Leveraging the Christoffel-Darboux Kernel for Online Outlier Detection

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

Leveraging the Christoffel-Darboux Kernel for Online Outlier Detection

Kévin Ducharlet*, †, Louise Travé-Massuyès*, Jean-Bernard Lasserre*,
Marie-Véronique Le Lann* and Youssef Miloudi†
*LAAS-CNRS, University of Toulouse, CNRS, INSA
Toulouse, France
Email: {kducharlet, louise, lasserre, mvlelann}@laas.fr
†Carl Berger-Levrault
Limonest, France
Email: {kevin.ducharlet, youssef.miloudi}@carl.eu

***Abstract*—Outlier detection is a subject of interest in data mining. With the evolution of data acquisition methods such as wireless sensor networks, there is a need to detect outliers in data streams. However, dealing with data streams is challenging due to the amount of data that grows infinitely and the non-stationarity of the distribution. On top of that, the detection has generally to be done in an unsupervised way. Some methods have been proposed to tackle this problem but none of them can be easily parameterized. This paper proposes a novel method based on the Christoffel-Darboux kernel borrowed from the theory of approximation and orthogonal polynomials. This method perfectly applies to data streams while being deployable with no tuning at all. It is compared to some state-of-the-art methods for outlier detection in data streams and applied to the data from an industrial luggage conveyor showing very convincing results.**

***Index Terms*—Outlier detection, Anomaly detection, data streams, Christoffel-Darboux kernel, Kernel density estimation, Data mining**

I. Introduction

Data mining is defined as the process of discovering and extracting knowledge from usually very large datasets. The purpose of this task is to identify unkwown patterns that can prove to be useful in decision making such as groups of similar points, dependencies or unusual observations.

The discovery and analysis of unusual observations, also known as outlier, anomaly, out-of-distribution or novelty detection, is particularly important in data mining. First, outliers can carry a lot of information about the data and are thus of special interest in applications such as network traffic [1], medical diagnosis [2] or fraud detection [3], where it is crucial to find any abnormal behaviour. Outliers also greatly disturb most of machine learning methods used for tasks such as prediction or decision making, so their removal is required to assert the accuracy of obtained results.

Nowadays, many data sources such as wireless sensor networks, social networks, medical systems, web traffic or online transactions generate data continuously. The resulting datasets have the peculiarity of uncertainty and constant evolution. Outlier detection is thus even more important in this context, but also more challenging. Indeed, methods for static datasets usually look for a mapping function that gives an outlierness score to instances based on the observation of a whole set of historical instances. In some cases, only historical instances can be assigned an outlierness score. In other cases, the method allows for qualifying new instances as inliers or outliers but the function does not evolve through time. Yet, to be able to detect outliers in data streams, methods need to look for a mapping function that adapts to new instances and to deal with infinite lengths of datasets.

Most of the time, labels are not available for the task of outlier detection, which means that one does not know if instances in datasets are actually outlying. This makes most of outlier detection methods unsupervised. For static cases, it is possible to perform a preprocessing task to label data and thus meet the supervised learning conditions. It is also possible to get a trusted set of normal instances or to select some known outlying instances to apply semi-supervised tasks. However, in data streams, the evolution of the data distribution makes labelling useless because quickly obsolete.

The contribution of this paper is the proposal of an entirely new method for outlier detection in data streams that can be easily applied with few parameters and no tuning. This method responds to the previously mentioned challenges, being able to deal with evolving distributions and with infinite datasets by summarization. It is evaluated and compared with state-of-the-art methods for outlier detection in data streams based on a set of key characteristics.Its performance is also shown on real industrial data coming from a luggage conveyor system.

The paper is organized as follows. Section II provides a state of the art of outlier detection in data streams. Section III introduces the two datasets that will be used throughout the paper. Then, Section IV presents the Christoffel-Darboux Kernel (CD-Kernel) and the outlier score it can compute in data streams. Section V describes our contribution, explaining how it aggregates multiple CD-Kernel methods to reduce the tuning requirements. Some experiments are presented in Section VI to evaluate the proposed method and compare it with few other methods from the state-of-the-art. To conclude, Section VII discusses its advantages, drawbacks and the improvement

This project is supported by ANITI through the French "Investing for the Future – PIA3" program under the Grant agreement noANR-19-PI3A-0004.

points while Section VIII draws a conclusion to the paper.

II. State of the Art and Related Work

This section overviews the different categories of outlier detection methods and highlights the issues related to outlier detection for data streams. The analysis of several surveys allows us to position the method that is proposed in this paper.

A. Outlier Detection

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 [4]. For this reason, different definitions have been given to describe an outlier. The survey [5] notes that this is thus difficult to find a reliable definition although retaining this one : "a data point that is significantly dissimilar to other data points or a point that does not imitate the expected typical behavior of the other points". This definition is very close to the most cited one given by Hawkins [6] as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism".

With more than a century of interest in outlier detection, a lot of different methods have been proposed and different surveys aim to tackle the task of listing, describing, categorizing and comparing these methods. Wang *et al.* [5] is one of those but the most prominent is the one by Chandola *et al.* [7] that discusses methods from their application and category view points. The survey mentions seven application fields: cyber-intrusion detection, fraud detection, medical anomaly detection, industrial damage detection, image processing, textual anomaly detection and sensor networks. Interestingly, data streams can be met in most of these fields. From the categorization perspective, some categories are given in both surveys. One can find clustering based methods, statistics methods and nearest neighbor based methods (which are separated in Distance based and Density based methods in [5]). Chandola *et al.* also mention classification based methods, information theoretic methods and spectral methods [7], whereas Wang *et al.* report ensemble, learning, graph and network based methods [5].

The need of labels during the learning process can also be used to distinguish the methods [7]. Labels are information associated to each data point qualifying it as normal or outlying. Supervised methods learn a predictive model from a training set containing labelled points to determine if new points are inliers or outliers. At the other end, unsupervised methods do not assume the availability of such labels and learn a model while making hypothesis on outlier rate in the dataset or their distance to inliers. Semi-supervised methods make a trade-off, using a set of inliers to learn a model representing normal behaviour and then checking if new points fall in the learned model (inliers) or outside (outliers). Referring to the output of the methods, it takes typically two forms: 1) a label, each point is classified as inlier or outlier, or 2) a score, an anomaly score is given to each point, a label can then be assigned using a threshold on the score.

B. Outlier Detection for Data Streams

The definition of an outlier given in the context of data streams is the same as in its static counterpart. Hawkins definition is found in [8] and [9], while [10] defines an outlier as "an object that does not conform to the expected behavior", such an object corresponding to "either noise or anomaly". However, there are some peculiarities to data streams [8], [9]:

1) *Continuous arrival rate*. Points arrive continuously, forcing methods to process arriving points before new points arrive. Points are transient and lose importance after a certain amount of time. In addition, the arrival rate may be variable.
2) *Unbounded datasets*. Datasets have theoretical infinite length. Thus, all points cannot be stored and methods must deal with limited resource, storing for instance a summary of the data.
3) *Non stationary distributions*. The distribution of data can change over time so the learned models also need to be able to evolve, which is known as concept drift.

Methods can be categorized in four categories:

1) *Statistical methods*. These methods, divided into parametric and non-parametric, aim to find a statistical distribution likely to have generated the data. Most methods are not well suited to data streams because the assumed distribution is fixed in time [9]. Nonetheless, some non-parametric methods based on histogram construction or kernel density estimation can be used for data streams [11]. Histogram construction can be made online in a supervised or semi-supervised way, in which case new points that do not fall into a bin of the histogram are considered as outlying and the other points are used to update the histogram. The histogram can also be constructed in an unsupervised way, then outliers are those instances falling in empty or low density bins [12], [13]. However, these methods work fine for a single variable but need to be aggregated in multivariate cases, which which is quite limiting. Methods based on kernel density estimation are more flexible from this point of view [14], [15]. The CD-Kernel method proposed in this paper falls in this category.
2) *Distance based methods*. Most of these methods are based on the nearest neighbors approach.To adapted to the data stream context, windowing techniques that capture a subset of recent and thus meaningful points in the dataset are used [8], [10]. The main idea is to study the neighborhood of the points in the current window and to evaluate outlierness based on this neighborhood. Several methods, like Exact-Storm and Approx-Storm [16], Abstract-C [17], DUE and MCOD [18] or Thresh_LEAP [19], propose specific indexed structure to easily access the closest neighbours of each new point and to discard old points. According to [10], MCOD is the most effective because of the use of micro-clusters to describe the neighborhoods. Besides, HalfSpaceTrees (HST) [20] is a Distance based method which gener-

alizes the well-known IsolationForest algorithm [21] to data streams.

3) *Density based methods*. These methods, especially Incremental LOF (iLOF) [22], generalize the well-known LocalOutlierFactor (LOF) approach [23]. The metric to quantify outlierness is the density around a point compared to that of its neighbors. I-IncLOF [24] and MiLOF [25] are two improvements of iLOF that aim to distinguish between new behaviours and true outliers and to reduce the memory usage respectively. More recently, TADILOF [26] improves the previous methods by considering the arrival order of data points and forgetting the oldest ones.
4) *Dynamic clustering based methods*. The main goal of these methods, exemplified by BIRCH [27], CluStream [28], DenStream [29], DStream [30], SDstream [31], STING [32], STREAM [33], and DBSTREAM [34], is to cluster data points in place of detecting outliers, which is why they are subject to some criticism [9]. The main idea is that outliers either fall in low density clusters or in no clusters at all or are far from the cluster centroïds. Most methods advantageously use a summarized representation of the points, the type of representation being where their difference lies. BIRCH and CluStream use cluster features (CFs) that are tuples containing the number of points in a cluster and the sum and squared sum of these points. DenStream, DStream and SDstream are improvements of CluStream. Another difference between these methods is how the space is divided. Micro-clusters are often used as a first division, then macro-clusters are constructed by grouping micro-clusters. Among those, Dyclee [35] can reject outlying micro-clusters.

Table I recaps all the methods cited in this section and their Python implementations in the River Project[1] or the Scikit-learn library. However, clustering methods implemented in those libraries have not been designed and optimized for outlier detection.

TABLE I
OUTLIER DETECTION METHODS FOR DATA STREAMS AND THEIR PYTHON IMPLEMENTATIONS

Category	Method	River	Scikit-learn	Home-made
Statistical	Histograms			
	Our Method			✔
Distance	Exact-Storm			
	Approx-Storm			
	Abstract-C			
	DUE			
	MCOD			
	HST	✔		
Density	iLOF			
	I-IncLOF			
	MiLOF			
	TADILOF			
CLustering	BIRCH		✔	
	CluStream	✔		
	DenStream	✔		
	DStream			
	SDstream			
	STING			
	STREAM	✔		
	DBSTREAM	✔		
	DyClee			✔

III. DATASETS

In this section, two datasets that will be used throughout the paper are presented. The first one is a synthetic dataset while the second one is an industrial dataset.

A. Synthetic Dataset: Two Moons

Two Moons is a well-known dataset for clustering containing two moon-shaped clusters to which outliers are added, uniformly distributed in the window around the moons. To adapt this dataset for data streams, the clusters are sorted to first get the points from a first moon and then get the ones from the other moon. Thus, the second moon can be considered as a new normal behaviour. Regarding outliers, they are uniformly spread in the dataset. The whole dataset is given in Figure 1. It has been generated using the *make_moons* method from Scikit-learn with 5000 samples, setting the noise parameter at 0.05. 20 points have been added as outliers.

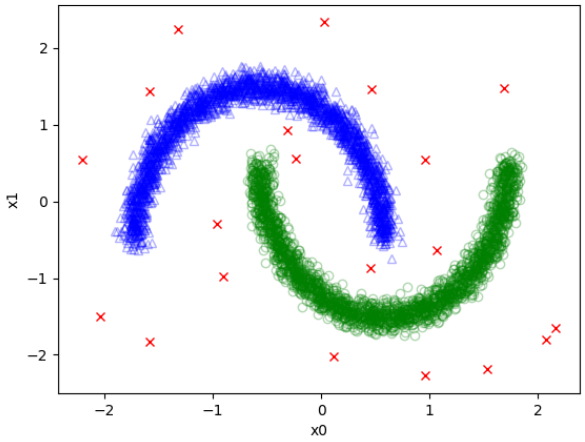

Fig. 1. Representation of the Two Moons dataset. Blue triangles represent the first cluster. Green circles represent the second cluster. Red crosses represent outliers.

B. Industrial Dataset: Luggage Conveyor

The second dataset is generated by sensors on a luggage conveyor. Those sensors measure two physical quantities: speed of the conveyor belt and intensity of the engine. This dataset is quite specific as it is composed of three normal operating modes with non-linear transitions in between. The *stop* operating mode is predominant and matches the conveyor stopped with speed and intensity at zero. The *standard* operating mode matches the operation of the conveyor with nominal speed and intensity. A less frequent *heavy_load* operating mode with lower speed and higher intensity can be

[1]The River project [36] is a library for machine learning on data streams.

distinguished for heavy luggages. Finally, there are transitions between the different modes. When the conveyor starts, there is an intensity peak followed by a raise in speed. When it stops, intensity reduces faster than speed. The complete dataset and the different modes can be seen in Figure 2 and is composed of 166926 observations.

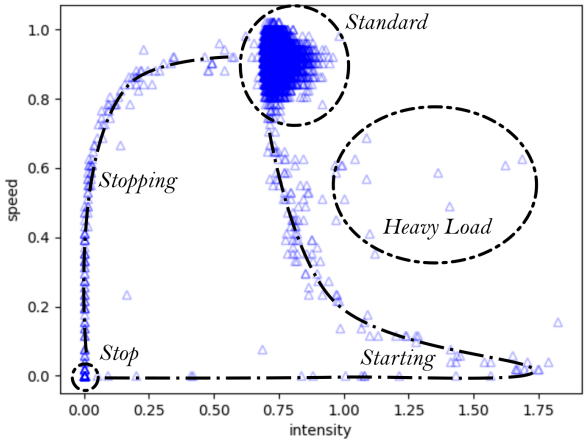

Fig. 2. Representation of the Luggage Conveyor dataset. Operating modes and transitions are explained.

IV. CD-Kernel based Outlier Score

The CD-Kernel and the associated Christoffel Function (CF) is a well-known tool from the theory of approximation and orthogonal polynomials. The CD-kernel and the CF depend on a parameter d (the "degree") and are associated with a measure μ with support $\Omega \subset \mathbb{R}^p$ (usually compact with nonempty interior). One of the main and salient features of the CF (denoted Λ_d^μ) is its ability to encode the support Ω in the sense that for every $\mathbf{x}$ *outside* Ω, the function $\mathbf{x} \mapsto \Lambda_d^\mu(\mathbf{x})$ converges *exponentially fast* to zero with d, and *not* for $\mathbf{x} \in \Omega$. In particular, for dimension $p = 2, 3$ one observes that the level sets $\Omega_\gamma := \{\mathbf{x} : \Lambda_d^\mu(\mathbf{x}) \geq \gamma\}$ nicely capture the geometric shape of Ω quite accurately, even for a low degree d. Relatively recently, Lasserre and Pauwels [37] and Lasserre et al. [38] have promoted the CD-kernel and the CF as a powerful tool for data analysis. They have shown how some of its key properties can be helpful to address important problems like density approximation, support inference, and outlier detection, where the measure μ of interest is now a *discrete measure* whose support is the set (or "cloud") of data points.

A. Brief outline

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 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 (which means: 1) ordered according to ascending monomial degree and then 2) using lexicographic order on variables considering $X_1 = a$, $X_2 = b$, ...). The size of the vector $\mathbf{v}_d(X)$, that is noted $s(d)$, depends on p and d and is equal to $\binom{p+d}{d}$.

Interestingly, given a finite Borel probability measure μ on a compact set $\Omega \subset \mathbb{R}^p$ with nonempty interior, its moment matrix $M_d(\mu)$ is a matrix indexed by the monomials of $\mathbf{v}_d(X)$. Let

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

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

$$M_2(\mu): \begin{array}{c|cccccc} & 1 & X_1 & X_2 & X_1^2 & X_1X_2 & X_2^2 \\ \hline 1 & 1 & y_{1,0} & y_{0,1} & y_{2,0} & y_{1,1} & y_{0,2} \\ X_1 & y_{1,0} & y_{2,0} & y_{1,1} & y_{3,0} & y_{2,1} & y_{1,2} \\ X_2 & y_{0,1} & y_{1,1} & y_{0,2} & y_{2,1} & y_{1,2} & y_{0,3} \\ X_1^2 & y_{2,0} & y_{3,0} & y_{2,1} & y_{4,0} & y_{3,1} & y_{2,2} \\ X_1X_2 & y_{1,1} & y_{2,1} & y_{1,2} & y_{3,1} & y_{2,2} & y_{1,3} \\ X_2^2 & y_{0,2} & y_{1,2} & y_{0,3} & y_{2,2} & y_{1,3} & y_{0,4} \end{array}$$

It 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 based outlier score can then be computed for any $\mathbf{x} \in \mathbb{R}^p$ as $Q_{\mu,d}(\mathbf{x}) = \mathbf{v}_d(\mathbf{x})^T M_d(\mu)^{-1} \mathbf{v}_d(\mathbf{x})$. $Q_{\mu,d}$ is a sum of squares polynomial whose degree is equal to $2d$ and its inverse function $\mathbf{x} \mapsto Q_{\mu,d}(\mathbf{x})^{-1}$ is known as the Christoffel Function (CF), i.e $\Lambda_d^\mu = Q_{\mu,d}(\mathbf{x})^{-1}$. $Q_{\mu,d}(\mathbf{x})$ can be used as a tool, for kernel density estimation and outlier detection. Indeed outside the support of μ, the function $\mathbf{x} \mapsto Q_{\mu,d}(\mathbf{x})$ (the reciprocal of the CF) takes high values and increases exponentially fast with d, and so provides a great outlier scoring function [2].

For the outlier detection application in data analysis, only an *empirical* moment matrix is available, associated with a discrete measure μ_n whose support is a set of n observations $\mathcal{X} = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ sampled from the distribution of μ. In this case, the empirical version of Equations (1) and (2) read

$$y_\alpha(\mu_n) = \frac{1}{n} \sum_{x \in \mathcal{X}} \mathbf{x}^\alpha \tag{3}$$

and

$$M_d(\mu_n) = \frac{1}{n} \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{v}_d(\mathbf{x})^T \mathbf{v}_d(\mathbf{x}) \tag{4}$$

Note that, considering $\mathcal{X}$ as a dataset, $M_d(\mu_n)$ can be seen as a summary (or an encoding) of this dataset that avoids keeping

[2]The choice of $Q_{\mu,d}$ instead of its inverse, i.e the CF $Q_{\mu,d}(\mathbf{x})^{-1} = \Lambda_d^\mu$, comes from the fact that outliers are generally assigned higher scores than normal points and not the inverse.

in memory all the points, as required for methods dealing with data streams. It is also important to note that the invertibility of $M_d(\mu_n)$ is not guaranteed for low values of n. Figures 3 and 4 show level sets for $Q_{\mu,d}$ obtained using the presented datasets as supports. The ability to capture the geometric shape of the support is quite visible.

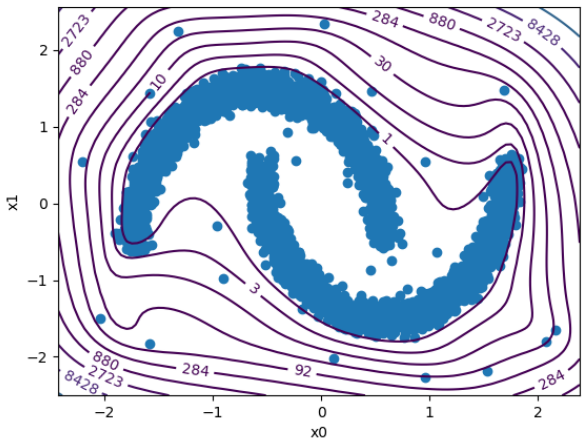

Fig. 3. Levelsets obtained from the Two Moons dataset for $d = 6$. Note that outliers between the moons are considered as inliers with the levelset labelled as 1 which yet rejects some inliers.

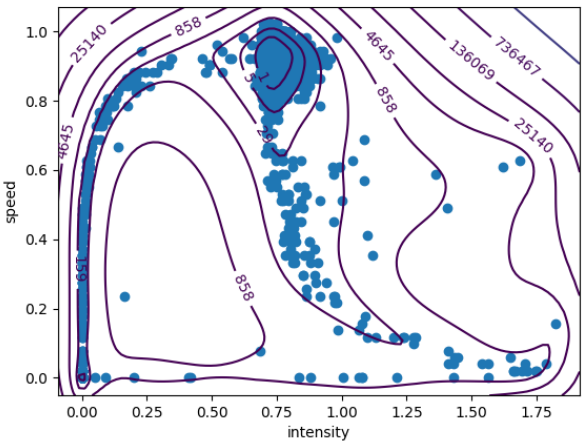

Fig. 4. Levelsets obtained from the Luggage Conveyor dataset for $d = 6$. The levelset labelled as 858 captures the two main operating modes, the transitions and most of the points from the dragged mode.

B. Interesting Properties

The CD-Kernel based outlier score has several interesting properties.

1) Incremental Update: Using Equations (3) and (4), one can easily update the support with new points or in a sliding window. For instance, let $\mathcal{X}' = \{\mathbf{x}_3, \mathbf{x}_4, ..., \mathbf{x}_{n+2}\}$ be the sample obtained at time indexed by $n+2$ by sliding a fixed length window. Let denote as μ_{n+2} the measure supported by $\mathcal{X}'$. Then, the associated moment matrix is given by

$$\begin{aligned} M_d(\mu_{n+2}) = \frac{1}{n}(nM_d(\mu_n) &- \mathbf{v}_d(\mathbf{x}_1)^T\mathbf{v}_d(\mathbf{x}_1) \\ &- \mathbf{v}_d(\mathbf{x}_2)^T\mathbf{v}_d(\mathbf{x}_2) \\ &+ \mathbf{v}_d(\mathbf{x}_{n+1})^T\mathbf{v}_d(\mathbf{x}_{n+1}) \\ &+ \mathbf{v}_d(\mathbf{x}_{n+2})^T\mathbf{v}_d(\mathbf{x}_{n+2})). \end{aligned} \tag{5}$$

Inverting the moment matrix can be expensive. So updating directly the inverse of the moment matrix is a better choice. This can be done using the Sherman-Morrison formula that states that, for an invertible square matrix A and any vectors u, v,

$$(A + v^T u)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}. \tag{6}$$

Figure 5 shows the evolution of levelsets with incremental update of the inverse of the moment matrix for the synthetic dataset. The moment matrix first fits the first half of the dataset with the first moon. Then, the rest of the points of the dataset are splitted in three batches of sizes 836, 836 and 838, and the matrix is updated three times. Here, windowing methods are not used to remove the oldest points, only new ones are added.

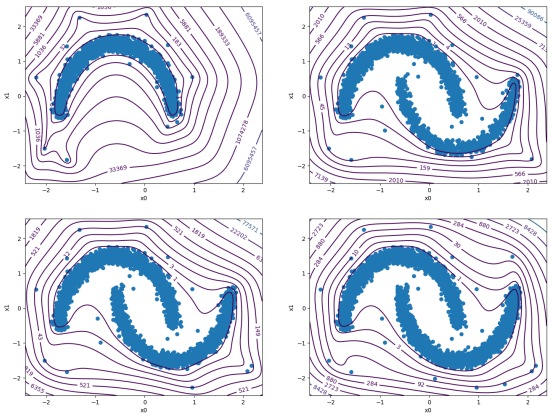

Fig. 5. Levelsets obtained from the Two Moons dataset for $d = 6$ with incremental updates. Note that the first update (top right) is enough to capture the shape of the second moon.

2) Forgetting capability: In spite of using fixed length sliding windows, which requires keeping in memory all the points in the current window to be able to remove them, it is possible to assign a weight to each point in order to reduce the effect of older points in the support. For instance, let $0 < \gamma < 1$ be the forgetting coefficient, with γ close to 1 for persistent points, Equation (4) for the moment matrix can be written as

$$M_d(\mu) = \frac{1-\gamma}{1-\gamma^n}\sum_{i=1}^{n}\gamma^{n-i}\mathbf{v}_d(\mathbf{x}_i)^T\mathbf{v}_d(\mathbf{x}_i) \tag{7}$$

Thus, the weight γ^{n-1} given to the first point in the dataset $\mathcal{X}$ tends to 0 when the length of the dataset n tends to infinity.

On Figure 6, one can observe the impact of a forgetting factor $\gamma = 0.998$ on the levelsets with the same incrementation as the one presented in Figure 5. The more new points are added from the second moon and the less the first moon affects the support. However, the results are sensitive to the value of γ; with $\gamma = 0.995$, the first moon is already out of the support on the second graph, while it is still in the support on the last graph for $\gamma = 0.999$.

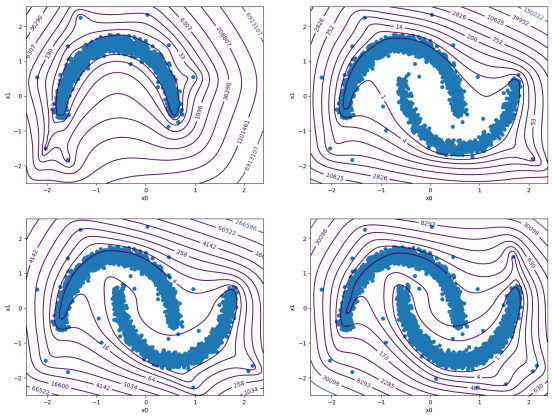

Fig. 6. Levelsets obtained from the Two Moons dataset for $d = 6$ with incremental updates. Note that the first moon is pushed out of the captured support as soon as the second update (bottom left).

3) Characteristic score evolution: It has already been said that the score, obtained with $Q_{\mu,d}$, is greater for points outside the support of μ than for points inside of μ. More precisely, it grows exponentially when the points go away from the center of the support. This can be seen by looking at the levelset values of Figures 3 and 4.

Moreover, for a given point $\mathbf{x} \in \mathbb{R}^p$, the growth of $Q_{\mu,d}(\mathbf{x})$ when d increases depends on whether $\mathbf{x}$ is in the support of μ or not. The growth is polynomial for points inside the support and exponential for points outside the support, as shown in the score evolution of points in Figure 7, where the point of the top graph is between the two moons, the point of the middle graph is an inlier and the point of the bottom one is an outlier (watch the e7 coefficient for the scale). This property proves to be helpful to discriminate between inliers and outliers and it is leveraged in the method that is proposed in section V.

4) Easy-to-tune parameters: In applications treated in [37], the levelset defined by $\Omega_\gamma := \{\mathbf{x} \in \mathbb{R}^p \mid Q_{\mu,d}(\mathbf{x}) \leq \binom{p+d}{d}\}$ captured most of the points in the support of μ. However, changing the threshold value $\gamma := \binom{p+d}{d}$ may have a great impact on the resulting levelset Ω_γ. Noticeably, this particular levelset does not always capture all the points representing normal behaviours. It is possible to add a regularization term r to the threshold value and get steadier results, i.e. the levelset $\Omega_\gamma := \{\mathbf{x} \in \mathbb{R}^p \mid Q_{\mu,d}(\mathbf{x}) \leq r\binom{p+d}{d}\}$ may be considered, with $0 < r \leq 1$ being a parameter to tune to capture a tighter or looser approximation of the support.

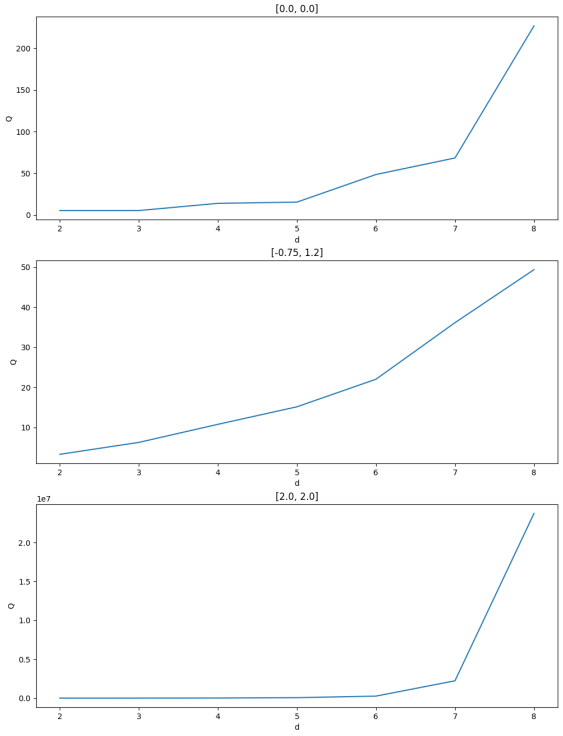

Fig. 7. Evolution of the score for different degrees and for three different points of given coordinates. The used model is the one described by Fig. 3 for the synthetic dataset, it can be used to position the points. The top graph is for a point between the two moons, at the center of the graph. The middle graph is for a point inside the first moon. The bottom graph is for a point outside the support (watch the e7 coefficient for the scale).

If $\gamma := \binom{p+d}{d}$ is used as threshold, an outlier detection approach based on $Q_{\mu,d}$ requires to set only one parameter, i.e. d, which is an integer usually in the interval $[2, 8]$. This makes the CD-Kernel based scoring function an easy-to-tune tool. With $r\binom{p+d}{d}$ as threshold γ, the parameter $r \in]0, 1]$ has also to be tuned. Note that $r = 0.5$ is adequate in most simple cases. In more complex cases, the usual way to set a threshold for outlier detection is to base it on top k outliers or on an expected outlier ratio in the dataset. However, this is not convenient in the case of data streams where the threshold and the outlier ratio can evolve.

5) Low computational complexity: Finally, it is important to notice that the computation of the score for a point does not depend on the length n of the dataset $\mathcal{X}$ but solely on p and d. This is a significant advantage for the purpose of computing the score in a data stream context.

V. MCD-Kernel method: a non-tuning online method based on the evolution of the CD-Kernel score

Using the CD-Kernel based outlier score as a basis, an outlier detection method for data streams that can be deployed without tuning is proposed. Note that parameter tuning is a difficult task in unsupervised learning because the search for optimal parameters is usually done with labelled data. On top of that, for data streams, parameters that work well at a time can become obsolete later. Thus, a non-tuning solution is much better for this problem. This can be achieved leveraging the property of the score evolution presented in section IV-B3.

A. The MCD-Kernel method

The proposed method, named Multiple CD-Kernel outlier detection method (MCD-Kernel method for short), extends the outlier scoring described in Section IV to a complete non-tuning method for outlier detection. The idea is to use the property stating how the score grows depending on the outlierness of points when the degree d increases. For this purpose, a multi-degree model that maintains multiple moment matrices is proposed as explained below.

Let us denote $\mathcal{X} = (x_j)_{j\in\mathbb{N}^*}$ the theoretical full stream dataset, where n indicates the rank in the sequence, which corresponds to a date. $\mathcal{X}_n$ is defined as the sequence extracted from $\mathcal{X}$ ending for $x_n, n \in N^*$. Note that $\mathcal{X}_n$ can either be defined as the sequence of all the points ending with x_n in the seqence u$\mathcal{X}$ or as a sliding window of fixed size with x_n as end point.

Define $\Delta = (d_i)_{1\leq i\leq k}$ as a sequence of k degrees in ascending order and consider some n. Using $\mathcal{X}_n$ as support, build the k associated empirical moment matrices $(M_{d_i}(\mu_n))_{1\leq i\leq k}$ which leads to k functions $(Q_{\mu_n,d_i})_{1\leq i\leq k}$. Note that the number of points in the batch ending at n must be chosen great enough to ensure invertibility of the matrices.

Define a normalized scoring function[3] for every n and d as

$$S_{\mu_n,d} = \frac{Q_{\mu_n,d}}{rd\binom{p+d}{d}}. \tag{8}$$

While the non-normalized score $Q_{\mu_n,d}$ would have a polynomial growth for inliers, it is now decreasing with the normalized score. This can be illustrated by comparing the score evolution on Figure 7 obtained with $Q_{\mu_n,d}$ and the one on Figure 8 obtained with $S_{\mu_n,d}$.

Summarizing, the growth of most of inliers decreases while the evolution stays increasing for outliers because of the exponential growth of $(Q_{\mu_n,d_i}(x_j))_{1\leq i\leq k}$, so a satisfying decision criteria is the average growth of the score. This is assessed as follows for every point $x_j \in \mathcal{X}_{n_0}$:

1) Compute the scores $(S_{\mu_n,d_i}(x_n))_{1\leq i\leq k}$,
2) Compute the difference

$$S'_{\mu_n,d_i}(x_j) = \frac{S_{\mu_n,d_i}(x_j) - S_{\mu_n,d_{i-1}}(x_j)}{d_i - d_{i-1}}$$

for $i \in [2, k]$,

3) Qualify x_j as outlier if the following condition is satisfied

$$\frac{1}{k-1}\sum_{i=2}^{k} S'_{\mu_n,d_i}(x_j) > 0. \tag{9}$$

Considering the data in the stream $\mathcal{X}$, for every new point x_{n+1}, update the moment matrices $(M_{d_i}(\mu_n))_{1\leq i\leq k}$ from the support $\mathcal{X}_n$ to fit the new support $\mathcal{X}_{n+1}$ using Equations (5) and (6) and repeat the above process.

To compute $S_{\mu_n,d}$ the sequence of degrees Δ and the parameter r must be set. The latter being a constant, it does not impact the score growth and the decision. It is generally set to $r = 0.5$ as mentioned above. The sequence of degrees determines the method but it is easy to decide. Indeed, d usually takes values in $[2, 8]$ so the set $\Delta = 2, 3, 4, 5, 6, 7, 8$ is chosen. Interestingly, the experiments show that the results do not vary much over the choice of Δ, making of the MCD-Kernel a non-tuning method.

[3]This is the score used for the levelsets in Figures 3, 4, 5 and 6.

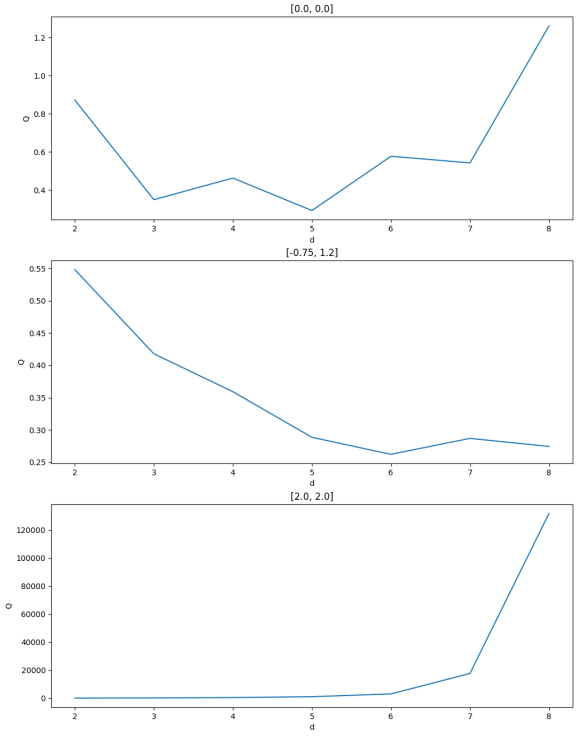

Fig. 8. Same as Fig. 7 but using $S_{\mu_n,d}$ as scoring function in place of $Q_{\mu_n,d}$.

B. Possible Variations

In this subsection, some variations that can be introduced to the MCD-Kernel method are presented.

1) Processing batches of points: The MCD-Kernel method can be adapted to points that arrive in batches. For every new batch, all the points included in the batch are evaluated and the moment matrices are then updated using the new support containing the whole batch. However, this means that new points are not evaluated using the most recent support, particularly for the last points in large batches.

2) Reducing computation time of the evaluation process: To avoid evaluating all the incoming points, an evaluation based on a reference model can be proposed. Let us choose $d_{ref} \in \Delta$ with its associated moment matrix $M_{d_{ref}}(\mu_n)$ and function $Q_{\mu_n, d_{ref}}$. Then, check the following condition on the reference score $S_{\mu_n, d_{ref}}(x_n) \leq 1$. If the inequality holds, then the point is considered an inlier, else it is a possible outlier and it should be evaluated with the complete process.

3) Selective support: The most outlying points should not be added to the support. If all outliers are rejected, the model will not adapt to new behaviour. However, including all outliers can lead to fail to detect outliers that appear in the same region in space. One can want to reject outliers that have the greatest scores.

VI. Evaluation and Comparison

A. How to Evaluate?

It is commonly accepted that evaluating unsupervised outlier detection is a difficult task. Wang et al. [5] show that when an evaluation is done, it is based on supervised metrics using labelled datasets, which involves selecting these datasets and limiting the scope of the results. Some works have been focused on creating metrics to evaluate methods using unlabelled datasets [39], [40] but the results are not always in accordance with supervised metrics when compared on labelled datasets. In addition, evaluation is done at a given time and is not applicable to evolving models. On top of that, [39] bases its evaluation on the outlier score, which is not available for all methods. In the case of data streams, memory usage and CPU time seem among the relevant metrics [10].

There is obviously a need for a benchmark to evaluate and compare effectively outlier detection algorithms for data streams taking their peculiarities into account [5]. The only existing one is the Numenta Anomaly Benchmark (NAB) framework [41] but it relies on labelled datasets.

The evaluation metrics that are proposed to assess Some specific properties are important to be evaluated for online outlier detection methods. These are listed below with the procedure that is used in the experiments reported in section VI-B to evaluate them :

1) *Consistency*. Evaluating this property is based on the CC-Eval method proposed in [40] and aims to evaluate how consistent the decisions are in the sense explained below. To do so, a supervised classifier is trained using the labels generated as decisions by the outlier detection method. The supervised classifier is then asked to classify new unseen instances. Those unseen instances are also evaluated with the outlier detection model. The results of both models are then compared and the precision is used as consistency metric. As so, this method is not suited to online methods but it can be easily extended by considering the model at consecutive times and performing a series of consistency evaluations.
2) *Learning Speed*. Online outlier detection methods ought to quickly recognise new persistent behaviour as normal. It is hence interesting to measure how fast outlier detection methods are able to learn novel behaviors. The experiment that is proposed for this is based on the synthetic Two Moons dataset. All outliers are removed from the dataset. Training is performed on the first moon data. Then the data from the second moon is given as input. The graph of the number of detected outliers as a function of the number of evaluated instances is used as metric.
3) *Processing Time*. Processing time is obtained by measuring how long it takes to train the models, to evaluate new points and to update the models.
4) *Memory Usage*. Memory usage is simply measured through the process.

B. Experimental Results

In this section, the different evaluation metrics used and the results obtained for the proposed method are presented. These results are divided according to the dataset used for the tests, synthetic or industrial. Consistency is evaluated on both datasets, learning speed is only measured for the Two Moons dataset and computing time and memory usage are measured on the Luggage Conveyor dataset.

The implemented methods applicable to outlier detection in a data stream context are very few. Experiments and comparative analysis have been performed with HST relying on the implementation in the River project [36] and with DyClee (cf. Table I).

1) Experiments based on the Two Moons dataset:

a) Consistency: For these tests, the method CC-Eval [40] has been applied as presented in section VI.A.1 with kNeigborsClassifier from Scikit-learn as supervised classifier. kNeigborsClassifier has been chosen because it implements a methods to immediately compute the mean accuracy of the prediction and to estimate the probability that a point belongs to a class.

Four timesteps are evaluated, which correspond to the ones used in Figure 5. Figure 9 shows decisions from both models. The drawn points correspond to the set of data on which the metric is computed, and their colour depends on the decision of the MCD-Kernel model, while the colours in the background describe the probability of a point to belong to a class according to the kNeigborsClassifier. Table II gives the accuracy metric at each step. Note that accuracy decreases through steps, which means that a classification based on the decision becomes harder.

In order to estimate the model quality, the same process has been applied on DyClee and HST for this dataset. Figure 10 shows the result for DyClee with 0.01 as micro-cluster size.

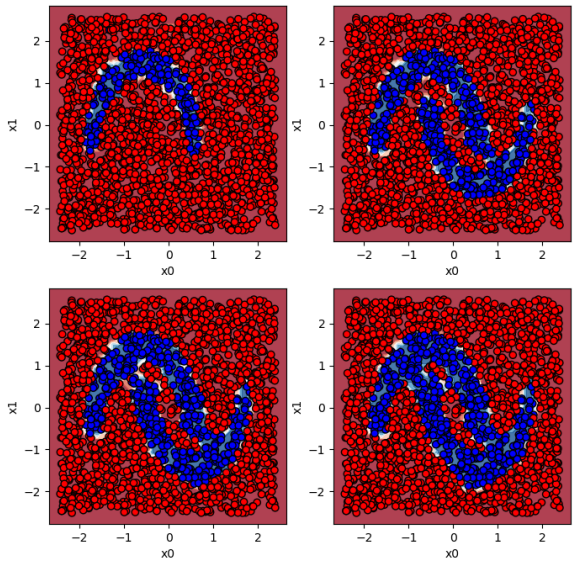

Fig. 9. Results for consistency evaluation of MCD-Kernel on the Two Moons dataset. Blue colour identifies normal points while red colour identifies outliers. (1) Top-left: at this point, only one moon has been processed. (2) Top-right: from this point on, observations from the second moon have been processed by the MCD-Kernel model. (4) Bottom-right: one can observe that, with the multiplication of processed points, the decision takes more points as inliers than on (1), meaning the the model learns novel normal behavior.

TABLE II
MEAN ACCURACY OBTAINED WITH MCD-KERNEL FOR THE DIFFERENT STEPS ON THE TWO MOON DATASET

	Step 1	Step 2	Step 3	Step 4
Mean Accuracy	0.981	0.968	0.9645	0.9635

The results quality is slightly worse than the one with MCD-Kernel with more inliers considered as outliers but less outliers considered as inliers. However, HST performs poorly on this dataset as shown in Figure 11 with 50 trees of height 8 and a window size fixed at 1024. Note HST would probably give better results with a better parameterization, but its parameters are difficult to tune and raise significantly the memory usage.

b) Learning Speed: Here, the goal is to measure how fast a novel normal behaviour is integrated to the model. To be able to do so, the number of detected outliers throughout the incrementation of new points is studied. This test is done on the Two Moons dataset because it is easier to split in two normal clusters. A fast growth can be observed for the first observations but it quickly lessens. However, there is still a more-or-less linear growth after this point. This linear growth means that the model still declares normal points as outliers, possibly at the normal cluster borders. The same experiment has been done for HST, with 50 trees of heights 8 and a window size fixed at 5000. The results are shown in Figure 12 and one can see that MCD-Kernel is faster to learn but detects slightly more outliers than HST after stabilisation, which can

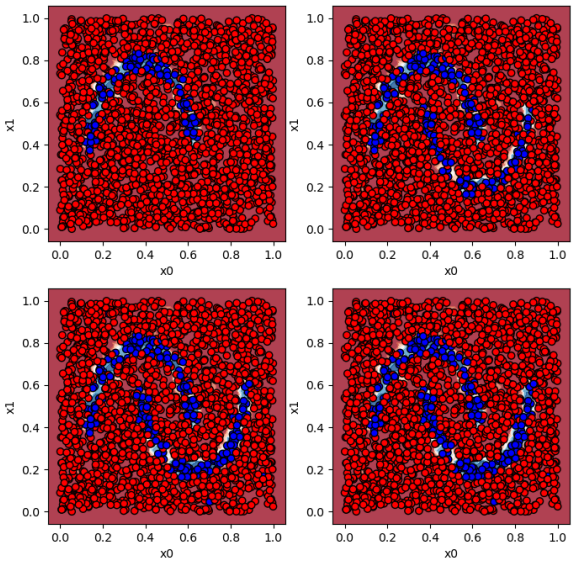

Fig. 10. Results for consistency evaluation of DyClee on the Two Moons dataset. Blue colour identifies normal points while red colour identifies outliers. The denser parts of the moons are considered normal, but the peripheral points are rejected.

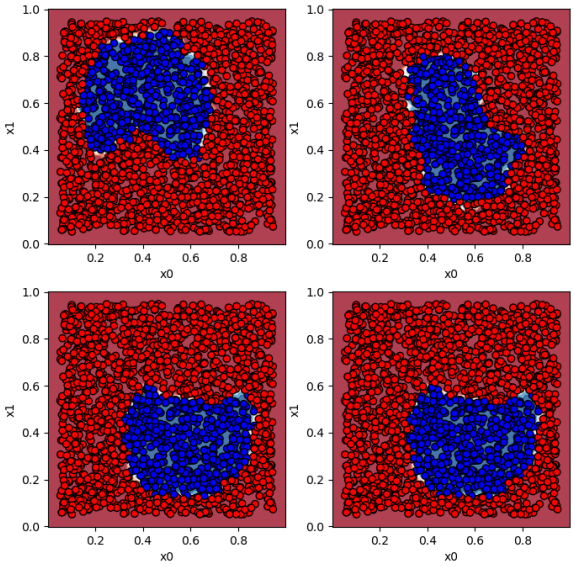

Fig. 11. Results for consistency evaluation of HST on the Two Moons dataset. Blue colour identifies normal points while red colour identifies outliers.

be explained by the large size of normal clusters on Figure 11. Note that far less outliers are detected by MCD-Kernel in the end.

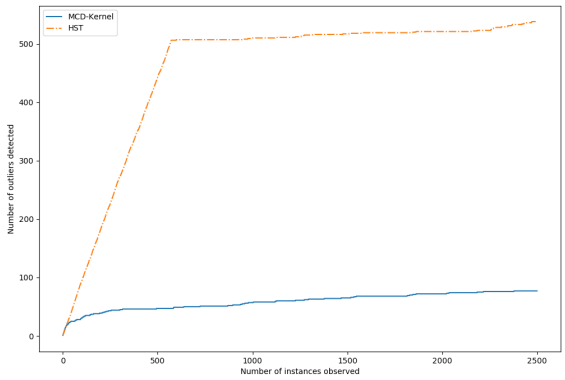

Fig. 12. Number of detected outliers through observations with the MCD-Kernel and HST methods. A good curve would quickly reach its maximum value and this value would be low. A method that would not learn would show linear growth. For both methods, the growth lessens on first observations but then follows linear growth until the end. The growth is nonetheless far more important at the beginning for HST, with a total of more than 500 outliers detected while MCD-Kernel stops at 80 for those 2500 observations.

2) Experiments based on the industrial conveyor dataset:

a) Consistency: The consistency evaluation applied to the synthetic dataset has also been applied to the industrial dataset. Once more, four different timesteps are chosen; the first step follows a training phase on the 100000 first observations, then the rest of the dataset is divided in three parts of approximately equal sizes. Figure 13 shows decisions from the MCD-Kernel model and the supervised classifier while Table III gives the mean accuracy obtained at each step. The results are worse on this dataset, which can be retrieved in the accuracy table. Indeed, points considered as normal are spread in area of space where most of the points are considered outlying.

TABLE III
MEAN ACCURACY OBTAINED WITH MCD-KERNEL FOR THE DIFFERENT STEPS ON THE LUGGAGE CONVEYOR DATASET

	Step 1	Step 2	Step 3	Step 4
Mean Accuracy	0.9485	0.9480	0.9455	0.9555

However, the differences in densities between the different operating modes make learning the normal behaviour a difficult task. To better understand this point, the plots for DyClee and HST have been displayed on Figure 14 and Figure 15 respectively. DyClee partially captures the normal behaviour but miss a lot while HST only captures the main normal modes, *stop* and *standard*.

b) Duration and Memory Usage: The last evaluation criterion refers to processing time and memory usage. This metrics were measured on the execution of the MCD-Kernel method on the industrial conveyor dataset because of its large

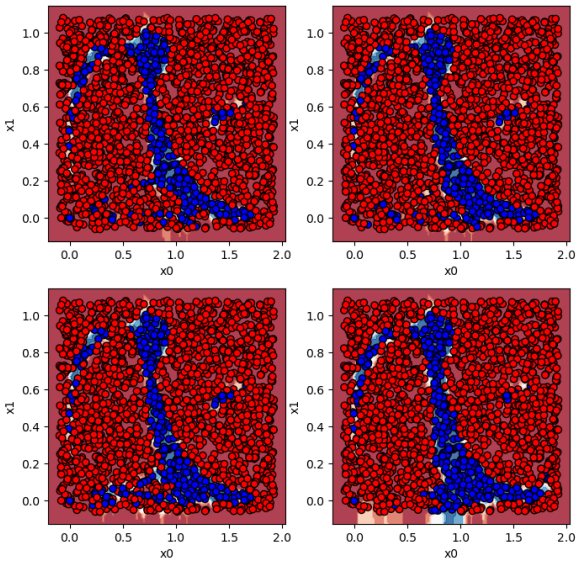

Fig. 13. Results of consistency evaluation of MCD-Kernel on the Luggage Conveyor dataset. Blue colour identifies normal points while red colour identifies outliers. Stop and standard operating mode are captured, as well as a part of starting and stopping transitions. Even some points from the heavy load condition are declared normal.

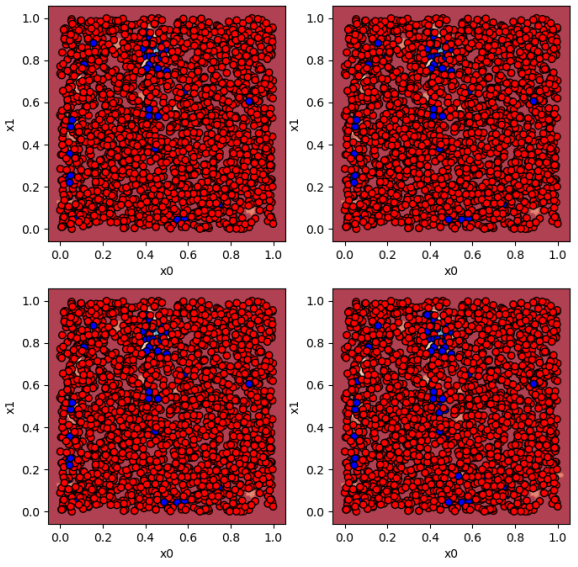

Fig. 14. Results of consistency evaluation of DyClee on the Luggage Conveyor dataset. Blue colour identifies normal points while red colour identifies outliers. Points that are close to each other in areas of uniform densities are captured because of how DyClee works, but a lot of the normal behaviour is missed. A better parameters tuning could give slightly better results.

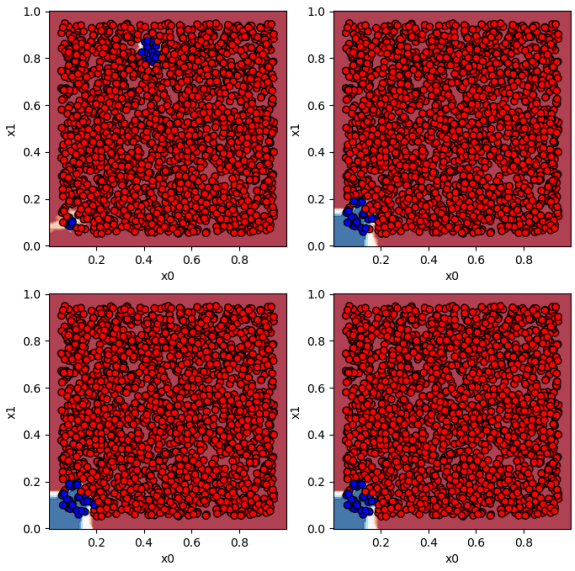

Fig. 15. Results of consistency evaluation of HST on the Luggage Conveyor dataset. Blue colour identifies normal points while red colour identifies outliers. Points from the two main modes are captured on the first step but the standard behaviour then disappear. It can be explained by the ratio of points from this second mode in the window on which the model is based for the three other steps.

size. The training and updating phases were separated for processing time estimation. 100000 points were taken for the training process and the rest of the dataset was used for the updating process. To get a fair estimation, durations were aggregated over 100 executions. Measuring the memory usage during execution was done on Python with the tracemalloc library. The results can be found in Table IV.

TABLE IV
PERFORMANCE METRICS MEASURED FOR THE CC-EVAL BASED METHOD

	Min(time)	Mean(time)	Max(time)	Memory
Training	22.6875s	24.0166s	27.8125s	631Mib
Updating	198.4844s	210.4723s	236.250s	

VII. DISCUSSIONS AND FUTURE WORKS

In this section, two main issues are discussed. The first one refers to the related problems of model persistence and novelty detection, i.e. model adaptation. The second deals with the difficulty to evaluate and compare different methods.

A. Model adaptation

All the outlier detection methods for data streams face the problem of deciding whether the model learned at some point in time should be considered persistent and how new arriving points should update this model. This is a tedious problem that is often tackled with a forgetting factor.

This has been tested with the MCD-Kernel method as explained in IV-B2. During these tests, instabilities in the levelsets were faced. Those were due to negative values of the score, which is theoretically impossible because of the positiveness of $M_d(\mu)$. The cause was identified to be computational approximations in Python that would result in slight negative singular values in the spectral decomposition of the matrix. Even if those values are small, they highly impact the score computation because the score is based on the moment matrix inverse. Different solutions were tested to tackle this issue:

1) Standardizing data. Slight negative singular values only seem to appear when data are not standardized. However, this leads to another issue; how to keep data standardized throughout the stream? Methods exist to keep a data stream standardized, however the model has to follow the transformations, which is pretty difficult. In the end, it seems easier to rebuild the whole model each time negative singular values are detected.
2) Matrix disturbance. Disturbing the matrix by adding $\rho I_{s(d)}$, $I_{s(d)}$ being the identity matrix of the same size $s(d) \times s(d)$ as the moment matrix, can lead to positiveness depending on the chosen value of ρ. This can also be achieved by changing the sign of negative singular values in the singular value decomposition of $M_d(\mu)$. However, those methods cause the support to also capture the origin of the coordinate system which is not desired if points are far from the origin.
3) Precise computation. The last solution that has been tested was to force more precise computations. Using the mpmath Python library, it is possible to force a 128 bit precision on floating numbers. With this precision, the previously mentioned issue disappeared, but at the cost of processing time because of the use of non-built-in types. Combining the Numpy optimization for vectors and matrices computations would reduce time but it stays very long in comparison with the processing time for built-in floating numbers.

Forgetting old behaviour is a crucial aspect since it greatly affects the results. It can be chosen to give as much importance to any point or to reduce the importance of oldest points. In the case all the points are considered equally, old behaviour is not forgotten and, if it appears again, then the model is able to instantly consider it as normal. The other advantage is that it does not require a new parameter to play with the forgetting speed. The second option is to forget oldest points or to assign a weight to these points. In this case, the method is faster to learn new behaviours because each point has more importance in the ensemble of points composing the support. Nevertheless, in most cases it requires fixing a forgetting parameter. For instance, if one wants to have the support defined by a fixed length window, then the size of the window has to be given. In the weighted sum presented in Section IV, a forgetting coefficient γ has to be set. On top of that, this parameter may have a great impact on results.

Conversely, the impact of already seen points must be considered. If a point is taken outside of the support, its score is great and the evolution of this score depending on d is exponential. However, if the same point is seen a second time, then the score is reduced and the evolution is not exponential anymore. This effect is reduced is there is a sufficient number of observations constituting the support, but it stays important.

Another approach to deal with this issue would be to not integrate outliers to the support for further processing. However, in this case, new behaviours that are first considered as outliers would never be included inside the support.

As a matter of fact, this issue really calls for knowledge about the process generating the data. "How persistent the different behavior modes are?" and "Should some novelty be expected?" are the questions to be answered.

B. Evaluation

As discussed in Section II, there is a lack of frameworks to evaluate and compare online outlier methods. For a proper evaluation, a framework must fulfil the following criteria:

1) Be independent of a specific use case so that the results can be generalized to other use cases.
2) Provide the means to compare abilities that are common to any anomaly detection problem on data streams such as memory usage, time complexity or ability to detect new behaviours.
3) Take into account the tuning requirements and be able to get a fair assumption on the quality of results with different parameterizations.
4) Be applicable to any kind of method whether it returns a score or a binary decision.

This paper makes a step forward in the direction of such a framework (cf. section VI-A). Nevertheless, comparison with other methods stay challenging primarily because there is a lack of implementations available. To get a better comparison of the MCD-Kernel method with the state-of-the-art, it would at least be necessary to adapt the implementations of clustering methods for outlier detection. To go further on the last point, future work will consider to implement the MCD-Kernel method in the River project.

VIII. Conclusion

This paper proposes a novel method for outlier detection in data streams based on the CD-kernel borrowed from the theory of approximation and orthogonal polynomials. Unlike existing methods which require a significant tuning effort, this method is deployable with no tuning at all and can hence be applied to different datasets very conveniently. The paper presents the CD-Kernel and the associated Christoffel function framework and how it leads to the definition of an outlier score when considering a discrete measure whose support is a set (or "cloud") of data points. The key properties of the Christoffel function are given and leveraged to obtain the MCD-Kernel method that advantageously does not require tuning. The performance of the MCD-Kernel method are presented with two data sets, the Two Moons synthetic data set and an industrial conveyor dataset. The results are very promising as the method shows to be able to deal with highly non linear distributions and it can be easily framed in an incremental way suitable to data streams. Some comparisons have been performed with two state-of-the art methods, namely, HST and DyClee. They show that the MCD-Kernel method obtains better results with much less parameterization effort. One of the limitations of the MCD-Kernel method may be to deal with dimesional datasets. Future work will consider to benchmark the scalability of the method.

References

[1] J. Zhang and M. Zulkernine, "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection," in *2006 IEEE International Conference on Communications*, vol. 5, Jun. 2006, pp. 2388–2393.

[2] S. Dreiseitl, M. Osl, C. Scheibböck, and M. Binder, "Outlier Detection with One-Class SVMs: An Application to Melanoma Prognosis," *AMIA Annual Symposium Proceedings*, vol. 2010, pp. 172–176, 2010.

[3] N. Malini and M. Pushpa, "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)*, Feb. 2017, pp. 255–258.

[4] F. Y. Edgeworth, "XLI. On discordant observations," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 23, no. 143, pp. 364–375, Apr. 1887.

[5] H. Wang, M. J. Bah, and M. Hammad, "Progress in Outlier Detection Techniques: A Survey," *IEEE Access*, vol. 7, pp. 107 964–108 000, 2019.

[6] D. Hawkins, *Identification of Outliers*, ser. Monographs on Statistics and Applied Probability. Springer Netherlands, 1980.

[7] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[8] M. Salehi and L. Rashidi, "A Survey on Anomaly detection in Evolving Data: [with Application to Forest Fire Risk Prediction]," *ACM SIGKDD Explorations Newsletter*, vol. 20, no. 1, pp. 13–23, May 2018.

[9] P. Thakkar, J. Vala, and V. Prajapati, "Survey on outlier detection in data stream," *Int. J. Comput. Appl*, vol. 136, pp. 13–16, 2016.

[10] L. Tran, L. Fan, and C. Shahabi, "Distance-based outlier detection in data streams," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1089–1100, Aug. 2016.

[11] J. Zhang, "Advancements of outlier detection: a survey," *ICST Transactions on Scalable Information Systems*, vol. 13, no. 1, pp. 1–26, Feb. 2013.

[12] P. Helman and J. Bhangoo, "A statistically based system for prioritizing information exploration under uncertainty," *Ieee transactions on systems, man, and cybernetics-part a: Systems and humans*, vol. 27, no. 4, pp. 449–466, 1997.

[13] H. S. Javitz, A. Valdes *et al.*, "The sri ides statistical anomaly detector," in *IEEE Symposium on Security and Privacy*, 1991, pp. 316–326.

[14] M. Desforges, P. Jacob, and J. Cooper, "Applications of probability density estimation to the detection of abnormal conditions in engineering," *Proceedings of the institution of mechanical engineers, part c: Journal of mechanical engineering science*, vol. 212, no. 8, pp. 687–703, 1998.

[15] T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos, "Distributed deviation detection in sensor networks," *Acm sigmod record*, vol. 32, no. 4, pp. 77–82, 2003.

[16] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ser. CIKM '07. New York, NY, USA: Association for Computing Machinery, Nov. 2007, pp. 811–820.

[17] D. Yang, E. A. Rundensteiner, and M. O. Ward, "Neighbor-based pattern detection for windows over streaming data," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT '09. New York, NY, USA: Association for Computing Machinery, Mar. 2009, pp. 529–540.

[18] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," in *2011 IEEE 27th International Conference on Data Engineering*, Apr. 2011, pp. 135–146.

[19] L. Cao, D. Yang, Q. Wang, Y. Yu, J. Wang, and E. A. Rundensteiner, "Scalable distance-based outlier detection over high-volume data streams," in *2014 IEEE 30th International Conference on Data Engineering*, Mar. 2014, pp. 76–87.

[20] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, Jul. 2011, pp. 1511–1516.

[21] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-Based Anomaly Detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 3:1–3:39, Mar. 2012.

[22] D. Pokrajac, A. Lazarevic, and L. J. Latecki, "Incremental Local Outlier Detection for Data Streams," in *2007 IEEE Symposium on Computational Intelligence and Data Mining*, Mar. 2007, pp. 504–515.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '00. New York, NY, USA: Association for Computing Machinery, May 2000, pp. 93–104.

[24] S. H. Karimian, M. Kelarestaghi, and S. Hashemi, "I-IncLOF: Improved incremental local outlier detection for data streams," in *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*, May 2012, pp. 023–028.

[25] M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan, and X. Zhang, "Fast Memory Efficient Local Outlier Detection in Data Streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 12, pp. 3246–3260, Dec. 2016.

[26] J.-W. Huang, M.-X. Zhong, and B. P. Jaysawal, "TADILOF: Time Aware Density-Based Incremental Local Outlier Detection in Data Streams," *Sensors*, vol. 20, no. 20, p. 5829, Jan. 2020.

[27] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," *ACM SIGMOD Record*, vol. 25, no. 2, pp. 103–114, Jun. 1996.

[28] C. C. Aggarwal, P. S. Yu, J. Han, and J. Wang, "- A Framework for Clustering Evolving Data Streams," in *Proceedings 2003 VLDB Conference*, J.-C. Freytag, P. Lockemann, S. Abiteboul, M. Carey, P. Selinger, and A. Heuer, Eds. San Francisco: Morgan Kaufmann, Jan. 2003, pp. 81–92.

[29] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-Based Clustering over an Evolving Data Stream with Noise," in *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*, ser. Proceedings. Society for Industrial and Applied Mathematics, Apr. 2006, pp. 328–339.

[30] Y. Chen and L. Tu, "Density-based clustering for real-time stream data," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '07. New York, NY, USA: Association for Computing Machinery, Aug. 2007, pp. 133–142.

[31] J. Ren and R. Ma, "Density-Based Data Streams Clustering over Sliding Windows," in *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 5, Aug. 2009, pp. 248–252.

[32] W. Wang, J. Yang, and R. R. Muntz, "STING: A Statistical Information Grid Approach to Spatial Data Mining," in *Proceedings of the 23rd International Conference on Very Large Data Bases*, ser. VLDB '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Aug. 1997, pp. 186–195.

[33] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proceedings 18th International Conference on Data Engineering*, Feb. 2002, pp. 685–694.

[34] M. Hahsler and M. Bolaños, "Clustering Data Streams Based on Shared Density between Micro-Clusters," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1449–1461, Jun. 2016.

[35] N. B. Roa, L. Travé-Massuyès, and V. H. Grisales, "DyClee: Dynamic clustering for tracking evolving environments," *Pattern Recognition*, vol. 94, p. 162, Oct. 2019.

[36] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, and A. Bifet, "River: machine learning for streaming data in Python," *arXiv:2012.04740 [cs]*, Dec. 2020.

[37] J.-B. Lasserre and E. Pauwels, "The empirical Christoffel function with applications in data analysis," *Advances in Computational Mathematics*, vol. 45, no. 3, pp. 1439–1468, Jun. 2019.

[38] J. B. Lasserre, E. Pauwels, and M. Putinar, *The Christoffel-Darboux Kernel for Data Analysis*, ser. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2022.

[39] N. Goix, "How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?" *arXiv:1607.01152 [cs, stat]*, Jul. 2016.

[40] K. Ducharlet, L. Travé-Massuyès, M.-V. Le Lann, and Y. Miloudi, "A Multi-phase Iterative Approach for Anomaly Detection and Its Agnostic Evaluation," in *Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices*, ser. Lecture Notes in Computer Science, H. Fujita, P. Fournier-Viger, M. Ali, and J. Sasaki, Eds. Cham: Springer International Publishing, 2020, pp. 505–517.

[41] A. Lavin and S. Ahmad, "Evaluating Real-Time Anomaly Detection Algorithms – The Numenta Anomaly Benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2015, pp. 38–44.