia Tech, Falls Church, VA, USA
e-mail: ctlu@vt.edu

F. Chen
Computer Science Department, Carnegie Melon University, Pittsburgh, PA, USA
e-mail: fchen1@andrew.cmu.edu

## 1 Introduction

Advances in hardware and software technologies have caused a surge in the growth and availability of voluminous information. Data streams are realizations of vast information that are fast, continuous, mutable, ordered, and unbounded [5]. Numerous data streams stem from the ubiquitous time series and span a wide range of applications such as finance, medicine, and sensor networks. Some well-known streams are traded stock prices, brain electrical impulses (e.g., electroencephalograms), and roadway performance metrics (e.g., vehicle speed). Applying analysis and mining techniques can help deepen knowledge in these domains and enhance their applications. Therefore, the development of stream analysis tools can provide far-reaching impacts to the general discipline of stream mining.

Underpinning many stream mining tasks is the use of the probability density function (PDF) for the most recent data [3,5,15,34]. However, in real-world situations, the PDFs are usually unknown and therefore must be estimated. Examples of stream mining tasks that employ estimated PDFs include outlier detection by modeling a sensor's sample distribution and selecting data points of low probability [32]; concept drift detection via comparing the current and past data streams' probability density estimates [2]; and pattern discovery in Internet traffic by visualizing the estimated probability density function of arriving packets [34]. One could further the efficacy of probability density estimates by enabling queries for an explicit time range using a sliding window [3]. The extension would be able to respond to questions such as "What is the distribution of telnet connections within the last hour?" and "How have the roadway's speed and volume distributional patterns changed in the past two hours since the snow began?" Another desired feature of probability estimation is the inclusion of an efficient approach for estimating densities for multiple queries. As in the examples above, the outlier detection strategy in [32] approximates the distance-based outlier measure by invoking multiple queries within an interval. A variant to the above scenario is the case of data stream monitoring where several users simultaneously query various parts of the data distributions. Because existing stream-based density estimators are designed to generate a single query estimate, processing multiple queries within these estimators can lead to computational redundancies that degrade throughput. Hence, new estimation techniques must be developed which can effectively remove these redundancies to rapidly process multiple queries.

An effective technique to estimate an unknown probability density function is the non-parametric kernel density estimator (KDE). KDE possesses several advantages that include asymptotic consistency; rigorous mathematical foundation; generalization to other density estimators such as orthogonal series and histograms; and inheritance of the kernel function's continuity and differentiability properties [29,30]. The standard formulation of the univariate KDE for a query point $x$ is given as follows: For $n$ independent and identically distributed sample points (i.e., kernel centers) $x_1 \ldots x_n$ with corresponding weights $w_1 \ldots w_n$, bandwidth $h$, and a kernel function $K(\cdot)$, the kernel density estimate is

$$p_{KDE}(x) = \frac{1}{\sum_1^n w_i} \sum_{i=1}^{n} \frac{w_i}{h} K\left(\frac{x - x_i}{h}\right)$$

The accuracy of the KDE does not significantly depend on the choice of kernel function $K(\cdot)$, but rather on the selection of bandwidth $h$ [29]. The bandwidth is regarded as a smoothing parameter: A high $h$ value can generate a smooth shape density (reduce variance), whereas a low $h$ value tends to provide an under-smoothed estimate (reduce bias). The drawback of the KDE formulation is the requirement of assigning a global bandwidth. Due to the existence

of local features, a single global bandwidth may not be sufficient to model complex density structures (e.g., multimodal distributions). When the global bandwidth KDE technique is used as the core step in mining tasks, the generated results can be misleading and potentially disastrous for mission critical applications (e.g., military sensor surveillance). To overcome this problem, this paper proposes the use of an adaptive KDE (AKDE) to improve the estimation accuracy of local features within data streams. The AKDE is a variable bandwidth technique, which has been shown to be effective in capturing local density features [28–30]. The general formulation of the AKDE is as follows:

$$p_{AKDE}(x) = \frac{1}{\sum_1^n w_i} \sum_{i=1}^{n} \frac{w_i}{H(x_i)} K\left(\frac{x - x_i}{H(x_i)}\right), H(x_i) \propto f(x_i)^{-1} \tag{1}$$

where $H(x_i)$ is the bandwidth function that is inversely proportional to the true density $f(x_i)$.

### 1.1 Linear time adaptive KDE

Although the AKDE can produce superior estimation quality to the classical KDE, its computational cost ($O(n^2)$) far exceeds the traditional KDE ($O(n)$). In the AKDE, the bandwidth $H(\cdot)$ is computed from the true density $f(\cdot)$. Because the true density is unknown, a pilot function is defined to provide an estimate for $f(\cdot)$. Generally, the pilot function is modeled by the classical KDE. This choice implies that evaluating $H(\cdot)$ is an $O(n)$ operation. Therefore, computing a query under the AKDE is a $O(n^2)$ because $H(\cdot)$ is computed at least once for each sample point. This evaluation approach cannot be applied to data streams since it clearly infringes upon the linear-pass restriction. Due to AKDE's extensive computational cost, there are currently no works that provide adaptive kernel density estimates for the data stream environment.

### 1.2 Rapid multi-query estimation algorithms

Existing stream-based KDEs employ algorithms that operate on a single query element at a time. However, there are several stream mining tasks that require the generation of multiple estimates such as outlier detection and visualization. These stream mining tasks must submit the queries in sequence, resulting in several redundant computations and throughput degradation. Because the query points are determined ahead of time either by the mining algorithm or by application usage (e.g., multiple users submitting queries), new query processing methods can be constructed to exploit this prior information in order to improve overall query throughput.

This paper tackles the issue of developing efficient AKDE capable of rapidly generating multiple density estimates over data streams. To that end, we propose the online local region KDE (LR-KDE), an adaptive kernel density estimation framework for processing univariate data streams. The major components of the proposed framework are (1) a new partition-based variable bandwidth technique to capture local density structures and to enhance estimation quality, (2) a suite of *linear* pass methods to construct local regions and its corresponding kernel objects online, (3) a multiple density point evaluation algorithm that reduces the costs of local region and kernel object searches, (4) a set of interpolation-based techniques that provides approximation guarantees and further increases the query throughput, and (5) a fixed-size memory time-based sliding window that updates and expires the kernel objects in linear time. We also analyze the asymptotic costs and consistency property of the online LR-

KDE (see Sect. 6). Extensive experiments were conducted to demonstrate the effectiveness and efficiency of the approach.

The proceeding sections are organized as follows. Section 2 surveys the related works. Section 3 provides the theoretical preliminary. Section 4 details our proposed framework. The suite of algorithms for processing multiple density queries are described in Sect. 5, and the consistency results and cost complexities are presented in Sect. 6. Section 7 demonstrates empirical results and validation. The paper concludes with a summary of the research in Sect. 8.

## 2 Related works

Early efforts in computationally tractable KDEs can be found in the domain of *off-line* analysis. Zhang et al. [36,37] proposed a method to maintain a space-efficient KDE using the CF-tree to cluster a set of kernels into a single kernel known as the CF-kernel. The CF-kernel employs a global bandwidth that can lead to oversmoothing and loss of local density information. Gray et al. [13] proposed a kernel space partitioning technique by utilizing a KD-tree and bounded support kernels for off-line processing. The KD-tree reduces computations by effectively pruning kernels, which do not contribute to the density query.

Several recent works have been proposed for the online management of KDE. These online techniques can be classified into the following three categories:

1. *Grid-based KDE*—provides a uniform and discretized representation of the kernel points
2. *Sample-based KDE*—employs a sampling methodology on the data stream to reduce the total kernel size
3. *Cluster-based KDE*—utilizes kernel merging techniques to maintain a fixed storage and minimize the kernel merge error

### 2.1 Grid-based and sample-based

Aggarwal proposed a framework to capture the structural evolution of data streams using a grid-based KDE [2]. The kernels are summarized in a multidimensional grid where a common bandwidth is employed for each dimension. Concepts of forward and reverse density profiles are introduced to discover the occurrences of concept drifts. Subramaniam et al. [32] proposed an approach for outlier detection in sensor networks by modeling the densities of node observations. Their scheme employs a uniformly sampled sequence-based sliding window to summarize the kernels, and a global bandwidth is applied. Wegman et al. [34] introduced an online KDE for the analysis of Internet traffic. They suggested the use of a sequence-based and exponentially aging sliding window to accommodate a fixed storage environment. To derive estimates, a single bandwidth KDE is employed on the sliding window.

### 2.2 Cluster-based

Zhou et al. [38] introduced the M-kernel, a cluster-based KDE maintenance strategy that performs numerically based kernel mergers under a fading window. The merging algorithm combines two kernels to produce the minimum integrated absolute error between the original pair and the merged result. This scheme allows for a fixed memory representation of the kernels. However, under the M-kernel, the consistency of the estimate is not guaranteed and the approach can exhibit high update costs due to its numerical-based merging strategy. In a similar vein, Heinz et al. [16] proposed a constant time pairwise merging technique that

guarantees consistency. Their kernel method employs a single bandwidth that has been shown to be effective in approximating the classical KDE.

### 2.3 Multiple query processing

Efficient estimation of multiple density points has been proposed for off-line KDE techniques. These methods operate on a transformed estimation problem expressed as a convolution of the kernel weight and kernel influence function [33]. The convolution is efficiently computed via orthogonal series techniques such as discrete Fourier transform or fast Fourier transform [10,26]. However, estimating multiple points through convolution-based approaches is limited in two critical ways: (1) A grid-based KDE must be employed and (2) orthogonal series computation can only be applied when the underlying KDE utilizes a global bandwidth [33]. These two drawbacks can reduce the estimation quality due to the uniform space partitioning of the grid-based method and the global bandwidth's inability to effectively model local structures. For the stream-based estimators, multiple queries are resolved by invoking several single query estimates in a first-in-first-out order. These multiple invocations can result in significant computational overhead due to the redundant searches of (potential noncontributing) kernel objects and suboptimal sequencing of query order. For data streams, any excess time allocated to computations can result in denials of much needed estimates to the application. Due to the above issues, this work develops efficient algorithms for the LR-KDE that can rapidly estimate multiple density queries.

Most KDE approaches employ a single bandwidth strategy; hence, they cannot accurately estimate the stream's local features. The M-kernel, although it applies a variable bandwidth approach, is not assured to be asymptotically consistent. As a consequence, the M-kernel is not guaranteed to converge as the sample size increases. The proposed LR-KDE differentiates itself from existing works in the following aspects:

1. *Local feature estimation*—models local density features via partition-based bandwidth to improve estimation quality.
2. *Consistency*—assures asymptotic consistency under the proposed variable bandwidth strategy.
3. *Multi-query optimization*—generates multiple density estimates that eliminate redundant query operations and enhance overall query throughput with a worst-case cost of $O(DM)$ where $D$ is the query size and $M$ is the number of kernels.
4. *Linear-pass processing*—employs $O(M)$ algorithms to process kernel updates and density queries.
5. *Time-based window*—provides a fixed-size time-based sliding window.

Online histograms have also been proposed in the field of database optimization to provide query selectivity estimates and approximate queries [21]. Some online histograms include dynamic quantiles [12], equidepth histograms [11], and V-optimal histograms [14]. Due to the histogram's inherent discontinuities and slower convergence, the histogram may not be suited for the tasks of stream analysis [30]. Feature discretization (binning) methods have been proposed in the machine learning community to support the modeling of mixed format data [9,24]. There are two major categories, including supervised and unsupervised methods. Supervised methods include adaptive quantizers [7], supervised monothetic contrast criterions (MCC) [25], and predictive value maximization (PVM) [35]. Unsupervised methods include equal width binning, equal frequency binning, and unsupervised MCC [25]. Most of these methods were designed for classical prediction models such as decision tree and naive Bayes classifier. However, these methods did not consider the special properties of KDE,

such as the relative density variance of density estimates, and may not be directly applicable to the problem of adaptive KDE estimation.

## 3 Theoretical preliminary

A formal description of the density estimation problem is provided in the following: Given a data stream $S = \{x^{(1)}, x^{(i)}, \ldots, x^{(n)}\}$ where $x^{(i)}$ is a real-valued scalar (i.e., sample point) and each $x^{(i)}$ is associated with a time stamp $\tau(x^{(i)}) \in [\tau_{\min}, \tau_{\max}]$, it generates and maintains an adaptive kernel density estimator $\hat{f}_{AKDE}(\cdot)$ of $S$. The storage, update, and query costs of $\hat{f}_{AKDE}(\cdot)$ should be at most $O(M)$ where $M$ is a constant and $M \ll \|S\| = n$. For multiple density estimation, a multiple density estimator is defined and described as follows: $\hat{f}_{MULTI\_AKDE}\left(\langle d_1, d_2, \ldots, d_{|D|}\rangle\right) = \langle \hat{f}_{AKDE}(d_1), \hat{f}_{AKDE}(d_2), \ldots, \hat{f}_{AKDE}(d_{|D|})\rangle$ where $D$ is the query set and $d_i$ is an element of $D$. The storage and update costs should not exceed those of $\hat{f}_{AKDE}(\cdot)$ and the query cost bounded to $O(DM)$.

With respect to stream applications, one of the fundamental issues of the AKDE is its high computational cost for determining the bandwidth $H(\cdot)$ (Eq. 1). To reduce this computational requirement, a bandwidth approximation technique is proposed for the AKDE. Let $T = \{x_i : x_i \leq x_{i+1}, 1 \leq i \leq n, x_i \in S\}$ be an ordered representation of the kernels in $S$. Define the relative density variance, $R(k, l)$, as the sample variance of the set of estimated densities at $x_k \ldots x_l \in T$ where $1 \leq k \leq l \leq n$. The bandwidth approximation procedure is given as follows:

1. Partition $T$ into $Q$ local regions (i.e., intervals) such that each local region's $R(\cdot, \cdot)$ value is minimized
2. For each local region, assign a bandwidth that is unique to its constituent kernels

The above scheme serves to capture the local densities within the total span of the distribution. The obtained approximation is consistent with the structure of AKDE's bandwidth, i.e., similar bandwidth values are assigned to kernels of similar densities. Hence, the local regions can be seen as a piecewise constant representation of $H(\cdot)$.

Two design challenges exist in applying the above approximation approach: (1) the efficient derivation of the relative density variance and (2) the development of a technique for local region identification. In the following, the pairwise adjacent distance uniformity theorem is introduced to provide an efficient venue for estimating the density variance. The theorem is followed by the definition of an optimization problem for the task of identifying local regions.

### 3.1 Derivation of relative density variance

As previously defined, local regions provide a total and disjoint partitioning of the kernel domain which minimizes the intra-variance of the density estimates. A unique bandwidth is assigned to each local region based on the regional kernel characteristics. If each kernel is given its own unique local region, then the local region-based KDE (with the appropriate bandwidth function) is a reformulation of the traditional AKDE. However, if the number of the local regions is less than the number of kernels, then the local region KDE is an approximation of the AKDE. The problem now is how to derive a method that can efficiently calculate the density estimate variance for a given range of kernels. One possible approach is to estimate the densities via the traditional KDE approach. However, this poses the same computational issue as the AKDE. An alternative and more viable solution is to employ the

pairwise and adjacent kernel distances to derive relevant properties of the density variance. To that endeavor, the pairwise adjacent distance uniformity theorem is introduced.

Let $G$ be a set of ordered and identically weighted kernels whose pairwise and adjacent distance variance is zero; then under certain conditions, it can be shown that the densities at the kernel centers (under a global bandwidth KDE) in $G$ are uniform. The significance of this theorem is that it provides a venue of estimating the density variance through information of the kernels' pairwise and adjacent distances (PAD). As a result, computationally tractable bandwidth approximations can be developed from the kernels' PAD information. The proof of the PAD uniformity theorem is as follows:

**Theorem 3.1** (Pairwise adjacent distance uniformity) *Let* $V = \{x_i : x_i \leq x_{i+1} and 1 \leq i \leq n\}$ *be a set of sorted kernel centers associated with a bounded and radially symmetric kernel function with uniform weight and bandwidth. Furthermore, let the sorted kernels be adjacently equidistant such that* $|x_{i+1} - x_i| = |x_{j+1} - x_j| \forall i, j$. *Suppose* $G = \{x_k : x_1 + bandwidth \leq x_k \leq x_n - bandwidth\}$. *Then for the kernel density estimate,* $\hat{f}(x)$, *the following property must hold:* $\hat{f}(x_k) = \hat{f}(x_l) \forall x_k, x_l \in G$.

*Proof* Let $x_k, x_l \in V$ be any two kernel centers and define a set $I_x = \{x_i : |x_i - x| < bandwidth\}$. $I_x$ is the set of kernel centers for which their corresponding bandwidths intersect $x$; thus, $I_x$ possess all the kernels that contribute nonzero values to the kernel density estimate $\hat{f}(x)$. Consider $I_{x_k}$ and $I_{x_l}$, and without loss of generality, choose a kernel center, $\alpha \leq x_k$, from $I_{x_k}$. Because all the kernel centers within $G$ are adjacently equidistant, there must exist an element, $\beta \leq x_l$, from $I_{x_l}$ such that $x_k - \alpha = x_l - \beta$. Since the kernels are radially symmetric with equal bandwidth and identically weighted, the contribution of $\alpha$ to $\hat{f}(x_k)$ is equal to the contribution of $\beta$ to $\hat{f}(x_l)$. For any chosen $\alpha$ in $I_{x_k}$, there exists a corresponding $\beta$ in $I_{x_l}$ for which their contributions are equal. This relationship also holds for any $\beta$ in $I_{x_l}$ corresponding to an $\alpha$ in $I_{x_k}$. Hence, there is a one-to-one relationship between $\alpha$ in $I_{x_k}$ and $\beta$ in $I_{x_l}$, which implies that the sum of the kernel contributions of $I_{x_k}$ to $\hat{f}(x_k)$ and $I_{x_l}$ to $\hat{f}(x_l)$ is equal. Therefore, $\hat{f}(x_k) = \hat{f}(x_l) \forall x_k, x_l \in G$.

### 3.2 Optimization problem for local region identification

Leveraging upon Theorem 3.1, identification of the local regions can be achieved by minimizing the variance of the kernels' pairwise and adjacent distance values. In the following, an optimization problem is established to identify local regions based on the kernels' PAD information. Assume that a local region, $L$, possesses a set of kernels (centers), $y_j \ldots y_m$, where $y_j \leq y_{j+1} \leq y_m$ for $1 \leq j \leq m \leq n$, and $n$ is the total number of kernels. Set $L$'s $i$th adjacent distance to be $L_d(i) = y_{j+i+1} - y_{j+i}$, and define the variance of $L$'s kernel pairwise and adjacent distances as follows:

$$\text{var}(L_d) = \frac{1}{\|L\| - 1} \sum_{i=1}^{\|L\|-1} \left( L_d(i) - \frac{\sum_1^{\|L\|-1} L_d(i)}{\|L\| - 1} \right)^2 \tag{2}$$

where $\|L\|$ is the number of kernels in $L$.

In practice, the density estimate makes use of arbitrarily weighted kernels. Hence, consideration of varying weights must be made in the local region formulation. Let $z_1 \ldots z_n$ be a set of sorted kernel centers assigned to a bounded, radially symmetric, and equal bandwidth kernel function, such that $|z_{i+1} - z_i| = |z_{j+1} - z_j| \forall i, j$. Let $w(z_i)$ be the nonnegative weight of kernel center $z_i$ and $\hat{f}(z_i)$ be the density estimate at $z_i$. From the KDE definition,

$\hat{f}(z_i) \propto w(z_i) \Rightarrow var(\hat{f}(Z)) \propto var(w(Z))$. Therefore, minimizing $var(\hat{f}(Z))$ is equivalent to minimizing $\text{var}(w(Z))$. Let $L_{kw}(i)$ be the weight of kernel $y_{j+i}$ in $L$, and define $L$'s kernel weight variance, $\text{var}(L_{kw})$, as follows:

$$\text{var}(L_{kw}) = \frac{1}{\|L\|} \sum_{i=1}^{\|L\|} \left( L_{kw}(i) - \frac{\sum_1^{\|L\|} L_{kw}(i)}{L} \right)^2 \tag{3}$$

Using the local region PAD (Eq. 2) and the kernel weight variance (Eq. 3) formulae, partition the entire sample points $y_1 \ldots y_n$ into $Q$ disjoint local regions by minimizing the aggregate variance of all $L_d$ and $L_{kw}$:

$$\text{Min} \sum_{i=1}^{Q} \|L\| \left( \text{var}(L_d^{(i)}) + \mu \cdot \text{var}\left( L_{kw}^{(i)} \right) \right) \tag{4}$$

where $\mu$ is the weight assigned to $L$'s kernel weight variance.

The solution to the above local region optimization problem generates local regions with minimum overall intra-density variation. Moreover, the problem is solved by exclusively employing the kernels' PAD and weight information which are amenable to efficient implementations.

## 4 Proposed approach: online local region KDE

The local region identification problem can be solved via dynamic programming techniques in time $O(n^2 Q)$ where $Q$ is the number of local regions; however, this solution exceeds the constraints of the data stream problem. Therefore, an incremental local region identification technique is proposed. The technique employs a locally optimal decision strategy to reduce the identification task to $O(nQ)$. As a result, the online local region KDE (LR-KDE) is proposed for the efficient generation of density estimates in data streams. An overview of the LR-KDE architecture is given in Sect. 4.1. Detailed descriptions of the local region and kernel constructions are provided in Sects. 4.2 and 4.3. Lastly, the density computation algorithm is described in Sect. 4.4.

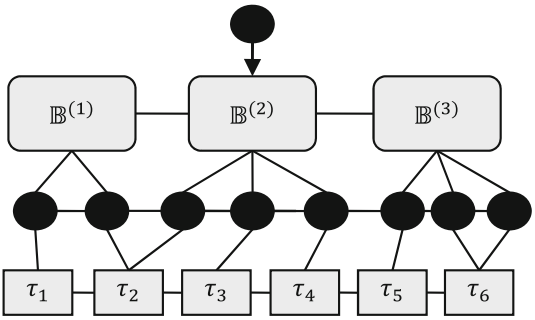
### 4.1 Online LR-KDE overview

The online LR-KDE is composed of the following primary components:

1. *Local region management*—identifies and manages local regions by employing a locally optimal dynamic binning method for the entire set of kernels.
2. *Kernel maintenance*—updates the kernels with new data points in a fixed-size memory environment, and maintains and provides density evaluation of kernels whose time stamps are within $[\tau_{\min}, \tau_{\max}]$.

Figure 1 illustrates the LR-KDE approach. An online binning method is applied on the set of all kernels where each bin represents a local region. The bins are maintained in the data structure, *bin list*, which can store at most $Q$ number of bins. The *kernel* set manages and organizes *all* kernels in a sorted queue ordered by their centers. A maximum of $M \geq Q$ kernels is maintained in the kernel set. The *time list* structure is a sorted queue of kernel arrival times. The objective of the *time list* is to model a time-based sliding window (using the FIFO policy) and to support efficient operations of kernel expirations and insertions. The following is a summary of the main operations for inserting a new data point $K_{new}$:

**Fig. 1** LR-KDE architecture




1. *Bin list update*: Find the bin whose interval (bounded by its corresponding minimum and maximum kernels) intersects $K_{new}$; add a reference to $K_{new}$ to the bin's set of kernels; and forward $K_{new}$ to the *kernel set* list. If no intersecting bin exists, create a new bin for $K_{new}$ and (possibly) merge two adjacent bins to maintain a maximum of $Q$ bins.
2. *Kernel set insertion*: Insert $K_{new}$ into position by searching the *kernel set*. If after the insert, the kernel set size is greater than $M$, then merge two kernels that minimize the overall accuracy loss.
3. *Time list synchronization*: After $K_{new}$ is added to the *kernel set*, its arrival time is appended to the tail of the *time list*. If a kernel merger occurs, then its corresponding time stamps in the time list will also need to be updated. The head of the *time list* will be verified against $\tau_{\min}$ in order to remove expired kernels.

### 4.2 Local region management

The following describes an incremental bin management approach for the identification of local regions. A formal definition of the bin and its property are given as follows:

*Bin Vector*—a bin vector, $\mathbb{B}$, is defined as follows: $\mathbb{B}=\left[b_{ks}, b_{ss}, b_{ws}, b_{wss}, b_{wcs}, b_{wcss}, b_{sp}\right]$

$b_{ks}$ is the set of kernels in $\mathbb{B}$ sorted by their kernel centers. The following feature definitions refer to set $b_{ks}$. $b_{ss}$ denotes the squared sum of the kernel centers. $b_{ws}$ gives the sum of the kernel weights. $b_{wss}$ yields the squared sum of the kernel weights. $b_{wcs}$ indicates the sum of the weighted kernel centers. $b_{wcss}$ provides the weighted squared sum of the kernel centers. Let $x_1 \ldots x_{b_c}$ be the sorted kernel centers in $b_{ks}$, and then the sum of adjacent kernel centers product is $b_{sp} = \sum_{i=1}^{b_c-1} x_i x_{i+1}$.

The bin vector possesses the additivity property [36]. This property allows the combining of bins in constant time. Let $\mathbb{B}^{(1)}$ and $\mathbb{B}^{(2)}$ be a pair of disjoint bins, $\max(b_{ks}^{(1)})$ = the maximum kernel center value in $b_{ks}^{(1)}$, and $\min(b_{ks}^{(2)})$ = minimum kernel center value in $b_{ks}^{(2)}$. If $\max(b_{ks}^{(1)}) < \min(b_{ks}^{(2)})$, then the sum of bin vectors $\mathbb{B}^{(1)} + \mathbb{B}^{(2)}$ is:

$$\mathbb{B}^{(1)} + \mathbb{B}^{(2)} = \Big[ lb_{ks}^{(1)} \cup b_{ks}^{(2)}, b_{ss}^{(1)} + b_{ss}^{(2)}, b_{ws}^{(1)} + b_{ws}^{(2)}, b_{wss}^{(1)} + b_{wss}^{(2)}, b_{wcs}^{(1)} + b_{wcs}^{(2)}, b_{wcss}^{(1)} + b_{wcss}^{(2)}, b_{sp}^{(1)} + b_{sp}^{(2)} + \max(b_{ks}^{(1)}) \cdot \min(b_{ks}^{(2)}) \Big] \quad (5)$$

*Bin creation and merger* Bins are updated as new kernels enter the system. For the *bin list*, there are two cases to address when $K_{new}$ enters: (Case 1) The number of bins in *bin list* is less than $Q$, and (Case 2) the number of bins in *bin list* is equal to $Q$.

**Case 1** *(The number of bins in bin list is less than* $\boldsymbol{Q}$*)*: Suppose that the kernel center value of $K_{new}$ does not intersect any of the bins' regions, then a new bin is created with $K_{new}$ as its

center point and its bin vector values initialized with $K_{new}$. If the kernel center value $K_{new}$ does intersect a bin's region, then the vector of elements of that bin is updated with $K_{new}$. After a bin is created or an existing one is updated, a reference to this kernel is added to the *kernel set* (as shown in Fig. 1) if this kernel center does exist in the *kernel set*. Otherwise, the kernel with the same center has its weight updated. The bin management algorithm continues in this manner until $Q$ bins have formed.

**Case 2** *(The number of bins in bin list is equal to* $\boldsymbol{Q}$*)*: Assume that the kernel center of $K_{new}$ does not intersect any of the bins' regions. In this scenario, a new bin is created for $K_{new}$ and two bins (including one formed by $K_{new}$) are merged to maintain $Q$ number of bins. Details of the merger are described below. In the case that kernel center of $K_{new}$ intersects a bin's region, then the intersecting bin's vector is updated with $K_{new}$. The *kernel set* is updated with $K_{new}$ in a similar fashion to Case 1.

The merger of two bins is defined by its additivity property (Eq. 5). Because the bins represent local regions, they must remain continuous and mutually disjoint. Therefore, the merger can only occur between adjacent bins. The merger candidates are selected based on the objective function of the local region optimization (Eq. 4). The following is an expanded expression of the local region PAD variance (Eq. 2) in terms of kernel centers $x_1, x_i, \ldots, x_n$:

$$var(L_d) = \frac{2}{\|L\| - 1}\left(\sum_{i=1}^{\|L\|} x_i^2 - \sum_{i=1}^{\|L\|-1} x_i x_{i+1} - \left(\frac{x_1^2 + x_{\|L\|^2}}{2}\right)\right) - \left(\frac{x_{\|L\|} - x_1}{\|L\| - 1}\right)^2 \tag{6}$$

With the above equation, the PAD variance of a bin can be directly computed using its corresponding kernel center values. Because the bin vectors maintain several statistics of the kernel centers, we can exploit these summaries to efficiently calculate the $i$th bin's PAD variance in constant time as follows:

$$\mathbb{B}_{dv}^{(i)} = \frac{2}{\|b_{ks}^{(i)}\| - 1}\left(b_{ss}^{(i)} - b_{sp}^{(i)} - \left(\frac{\min\left(b_{ks}^{(i)}\right)^2 + \max\left(b_{ks}^{(i)}\right)^2}{2}\right) - \frac{(\max(b_{ks}^{(i)})\min(b_{ks}^{(i)}))^2}{2\|b_{ks}^{(i)}\| - 2}\right) \tag{7}$$

Similarly, the statistics can also be used to calculate the $i$th bin's kernel weight variance in constant time as follows (reformulation of Eq. 3):

$$\mathbb{B}_{wv}^{(i)} = \frac{b_{wss}^{(i)}}{\|b_{ks}^{(i)}\|} - \left(\frac{b_{ws}^{(i)}}{\|b_{ks}^{(i)}\|}\right)^2 \tag{8}$$

Therefore, the objective function of bin mergers can be computed as follows:

$$\text{Min } J = \sum_{i=1}^{Q} \|b_{ks}^{(i)}\| \left(\mathbb{B}_{dv}^{(i)} + \mu \cdot \mathbb{B}_{wv}^{(i)}\right) \tag{9}$$

for $i = 1 \ldots Q$.

When $K_{new}$ is added, updating all of the bin's features (with the exception of $b_{sp}$) follows straightforward algebraic calculations and incurs constant time execution. To efficiently update $b_{sp}$, the following operations are employed:

Find the position of $K_{new}$ within $b_{ks}$, and set $r$ to this index position. This implies that the kernels, $x_{r-1}$ and $x_{r+1}$, are $K_{new}$'s adjacent neighbors. Then update $b_{sp}$ as follows:

$$b_{sp}^{(new)} = b_{sp}^{(old)} - x_{r+1}x_{r-1} + x_{r-1}x_r + x_r x_{r+1} \tag{10}$$

### 4.3 Kernel maintenance

In order to maintain the kernels in a fixed memory environment, a kernel clustering paradigm is employed. If the size of the *kernel set* is $M$, then the insertion of $K_{new}$ will cause an overflow and invoke the merger of two kernels. Let $G^{(1)}(x)$ and $G^{(2)}(x)$ be weighted kernels, and then the merged kernel, $G^{(merge)}(x)$, is determined by utilizing a kernel merging approach as follows [19]:

$$G^{(1)}(x) = w^{(1)} K\left(\frac{x_1 - x}{h}\right), \quad G^{(2)}(x) = w^{(2)} K\left(\frac{x_2 - x}{h}\right)$$
$$G^{(merge)}(x) = (w^{(1)} + w^{(2)}) K\left(\frac{x_{merge} - x}{h}\right) \tag{11}$$

where $x_{merge}$ is the merged kernel center of $x_1$ and $x_2$.

$G^{(1)}(x)$ and $G^{(2)}(x)$ are selected such that the following $L_2$ error distance is minimized:

$$L_2 = \int_{-\infty}^{\infty} \left(G^{(1)}(x) + G^{(2)}(x) - G^{(merge)}(x)\right)^2 dx \tag{12}$$

It can be shown that the $L_2$ distance increases proportionally with the kernel distance; hence, the mergers only occur between adjacent kernels. It can be shown that minimizing $L_2$ error can be done in constant time [15].

The remainder of this section presents the time-based sliding window algorithm that ensures all elements in *kernel set* are within the time range, $[\tau_{\min}, \tau_{\max}]$. The algorithm is followed by a description of the density evaluation which leverages upon the *bin list* structure for efficient computation. Lastly, the selected kernel function and bandwidth forms are provided.

*Time-based sliding window* Let $\tau_{tl}$ be the length of the time window. To produce a sliding window, set $\tau_{\max}$ to the current time and set $\tau_{\min} = \tau_{\max} - \tau_{tl}$. The *time list* is implemented as a first-in-first-out queue with the head node being the oldest time stamp. When $K_{new}$ is inserted into the *kernel set*, the time handling algorithm inserts the kernel's arrival time, $g_{ta}$, and the kernel's extended time span, $g_{ets}$ (i.e., time to remain in window *after* expiration and initially set to 0) to the tail of time list. Assume that two kernels, $G^{(1)}$ and $G^{(2)}$, are merged to become $G^{(merge)}$. The *time list* updating process proceeds as follows:

1. Remove the corresponding *time list* nodes of $K^{(1)}$ and $K^{(2)}$
2. Define $G^{(merge)}$'s arrival time and time span as follows:

$$G_{ta}^{(merge)} = \min\left\{g_{ta}^{(1)}, g_{ta}^{(2)}\right\} \tag{13}$$
$$G_{ts}^{(merge)} = \max\left\{g_{ta}^{(1)} + g_{ets}^{(1)}, g_{ta}^{(2)} + g_{ets}^{(2)}\right\} - g_{sta}^{(merge)} \tag{14}$$

3. Insert $K^{(merge)}$'s arrival time into the time list

Kernel expiration is performed by comparing the *time list*'s head node, deleting all associated kernels with $g_{ta} < \tau_{\min}$ and $g_{ets} = 0$, and updating the *bin list*. As for expiring a kernel with $g_{ets} > 0$, assume that the weight of the kernel is uniformly distributed across its time span. Therefore, the kernel will remain in the *kernel set*, but its weight will be adjusted based on the following function:

$$g_w^{(updated)} = \left(1 - \frac{\tau_{\min} - g_{ta}^{(old)}}{g_{ets}^{(old)}}\right) g_w^{(old)} \tag{15}$$

where $g_w^{(i)}$ is the weight of kernel $i$.

### 4.4 Single query computation

The following describes the algorithm to solve a single density query which takes advantage of the local region pruning capability. The corresponding kernel function and bandwidth forms are also discussed. For a single query point $d$, the evaluation algorithm proceeds as follows:

1. *Bin search and kernel filtering:* Find the relevant set of bins that can potentially contribute a nonzero value to $d$. Let $h^{(\mathbb{B})}$ be the bandwidth of kernels in bin $\mathbb{B}$, and then the bins can be found by scanning the *bin list* and selecting those bins that fulfill the following condition: $d \cap \left[\min(\mathbb{B}) - h^{(\mathbb{B})}, \max(\mathbb{B}) + h^{(\mathbb{B})}\right] \neq \emptyset$.
2. *Kernel search:* Scan the kernels within each relevant bin and sum the density contribution of kernels whose support intersect is $d$. Formally stated, let $K$ be a kernel in $\mathbb{B}$, then compute the sum of kernel density contributions where $d \cap \left[K - h^{(\mathbb{B})}, K + h^{(\mathbb{B})}\right] \neq \emptyset$.

The evaluation technique presented above capitalizes upon the bin list structure by effectively pruning kernels which do not contribute to the final estimation result. Furthermore, the bandwidth is computed from bin features in constant time.

*Kernel function and bandwidth forms* The class of admissible kernel functions must satisfy the conditions of Theorem 3.1 which requires that the kernels be (1) radially symmetric and (2) compactly supported. The compact support property also eases the burden of computing the density estimates by eliminating kernels with $\left|\frac{x}{h}\right| \geq 1$. Some kernels of the admissible class are the Bi-weight, Rectangular, and Epanechnikov kernels [30]. It has been shown that the Epanechnikov kernel minimizes the asymptotic mean integrated squared error (AMISE), and therefore, it is optimal among all other kernels [23]. The Epanechnikov kernel is given as follows:

$$K(x) = \frac{3}{4}(1 - x^2) \text{ for } |x| < 1, \text{ and 0 otherwise} \tag{16}$$

Due to Epanechnikov kernel's compact support, radial symmetry, and optimality w.r.t. the ASIME, this kernel is chosen for the proposed LR-KDE.

Each bin of the LR-KDE describes a region of similar densities; hence, the captured distribution within each bin can be expected to be unimodal. Also recall that the LR-KDE assigns a unique bandwidth to each bin. Therefore, for each bin, the chosen bandwidth form is the *normal rule* which has been shown to perform well under a wide range of unimodal distributions [30]:

$$h^{(\mathbb{B})} = \sqrt{5}\sigma^{(\mathbb{B})} \quad \sqrt[-5]{n^{(\mathbb{B})}} \tag{17}$$

where $\sigma^{(\mathbb{B})}$ is the kernel centers' standard deviation of bin $\mathbb{B}$ and $n^{(\mathbb{B})}$ is the number of kernels in bin $\mathbb{B}$.

Note that $\sigma^{(\mathbb{B})}$ can be directly calculated from the bin features, i.e., $\sigma^{(\mathbb{B})} = \sqrt{\frac{b_{wcss}}{b_{ws}} - \left(\frac{b_{wcs}}{b_{ws}}\right)^2}$; therefore, computing the bandwidth, $h^{(\mathbb{B})}$, is achieved in $O(1)$ time.

## 5 Multiple density query processing framework

This section describes two main approaches to process multiple density queries within the LR-KDE. First, a multiple density query algorithm is provided that reduces the sequential

processing of the single query estimates by minimizing the redundant searches for local regions and kernel objects. Second, an optimized multiple density query algorithm is proposed that *approximates* the above exact multiple density query algorithm by utilizing constant time interpolation techniques. For the optimized approach, three variants are developed based on different data models.

### 5.1 Multiple kernel density estimate generation

When evaluating multiple density queries using the single query algorithm (Sect. 4.4), the bin search and kernel filtering (Step 1) are executed in proportion to the number of query points. Additionally, the kernel search (Step 2) will visit noncontributing objects (if they exist) for each density query. Consequently, this results in redundant operations for queries that intersect an identical set of bins. The situation occurs for any pair of queries that are strictly within $h^{(\mathbb{B}_1)} + h^{(\mathbb{B}_2)}$ distance apart where $\mathbb{B}_1 = \mathbb{B}_2$ or $\mathbb{B}_1$ and $\mathbb{B}_2$ are adjacent bins. Another source of inefficiency arises from the kernel search routine where potentially noncontributing kernel objects are visited multiple times. Since these noncontributing kernels do not affect the final estimates, their processing can be eliminated when evaluating multiple density queries. The set of noncontributing kernels are guaranteed to be empty for a query $d$ when the intersecting bin's bandwidth-adjusted boundaries $\left[\min(b_{ks}) - h^{\mathbb{B}}, \max(b_{ks}) + h^{\mathbb{B}}\right]$ are strictly within the query's contribution range $(d - h^{\mathbb{B}}, d + h^{\mathbb{B}})$. However, depending on the distribution of the query set, the queries can produce unnecessarily large amounts of visits to noncontributing kernel objects (e.g., queries close to the bin boundaries). The following multiple density query algorithm minimizes the redundant processing associated with the single query algorithm.

*Multiple density query algorithm* The multiple density query algorithm for a given input query set $D = \left\{d_1, d_2, \ldots, d_{|D|}\right\}$ is as follows:

1. *Query set preprocessing:* Sort the query elements into an array.
2. *Kernel filtering:* Scan the *bin list* to obtain a set of bins for which their bandwidth-adjusted intervals intersect the query range, i.e., $[\min(D), \max(D)]$. This process removes and filters out kernel objects that are guaranteed to provide no contribution to $D$.
3. *Density aggregation:* For each $\mathbb{B}$ of the intersecting bins obtained from Step 2, visit each of the corresponding kernel object, perform a binary search on the sorted queries $D$ using the kernel object center $x$ as key, and calculate/aggregate the density contributions of all queries that intersect the kernel's bandwidth range $\left[x - h^{(\mathbb{B})}, x + h^{(\mathbb{B})}\right]$.

The multiple density query algorithm above removes the redundant filtering of kernels by invoking a one-time pass of the *bin list* to derive a filtered set of kernel objects. The redundant visits to noncontributing kernels within the filtered set are minimized by only invoking a single pass over the candidate kernels that can be much less than the multiple passes required (proportional to $|D|$). Lastly, the presorting of the queries reduces each query search to a logarithmic cost when scanning the kernel objects.

### 5.2 Optimized multiple query processing

The following describes an approach that further improves the multiple density query algorithm. If some errors with respect to the single density query are allowed, then the operations related to the kernel object search and aggregation can be reduced and substituted with a fast interpolation technique. A proof on the error bound for the optimized methods is provided in Sect. 6.2. The following provides the optimized multiple density query approach.

*Optimized multiple density query framework* The optimized multiple density query approach for a given input query set $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is as follows:

1. *Control point extraction:* Generate a sufficiently small set of control points from $D$ (with size $s|D|$ where $0 < s \leq 1$) and output the control points in sorted order.
2. *Control point evaluation:* Estimate the densities of the control points obtained from Step 1 by employing the exact multiple density query algorithm.
3. *Query set interpolation:* Approximate the queries in $D$ by utilizing a linear regression technique to interpolate density estimates. Other regression techniques can also be used to further enhance accuracy or reduce the number of control points required.

The above algorithm significantly reduces the calculation necessary for exact densities ($O((|D| + M)\log(|D|) + |D|\,M)$) and generates approximate solutions ($O(M\log(|C|)) + |D|\,M$) where $|C| \leq |D|$) through fast linear interpolation (see Sect. 5.3 for derivations). In order to provide accurate results (relative to the exact method), the extracted control points are designed to model the distribution of the query set. Hence, three variants are proposed which employ different approaches to generate the control points in Step 1. The three variants are *random-based generation, uniform-based generation, and histogram-based generation*.

The **random-based generation** technique employs a random sampling technique that produces a sorted set of independent and identically distributed subsamples of the query set $D$ which results in a total query cost of $O((|C| + M)(\log(|C|)) + |D|M)$ where $|C|$ is the size of the control points. In the **uniform-based generation** approach, the explicit requirement to sort the control points is removed. In this approach, equidistant and ordered control points for the span of $D$ are generated, which requires only a single pass on $D$. Hence, the resulting query cost is reduced to $O(M\log(|C|) + |D|M)$. However, a drawback of this approach is its explicit embedding of a uniformly distributed query set which can generate inaccurate control points for highly skewed query distributions. In order to better model the queries' distribution and avoid the need to sort the control points, the **histogram-based generation** is proposed. First, a histogram of the query set is calculated to guide the assignment of control points. Second, the control points are assigned to locations that are locally equidistant within a histogram bucket, and the ratio of locally equidistant points to the total number of control points is identical to the bucket's frequency. Using the histogram-based generation method, control points are generated that can effectively model varying query distributions (e.g., skewed distribution). Because the histogram-based approach generates control points with a cost linear to $|D|$, the total query cost is $O(M\log(|C|) + |D|M)$. For all of the above three approaches, an initial pass over the query set is performed to determine the minimum and maximum elements required for interpolation.

## 6 Analysis

This section provides the consistency results and cost analyses of the LR-KDE. An important criterion for any KDE is its consistency, that is, as the number of samples approaches infinity, the KDE converges to the true density. The time/space complexities of the online maintenance technique and density evaluation approach are analyzed to guarantee that the LR-KDE heeds the constraints of the data stream environment. Section 6.1 provides the proof of LR-KDE's asymptotic consistency. Section 6.2 gives the error bound for the optimized multiple density evaluation. Section 6.3 analyzes the costs for maintaining the local regions and kernel objects. Lastly, Sects. 6.4 and 6.5 provide the density evaluation costs for the single and multiple density queries, respectively.

### 6.1 Asymptotic consistency

Assume that all $n$ samples are uniquely accessible and continuously persistent. To prove the consistency of the local region KDE, it suffices to show that the density estimate within any local region fulfills Parzen's condition for the asymptotic convergence of a KDE [27]. Parzen provides a sufficient condition described as follows:

If kernel $K(\cdot)$ is a bounded Borel function with

$$\int |K(t)|\,dt < \infty, \int K(t)dt = 1 \text{ and } |tK(t)| \to 0 \text{ as } |t| \to \infty \tag{18}$$

and bandwidth $h_n$ indexed on $n$ sample points satisfies

$$h_n \to 0 \text{ and } nh_n \to \infty \text{ as } n \to \infty \tag{19}$$

then for the KDE, $\hat{f}(x)$, and true density, $f(x)$,

$$\hat{f}(x) \to f(x) \text{ in probability as } n \to \infty \tag{20}$$

Since a local region employs the Epanechnikov kernel, the kernel conditions given by Parzen are completely satisfied. Recall that the bandwidth selected for a local region is the normal rule: $h_n = \sqrt{5}\sigma \quad \sqrt[-5]{n}$. Hence, it can be seen that the following holds: $h_n \to 0$ as $n \to \infty$. Therefore, Parzen's first bandwidth condition is satisfied. As for the second condition, note that a local region is continuous and has compact support, which implies

$$\sup(\sigma) = c, \text{ where } c \text{ is a constant}$$

which implies

$$\frac{h_n}{n} = \frac{\sqrt{5}\sigma \ \sqrt[-5]{n}}{n} = \frac{\sqrt{5}c}{n^{6/5}} \to 0 \text{ as } n \to \infty \Rightarrow nh \to \infty \tag{21}$$

Therefore, $\hat{f}(x) \to f(x)$ in probability as $n \to \infty$

### 6.2 Error bound for optimized multiple density evaluation

Given that the *normal rule* is used to calculate the local region bandwidths and linear interpolation is employed to approximate the densities, then the error at location $z$ of the optimized multiple density evaluation techniques (random-based, uniform-based, and histogram-based generation methods) with respect to the exact LR-KDE is as follows:

$$e(z) \leq \frac{5^{2/3}(C_0 - C_1)^2 w_{\max}^{3/5}}{8W\sigma_{\min}^3} max_{y\in[C_0,C_1]} \sum_{i=1}^{M} \left| K'' \left( \frac{(y - x_i)w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \right| \tag{22}$$

where $C_0$ and $C_1$ are the control points used to evaluate $z$, $W$ is the sum of kernel weights in the *kernel list*, $\sigma_{\min}$ is the minimum standard deviation $\geq 0$ for a bin in the *bin list*, $w_{\max}$ is the maximum sum of kernel weights for a bin in the *bin list*, $w_i$ is the sum of kernel weights in the bin for which kernel $i$ intersects, and $\sigma_i$ is the standard deviation for the bin in which kernel $i$ intersects.

We first note that the bound for linear interpolation is as follows [8,31]:

$$e(z) \leq \frac{(C_0 - C_1)^2}{8} \max_{y\in[C_0,C_1]} \left| g''(y) \right| \tag{23}$$

where $g$ is a twice differentiable function. If $f$ is a twice differentiable density function and we let $g = f$, then the above expression gives a bound on the interpolation error of the density $f$.

Since we are deriving a bound on the approximate methods' (i.e., optimized multiple query approaches) error from the exact LR-KDE, we obtain an expression of the local region-based estimator using the *normal rule* as follows:

$$\hat{f}(z) = \frac{1}{W} \sum_{i=1}^{M} \frac{w_i^{1/5}}{\sqrt{5}\sigma_i} K\left( \frac{(z - x_i) w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \tag{24}$$

The maximum value of $\hat{f}''$ is as follows:

$$\hat{f}''(z) = \frac{1}{W} \sum_{i=1}^{M} \left( \frac{w_i^{1/5}}{\sqrt{5}\sigma_i} \right)^3 K''\left( \frac{(z - x_i) w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \tag{25}$$

$$\max \hat{f}''(z) \leq \frac{1}{W} \sum_{i=1}^{M} \left( \frac{w_{\max}^{1/5}}{\sqrt{5}\sigma_{\min}} \right)^3 K''\left( \frac{(z - x_i) w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \text{ where } \sigma_{\min} > 0 \tag{26}$$

$$\leq \frac{1}{W} \sum\nolimits_{i=1}^{M} \left( \frac{w_{\max}^{1/5}}{\sqrt{5}\sigma_{\min}} \right)^3 \max\nolimits_{y \in [C_0, C_1]} \sum_{i=1}^{M} \left| K''\left( \frac{(y - x_i) w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \right| \tag{27}$$

$$= \frac{5^{2/3} w_{\max}^{3/5}}{W \sigma_{\min}^3} \max\nolimits_{y \in [C_0, C_1]} \sum_{i=1}^{M} \left| K''\left( \frac{(y - x_i)\, w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \right| \tag{28}$$

Substituting $\max \hat{f}''$ into the linear interpolation error (Eq. 23), we have the following error bound for the optimized multiple density evaluation estimators:

$$e(z) \leq \frac{5^{2/3} (C_0 - C_1)^2 w_{\max}^{3/5}}{8 W \sigma_{\min}^3} \max\nolimits_{y \in [C_0, C_1]} \sum_{i=1}^{M} \left| K''\left( \frac{(y - x_i)\, w_i^{1/5}}{\sqrt{5}\sigma_i} \right) \right|$$

### 6.3 Online maintenance cost

Let $InsertCost_{total}(K_{new})$ be the total cost of inserting $K_{new}$, then the total cost of the insertion procedure is:

$$\begin{aligned} InsertCost_{total}(K_{new}) &= Insert_{binlist}(K_{new}) + Insert_{kernellist}(K_{new}) \\ &+ Insert_{timelist}(K_{new}) \end{aligned} \tag{29}$$

*Bin list cost* : $Insert_{binlist}(K_{new})$ is composed of a sequential search over the *bin list* and merging a pair of bins. Hence, the cost of inserting $K_{new}$ into the *bin list* is

$$Insert_{binlist}(K_{new}) = O(Q) \tag{30}$$

*Kernel set cost:* Similar to the bin insertion cost, the insertion to the *kernel list* is dominated by the kernel search and the $L_2$ error updates for each pair of adjacent kernels within the affected bin. Let $R$ be the number of kernels in the updated bin, and then the total cost of the *kernel list* insertion is:

$$Insert_{kernellist}(K_{new}) = O(R)) \tag{31}$$

*Time list cost:* The dominant cost for inserting into the *time list* is the removal of all expired kernels. This operation involves updates on the $L_2$ errors and bin statistics. Let $S$ be the number of expired kernels and $T$ be the total number of kernels within a bin of an expired kernel; then the cost of inserting $K_{new}$ into the *time list* is:

$$\begin{aligned} InsertCost_{timelist}(K_{new}) &= Remove_{kernellist}(K_{new}) + Update_{bin} + UpdateL_{2_{kernellist}} \\ &= O(S) + O(Q) + O(T) = O(S) \end{aligned} \tag{32}$$

Since $Q, R, S \leq M$, where $M$ is the maximum size of the *kernel list*, the total cost of inserting $K_{new}$ is:

$$InsertCost_{total}(K_{new}) = O(M) \tag{33}$$

*Total space cost:* The total space cost of the online maintenance algorithm is derived from storing the three primary structures, *bin list, kernel list*, and *time list* in memory:

$$SpaceCost_{total} = O(Q) + O(M) + O(M) = O(M) \tag{34}$$

The above analysis shows that the time and space complexities of the proposed online kernel maintenance algorithm are $O(M)$. Since $M$ is fixed, the proposed maintenance strategy provides constant runtime and space complexities that meet the linear-pass constraint.

### 6.4 Single density evaluation cost

A single density evaluation is composed of a sequential search of the *bin list* and a scan of all kernels which provides a nonzero contribution to the query point. Let $EvalCost_{total}(D)$ be the total cost of determining the density at all the query points in $D$; then the total evaluation cost is:

$$\begin{aligned} EvalCost_{total}(D) &= \sum\nolimits_{d \in D} \left( Search_{binlist}(d) + \sum_{q \in (binlist \cap d)} Search_{kernellist}(d, q) \right) \\ &= \sum\nolimits_{d \in D} (Q + M) = |D|Q + |D|Mp_{kernellist} \end{aligned} \tag{35}$$

where $p_{kernellist}$ is the expected ratio of the kernel list that are visited for each query $d$.

Since $Q \ll M$ and $Q \perp M$, the evaluation runtime cost for a single query $d \in D$ is:

$$EvalCost_{single}(d) = Q + M = O(M) \tag{36}$$

In practical applications, only a fraction of the kernels contribute to $d$, which implies that $Search_{kernellist}(d, q) \ll M$. Therefore, the total evaluation cost can be much less than the asymptotic cost described above. The space cost of the density evaluation is $O(1)$ since the evaluation algorithm makes use of a single counter to store the current density sum. Similar to the time and space complexities of the maintenance algorithm, the evaluation runtime and space costs are bounded by $O(M)$ and hence meet the linear-pass constraint.

### 6.5 Multiple density evaluation cost

In this section, analyses of the exact multiple density query costs and the three variants of the optimized density query algorithms are provided. For the exact method, the computation of a density is composed of presorting the query set $D$, finding the set of kernel objects that can potentially provide positive contribution to at least one element of $D$, and searching through

the presorted query set. The total evaluation cost of the multiple density query algorithm for a query set $D$ is:

$$\begin{aligned} MultiEvalCost_{total}(D) &= Sort(D) + Search_{binlist}(D) \\ &\quad + \sum_{m \in (kernellist \cap D)} (Search_{queryset}(m) + Aggregate_{queryset}(m)) \\ &= |D| \log(|D|) + Q + M \log(|D|) + |D| M p_{queryset} \\ &= (|D| + M)\,(\log(|D|)) + Q + |D| M p_{queryset} \end{aligned} \tag{37}$$

where $p_{queryset}$ is the expected ratio of the query set $D$ that is visited for each kernel object.

As shown in Eq. 37, the exact multiple density query approach reduces the *bin list* search to $(|D| + M)\,(\log(|D|)) + Q$ from $|D|Q$ of the single query algorithm (see Eq. 35). Furthermore, due to the sorted query set $D$ and utilization of a logarithmic search approach, visits to noninfluenced query points are also diminished, which leads to $p_{queryset} \leq p_{kernellist}$. The combined reduction results in a total cost reduction over multiple invocations of the single density query algorithm. The space cost of the multiple density query algorithm is $O(|D|)$, which is required to store the iteratively updated densities of the query set.

In the following, the evaluation cost of the three multiple density query variants is analyzed. For all the following cost analyses, $C$ is defined to be the set of control points obtained from $D$ and $|C| \leq |D|$. The first multiple density query algorithm variant, **random-based generation (RG)** approach, has the following evaluation cost:

$$\begin{aligned} RG\_MultiEvalCost_{total}(D) &= (|C| + M)(\log(|C|)) + Q + |C| M_{queryset} + 2\,|D| - |C| \\ &= O((|C| + M)(\log(|C|)) + |D| M) \end{aligned} \tag{38}$$

The first three additive terms (i.e., $(|C| + M)\,(\log(|C|)) + Q + |C| M p_{queryset}$) correspond to Step 2 of the algorithm where the exact multiple density query algorithm is invoked to obtain the density values of $C$. The latter two terms (i.e., $2\,|D| - |C|$) refer to the cost of determining the minimum/maximum elements in $D$ and interpolating the remaining elements $D - C$. The space cost of the random-based generation approach is $O(|D|)$.

For the second multiple density query algorithm variant, **uniform-based generation (UG)**, the evaluation cost is as follows:

$$\begin{aligned} UG\_MultiEvalCost_{total}(D) &= M \log(|C|) + Q + |C|\, M p_{queryset} + 2\,|D| \\ &= O(M \log(|C|) + |D| M) \end{aligned} \tag{39}$$

The uniform-based generation approach differs from the random-based generation cost by eliminating the cost associated with query presorting ($|C| \log(|C|)$) and executing an extra $|C|$ number of interpolations. The space cost remains unchanged with $O(|D|)$.

The third multiple density query algorithm variant, **histogram-based generation (HG)**, has the following cost:

$$\begin{aligned} HG\_MultiEvalCost_{total}(D) &= M \log(|C|) + Q + |C| M p_{queryset} + 3|D| \\ &= O(M \log(|C|) + |D| M) \end{aligned} \tag{40}$$

The cost of the histogram-based generation approach imposes an additional operation over the uniform-based approach to perform the histogram estimation, resulting in the last term $3|D|$. Asymptotically, the evaluation cost of the histogram-based generation is equivalent to the uniform-based approach. The space cost is $O(|D| + H)$ where $H$ is the bucket size of the histogram.

## 7 Experiments

A set of comprehensive experiments have been conducted to validate the effectiveness and efficiency of the proposed online LR-KDE. The experiments focused on three metrics: estimation quality, maintenance time, and density evaluation time. Other existing online KDE techniques were included for performance comparisons. The experiment design and metrics are given in Sect. 7.1. The results of the estimation quality of the LR-KDE and competing techniques are provided in Sect. 7.2. The LR-KDE density evaluation and construction costs are evaluated in Sects. 7.3-7.4. Section 7.5 provides results of the sensitivity analysis. Lastly, an in-depth discussion of the implications of the experimental results to practical streaming applications is provided in Sect. 7.6.

### 7.1 Experiment design

The experiments applied a battery of synthetic and real-world data sets to study the effects of various streaming conditions. The data sets were comprised of two synthetic and four real-world time series data sets. The first 25K data samples were used for the experiments. The data set reflects a wide range of streaming scenarios which includes distributions with simple structures (e.g., unimodal) to distributions with complex and highly varied features (e.g., multi-scaled and multimodal). The data set also encompasses varying stationary properties with different periodicity (e.g., power demand load (POWER) and traffic volume (TRAFFIC)). A detailed description of the data sets is shown in Table 1:

**KDE techniques and parameters:** Table 2 provides all of the evaluated techniques and parameters:

For all of the evaluated techniques in Table 2 (except for time sample KDE), the time windows were set to be the total length of the data stream.

**Test methodology:** The first component of the experiments was to measure the estimation quality of the online KDE techniques. This was accomplished by establishing the ground truths for all data sets. In the synthetic case, the exact density structures are given in Table 1. For the real-world data sets, the true densities are defined to be the density estimates produced by the off-line AKDE. For the AKDE, the nearest-neighbor KDE was used as the pilot estimate [30]. The error measure used was the root mean square error (RMSE), which is defined as follows:

$$RMSE(\rho) = \sqrt{\frac{1}{1{,}000} \sum_{i=1}^{1{,}000} (f(x_i) - \hat{f}^{(\rho)}(x_i))^2} \tag{41}$$

where $\hat{f}^{(\rho)}(\cdot)$ is the $\rho$ density estimation technique and $x_1 \ldots x_{1{,}000}$ are query points that uniformly divide the entire span of the distribution.

The maintenance time of an online KDE is defined as the *total* amount of time required to insert and process a given set of data points. This measures the efficiency of the online KDE in updating its kernel data structures to match the current stream. The density evaluation time is defined as the average time to evaluate and compute a single density query. The density evaluation time was measured after all of the data points were processed. In these experiments, ten trials were conducted for each evaluation component and the averaged results were reported.

Multiple query processing evaluation: Experiments were also conducted on the multiple query approaches to test the impact on query efficiency and estimation quality. In particular, the cumulative query times and the relative *L1* deviations to the single query algorithm

**Table 1** Experimental data sets

| Name | Source | Sampling interval length | Type | # of Attributes | Size | Description |
|---|---|---|---|---|---|---|
| MIX2 | Synthetic | 1 s | Float | 1 | 25K | Time series randomly generated from the following mixture of normal distributions with randomly selected $\mu_i$ and $\sigma_i$ : $F_2 = \frac{1}{2}\sum_{i=1}^{2} N(\mu_i, \sigma_i^2)$ |
| MIX8 | Synthetic | 1 s | Float | 1 | 25K | Time series randomly generated from the following mixture of normal distributions with randomly selected $\mu_i$ and $\sigma_i$ : $F_8 = \frac{1}{8}\sum_{i=1}^{8} N(\mu_i, \sigma_i^2)$ |
| EEG | Real-world | 0.0078 s | Integer | 1 | 25K | EEG of a rat in wake/sleep cycle [22] |
| POWER | Real-world | 15 min | Integer | 1 | 25 K | Power demand from a Dutch research facility [22] |
| ROBOT | Real-world | 0.008 s | Float | 3 ($x$-axis selected) | 24.5 K | $x$-axis Accelerometer measurements of a Sony Aibo Robot playing soccer [22] |
| TRAFFIC | Real-world | 5 min | Integer | 1 | 25 K | Car volume readings from a loop detector near a California baseball stadium [1,4] |

were measured on various query sizes and profiles. Query profiles were generated to reflect real-world mining applications that were categorized into two profiles: UNIFORMprofile and SKEWprofile. UNIFORMprofile reflects mining tasks that perform queries on a large domain interval of equidistant points. This type of query is employed in mining algorithms such as concept drift detection and visualization [15,17,18]. SKEWprofile models queries that cover small subsets of the domain and tend to exhibit skewed distributions. This profile is modeled by a unimodal distribution with a randomly assigned center and scale. This query profile typifies mining tasks such as outlier detection and cluster analysis [20,32].

**Experimental platform:** The experiments were conducted on a Windows Server 2003 Enterprise Edition (32-bit) operating system. The hardware platform was a 2.0 GHz Intel Pentium Core Duo 2 with 3 GB of RAM.

### 7.2 Estimation quality

Figure 2 gives the estimation quality results of all data sets and KDE techniques. Each clustered bar plot represents the estimation errors for a particular data set where the *x-axis* is the number of processed sample points and the *y-axis* is the RMSE of the density estimates. The experiment showed that the LR-KDE (both single and multiple query approaches) pro-

**Table 2** Evaluated online KDE techniques

| Name | Technique | Parameter |
|---|---|---|
| *Sequence sample KDE* | Sequence sample-based KDE [32,34] | Max. # of kernels = 1,000 |
| *Time sample KDE* | Time sample-based KDE using the priority-sample algorithm [6] | Max. # of kernels = 1,000 |
| *M-Kernel KDE* | Variable bandwidth cluster KDE [38] | Max. # of kernels = 1,000<br>Simplex max. iter. = 5,000 |
| *Heinz KDE* | List-based cluster KDE [19] | Max. # of kernels = 1,000 |
| *LR-KDE* | Single query Local Region KDE | Max. # of kernels = 1,000, $\mu = 1, 4 \leq Q \leq 8$ |
| *Multi-LR-KDE* | Multiple query approach that gives identical estimates to the single query LR-KDE | Same as LR-KDE |
| *Optimized Multi-LR-KDE (SAMPLE)* | Multiple query approach which employs a sampling based methodology | Same as LR-KDE, $s = .4$ |
| *Optimized Multi-LR-KDE (UNIFORM)* | Multiple query approach which employs a uniform gridding methodology | Same as LR-KDE, $s = .4$ |
| *Optimized Multi-LR-KDE (HISTOGRAM)* | Multiple query approach which employs a histogram methodology | Same as LR-KDE, $s = .4$ |

vided lower RMSE than all competing techniques in the MIX2, MIX8, ROBOT, TRAFFIC, POWER, and EEG data sets. The LR-based approaches produced significant RMSE reductions in MIX2 and MIX8 with errors that were at most half of the next best-performing technique. Note that the time sample and sequence sample KDEs provided almost identical performance in MIX2 and MIX8.

Both the single and multiple query LR-KDE converged as more samples were processed in the MIX2, MIX8, POWER, and EEG data sets. However, for TRAFFIC and ROBOT, the RMSE of LR-KDE increased at 20K and 24.5K points, respectively. This behavior was similarly exhibited in other techniques such as M-kernel KDE. In the TRAFFIC data set, the sequence sample KDE produced a drastic change in its RMSE at the 20K mark. All of these observations suggest the presence of concept drifts in the POWER and EEG data sets.

Figures 3 and 4 show the plotted estimates of MIX2 and MIX8 by two of the lowest error attaining techniques, LR-KDE and Heinz KDE. The *x-axis* represents the query points, and the *y-axis* shows the density. MIX2 and MIX8 possess multiple isolated modes that can be difficult to estimate with a single bandwidth KDE. For example, the Heinz KDE tended to oversmooth the distributions as indicated by the underestimated peaks and overestimated troughs. The oversmoothing can be attributed to Heinz KDE's use of the *normal rule* bandwidth which is known to oversmooth multimodal distributions [30]. In an attempt to improve the accuracy of Heinz KDE, the only available parameter, kernel size, was increased from 1 to 25 K (treats all available data samples as kernels). The increased kernel size produced $\leq 1\%$ improvement in the RMSE and showed no observable difference in the plotted estimates. When the kernel size for LR-KDE was increased to 25 K, it resulted in 5.5 % (MIX2) and 7.8 % (MIX8) improvements in the RMSE. Although the LR-KDE employs the *normal rule* bandwidth, it restricts uniform bandwidth assignment to regions of similar densities. As a result, the LR-KDE identified all of the modes and accurately captured the peaks and valleys.

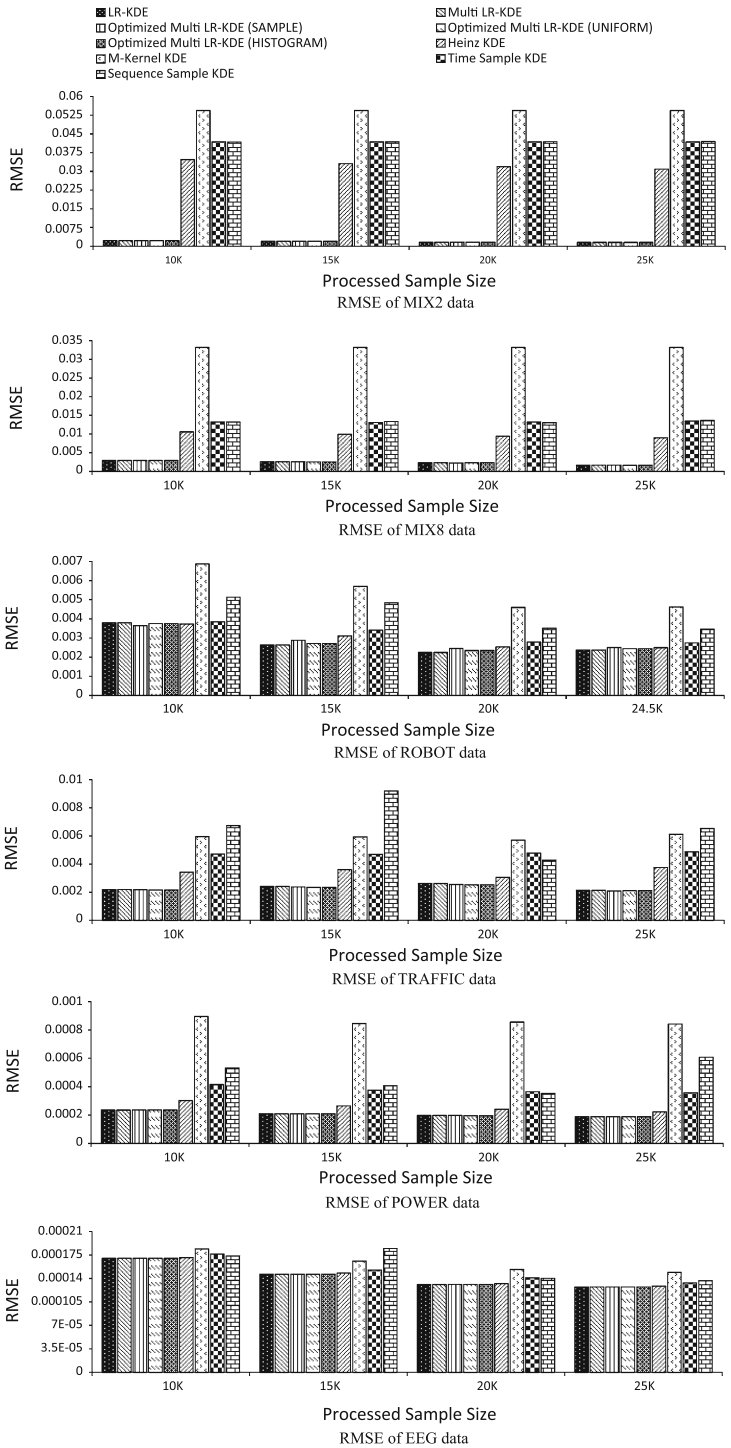



**Fig. 2** Estimation quality results

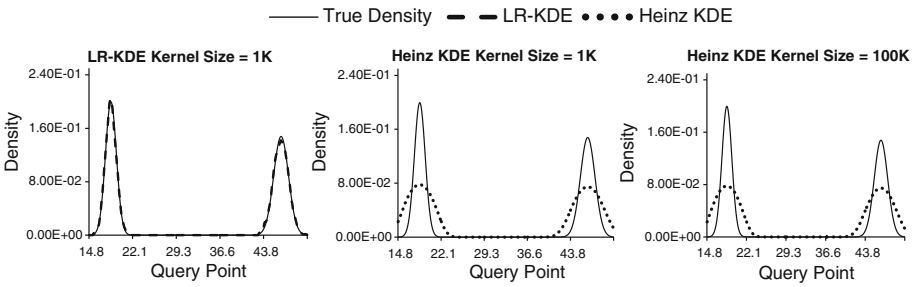



**Fig. 3** Plots of estimated densities by LR-KDE and Heinz KDE for MIX2

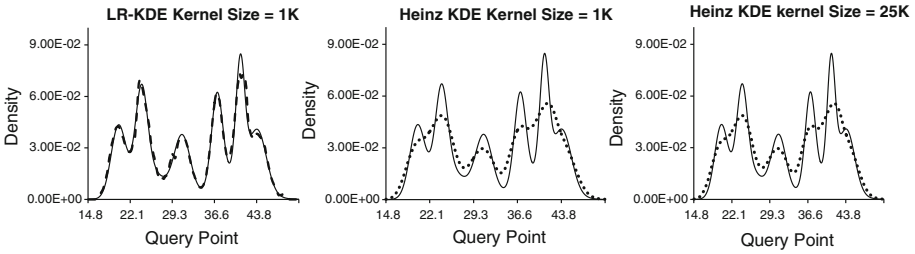



**Fig. 4** Plots of estimated densities by LR-KDE and Heinz KDE for MIX8

### 7.3 Single query processing

The following section shows the runtime results of updating the kernel objects and the evaluation times of the single query approaches.

**Maintenance time:** Figure 5 illustrates the impact of the various data streams on LR-KDE's kernel maintenance times. The *x-axis* indicates the data sets, and the *y-axis* measures the maintenance times for processing the entire data. The maintenance times of the multiple queries LR-KDE are identical to the single query approach since the only difference between the two methods is in their query processing approaches. The real-world POWER, TRAFFIC, and ROBOT data sets showed the most improvements for the LR-KDE. This observation is attributable to the data sets' largely ordered samples that reduced the amount of scans the LR-KDE needed to perform. In the MIX8 and EEG data sets, the LR-KDE provided lower or comparable times than all of the non-sample-based approaches. Note that the sample-based techniques produced nearly identical results within the various data sets. The similar maintenance times of the sample-based methods are expected since their required times to select and replace kernels are similar and exclusively depend on the kernel set size. However, the same was not observed on their estimation quality as shown in Fig. 2.

Figure 6 shows the relationship between maintenance time and sample size. The *x-axis* is the number of samples processed, and the *y-axis* is the maintenance time. The POWER data set is shown, but similar trends can be observed in the other data sets. All of the KDE techniques exhibited times that were linear to the sample size; however, LR-KDE and Heinz KDE provided the lowest cost rates in POWER, TRAFFIC, and ROBOT. The linear trend of LR-KDE is also consistent with the analyses in Sect. 6. Standard deviation for all trials was $\leq 5\,\%$.

**Density Evaluation Time:** Figure 7 shows the density query times of all data sets. The *x-axis* represents all of the data sets, and the *y-axis* gives the average evaluation time for a

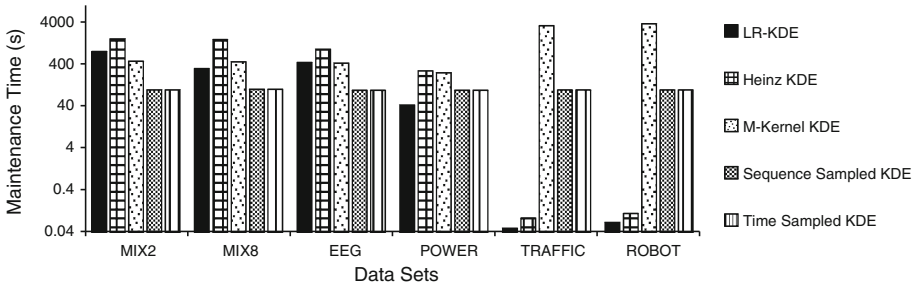



**Fig. 5** Log-scaled maintenance time of all data sets

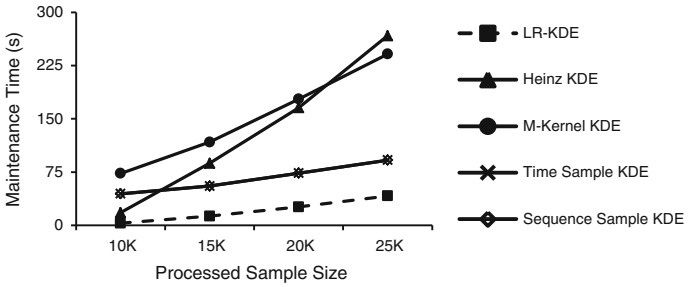



**Fig. 6** Maintenance times of POWER

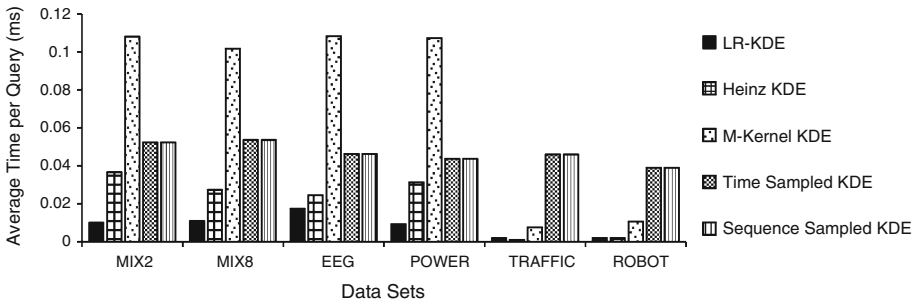



**Fig. 7** Average density evaluation time for a single query

single density query. For MIX2, MIX8, EEG, and POWER data sets, the LR-KDE consistently produced lower evaluation times than all of the competing techniques. For TRAFFIC and ROBOT, the LR-KDE performed equally to Heinz KDE but better than the other techniques. For the sample-based KDE, regardless of the kernel function employed, the entire kernel set must be scanned to generate a density estimate. This approach resulted in a consistent, but higher evaluation times than the LR-KDE and Heinz KDE. In summary, the results demonstrated that the LR-KDE evaluation performance was better than or at least equal to all of the competing techniques. Standard deviation for all trials was $\leq$10%.

### 7.4 Multiple query processing

This section provides an in-depth evaluation of the multiple query approaches for the LR-KDE. Query runtimes were measured for all data sets under varying query profiles and sizes.

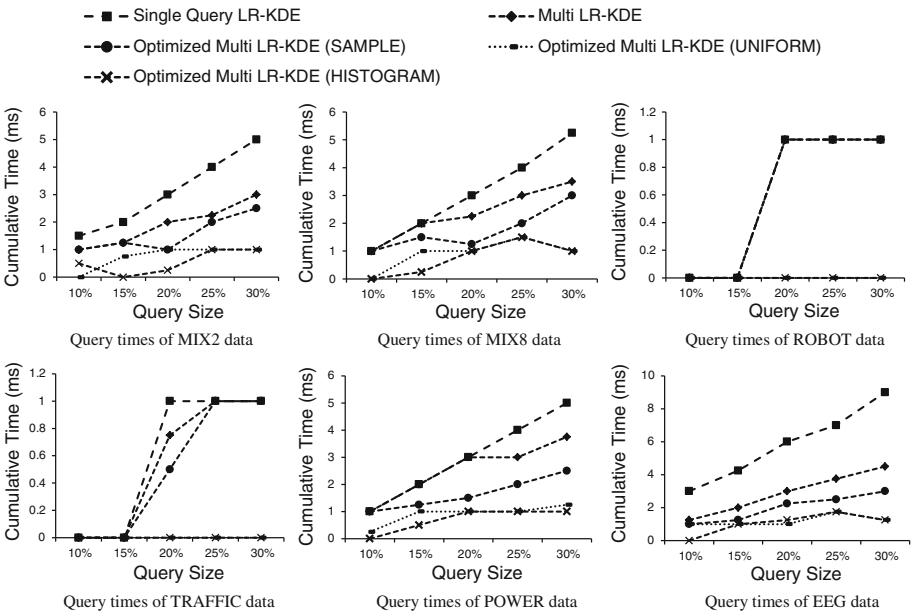



**Fig. 8** Query times for UNIFORMprofile

The relative deviations with respect to the single query approach were also measured for the optimized methods.

**Density evaluation time:** The query times of the single and multiple query LR-KDEs are provided in Figs. 8 and 9. The *x-axis* represents the query size ratio with respect to the kernel set size, and the *y-axis* gives the average cumulative times. For all data sets and query profiles (i.e., UNIFORMprofile and SKEWprofile), the multiple query approaches consistently matched or outperformed the single query method. The improvements in throughput for the multiple query approaches are due to the reduction in the bin list scans and kernel list computations. However, a small overhead is incurred to sort the query set. Assuming that the query set is much smaller than the kernel list, the empirical time reduction rates achieved (Figs. 8 and 9) are approximate upper bounds on the kernel computation pruning rates. The optimized techniques provided even lower cumulative query times than the nonoptimized multiple query approach. However, between the optimized techniques, the UNIFORM and HISTOGRAM techniques gave faster running times than the SAMPLE-based approach due to the elimination of query presorting and linear time control point construction. In the ROBOT and TRAFFIC data sets, the single query LR-KDE gave very low query times (see Fig. 7), which resulted in even faster times (near 0) by the multiple query approaches. Overall, the exact multiple query approach was effective in significantly lowering the evaluation of the single query approach for all the data sets and query profiles. These runtimes were further reduced by the optimized techniques which produced dramatic improvements over the single query approach.

**Relative deviation:** Figures 10 and 11 show the mean relative *L1* deviation ratio to the single query algorithm for all query profiles. The results of the exact multiple query approach are not given since they generate identical estimates to the single query algorithm. In Fig. 10 (UNIFORMprofile), all of the estimates produced less than 9 % deviation, while the HISTOGRAM and UNIFORM techniques both gave less than 7.5 % deviation. The

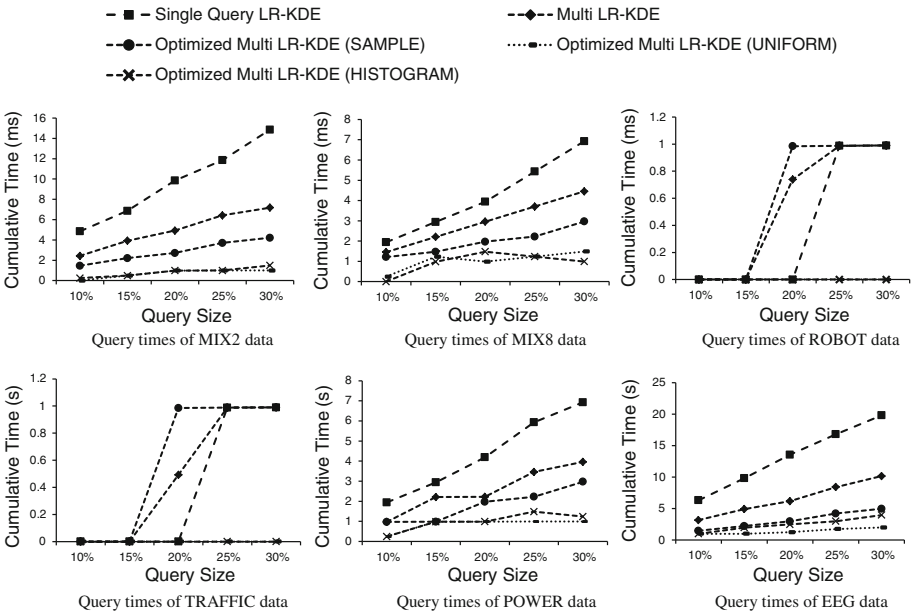



**Fig. 9** Query times for SKEWprofile

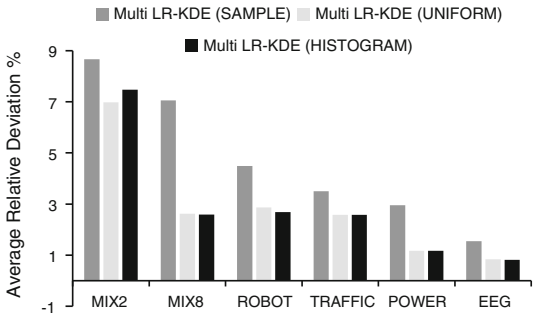



**Fig. 10** Relative deviation from single query algorithm for UNIFORMprofile

HISTOGRAM- and UNIFORM-based variants consistently produced lower deviations than the SAMPLE-based technique due to its ability to model sparsely populated query points. The SAMPLE-based technique requires higher sampling rate to produce more accurate control points. However, for the SKEWprofile, the SAMPLE- and HISTOGRAM-based techniques are able to obtain significantly lower deviation than the UNIFORM-based approach. In this scenario, the uniform assumption of the query distribution is not met, which results in higher deviation for the UNIFORM approach. Because the HISTOGRAM- and SAMPLE-based techniques do not impose such a stringent assumption on the queries' distributional structure, these techniques produce significantly lower deviations.

### 7.5 Sensitivity analysis

In this section, we describe a study on the effects of LR-KDE's estimation quality for various parameter values of bin size $Q$, kernel weight variance factor $\mu$, and control points size ratio

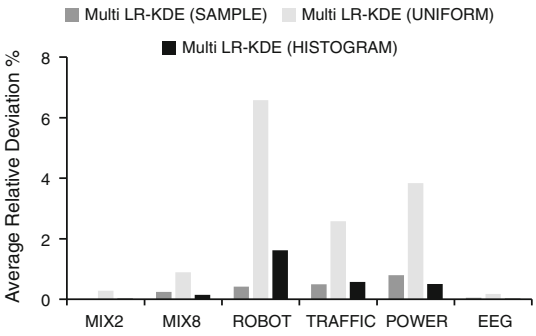



**Fig. 11** Relative deviation from single query algorithm for SKEWprofile

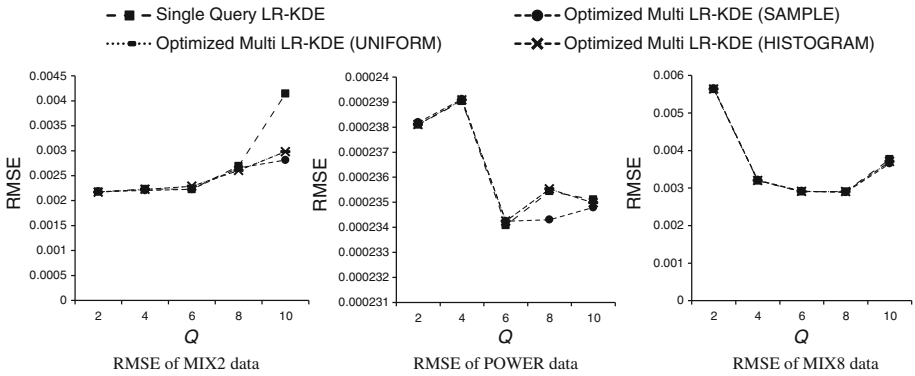



**Fig. 12** RMSE of varying bin size ($Q$)

$s$. The data sets used for this study are MIX2, POWER, and MIX8. These data sets represent a spectrum of simple to complex densities. The quality measure used for varying $Q$ and $\mu$ is the RMSE. The relative $L1$ deviation score is used to test the effect of differing $s$ values for the multiple query approaches. To apply the LR-KDE in practice, the parameters $Q$, $\mu$, and $s$ are estimated by checking the LR-KDE quality measures for various values, and the settings with the highest quality measures will be returned as the final parameter values.

**Effect of varying bin size** ($Q$): Figure 12 shows the estimation quality results of the LR-KDE approaches for a range of bin sizes $Q$. The *x-axis* represents $Q$, and the *y-axis* gives the RMSE of the estimates. The results of the exact multiple query LR-KDE are not shown since they produced identical estimates to the single query LR-KDE. In MIX2, it is observed that the RMSE increased slightly with the increase in $Q$. This is an expected behavior since the structure of MIX2 is relatively simple (i.e., MIX2 has two modes) and a higher $Q$ increases the chance of overfitting. Within that data, the multi-query methods obtained lower errors than the single query algorithm at $Q = 10$, and it appeared that the selected control points and interpolations were able to reduce some of the variability. However, in general this is not the case since the multiple query SAMPLE-based method can give volatile estimates when $s$ is low. For the MIX8 and POWER data sets, the estimation quality improved as $Q$ is increased but degraded slightly at the higher range $Q \geq 8$. However, the impact of varying $Q$ with respect to the estimation quality obtained by the competing methods is dampened by the fact that the errors produced by the other methods are still higher than the LR-KDE techniques (see Fig. 2) across the entire range of $Q$.

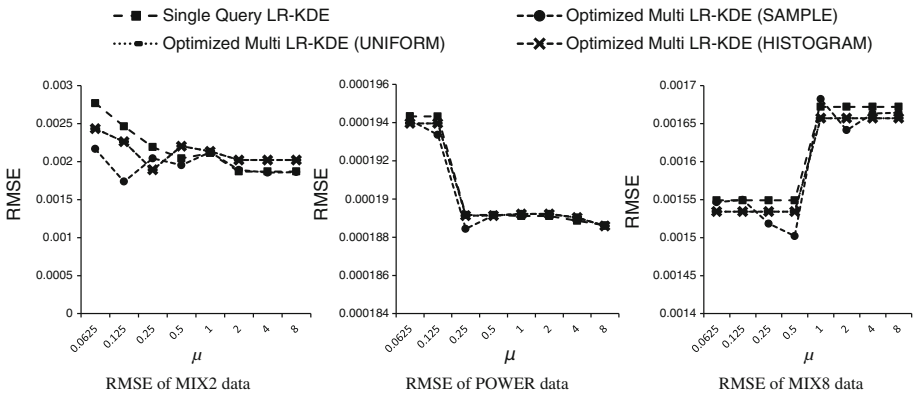



**Fig. 13** RMSE of varying kernel weight variance factor ($\mu$)

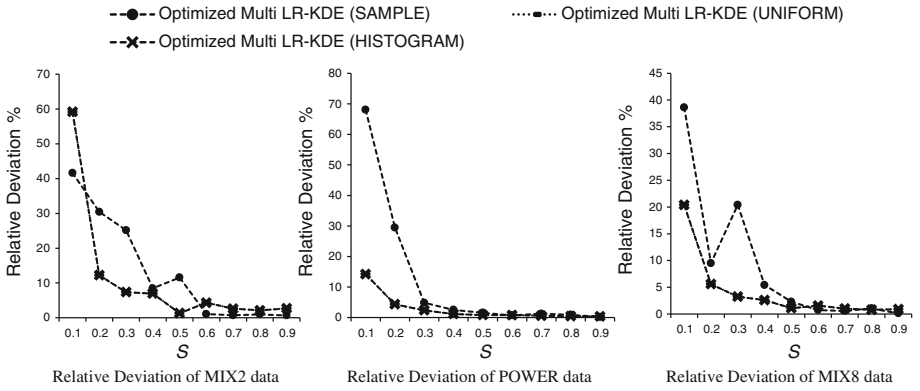



**Fig. 14** Relative *L1* deviation percentage of varying control point size ratio ($s$) for UNIFORMprofile

**Effect of varying kernel weight variance factor** ($\mu$): Figure 13 provides the estimation quality results for varying values of $\mu$. The *x-axis* represents $\mu$, and the *y-axis* gives the RMSE of the estimates. The results of the exact multiple query LR-KDE are not shown since they produced identical estimates to the single query LR-KDE. In the MIX8 data set, low values of $\mu \leq 0.5$ provided higher estimation quality. For this data set, the PAD criterion played a more crucial role than kernel weight variance in generating appropriate local regions, except the POWER data set where the kernel weight variance criterion indicated a stronger influence on the estimation quality. These two scenarios show the importance of including both the PAD and kernel weight variance criterions into the local region generation. The MIX2 data set showed a similar pattern to POWER; however, its effect from increasing the contribution of the kernel weight variance is not as pronounced as POWER since the error values remain largely unchanged throughout the entire $\mu$ range.

**Effect of varying control points size ratio** ($s$): Figures 14 and 15 provide the relative *L1* deviation percentage (w.r.t. exact method) of the approximate multiple query approaches across a range of $s$ values. The *x-axis* represents $s$, and the *y-axis* gives the relative L1 deviation percentage. In the UNIFORMprofile query set (Fig. 14), the deviation decreased drastically from 0.1 to 0.4 and continued to decrease (at a slower rate) as $s$ increased. The SAMPLE-based approach showed more variability as $s$ decreased since it does not have suf-

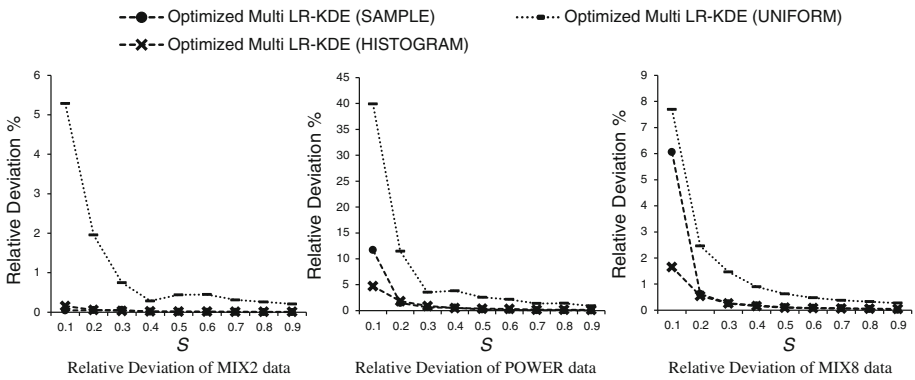



**Fig. 15** Relative *L1* deviation percentage of varying control point size ratio ($s$) for SKEWprofile

ficient control points to accurately estimate the query distribution. Both the HISTOGRAM and UNIFORM methods performed identically well since they both can capture the uniform query distribution. In Fig. 15 of the SKEWprofile query set, the HISTOGRAM- and SAMPLE-based approaches exhibited lower deviation to the UNIFORM approach. In both the UNIFORMprofile and SKEWprofile for all data sets, the HISTOGRAM approach provided the lowest deviations.

### 7.6 Discussion

Application of the local region concept to kernel density estimates has shown to be effective in modeling the local density features in data stream. As a result, the LR-KDE provided superior estimation quality over the competing techniques in both real-world and synthetic data sets. The LR-KDE was also able to improve the estimation accuracy in data sets that did not exhibit strong localities (e.g., predominantly unimodal data sets such as ROBOT). Since real-world data streams can exhibit strong local features (e.g., clustered outliers), it can be expected that stream mining applications would benefit from the use of the LR-KDE.

LR-KDE also improved the computational efficiency over the competing techniques. Local regions allow nonrelevant kernels to be pruned from further processing, which results in the simultaneous reduction in maintenance and query times. Furthermore, the LR-KDE retained the same order of space cost as the other online KDE techniques. The multiple query algorithms for LR-KDE can further improve the throughput of density estimates for mining tasks that generate multiple point queries. These cost reduction techniques are essential to stream mining tasks where results need to be furnished in real time and processed in a fixed-size memory environment.

The space/time complexity of LR-KDE shown in Sect. 6 (Eqs. 33, 34) applies to massive data sets. The space/time cost of updating the LR-KDE with a new data element is $O(M)$ where $M$ is the number of maintained kernels and $M$ remains constant throughout its execution. Hence, the total cost of updating the LR-KDE for an arbitrary data size $N$ is $O(NM)$, which is linear to the data set size. The experimental result of Fig. 6 shows this linear trend in the update cost of the LR-KDE. The figure also shows that LR-KDE achieved faster times than all the other techniques. Density evaluation on the LR-KDE is only invoked on the maintained kernels and therefore does not depend on the data size. The cost to perform density evaluation is $O(M)$ (see cost analysis in Sects. 6.3, 6.4). In addition, Fig. 7 shows that LR-KDE provides superior or comparable performance to all the other stream-based methods. Because LR-KDE is guaranteed to expend at most $O(M)$ space/time for an update

and density evaluation and the empirical results showed support of this property, the LR-KDE is a viable and efficient approach for processing massive data streams.

A concrete mining task that would benefit from LR-KDE's improved estimation quality and throughput is density-based clustering, in particular those density-based approaches that employ bump hunting algorithms for determining the cluster centers [30]. For example, Figs. 3 and 4 show that the LR-KDE captures and differentiates all of the modes almost exactly, which would allow the bump hunting method to optimally isolate the cluster centers. In contrast, the next best-performing technique (Heinz KDE) could not capture some of these modes that would increase the likelihood for the bump hunting algorithm to dismiss some potentially vital clusters. Such false dismissals can lead to missed emergent events and potential disastrous results. Within the context of a real-time environment, minimizing the time required to discover emergent events is essential to the overall success of the mining algorithm.

Density estimation can also be applied to the problem of outlier detection. An outlier can be defined as a sample point whose probability of occurrence falls below a predefined threshold [32]. Suppose that density estimates are employed to perform the above outlier detection scheme, then the rates of false dismissals are dependent on the accuracy of the estimated model. For instance, if the estimate is oversmoothed, then the rate of false dismissals may be increased in regions of low probability. In these regions, the true density of the sample points may be much smaller than their estimated values. Hence, a more accurate density estimation model, such as LR-KDE, can help reduce the false dismissals and improve the outlier detection performance.

## 8 Conclusion

This paper addresses the issue of developing an efficient and asymptotically consistent online adaptive density estimation technique to meet the stringent constraints of the data stream environment. In that endeavor, we propose an online- and local region-based AKDE framework (LR-KDE) for univariate streams. The contributions of this work include the first KDE approach that supports adaptive bandwidth and efficient multiple query processing over data streams with approximation guarantees, the development of the pairwise distance criterion that effectively approximates the AKDE and guarantees asymptotic consistency, the design of linear-pass algorithms to maintain and compute kernel density estimates over a time-based sliding window, and the construction of a set of efficient algorithms to accurately estimate multiple density queries. Theoretical analyses are provided to validate the asymptotic consistency and computational complexities of the LR-KDE. Experiments demonstrated that the LR-KDE enhanced estimation quality, improved maintenance performance, and reduced density evaluation time over the existing techniques. The experiments also showed that LR-KDE can improve the effectiveness of real-time stream mining tasks such as clustering and outlier detection.

## References

1. Freeway performance measurement system (PeMS) [http://pems.eecs.berkeley.edu]
2. Aggarwal C (2003) A framework for diagnosing changes in evolving data streams. In: Proceedings of 2003 ACM SIGMOD international conference on management of data. San Diego, CA, pp 575–586
3. Aggarwal C, Yu PS (2007) Data streams: models and algorithms. In: Aggarwal C (ed) A survey of synopsis construction in data streams. Springer Science and Business Media, New York, pp 69–202

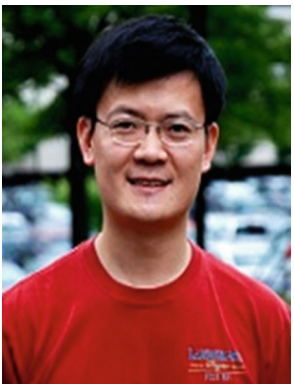
4. Asuncion A, Newman DJ (2007) UCI machine learning repository. [http://www.ics.uci.edu/~mlearn/MLRepository.html]
5. Babcock B, Babu S, Datar M, Motwani R, Widom J (2002) Models and issues in data stream systems. In: Proceedings of 21st ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems. Madison, WI, pp 1–16
6. Babcock B, Datar M, Motwani R (2002) Sampling from a moving window over streaming data. In: Proceedings of 13th Annual ACM-SIAM symposium on discrete algorithms. San Francisco, CA, pp 633–634
7. Chan CC, Batur C, Srinivasan A (1991) Determination of quantization intervals in rule based model for dynamic systems. In: Proceedings of IEEE conference of systems, man, and, cybernetics. pp 1719–1723
8. Clear R, Berman S (1988) Estimation of linear interpolation error. In: Proceedings of the annual illuminating engineering society conference
9. Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. In: Proceedings of 12th international conference on machine learning. pp 194–202
10. Duoandikoetxea J (2001) Fourier analysis: American mathematical society
11. Gibbons P, Matias Y, Poosala V (2002) Fast incremental maintenance of approximate histograms. ACM Trans Database Syst 27:261–298
12. Gilbert A, Kotidis Y, Muthukrishan S, Strauss MJ (2002) How to summarize the universe: dynamic maintenance of quantiles. In: Proceedings of the 28th international conference of very large data bases. Hong Kong, China, pp 454–465
13. Gray A, Moore A (2003) Rapid evaluation of multiple density models. In: Proceedings of 9th international workshop on artificial intelligence and statistics. Key West, FL
14. Guha S, Koudas N, Shim K (2006) Approximation and streaming algorithms for histogram construction problems. ACM Trans Database Syst 31:396–438
15. Heinz C (2007) Density estimation over data streams. Phd, Mathematics, Phillipps-University Marburg
16. Heinz C, Seeger B (2008) Cluster kernels: resource-aware kernel density estimators over streaming data. IEEE Trans Knowl Data Eng 20:880–893
17. Heinz C, Seeger B (2006) Exploring data streams with nonparametric estimators. In: Proceedings of 18th international conference on statistical and scientific database management. Vienna, Austria, pp 261–264
18. Heinz C, Seeger B (2006) Resource-aware kernel density estimators over streaming data. In: Proceedings of 15th ACM international conference on information and knowledge management. Arlington, VA, pp 870–871
19. Heinz C, Seeger B (2006) Towards kernel density estimation over streaming data. In: Proceedings of 13th international conference on management of data. Delhi, pp 91–102
20. Hinneburg A, Keim D (1998) An efficient approach to clustering in large multimedia databases with noise, in proceedings of ACM Knowledge Discovery and Data Mining 58–65
21. Ioannidis Y (2003) The history of histograms (abridged). In: Proceedings of 29th international conference on very large databases. Berlin, pp 19–30
22. Keogh E, Xi X, Wei L, Ratanamahatana CA (2008) The UCR time series classification/clustering. [http://www.cs.ucr.edu/~eamonn/time_series_data]. Available: http://www.cs.ucr.edu/~eamonn/
23. Ledl T (2004) Kernel density estimation: theory and application in discriminant analysis. Aust J Stat 33:267–279
24. Liu H, Hussain F, Tan CL, Dash M (2002) Discretization: an enabling technique. Data Min Knowl Discov 6:393–423
25. Merckt TV (1993) Decision trees in numerical attribute spaces. In: Proceedings of the 13th international joint conference on artificial intelligence, pp 1016–1021
26. Nussbaumer HJ (1982) Fast Fourier transform and convolution algorithms, 2nd edn. Springer, New York
27. Parzen E (1962) On estimation of a probability density function and mode. Ann Math Stat 33:1065–1076
28. Sain SR, Scott DW (1996) On locally adaptive density estimation. J Am Stat Assoc 91:1525–1534
29. Scott DW (1992) Multivariate density estimation. Wiley, New York
30. Silverman BW (1986) Density estimation for statistics and data analysis. Chapman and Hall, London
31. Smith JO (2011) Digital audio resampling home page. Available http://www-ccrma.stanford.edu/~jos/resample
32. Subramaniam S, Palpanas T, Papadopoulos D, Kalogeraki V, Gunopulos D (2006) Online outlier detection in sensor data using non-parametric models. In: Proceedings of the 32nd international conference on very large databases. Seoul, pp 187–198
33. Wand MP, Jones MC (1995) Kernel smoothing. CRC Press, Boca Raton
34. Wegman EJ, Marchette DJ (2003) On some techniques for streaming data: a case study of internet packet headers. J Comput Graph Stat 12:1–22

35. Weiss SM, Galen RS, Tadepalli PV (1991) Maximizing the predictive value of production rules, artificial intelligence, pp 47–71
36. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Proceedings of ACM SIGMOD international conference on management of data. Montreal, pp 103–114
37. Zhang T, Ramakrishnan R, Livny M (1999) Fast density estimation using CF-kernel for very large databases. In: Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining. San Diego, CA, pp 312–316
38. Zhou A, Cai Z, Wei L, Qian W (2003) M-Kernel merging: towards density estimation over data streams. In: Proceedings of the 8th international conference on database systems for advanced applications. Kyoto, pp 285–292

## Author Biographies

**Arnold P. Boedihardjo** received his BS degree in Mathematics and Computer Science from Virginia Tech in 2001. He received his MS and Ph.D. degrees in Computer Science from Virginia Tech in 2006 and 2010, respectively. He has published in various scholarly venus such as IEEE International Conference on Data Engineering, IEEE International Conference on Data Mining, ACM Conference on Information and Knowledge Management, ACM International Symposium on Advances in Geographic Information Systems, and IET Communications Journal. His research interests include data stream systems, spatial database, information retrieval, optimizations, networking, and statistical learning. He is currently a research scientist at the U.S. Army Engineer Research and Development Center.

**Chang-Tien Lu** received the M.S. degree in computer science from the Georgia Institute of Technology in 1996 and the Ph.D. degree in computer science from the University of Minnesota in 2001. He is an associate professor in the Department of Computer Science, Virginia Polytechnic Institute and State University, and is founding director of the Spatial Lab. He served as Program Co-Chair of the 18th IEEE International Conference on Tools with Artificial Intelligence in 2006, and General Co-Chair of the 20th IEEE International Conference on Tools with Artificial Intelligence in 2008 and 17th ACM International Conference on Advances in Geographic Information Systems in 2009. He is also serving as Vice Chair of the ACM Special Interest Group on Spatial Information (ACM SIGSPATIAL). His research interests include spatial databases, data mining, geographic information systems, and intelligent transportation systems.



**Feng Chen** is a postdoctoral research fellow at Carnegie Mellon University. He received his B.S. from Hunan University, China, in 2001, M.S. degree from Beihang University, China, in 2004, and Ph.D. degree from Virginia Polytechnic Institute and State University in 2012, all in Computer Science. He has published 25 refereed articles in major data mining venues, including ACM-SIGKDD, ACM-CIKM, ACM-GIS, IEEE-ICDM, and IEEE-INFOCOM. He holds two U.S. patents on human activity analysis filed by IBM's T.J. Watson Research Center. His research interests are in the areas of statistical machine learning and data mining, with an emphasis on spatiotemporal analysis, social media analysis, and energy disaggregation.