KDE-Track: An Efficient Dynamic Density Estimator for Data Streams

Abdulhakim Qahtan, Suojin Wang and Xiangliang Zhang

Abstract—Recent developments in sensors, global positioning system devices and smart phones have increased the availability of spatiotemporal data streams. Developing models for mining such streams is challenged by the huge amount of data that cannot be stored in the memory, the high arrival speed and the dynamic changes in the data distribution. Density estimation is an important technique in stream mining for a wide variety of applications. The construction of kernel density estimators is well studied and documented. However, existing techniques are either expensive or inaccurate and unable to capture the changes in the data distribution. In this paper, we present a method called KDE-Track to estimate the density of spatiotemporal data streams. KDE-Track can efficiently estimate the density function with linear time complexity using interpolation on a kernel model, which is incrementally updated upon the arrival of new samples from the stream. We also propose an accurate and efficient method for selecting the bandwidth value for the kernel density estimator, which increases its accuracy significantly. Both theoretical analysis and experimental validation show that KDE-Track outperforms a set of baseline methods on the estimation accuracy and computing time of complex density structures in data streams.

Index Terms—Adaptive Resampling, Bandwidth Selection, Data Streams, Dynamic Density Estimation, Interpolation

1 INTRODUCTION

Recent advances in computing technology allow for collecting vast amount of data that arrive continuously in data streams. Examples of data streams can be found in fields such as sensor networks, mobile data collection platform, and network traffic. The data need to be processed and analyzed once they arrive. However, the unbounded, rapid and continuous arrival of data streams disallow the usage of traditional data mining techniques. Therefore, the development of algorithms for processing data streams instantaneously becomes highly important.

Density estimation has been widely used in various applications. Estimating the Probability Density Function (PDF) for a given data set provides knowledge about the underlying distribution of the data. Consequently, dense regions can be recognized as clusters and quantities such as medians and centers of clusters can be computed [1]. By contrast, sparse regions are reported as outliers that can be used for fault detection, e.g., in sensor networks [2].

This paper aims to estimate the dynamic density that comes with the evolving spatiotemporal data streams, e.g., traffic streams in a city. In the year of 2013, more than 170 million taxi trips were recorded in the city of New York. Monitoring and visualizing the density of spatiotemporal streams will help on placing taxicabs [3], reducing ambulance emergencies response time [4] and reflecting people’s interest at a particular location for specific seasons [5].

However, estimating the dynamic density that comes with evolving spatiotemporal streams is a challenging task. Besides the problem of estimating the density using samples drawn from an unknown distribution in case of stationary data, spatiotemporal data streams have more challenging properties that complicate the estimation of density. First, the data distribution changes dynamically in an unpredictable fashion. Therefore, density estimation should rely more on the recently received data samples [6], [7], e.g., by using a sliding window. Second, an anytime-available model should be efficiently updated to allow real-time monitoring of the density. Meanwhile, the density function value of any new arriving data may need to be instantly estimated. Third, the spatial non-uniformity of data distribution requires higher resolutions in dense areas and lower resolutions in sparse areas, so that the estimation is accurate to catch the details.

Most of the existing approaches for estimating the density of data streams are based on the Kernel Density Estimation (KDE) method due to its advantages for estimating the true density [8]. Given a set of samples, \( S = \{x_1, x_2, \ldots, x_n\} \), where \( x_j \in \mathbb{R}^d \), KDE estimates the density at a point \( x \) as:

\[
\hat{f}(x) = \frac{1}{n} \sum_{j=1}^{n} K_h(x, x_j),
\]

where \( K_h(x, x_j) \) is a kernel function, which is usually
a radially symmetric unimodal function that integrates to 1. Eq. 1 shows that KDE uses all the data samples to estimate the PDF of any given point. In the problem of online density estimation of data stream, i.e. estimating the density of every arriving data sample, KDE has quadratic time complexity with respect to (w.r.t.) the stream size. Also, the space requirement for KDE significantly increases with the dataset size.

In order to reduce the high computational and space costs of KDE, the number of used samples was controlled to be a small value, by different ways; 1) merging kernel\(^3\) in 1, 6, 7, 2 where each merged kernel summarizes a set of similar samples; 2) sampling 10, 11 where only selected kernels are used in the calculation of \(f(x)\). Table 1 summarizes the key characteristics of the most popular density estimators. Using merging or sampling reduces the computational cost of KDE, however, sacrificing the estimation accuracy due to the reduction of used kernels. More discussions of the limitations of these approaches will be presented in Section 2.

In this paper, we propose a method called KDE-Track to model the data distribution as a set of resampling points with their estimated PDF. To guarantee the estimation accuracy and to lighten the load on the model, an adaptive resampling strategy is employed to control the number of resampling points, i.e., more points are resampled in the areas where the PDF has a larger curvature, while less number of points are resampled in the areas where the function is approximately linear. In order to overcome the quadratic time complexity of KDE when evaluating the PDF for each new observation, linear interpolation is used with KDE for online density estimation. This technique was used in 12 with univariate stationary data and known as Kernel Polygons. The new interpolation KDE computes the PDF value of a new arriving data sample using interpolation of selected resampling points. It therefore has advantages of evaluating the PDF for any new observation in linear time complexity and space complexity w.r.t. the number of resampling points. Evaluating the PDF for all received observations will then take linear time compared with the quadratic time complexity of KDE. To timely track the evolving density, we use a sliding window strategy in KDE-Track to estimate the density using the most recent data samples.

Our KDE-Track has unique properties as follows: (1) it generates density functions that are available to visualize the dynamic density of data streams at any time. At any time \(t\), after receiving one streaming data sample \(x_t\), KDE-Track updates the PDF of the data stream and also estimate \(f(x_t)\). As a real-world application example, KDE-Track is employed to visualize the density of pickup events in New York Taxi data in an online fashion and helps on finding interesting patterns. (2) it has linear time and space complexities w.r.t. the model size for maintaining the dynamic PDF of data stream upon the arrival of every new sample. It is thus 8—85 times faster than the traditional KDE depending on the window size; (3) the estimation accuracy is guaranteed by adaptive resampling and optimized bandwidth (\(h\)), which also address the spatial non-uniformity issue of spatiotemporal data streams. Comparing with a set of baseline methods, it achieves the lowest estimation error, especially when the density function is multimodal and complex.

Both theoretical analysis and experimental results on synthetic and real-world data show the effectiveness of our approach for estimating the dynamic density functions that come with spatiotemporal data streams.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 discusses KDE and its related issues. Section 4 introduces our approach KDE-Track. Section 5 discusses the implementation details of KDE-Track. Section 6 presents the evaluation results of our approach and Section 7 concludes our work.

2 Related Work

This section discusses the work that is related to our study and also focuses on estimating the dynamic density of data streams. We should consider not only the constraints of using limited memory and processing the data in real time (16, 17) due to the nature of streams, but also the dynamic changes of the underlying density function over time.

To reduce the computational cost and space requirement of KDE, methods have been proposed based on kernel merging, sampling or space partitioning. Kernel merging is used in 1, 7 and 6 where a specific number of kernels are maintained through merging two or more kernels. Each kernel summarizes a cluster of similar samples. A new arriving sample can either fall into one existing kernel or trigger a new kernel. Two kernels are merged if the number of kernels exceeds the specified number. Since merging kernels is a lossy approximation, cost functions were proposed to decide which kernels to be merged by measuring the amount of discarded information during the merging process. The three methods differ in the way of selecting the bandwidth value. Cluster Kernels (CK) uses global bandwidth value for all kernels, M-Kernel uses different bandwidth for each kernel point and AKDE defines a set of local regions with minimum intra-density variations and uses different bandwidth value for each region. When estimating the PDF for one sample, all the methods require to resample from kernels. Single-value-resampling can

1. To be consistent with all KDE related work 1, 6, 7, 9, 10, 11, we also use the term kernel or kernel point to represent the data samples, and kernel function to represent the function \(K(\cdot)\).
TABLE 1: Summary of the key characteristics of the density estimators (MV = Multivariate data, RS = Random Sampling, GS = Group Selection, SS = Sort Selection and BU = Batch Update).

<table>
<thead>
<tr>
<th>Method</th>
<th>MV</th>
<th>Data streams</th>
<th>Bandwidth selection</th>
<th>Bandwidth on each dimension in MV</th>
<th>data points reduction technique</th>
<th>MV Kernel function</th>
<th>Online update</th>
</tr>
</thead>
<tbody>
<tr>
<td>CK [7]</td>
<td>No</td>
<td>Yes</td>
<td>Normal rule</td>
<td>N/A</td>
<td>Merge kernels</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>M-Kernels [1]</td>
<td>No</td>
<td>Yes</td>
<td>Normal rule</td>
<td>N/A</td>
<td>Merge kernels</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SOMKE [9]</td>
<td>Yes</td>
<td>Yes</td>
<td>Normal rule</td>
<td>Different</td>
<td>Trained neurons</td>
<td>MV kernel function</td>
<td>BU</td>
</tr>
<tr>
<td>FFT-KDE [13]</td>
<td>Yes</td>
<td>Yes</td>
<td>Normal rule</td>
<td>Different</td>
<td>none</td>
<td>MV kernel function</td>
<td>No</td>
</tr>
<tr>
<td>KDE [14]</td>
<td>Yes</td>
<td>Yes</td>
<td>Normal rule</td>
<td>Different</td>
<td>none</td>
<td>Product kernel function</td>
<td>No</td>
</tr>
<tr>
<td>kd-tree [15]</td>
<td>Yes</td>
<td>No</td>
<td>Cross validation</td>
<td>Fixed</td>
<td>none</td>
<td>Rotation-invariant function</td>
<td>No</td>
</tr>
<tr>
<td>RS, GS and SS [10]</td>
<td>Yes</td>
<td>No</td>
<td>User input</td>
<td>Fixed</td>
<td>sampling</td>
<td>Rotation-invariant function</td>
<td>No</td>
</tr>
<tr>
<td>MPLKernels [11]</td>
<td>Yes</td>
<td>Yes</td>
<td>Normal rule</td>
<td>Different</td>
<td>sampling</td>
<td>Product kernel function</td>
<td>Yes</td>
</tr>
<tr>
<td>KDE-Track (proposed)</td>
<td>Yes</td>
<td>Yes</td>
<td>Normal rule</td>
<td>Different</td>
<td>none</td>
<td>Product kernel function</td>
<td>Yes</td>
</tr>
</tbody>
</table>

be inaccurate since only one sample is used to represent all samples in that kernel. Regeneration of all samples in kernel might be accurate but very costly in computation. Moreover, in [6], the dynamic changes in the data distribution of evolving data streams will require rebuilding the model frequently to define the set of local regions, which will be very costly. Another method merges kernel based on clustering by Self-Organizing Maps (SOM) [9]. Only trained SOM neurons are utilized in density estimation, rather than the whole set of kernels. In order to train the neurons and to minimize the time complexity, a data stream is considered as a sequence of disjoint windows where data in each window are assumed to have the same distribution. Depending on the window size, this assumption may be violated in scenario of data streams.

**Sampling** was used in [10] to reduce the number of kernels in large datasets while guaranteeing an \( \epsilon \)-approximation of the density function. The authors studied random sampling and proposed group-selection and sort-selection, which achieve the same accuracy as random sampling but with a smaller number of samples. Sampling can significantly reduce computation cost, but sacrificing estimation accuracy.

**Space partitioning** is also used to reduce the computational cost of KDE. A **kd-tree** structure is used in [11] and [15] where the leaves contain a small number of kernels and each internal node contains a statistical summary about the subspace represented by that node. Estimating the density at any given point involves depth-first traversal of the tree where only close-by nodes will be visited. These methods have the same problem of representing many kernels by a single value. Controlling the size of the tree is another issue. Grid-based methods were presented in [12], [18], [19], [20], [21] for static datasets and concentrate on the best setting of the bin width using a fixed number of resampling points. This approach will not work for data streams as the data are not available in advance and the range of the data is changing over time which requires an adjustable model.

Our study differs from the above-mentioned methods by updating the estimated density with the con-

3. **Theoretical Bases**

In this section, we discuss the traditional KDE and its related issues, e.g., selection of kernel functions and smoothing parameter (bandwidth), and complexity. KDE estimates the density \( f(x) \) by Eq. (1). For the case of univariate data, Eq. (1) is written as:

\[
\hat{f}(x) = \frac{1}{nh} \sum_{j=1}^{n} K \left( \frac{x - x_j}{h} \right). 
\]

For the 2-d spatial samples, where \( x_j = (x_{1j}, x_{2j})^T \in \mathbb{R}^2 \), kernel functions \( K_h(x, x_j) \) are defined as:

\[
\frac{1}{h_{1}h_{2}} K \left( \frac{x_{1} - x_{1j}}{h_{1}}, \frac{x_{2} - x_{2j}}{h_{2}} \right),
\]

where \( h_i \) is the smoothing parameter, called the bandwidth, on dimension \( i \).

A popular kernel function in case of multivariate data is called the multipliciative (product) kernel [8], which uses the product of univariate kernel functions on each dimension, and computes \( \hat{f}(x) \) as:

\[
\hat{f}(x) = \frac{1}{n} \sum_{j=1}^{n} \prod_{i=1}^{2} \left\{ \frac{1}{h_{i}} K \left( \frac{x_{i} - x_{ij}}{h_{i}} \right) \right\}. 
\]

Another option is to use the orientation-invariant kernel function [10] and [15], which is:

\[
\hat{f}(x) = \frac{1}{nh^2} \sum_{j=1}^{n} K \left( \frac{||x - x_j||}{h} \right). 
\]

This kernel function assumes that the data variation along all the dimensions is the same, which may fail to capture densities of arbitrary shapes.

3.1 **KDE related issues**

The choice of a kernel function is relatively unimportant provided that a kernel function is continuous with finite support [14]. It is recommended that the selected kernel is smooth, clearly unimodal and symmetric about the origin [8]. In our density estimator, we choose the multipliclicative Epanechnikov kernel where the same univariate kernel function \( K(x) = \frac{3}{4}(1 - x^2)I_{[-1, 1]}(x) \) is used in each dimension with different bandwidth value. We used the Epanechnikov.
kernel because of its asymptotically-optimal efficiency among all other kernel functions [22]. The estimation accuracy of KDE is mainly affected by the bandwidth value [8], [14]. A large bandwidth value over-smooths the density function curve and hides a lot of useful information, while a small bandwidth value makes the density function’s curve too fluctuated. A general rule for selecting the bandwidth is to decrease the bandwidth value \( h \to 0 \) as the number of samples used in the estimation increases \( n \to \infty \). However, the rate at which \( h \) approaches 0 is much slower such that \( nh \to \infty \).

3.2 The bandwidth \( h \nabla \)

Eqs. (1) and (3) use different bandwidth values to capture the spread of the data on each dimension. This suggests using the same analysis of estimating the bandwidth for the case of univariate data on the marginal distribution of the data on each dimension. A typical rule of bandwidth setting is to minimize the deviation between the true and the estimated densities. This deviation is commonly measured by the Mean Integrated Square Error (MISE) [23]. Let \( \mu_k(x), R(f) \) be defined as

\[
\mu_k(x) = \int x^k K(x)dx \quad \text{and} \quad R(f) = \int f^2(x)dx.
\]

The MISE of the estimator using a bandwidth value \( h \) is

\[
\text{MISE}(h) = \int E \left( \hat{f}(x) - f(x) \right)^2 dx,
\]

which has the asymptotic expansion \( \text{MISE}(h) = \text{AMISE}(h) + O(n^{-1} + h^4) \) under suitable regularity conditions on \( K \) and \( f \). The minimizer of the AMISE\( (h) = \frac{1}{hn}R(K) + h^4 \left( \frac{\mu_2(K)}{4} \right)^2 R(f''(0)) \) is considered a good approximation for the optimal bandwidth value, which can be estimated as

\[
\hat{h} = \left( \frac{R(K)}{\mu_2(K)R(f''(0))} \right)^{1/4}.
\]

However, this minimizer cannot be computed as it depends on the unknown density \( f \).

Many methods have been introduced to estimate \( R(f'') \) in Eq. (5). The normal rule [14] is the most popular method for estimating the bandwidth, which assumes the unknown density \( f \) as a normal density and scales it according to the standard deviation of the data samples. The bandwidth value selected using the normal rule is computed as:

\[
\hat{h} = c\sigma n^{-1/5},
\]

where \( c \) is a constant that depends on the used kernel function \( K \), \( \sigma \) is the sample standard deviation and \( n \) is the number of kernels. This method is timely efficient but it does not work well when the density deviates significantly from normality. Other methods based on cross-validation have been proposed in the literature [24], [25], [26], [27], [28], [29]. These methods require performing density estimation for each candidate of the bandwidth values, which multiplies the computational cost by another factor equal to the number of candidates.

Plug-in methods [30], [31], [32], [33] estimate an approximation of \( R(f'') \) and plug it in Eq. (5) to compute the optimal bandwidth. Estimating \( R(f'') \) requires also making assumptions about the density function but it becomes more accurate than using the normal rule. Sheather and Jones [29] estimate \( R(f'') \) by estimating \( f^{(4)} \), which in turn is estimated using \( R(f^{(6)}) \). The value \( R(f^{(6)}) \) is computed by assuming that \( f^{(8)} \) is the eighth derivative of a normal density. After estimating \( R(f^{(6)}) \), a backward substitution is performed to estimate \( R(f'') \). Shimazaki and Shinomoto [34] assume that the true density follows a Poisson distribution and use Estimation-Maximization (EM) method to find the optimal bandwidth value. This method requires estimating the density for each estimation step of the EM optimization procedure, which will be very expensive in the case of streaming data where the density is changing dynamically and the bandwidth value needs to be estimated frequently.

In the case of multidimensional data, most of the bandwidth selection methods either consider a fixed bandwidth value for all the dimensions [10], [5], [15], [35], [36], [32] or use the marginal distribution to estimate the bandwidth on each dimension [11], [14]. KDE fails to capture densities of arbitrary shapes when using the same bandwidth value for all dimensions. Methods that select different bandwidth values for each dimension rely on the marginal distribution of the data on that dimension.

In this work, we minimize the effect of the normality assumption of \( f \) by using the data samples to estimate \( f'' \). The numerical integration technique is then used to compute \( R(f'') \), which is plugged in Eq. (5) to estimate the bandwidths.

4 KDE-Track Method

In this section, we describe our model (KDE-Track) for estimating the dynamic density functions that come with data streams. We theoretically analyze how KDE-Track minimizes the estimation error.

4.1 KDE-Track overview

We model the distribution of the streaming data as a grid of resampling points and their corresponding estimated density values. Let \( U^1 = \{ u_0^1, u_1^1, \ldots, u_{l_1-1}^1 \} \) and \( U^2 = \{ u_0^2, u_1^2, \ldots, u_{l_2-1}^2 \} \) be the set of points that discretize the range of the data on the first and the second dimensions, respectively. The KDE-Track model \( M \) is defined as the set of the grid points from \( U^1 \times U^2 \) with their estimated densities. That is, \( M = \{ M_0, M_1, \ldots, M_{q-1} \} \), where \( q = U^1U^2 \) is the number of the resampling points and \( M_s \) is an ordered pair representing a grid point and its estimated PDF \( (M_s = (m_s, \hat{f}(m_s))) \). Here \( m_s = (u_k^1, u_l^2) \in U^1 \times U^2 \) is the \( s \)-th resampling point with \( k, l \) being the quotient.
and the remainder of the division of $s$ by $U^1$ and $f(m_s)$ is the density estimated using KDE at $m_s$.

Density estimation using bilinear interpolation is based on constructing the grid of resampling points and estimating their corresponding density values. Estimating the PDF at a data sample $a$ by bilinear interpolation of the resampling points has 2 steps: (1) fetch the estimated PDF values at resampling points $m_{s1}, m_{s1+1}, m_{s2}$ and $m_{s2+1}$ that surround the point $a$ as in Figure 1. Let $y^{(i)}$ be the projection of vector $y$ on $i$-axis, then $m_{s1}^{(1)} = m_{s2}^{(1)} = m_{s2+1}^{(1)} = m_{s1}^{(1)} + 1 < a^{(1)} < m_{s2+1}^{(1)}$ and $m_{s1}^{(2)} = m_{s2}^{(2)} = m_{s2+1}^{(2)} = m_{s2+1}^{(2)} + 1$. (2) estimate the density at $a$ using the interpolation.

4.2 Density estimation by interpolation

Estimating the density at $a$ by bilinear interpolation can be done by linearly interpolating the density at $m_{s1}, m_{s1+1}$ to estimate the density at $r_{s1}$ and interpolating the density at $m_{s2}, m_{s2+1}$ to compute the density at $r_{s2}$. The final step interpolates the density at $r_{s1}, r_{s2}$ to find the required density at $a$. Let $D(b,c)$ be the Euclidean distance between $b$ and $c$. The density at $a$ will be computed as follows:

$$
\hat{f}(a) = \frac{D(a, r_{s2}) \hat{f}(r_{s1}) + D(r_{s1}, a) \hat{f}(r_{s2})}{D(r_{s1}, r_{s2})},
$$

(7)

where

$$
\hat{f}(r_{s1}) = \frac{D(r_{s1}, m_{s1+1}) \hat{f}(m_{s1}) + D(m_{s1}, r_{s1}) \hat{f}(m_{s1+1})}{D(m_{s1}, m_{s1+1})},
$$

and

$$
\hat{f}(r_{s2}) = \frac{D(r_{s2}, m_{s2+1}) \hat{f}(m_{s2}) + D(m_{s2}, r_{s2}) \hat{f}(m_{s2+1})}{D(m_{s2}, m_{s2+1})}.
$$

KDE interpolation is efficient as it stores only the function at the resampling points whose total number is in the constant order and is small compared to the stream size. The running time for estimating the PDF for all $n$ arriving data samples will be in $O(n|M|)$.

KDE-Track incurred three types of error: the estimation error inherited from KDE, the interpolation error and the rounding error. Since rounding error (occurring when infinite number of digits after the decimal point are squeezed in a finite number of bits) is machine dependent, we focus on the interpolation error and propose an adaptive resampling model to minimize this type of error. The error inherited from KDE will be minimized by selecting the optimal bandwidth values for the KDE.

In [33], we studied the interpolation error for the case of univariate data. Let $\hat{f}(a)$ and $\hat{f}(a)$ be the estimated PDFs using the traditional KDE and the KDE-Track, respectively, and $D_m$ be the maximum distance between two consecutive resampling points. The error is found to be

$$
\hat{f}(a) - \hat{f}(a) = \frac{D^2}{8} f''(a) + O_D(D^3) .
$$

The interpolation error will increase in the case of spatial data. The bounding box of a sample $a$ contain four resampling points such that, in each dimension, the projection of $a$ lies between two of these resampling points. An upper bound of the bilinear interpolation error can be found in the following way. Let $\hat{f}$ be the estimated density using the traditional KDE. Assume that $\hat{f}$ is twice continuously differentiable in open neighborhood around $r_{s1}, r_{s2}$ and $a$. We use the notation $\hat{f}_x = \frac{\partial \hat{f}}{\partial x}$ and $\hat{f}_{xx} = \frac{\partial^2 \hat{f}}{\partial x^2}$. Writing

$$
\hat{f}(m_{s1}), \hat{f}(m_{s1+1}), \hat{f}(m_{s2}) \text{ and } \hat{f}(m_{s2+1}) \text{ as Taylor expansions of } \hat{f} \text{ around } r_{s1} \text{ and } r_{s2}, \text{ we get:}
$$

$$
\hat{f}(m_{s1}) = \hat{f}(r_{s1}) - D_{11} \hat{f}_{x1}(r_{s1}) + \frac{D_{12}^2}{2} \hat{f}_{x1x1}(r_{s1}) + O_D(D_{11}^3),
$$

$$
\hat{f}(m_{s1+1}) = \hat{f}(r_{s1}) + D_{12} \hat{f}_{x1}(r_{s1}) + \frac{D_{12}^2}{2} \hat{f}_{x1x1}(r_{s1}) + O_D(D_{12}^3),
$$

$$
\hat{f}(m_{s2}) = \hat{f}(r_{s2}) - D_{12} \hat{f}_{x2}(r_{s2}) + \frac{D_{12}^2}{2} \hat{f}_{x2x2}(r_{s2}) + O_D(D_{12}^3),
$$

$$
\hat{f}(m_{s2+1}) = \hat{f}(r_{s2}) + D_{12} \hat{f}_{x2}(r_{s2}) + \frac{D_{12}^2}{2} \hat{f}_{x2x2}(r_{s2}) + O_D(D_{12}^3).
$$

We interpolate these function values to compute the PDF at $r_{s1}$ and $r_{s2}$ to get

$$
\hat{f}(r_{s1}) = \hat{f}(r_{s1}) + \frac{D_{22}D_{11}}{2} \hat{f}_{x1x1}(r_{s1}) + O_D(D_{11}^3),
$$

$$
\hat{f}(r_{s2}) = \hat{f}(r_{s2}) + \frac{D_{22}D_{11}}{2} \hat{f}_{x1x1}(r_{s2}) + O_D(D_{11}^3).
$$

Writing $\hat{f}(r_{s1})$ and $\hat{f}(r_{s2})$ as Taylor expansion of $\hat{f}$ around $a$, we get:

$$
\hat{f}(r_{s1}) = \hat{f}(a) - D_{22} \hat{f}_{x2}(a) + \frac{D_{22}^2}{2} \hat{f}_{x2x2}(a) + O_D(D_{22}^3),
$$

$$
\hat{f}(r_{s2}) = \hat{f}(a) + D_{22} \hat{f}_{x2}(a) + \frac{D_{22}^2}{2} \hat{f}_{x2x2}(a) + O_D(D_{22}^3).
$$

and

$$
\hat{f}_{x1x1}(r_{s1}) = \hat{f}_{x1x1}(r_{s1}) + O_D(D_{11}^2, D_{12}^2),
$$

$$
\hat{f}_{x1x1}(r_{s2}) = \hat{f}_{x1x1}(r_{s2}) + O_D(D_{11}^2, D_{12}^2).
$$

Now, compute $\hat{f}(a)$ by interpolating $\hat{f}(r_{s1})$ and $\hat{f}(r_{s2})$

$$
\hat{f}(a) = \hat{f}(a) + \frac{1}{2} \left(D_{22}D_{22} \hat{f}_{x2x2}(a) + D_{21}D_{12} \hat{f}_{x1x1}(a) \right) + \cdots
$$
Let \( D_m \) be the maximum distance between two consecutive resampling points in any dimension. Then the maximum error will be observed when \( D_{11} = D_{12} = D_{21} = D_{22} = \frac{D_m}{2} \). Then
\[
\bar{f}(a) = \hat{f}(a) + \frac{D_m^2}{8} \left\{ \frac{\hat{f}_{2,2}(a) + \hat{f}_{2,2}(a)}{\hat{f}_{2,2}(a)} \right\} + O_p(D_m^3).
\]
Note that the term \( O_p(D_m^3) \) includes also the terms with higher order derivatives. When using the Epanechnikov kernel function, the second partial derivative will be constant and the partial derivatives of higher order will be zeros.

### 4.4 Adaptive resampling model

From Subsection 4.3, we know that the accuracy of the linear interpolation depends on 1) the distance between two adjacent resampling points; and 2) the second derivative of the density function. To minimize the error while keeping the number of resampling points within a reasonable margin, we add more resampling points in the regions where the density function has high curvature, as shown in Figure 2. By contrast, in the regions where the function is approximately linear, we use less resampling points.

In spatiotemporal data streams, the distribution is spatially non-uniform and dynamic. Therefore, high resolution with sufficient resampling points is required 1) in dense areas with high PDF values to catch the details; and 2) in sensitive areas which are the boundary between dense and sparse to catch dynamic changes. Adaptive resampling meets the requirement perfectly because both dense areas and sensitive areas generally have density function with high curvature. It detects the high curvature by the second derivative of the density function, which will be introduced in next section.

![Fig. 2: Adaptive resampling: more resampling points are used in regions with high curvature of the function.](image)

### 4.5 Bandwidth selection

Based on the discussion in Subsection 3.2, we use different bandwidth values for each dimension. The marginal distribution of the data is used to compute the bandwidth value on that dimension. From Eq. (5), the bandwidth value on the dimension \( i \in \{1, 2\} \) can be computed as
\[
\hat{h}_i = \left( \frac{R(K)}{\mu_x^2(K) R(f''_i)n} \right)^{\frac{1}{5}},
\]
where \( f''_i \) is the second derivative of the density function on the \( i \)-th dimension. We start by estimating a pilot bandwidth using the normal rule \( \hat{h}_i = c\sigma_i n^{-1/5} \), where \( c \) is a constant that depends on the kernel function \( K \) and \( \sigma_i \) is the standard deviation of the projection of the data on the axis \( i \). This pilot bandwidth is used to estimate the second derivative of the marginal distribution as
\[
f''_i(x) = \frac{1}{n\hat{h}_i} \sum_{j=1}^{n} K''(\frac{x - x_j}{\hat{h}_i}).
\]
In this case, \( f''_i \) will be a better approximation of \( f''_i \) than considering \( f_i \) to be a normal density. Using KDE-Track on the one dimensional data will speed up the computation of \( R(f''_i) \) and \( \hat{h}_i \). Since the distribution of the data will change over time with the arrival of new samples from the stream, the bandwidth values \( \hat{h}_i \) should be updated accordingly to represent the variation of the data along the different dimensions. Using KDE-Track will also allow for updating the values of \( \hat{h}_i \) online and efficiently.

In this way, the estimated \( f''_i \) will serve two roles. First, it is used to approximate \( R(f''_i) \) to compute the bandwidth value. Second, it is used as a more accurate indicator of the high curvature of the density function’s curve, which facilitates the adaptive resampling in KDE-Track for obtaining more accurate estimation as we will discuss in the following subsections.

### 5 KDE-Track Implementation

Estimating the density for each incoming data samples using KDE-Track requires access to four resampling points only as discussed in Section 4. The key step is thus the maintenance of the resampling model (resampling points and their PDF values). Algorithm shows the maintenance of KDE-Track’s model and using it for online density estimation. The lines starting with the # sign represent comments.

#### 5.1 Initializing the resampling model

The resampling model is initialized by the beginning part of streaming data, e.g., the first 5000 points. The resampling points, \( m_{s, s} = 1, \ldots, q_0 \), are defined as the cartesian product of the set of equidistant

2. This first batch of data is used for initializing the resampling points and setting bandwidth values. KDE-Track is not sensitive to how many points are used in this batch, as the resampling model and bandwidth are updated online with new arriving data after initialization.
Algorithm KDE-Track: Online Density Estimation

Parameters: $w$ (window size)

Online flow in: streaming data $S = \{x_1, \ldots, x_t, \ldots\}$

Online output: $f(x_t)$ (the PDF value at $x_t$)

Procedure:

1. **# Initialization:**
   
2. $W = \{x_1, \ldots, x_w\}, \text{update\_step} = 0.05w$ and $M = \phi$
3. Compute $h_{i1} = \left(\frac{R(K)}{\mu^2_{i1}(K)\mu_{i1}(K)}\right)^{\frac{1}{2}}$ (Sec. 4.5)
4. for $k = 0$ to $U^1 - 1$
5. for $l = 0$ to $U^2 - 1$
6. Put $s = U^1 + k$ and $m_s = (u_{i1}^s, u_{i1}^{s+1})$
7. Compute $\tilde{f}(m_s)$ using Eq. (3)
8. Put $M_s = (m_s, \tilde{f}(m_s))$
9. $M = M \cup \{M_s\}$
10. end for
11. end for
12. while a new sample $x_t$ arrives in the stream do
13. if $(1 \leq t < w)$ then
14. Compute $\tilde{f}(x_t)$ using Eq. (7)
15. else
16. **# Update the resampling model:** (Sec. 5.3)
17. Remove $x_{t-w}$ from $W$ and add $x_t$ to $W$
18. for each dimension $i$ do
19. Update $\hat{s}_i$ using $x_t, x_{t-w}$
20. Compute $\hat{h}_{i1} = \sigma\hat{s}_i w^{-1/5}$
21. Update $\hat{f}_i''(u_k^{i1})$
22. end for
23. for each dimension $i$ do
24. Compute $R(\hat{f}_i'')$, $\hat{h}_{i1} = \left(\frac{R(K)}{\mu^2_{i1}(K)\mu_{i1}(K)}\right)^{\frac{1}{2}}$
25. $\forall s(0 \leq s \leq U^1 U^2 - 1)$, update $\tilde{f}(m_s)$ by Eq. (11)
26. **# Update the adaptive resampling:**
27. if mod($t, \text{update\_step}$) = 0 then
28. Compute $\hat{f}_i'' = \frac{1}{U^1 U^2} \sum_{j=0}^{U^2-1} |\hat{f}_i''(u_j^{i1})|$
29. if $\max(|\hat{f}_i''(u_k^{i1-1})|, |\hat{f}_i''(u_k^{i1})|) > \hat{f}_i''$ then
30. $u_{i1}^{\text{temp}} = (u_{i1}^{k-1} - u_{i1}^{k-2})/2$
31. \text{Compute $\hat{f}_i''(u_{i1}^{\text{temp}})$}
32. Insert $u_{i1}^{\text{temp}}$ in $U^1$
33. end if
34. $\Theta_t = \max(|\hat{f}_i''(u_{i1}^{k-1})|, |\hat{f}_i''(u_{i1}^{k})|, |\hat{f}_i''(u_{i1}^{k+1})|)$
35. if $\Theta_t < 0.05 \hat{f}_i''$ then
36. Merge $[u_{i1}^{k-1}, u_{i1}^{k}]$ and $[u_{i1}^{k}, u_{i1}^{k+1}]$
37. end if
38. end if
39. Compute $\tilde{f}(x_t)$ using Eq. (7)
40. end if
41. end while

5.2 Estimation based on the resampling model

Once all $M_t$ have been initialized, we can estimate the density at each arriving point $x_t$. Due to our interpolation method, the density at $x$ can be estimated by 1) calculating, on each dimension, the index of one resampling point who and whose successive neighbor will contribute; 2) fetching the four resampling points in the cell surrounding the point $x$ and their densities; and 3) computing $\tilde{f}(x)$ using Eq. (7).

5.3 Updating the resampling model

The resampling model is the basis of our density estimator. The resampling points and their PDF values should be updated after receiving a new data point. As we discussed in Subsection 4.4 the resampling points are adaptively maintained according to the curvature of the density function. An interval $[u_{i1}^j, u_{i1}^{j+1}]$ is divided into two equal subintervals when $\max\{|\hat{f}_i''(u_{i1}^j)|, |\hat{f}_i''(u_{i1}^{j+1})|\} > \hat{f}_i''$, where $\hat{f}_i'' = \frac{1}{U^1 U^2} \sum_{j=0}^{U^2-1} |\hat{f}_i''(u_j^{i1})|$ is the average of the second derivative absolute values. In this way, more resampling points are inserted in areas which are the boundary between dense and sparse or are dense with high peak values in density function. Two intervals $[u_{i1}^{j-1}, u_{i1}^j]$ and $[u_{i1}^j, u_{i1}^{j+1}]$ are merged to reduce the number of resampling points when the density function is close to linear, which means $|\hat{f}_i''(u_{i1}^{j-1})|, |\hat{f}_i''(u_{i1}^j)|$ and $|\hat{f}_i''(u_{i1}^{j+1})|$ are close to zero (less than 0.05 $\hat{f}_i''$). In sparse regions, the PDF values are close to zero and the function is almost linear so the intervals are also merged to reduce the number of resampling points. Note that once a given interval has been split, it will not be split again until it gets merged with another interval. The same condition is applied for merging the intervals.

When updating the densities of resampling points in the model $M_t$, we should consider the evolution of the data distribution. We use a sliding window strategy to catch the evolution over time. The window size $w$ is an application dependent parameter and can be set based on the arrival rate of the data samples and the time interval during which we need to estimate the dynamic density. The window size also controls the robustness of KDE-Track against noisy data where an isolated outlier will increase the height of the PDF curve by maximum $1/w$, which will not be noticed. However, when a new pattern arrives, the new points will replace points in the sliding window from the old pattern and their contribution will be observed on the shape of the density function after receiving a reasonable number of data points from the new pattern. Let $n_{it}$ denote the number of points we have received until time $t$. Due to the difference between $w$ and $n_{it}$, there are two different scenarios when updating the model $M_t$ more specifically, updating the bandwidths and density $\tilde{f}(m_s)$:

(i) When $n_{it} \leq w$. The received points cannot fill the whole window. The pilot bandwidth value at time $t$
is calculated using all \( n_t \) points by the formula \( \hat{h}_{i,t} = c\hat{\sigma}_i n_t^{-1/5} \), where \( \hat{\sigma}_i^2 \) is the sample variance of the received data samples projected on the \( i \)-th axis calculated as \( \hat{\sigma}_i^2 = \frac{1}{n_t-1} \sum_{j=1}^{n_t} x_{ij}^2 - \frac{1}{n_t} \left( \sum_{j=1}^{n_t} x_{ij} \right)^2 \), \( i \in \{1, 2\} \) [59], which can be updated with a constant time at each \( t \). The pilot bandwidth is used to update the estimation of the second derivative of the data marginal distribution on dimension \( i \). The roughness \( R(f_i^{(n)}) \) is then computed to estimate the bandwidth value on that dimension.

After receiving a point \( x_t \), the density at a resampling point \( m_s \) at time \( t \) is updated using sample-point estimator [40]

\[
\hat{f}_t(m_s) = \frac{n_t - 1}{n_t} \hat{f}_{t-1}(m_s) + \frac{1}{n_t \hat{h}_1 \hat{h}_2} K_{\hat{h}}(m_s, x_t),
\]

where \( K_{\hat{h}} \) is defined in Eq. (3). It is straightforward to show that the updated density \( \hat{f}_t(m_s) \) is a good approximation to the estimated density using all the \( n_t \) points. In particular, since

\[
\hat{f}_t(x) = \frac{1}{n_t} \sum_{j=1}^{n_t} \frac{1}{\hat{h}_1 \hat{h}_2} K_{\hat{h}}(x, x_j) \geq 0 \quad \text{and} \quad \forall \ j \ \text{the integration over} \ x \ \text{of} \ \frac{1}{\hat{h}_1 \hat{h}_2} K_{\hat{h}}(x, x_j) = 1,
\]

averaging the integrations of the two terms in (10) results in 1.

(ii) When \( n_t > w \). In this case, the pilot bandwidth is calculated on the most recently received \( w \) points inside the window as follows: \( \hat{h}_{i,t} = c\hat{\sigma}_i w^{-1/5} \), where the sample variance \( \hat{\sigma}_i^2 \) of the projected data on the \( i \)-th dimension can be easily updated by

\[
\hat{\sigma}_i^2 = \frac{1}{w - 1} \left( \sum_{j=-w+1}^{t} x_{ij}^2 - \frac{1}{w} \left( \sum_{j=-w+1}^{t} x_{ij} \right)^2 \right).
\]

The pilot bandwidth is used to update the estimation of the second derivative values and to compute the bandwidth \( \hat{h}_{i,t} \), which is used to update the density function at the resampling points. The PDF values at the resampling points \( \hat{f}_t(m_s) \) are updated by absorbing the new arrived point \( x_t \) and deleting the old point that was moved out of the window:

\[
\hat{f}_t(m_s) = \hat{f}_{t-1}(m_s) + \frac{K_{\hat{h}}(m_s, x_t)}{w \hat{h}_1 h_2} - \frac{K_{\hat{h}}(m_s, x_{t-w})}{w \hat{h}_1 \hat{h}_2}.
\]

The probabilistic properties of updated density function \( \hat{f}_t(x) \) can be proved as:

1) \( \hat{f}_t(x) \geq 0, \forall x \), due to the fact that \( \int_{-\infty}^{\infty} \hat{f}_t(x) dx_1 dx_2 = 1 \) is a summation of nonnegative terms;
2) the integration \( \int_{-\infty}^{\infty} \hat{f}_t(x) dx_1 dx_2 = 1 \).

When updating the resampling model, we should consider the changes in the regions of the PDF with high curvature as the density function is changing over time. Furthermore, when updating the model, we consider extending and shrinking the resampling model to cover the density function support. When data from a new distribution arrive, we extend the resampling model to cover the range of the data. As a result of the evolution in data streams, distributions might disappear from the stream so that the resampling points in the regions where no data samples were received for long time should be removed. As the changes in the density function curved regions cannot be observed after receiving a single data sample, we are checking for such changes after receiving a predefined number of points, e.g., 5–20% of the sliding window size.

### 5.4 Time and Space Complexity Analysis

Based on the discussion earlier in this section, the time complexity of estimating the density for a new incoming data point is \( O(U_1^1 + U_2^2) \), where \( U_1 \) and \( U_2 \) were given in Subsection 4.1. They are independent of the number of points that have been received from the data stream. Updating the model when receiving a new point requires computing time linear to the total number of the resampling points \( |M| \), since all the function values at the resampling points are updated. The overall time complexity of processing each arriving point is linear to the model size, which is usually a limited small number. The time required for online density estimation of a data stream with \( n \) points is \( O(n|M|) \), which linearly increases with the number of received points from the stream.

Bandwidth selection requires maintaining a one dimensional KDE-Track model on each dimension. Since \( U_1^1 \times U_2^2 = |M| \) and \( U_1^1, U_2^2 \geq 1 \), the total number of resampling points in both models of the one dimensional KDE-Track is \( U_1^1 + U_2^2 \leq |M| + 1 \), which will increase only the constant in the KDE-Track’s time complexity formula. Thus, the KDE-Track time complexity is \( O(n|M|) = O(n \times U_1^1 \times U_2^2) \).

During the online density estimation process, KDE-Track keeps the resampling model \( M \) and the points in the sliding window in memory. Therefore, the memory usage is \( |M| w = U_1^1 \times U_2^2 \times w \). Note that the model size \( |M| \) changes upon the distribution variation in data streams due to merge/split operations in adaptive resampling.

### 5.5 Multidimensions

Extending the two dimensional KDE-Track to higher dimensions is straightforward. The same technique can be used for selecting the bandwidth using the marginal distribution of the data on each dimension. The KDE-Track model for estimating the density of \( d \)-dimensional data can be constructed as follows:

1) discretize the range of the data on the \( i \)-th dimension, with \( 1 \leq i \leq d \), using a set of points \( U_i^1 \); 2) define the set of resampling points as the cartesian product \( U_1^1 \times U_2^2 \times \cdots \times U_d^d \); and 3) estimate the density function values at the set of resampling points and store them with their estimated density in the model \( M \).
The product kernel defined in Eq. (3) will be
\[ K_h(\mathbf{x}, \mathbf{x}_j) = \prod_{i=1}^{d} \frac{1}{h_i} K\left( \frac{x_i - x_{ij}}{h_i} \right). \]

Researchers avoid to use the product kernel for the case of high dimensional data and replace it with an orientation-invariant kernel function
\[ K_h(\mathbf{x}, \mathbf{x}_j) = \frac{1}{h} K\left( \frac{\|\mathbf{x} - \mathbf{x}_j\|}{h} \right), \]
which may not be able to estimate densities with arbitrary shapes as it assumes equal variance values of the data on each dimension.

The interpolation error for the case of \( d \)-dimensional data can be bounded as follows: let \( D_m \) be the maximum distance between the resampling points in dimension \( i, i \in \{1, 2, \ldots, d\} \) and \( D_m = \max\{D_{m1}, 1 \leq i \leq d\} \). Then we can have the error as
\[
|f(\mathbf{a}) - \hat{f}(\mathbf{a})| \leq \frac{D_m^{2d}}{8} \left( \sum_{i=1}^{d} f_{x_i}(\mathbf{a}) \right) + O_p(D_m). 
\]

This error is reducible by including more resampling points in certain regions.

The KDE-Track’s time complexity will remain linear in the size of the stream \( O(n|M|) \). However, \( |M| = |U|^d \times |U|^2 \times \cdots \times |U|^d \) will increase with the number of dimensions \( d \) and the number of resampling points \( |U|^d \) on each dimension \( i \).

6 Performance Evaluation

In order to evaluate our method, we have run extensive number of experiments on synthetic data. Here we report the results for one 1D stream and one 2D stream. Since the true densities are known, we evaluate the performance of the density estimators by comparing the estimated densities with the true ones. We compare KDE-Track with many baseline methods in terms of estimation accuracy and running time. The accuracy of the different methods is measured by the Mean Absolute Error (MAE) and the \( l_\infty \) error.

6.1 Estimation accuracy on synthetic data

6.1.1 Datasets

The one-dim stream (1D) was generated by extracting data samples from the fifteen densities suggested...
by Marron and Wand [41] and presented in Figure 3. The stream is constructed by extracting $3 \times 10^4$ data samples from each density and concatenating the batches to get one stream of $4.5 \times 10^5$ data samples. The two-dim data stream (S2D) is generated by extracting data segments of size $10^5$ from the seven densities presented in Figure 4. The total size of the stream is $7 \times 10^5$ data samples. These streams are separated because they contain challenging densities that are hard to be estimated accurately. Using batches of the same size is to simplify the calculation of the true density only.

### 6.1.2 Bandwidth selection

The first experiment evaluates the proposed bandwidth selection method. We compare the accuracy of KDE-Track when using different bandwidth selection methods to estimate the density of S1D. The baseline methods include 1) the normal rule, which estimates the bandwidth by assuming the underlying distribution is normal; 2) the analytical bandwidth, which is computed by Eq. (5) with $R(f'')$ analytically derived from the true density (known in synthetic data) and 3) Shimazaki’s method [24]. Note that Sheather-Jones method [29] is not compared due to its quadratic complexity w.r.t. window size. We had to stop its running after one month without any results.

The different bandwidth selection methods are evaluated in terms of the MAE and the running time. The MAE, for a given window size, is computed as follows: i) define a set of evaluation checkpoints $C = \{c_1, c_2, \ldots\}$, where the occurrence of $c_j, c_{j+1}$ are separated by receiving 1000 samples from the stream; ii) at each checkpoint $c_j$, generate an evaluation set $E = \{c_1, c_2, \ldots, c_{1000}\}$ of 1000 data samples from the same distribution of the data in the sliding window; iii) compute the MAE$_{c_j}$ by averaging the difference between the true and the estimated density values $\frac{1}{1000} \sum_{k=1}^{1000} |f(e_k) - \tilde{f}(e_k)|$; iv) compute MAE$_w = \frac{1}{|C|} \sum_{c_j \in C} \text{MAE}_{c_j}$, where $|C| = n/1000$ is the number of checkpoints, $n$ is the size of the stream and $w$ is the window size; v) the experiment is repeated for 20 instances of S1D and the average of the MAE$_w$ is reported in Figure 5. The window size changes from $1 \times 10^5$ to $3 \times 10^5$. Note that, at every checkpoint $c_j$, the set of evaluation points differs from the set of evaluation points at checkpoint $c_{j-1}$ and all the methods are evaluated using the same set of evaluation points.

First, we see that the estimation error decreases with increased window size $w$ by including more data in the estimation. However, the MAE does not keep decreasing noticeably, because a large window contain more difficult densities to estimate. Second, our proposed method has always comparable estimation error with Shimazaki’s and the analytical method, but is much more efficient. When $w$ increases, our method is continuously efficient, while Shimazaki’s and the analytical method take more time, due to the heavy calculations for approximating $R(f'')$. The normal rule is most efficient but has much higher error.

### 6.1.3 Estimation accuracy

The estimation accuracy of KDE-Track is compared with the accuracy of four baseline methods (except CK for S2D data as CK is proposed to estimate the density for univariate data only). The baseline methods are: 1) the traditional KDE [14] defined in Eq. (1); 2) the FFT-KDE [14], [13], which deploys FFT to convolve a very fine histogram of the data with a kernel function to produce a continuous density function; 3) the Cluster Kernels (CK) [7], which maintains a specific number of kernels by merging similar kernels; and 4) SOMKE [9], which employs SOM to cluster the data into a specific number of clusters and uses the centroids of the clusters as the set of kernels.

Selecting the bandwidth values for each estimator is done using the same settings as in the references. All the baseline methods use the normal rule because of its efficiency, except the CK method, which uses the Epanechnikov kernel function with a recommended constant $c = 1.06$. This setting enables CK to perform well when densities have high peaks and are multimodal. KDE-Track uses our proposed method for setting the bandwidths, i.e., estimating the roughness of the second derivative $R(f'')$ and plugging it in Eq. (5), which increases its accuracy significantly.

The performance of all the estimators is evaluated by the MAE and the $L_\infty$ error. The MAE measures how the estimated density curve fits the curve of the true density, while the $L_\infty$ measures the maximum variation between the true and the estimated curves. The error is computed by defining a set of checkpoints with step of 1000. For each estimation method, at each checkpoint an evaluation set $E = \{\epsilon_1, \ldots, \epsilon_{1000}\}$ of 1000 samples is generated from the same distribution of the data in the sliding window. The true and the estimated density values of the evaluation points are then compared to compute the MAE and the $L_\infty$ error.

The CK and FFT-KDE methods are not designed to capture the dynamic density of the data streams using the sliding window approach. To adapt these methods with sliding windows, we rebuild their model at each checkpoint an evaluation set $E = \{\epsilon_1, \ldots, \epsilon_{1000}\}$ of 1000 samples is generated from the same distribution of the data in the sliding window.
by smaller MAE values. However, the decrease in the MAE is not as expected because larger windows include data from different densities, which complicates the density estimation. As KDE-Track, CK, SOMKE and FFT-KDE are approximations of the KDE, they are supposed to have comparable results with KDE if not worse; however, the estimation accuracy depends on the bandwidth selection method. KDE-Track is shown to have the most accurate results. KDE, FFT-KDE and SOMKE have comparable results. In addition to the MAE, Figure 8 shows the standard deviation for the sensitivity analysis of the window size, where KDE-Track is the most accurate (with the lowest error) and most stable (with the smallest standard deviation in error), especially in the S2D streams.

We also evaluate the number of merges/splits in our resampling model, which contributes on the reduction of estimation error and running time. Table 2 reports the number of intervals that were merged/split in the two streams S1D and S2D, when the window size $w$ changes. The frequency of merge/split depends on the dynamics of the density. When $w$ is small, the changes in the density are more observable and cause more updates of intervals. When $w$ is larger, more complex densities are expected but change slowly, and thus require less updates of intervals. In our S1D and S2D streams, the densities become more complex with time as shown in Fig. 3 and 4, and thus require more splits than merges especially for large windows.

### 6.2 Computational time cost and space usage

Other important factors in the success of an online density estimator are its running time and space usage, as streaming data arrive fast and have unlimited...
The SOMKE model is built by training the SOM neurons with the current window which has a time estimation of the evaluation samples. The first step is the histogram with kernel function; and 2) density estimation using all the kernels in KDE. It is expected that CK will be more timely efficient if the density estimation step of CK is more expensive than the reconstruction step of CK by visualizing the density velocity upon the arrival of a new sample from the stream instead of using disjoint time intervals, which may miss critical changes that occur at the end of the different intervals.

Service planners also can benefit from monitoring the density by forwarding more service providers to regions that demand more services at a specific time. For example, monitoring the density of taxi pickup data can tell the planners of taxi companies to forward more taxicabs to a specific region of the city.

In this subsection, we visualize the dynamic traffic distribution in the New York Taxi trips dataset. The dataset is freely available and contains records of trips that include pickup time, longitude and latitude of the pickup and drop off location, etc. We are mainly interested in the pickup time and location. Figure 10 shows the density estimated using the pickup location window size of $10^4$ data points, where the data records are sorted according to their pickup time. The first three subfigures show the pickup events occurring in the early morning of a weekend day (subfigure (a)), of a regular working day (subfigure (b)) and of a national holiday (subfigure (c)). These subfigures show more pickup events during the weekends and holidays than during regular working days in the Greenwich and the East villages where there are many restaurants and nightclubs. The frequency of pickup events also increases during the weekends as it took less than 30 minutes to record $10^4$ events in a weekend but more than 3 hours in the early morning of a working day. More taxicabs are thus suggested in that region on similar events to satisfy the high demand.

Interesting patterns of community behavior can also be found in a regular working day. Subfigures (e), (f), and (g) show the pickup events on November 7 and CK. Figure 9 shows the running time for using the density estimators for online density estimation of S1D and S2D streams. The results in the figure confirm our analysis. KDE-Track and FFT-KDE are most efficient with very small running time, which is not affected by the size of $w$.

6.3 New York Taxi Trips Data Example

One of the main advantages of using KDE-Track for density estimation is the availability of the density function values at the set of resampling points at any time point. This can be used for visualizing the density function in real time without any further processing. Density estimation based visualization is preferable over scatterplots, the most prominent success stories in statistics and visualization, since scatterplots are challenged by overdraw and clustering in the case of large datasets. Applications such as change diagnosis of data streams will benefit from using KDE-Track by visualizing the density velocity upon the arrival of a new sample from the stream instead of using disjoint time intervals, which may miss critical changes that occur at the end of the different intervals.

Available at: http://www.andresmh.com/nyctaxitrips/

Fig. 9: The running time of the different density estimators when estimating the density of the S1D (a) and S2D (b) streams. The window size varies from $1 \times 10^5$ to $3 \times 10^5$. 

and 8, 2013 at different time intervals. The pickup events during the working hours (subfigure (e)) show close to uniform distribution within the area around the Central Park. Subfigure 10 (f) shows high density at the Lincoln Center during the time interval 21:48–22:55, when a concert or other events may be over. After midnight, we can observe a small number of pickup events occurred as it took six hours to accumulate $10^4$ events with more pickup events occurred around Trump and Freedom towers.

Subfigures 10 (d) and 10 (h) show the density estimated using 3D data (longitude, latitude and trip distance). The density function is colored in blue/red for low/high density regions. Fig. 10 (d) shows that most of the trips during working hours are short trips for people to move within the island, whereas there is an increase in the number of long trips during the early morning as a result of the unavailability of public transportation (Fig. 10 (h) shows larger blue rendered volume than Fig. 10 (d) at large Distance values). Note that these snapshots are provided as examples only while KDE-Track provides an online visualization for the density function 6. Similar patterns of the density function are repeated over time with minor changes. Such patterns not only are useful in planning better services but also provide critical information to reduce social and environmental costs in the transportation systems.

7 CONCLUSION

In this paper, we studied the problem of estimating the dynamic density that comes with evolving spatiotemporal data streams. We proposed the KDE-Track method to timely track the evolving distribution and accurately estimate the probability density function of these data streams. The density function is available at any time point for visualizing and monitoring the data streams. The effectiveness and efficiency of KDE-Track have been analytically studied and experimentally demonstrated on both synthetic data streams and real-world New York Taxi trips data. In our future work, we will study the application of KDE-Track for data stream clustering. In [44], streaming data were mapped into a discretized density grid, which is similar to our resampling model but used for recoding mapped characteristics. Clustering was performed off-line by connecting neighboring dense grids. We will investigate the usage of estimated density for clustering, since ‘dense’ areas are explicitly indicated by the estimated densities.

REFERENCES


Abdulhakim Qahtan received his Ph.D. degree in computer science from King Abdulaziz University of Science and Technology (KAUST), Saudi Arabia in 2016. He received his M.Sc. in information and computer science from King Fahd University of Petroleum and Minerals, Saudi Arabia in 2008. His B.Sc. in computer science was earned from Cairo University, Egypt in 2000. His main research interests include data stream mining, machine learning and statistics.

Suojin Wang is a Professor of Statistics and Epidemiology & Biostatistics at Texas A&M University. He received his Ph.D. from the University of Texas at Austin. His research interests include semi- and non-parametric statistical methodology, and biostatistical applications. He was the Editor-in-Chief of Journal of Nonparametric Statistics during 2007–2012. He is an elected Fellow of the American Statistical Association, an elected Fellow of the Institute of Mathematical Statistics and an elected member of the International Statistical Institute.

Xiaoliang Zhang is currently Assistant Professor and directs the Machine Intelligence and Knowledge Engineering (MINE) Laboratory in King Abdullah University of Science and Technology (KAUST), Saudi Arabia. She was an European ERCIM research fellow in Norwegian University of Science and Technology, Norway, in 2010. She earned her Ph.D. degree in computer science from INRIA-Univ Paris-Sud, France in 2010. Her main research interests and experiences are in machine learning, data mining, and cloud computing.