Online Outlier Detection for Data Streams

Shiblee Sadik
University of Oklahoma
Norman, Oklahoma
shiblee.sadik@ou.edu

Le Gruenwald
University of Oklahoma
Norman, Oklahoma
ggruenwald@ou.edu

ABSTRACT

Outlier detection is a well established area of statistics but most of the existing outlier detection techniques are designed for applications where the entire dataset is available for random access. A typical outlier detection technique constructs a standard data distribution or model and identifies the deviated data points from the model as outliers. Evidently these techniques are not suitable for online data streams where the entire dataset, due to its unbounded volume, is not available for random access. Moreover, the data distribution in data streams change over time which challenges the existing outlier detection techniques that assume a constant standard data distribution for the entire dataset. In addition, data streams are characterized by uncertainty which imposes further complexity. In this paper we propose an adaptive, online outlier detection technique addressing the aforementioned characteristics of data streams, called Adaptive Outlier Detection for Data Streams (A-ODDS), which identifies outliers with respect to all the received data points as well as temporally close data points. The temporally close data points are selected based on time and change of data distribution. We also present an efficient and online implementation of the technique and a performance study showing the superiority of A-ODDS over existing techniques in terms of accuracy and execution time on a real-life dataset collected from meteorological applications.

Keywords

Knowledge Discovery, Data Mining, Stream Databases

1. INTRODUCTION

An outlier is a data point which either is significantly different from other data points, or does not conform to the expected normal behavior, or conforms well to a defined abnormal behavior [7]. Outliers may appear in a dataset for numerous reasons, like malicious activity, instrumental error, setup error, change of environment, human error, catastrophe, etc. Regardless of the reason, outliers are interesting to the user because they entail some interesting information compared to regular data points. Historically, outliers are unavoidable in a data acquisition process [7, 5, 4]; therefore outlier detection has been receiving a great deal of attention for a long time and plenty of techniques are available for statistical data; however, there exist much controversy regarding "exactly what constitutes an outlier" and no well accepted definition exists for outliers. A handful of techniques are available for data streams [3, 6, 10], which are adopted from existing outlier detection techniques for regular data with ad-hoc modifications and do not address all characteristics of data streams. The characteristics, which we describe below, make outlier detection challenging.

Applications for data streams are significantly different from those for regular data in many aspects. In data stream applications, data have the essence of time, are mostly append only and, in many cases, are transient [25]; therefore offline storing and processing approaches are not suitable for data streams; consequently, data processing has to be online and incremental. Data are continuously coming in a streaming environment with a very fast rate and changing data distribution (change of data distribution is known as concept drift) [16], and thus, any fixed data distribution is not adequate to capture the dynamic behavior of data streams. On top of this, the data sources are characterized as unreliable [25] which introduces uncertainty into the collected data and makes processing more complicated. Moreover, the existing outlier detection techniques require user-defined parameters which demands extensive domain knowledge for effective operation over data streams. Due to the dynamic nature of data streams, the parameter values are hard to predict and the chosen parameter values may not be appropriate for the entire lifetime of a data stream; thus the existing techniques still require further improvements.

In this paper, we propose a technique called Adaptive Outlier Detection for Data Streams (A-ODDS) that is based on data deviations in global and local contexts. By global context we refer to all the history data points, and local context we refer to the temporally close data points that are generated from the same data distribution. Since data streams change dynamically, their recent trend may become a rare event in the future; thus the use of global context ensures an outlier is deviated from all existing data points and the use of local context ensures an outlier is deviated from the recent trend of data values. A-ODDS computes two relative deviation factors for each data point with respect to the global context and the local context of the data point. A data point is identified as an outlier if any of the devia-

IDEAS11 2011, September 21-23, Lisbon [Portugal]
Editors: Bernardino, Cruz, Desai

tion factors becomes excessively high. By considering both the global and local contexts, outliers can still be detected despite change in data distribution, and therefore, outlier detection accuracy would be increased. A-ODDS tackles the challenging problem of identifying the local context with the help of our novel, automatic concept drift detection technique. Moreover, A-ODDS includes an adaptive data distribution function which addresses the three key characteristics of data streams: uncertainty, transiency, and incompleteness. To the best of our knowledge, A-ODDS is the only adaptive outlier detection technique designed for data streams which identifies the relatively deviated data points as outliers in global and local contexts.

The rest of the paper is organized as follows: Section 2 discusses the related work; Section 3 describes our technique and its implementation; Section 4 presents the experimental results comparing the performance of A-ODDS with those of the existing techniques; and finally, Section 5 provides our conclusions and future research.

2. RELATED WORK

A number of existing outlier detection techniques for data streams use a sliding window to capture the recent data values and detect the outliers inside the window [3, 6, 26] with multi-pass algorithms. Data streams change over time and an outlier for a particular window may appear as an inlier in another window; hence the notion of outlier in a data stream window is not very concrete. Nevertheless, an inlier can be shown as an outlier by changing the window size [6]; thus the outlier detection techniques that use a sliding window work well if the window size is chosen correctly. Ideally, a sliding window should capture all the data points of the recent trend of the data streams, but practically, the choice of sliding window size is independent of the number of data points in the recent trend; thus the sliding windows do not capture all the data points of the current trend which might cause poor results. Moreover, different techniques interpret the window size differently. In most situations, it is difficult for the domain expert to choose the window size correctly without knowing the interpretation of the sliding window for a particular technique.

Auto-regression based techniques for outlier detection are very popular for time series outlier detection [4]. Some outlier detection techniques for data streams adopt auto-regression [10, 19]. Most of the auto-regression based techniques work similarly in which a data point is compared with an estimated model and a metric is computed based on the comparison. If the metric is beyond a certain limit (called cut-off limit), the data point is identified as an outlier. The advantages of the auto-regression based models are that they are computationally inexpensive compared to distance based outlier detection techniques [3] and they provide an estimated value for an outlier. However, the success of these techniques depends on the quality of the auto-regression models and the efficacy of the cut-off limits. Different data streams show different natures in their changing patterns; therefore it is difficult to select an appropriate auto-regression model for data streams [10]. This problem becomes more severe for dynamically changing data streams since a fixed auto-regression model is not appropriate for them. Moreover, the selection of a magic cut-off point depends upon not only the data but also the auto-regression model chosen. Since the trends of data streams change over time, the same cut-off limit is not appropriate for the entire life cycle of data streams. In addition, if data streams show irregular patterns, the auto-regression models fail to capture the trends and perform very poorly; therefore they are not appropriate when it is difficult to fit the data into an auto-regression model.

Outlier detection techniques for sensor networks and multiple data streams have been proposed in the literature [14, 26]. The underlying assumptions are the availability of multiple homogeneous data streams and their synchronous behavior in many sensor networks. In real life, multiple homogeneous data streams may not be available for all applications or one data stream may behave very differently from the others like credit card transaction monitoring. In the later case, comparing two heterogeneous data streams does not help to point out the outliers in one stream.

Techniques that are statistics based [4, 24] and machine learning based [1] assume a fixed distribution for the dataset. If a data point belongs to a very low probability density region, the data point is more likely to be an outlier. Data streams are highly dynamic in nature and their distribution changes over time [11]. No fixed data distribution is good enough for the entire data stream; hence summarizing a dynamic data stream with a static data distribution produces questionable results. The data points in a data stream may have temporal correlations with one another, but the statistical techniques fail to capture such correlations. Moreover, statistics based techniques assume the entire dataset is available for random access which is not the case for data streams.

Data clustering algorithms produce outliers as a byproduct [7, 17]; but as outlier detection is not the focus of clustering algorithms, they are not optimized for outlier detection. Moreover, the time series clustering based outlier detection techniques that are optimized for outlier detection [23, 2] fail to address concept drift. They require a training phase, build a model and compare each temporal sequence with the captured model. The techniques assume that the trend in the dataset is fixed, which is not true for data streams; hence their efficacy and correctness are still in question.

None of the above outlier detection techniques considers all the data stream characteristics like data uncertainty, concept drift, transiency, temporal correlations among the data points, etc. Moreover, not all the outlier detection algorithms are truly incremental; rather they store a subset of the data points for random access and detect the outlier inside the subset. To fill the gap, previously we proposed a technique to detect outliers for data streams, called Distance Based Outlier Detection for Data Streams (DBOD-DS), with the help of a continuously adaptive data distribution function that addresses the uncertainty, transiency and temporal relationship among the data points [21]. DBOD-DS detects outliers based on two user-defined parameters: neighbor radius (in short, radius) and minimum neighbor density. A data point is identified as an outlier if the neighbor density within the user-defined radius of the data point is smaller than the user-defined minimum neighbor density. However, DBOD-DS does not address concept drift properly which is very common for data streams; hence if the dispersion of the data points changes over time, DBOD-DS fails to adapt to the modified situation. Moreover for DBOD-DS to be effective, the user must have a very good idea about what constitutes an outlier and the dispersion of data points in

order to fix the appropriate settings for the required parameters. However, the same parameter settings may not work after a concept drift occurs which makes DBOD-DS fail to identify some outliers.

The technique we propose in this paper, A-ODDS, is based on relative deviation, hence capable of dynamic adaptation. A-ODDS calculates the relative deviation of a data point to all history data points (deviation in global context) and temporally close data points having the same data distribution (deviation with respect to local context). Thus if the concept drift occurs, A-ODDS immediately detects the change, replaces the local context and starts identifying outliers based on new local context. Hence A-ODDS is adaptive to concept drift. The next section describes A-ODDS in detail.

3. THE PROPOSED TECHNIQUE: A-ODDS

There is no concrete definition of outlier and not everyone and every dataset have the same notion of what constitutes an outlier; therefore there is a reasonable flexibility exists for outlier detection and it is widely acceptable [18]. In our case the most deviated data points constitute outliers. In this section, we first present an overall description of our proposed technique followed by the details of its individual modules and implementation issues.

3.1 Overall Approach

Our approach is based on two deviation factors with respect to the global and local contexts, called Global Deviation Factor (GDF) and Local Deviation Factor (LDF), respectively. GDF represents the deviation of a data point with respect to the entire history data points; and LDF represents the deviation of a data point with respect to the recent data points (the recent data points are selected based on the change of data distribution). Both deviation factors are calculated from neighbor density. The neighbor density of a newly arrived data point is computed on-the-fly from our data distribution function. GDF is the relative distance between the neighbor density of a data point with respect to the average neighbor density of the entire history data points; and LDF is the relative distance between the neighbor density of a data point and the average neighbor density of the recent data points. A data point is identified as an outlier if any of the above deviation factors goes beyond three standard deviations away from its average. The choice of three standard deviations dispersion ensures a significant dispersion of a data point from other data points [4]. In addition it relieves the user from specific cut-off selection for significantly big GDFs and LDFs. The detailed justifications for the choice of three standard deviations are provided in Section 3.3.

3.2 Preliminaries: The Data Distribution Function

A-ODDS uses the same data distribution function DBOD-DS [21] uses which addresses all the characteristics of data streams. In this data distribution function $f(x)$, as each data point d is received, it is updated by the frequency of occurrence of the data value v. Since data points are uncertain, the frequency of occurrence of v is not increased by 1, rather it is increased by p_v. The frequencies of occurrence of other values are also increased by a fraction of $(1-p_v)$. Thus the proposed data distribution function uses a kernel probability density estimator (or kernel estimator in short) which distributes the frequency of occurrence of a data value into other values.

The temporal characteristic of data streams is addressed by weighting the data points based on their freshness. Heuristically, the recent data points have a stronger relationship with the current data point and receive higher weight compared to the old data points [19]. If $(d_1, d_2, \ldots, d_n)$ are the data points with values $(v_1, v_2, \ldots, v_n)$ where d_n is the most recent one and d_1 is the oldest one, the relative weight would be $(\lambda^{n-1}, \lambda^{n-2}, \ldots, 1)$. λ is often called the forgetting factor which is selected by a bootstrapping method used in [15]. Thus in the data distribution function, the frequency of occurrence of v_i is increased by $p_{v_i}\lambda^{i-1}$ and the frequencies of occurrence of other values is increased by a fraction of $(1-p_{v_i})\lambda^{i-1}$ for each occurrence of v_i.

The data distribution function does not assume any particular fixed/standard data distribution; rather it is adjusted on-the-fly. Moreover, the proposed technique uses a variable bandwidth for the kernel estimator which eases the online incremental implementation and makes the data distribution function adaptive to the dispersion of the data points [22].

Formally, if $(v_1, v_2, \ldots, v_n)$ are the data values for data points $(d_1, d_2, \ldots, d_n)$ at time $(T-n+1, T-n+2, \ldots, T)$, the proposed data distribution function becomes the equation 1, where $f_T(x)$ is the data distribution function at time T, $(h_1, h_2, \ldots, h_n)$ are the bandwidths, and $k_{h_i}(x)$ is the kernel function with bandwidth h_i. Equation 1 shows the relation between data distribution and data values.

$$f_T(x) = \frac{\sum_{i=1}^{n} \lambda^{n-i} k_{h_i}(v_i - x)}{\sum_{i=1}^{n} \lambda^{n-i}} = \frac{1}{\omega_T} \sum_{i=1}^{n} \lambda^{n-i} k_{h_i}(v_i - x)$$

$$where \quad \omega_T = \sum_{i=1}^{n} \lambda^{n-i} \tag{1}$$

In Equation 1, ω_T is the weighted counter or sum of the weights at time T. ω_T approximates the appropriate number of data points constituting the data distribution function. The advantage of using this data distribution function is that it addresses the data stream characteristics of uncertainty, transiency and temporal correlation. Interested readers are referred to [21] for further details. In this paper we refer to the current data distribution function by $f(x)$ and current weighted counter by ω that theoretically include all the previous data points received.

3.3 Global Deviation Factor

Definition 1. For the current data point d with value v we define the global deviation factor (GDF) at radius r as:

$$GDF(v, r) = \frac{\hat{N}_g(r) - N(v, r)}{\hat{N}_g(r)}$$

where $\hat{N}_g(r) = \frac{1}{n}\sum_{i=1}^{n} N(v_i, r)$ (n is the number of data points) and it is the average neighbor density of all the history data points at radius r and $N(v, r) = \int_{v-r}^{v+r} f(x)dx$ is the neighbor density of the current data point at radius r.

GDF is defined with respect to all the history data points. GDF is motivated from Multi-granularity Deviation Factor (MDEF) in [18]. MDEF is the relative distance between

neighbor counts of a data point to average neighbor count of all data point in the neighborhood. The neighborhood consists of a set of data point within a user defined radius. Whereas, GDF is the relative distance between the average neighbor densities of all the data points to the neighbor density of the current data point. The GDF of the current data point represents the deviation of the current data point with respect to the entire history data points. If the GDF of the current data point is negative, then the current data point is surrounded by more data points than the average data point. Since the current data point has more than the average neighbor density, it is very unlikely to be an outlier. If the GDF of the current data point is zero, then the current data point has the same neighbor density as the average data point. In that case the current data point follows the regular trend; hence it is not likely to be an outlier. If the GDF of the current data point is positive, then the data point has less neighbor density than the average data point and, thus, is very likely to be an outlier. If the GDF of the current data point is significantly big, we identify the data point to be an outlier.

Inevitably, we have to define what the maximum value (cut-off) of the GDF must be in order for it to be classified as "significantly big." Instead of letting the user define the cut-off, our technique chooses the cut-off by itself based on a statistically sound definition of significance. The GDF of the current data point is called significantly big if the GDF is three standard deviations apart from the average GDF of the history data points [18]. The three standard deviations test is a very popular test for outlier detection. In [4] Barnett and Lewis compiled many popular outlier detection techniques; many of which use the three standard deviations test because it practically separates the rare events from the regular patterns. In normal distribution, almost 99% of the data points are within three standard deviations from the mean. More interesting results can be inferred from Chebyshev's inequality , where it can be proved that irrespective of the underlying data distribution, more than 90% of the data points are within three standard deviations from the mean; [18] includes a detailed proof of this claim. The standard deviation of the GDFs of the dataset at radius r is shown in the equation 2.

$$\sigma_{GDF}(r) = \frac{1}{\hat{N}_g(r)} \sqrt{\frac{\sum_{i=1}^{n} (N(v_i, r) - \hat{N}_g(r))^2}{n}} \qquad (2)$$

where $\sigma_{GDF}(r)$ is the standard deviation of the GDFs for the data points at radius r. Formally, a data point d with the value v is an outlier if $GDF(v, r) > 3\sigma_{GDF}(r)$. More precisely, we call it a global outlier since it is an outlier with respect to the entire history. Since the GDF accounts for the entire history, the more data we have, the more stable and reliable average GDF we have. In our approach we do not wait for the entire dataset to be available to calculate the neighbor density as it is not possible with data streams, rather we do it online based on the data points we have so far. Storing the first two moments of the neighbor density of each data point is sufficient for the online calculation of the average and standard deviation of the neighbor density of the data points.

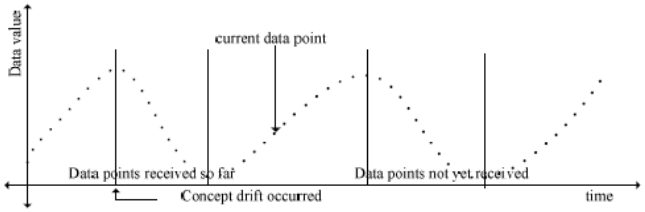

Figure 1: Time series of a data stream

3.4 Local Deviation Factor

Definition 2. For the current data point d with value v we define the local deviation factor (LDF) at radius r as:

$$LDF(v, r) = \frac{\hat{N}_l(r) - N(v, r)}{\hat{N}_l(r)}$$

where $\hat{N}_l(r) = \frac{1}{n} \sum_{i=1}^{n} N(v_i, r)$ (n is the number of data points) and it is the average neighbor density of the data points in the recent concept at radius r.

The following definitions (3- 6) are used to define the term "recent concepts."

Definition 3. A **concept** is a data distribution function which describes the likelihood of a data point to occur with a particular value.

A concept demonstrates the distribution of a set of data points. However, the entire series of data points in a data stream do not follow the same concept, i.e., the data distribution changes, which is known as concept drift [16]. In a data stream, the data points that are temporally close to each other tend to follow the same concept; therefore a data stream can be seen as a series of subsequences where each data point in a subsequence follows the same concept. Figure 1 shows the time series of a data stream. The left side of the current data point shows the already received data points and the right side of the current data point shows the data points not yet received. The vertical lines divide the entire data stream into subsequences. Every data point in a subsequence follows the same concept. Each concept has a starting data point and an ending data point. The timestamp of a concept is obtained from the timestamp of the ending data point.

Definition 4. The concept which is followed by the current data point is called the **current concept**.

The ending data point of the current concept is always the most recent data point arrived; hence the current concept is the most recent one. At any time, there exists only one current concept.

Definition 5. Any concept which has an older timestamp than the current concept is called **history concept**.

The entire stream can be seen as a sequence of history concepts followed by the current concept. The most recent history concept has the most recent timestamp and the least recent history concept has the oldest timestamp.

Definition 6. The recent concepts are constituted from the current concept and the most recent history concept, i.e., the history concept with the most recent timestamp.

The concepts are detected by our novel concept drift detection module, which we describe in Section 3.5. The use of the most recent history concept along with the current concept ensures that there is at least one more data point other than the current data point in the recent concepts. This is necessary because without one more data point, the average neighbor density is undefined which may lead to incorrect results.

A LDF is the relative distance between the average neighbor density of the data points in the recent concepts and the neighbor density of the current data point. The LDF of the current data point represents the deviation with respect to the recent data points. If the LDF of the current data point is negative, then the current data point is surrounded by more data points than the average data point in the recent concepts. Since the current data point has more than the average neighbor density of the data points in the recent concepts, it is very unlikely to be an outlier. If the LDF of the current data point is zero, the current data point has the same neighbor density as the average data point in the recent concepts. In that case the current data point follows the recent trend of the data streams; hence it is not likely to be an outlier. If the LDF of the current data point is positive which implies that the data point has less neighbor density than the average neighbor density in the recent concepts, then the data point is very likely to be an outlier. A data point is identified as an outlier if the LDF of the current data point is three standard deviations apart of the average LDFs of the data points in the recent concepts. The standard deviation of the LDFs of the data points in the recent concepts at radius r is shown in the equation 3.

$$\sigma_{LDF}(r) = \frac{1}{\hat{N}_l(r)} \sqrt{\frac{\sum_{i=1}^{n} (N(v_i, r) - \hat{N}_l(r))^2}{n}} \qquad (3)$$

where $\sigma_{LDF}(r)$ is the standard deviation of the LDFs for the data points in the recent concepts at radius r. Formally, a data point d with the value v is an outlier if $LDF(v, r) > 3\sigma_{LDF}(r)$. More precisely, we call it a local outlier since it is an outlier with respect to the recent data points. To compute LDF we keep updating the local average neighbor density (more precisely, we update the first two moments of the neighbor density of the data points in the recent concepts and those moments are used to calculate the average and standard deviation of the average local neighbor density). Our concept drift detection module keeps track of the change of concepts. If the concept changes, we refresh the local average neighbor density. Each time the concept changes the old history of neighbor density is replaced by the recent history of neighbor density of the data points. Therefore, the LDF is estimated compared to the data points which are temporally close to each other and generated from the same data distribution. If a data point is a local outlier, it means the data point is nonconformist with respect to its temporal neighbors. The concept drift detection module is a major part of A-ODDS and LDF calculation and is discussed in the next section (3.5).

3.5 Concept Drift Detection

Concept drift detection also known as change detection in data streams is a very active research area [13, 8]. Concept drift can be seen as a change of data distribution. Existing techniques compute the distance between new and old data distribution and identify concept drift if it goes beyond a certain threshold [13]. The success of these techniques very much relies on the user's correct selection of the distance-threshold, a burden which does not exist in our technique.

Our concept drift detection module detects changes based on the underlying distribution using the Pearson system [11]. Carl Pearson defined a set of data distribution types called Pearson distribution types and proved that every data distribution can be mapped to an appropriate Pearson distribution type [11]. He also proposed a mathematical framework for mapping known as Pearson system. The Pearson system maps a data distribution to a corresponding Pearson distribution type from the data values. We map our data distribution function to a Pearson distribution type with the help of the Pearson system. In our technique we use a set of equally spaced sample values from our data distribution function and their corresponding frequencies of occurrence. The Pearson system stores the first four moments of each data value and estimates the Pearson distribution type from the moments [11]. As each data point arrives, we update the moments of the sampling data points and approximate the current Pearson distribution type. If the current Pearson distribution type is not the same as the previous Pearson distribution type, we call this a concept drift. The online efficient implementation of the concept drift detection module is discussed in the next section.

3.6 The online Implementation of the Proposed Technique

One of the most important requirements for data stream outlier detection is that the technique has to be online and incremental. Taking this issue into consideration, we design the technique in such a way that the implementation can be done in an online and incremental fashion, which we describe in this section.

3.6.1 The Online Implementation of the Proposed Data Distribution Function

The proposed implementation divides the entire range of data values into equally spaced bins where the range is assumed by application. Each bin is represented by one representing value. The value of the kernel function and the derivative of the kernel function at the representing value are stored for each bin for each data point. The beauty of technique is the distribution function can be computed incrementally on-the-fly. Moreover, the binned implementation eases the neighbor density computation from the data distribution function. We omit the details from this paper due to page limitation; interested readers are referred to [21] for a detail description of the proposed binned implementation.

3.6.2 The Online Concept Drift Detection

We exploit the binned implementation of the proposed data distribution function for the online implementation of the concept drift detection module. As we discussed in Section refsec:cdd, a set of equally spaced sample values and their frequencies of occurrences are fed into the Pearson system to map the data distribution function to an ap-

propriate Pearson distribution type. The representing values of the binned implementation are chosen as the set of equally spaced sample values. As each data point arrives, the data distribution changes; hence the frequency of occurrence changes for the representing values; therefore when we update a bin, we also update the frequencies of occurrence and the moments of the representing value in the Pearson system.

The Pearson system requires updating four moments and the count of sample values. The first moment at time T becomes $m_1^{(T)} = \sum_{i=1}^{binCount} 2\delta b_i f_T(b_i)$ where b_i is the representing value of the i-th bin, 2δ is the bin width, and bincount is the total number of bins in the entire vector space of data values. Therefore, at time $T+1$ the first moment becomes $m_1^{(T+1)} = \sum_{i=1}^{binCount} 2\delta b_i f_{T+1}(b_i)$. Thus,

$$m_1^{(T+1)} - m_1^{(T)} = \sum_{i=1}^{binCount} 2\delta b_i f_{T+1}(b_i) - \sum_{i=1}^{binCount} 2\delta b_i f_T(b_i)$$
$$= \sum_{i=1}^{binCount} \Big(2\delta b_i [\frac{k_{h_{T+1}}(v_{T+1} - b_i)}{1 + \lambda\omega_T}]\Big) - \frac{m_1^{(T)}}{1 + \lambda\omega_T}$$

Therefore,

$$m_1^{(T+1)} = \Big(\frac{\lambda\omega_T}{1 + \lambda\omega_T}\Big) m_1^{(T)} + \Big(\frac{m_1^{(T)}}{1 + \lambda\omega_T}\Big) 2\delta \sum_{i=1}^{binCount} b_i k_{h_{T+1}}(v_{T+1} - b_i),$$

and so,

$$M_1^{(T+1)} = \lambda M_1^{(T)} + 2\delta\Big[\sum_{i=1}^{binCount} b_i k_{h_{T+1}}(v_{T+1} - b_i)\Big] \quad (4)$$
$$where \quad M_1^{(T)} = \omega_T m_1^{(T)}$$

$M_1^{(T)}$ is the first summation which was used to compute the first moment. From equation 4 we can see that the first summation can be calculated online incrementally by adding the values of the kernel function at the representing values to the previous first summation. So, when we calculate the kernel values to update the bins, we store the summation part of equation 4 as well. The other moments (and summations) can be calculated in a similar way; therefore, without adding any significant extra overhead to the algorithm, we can obtain the four summations (otherwise four moments) and approximate the Pearson distribution type of the current data distribution from it with $O(1)$ time complexity. For convenience, we represent the current moments and summations without the time superscript (e.g., the current moments are m_0, m_1, m_2, m_3 and m_4 and the current summations are M_0, M_1, M_2, M_3 and M_4).

We keep the summations, instead of the moments, of the representing values in practice because the summations can be computed incrementally and the moments can be calculated from the summations by arithmetic division. Moreover, the summations can also be computed incrementally by adjusting the frequencies of the recently updated bins only. [20] includes a detailed proof of this claim. As shown in Algorithm 1, our concept drift detection algorithm takes the five summations, M_0, M_1, M_2, M_3, M_4 and weighted counter, ω, as input and calculates the raw moments (lines 2-6). The raw moments are adjusted according to [11] in lines 7-12. The algorithm calculates the current Pearson distribution type and stores it in *curType* in lines 19-41. If the current Peason distribution type (*curType*) is not the same as the previous Pearson distribution type (*prevType*), the algorithm identifies the concept drift.

Algorithm 1 Online concept drift detection algorithm

```
1: procedure isDrift(M_0, M_1, M_2, M_3, M_4, M_5, prevType)
2:     m_0 ← M_0/ω                                  ▷ Raw moment calculation
3:     m_1 ← M_1/(m_0 × ω)
4:     m_2 ← M_2/(m_0 × ω)
5:     m_3 ← M_3/(m_0 × ω)
6:     m_4 ← M_4/(m_0 × ω)
7:     mean ← m_1
8:     m_2 ← m_2 − mean^2                           ▷ Moment adjustment
9:     m_3 ← m_3 − 3 × mean × m_2 − mean^3
10:    m_4 ← m_4 − 4 × mean × m_3 − 6 × mean^2 − mean^4
11:    m_4 ← m_4 − 2 × m_2 + 7.0/240.0
12:    m_2 ← m_2 − 1.0/12.0
13:    σ ← √m_2
14:    β1 ← m_3^2/m_2^3
15:    β2 ← m_4/m_2^2
16:    num ← β1 × (β2 + 3)^2
17:    denum ← 4 × (4β2 − 3β1) × (2β2 − 3β1 − 6)
18:    κ ← num/denum
19:    if denum = 0 then
20:        curType ← III
21:    else
22:        if κ < 0 then
23:            curType ← I
24:        else if κ = 0 then
25:            if β2 < 3 then
26:                curType ← II
27:            else if β2 = 3 then
28:                curType ← N
29:            else
30:                curType ← VII
31:            end if
32:        else if 0 < κ < 1 then
33:            curType ← IV
34:        else if κ = 1 then
35:            curType ← V
36:        else if κ > 1 then
37:            curType ← VI
38:        else
39:            curType ← Unknown
40:        end if
41:    end if
42:    if prevType ≠ curType then
43:        return true
44:    else
45:        return false
46:    end if
47: end procedure
```

3.6.3 The Online Implementation of A-ODDS

Algorithm 2 shows the algorithm for A-ODDS. As each data point arrives, we compute the neighbor density of the data point (line 4). The neighbor density is further used for the computation of the GDF and LDF (lines 5 & 7). The

obtained GDF and LDF are compared with their respective standard deviations (line 6 & 8); a data point is identified as an outlier if any of them goes beyond three standard deviations. Finally the proposed data distribution function is updated in line 14 and the updated data distribution function is mapped to a Pearson distribution type function as a part of concept drift detection (line 15). If the concept drift occurs, we update the average local neighbor density (line 16). Each component of our technique works incrementally. Lines 20-34 present the neighbor density computation algorithm which calculates the area under the curve of the obtained data distribution function. The area computation algorithm takes two parameters (left and right bounding values) as input and adds the areas of the necessary bins within the bounding values (lines 24-29). Each bin is considered as a trapezoid and the total area of all the trapezoids is counted as the area under the curve or neighbor density.

Algorithm 2 A-ODDS

1: **procedure** ISOUTLIER($f(x), v, \hat{N}_g(r), \hat{N}_l(r)$) ▷ v is the data value
2: $left \leftarrow v - r$ ▷ left margin
3: $right \leftarrow v + r$ ▷ right margin
4: $N(v,r) \leftarrow density(f(x), left, right)$ ▷ Compute neighbor density
5: $GDF(v,r) \leftarrow 1 - N(v,r)/\hat{N}_g(r)$
6: $\sigma_{GDF}(r) \leftarrow compute()$ ▷ compute from moment of $N(v,r)$
7: $LDF(v,r) \leftarrow 1 - N(v,r)/\hat{N}_l(r)$
8: $\sigma_{LDF}(r) \leftarrow compute()$
9: $isOutlier \leftarrow false$
10: **if** $GDF(v,r) > 3\sigma_{GDF}(r)$ OR $LDF(v,r) > 3\sigma_{LDF}(r)$ **then**
11: $isOutlier \leftarrow true$
12: **end if**
13: update($\hat{N}_g(r), \hat{N}_l(r)$)
14: update($f(x)$)
15: **if** Concept drift occured **then**
16: update($\hat{N}_l(r)$)
17: **end if**
18: return $isOutlier$
19: **end procedure**
20: **procedure** DENSITY($f(x), left, right$) ▷ takes left and right boundary as input
21: $s \leftarrow left$
22: $e \leftarrow representingValue(left) + binWidth/2$
23: $density \leftarrow 0$
24: **while** $e < right$ **do**
25: $tDensity \leftarrow \int_s^e f(x)$
26: $density \leftarrow density + tDensity$
27: $s \leftarrow e$
28: $e \leftarrow e + binWidth$
29: **end while**
30: $e \leftarrow right$
31: $tDensity \leftarrow \int_s^e f(x)$
32: $density \leftarrow density + tDensity$
33: return $density$
34: **end procedure**

A-ODDS does not require the entire dataset; rather it works online and incrementally. Moreover, it addresses all the issues of data streams: uncertainty, transiency, temporal relations among the data points, and concept drifts. Section 4 presents our performance evaluation.

4. PERFORMANCE EVALUATION

We conducted simulation experiments using a real dataset collected from California Irrigation Management [9] to compare A-ODDS with the three existing algorithms, ART [10], ODTS [6] and DBOD-DS [21], in terms of outlier detection accuracy and execution time. ART is an auto-regression based outlier detection technique which estimates a value using an auto-regression model. A data point is an outlier if the distance between the estimated value and the obtained value is greater than the user-defined threshold. ODTS is a representative technique for sliding window based algorithms. It uses a sliding window to store the recent subset of the data and compares each data point with the median of the subset; if the distance between a data point and the median value is greater than the user-defined threshold, the data point is identified as an outlier. DBOD-DS computes the neighbor density for each data point and identifies a data point as an outlier if its neighbor density is lower than the user-defined threshold. We ran the experiments with the window sizes, 10, 15, 20, ..., 100 for ODTS and reported its average performance. In this section, we first describe the dataset and the simulation model and then present the experimental results.

4.1 Dataset

The California Irrigation Management Information System (CIMIS) manages a network of over 120 automated weather stations in California [9]. Each weather station collects data in every minute and calculates hourly and daily values which are publicly available. The measured attributes are solar radiation, air temperature, soil temperature, etc. For our experiments, we used the daily soil temperature data collected from 1998 to 2009, and implant the random generated synthesized outliers along with inherent outliers. We used soil temperature because it contains some outliers in it. On average, each station has 4000 rounds of data. The first 500 data points were used for bootstrapping and the rest were used for performance evaluation for each data stream.

4.2 Simulation Model

In our simulation model, we mimicked the typical data streams architecture where data are coming from a source and gathered into a server. We executed A-ODDS, ART, ODTS and DBOD-DS, one technique at a time, at the server to detect outliers.

We used the Epanechnikov kernel function [22] with the optimal bin width proposed in [21, 12]; but the choice of the kernel function does not affect the data distribution function very much [22]; hence any type of kernel function can be used for data distribution computation. The bin width we used for data distribution function is the bin width proposed by Fan and James [12]. They propose the use of four hundred bins as they argued that fewer than four hundred bins often deteriorates the correctness of the distribution function and more than four hundred bins offer insignificant improvement.

In our technique, A-ODDS, the user does not have to emphasize too much on the selection of radius which constitutes the neighbor density. This is because the neighbor density is computed for each data point and is compared with the average neighbor densities of the other data points, and only

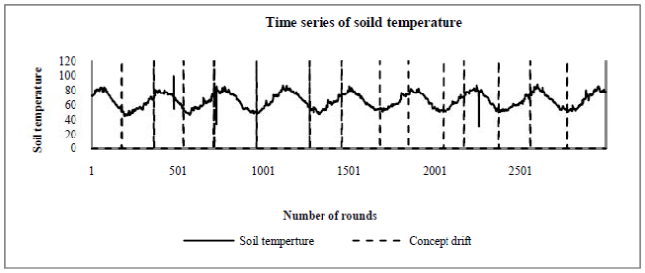

Figure 2: Time series of data points with detected concept drifts

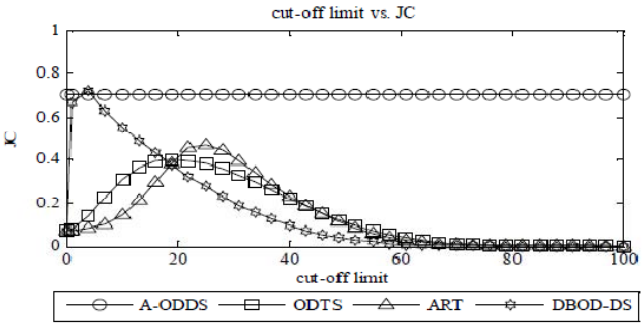

Figure 3: Impacts of user-defined threshold on JC

the relative value is considered for deviation factor calculation; therefore the change of radius does not affect the result much. In our experiments, we calculated the maximum dispersion of the data points from the bootstrapping rounds, and one eighth of the maximum dispersion of the data points is selected as the radius. All radiuses from 1 to one fourth of the maximum dispersion produced similar results and we reported our result using the middle value.

4.3 Experimental Results

The concept drift detection module is a major part of this paper and very important for proper functionality of A-ODDS. In this section, we first present the results of the concept drift detection module, and then present the performance study of A-ODDS.

4.3.1 Results of Concept drift detection

Figure 2 shows the time series of the data points with identified concept drift in it as vertical dashed lines. The time series of the soil temperature shows a periodic trend. The data in Figure 2 clearly show decreasing and increasing trends over time and the dashed vertical lines appear once the series start to increase or decrease; thus our concept drift detection module is capable of identifying the trends most of the times.

4.3.2 Performance study of A-ODDS

We measured the outlier detection complexity in terms of execution time and accuracy in terms of Jaccard Coefficient (JC). The execution time is the time elapsed between when a data point is received and when its outlier-ness is decided (for A-ODDS it also includes the concept drift detection time). JC measures the similarity between the correct set of outliers and the detected set of outliers; moreover JC assigns equal weight to the correct outlier classification and the wrong outlier classification [6]. Execution time is proportional to the complexity of an algorithm. So, higher JC represents higher accuracy and longer execution time represents greater complexity and vice versa.

As A-ODDS does not require the user to select values for its parameters while the other three algorithms do, we conducted experiments comparing the four techniques under different values of their user-defined parameters. Figures 3 shows the JCs of the four techniques with respect to their following user-defined thresholds/cut-off limits: the absolute distance between the estimated value and the true value for ART, the absolute distance between the median and true value for ODTS and the radius for DBOD-DS. A-ODDS uses automatic parameter selection; thus it shows

Table 1: Average JC and Execution Time

	Average accuracy (JC)	Average execution time (ms)
A-ODDS	0.7095	1.3656
DBOD-DS	0.1585	1.8485
ODTS	0.1467	0.0405
ART	0.1373	0.3005

a constant performance with respect to the change of the threshold/cut-off limit. Figure 3 shows that regardless of the threshold/cut-off limit value, A-ODDS always performs better than ART and ODTS in terms of accuracy. It also performs better than DBOD-DS, except when the threshold value of DBOD-DS is very low, in which case DBODS gives a better accuracy but the difference in accuracy between the two techniques is very small.

On average, as shown in Table 4.3.2, A-ODDS gives the best accuracy among all the techniques; its accuracy is 5.16, 4.84 and 4.46 times higher than that of ART, ODTS and DBOD-DS, respectively. The window size and the content of a window in ODTS are independent of the change of data distribution. The auto-regression model in ART fails to cater to the dynamic data distribution; hence their accuracy is poor compared to A-ODDS. DBOD-DS works well only for a very specific choice of radius; hence it also shows poorer average accuracy compared to A-ODDS when examining many different values of radius.

However, when measuring execution time, while A-ODDS requires less time than DBOS-DS, it does take more time than ART and ODTS. In identifying whether a data point is an outlier, ODTS compares the true value of the data point to the median value of the data points in the window and ART compares the true value to the estimated value. The computations of the median value in ODTS and the estimated value using auto-regression in ART are computationally inexpensive compared to the computation of neighbor density in DBOD-DS and A-ODDS; hence the execution time of ODTS and ART is lower than that of DBOD-DS and A-ODDS. Again, the execution time varies with the change of radius for DBOD-DS; a bigger radius requires more bins for density computation and takes more time; thus DBOD-DS has higher execution time than A-ODDS.

Even though A-ODDS has longer execution time compared to ART and ODTS, it takes less than 1.4 milliseconds. This time is very small compared to the time interval between two rounds of data in many data stream applications (e.g. 1 minute in our dataset). Moreover, in [5] the authors

describe a real life application for environment monitoring and argue that a sampling rate of less than two seconds is useless for environment monitoring. Compared to this type of practical applications, the outlier detection time is practically negligible which takes place within two rounds of data.

The above experiments were performed with 7% as the default percentage of outliers. GDF alone can identify 90.29% of them and LDF alone can identify 62.29% of them, while 52.60% of them are identified by both GDF and LDF. To study the impacts of the percentage of outliers on the techniques' performance, we varied the percentage of outliers within the range from 1% to 11%. The experimental results show that the accuracy of each of the techniques is quite resilient with respect to change of percentage of outliers, and A-ODDS consistently gives the best average accuracy. We also conducted experiments to study the impacts of other parameters, such as bin width, number of data rounds, and neighbor density, on the same dataset as well as on two additional datasets. Due to page limitation we omitted the results from this paper; interested readers are referred to [20] for details.

5. CONCLUSIONS AND FUTURE WORK

We have proposed an adaptive outlier detection algorithm for data streams based on our definitions of data deviations with respect to global and local contexts. Our technique is well suited for dynamically changing data streams. The performance of our technique compared with those of the existing techniques is shown by extensive empirical studies on real data. On average our technique performs much better than ART, ODTS and DBOD-DS in terms of accuracy, and while it gives higher execution time than ART and ODTS, its execution time is reasonable for many data stream applications. For future work, we will extend our technique to consider multi-dimensional data and multiple heterogeneous data streams.

6. ACKNOWLEDGMENTS

This work has been supported in part by the NASA under the grants No. NNG05GA30G.

7. REFERENCES

[1] D. Agarwal. An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *ICDM*, pages 26–33, 2005.

[2] S. Ando and E. Suzuki. Detection of unique temporal segments by information theoretic meta-clustering. In *SIGKDD*, pages 59–68, 2009.

[3] F. Anguiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *CIKM*, pages 811 – 820, 2007.

[4] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons Inc, 1994.

[5] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli. The hitchhiker's guide to successful wireless sensor network deployments. In *SenSys*, pages 43–56, 2008.

[6] S. Basu and M. Meckesheimer. Automatic outlier detection for time series: an application to sensor data. *Knowledge Information System*, pages 137 – 154, 2007.

[7] V. Chandola, A. Banarjee, and V. Kumar. Outlier detection: A survey. Technical report, University of Minnesota, 2007.

[8] S. Chen, H. Wang, S. Zhou, and P. Yu. Stop chasing trends: Discovering high order models in evolving data. In *ICDE*, pages 923–932, 2010.

[9] CIMIS. California irrigation management information system, August.

[10] D. Curiac, O. Banias, F. Dragan, C. Volosencu, and O. Dranga. Malicious node detection in wireless sensor networks using an autoregression technique. In *ICNS*, pages 83 – 88, 2007.

[11] W. Elderton and N. Johnson. *System of Frequency Curves*. Cambridge University Press, 1969.

[12] J. Fan and J. S. Fast implementations of nonparametric curve estimators. *Journal of Computational and Graphical Statistics*, pages 35–56, 1994.

[13] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *KDD*, pages 97 – 106, 2001.

[14] K. Ishida and H. Kitagawa. Detecting current outliers: Continuous outlier detection over time-series data streams. In *DEXA*, pages 255 – 268, 2008.

[15] B. T. J., P. J. H. W., and T. R. D. Selecting the forgetting factor in subset autoregressive modelling. *Time Series Analysis*, 23:629–650, 2002.

[16] N. Jiang and L. Gruenwald. Research issues in data stream association rule mining. *ACM SIGMOD Record*, pages 14 – 19, 2006.

[17] R. Ng and H. J. Efficient and effective clustering methods for spatial data mining. In *VLDB*, pages 144 – 155, 2000.

[18] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. Loci: Fast outlier detection using the local correlation. In *ICDE*, pages 315–326, 2003.

[19] V. Puttagunta and K. Kalpakis. Adaptive methods for activity monitoring of streaming data. In *ICMLA*, pages 197–203, 2002.

[20] S. Sadik. Outlier detection for data streams. Master's thesis, University of Oklahoma, 2010.

[21] S. Sadik and L. Gruenwald. Dbod-ds: Distance based outlier detection for data stream. In *DEXA*, pages 122–136, 2010.

[22] D. Scott. *Multivariate Density Estimation*. A Wiley-Interscience Publication, 1992.

[23] K. Sequeira and M. Zaki. Admit: Anomaly-based data mining for intrusions. In *SIGKDD*, pages 386 – 395, 2002.

[24] H. Solberg and A. Lahti. Detection of outliers in reference distributions: Performance of horn's algorithm. *General Clinical Chemistry*, pages 2326–2332, 2005.

[25] M. Stonebraker, Çetintemel U., and S. Zdonik. The 8 requirements of real-time stream processing. *ACM SIGMOD Record*, pages 42–47, 2005.

[26] S. Subramaniam, T. Palpanas, D. Papadoppoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB*, pages 187–199, 2006.