Author/s:
Moshtaghi, M;Erfani, SM;Leckie, C;Bezdek, JC

Title:
Exponentially Weighted Ellipsoidal Model for Anomaly Detection

Date:
2017-09-01

Citation:
Moshtaghi, M., Erfani, S. M., Leckie, C. & Bezdek, J. C. (2017). Exponentially Weighted Ellipsoidal Model for Anomaly Detection. International Journal of Intelligent Systems, 32 (9), pp.881-899. https://doi.org/10.1002/int.21875.

Persistent Link:
https://hdl.handle.net/11343/292369

Exponentially Weighted Ellipsoidal Model for Anomaly Detection

M. Moshtaghi* S. M. Erfani* C. Leckie* J.C. Bezdek*

November 16, 2016

1 Abstract

Efficient localized data modeling techniques in *Internet of Things* (IoT) applications enable the nodes to change their behaviour upon observing events of interest. Additionally, battery-powered IoT nodes can conserve their energy resources by limiting their data communications to specific events. Despite the real-time nature of the data collected in the IoT and limited memory and computational resources, most of the current data modeling approaches for the IoT involve batch training. Recently, an online efficient anomaly detection technique called Iterative Data Capture Anomaly Detection has been proposed for environmental sensing and monitoring applications. However, this approach cannot handle changing environments. So far, efforts in extending this algorithm to adapt to changes in the environment have met with limited success. In this paper, we generalize this algorithm to adapt to changes in the data stream by exponentially weighting past observations. We illustrate the proposed algorithm with numerical results on both real-life and simulated data sets, which demonstrate the efficiency and accuracy of our approach compared to existing methods.

*Department of Computing and Information Systems, The University of Melbourne, AUSTRALIA, E-mail: masud.moshtaghi@unimelb.edu.au, sarah.erfani@unimelb.edu.au, caleckie@unimelb.edu.au, jbezdek@unimelb.edu.au

This is the author manuscript accepted for publication and has undergone full peer review but has not been through the copyediting, typesetting, pagination and proofreading process, which may lead to differences between this version and the Version of Record. Please cite this article as doi: 10.1002/int.21875

2 Introduction

The availability of low-cost sensing technologies prompted the emergence of the *Internet of Things* (IoT) and enabled numerous monitoring applications where environments are equipped with sensing devices that provide real-time information for decision support. These new sensing capabilities necessitate the development of efficient data analysis methods to model normal behaviour and detect events of interest in the distributed data collected by the sensors.[1] Anomaly detection is an important unsupervised data processing task which enables the detection of events of interest as deviations from the normal behaviour without having *a priori* knowledge of possible abnormalities.[1,2] Anomaly detection in resource constrained IoT networks becomes more challenging due to the limited local computational, memory and power resources (mostly battery-operated nodes) of these networks.

Anomaly detection is usually performed by modeling the normal data and then identifying deviations from it.[2] These two tasks can be solved either in *batch* or *online* (*incremental*) modes.[3] Suitable anomaly detection techniques for monitoring applications should match the time-series nature of the data, where the measurements arrive as a continuous stream of (historically) correlated data. In data streaming environments, it is infeasible and actually undesirable to buffer all the measurements for a complete batch analysis. Moreover, the accuracy of batch modeling algorithms depends on proper selection of the initial training period, while the amount of data that can be measured before the system goes into operation is usually not enough to train an accurate model. Micro-batch analysis over sliding windows alleviates some of the problems of batch methods but introduces other problems, including the choice of window size,[3–5] so online detection and modeling in the observed data stream is essential.

The most common online approaches for detecting anomalies in time-series data are based on incremental statistical testing, mainly *cumulative sum* (CUSUM) charts.[3,4,6] These approaches aim to detect mean-shifts either in the raw data or the residuals of a state estimation model.[7] The general multivariate CUSUM developed by Crosier[8] is usually considered the best performing CUSUM-based method over the raw data.

However, this model requires the expected mean and covariance as an input from the user. Mahmoud and Maravelakis[6] have shown that these two parameters can be estimated from the data. However, this model still requires a threshold parameter and re-initialization of the mean and covariance after an anomaly is detected.

The techniques that use the CUSUM analysis of residuals usually make strong assumptions about the data which simplify parameter selection in the CUSUM test. These techniques assume that the dynamics in the data stream can be completely captured by the model and the residual of the model is expected to be random Gaussian noise.[9] The dependence of the detection algorithms on the state estimation model usually increases their complexity and reduces their accuracy in continuously changing environments. High complexity, manual parameter tuning[10] as well as limited detection capabilities are the main shortcomings of the current models.

To overcome the shortcomings of CUSUM-based methods, we have developed[11] an online version of a model-based anomaly detection scheme initially proposed as a batch anomaly detection technique.[12] The online algorithm [11] has simple parameter selection with a clear statistical interpretation and does not need a restart after detecting an anomaly. The detection threshold of this algorithm has a clear statistical interpretation which simplifies the choice of parameters. Another advantage of this algorithm is that it incorporates a *forgetting factor*, which enables the algorithm to track changes in the data. However, the incremental formula this algorithm[11] was based on an approximation of the batch formula for the exponentially weighted sample mean and covariance matrix. This led to instability, which was dealt with through a heuristic parameter.[11]The use of this approximation and the heuristic parameter introduced a small estimation error for each sample which could accumulate over time. This resulted in a loss of estimation quality which in turn affected the accuracy of the algorithm.

In this paper, we derive an incremental update formula from the exact batch formula for the exponentially weighted sample mean and covariance matrix. Solving the exact update formula provides more accurate estimation of these two parameters. We show that the estimated values maintain their expected statistical interpretation and stability. Based on these new formulas we propose a novel online hyperellipsoidal approach

for anomaly detection, called *Exponentially Weighted Iterative Data Capture Anomaly Detection* (ewIDCAD). We also use these update formulas to generalize the multivariate CUSUM method on raw data to obtain an *Exponentially Weighted Multivariate CUSUM* (ewMCUSUM). Numerical comparisons demonstrate better performance of the two new methods based on the proposed incremental update.

The next section is a formal statement of the problem. Section 4 briefly describes our previous method.[11] In Section 5 we present the mathematical details of our approach. In Section 6 we describe the two CUSUM-based methods that form the basis for our comparison. Section 7 contains a numerical evaluation of our method and a comparison to previous approaches. Section 8 contains a discussion and the conclusions of our work.

3 Problem Statement

Our aim is to detect local anomalies in an *online* manner in the multi-dimensional (vector) time-series collected by a node in a resource constrained IoT network. Anomalies can occur at any time $\{t_1,\ldots,t_l,\ldots\}$ in the data. Let $AP_{X_{n,k}} = x_{t_1},\ldots,x_{t_l}$ be set of anomalies observed up to time k, where $t_l \leq k$. To detect anomalies, we need an accurate model of the normal behaviour that captures the characteristics of the data. The model of the normal behaviour in online anomaly detection systems yields a decision threshold $\delta_k(x|\phi_k)$ at time k, where ϕ_k contains the parameters of the model. This paper focuses on techniques with online modeling and decision making capabilities. In these techniques the decision about any point x_k is made at the time of observation and the parameters of the model are updated at the same time using a function f which depends on the new observation and the updated parameters of the model from the last observation, $\phi_k = f(\phi_{k-1}, x_k)$. Since decisions about anomalous samples are based on the data that has so far been observed, normal changes and small deviations in the time-series may result in anomalies according to the model while it learns these changes. A retrospective analysis can be performed on $AP_{X_{n,k}}$ to reduce the number of anomalies; however this is outside the scope of this paper.

According to our problem statement, an online algorithm requires: (1) A parametric model definition, i.e., definition of ϕ_k; (2) A definition of the update function f; and (3) A definition of the decision threshold δ_k. In this paper, we describe these three requirements using a family of ellipsoidal models. In the next section, we describe our previous work[13] in extending batch ellipsoidal models to online models.

4 IDCAD

We start this section with a brief description of a batch anomaly detection algorithm proposed by Rajasegarar *et al.*[12] Let $\{x_1, x_2, \ldots, x_k\}$ be the first k samples at times $\{1, 2, \ldots, k\}$, where each sample is a $p \times 1$ vector in $\Re^p$. The parameters of the hyperellipsoidal models in this algorithm[12] are the sample mean m_k and inverse of the sample covariance matrix S_k^{-1}.

The hyperellipsoid of effective radius t centered at the sample mean of the first k data, viz. m_k, with covariance matrix S_k is defined as

$$e_k(m_k, S_k^{-1}; t) = \left\{ x \in \Re^p | (x - m_k) S_k^{-1} (x - m_k)^T \leq t^2 \right\}. \quad (1)$$

Remark: $(x - m_k) S_k^{-1} (x - m_k)^T$ is the Mahalonobis distance from x to m_k and S_k^{-1} is the matrix of the hyperellipsoid e_k. The boundary of this hyperellipsoid is defined as

$$\delta_{e_k}(m_k, S_k^{-1}; t) = \left\{ x \in \Re^p | (x - m_k)^T S_k^{-1} (x - m_k) = t^2 \right\}. \quad (2)$$

Remark 2: Using $t^2 = (\chi^2)_p^{-1}(\gamma)$ (i.e., the inverse of the chi-squared statistic with p-degrees of freedom) results in a hyperellipsoidal boundary that covers at least $100\gamma\%$ of the data under the assumption that the data has a normal distribution.[14]

Definition 1 - We define a *single point first order anomaly* with respect to e_k as any data vector $x \in \Re^p$ that is outside it:

$$\mathbf{x} \text{ is anomalous for } e_k \Leftrightarrow (x - m_k)^T S_k^{-1} (x - m_k) > t^2. \quad (3)$$

Incremental Data Capture Anomaly Detection (IDCAD) is an online version of the algorithm proposed by Rajasegarar *et al.*, where the incremental update function $f(m_k, S_k^{-1}; x)$ is defined for the ellipsoid e_k. The details and characteristics of IDCAD are described elsewhere.[11,13] The good features of this algorithm are its simplicity, speed of the updates and its ellipsoidal interpretation, which can be used for data summarization. Next, we briefly review the details of this algorithm.

The parameters of the ellipsoid e_k in (2), i.e., m_k and S_k^{-1}, can be calculated using the following incremental update formulas.[13]

$$m_{k+1} = m_k + \frac{1}{k+1}(x_{k+1} - m_k); \tag{4}$$

$$S_{k+1}^{-1} = \frac{kS_k^{-1}}{k-1}\left[I - \frac{(x_{k+1} - m_k)(x_{k+1} - m_k)^T S_k^{-1}}{\frac{k^2-1}{k} + (x_{k+1} - m_k)^T S_k^{-1}(x_{k+1} - m_k)}\right]. \tag{5}$$

While this anomaly detection method works well in static environments and terminates rapidly, the algorithm loses its detection capabilities with changes in the underlying distribution of the data. To cope with changing environments, an exponential forgetting factor $0 < \lambda < 1$ for older measurements is inserted into the incremental formulas (4) and (5). This type of exponential forgetting is widely used in the estimation literature.[15] The *Exponential Moving Average* (EMA) estimation of the sample mean shown in (6) enables exponential forgetting of older samples ($k > 2$).

$$m_{k+1,\lambda} = \lambda m_{k,\lambda} + (1-\lambda)x_{k+1} \tag{6}$$

The forgetting factor in the incremental formula for S^{-1} used in this previous method[13] is an approximation for the weighted sample covariance with exponential forgetting factor λ. This approximation is shown for $k > 1$ samples in (7):

$$S_{k,\lambda} = \frac{1}{k-1}\sum_{i=1}^{k}(x_i - m_{k,\lambda})(x_i - m_{k,\lambda})^T \lambda^{k-i}. \tag{7}$$

This formula uses the multiplier $1/(k-1)$ instead of $\frac{\sum_{i=1}^{k}\lambda^{k-i}}{(\sum_{i=1}^{k}\lambda^{k-i})^2 - \sum_{k=1}^{k}\lambda^{2(k-i)}}$, the multiplier for the unbiased estimator of the covariance matrix with exponential weights

for samples. This approximation simplifies the calculation of the direct incremental update formula for S^{-1}. The direct incremental update formula that is calculated from (7) is

$$S^{-1}_{k+1,\lambda} = \frac{kS^{-1}_{k,\lambda}}{\lambda(k-1)} \times \left[I - \frac{(x_{k+1} - m_{k,\lambda})(x_{k+1} - m_{k,\lambda})^T S^{-1}_{k,\lambda}}{\frac{(k-1)}{\lambda} + (x_{k+1} - m_{k,\lambda})^T S^{-1}_{k,\lambda} (x_{k+1} - m_{k,\lambda})} \right]. \quad (8)$$

The update formula in (8) becomes unstable as $k \longrightarrow \infty$. The main reason is that exponential forgetting does not increase the number samples used to build S^{-1} by one. The forgetting factor slightly reduces the effect of the previously observed samples after each update. Intuitively, this means that each new observation increases k by a number less than 1. Previously[11] a heuristic parameter called *effective N* was introduced to prevent this type of instability in the algorithm for large data streams. This heuristic caps the growth of k to $3/(1-\lambda)$. The intuition behind this heuristic cap is that after the cap is reached, the weight of samples that are removed from sequential algorithm using the forgetting factor is approximately 1, so k stays unchanged from this point. This introduces errors in the sequential update formula before and after attaining $3/(1-\lambda)$. First when $k < 3/(1-\lambda)$, each new observation has an effect of less than 1. The correct amount of increase in k is not accounted for in the sequential formula. When $k \geq 3/(1-\lambda)$, each sample has still an effect on K. The effect only goes to 0 in the limit. As a result the algorithm shows sensitivity to the value of the forgetting factor and it has a higher than expected number of false alarms with respect to the parameters which are chosen according to *Remark 2*. These issues impact the practicality of the previous approach. In the next section, we formulate an iterative derivation from the exact batch formula to solve these problems.

5 Exponentially Weighted IDCAD

We first introduce an exponential forgetting factor so that the input vector x_{k-j} from j samples ago has a weight of λ^j. The batch formulas for calculation of this exponential

weighting scheme for the sample mean and covariance matrix at step k are,

$$m_{k,\lambda} = \frac{\sum_{i=1}^{k} \lambda^{k-i} x_i}{\sum_{i=1}^{k} \lambda^{k-i}}, \tag{9}$$

$$S_k = \frac{\sum_{i=1}^{k} \lambda^{k-i}}{(\sum_{i=1}^{k} \lambda^{k-i})^2 - \sum_{i=1}^{k} \lambda^{2(k-i)}} \times \sum_{i=1}^{k} \lambda^{k-i} (x_i - m_{i,\lambda})(x_i - m_{i,\lambda})^T. \tag{10}$$

We denote $\alpha_k = \sum_{i=1}^{k} \lambda^{k-i}$ and $\beta_k = \sum_{i=1}^{k} \lambda^{2(k-i)}$. By re-arranging (9) and (10), we can write the update formulas for the weighted sample mean and covariance matrix at time k based on their previous values and an update value. Equations (11) and (12) show the one step update formulas for these two parameters.

$$m_{k,\lambda} = m_{k-1,\lambda} + \frac{x_k - m_{k-1,\lambda}}{\alpha_k}, \tag{11}$$

$$S_{k,\lambda} = \chi_k \left[\frac{\lambda}{\chi_{k-1}} S_{k-1,\lambda} + (x_k - m_{k,\lambda})(x_k - m_{k,\lambda})^T \right], \ \chi_k = \frac{\alpha_k}{\alpha_k^2 - \beta_k} \tag{12}$$

where $\alpha_k = \lambda \alpha_{k-1} + 1$ and $\beta_k = \lambda^2 \beta_{k-1} + 1$. If (3) is used to define anomalies, we need $S_{k,\lambda}^{-1}$. We use the matrix inverse lemma in (13) for the inverse of the sum of two matrices to calculate the direct update formula for $S_{k,\lambda}^{-1}$. The assumption in this equation is that E is invertible and B is a square matrix that matches the incremental covariance matrix formula in (12). In our case E is a number and C and D are vectors. Applying this lemma to (12) results in (14).

$$(B + CED)^{-1} = B^{-1} - B^{-1}C(E^{-1} + DB^{-1}C)^{-1}DB^{-1} \tag{13}$$

Define $A_{k-1} = S_{k-1,\lambda}^{-1} / (\lambda \chi_{k-1})$. Then we have

$$S_{k,\lambda}^{-1} = \frac{1}{\chi_k} \times \left[A_{k-1} - \frac{A_{k-1}(x_k - m_{k,\lambda})(x_k - m_{k,\lambda})^T A_{k-1}}{1 + (x_k - m_{k,\lambda})^T A_{k-1} (x_k - m_{k,\lambda})} \right] \tag{14}$$

Comparing equations (11) and (14) with equations (6) and (8) shows that the main

difference between the two sets of formulas are in the way the induced weights by the forgetting factor are used in the updates. Moreover, (14) is based on a principled theoretical foundation, whereas the previous method uses (8) which is based on a more heuristic model.

The pseudo-code in Algorithm 1 outlines the steps of the ewIDCAD algorithm. At any step, the algorithm stores the parameters of the ellipsoids from the last step and two incrementally updated values α and β. The user-defined inputs are the forgetting factor λ and the probability of misclassification of a normal sample γ. The suggested range for λ in the estimation theory literature is $[0.9,\ 1)$.[13] This parameter controls how much of the past data is still relevant. An incremental algorithm with an exponential forgetting factor λ is said to have a *memory horizon* of $\tau = \frac{1}{1-\lambda}$. The data in the memory horizon will be most relevant to the current estimation and the data is said to be completely forgotten after 3τ. In this paper, we use $\lambda = 0.95$ which gives the algorithm a memory horizon of 20 samples. This parameter is normally chosen based on the sampling rate in the time-series. With a high sampling rate a value closer to 1 is chosen as more samples might be relevant to the current observation.

The second parameter, i.e., γ, directly affects the detection performance of the algorithm. Our experiments indicate that γ should be chosen in the interval $[0.85,\ 0.99]$. Higher values reduce the possibility of false alarms but may adversely affect the detection rate of the algorithm. The choice of this parameter depends on balancing the competing goals of low false alarm rate and high detection rate.

Input : x_{k+1}, the parameters from the last step $S^{-1}_{k,\lambda}$, α_k, β_k, $m_{k,\lambda}$, Forgetting factor λ, probability of misclassification γ
Output: $S^{-1}_{k+1,\lambda}$,α_{k+1},β_{k+1},$m_{k+1,\lambda}$, Anomaly Flag f
f=true;
$t = (x_{k+1} - m_{k,\lambda})^T S^{-1}_{k,\lambda}(x_{k+1} - m_{k,\lambda})$;
if $t < (\chi^2)^{-1}_p(\gamma)$ **then** f=false ;
$\alpha_{k+1} = \lambda\,\alpha_{k-1} + 1$;
$\beta_{k+1} = \lambda^2\beta_{k-1} + 1$;
$\chi_{k+1} = \frac{\alpha_{k+1}}{\alpha^2_{k+1} - \beta_{k+1}}$;
Compute $m_{k+1,\lambda}$ using (11);
Compute $S^{-1}_{k+1,\lambda}$ using (14);

Algorithm 1: ewIDCAD algorithm for input x_{k+1}

6 Comparison Algorithms - Online Anomaly Detection

In this section, we describe two prominent online approaches for anomaly detection in multivariate time-series. We compare our proposed ewIDCAD with these two methods in Section 7.

6.1 Multivariate CUSUM

A number of different multivariate CUSUM algorithms exist in the literature but generally the following *multivariate CUSUM* algorithm by Crosier[8] is the preferred option. This algorithm considers the statistic

$$T_k = (T_{k-1} + x_k - \mu)(1 - l/C_k), \;\; if\, C_k > l \tag{15}$$

where $C_k = \left[(T_{k-1} + x_k - \mu)^T \Sigma^{-1} (T_{k-1} + x_k - \mu)\right]^{1/2}$ and $T_0 = 0$. Let

$$Y_k = [T_k^T \Sigma^{-1} T_k]^{1/2}. \tag{16}$$

In this algorithm, the vector of the reference mean μ, the covariance matrix Σ, h and l are set by the user. Crosier suggested $l = \sqrt{(a-\mu)^T \Sigma^{-1} (a-\mu)}/2$ to detect a shift of mean significant enough to result in a new mean vector a. In $p = 2$ dimensions, $l = 1.41$ is similar to the well-known 3σ rule for one dimensional data.[8] The MCUSUM algorithm signals change when $Y_k > h$

In this paper, we use estimated values for Crosier's mean and covariance matrix. These estimations are calculated iteratively using the equations in (4) and (5) and are reinitialized after each anomaly is declared. To continue the algorithm after an anomaly we maintain a buffer of the 10 most recent data points and use this buffer to reinitialize m and S^{-1}. We also introduce a new ewMCUCUM approach that uses the update formulas from ewIDCAD to maintain these two parameters.

6.2 Auto-regression with CUSUM

In data streaming analysis it is common to estimate a dynamic prediction model for one of the features and use residual analysis to detect change or anomalies in the data stream. The assumption is that the residual of the prediction model is white Gaussian noise. This assumption simplifies the selection of the parameters in the CUSUM algorithm for residual analysis. Ross *et al.*[9] proposed iterative estimation of an auto-regressive model using *Recursive Least Square* (*RLS*) and used the CUSUM algorithm over the residuals for online detection of anomalies. In this paper, we call this algorithm *auto-regressive CUSUM* (ARCUSUM). Similar to Ross *et al.*[9] and using procedures recommended by Brown *et al.*,[16] we iteratively build an ARX (AutoRegressive model with eXternal inputs) model of order $n_p = 4$ (using the same order for all the features). We estimate a one-step prediction model for one of the features y and use the other features as the inputs (stimulus signal) using RLS, and apply CUSUM on the residuals of the model to find changes in the data stream.

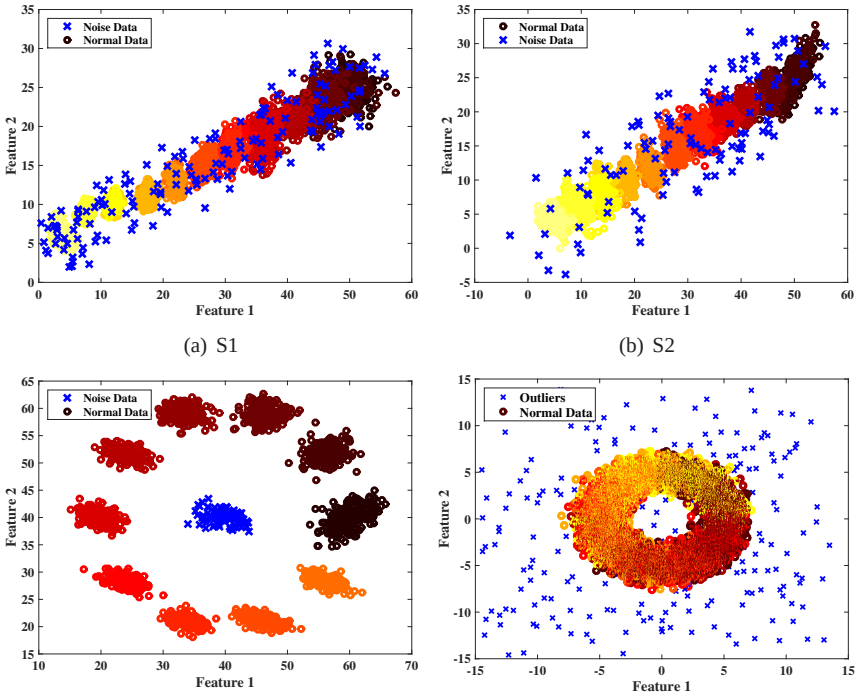
We briefly explain the CUSUM test proposed by Brown *et al.* for measuring the constancy of a regression model over time. Let E_k be the sum of squared residuals, ε_k $(y_k - \hat{y}_k)$ be the prediction error and X_k be a $k \times p$ matrix of the first k samples in the time-series. We calculate three quantities w_k, E_k and W_k as follows:

$$w_k = \frac{\varepsilon_k}{\sqrt{1 + x_k^T (X_k^T X_k)^{-1} x_k}} \tag{17}$$

$$E_k = E_{k-1} + w_k^2 \tag{18}$$

$$W_k = \left(\frac{1}{\sqrt{E_k/k - 1}}\right) \sum_{i=1}^{k} w_i \tag{19}$$

The value of $(X_k^T X_k)^{-1}$ in (17) is estimated iteratively as part of the RLS estimation. An anomaly is detected by the algorithm if W_k exceeds a threshold. In our experiments, x_k is declared anomalous when $|W_k| > R\sqrt{k}$, as recommended by Brown *et al.*[16] The value of R can be calculated based on a measure resembling the significance-level of



(a) S1 (b) S2

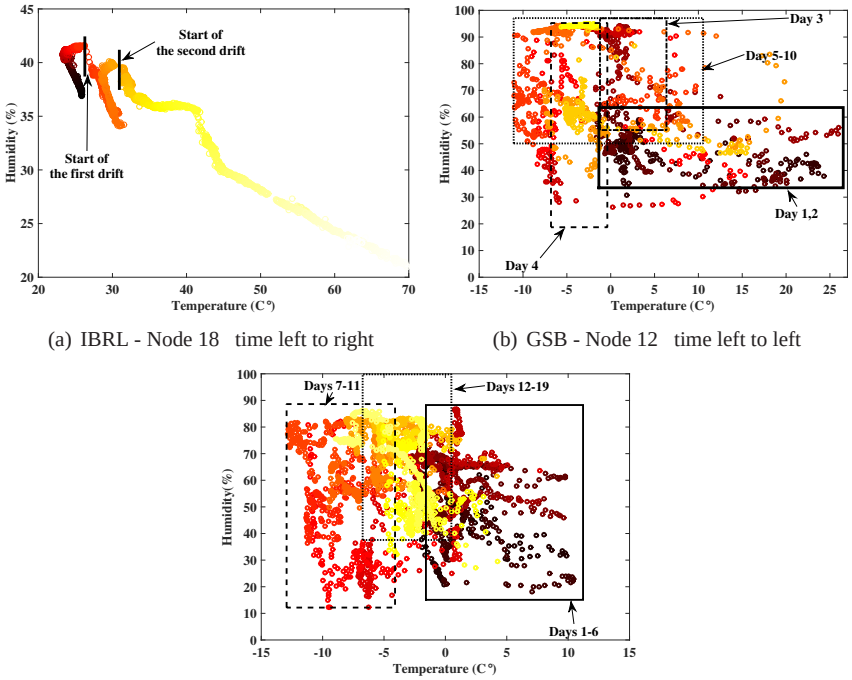
(c) S3 (d) S4

Figure 1: Scatter plots of the synthetic data $S1 - S4$: color shows the progression of time, starting with black and getting lighter colored as time moves forward.

the statistical test. For example, to reach significance levels of 0.01, 0.05 and 0.1 in the results, R has to be set to 1.143, 0.948 and 0.850 respectively.

7 Evaluation

We measure the performance of the algorithms for detecting two different kinds of anomalies, one-off anomalies (outliers or point anomalies) and significant changes for the data streams (sometimes called collective anomalies or concept drift). Section 7.1 contains information about our data sets and the expected anomalies. In Section 7.2, we compare our algorithm based on the exact formulas with the algorithm in our earlier work IDCAD.[13] In the subsequent sections, we compare ewIDCAD and ewMCUSUM with two widely used techniques over both synthetic and real-life data sets.



(a) IBRL - Node 18 time left to right

(b) GSB - Node 12 time left to left

(c) LG - Node 18 time left to left

Figure 2: Scatter plots of three of the real data sets. Color shows the progression of time, starting with black and getting lighter colored as time moves forward.

7.1 Data sets

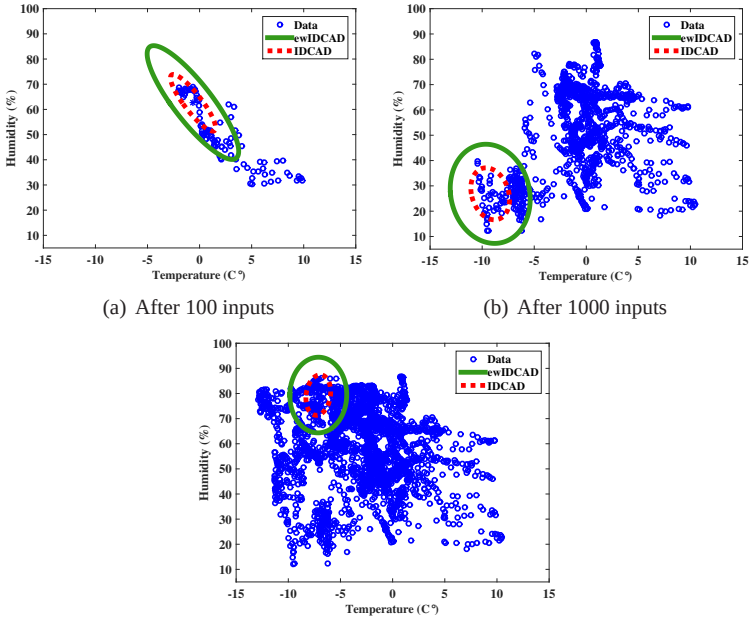
We consider four synthetic and four real-life data sets. Seven of the data sets are two dimensional. The eighth data set has p = 8 dimensional input vectors. We generate three of our synthetic data sets using Gaussian distributions with different levels and types of noise. Fig. 1 shows scatter plots of the four synthetic data sets. The synthetic data sets S1 and S2 are generated by shifting Gaussian distributions. We generate 200 samples from a Gaussian distribution before each shift and then add noise to a small number (N_s) of randomly chosen samples. The number N_s is chosen randomly between 1 and 20. We choose two levels of noise from uniform distributions on: $[-4,\ 4]$ for S1 and $[-8,\ 8]$ for S2. The third synthetic data set, S3, is generated by drawing samples from Gaussian distributions that rotate around a circle with 10 equal shifts. Before each shift, 200 samples are generated using the current Gaussian. In this data set, the noise (the blue cluster in the center of Fig. 1(c)) is generated by a Gaussian at the center of the

circle. At each of 10 steps a random number of samples between 1 and 20 are removed from the outer distribution and then this number of samples are drawn from and added to the inner noise distribution. Note that in S1 and S2, the noise is imbedded among the normal points, so any effort to separate them from the normal data will result in a very high false alarm rate.

The fourth synthetic data set, S4, is generated by considering a multimodal distribution that drifts over time. To generate the data set we use the *Dd_tools* package in Matlab by Tax.[17] First, we generate a Banana shaped distribution and draw 2000 normal samples from the distribution. Then 40 outlier samples are generated using the function *gendatblockout()* in *Dd_tools*, which uses a block-shaped uniform distribution to add outliers to the data. We randomly spread the outliers among the 2000 samples. Then we rotate the Banana distribution 90 degrees and generate 2040 more samples using the same approach. In this data set, we rotate the distribution 5 times, ending up with a total of 12,240 samples. The scatter plot of this data set is shown in Fig. 1(d). This data set enables us to study how different algorithms cope with evolving complex non-Gaussian distributions.

Three of the real-life data sets are from *Wireless Sensor Networks* (WSNs) measuring temperature and humidity: the Intel Berkeley Research Lab (IBRL) data;[18] the Grand-St-Bernard (GSB) data;[19] and the Le Genepi (LG) data.[19] For IBRL, we extracted data from epochs 25000 to 28800 of node 18. In this period, there is a small drift in the temperature and humidity readings. The node recovers from the drift for a short period and then temperature and humidity both resume a long drift away from normal operation. This behaviour is shown in Fig. 2(a). The GSB and LG data sets are both weather station data collected in mountainous areas and the samples are roughly 2 minutes apart. We consider the average surface temperature and relative humidity over 10-minute intervals for 10 days of data from GSB at node 12 and 19 days at node 18 in LG data set during October 2007. The scatter plots of these data sets are shown in Fig. 2. The known significant changes in these data sets are mainly caused by snowy periods during the experiments. According to the set of images from a GPRS-enabled camera from the Le Genepi site there are two snowy periods on the 18th, and on the

late 29th and 30th of October and there are no other significant phenomena in this period. The GSB data set is not complemented with photos from a camera but there is a known snowy period around 20 October at the site.[20] The fourth real-life data set is a gas sensor array under dynamic gas mixtures from the UCI repository and it contains continuous observations obtained from 16 sensors (4 unique sensors), which measure concentrations levels of three target gases changing at random times during 12 hours. We consider the average values of the signals over one second from two pairs of each unique sensor. This results in 8-dimensional data vectors. We select 5 minutes of the data (which generate a stream of 300 eight dimensional input vectors) where the sensors are exposed to two different concentrations of gases CO and Ethylene to illustrate how the algorithms track changes in the input stream.



(a) After 100 inputs (b) After 1000 inputs

(c) After inputs

Figure 3: IDCAD and ewIDCAD ellipsoids for $\gamma = 0.98$ at different times in the GSB input sequence.

7.2 Comparison between IDCAD and ewIDCAD

In general, incremental updating algorithms cannot tolerate a buildup of round-off errors if large (possibly unbounded) numbers of updates are to be performed. So it is

important to avoid any approximations, specially in large streams of data. The ewIDCAD algorithm uses exact update formulas to remove the heuristic parameter in the IDCAD algorithm with forgetting factor and to improve the statistical properties of the algorithm.

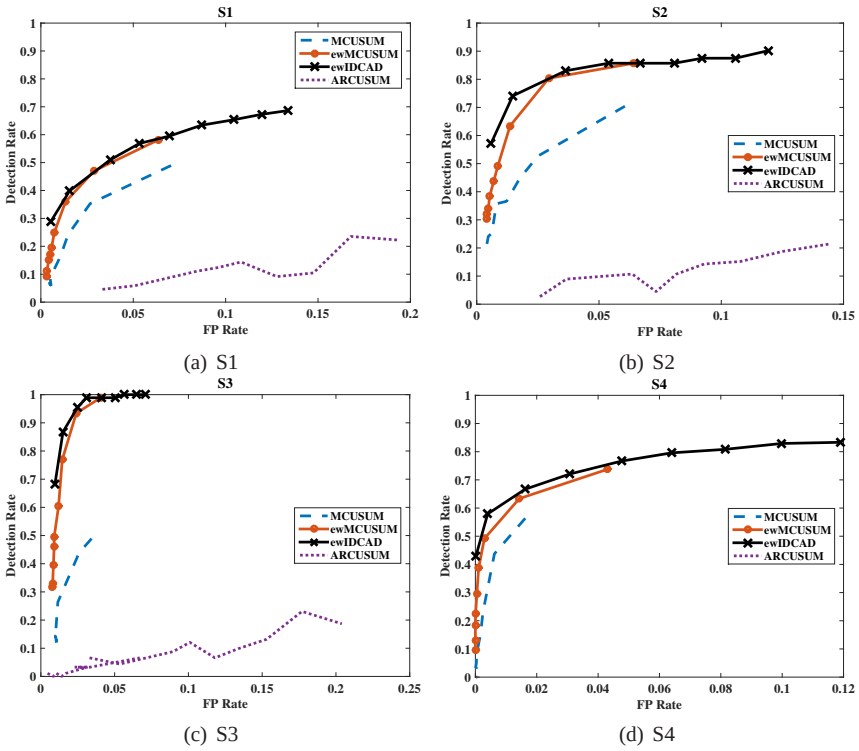
To illustrate the effect of the approximations on IDCAD, Fig. 3 shows the ellipsoids produced by ewIDCAD and IDCAD for the same parameter $\gamma = 0.98$ over the GSB data set at different times during the experiment. In all the steps, IDCAD underestimates the radius of the ellipsoids, however with only small variations in the main directions of the ellipsoidal boundaries. To further demonstrate the effect of underestimation by the IDCAD method we compared the two methods on the four Synthetic data sets with the same $\gamma = 0.99$. The expected false positive rate with this parameter is 1%. Table 1 shows the values of false positive rates for the two algorithms on the four data sets. IDCAD experiences a significantly higher than expected false positive rate. This highlights the ineffectiveness of the parameter γ to correctly regulate anomaly capture. We tested IDCAD on S1 after removing all the noisy data and it achieved an 8% false positive rate with $\gamma = 0.99999$. This shows that the approximations result in the contraction of the ellipsoidal boundary and higher values for γ have to be chosen to compensate for this effect. We expect that higher frequency sampling, which results in greater similarity between the adjacent data samples in the time-series, would result in even higher contractions in the ellipsoidal boundary calculated using the IDCAD formulas with the forgetting factor. This example shows that IDCAD is unreliable, so we concentrate on ewIDCAD in the remainder of this article.

Table 1: False positive rate for IDCAD and ewIDCAD with parameter $\gamma = 0.99$.

	S1	S2	S3	S4
ewIDCAD	0.015	0.014	0.015	0.005
IDCAD	0.284	0.266	0.169	0.319

7.3 Results on synthetic data sets

The main parameter in the anomaly detection algorithms is the threshold used for anomaly detection, viz., H in MCUSUM and ewMCUSUM, R in ARCUSUM and t



(a) S1 (b) S2

(c) S3 (d) S4

Figure 4: ROC plots for the synthetic data sets (points on each curve correspond to the varying thresholds).

in ewIDCAD. The thresholds used by ARCUSUM and ewIDCAD provide statistical interpretations for the expected outcome in terms of false alarms if the initial assumption about the model is correct.

To compare the algorithms on the synthetic data sets, we plot the *Receiver Operating Curve* (ROC) curve of the algorithms for different values of their thresholds. We used thresholds $16, 14, 12, 10, 8, 6, 4, 2, 1$ for H; $5, 4, 3, 2.8, 2.6, 2.4, 2.2, 2, 1.8$ for R; and 0.999, 0.99, 0.98, 0.95, 0.93, 0.91, 0.89, 0.87, 0.85 for γ in $t^2 = (\chi^2)_p^{-1}(\gamma)$. The horizontal axis corresponds to the *False Positive* (FP) rate; the vertical axis to the anomaly detection rate. We did not try to lower the thresholds for ARCUSUM as suggested by Brown *et al.*,[16] because this would significantly increase the number of false alerts produced by the algorithm. In all four data sets, ewIDCAD and ewMCUSUM performed better than the other two algorithms. The ewIDCAD method seems best due to its high *detection rate* coupled with very low values for the FP rate. The data

set *S*4 significantly deviates from the initial assumption of following a Gaussian distribution; however ewIDCAD still performs well. The worst performing algorithm is ARCUSUM. In the first three data sets, its ROC curve consistently stays below the other three algorithms and in *S*4, where the data comes from a complex distribution, it completely loses its detection capabilities (notice there is no curve in Fig. 4(d) for this method).

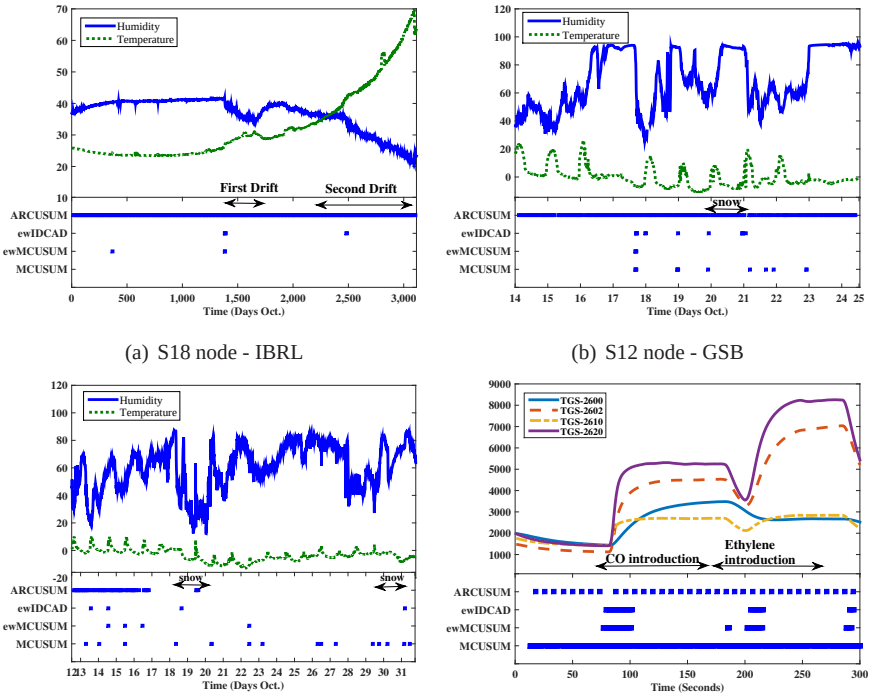
To perform a numerical comparison of the results, we calculate partial *Area Under Curve* (AUC) values. Normal AUC calculation is not possible as the curves cover different ranges and do not cover the entire range. The ROC curve for MCUSUM covers the smallest part of the range over the four data sets. Therefore, we consider only this part of the ROC curves to calculate the partial AUC for all the methods using trapezoidal numerical integration. If we don't have a detection rate value for the FP rate at the end of the range for any of the algorithms, we use the detection rate for the closest FP rate as an approximation for the end point. Table 2 shows the partial AUC values. The values show that ARCUSUM consistently has the lowest partial AUC values while ewIDCAD has the highest. The ewCUSUM algorithm trails ewIDCAD very closely in all the data sets.

Table 2: Partial AUC values for the common range of values for the false positive rate in the ROC curves of the four algorithms.

	S1	S2	S3	S4
ewIDCAD	0.033	0.043	0.027	0.018
ewMCUSUM	0.032	0.042	0.026	0.017
MCUSUM	0.024	0.030	0.011	0.013
ARCUSUM	0.002	0.003	0.000	0.000

Parameter selection in these algorithms is difficult because the input parameters do not have a clear interpretation. Additionally, in algorithms such as ARCUSUM, where the parameter reflects a statistical bound on the FP rate of the algorithm, the expected results are often hard to reach. This is due to the fact that these algorithms have multiple assumptions about the characteristics of the data set and often some of these assumptions are only weakly satisfied. However in these experiments, the false positive rate generated by the ewIDCAD algorithm stayed around the expected value,

i.e., $(1-\gamma)\%$. This value, $(1-\gamma)\%$, provides a good estimation for the FP rate of ewIDCAD.



(a) S18 node - IBRL

(b) S12 node - GSB

(c) S12 node - LG

(d) UCI - Gas Sensor

Figure 5: Time-series plot of the temperature and humidity readings in three real-life data sets. The locations of significant anomalies identified by each algorithm are marked by a dot.

7.4 Results on real data sets

Analysis of the results of the algorithms on real-data sets are difficult because there is no "ground truth" information that identifies anomalies. Since we have only information about significant events in these real-life data sets, we cannot evaluate the results in terms of the anomalies detected by different models. Therefore, we focus on the identification of significant events or change-points. In incremental online algorithms, significant collective anomalies are defined as a sequence of n_a consecutive point anomalies. For the four algorithms in this study, only ARCUSUM is unable to detect consecutive anomalies, so for this algorithm, we assume all anomalies are sig-

nificant. The other two CUSUM-based algorithms reset the cumulative sum after an anomaly is declared, thus only very significant anomalies can cause consequent anomalies. In contrast, ewIDCAD is based on a decision boundary and significant changes will result in multiple consequent anomalies. Therefore, a different value of n_a should be considered for each algorithm. In addition, the threshold for what is a significant anomaly is also an application dependent decision. For illustrative purposes in this section, we use $n_a = 1$ for ARCUSUM, $n_a = 3$ for MCUSUM and ewMCUSUM and $n_a = 5$ for ewIDCAD.

Based on the results obtained for the synthetic data sets, which have similar value ranges to the real-life data sets, we choose $H = 8$, $R = 2.5$ and $\gamma = 0.98$. The choice of these values influences the FP and detection rates. Increasing the values will increase the detection rate but also increase the FP rate. We chose H for ewMCUSUM and γ for ewIDCAD to insure that no more than 5% anomalies were detected by these methods in all the data sets. We selected the highest γ and lowest $H = 8$ from the range of values considered in the synthetic dataset analysis that satisfy the 5% anomaly rate constraint in GSB, LG and IBRL data sets. The Gas Sensor data has a high number of changes with respect to the size of the data so the 5% limit is not applied for the algorithms in this data set. This limit ensures that the algorithms do not generate excessive anomalies to reach higher detection rates.

We do not apply the 5% limit to the other two algorithms. In fact, with the selected thresholds, the anomaly rate of MCUSUM and ARCUSUM stays below 5% in IBRL but increases to around 20% for MCUSUM and 11% for ARCUSUM in the GSB and LG data sets. This excessive rate of anomalies suggests a higher threshold would be more appropriate for these algorithms. However, a higher threshold would lower their detection rate. So by not applying the 5% limit over these algorithms, they should potentially be able to generate a better detection rate than ewMCUSUM and ewIDCAD.

Fig. 5 shows the significant anomalies detected by the four algorithms marked by dots below the time-series plots of the features in each data set. The performance of ARCUSUM is hard to judge as it produces alarms every few samples and no particular connection between the alarms and significant events can be highlighted. For the LG

data set, this algorithm generates some alerts initially and during the first snow period and no alerts in the rest of the experiment. The alarms by ewMCUSUM do not pinpoint the significant events in any data set except in IBRL where this algorithm identifies the start of the first drift. The results of MCUSUM and ewIDCAD more closely match the physical information. In IBRL, ewIDCAD identifies the start of the first drift and also the second drift after a short delay. In this data set, MCUSUM does not identify any significant anomalies. In LG and GSB, we expect to see a significant anomaly around the time of the start of the snow and when the snow stops. In GSB, both MCUSUM and ewIDCAD identify the start and end of the snow period as significant anomalies. In GSB, ewIDCAD only identifies the start of the first snow period and the end of the second one, while MCUSUM identifies both start and end of the two snowy periods. But in general, MCUSUM indicates more unexplained significant anomalies during the experiments in LG and GSB and has an anomaly detection rate around 20%.

Fig. 5(d) shows the results of the four algorithms on the UCI Gas Sensor data set. The upper part of the figure shows the signal from the first four sensors in the data set. The times where CO and Ethylene are introduced to the sensors are marked on the figure. The sensors react to the introduction of gases (or removal of gases) within a few seconds of the change. As shown at the bottom of Fig. 5(d), ewIDCAD and ewMCUSUM accurately detect the reaction of the sensors and introduction of the gases as significant anomalies. The other two algorithms do not show the expected significant anomalies in the readings. A difference between ewIDCAD and ewMCUSUM is at the time of discontinuation of CO gas and introduction of Ethylene. At this time ewMCUSUM identifies two significant anomalies close to each other, which matches the sensors' reactions, first to the discontinuation of CO and then to the introduction of Ethylene. At this time, ewIDCAD identifies one set of significant anomalies. However, a choice of $n_a = 4$ for ewIDCAD would result in the algorithm detecting removal of CO, as well. So with $n_a = 4$ the event of removing CO from the gas chamber becomes significant according to our ewIDCAD algorithm.

8 Discussions and Conclusion

In this paper, we proposed an online model for anomaly detection that features exact updates for the inverse of the covariance matrix and the sample mean where an exponential weighting algorithm is applied to allow the model to gracefully forget the data in the distant past. The proposed model is suitable for non-stationary environments where the environment is continuously changing over time. We use these formulas to define two different methods - ewIDCAD and ewCUSUM - for anomaly detection. Our evaluation has shown that the proposed methods achieve better accuracy in non-stationary environments than two well-known techniques.

The complexity of algorithms for resource constrained networks is an important factor. Fully online algorithms, like the ones discussed in this paper, have constant memory requirements and linear complexity with respect to the number of data samples $O(N)$. Hence, they are very suitable for this style of network processing. Among the models discussed, ARCUSUM has the highest complexity as it has a two-stage calculation. First it requires updating an ARX model and then applies CUSUM to the residuals. The parameter-update in the other three algorithms has similar complexity but the ewIDCAD has slightly lower complexity when deciding whether an observation is an anomaly, because ewIDCAD is based on a decision boundary rather than the cumulative sum of the likelihood function.

Another advantage of online algorithms is their adaptability to changes in the underlying data. In fully online algorithms, adaptability is usually achieved through restarts or a mechanism that enables graceful degradation of inputs in the distant past (fading memory). Fading memory is usually implemented by exponentially weighting older observations, the approach taken in this paper for ewIDCAD and ewMCUSUM. The main advantage of algorithms with fading memory is that they are usually able to identify collective anomalies. However, these algorithms can be misled by *drift* anomalies. Slow changes in the data may be treated as normal changes in the environment, so these algorithms are not suitable for drift detection. However, if the speed of the drift is higher than the tracking capability of the algorithm, algorithms with fading memory

can identify drifts as shown, for example, in the performance of ewIDCAD in the IBRL data set.

Ease of parameter selection is an important aspect of any anomaly detection technique. Among the four algorithms discussed, only the threshold parameter in ewIDCAD has a clear interpretation, which greatly simplifies its selection. The bound defined in ewIDCAD is based on the assumption that the underlying data follows a Gaussian distribution. This bound maintains its statistical power for the majority of unimodal distributions. However, unimodality is often a strong assumption in time-series data as the data often change over time. With the help of fading memory, the boundary defined by ewIDCAD maintains its statistical properties in time-series with locally Gaussian (or unimodal) data distributions. According to the *principle of locality*, real-life time-series are expected to have locally consistent behaviour.

The introduction of exponentially weighted update formulas to the multivariate CUSUM shows some merits, but considering all the aspects of these four online algorithms for anomaly detection, ewIDCAD seems to be the best algorithm in terms of its overall performance, complexity and ease of parameter selection.

There are several aspects of the proposed algorithms that require further study. Currently in Algorithm 1, all data points are used for updating the model, even anomalies. With this approach, large-valued anomalies may affect the performance of the algorithm. Another worthwhile effort is to devise an update rule, which performs selective updates based on the distance of a data point to the current decision boundary. we will focus on devising methods for automatically detecting the difference between point anomalies and collective anomalies without the need for the unintuitive n_a parameter. This might be done by assigning degrees of significance to anomalies or by exploiting the sequence of changes in ellipsoidal summaries obtained by ewIDCAD. We expect the performance of the proposed algorithms to deteriorate for high dimensional input data, so we want to investigate methods that are useful in this case.

9 Acknowledgment

Moshtaghi was supported under Australian Research Council's *Discovery Projects* funding scheme (project number DE150100104) and Bezdek was supported by Data61.

References

[1] M. Xie, S. Han, B. Tian and S. Parvin, *Journal of Network and Computer Applications*, 2011, **34**, 1302 – 1325.

[2] V. Chandola, A. Banerjee and V. Kumar, *ACM Comput. Surv.*, 2009, **41**, 1–15.

[3] M. Gupta, J. Gao, C. C. Aggarwal and J. Han, *IEEE Transactions on Knowledge and Data Engineering*, 2014, **26**, 2250–2267.

[4] R. Turner, Z. Ghahramani and S. Bottone, 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), 2010, pp. 385–390.

[5] D. Pokrajac, A. Lazarevic and L. Latecki, IEEE Symposium on Computational Intelligence and Data Mining, 2007, pp. 504–515.

[6] M. A. Mahmoud and P. E. Maravelakis, *Journal of Statistical Computation and Simulation*, 2013, **83**, 721–738.

[7] B. V. Jancee and S. Radha, *Modelling and Simulation in Engineering*, 2014, **2014**, 31–39.

[8] R. B. Crosier, *Technometrics*, 1988, **30**, 291–303.

[9] G. J. Ross, D. K. Tasoulis and N. M. Adams, Proceedings of the 2009 ACM Symposium on Applied Computing, (SAC '09), New York, NY, USA, 2009, pp. 1501–1505.

[10] M. A.Rassam, A. Zainal and M. A. Maarof, *Sensors*, 2013, **13**, 10087–10122.

[11] M. Moshtaghi, J. C. Bezdek, T. C. Havens, C. Leckie, S. Karunasekera, S. Rajasegarar and M. Palaniswami, *Wireless Communications and Mobile Computing*, 2014, **14**, 905–921.

[12] S. Rajasegarar, J. C. Bezdek, C. Leckie and M. Palaniswami, *ACM Transactions on Sensor Networks (ACM TOSN)*, 2009, **6**, 1–28.

[13] M. Moshtaghi, C. Leckie, S. Karunasekera, J. C. Bezdek, S. Rajasegarar and M. Palaniswami, IEEE Int. Conf. on Data Mining, 2011, pp. 467–476.

[14] D. M. Tax and R. P. Duin, *International Conference on Pattern Recognition*, 2000, **2**, 2672.

[15] L. Ljung, *System Identification: Theory for the User*, Prentice Hall PTR, 1999.

[16] R. L. Brown, J. Durbin and J. M. Evans, *Journal of the Royal Statistical Society. Series B (Methodological)*, 1975, **37**, 149–192.

[17] D. Tax, *DDtools, the Data Description Toolbox for Matlab*, 2015, version 2.1.2.

[18] *IBRL-Web 2006. http://db.lcs.mit.edu/labdata/labdata.html*, 2006, `http://db.lcs.mit.edu/labdata/labdata.html`.

[19] *SensorScope. http://lcav.epfl.ch/page-86035-en.html*, 2007, `http://lcav.epfl.ch/page-86035-en.html`.

[20] M. Moshtaghi, J. C. Bezdek, C. Leckie, S. Karunasekera and M. Palaniswami, *IEEE Transactions on Fuzzy Systems*, 2015, **23**, 688–700.