Leveraging the Christoffel function for outlier detection in data streams

Kévin Ducharlet, Louise Travé-Massuyès, Jean-Bernard Lasserre,
Marie-Véronique Le Lann, Youssef Miloudi

Leveraging the Christoffel Function for Outlier Detection in Data Streams

Kévin Ducharlet[1,2*], Louise Travé-Massuyès[1], Jean-Bernard Lasserre[1], Marie-Véronique Le Lann[1], Youssef Miloudi[2]

[1]LAAS-CNRS, ANITI, University of Toulouse, CNRS, INSA, Toulouse, France.
[2]Carl Berger-Levrault, Limonest, France.

*Corresponding author(s). E-mail(s): kevin.ducharlet@berger-levrault.com, 0000-0003-0053-8874;
Contributing authors: louise@laas.fr, 0000-0002-5322-8418; lasserre@laas.fr, 0000-0003-0860-9913; mvlelann@laas.fr, 0000-0002-0597-5152; youssef.miloudi@berger-levrault.com;

Abstract

Outlier detection holds significant importance in the realm of data mining, particularly with the growing pervasiveness of data acquisition methods. The ability to identify outliers in data streams is essential for maintaining data quality and detecting faults. However, dealing with data streams presents challenges due to the non-stationary nature of distributions and the ever-increasing data volume. While numerous methods have been proposed to tackle this challenge, a common drawback is the lack of straightforward parameterization in many of them. This article introduces two novel methods: DyCF and DyCG. DyCF leverages the Christoffel function from the theory of approximation and orthogonal polynomials. Conversely, DyCG capitalizes on the growth properties of the Christoffel function, eliminating the need for tuning parameters. Both approaches are firmly rooted in a well-defined algebraic framework, meeting crucial demands for data stream processing, with a specific focus on addressing low-dimensional aspects and maintaining data history without memory cost. A comprehensive comparison between DyCF, DyCG, and state-of-the-art methods is presented, using both synthetic and real industrial data streams. The results show that DyCF outperforms fine-tuning methods, offering superior performance in terms of execution time and memory usage. DyCG performs less well, but has the considerable advantage of requiring no tuning at all.

Keywords: Anomaly detection, Unsupervised learning, Christoffel-Darboux kernel, Data mining, Statistics

1 Introduction

The identification and examination of uncommon observations play a crucial role in data mining, as they may signal data corruption or faulty behavior. Such unusual observations can be categorized as outliers, anomalies, out-of-distribution samples, or novelties. We specifically adopt the term "outlier" along with Hawkins' definition [15] of "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". Outliers carry valuable information about underlying processes, making them especially relevant

in applications like network traffic analysis [47], medical diagnosis [7], and fraud detection [27], where detecting abnormal behavior is crucial. Furthermore, outliers can significantly disrupt various machine learning methods employed in tasks such as prediction or decision-making, necessitating their removal to ensure the accuracy of the results obtained.

In the contemporary landscape, various data sources such as wireless sensor networks, social networks, medical systems, web traffic, and online transactions continuously generate data. The resulting datasets exhibit characteristics of uncertainty and continuous evolution, posing significant challenges for outlier detection in this dynamic environment. Traditional methods designed for *batch datasets* typically seek a mapping function that assigns an outlierness score to new samples based on the observation of an entire set of historical samples. In some cases, only these historical samples can receive an outlierness score, leaving new, unseen samples unassessed. In other scenarios, methods may categorize new samples as inliers or outliers, but the mapping function remains static and does not adapt over time. However, for effective outlier detection in data streams, methods must actively seek a mapping function that adjusts to new samples and grapple with data streams of infinite length.

In the context of outlier detection, labels are frequently unavailable [13], making it uncertain whether historical samples are genuinely outliers. While batch scenarios allow for preprocessing to label data and satisfy supervised learning conditions, obtaining a reliable set of normal samples or choosing known outliers can facilitate semi-supervised tasks. However, in the realm of data streams, the continuous evolution of data distribution renders labeling impractical and it can swiftly become outdated. Consequently, outlier detection methods must operate in an unsupervised manner. The absence of labels also introduces challenges in fine-tuning these methods, as evaluating their performance becomes arduous without labeled data.

This paper focuses on unsupervised outlier detection for low-dimensional data streams. We highlight the applicability of the Christoffel function (CF), a well-established concept in the theory of approximation and orthogonal polynomials, in addressing this challenge. Our contributions encompass (1) adapting the CF to assess outliers in data streams, resulting in the *Dynamic Christoffel Function* (DyCF) method, (2) introducing a tuning-free approach called *Dynamic Christoffel Growth* (DyCG), capitalizing on the asymptotic growth properties of the CF, and (3) conducting comparisons with several state-of-the-art methods using synthetic and real industrial data streams.

The structure of this paper is as follows. In Section 3, we offer an overview of outlier detection in data streams, delving into the current state of the art. Section 4 introduces the Christoffel function (CF), illustrating its ability to effectively capture the support of a theoretical measure from a set of samples. Additionally, we compare it with the closely related method, Kernel Density Estimation (KDE). In Section 5, we present DyCF, an adaptation of the CF for handling data streams along with its tuning-free enhancement, DyCG. Section 6 provides the results of DyCF and DyCG, comparing them with state-of-the-art methods. Finally, Section 7 concludes the paper by discussing the results and suggesting potential enhancements for DyCF and DyCG, outlining avenues for future research.

2 Problem formulation

The problem that we consider is embedded unsupervised outlier detection in low dimensional data streams issued from low capacity sensoring devices.

The peculiarities of data streams that require consideration include [35]:

- *Transiency*: the significance of each data point diminishes over time; therefore, it should be processed promptly upon measurement.
- *Time dependency*: each data point is linked to a timestamp, which must be taken into account either as an attribute or in the order of arrival. In both scenarios, a data point is assessed in comparison to other points within the same temporal context.
- *Infinity*: as measurements are continuously generated, data streams constitute theoretically infinite sequences of samples and, therefore, cannot be stored in memory entirely, particularly in low memory sensors. Thus methods

should opt for a summary of the dataset rather than attempting to store the entire sequence.
- *Arrival rate*: the arrival rate may vary over time, but it is imperative to process points immediately upon measurement. Therefore, the algorithm's execution time must be sufficiently brief. In the event of a variable arrival rate, it might be necessary to adapt the process and be willing to compromise on accuracy.
- *Concept drift*: in many cases, data distribution is non-stationary[1] making outlier detection methods that assume a fixed distribution unsuitable.
- *Uncertainty*: In various application scenarios, measurements can be influenced by environmental disturbances. This justifies the use of outlier detection methods.
- *Multi-dimensionality*: while not exclusive to data streams, some challenges are associated with high dimensionality. In our work, we concentrate on problems that are low-dimensional yet multi-dimensional.
- *Embeddedness*: an additional consideration is related to the concept of edge computing. In various samples, especially within wireless sensor networks, computing capabilities are integrated into objects with limited capacities, including memory and CPU.

In this context, there is a demand for approaches that exhibit the following characteristics, such as DyCF and DyCG proposed in this paper :

- frugality allowing to embed outlier detection models in devices,
- fast update to match incoming measurement frequency,
- little or no fine-tuning to meet automation and generalization needs,
- explainability and interpretability so that human operators understand the results easily.

This being said, these properties exclude deep learning methods.

[1]Non-stationary distributions have means, variances, and covariances that change over time. Non-stationary behaviors can be trends, cycles, random walks, or combinations of the three.

3 Related Work

Outlier detection has been a research subject for a long time in different communities, starting with statisticians and the works of Edgeworth in the end of the 19^{th} century [11]. With more than a century of interest in outlier detection, a lot of different methods have been proposed and a significant number of surveys tackle the task of listing, describing, categorizing and comparing these methods, e.g., [4, 6, 34, 44].

Depending on the context, outlier detection methods are usually separated into three groups: 1) supervised models that rely on the availability of datasets labeled with the outlierness status of samples, 2) semi-supervised methods that rely on datasets in which only normal samples are labeled, 3) unsupervised methods that can accept datasets without any information on outlierness. Unsupervised methods are recognized to be less precise than supervised methods due to the absence of information. However, as mentioned earlier, a limitation of supervised methods in the case of data streams is the potential obsolescence of labels resulting from distribution changes. Consequently, unsupervised methods become the sole option when dealing with data streams. For this reason, extensive research has been conducted on outlier detection methods for data streams. The reader can refer to surveys that concentrate on specific techniques [36, 41, 42], or those that survey the advancements of the field [40, 44, 46].

Initially considered, it seems interesting to adapt time series methods [10], for example ARIMA models [2], prediction models based on exponential smoothing [17] and LSTM (Long Short-Term Memory) [26]. These methods employ trends and seasonal patterns to forecast future data points from past observations. Anomaly detection can then be based on comparing forecasted points to actual measurements. However, these methods are not suitable for data streams because the learned model fails to evolve with new incoming measurements. While trend and seasonality can bring about alterations in the distribution, these changes must follow a regular pattern for models to make accurate predictions, and this regularity is not guaranteed in the context of data streams.

The three main families of outlier detection methods for data streams are methods based

on dynamic clustering, those relying on nearest neighbors (kNN) logic, and statistical methods. A common strategy for making the two latter methods applicable to data streams involves the utilization of windowing techniques. Data windows retain a constant number of points, capturing the current temporal context and distribution. This effectively addresses the necessities for *transiency*, *infinity*, and *concept drift*. Four windowing techniques are known [36]:

- *Landmark windows* set a point as a landmark and process data between this point and the current data point.
- *Sliding windows* process the last W data points, W being the size of the window.
- *Damped windows* consider all the points but each point is assigned a diminishing weight corresponding to its age.
- *Adaptive windows* are like sliding windows but their size varies with the speed evolution of points; the faster the distribution changes, the smaller the window.

Note that simply combining static methods with windowing techniques often proves inefficient. Many methods encounter challenges when dealing with swift model updates because they often require large window sizes to achieve satisfying results. This goes hand in hand with the fact that they are not engineered to be updated, necessitating the computation of a new model for each subsequent window, a process that can be time-consuming.

Dynamic clustering

Clustering methods group samples in space according to some similarity criterion and have been used to detect outliers based on one of the following assumptions [6]:

- "normal samples belong to clusters while outliers do not" in the case where the method includes a rejection mechanism,
- "normal samples are close to their closest centroïd (center of cluster) while outliers are far" in the case where the method assigns all samples to clusters,
- "dense clusters are normal and sparse clusters are outlying".

To adapt to data streams, dynamic clustering methods make statistical properties of clusters or micro-clusters to evolve through time [1, 33, 48]. Their main advantage is that they tackle the *notion of infinity* since it is not necessary to keep all the dataset in memory. However, they are often criticized because they have not been developed for outlier detection purposes but mainly for clustering [41].

Methods relying on kNN

Many methods for data streams consider outliers through the k nearest neighbors (kNN) principle. These methods can be divided into two groups:

- Methods for detecting outliers define an outlier as a sample with at most a proportion r of points within a certain distance D, which can be thought of as having at most k neighbors within a distance d or being no farther than d from the k-th nearest neighbor [20]. These methods employ windowing techniques to reduce the number of samples stored in memory and use specialized data structures for efficient addition, removal, and kNN searches. Among these methods, a study by [42] finds that MCOD [21] is the most efficient, although it has a limitation related to window size dependency.
- Methods adapting the well-known LocalOutlierFactor (LOF) algorithm [5]. The LOF is a measure of how local density of a sample compares to local density of its neighbors. On the addition of a new sample, the incremental LOF (iLOF) uses the fact that only a fixed number of samples need to be updated to reduce the computational complexity [32]. However, the required search for kNN and reversed kNN remains costly, which explains that several methods have proposed to approximate the LOF measure [16, 18, 29, 37].

Statistical methods

Statistical methods make the hypothesis that data samples are generated by a statistical distribution and that outliers belong to areas of low probability [46].

Parametric methods are unsuitable to non stationary distributions for they make the hypothesis of a predefined distribution and estimate its parameters. However, the method SmartSifter [45]

is worth mentioning as a statistical method offering both parametric and non-parametric solutions, showing better results with its parametric version. Its main advantage is that it is able to deal with categorical and continuous variables. The parametric version of SmartSifter uses Gaussian Mixture Models (GMM).

On the non parametric side, histogram construction is a candidate in univariate settings. The number of elements falling in a cell of the histogram reflects the probability of a sample falling into this cell. An evident benefit is the ease with which new data points can be seamlessly incorporated into the model. Quantile sketches are also worth mentioning as an optimal solution for the resolution of this problem in the context of data streams [19, 49]. Interestingly, quantile sketches can be approximated based on moments [12], which, as we will see, are also at the heart of the proposed methods.

To address multivariate scenarios, it's a common practice to construct individual histograms for each variable and subsequently compute a score by aggregating the scores from these separate histograms, a technique employed by HBOS (Histogram-based Outlier Score) as described in Goldstein's work [14]. However, this approach encounters limitations in high-dimensional contexts, as it fails to consider the interdependencies between variables.

A more advanced solution is given by *Kernel Density Estimation (KDE)* methods, also known as Parzen-Rosenblatt methods [31]. KDE (Kernel Density Estimation) shares similarities with histogram construction but incorporates a concept of continuity, offering an approximation of the probability density function (pdf). In the univariate case [31], the estimator of the density function f of n samples $\mathcal{X} = \{\mathbf{x}_i, i = 1, \ldots, n\}$ issued from the theoretical measure μ is $\tilde{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{K}_h(\mathbf{x} - \mathbf{x}_i)$, where $\mathbf{K}_h(u) = \frac{1}{h}\mathbf{K}(\frac{u}{h})$, $\mathbf{K}$ being the kernel function [2] and h being the bandwidth parameter that affects the influence area of each sample, or in other words, the smoothness of the function. Multivariate KDE (KDE) uses multivariate kernels $\mathbf{K_H}(u) = |\mathbf{H}|^{-1/2}\mathbf{K}(\mathbf{H}^{-1/2}u)$, where $\mathbf{H}$ is a symmetric positive definite $p \times p$ bandwidth matrix [23, §2.3.1]. KDE methods give a better approximation than histograms and are able to deal with multivariate cases although their complexity raises quickly with the amount of variables. The KDE based method proposed in [22] applies to data streams, as well as the non-parametric version of SmartSifter [45].

Contributions of the proposed CF based methods

The methods that we propose in this paper, namely DyCF and DyCG, can be positioned as statistical methods. The CF indeed captures the statistics of the dataset. Among the methods discussed in this section, KDE methods are undoubtedly the most closely related. However, the CF introduces a distinct perspective compared to KDE, as it identifies the theoretical probability measure of a set of samples using the statistical moments.

DyCF and DyCG advance the state of the art and bring contributions in three directions:

- they are based on solid theoretical foundations as they inherit the proven properties of the CF,
- they satisfy all data stream requirements, in particular they achieve fast model update on the arrival of new samples while retaining memory of past data,
- they require very little tuning, i.e. only one hyperparameter for DyCF, or no tuning at all for DyCG, hence avoiding the painful and tedious tuning phase required by the state of the art methods.

4 The Christoffel function for outlier detection

Prior to this section, we provide, in Table 1, a list of mathematical notations used for characterizing the Christoffel function.

4.1 Main properties of the Christoffel function

The Christoffel-Darboux Kernel (CD-Kernel) and the associated Christoffel function (CF) are well-known tools from the theory of approximation and orthogonal polynomials. Although they have been largely ignored in analysis of discrete data, recent results show that some peculiar properties of the CF can be valuable [24, 25].

[2] The kernel function is often chosen as Gaussian or as the Epanechnikov one.

Notation	Description
μ	A measure with support $\Omega \subset \mathbb{R}^p$
Ω	Support of μ
p	Dimension of the support Ω
d	Parameter of the Christoffel function
Λ_d^μ	Christoffel function with degree d
$Q_{\mu,d}$	Scoring function based on Λ_d^μ
Ω_γ	Level set $\Omega_\gamma := \{\mathbf{x} : \Lambda_d^\mu(\mathbf{x})^{-1} \leq \gamma\}$
$\gamma_{d,p}$	Define a level set $\Omega_{\gamma_{d,p}}$ with $\gamma_{d,p} = d^{3p/2}$
$\mathbf{v}_d(X)$	Vector of monomials of degree less than d
$s_p(d)$	Size of $\mathbf{v}_d(X)$, equals to $\binom{p+d}{d}$
$y_\alpha(\mu)$	Moment α of μ
$M_d(\mu)$	Matrix of moments of size $s_p(d) \times s_p(d)$
$\mathcal{X}$	Set of n observations from μ
μ_n	The empirical measure supported by $\mathcal{X}$

Table 1: Table of notations

The CD-Kernel and the CF are associated with a measure μ with support $\Omega \subset \mathbb{R}^p$, usually compact with nonempty interior, empirically represented by the set of available p-variate points. They also depend on a parameter d defining the highest degree of monomials that index the moment matrix of the measure μ and is involved in the definition of the CF.

The CF is hence denoted Λ_d^μ, parameterized by the measure μ and by the degree d. One of its main and salient features is its ability to encode the support Ω. In particular, for dimensions $p = 2$ or $p = 3$, one observes that the level set $\Omega_\gamma := \{\mathbf{x} : \Lambda_d^\mu(\mathbf{x})^{-1} \leq \gamma\}$, defined for some $\gamma \in \mathbb{R}_+$, captures the geometric shape of Ω quite accurately, even for low degrees d. Used as a tuning parameter, *the integer d gives a trade off between regularity (with small values of d) and fitness (with higher values) of the shape.*

As presented formally in section 4.2 and given a measure μ, the associated CF is obtained from the moment matrix of μ. Now, moments serve to quantify three essential parameters of distributions: location, shape and scale. The location of a distribution pertains to the position of its center of mass. Scale, on the other hand, denotes the extent to which a distribution is spread out, with the scale factor influencing the stretching or compression of the distribution. Lastly, the shape of a distribution encompasses its overall geometry, including characteristics such as bimodality, asymmetry, and heavy-tailedness. Consequently, the first moment delineates a distribution's location, the second moment characterizes its scale, and higher moments collectively elucidate its shape. The CF inherits this knowledge through the moment matrix, which intuitively explains why it can be a powerful tool for data analysis.

Previous works [24, 25] have shown how some of the CF's key properties can be helpful to address important problems like density approximation, support inference and outlier detection, where the measure of interest is now a *discrete measure* μ_n whose support is a finite set (or "cloud") of n data points (or samples) sampled from μ.

When going from μ on Ω to the empirical measure μ_n on the data set of n samples, it is important to relate n and d so that the empirical Christoffel function $\Lambda_d^{\mu_n}$ captures essential features of the population. For fixed d, the fact that $\Lambda_d^{\mu_n}$ and Λ_d^μ share the same properties is essentially dictated by the *Strong Law of Large Numbers*; see e.g. [25] (§6.2), and so it is sufficient that n is large enough compared to d, which is often the case in practice for small d. When d increases, the condition relating the sample size n and the degree d for $\Lambda_d^{\mu_n}$ to be close to Λ_d^μ is proven in [25] (§6.3). In [43] and [25] (Corollary 6.3.2), one can find a recipe to choose n and d in an appropriate manner.

On top of that, note that having n large enough is not an issue regarding computational complexity since the empirical CF, as defined later in equations (5-7), does not depend on the size of the dataset but solely on the number of variables p and the degree d.

4.2 Formal definition of the Christoffel function

Let $X = (X_1, X_2, ..., X_p) \in \mathbb{R}^p$ and let $\alpha = (\alpha_i)_{i=1...p} \in \mathbb{N}^p$ be the vector of exponents (degrees) associated to each variable for the monomial $X^\alpha := X_1^{\alpha_1} X_2^{\alpha_2} ... X_p^{\alpha_p}$ of total degree $\sum_{i=1}^p \alpha_i$. Let $\mathbf{v}_d(X)$ be the vector of all monomials of degree less than or equal to d in the *graded lexicographic order*[3]. The size of the vector $\mathbf{v}_d(X)$, denoted $s_p(d)$, depends on p and d and is equal to $\binom{p+d}{d}$.

As defined in [25], given a finite Borel probability measure μ on a compact set $\Omega \subset \mathbb{R}^p$ with

[3] Graded lexicographic order means: 1) ordered according to ascending monomial degree and then 2) using lexicographic order on variables considering $X_1 = a$, $X_2 = b$, etc.

nonempty interior, its moment matrix $M_d(\mu)$ is a real symmetric matrix with rows and columns indexed by the monomials of $\mathbf{v}_d(X)$. More precisely, letting

$$y_\alpha(\mu) := \int_{\mathbb{R}^p} \mathbf{x}^\alpha d\mu(\mathbf{x}) \tag{1}$$

be the moment α of μ, this means that the element of the matrix, at row indexed by $\alpha = (\alpha_i)_{i=1\dots p}$ and column indexed by $\beta = (\beta_i)_{i=1\dots p}$, is $y_{\alpha+\beta}(\mu) = \int_{\mathbb{R}^p} \mathbf{x}^{\alpha+\beta} d\mu(\mathbf{x})$ with the notation $(\alpha+\beta) = (\alpha_i + \beta_i)_{i=1\dots p}$. For sample, in the case of $p = 2$ and $d = 2$ and denoting $y_\alpha = y_\alpha(\mu)$, the moment matrix is given by

$M_2(\mu):$

	1	X_1	X_2	X_1^2	X_1X_2	X_2^2
1	1	$y_{1,0}$	$y_{0,1}$	$y_{2,0}$	$y_{1,1}$	$y_{0,2}$
X_1	$y_{1,0}$	$y_{2,0}$	$y_{1,1}$	$y_{3,0}$	$y_{2,1}$	$y_{1,2}$
X_2	$y_{0,1}$	$y_{1,1}$	$y_{0,2}$	$y_{2,1}$	$y_{1,2}$	$y_{0,3}$
X_1^2	$y_{2,0}$	$y_{3,0}$	$y_{2,1}$	$y_{4,0}$	$y_{3,1}$	$y_{2,2}$
X_1X_2	$y_{1,1}$	$y_{2,1}$	$y_{1,2}$	$y_{3,1}$	$y_{2,2}$	$y_{1,3}$
X_2^2	$y_{0,2}$	$y_{1,2}$	$y_{0,3}$	$y_{2,2}$	$y_{1,3}$	$y_{0,4}$

$M_d(\mu)$ can also be written as

$$M_d(\mu) = \int_{\mathbb{R}^p} \mathbf{v}_d(\mathbf{x})^T \mathbf{v}_d(\mathbf{x}) d\mu(\mathbf{x}), \tag{2}$$

where the integral is understood elementwise. Note that $M_d(\mu)$ is positive definite for any d, i.e., $\mathbf{p}^T M_d(\mu)\mathbf{p} > 0$ for every $0 \neq \mathbf{p} \in \mathbb{R}^p$, and therefore $M_d(\mu)$ is non singular.

The CD-Kernel K_d^μ associated with μ is defined by

$$(\mathbf{x}, \mathbf{y}) \mapsto K_d^\mu(\mathbf{x}, \mathbf{y}) := \mathbf{v}_d(\mathbf{x})^T M_d(\mu)^{-1} \mathbf{v}_d(\mathbf{y}), \tag{3}$$

while the polynomial $Q_{\mu,d}$ reads

$$Q_{\mu,d}(\mathbf{x}) := K_d^\mu(\mathbf{x}, \mathbf{x}) = \mathbf{v}_d(\mathbf{x})^T M_d(\mu)^{-1} \mathbf{v}_d(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n. \tag{4}$$

$Q_{\mu,d}$ is a sum-of-squares polynomial of degree $2d$ and the CF $\Lambda_d^\mu(\mathbf{x})$ is then defined by

$$\mathbf{x} \mapsto \Lambda_d^\mu(\mathbf{x})^{-1} := Q_{\mu,d}(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{R}^n. \tag{5}$$

4.3 Outlier scoring with the Christoffel Function

In practical applications of outlier detection, only an *empirical* moment matrix is available, associated with a discrete measure μ_n whose support is a set of n observations $\mathcal{X} = \{\mathbf{x}_i, i = 1, \dots, n\}$ sampled from a theoretical distribution μ. In this case, the empirical version of equations (1) and (2) respectively read

$$y_\alpha(\mu_n) = \frac{1}{n} \sum_{x \in \mathcal{X}} \mathbf{x}^\alpha, \tag{6}$$

and

$$M_d(\mu_n) = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{v}_d(\mathbf{x})^T \mathbf{v}_d(\mathbf{x}). \tag{7}$$

Note that, considering $\mathcal{X}$ as a dataset, $M_d(\mu_n)$ can be seen as a summary or an encoding of this dataset. This property is very interesting because it avoids keeping in memory all the samples, which is definitively unacceptable when dealing with data streams (see the *"Infinity"* peculiarity of data streams in Section 3).

Given a cloud of points $(\mathbf{x}_i)_{i \leq n}$ sampled from a theoretical measure μ, the ability to capture the geometric shape of the support of the empirical measure μ_n comes with one valuable property of $\Lambda_d^\mu(\mathbf{x})^{-1}$. It has indeed been shown that, under some assumptions, the samples belonging to the support are confined by a specific level set $\Omega_{\gamma_{d,p}}$, where $\gamma_{d,p} = Cd^{3p/2}$ and C is a problem-related constant [25](Theorem 7.3.3). This level set will be used in the following sections, setting $C = 1$.

As a matter of fact, the level sets of $\Lambda_d^\mu(\mathbf{x})^{-1}$ match the density variations of the cloud of points $(\mathbf{x}_i)_{i \leq n}$, as shown in the illustrative example below, making of $\Lambda_d^\mu(\mathbf{x})^{-1}$ a good scoring function for outlier detection.

Additionally, the model is contained in the moments matrix $M_d(\mu_n)$ of size $s_p(d) \times s_p(d)$, that does not depend on n, thereby fixing the memory size. The computational complexity is also limited since $\Lambda_d^\mu(\mathbf{x})^{-1}$ only requires computing $\mathbf{v}_d(\mathbf{x})$, a vector of size $s_p(d)$, and $\mathbf{v}_d(\mathbf{x})^T M_d(\mu_n)^{-1} \mathbf{v}_d(\mathbf{x})$. This being, this implies that the complexity and memory size growths are essentially exponential with d or p, limiting the application to low dimensions and low degrees.

Illustrative example – In order to illustrate the behavior of the scoring function obtained,

Figure 1 compares scores from the CF with d=6 (Figure 1(a)) and scores obtained with KDE using the multivariate gaussian kernel (Figure 1(b)) on a dataset characterized by multiple densities. It consists of two clusters with different distributions; one is dense with 5000 samples circumscribed in a small circle and the other is sparse with 1000 samples circumscribed in a larger circle. On top of that, 50 points acting like outliers are sampled from a uniform distribution with its support around the two disks.

Fig. 1 clearly shows firsthand that the level sets generated by CF smoothly surround the cloud of points and some nicely capture the two clusters. On the other hand, the level sets generated by KDE do not capture precisely the dense cluster. In addition, the level set that captures at best this cluster rejects entirely the sparse cluster.

For a more rational evaluation, Table 2 considers the metrics AUROC, i.e., sensitivity (True Positive rate) versus specificity (False Positive Rate), and AP (Average Precision) approximating AUPRC, i.e., precision versus recall, that are recommended by [34] (section VII.B) for evaluating classification methods globally, independently of their tuning. Both scores are higher for CF. The results hence reinforce what is suggested by visual inspection of Fig. 1, i.e., that CF is better at capturing the support of the cloud of points for this multi-density dataset.

Finally, note that the red thicker level set of CF, which nicely capture the support of the measure, corresponds to $\Omega_{\gamma_{d,p}}$ with $\gamma_{d,p} = d^{3p/2}$ dictated by the CF theory.

Method	AUROC	AP
CF	0.9644	0.7250
KDE	0.9372	0.6042

Table 2: AUROC and AP results obtained for CF and KDE on the two disks dataset

5 Adapting the Christoffel Function to data streams

5.1 Fast Model Update

Most of the peculiarities of data streams listed in Section 3, like *transiency*, *infinity*, *arrival rate*, and *embeddedness*, boil down to requiring an efficient low computation and low memory incremental method.

From the computational point of view, interestingly the CF complexity does not depend on the number of points but is essentially exponential in the number of dimensions p and the chosen degree d. CF is hence expected to be competitive in low dimensions and for relatively small degrees.

From the memory point of view, the infinity of data streams is accounted by the use of the moment matrix $M_d(\mu_n)$ which contains the statistics of all the points without need to keep them in memory.

The capital gain of DyCF is incrementality and the ability of dealing with *concept drift*, i.e., to update the model so that it follows any change in the distribution. The moment matrix given by Equation (7) can be rewritten with the incremental formula

$$M_d(\mu_{n+1}) = \frac{1}{n+1}[nM_d(\mu_n) + \mathbf{v}_d(\mathbf{x}_{n+1})\mathbf{v}_d(\mathbf{x}_{n+1})^T]. \quad (8)$$

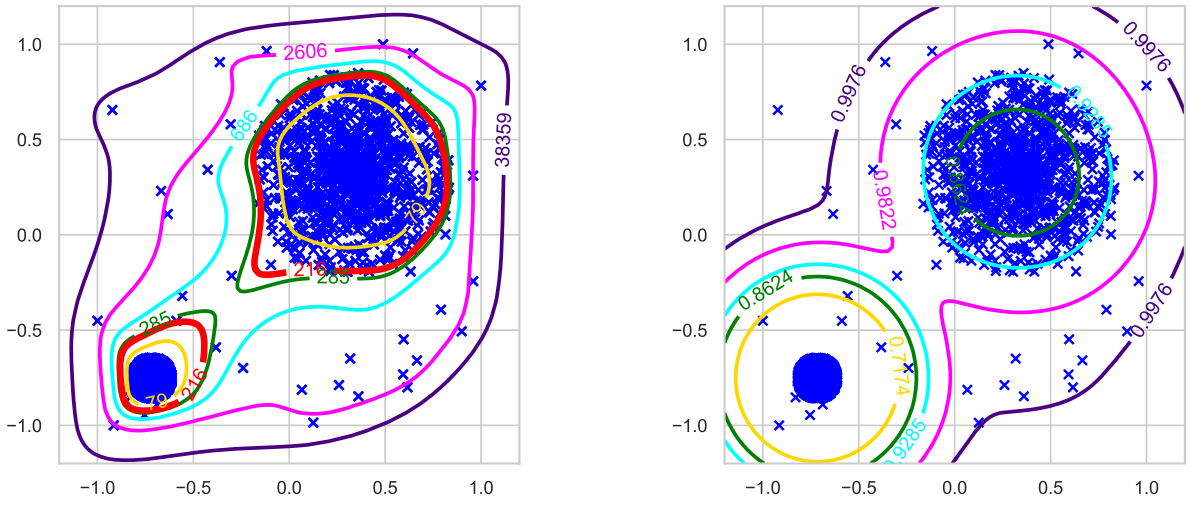
The CF outlier score given by Equations (3) and (4) requires to invert the moment matrix. Interestingly, the Sherman-Morrison formula provides an incremental way to invert a matrix of the form (8) as follows

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u} \quad (9)$$

Considering $A = nM_d(\mu_n)$ and $u = v = \mathbf{v}_d(\mathbf{x}_{n+1})$, this leads to

$$((n+1)M_d(\mu_{n+1}))^{-1} = (nM_d(\mu_n))^{-1} - \frac{(nM_d(\mu_n))^{-1}\mathbf{v}_d(\mathbf{x}_{n+1})\mathbf{v}_d(\mathbf{x}_{n+1})^T(nM_d(\mu_n))^{-1}}{1 + \mathbf{v}_d(\mathbf{x}_{n+1})^T(nM_d(\mu_n))^{-1}\mathbf{v}_d(\mathbf{x}_{n+1})} \quad (10)$$

Equation (10) can be used to compute the inverse CF $\Lambda_d^\mu(\mathbf{x})^{-1}$ in an incremental way, defining the proposed Dynamic Christoffel Function method named DyCF.



(a) CF levelsets

(b) KDE levelsets

Fig. 1: Level sets obtained for a dataset characterized by two disks and uneven densities with (a) CF using $d = 6$ and (b) KDE (graph obtained using PYTHON matplotlib library).

It is important to note that DyCF requires only one parameter to be chosen, which is the degree d. The theory then dictates to use the level set defined by $\Omega_{\gamma_{d,p}}$. The DyCF scoring function is hence defined as $\Lambda_d^\mu(\mathbf{x})^{-1}$ normalized by $\gamma_{d,p}$

$$S_{d,p}(\mathbf{x}) = \frac{\Lambda_d^\mu(\mathbf{x})^{-1}}{\gamma_{d,p}}, \tag{11}$$

from which a point $\mathbf{x}$ is defined as an outlier if $S_{d,p}(\mathbf{x}) \geq 1$.

5.2 Tuning Free

Tuning-free is a highly desirable property that can be considered the holy grail in machine learning. Yet, as far as we know, it is not achieved by any outlier detection method. Interestingly, the evolution of the CF score, obtained for different values of d, has been theoretically characterized. The proposed Dynamic Chistoffel Growth method, named DyCG, leverages this property to achieve an efficient tuning-free method.

For $\mathbf{x} \in \mathbb{R}^p$, fixed, the evolution of $\Lambda_d^\mu(\mathbf{x})^{-1}$ as d increases depends critically on whether $\mathbf{x}$ is in the support of μ or not. More precisely, for every $\mathbf{x} \notin \Omega$, the function $\mathbf{x} \mapsto \Lambda_d^\mu(\mathbf{x})^{-1}$ grows *exponentially fast* with d, while its growth is *at most polynomial* for $\mathbf{x} \in \Omega$.

The distinguishing property of exponential growth with d for $\mathbf{x}$ outside the support of the measure is quantified by Theorem 5.2.1.

Theorem 5.2.1. *([25] Lemma 4.3.1 p.50) Let μ be a positive Borel measure supported on the compact set $\Omega \subset \mathbb{R}^p$, and let $\mathbf{x} \notin \Omega$ and $\delta > 0$ be such that $\mathrm{dist}(\mathbf{x}, \Omega) > \delta$. Then*

$$\Lambda_d^\mu(\mathbf{x})^{-1} \;\geq\; s_p(d) 2^{\frac{\delta d}{\delta + \mathrm{diam}(\Omega)} - 3} d^{-p} \left(\frac{p}{e}\right)^p e^{-p^2/d}.$$

At the same time, the magnitude of the CF score for points inside the support is at most polynomial in d for p fixed according to Theorem 5.2.2.

Theorem 5.2.2. *([25] Lemma 4.3.2 p.51) Let μ be a positive Borel measure supported on the compact set $\Omega \subset \mathbb{R}^p$, the closure of a bounded domain U with nice boundary, and let $\mathbf{x} \in U$ and $\delta > 0$ be such that $\mathrm{dist}(\mathbf{x}, \partial U) \geq \delta$. Then*

$$\Lambda_d^\mu(\mathbf{x})^{-1} \;\leq\; s_p(d) \frac{C_p}{\delta^p} (1+p)^3,$$

where C_p does not depend on d but only on p.

Based on the asymptotic results of Theorem 5.2.1 and Theorem 5.2.2, DyCG is designed to assess the outlierness of a point based on two DyCF models of degrees d_{min} and d_{max}. d_{min} is naturally taken equal to 2 and

d_{max} is taken equal to 6 to make the problem tractable and can be reduced according to the available memory. The score $S_{d,p}(\mathbf{x})$ defined in Section 5.1 is used for both models. This way, if $\Lambda_d^{\mu}(\mathbf{x})^{-1}$ follows a growth in $d^{3p/2}$ at least, then $S_{d_{max},p}(\mathbf{x}) \geq S_{d_{min},p}(\mathbf{x})$. The DyCG scoring function is hence defined as

$$S'_{d_{max},d_{min},p}(\mathbf{x}) = \frac{S_{d_{max},p}(\mathbf{x}) - S_{d_{min},p}(\mathbf{x})}{d_{max} - d_{min}}, \tag{12}$$

and a sample $\mathbf{x}$ is considered outlying if $S'_{d_{max},d_{min},p}(\mathbf{x}) \geq 0$.

Note that DyCG requires to maintain two DyCF models simultaneously, which leads to an increase in memory use. Nevertheless, because DyCG is based on the evolution of the score, the value of the degrees d_{min} and d_{max} of the two models can be fixed once and for all, making of DyCG a tuning-free method.

6 Evaluation

6.1 Process description

To assess the effectiveness of the two proposed methods, an evaluation procedure is delineated in this section. This evaluation involves examining two types of data streams: synthetic data streams with labeled data, and real-world data streams without labels. All data streams can be found in the Git repository featuring our experiments[4].

6.1.1 Synthetic data streams

Using Markov chain logic, synthetic data streams simulating multi-modal behaviors are constructed. Modes are specified in a configuration file, with parameters indicating whether they follow a normal or uniform distribution. Transitions between modes are then defined, with assigned probabilities and shapes (logarithmic, linear, exponential). Outliers are generated using a similar process. Two types of outliers are considered:

- Type-I outliers are random values uniformly distributed around normal behaviors with a specified occurrence probability;
- Type-II outliers are defined as a short offset from normal behavior with appearing and lasting probabilities, enabling their persistence across successive measurements.

Three setups are employed for generating synthetic data streams. Two of them are bivariate, showcasing samples illustrated in Figures 2 and 3, while the third is trivariate in order to asses the effect of dimension on complexity. In each scenario, a behavioral alteration is introduced. This approach is intended to evaluate the model's capability to accommodate shifts from normal behavior. The alterations are as follows:

- in the first setup, a change in one mode's mean is implemented;
- in the second setup, an offset is applied to all data points;
- in the third setup, with three dimensions, a new mode is introduced at some point.

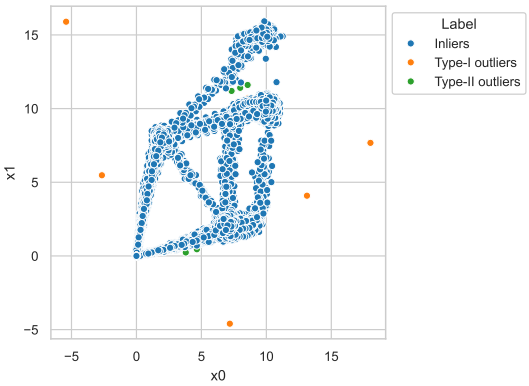


Fig. 2: Samples issued from the first synthetic data stream setup. Blue dots represent normal behavior, orange dots are type-I outliers, green dots are type-II outliers.

6.1.2 Real-world data streams

The real-world data streams originate from sensors installed on actual industrial luggage conveyor systems. The sensors specifically capture two physical variables: the speed of the conveyor belt and the intensity of the engine.

These data streams exhibit distinct characteristics, consisting of three primary operational modes with nonlinear transitions between them

[4]Code available on github [9].

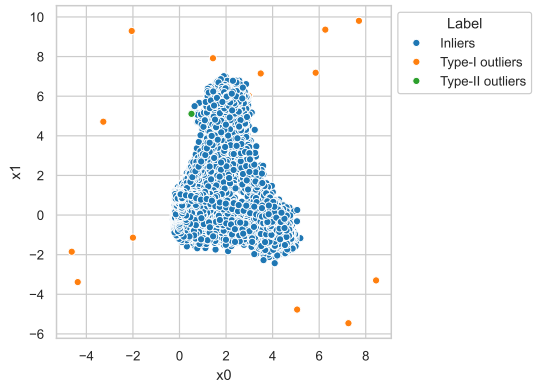


Fig. 3: Samples issued from second synthetic data stream setup. Blue dots represent normal behavior, orange dots are type-I outliers, green dots are type-II outliers.

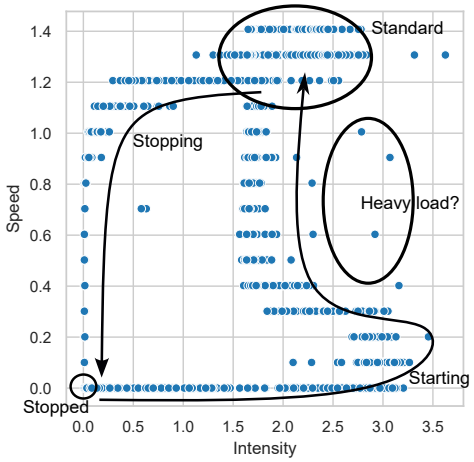


Fig. 4: Representation of a luggage_conveyor data stream. Operating modes and transitions are visible. (graph obtained using PYTHON matplotlib library and annotated manually)

(whose rationale guided the design of the afore mentioned synthetic datasets). The "stop" mode predominates, indicating the conveyor halted with both speed and intensity registering at zero. The "standard" mode reflects typical conveyor operation with nominal speed and intensity. An infrequent "heavy_load" mode is also discernible, characterized by reduced speed and increased intensity to accommodate heavy luggage. Furthermore, transitions occur between the three operational modes, such as an intensity peak followed by a speed increase at the conveyor's start, and a fast decrease in intensity compared to speed when the conveyor stops. Visual representations of the data acquired for the various modes are provided in Figure 4.

Five conveyors are considered with similar behaviors. All of them are working for seven successive days, with measurements issued every second (86400 samples per day). However, the data is sourced from wireless sensor networks and transmitted via radio transmissions with an unstable transmission frequency and potential packet losses. In this case, it is not critical as the exact measurement date is not considered (only the order of measurements is used). The packet loss rates during the measurement periods used for the five conveyors are respectively 11%, 2%, 3%, 5%, and 3%.

6.1.3 Evaluation process

The whole evaluation process is described in Figure 5. Data streams are organized in sub-data streams issued from different sources (different setups for the synthetic data streams and different conveyors for the real-world ones).

Synthetic data streams are composed of 200k points divided in 10 sub-data streams while conveyor data streams are each divided in 7 working days.

The process used to evaluate the performance of the methods on all sub-data streams is described in Figure 6. Sub-data streams are divided in an initialization set used to initialize models and an inference set used for evaluation. The initialization phase is described in Figure 7 while the inference phase is described in Figure 8.

Mean and standard deviation of all metrics are computed for each method and each data stream.

6.1.4 Evaluation metrics

Different metrics are used depending on the availability of labels:

- in the synthetic cases, labels are available and it is possible to use popular metrics such as AUROC and AP, already used in the illustration example of Section 4.3,

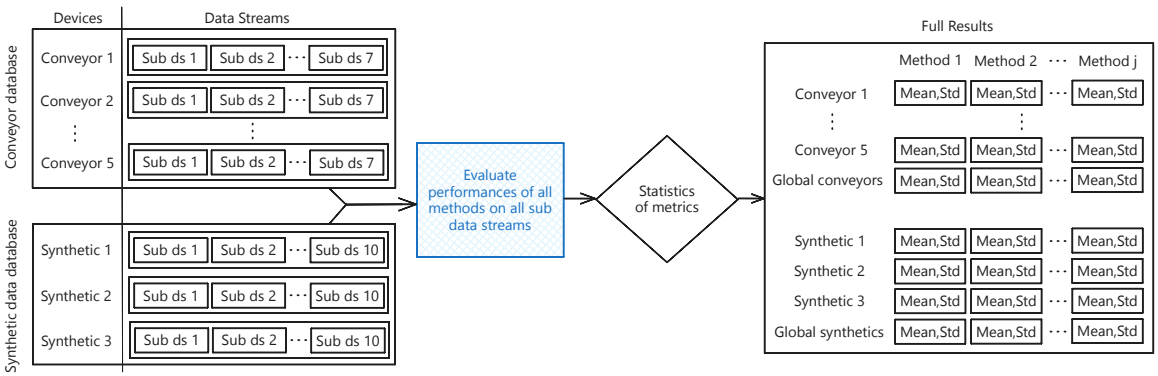


Fig. 5: Graph representing the full evaluation process.

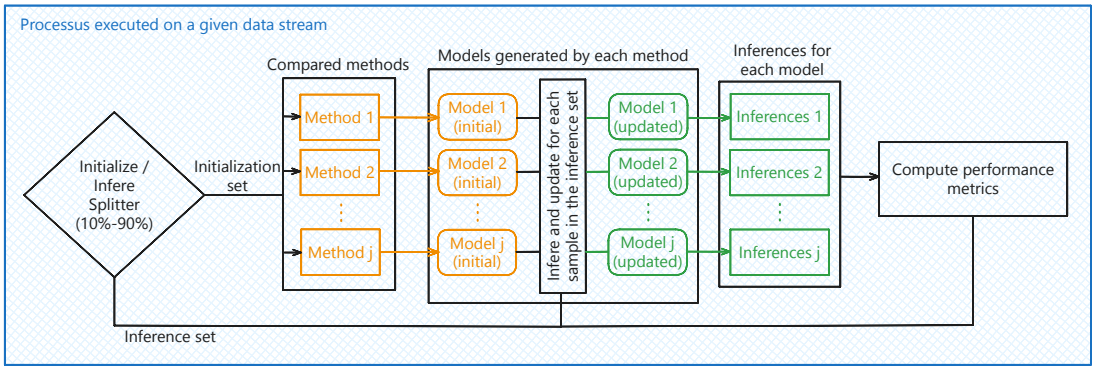


Fig. 6: Graph describing the process executed for each method.

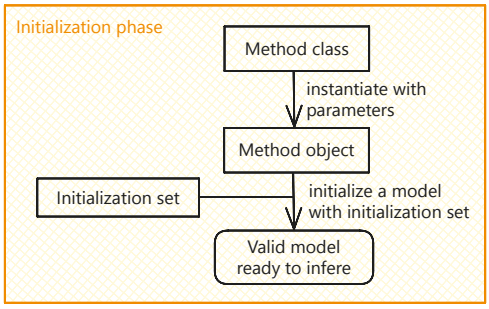


Fig. 7: Graph describing the initialization phase of a model.

- for conveyor cases, where no labels are available, AUROC and AP cannot be used. Instead, unsupervised metrics EM and MV [13], using the Excess-Mass and Mass-Volume curves respectively, are used.

No considered metric is threshold-sensitive, meaning that the choice of the threshold parameters does not impact the obtained score. Higher values of AUROC, AP, and EM are preferred, whereas lower values are sought for MV. It is important to note that EM and MV evaluate the extent to which a scoring function aligns with the statistical distribution of samples, which is not suitable for evaluating certain methods.

Finally, the average processing duration of a data point (computation of its outlierness score and model update) is computed to assess the speed of the methods, a characteristic highly esteemed in data stream contexts

6.2 Competing methods

The selected methods for comparison with DyCF and DyCG are all renowned outlier detection techniques for data streams. Each method has been re-implemented by us[5], with the exception of SmartSifter, which relies on the Python

[5]Code available on github [9].

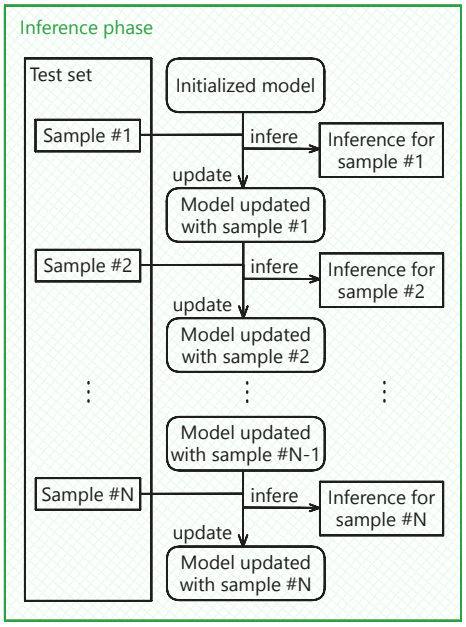


Fig. 8: Graph describing the inference phase of a model.

implementation found in [39]. Intensive parameter combinations have been tested to get the best out of the tuning for the comparison. The retained parameters are given in Table 3, outlining the number of parameters that need to be tuned for each method and pointing out the ease of use of DyCF and DyCG. Competing methods are commented below:

- **Kernel Density Estimation (KDE)** has been presented in Section 3 and illustrated in the example of Section 4.3 and in Fig. 1. In order to be applicable to data streams, a sliding window of the last arriving points is used. This approach aims to mitigate time complexity and memory usage. Because the bandwidth parameter **H** is set from the Scott's rule of thumb of [30, 38], there are only two parameters to tune, which are the size of the sliding window W (the number of points contained in the window) and the threshold on the score (or density estimate).
- **SmartSifter** is selected in its parametric version as presented in [45] and briefly in Section 3. In our experiments, likelihood was used as a scoring function. The different parameters to be tuned are the threshold on the score, the number k of gaussians in the GMM, a discounting parameter r and a stability parameter α.
- **Distance-based outliers using KDE (DBOKDE)** is derived from the kNN principle described in Section 3. To reduce the complexity of counting the elements in a neighborhood, the number of neighbors is estimated using kernel density estimation. This method has been proposed in [30].
- **Incremental Local Outlier Factor (iLOF)** as presented in Section 3, is implemented with R*-Trees [3] to reduce the kNN search complexity, as recommended in [32].[6]

Method	Parameters	Values
KDE	Threshold	Meaningless
	Window size	1000
	Kernel	Gaussian
	Bandwidth	Scott's rule
SmartSifter	Threshold	Meaningless
	Nb components	12
	Discounting parameter	1e-3
	Stability parameter	1.5
DBOKDE	Nb neighbors	Meaningless
	Search radius	0.1
	Window size	1000
	Kernel	Epanechnikov
	Bandwidth	Scott's rule
ILOF	Threshold	Meaningless
	Nb neighbors	10
	Window size	1000
	Min children	3
	Max children	12
	Reinsertion strategy	close
	Reinsertion tolerance	4
DyCF	Degree	6
	C (threshold-like)	Meaningless
DyCG	Degrees	(2, 6)

Table 3: Table of parameters used in the experiments.

[6] Note that iLOF was improved in [29] with the Density summarizing Incremental LOF (DILOF) that reduces, in theory, the complexity while maintaining accuracy. Note that this is only true with really small windows or if the deletion part of iLOF, that makes the use of sliding windows possible, is abandoned. Otherwise, DILOF is heavier than iLOF because of the "density summarizing part" that is executed every $\frac{W}{4}$ observations, W being the window length. For this reason, the comparison is done with iLOF solely.

6.3 Results

6.3.1 Synthetic data streams

The results obtained for the synthetic data streams are shown in Figures 9, 10 and 11 and values for each metric are given in Tables 4, 5 and 6.

DyCF demonstrates performance at least on par (close to the best) with the compared methods concerning AUROC and AP metrics. Conversely, DyCG exhibits slightly lower performance on AUROC, particularly in the case of three-dimensional data streams; however, it yields superior results in terms of AP.

Regarding the time metric, DyCF and DyCG outperform other methods when handling two-dimensional data streams, but SmartSifter is better with three-dimensional data streams. This is due to the dependence in p of DyCF and DyCG.

6.3.2 Real world data streams

The results obtained for the conveyor data streams are shown in Figures 12, 13, 14, 15 and 16 and values for each metric are given in Tables 7, 8 and 9.

In this case study, DyCF provides by far the best results in all categories. On the other hand, DyCG offers significantly lower performances. The poor performance of DyCG can be explained by the underlying properties of EM and MV metrics. As a reminder, these metrics, designed for unsupervised anomaly detection, evaluate the alignment of the scoring function with the statistical distribution of samples, which is not in line with the transformation used to obtain DyCG's scoring function.

Interestingly, despite KDE's scoring being based on density estimation, KDE also exhibits poor performance on both EM and MV metrics. DBOKDE, which utilizes KDE at its core for estimating the number of neighbors, outperforms KDE on EM and MV.

6.3.3 Discussion

On the evaluated data streams, DyCF achieves state-of-the-art results while being easier to tune and faster than most methods.

DyCG allows to make tuning even easier but at great costs on performance. However, it gives the best results with AP. On top of that, its scoring function suffers from the underlying concept of EM and MV evaluations, and none of the employed metrics rewards the fact that DyCG does not require to set a threshold on the score, which is obviously a great advantage.

Regarding time complexity, the dependency in p is noticeable between 2-dimensional datasets and the third synthetic setup which is 3-dimensional. To illustrate this further, in Figure 17 we plot two graphs: (1) the processing duration by DyCF with $d = 6$ of 500 data points drawn from uniform distributions of increasing dimensions p and (2) the size of the moments matrix, which is $s_d(p) \times s_d(p)$ as a function of p.

7 Conclusion and future work

The principles on which methods discriminate normal points from outliers are paramount since they condition robustness and bias.

In this article, two methods for unsupervised outlier detection in low dimensional data streams are proposed. Both leverage the properties of the Christoffel function and are built on solid theoretical foundations.

The first method, DyCF, only requires two parameters to be tuned, while the second, DyCG, is completely free of tuning requirements. In this sense, the two methods elegantly remove the painful step of tuning, which is all the more painful in the unsupervised case and for typical non-stationary distributions of data streams. DyCF and DyCG have also shown great execution time and memory use performances, which has been noted of paramount importance. DyCF surpasses most of the methods it has been compared to, utilizing both well-established supervised metrics and lesser-known unsupervised ones.

These promising results encourage us to continue the work to overcome two limitations that were identified during this study:

- *A numerical instability issue* has been observed for high values of d, i.e. for high degrees of the monomials that index the moment matrix. Actually, when the moment matrix has very small eigenvalues, some numerical instability occurs for its inversion. Future work will approach the problem in different ways and assess the impact on the accuracy of the resulting models:

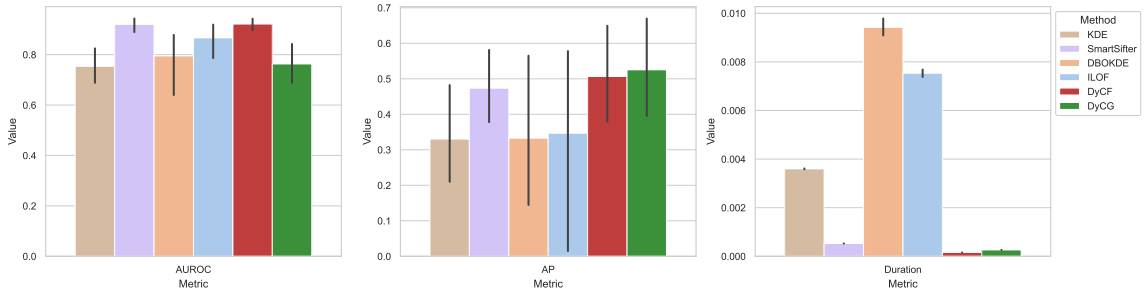


Fig. 9: Results for the synthetic data stream first setup.

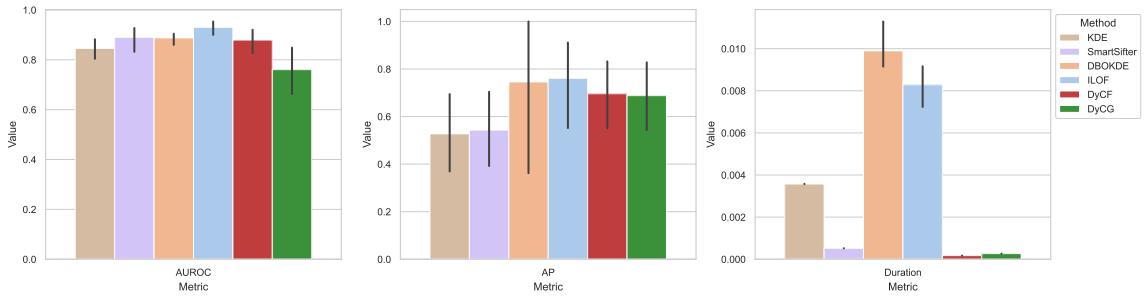


Fig. 10: Results for the synthetic data stream second setup.

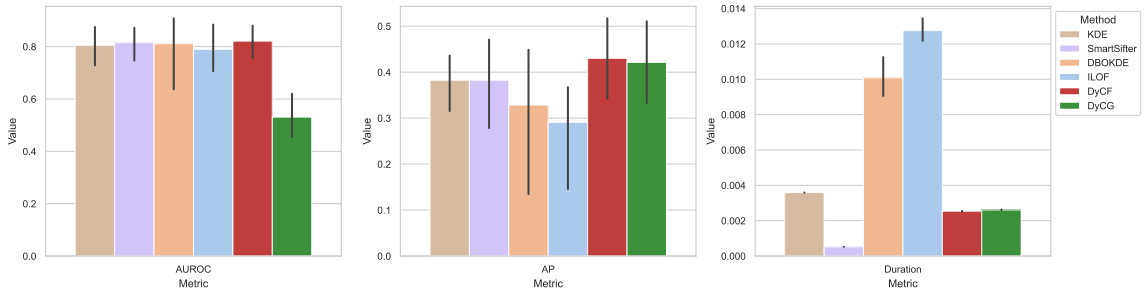


Fig. 11: Results for the synthetic data stream third setup.

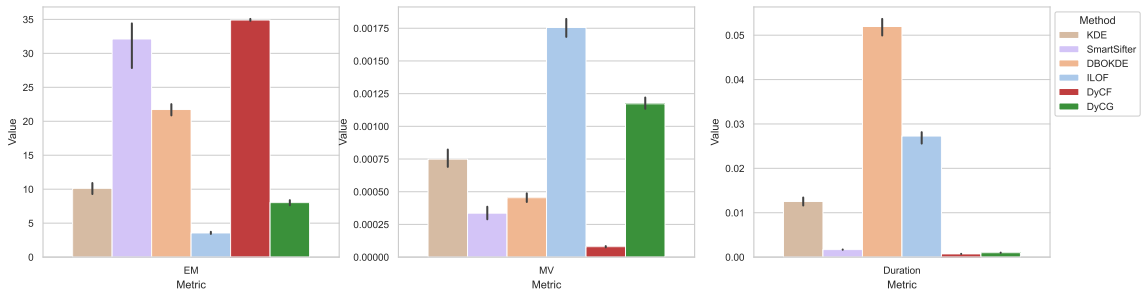


Fig. 12: Results for the first conveyor data stream (provided by sensor node MOTE-47).

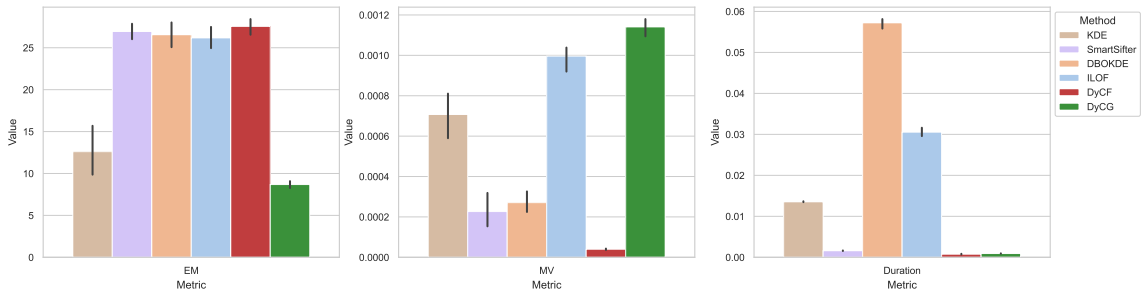


Fig. 13: Results for the second conveyor data stream (provided by sensor node MOTE-67).

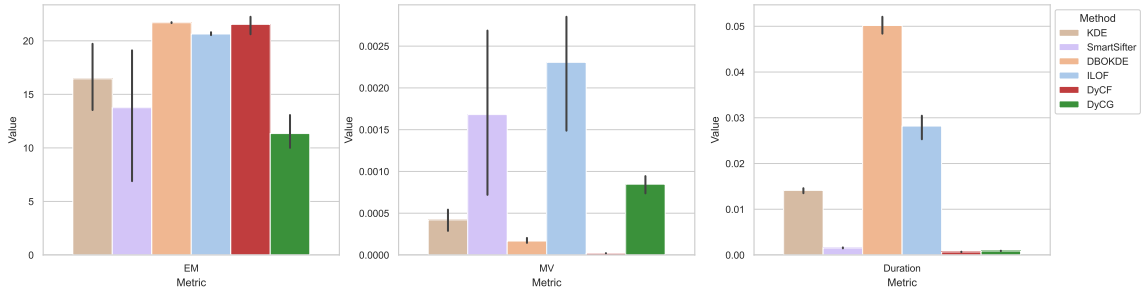


Fig. 14: Results for the third conveyor data stream (provided by sensor node MOTE-72).

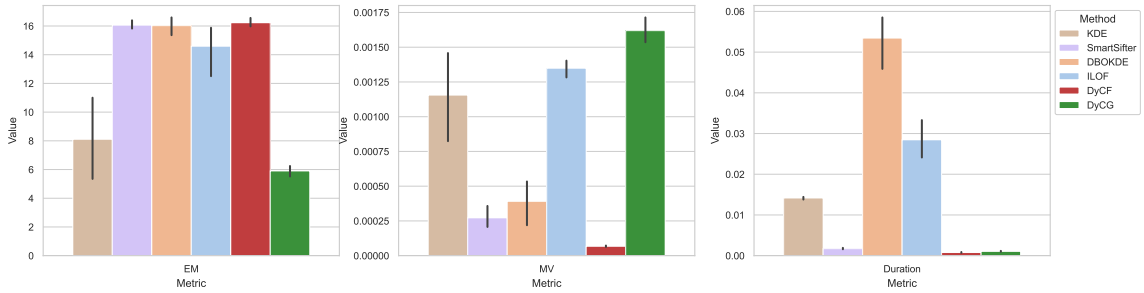


Fig. 15: Results for the fourth conveyor data stream (provided by sensor node MOTE-75).

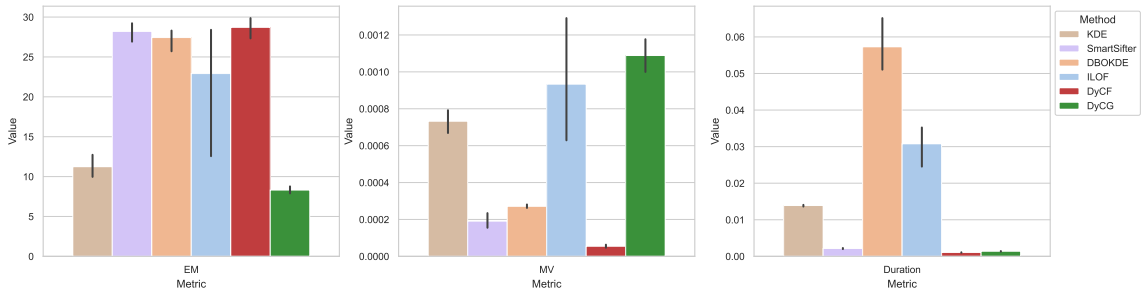


Fig. 16: Results for the fifth conveyor data stream (provided by sensor node MOTE-78).

Dataset	KDE	SmartSifter	DBOKDE	ILOF	DyCF	DyCG
Setup 1	0.754 (0.116)	0.920 (0.048)	0.795 (0.136)	0.867 (0.071)	**0.921 (0.042)**	0.763 (0.128)
Setup 2	0.846 (0.070)	0.890 (0.079)	0.888 (0.025)	**0.930 (0.028)**	0.879 (0.077)	0.761 (0.161)
Setup 3	0.805 (0.124)	0.815 (0.109)	0.811 (0.151)	0.790 (0.090)	**0.821 (0.111)**	0.531 (0.143)
Global	0.801 (0.109)	**0.875 (0.092)**	0.831 (0.111)	0.862 (0.085)	***0.874 (0.089)***	0.685 (0.178)

Table 4: AUROC mean (standard deviation in brackets) on synthetic data streams. Best value in bold and second best value in bold italic.

Dataset	KDE	SmartSifter	DBOKDE	ILOF	DyCF	DyCG
Setup 1	0.330 (0.230)	0.473 (0.174)	0.333 (0.214)	0.347 (0.295)	0.507 (0.237)	**0.525 (0.234)**
Setup 2	0.528 (0.293)	0.543 (0.272)	0.746 (0.338)	**0.761 (0.187)**	0.696 (0.245)	0.688 (0.253)
Setup 3	0.383 (0.105)	0.383 (0.161)	0.329 (0.169)	0.291 (0.125)	**0.430 (0.154)**	0.421 (0.154)
Global	0.413 (0.231)	0.466 (0.211)	0.469 (0.300)	0.466 (0.290)	***0.544 (0.237)***	**0.545 (0.238)**

Table 5: AP mean (standard deviation in brackets) on synthetic data streams. Best value in bold and second best value in bold italic.

Dataset	KDE	SmartSifter	DBOKDE	ILOF	DyCF	DyCG
Setup 1	3.60e-3	5.28e-4	9.42e-3	7.53e-3	**1.60e-4**	2.62e-4
Setup 2	3.57e-3	5.19e-4	9.89e-3	8.286e-3	**1.71e-4**	2.67e-4
Setup 3	3.59e-3	**5.21e-4**	1.01e-2	1.28e-2	2.54e-3	2.61e-3
Global	3.59e-3	**5.23e-4**	9.80e-3	9.52e-3	***9.56e-4***	1.05e-3

Table 6: Duration (in seconds per point) mean (standard deviation in brackets) on synthetic data streams. Best value in bold and second best value in bold italic.

Dataset	KDE	SmartSifter	DBOKDE	ILOF	DyCF	DyCG
Conveyor 1	10.1 (1.234)	32.1 (5.551)	21.7 (0.136)	3.56 (0.162)	**34.9 (0.224)**	8.02 (0.542)
Conveyor 2	12.6 (4.292)	26.9 (1.366)	26.6 (1.477)	26.2 (1.271)	**27.5 (1.404)**	8.68 (0.641)
Conveyor 3	16.5 (4.455)	13.8 (9.550)	**21.7 (0.047)**	20.6 (0.148)	21.5 (1.184)	11.3 (2.369)
Conveyor 4	8.11 (4.687)	16.1 (0.458)	16.0 (0.624)	14.6 (1.824)	**16.2 (0.441)**	5.90 (0.586)
Conveyor 5	11.2 (2.056)	28.2 (1.767)	27.4 (1.489)	22.9 (8.974)	**28.7 (1.932)**	8.31 (0.656)
Global	11.7 (4.435)	***23.4 (8.698)***	22.7 (4.328)	17.6 (8.956)	**25.8 (6.584)**	8.45 (2.088)

Table 7: EM mean (standard deviation in brackets) on conveyor data streams. Best value in bold and second best value in bold italic.

– slightly perturbing the moment matrix by adding the identity matrix times a factor that makes the order of the resulting smallest eigenvalue reasonable for numerical inversion. This is known to bring more numerical stability as proposed in [28] (Eq. (8), p. 401) under the name of "Tychonov regularization" .

– replacing monomials by other polynomial basis. The use of Chebyshev polynomials of first kind would, in theory, give more numerical stability to the moment matrix. Typically, in the basis of monomials, the univariate moment matrix is Hankel and its multivariate analogue has a Hankel-like structure. Therefore for numerical computation, this choice of

Dataset	KDE	SmartSifter	DBOKDE	ILOF	DyCF	DyCG
Conveyor 1	7.49e-4 (9.43e-5)	3.36e-4 (7.23e-5)	4.54e-4 (3.33e-5)	1.76e-3 (6.93e-5)	**7.97e-5** **(7.23e-6)**	1.17e-3 (6.06e-5)
Conveyor 2	7.07e-4 (1.657e-4)	2.26e-4 (1.20e-4)	2.71e-4 (5.17e-5)	9.96e-4 (6.66e-5)	**3.99e-5** **(5.50e-6)**	1.14e-3 (6.33e-5)
Conveyor 3	4.20e-4 (1.88e-4)	1.68e-3 (1.47e-3)	1.67e-4 (3.26e-5)	2.31e-3 (8.19e-4)	**2.02e-5** **(3.56e-6)**	8.47e-4 (1.54e-4)
Conveyor 4	1.16e-3 (4.64e-4)	2.72e-4 (1.16e-4)	3.91e-4 (1.60e-4)	1.35e-3 (6.13e-5)	**6.74e-5** **(7.37e-6)**	1.62e-3 (1.23e-4)
Conveyor 5	7.32e-4 (9.07e-5)	1.91e-4 (6.01e-5)	2.71e-4 (9.55e-5)	9.33e-4 (3.34e-4)	**5.45e-5** **(1.25e-5)**	1.09e-3 (1.29e-4)
Global	7.53e-4 (3.30e-4)	5.41e-4 (8.52e-4)	***3.11e-4*** ***(1.23e-4)***	1.47e-3 (6.28e-4)	**5.23e-5** **(2.24e-5)**	1.17e-3 (2.75e-4)

Table 8: MV mean (standard deviation in brackets) on conveyor data streams. Best value in bold and second best value in bold italic.

Dataset	KDE	SmartSifter	DBOKDE	ILOF	DyCF	DyCG
Conveyor 1	1.25e-2	1.69e-3	5.20e-2	2.73e-2	**7.24e-4**	9.96e-4
Conveyor 2	1.35e-2	1.60e-3	5.72e-2	3.05e-2	**7.56e-4**	9.20e-4
Conveyor 3	1.41e-2	1.54e-3	5.02e-2	2.82e-2	**6.68e-4**	8.76e-4
Conveyor 4	1.42e-2	1.72e-3	5.34e-2	2.85e-2	**7.74e-4**	1.05e-3
Conveyor 5	1.39e-2	2.15e-3	5.73e-2	3.08e-2	**1.06e-3**	1.39e-3
Global	1.37e-2	1.74e-3	5.40e-2	2.90e-2	**7.97e-4**	***1.05e-3***

Table 9: Duration (in seconds per point) mean (standard deviation in brackets) on conveyor data streams. Best value in bold and second best value in bold italic.

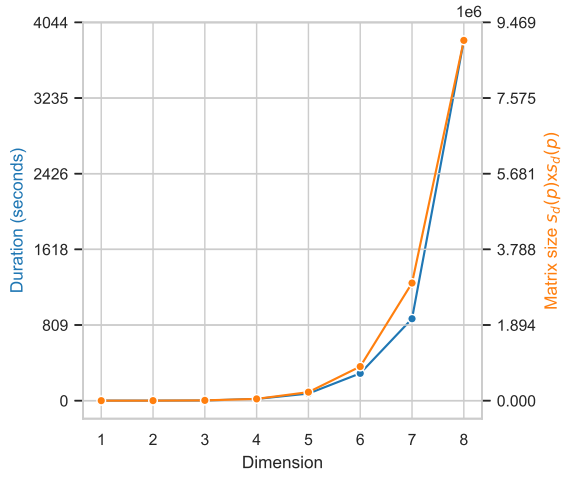


Fig. 17: Growth of processing duration compared to matrix size when dimension p increases

basis is not recommended in general, especially if the dimension of the matrix is large, in which case its inversion is severely ill-conditioned. Using the basis of Chebyshev polynomials is definitely better, as advocated (for many other purposes as well) in Chebfun [8].

The numerical instability issue has been observed to be reinforced when using the Sherman-Morrison formula of Equation (9), so for the evaluation section of this article we used the incrementation of the moment matrix and its inversion at each step. Theoretically, solving the numerical issue would mean being able to use the Sherman-Morrison formula, which would further reduce the time complexity of the two algorithms. Some experiments will be made in this direction.

- *A scaling up issue* stemming from the size of the moment matrix, which is $\binom{p+d}{d} \times \binom{p+d}{d}$, where p is just the problem dimension. This is why DyCF and DyCG are devoted to low dimensional outlier detection problems, as showcased by Figure 17. Nevertheless, the moment matrix size could be contained with a workaround consisting of randomly selecting a subsets of monomials. Future work will test this idea and assess its impact on the accuracy of the resulting models.

Acknowledgments

This project is related to ANITI through the French "Investing for the Future – PIA3" program under the Grant agreement n°ANR-19-PI3A-0004. Jean B. Lasserre's research is also part of the DesCartes' program supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program.

Statements and Declarations

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Aggarwal CC, Yu PS, Han J, et al (2003) - A Framework for Clustering Evolving Data Streams. In: Proceedings 2003 VLDB Conference. Morgan Kaufmann, San Francisco, p 81–92, https://doi.org/10.1016/B978-012722442-8/50016-1

[2] Asteriou D, Hall SG (2011) Arima models and the box–jenkins methodology. Applied Econometrics 2(2):265–286

[3] Beckmann N, Kriegel HP, Schneider R, et al (1990) The r*-tree: An efficient and robust access method for points and rectangles. In: Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data. Association for Computing Machinery, New York, NY, USA, SIGMOD '90, p 322–331, https://doi.org/10.1145/93597.98741

[4] Ben-Gal I (2005) Outlier Detection. In: Data Mining and Knowledge Discovery Handbook. Springer US, Boston, MA, p 131–146, https://doi.org/10.1007/0-387-25465-X_7

[5] Breunig MM, Kriegel HP, Ng RT, et al (2000) LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. Association for Computing Machinery, New York, NY, USA, SIGMOD '00, pp 93–104, https://doi.org/10.1145/342009.335388

[6] Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: A survey. ACM Computing Surveys 41(3):15:1–15:58. https://doi.org/10.1145/1541880.1541882

[7] Dreiseitl S, Osl M, Scheibböck C, et al (2010) Outlier Detection with One-Class SVMs: An Application to Melanoma Prognosis. AMIA Annual Symposium Proceedings 2010:172–176

[8] Driscoll TA, Hale N, Trefethen LN (2014) Chebfun guide

[9] Ducharlet K (2024) ODDS. URL https://github.com/kyducharlet/odds

[10] Duraj A, Szczepaniak PS (2021) Outlier Detection in Data Streams — A Comparative Study of Selected Methods. Procedia Computer Science 192:2769–2778. https://doi.org/10.1016/j.procs.2021.09.047

[11] F. Y. Edgeworth (1887) XLI. On discordant observations. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 23(143):364–375. https://doi.org/10.1080/14786448708628471

[12] Gan E, Ding J, Tai KS, et al (2018) Moment-based quantile sketches for efficient high cardinality aggregation queries. Proceedings of the VLDB Endowment 11(11):1647–1660. https://doi.org/10.14778/3236187.3236212

[13] Goix N (2016) How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms? arXiv:160701152 [cs, stat]

[14] Goldstein M, Dengel A (2012) Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. KI-2012: poster and demo track 9

[15] Hawkins D (1980) Identification of Outliers. Monographs on Statistics and Applied Probability, Springer Netherlands, https://doi.org/10.1007/978-94-015-3994-4

[16] Huang JW, Zhong MX, Jaysawal BP (2020) TADILOF: Time Aware Density-Based Incremental Local Outlier Detection in Data Streams. Sensors 20(20):5829. https://doi.org/10.3390/s20205829

[17] Hyndman RJ, Koehler AB, Snyder RD, et al (2002) A state space framework for automatic forecasting using exponential smoothing methods. International Journal of Forecasting 18(3):439–454. https://doi.org/10.1016/S0169-2070(01)00110-8

[18] Karimian SH, Kelarestaghi M, Hashemi S (2012) I-IncLOF: Improved incremental local outlier detection for data streams. In: The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), pp 023–028, https://doi.org/10.1109/AISP.2012.6313711

[19] Karnin Z, Lang K, Liberty E (2016) Optimal Quantile Approximation in Streams. In: 2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS), pp 71–78, https://doi.org/10.1109/FOCS.2016.17

[20] Knorr EM, Ng RT (1998) Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the 24rd International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB '98, p 392–403

[21] Kontaki M, Gounaris A, Papadopoulos AN, et al (2011) Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th International Conference on Data Engineering, pp 135–146, https://doi.org/10.1109/ICDE.2011.5767923

[22] Kristan M, Leonardis A, Skočaj D (2011) Multivariate online kernel density estimation with gaussian kernels. Pattern Recognition 44(10):2630–2642. https://doi.org/10.1016/j.patcog.2011.03.019

[23] Langrené N, Warin X (2019) Fast and stable multivariate kernel density estimation by fast sum updating. Journal of Computational and Graphical Statistics 28(3):596–608. https://doi.org/10.1080/10618600.2018.1549052

[24] Lasserre JB, Pauwels E (2019) The empirical Christoffel function with applications in data analysis. Advances in Computational Mathematics 45(3):1439–1468. https://doi.org/10.1007/s10444-019-09673-1

[25] Lasserre JB, Pauwels E, Putinar M (2022) The Christoffel–Darboux Kernel for Data Analysis. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, https://doi.org/10.1017/9781108937078

[26] Malhotra P, Vig L, Shroff G, et al (2015) Long short term memory networks for anomaly detection in time series. In: ESANN, pp 89–94

[27] Malini N, Pushpa M (2017) Analysis on credit card fraud identification techniques based on KNN and outlier detection. In: 2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), pp 255–258, https://doi.org/10.1109/AEEICB.2017.7972424

[28] Marx S, Pauwels E, Weisser T, et al (2021) Semi-algebraic approximation using Christoffel–Darboux kernel. Constructive Approximation 54(3):391–429

[29] Na GS, Kim D, Yu H (2018) Dilof: Effective and memory efficient local outlier detection in data streams. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery,

New York, NY, USA, KDD '18, p 1993–2002, https://doi.org/10.1145/3219819.3220022

[30] Palpanas T, Papadopoulos D, Kalogeraki V, et al (2003) Distributed deviation detection in sensor networks. ACM SIGMOD Record 32(4):77–82. https://doi.org/10.1145/959060.959074

[31] Parzen E (1962) On Estimation of a Probability Density Function and Mode. The Annals of Mathematical Statistics 33(3):1065 – 1076. https://doi.org/10.1214/aoms/1177704472

[32] Pokrajac D, Lazarevic A, Latecki LJ (2007) Incremental Local Outlier Detection for Data Streams. In: 2007 IEEE Symposium on Computational Intelligence and Data Mining, pp 504–515, https://doi.org/10.1109/CIDM.2007.368917

[33] Roa NB, Travé-Massuyès L, Grisales VH (2019) DyClee: Dynamic clustering for tracking evolving environments. Pattern Recognition 94:162. https://doi.org/10.1016/j.patcog.2019.05.024

[34] Ruff L, Kauffmann JR, Vandermeulen RA, et al (2021) A unifying review of deep and shallow anomaly detection. Proceedings of the IEEE

[35] Sadik S, Gruenwald L (2014) Research issues in outlier detection for data streams. SIGKDD Explor Newsl 15(1):33–40. https://doi.org/10.1145/2594473.2594479

[36] Salehi M, Rashidi L (2018) A Survey on Anomaly detection in Evolving Data: [with Application to Forest Fire Risk Prediction]. ACM SIGKDD Explorations Newsletter 20(1):13–23. https://doi.org/10.1145/3229329.3229332

[37] Salehi M, Leckie C, Bezdek JC, et al (2016) Fast Memory Efficient Local Outlier Detection in Data Streams. IEEE Transactions on Knowledge and Data Engineering 28(12):3246–3260. https://doi.org/10.1109/TKDE.2016.2597833

[38] Scott DW (1992) Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley Series in Probability and Statistics, Wiley, https://doi.org/10.1002/9780470316849

[39] sk1010k (2021) SmartSifter. URL https://github.com/sk1010k/SmartSifter

[40] Sreevidya S (2014) A survey on outlier detection methods. IJCSIT) International Journal of Computer Science and Information Technologies 5(6)

[41] Thakkar P, Vala J, Prajapati V (2016) Survey on outlier detection in data stream. Int J Comput Appl 136:13–16

[42] Tran L, Fan L, Shahabi C (2016) Distance-based outlier detection in data streams. Proceedings of the VLDB Endowment 9(12):1089–1100. https://doi.org/10.14778/2994509.2994526

[43] Vu MT, Bachoc F, Pauwels E (2022) Rate of convergence for geometric inference based on the empirical christoffel function. ESAIM: PS 26:171–207

[44] Wang H, Bah MJ, Hammad M (2019) Progress in Outlier Detection Techniques: A Survey. IEEE Access 7:107964–108000. https://doi.org/10.1109/ACCESS.2019.2932769

[45] Yamanishi K, Takeuchi JI, Williams G, et al (2004) On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. Data Min Knowl Discov 8(3):275–300. https://doi.org/10.1023/B:DAMI.0000023676.72185.7c

[46] Zhang J (2013) Advancements of outlier detection: a survey. ICST Transactions on Scalable Information Systems 13(1):1–26. https://doi.org/10/22596.pdf

[47] Zhang J, Zulkernine M (2006) Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection. In: 2006 IEEE International Conference on Communications, pp 2388–2393, https://doi.org/10.1109/ICC.2006.255127

[48] Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. ACM SIGMOD Record 25(2):103–114. https://doi.org/10.1145/235968.233324

[49] Zhao F, Maiyya S, Wiener R, et al (2021) KLL± approximate quantile sketches over dynamic datasets. Proceedings of the VLDB Endowment 14(7):1215–1227. https://doi.org/10.14778/3450980.3450990