A Relevance Weighted Ensemble Model for Anomaly Detection in Switching Data Streams

Mahsa Salehi[1], Christopher A. Leckie[1],
Masud Moshtaghi[2], and Tharshan Vaithianathan[3]

[1] National ICT Australia, Department of Computing and Information Systems,
The University of Melbourne, Victoria 3010, Australia
msalehi@student.unimelb.edu.au, caleckie@unimelb.edu.au
[2] Faculty of Information Technology,
Monash University, Victoria 3145, Australia
masud.moshtaghi@monash.edu
[3] National ICT Australia, Department of Electrical and Electronic Engineering,
The University of Melbourne, Victoria 3010, Australia
tharshan@nicta.com.au

Abstract. Anomaly detection in data streams plays a vital role in online data mining applications. A major challenge for anomaly detection is the dynamically changing nature of many monitoring environments. This causes a problem for traditional anomaly detection techniques in data streams, which assume a relatively static monitoring environment. In an environment that is intermittently changing (known as switching data streams), static approaches can yield a high error rate in terms of false positives. To cope with dynamic environments, we require an approach that can learn from the history of normal behaviour in data streams, while accounting for the fact that not all time periods in the past are equally relevant. Consequently, we have proposed a relevance-weighted ensemble model for learning normal behaviour, which forms the basis of our anomaly detection scheme. The advantage of this approach is that it can improve the accuracy of detection by using relevant history, while remaining computationally efficient. Our solution provides a novel contribution through the use of ensemble techniques for anomaly detection in switching data streams. Our empirical results on real and synthetic data streams show that we can achieve substantial improvements compared to a recent anomaly detection algorithm for data streams.

Keywords: Anomaly detection, Ensemble models, Data streams.

1 Introduction

Anomaly detection (also known as novelty or outlier detection) is an important component of many data stream mining applications. For example, in network intrusion detection, anomaly detection is used to detect suspicious behaviour that deviates from normal network usage. Similarly, in environmental monitoring, anomaly detection is used to detect interesting events in the monitored

V.S. Tseng et al. (Eds.): PAKDD 2014, Part II, LNAI 8444, pp. 461–473, 2014.

environment, as well as to detect faulty sensors that can cause contamination of the collected data. A major challenge for anomaly detection in applications such as these is the presence of nonstationary behaviour in the underlying distribution of normal observations. In addition, the high rate of incoming data in such applications adds another challenge to anomaly detection, as the anomalous data points should be detected in a computationally efficient way with high accuracy. Many anomaly detection techniques are based on an assumption of stationarity. However, such an assumption can result in low detection accuracy when the underlying data stream exhibits nonstationarity. In this paper, we present a novel method for robust anomaly detection for a special class of nonstationarity, known as switching data streams.

Many environments are intermittently changing, and thus exhibit piecewise stationarity. For example, sensors in an office environment may collect observations with different underlying statistical distributions between day-time and night-time. We refer to the data from such environments as switching data streams. In addition to large-scale switching behaviour, there can also be finer-scale variation in the distribution of observations, e.g., variation during day-time observations. To address these challenges, we propose a novel approach to anomaly detection that can selectively learn from previous time periods in order to construct a model of normal behaviour that is relevant to the current time window. This is achieved by maintaining a cluster model of normal behaviour in each previous time period, and then constructing an ensemble model of normal behaviour for the current period, based on the relevance of the cluster models from previous time periods to the current time period. This relevance-weighted ensemble model of normal behaviour can then provide a more robust model of normality in switching data streams.

While ensemble methods have been widely used in supervised learning [1,2], their use for unsupervised anomaly detection in data streams is still an open research challenge. A key advantage of our approach is that it can improve the accuracy of anomaly detection in switching data streams. We have evaluated the effectiveness of our approach on large-scale synthetic and real sensor network data sets, and demonstrated its ability to adapt to the intermittent changes in switching data streams. We show that our approach achieves greater accuracy compared to a state-of-the-art anomaly detection approach for data streams.

2 Related Work

A number of surveys [3,4] have categorized various techniques for anomaly detection. Most focus only on static environments and do not consider dynamic behaviour. In dynamic environments an unknown volume of data (data streams) are produced. Hence the major challenge is how to detect anomalies in such environments in the presence of dynamics in the distribution of the data stream. Based on existing surveys of anomaly detection in data streams [5,6], we categorize these methods into *Model based* and *Distance based* approaches.

Model based approaches build a model over the data set and update it as data evolves with each incoming data point. In this category, some authors learn a probabilistic model. This requires a priori knowledge about the underlying distribution of data, which is not appropriate if the distribution is not known [7,8]. Other approaches use clustering algorithms to build a model for normal behaviour in data streams. However, they are not really designed for anomaly detection [9,10]. An exception is [11], in which a segment based approach is proposed. It uses k-means as a base model and provides guidelines regarding how to use the proposed approach for anomaly detection. As a result, it assumes the number of clusters is known, which may not be the case in changing environments (since the number of clusters may change over time). In addition, while these clustering techniques work for gradually evolving data streams, they are not applicable in switching environments, which is our focus in this paper.

Distance based approaches are the second category of anomaly detection techniques in data streams. We have two types of distance-based outliers: 'global' and 'local'. Distance-based 'global' outliers are first introduced by [12], where a data point x is a distance-based outlier if less than k data points are within a distance R from it. In [13] and [14], a sliding window is used to detect such global outliers. Since parameter R is fixed for all portions of the data, these approaches fail to detect anomalies in non-homogenous densities. In contrast, distance-based 'local' outliers are data points that are outliers with respect to their k nearest neighbours without considering any distance R. Local outliers are first introduced in [15] and a measure of being an outlier - the Local Outlier Factor(LOF) - is assigned to each data point in a static environment. Later, in [16] and [6] two similar approaches are proposed to find local outliers in data streams. While the former made an assumption about the underlying distribution of data, the latter (Incremental LOF) extended [15], and could detect outliers in data streams by assigning the LOF to incoming data points and updating the k nearest neighbors. However, there are still some limitations with this approach in the presence of switching data streams: 1) It has problems with detecting dense drift regions in data streams, which results in false negatives. 2) In switching data streams, the distribution of normal data can change suddenly. Since incremental LOF keeps all of the history of the data points, it could not differentiate between different states, which again results in false negatives (e.g., a data point is an anomaly in one state while it is not an anomaly in another). 3) It is hard to choose the parameter value k in the presence of changing distribution environments. We propose to address these open problems for anomaly detection in switching data streams by using an ensemble approach.

Ensemble approaches have been shown to have benefits over using a single classifier, particularly in dynamic environments. So far, several ensemble learning methods have been proposed for data stream classification [1,2]. In contrast, there is little work done on anomaly detection using ensemble techniques [17]. In a recent comprehensive survey paper [18], a categorization of outlier detection ensemble techniques is based on the constituent components (data/model) in these

techniques. Nevertheless, none of these approaches are aimed at handling switching data streams or even streaming data.

Since incremental LOF (iLOF) is the only anomaly detection technique in streaming data which handles different densities and is reported to detect changes in data distributions, we use it as a baseline to evaluate our proposed approach.

3 Our Methodology

In this section, we formally define our problem and proposed algorithm.

3.1 Problem Definition

We begin by describing our notation for switching data streams. We consider the problem of anomaly detection in a switching data stream, where the underlying distribution of observations is only piecewise stationary. That is, the monitored environment switches between a number of "normal" states, such as day vs night in an episodic manner. Let the state of the system be a random variable over the domain of possible states $S = \{s_1, ..., s_D\}$, where the system has D normal states. The distribution of an observation X in state s_d is drawn from a mixture of K_d components. For example, each component of the mixture distribution of state s_d could be a multivariate Gaussian distribution with different means and covariance. Let $\Theta = \{\theta_{d,i}, i = 1, .., K_d\}$ denote the parameters corresponding to the K_d components of state s_d, e.g., $\theta_{d,1} = \{\mu_{d,1}, \Sigma_{d,1}\}$ for a multivariate Gaussian distribution. Further, let $\Phi = \{\phi_{d,i}, i = 1, .., K_d\}$ denote the mixture weights of the K_d components of state s_d, i.e., the prior probability that a random observation in state s_d comes from component i is $\phi_{d,i}$.

We observe a stream of observations, where $X(1:T)$ denotes the sequence of observation vectors collected over the time period $[t_1, ..., t_T]$. At time t, we receive a vector of observations $X(t) = [x_1(t), .., x_P(t)]^T$ corresponding to P different types of observation variables, e.g., temperature, pressure, humidity, etc, where $X(t) \in \Re^P$. If the environment is in state s_d at time t, then $X(t)$ is sampled from one of the K_d component distributions in Θ_d of state s_d.

The monitored environment is initially in some random state $s_{(1)} \in S$ at time t_1, and remains in that state until a later time t_{c1} when it switches to a different state $s_{(2)} \in S \backslash s_{(1)}$. It then remains in state $s_{(2)}$ until a later time t_{c2} when it switches again to $s_{(3)} \in S \backslash s_{(2)}$, and so on. Our aim is to detect anomalies in this data stream X. In order to detect anomalies, we require a model of "normal" behaviour for the environment, given that the number of possible states, the number and mixture of components in each state, and the parameters of each component in each state are unknown a priori.

In order to learn our model of normal behaviour, we could estimate all of the different component parameters for all states by keeping all data $X(1:t)$. However, this is impractical due to the memory requirements and it would mix all the states together when building the normal model. Alternatively we can learn only from the window of the w most recent observations $X(t-w+1:t)$.

While this is more computationally efficient, it only provides a limited sample of measurements from the current state. If the window size w is small, then this might yield a noisy estimate of the component distribution parameters.

The problem we address is how to find a balance between these two extremes, by maintaining multiple models of normal behaviour based on previous data windows. Our expectation is that this will provide us with a balance between minimising the memory requirements for our anomaly detection scheme while maximising accuracy. However, the key challenge in achieving this balance is that not all previous windows will be relevant to the current window of observations, due to the switching nature of the data stream. Consequently, we require a method to take into consideration the relevance of previous windows, so that we can construct a model of normal behaviour based on the current and previous windows. In particular, we need a way to weight the influence of previous windows on data in the current window, based on the degree of relevance of those previous windows. We consider different kinds of anomalies in the system: either localized in time, i.e., a burst of noise or a drift in the data stream, or uniformly distributed over the data stream.

3.2 Our Ensemble-Based Algorithm

We now describe the main steps of our algorithm for ensemble anomaly detection in switching data streams. Algorithm 1 shows the pseudo code of the ensemble algorithm, comprising three main steps: *Windowing*, *Weighting* and *Ensemble* formation. We assume the previous data streams have already been clustered based on windows of w observations and the problem is how to detect anomalies in the most recent (current) window of w observations.

1. *Windowing Step*: In general, in applications that generate data steams, the observations arrives sequentially. We consider a data stream that switches between different states, where each state comprises different component distributions. By breaking the whole data stream of observations into windows of size w, we can extract the different underlying distributions of the data streams.

In this step, we cluster the current w observations by using an appropriate clustering algorithm, in order to find out the current underlying component distributions. We chose the HyCARCE clustering algorithm [19], which is a computationally efficient density-based hyperellipsoidal clustering algorithm that automatically detects the number of clusters, and only requires an initial setting for one input parameter, i.e., the grid-cell size. In addition, it has a lower computational complexity in comparison to existing methods. Hence, it is an appropriate clustering algorithm for our data stream analysis problem in which time is a vital issue and the number of clusters is unknown. Interested readers can find a detailed description and pseudo code for the HyCARCE algorithm in [19]. At the end of this step a clustering model is built that comprises a set of cluster boundaries, $C_\kappa = \{c_{\kappa,1}, ..., c_{\kappa,|C_\kappa|}\}$. In this paper we use the boundaries of the ellipsoidal clusters as our decision boundaries.

Input : $W_\kappa = \{X(t-w+1), ..., X(t)\}$: most recent window of w observations from the data stream
$\mathcal{E} = \{C_1, C_2, ..., C_{\kappa-1}\}$: set of clusterings corresponding to previous windows $\{W_1, ..., W_{k-1}\}$, where a clustering $C_j = \{c_{j,1}, ..., c_{j,|C_j|}\}$
Output: $A_\kappa = \{a_1, ..., a_w\}$: set of anomaly scores in most recent window W_κ

$C_\kappa \longleftarrow Cluster(W_\kappa)$; `// cluster` W_κ `using the HyCARCE clustering [19]`
$R \longleftarrow \{\}$; `// compute relevance of previous clustering to current one`
foreach *clustering* $C_j \in \mathcal{E}$ **do**
  $R \longleftarrow R \cup \{Relevance(C_\kappa, C_j)\}$; `// Algorithm 2`
end
$R \longleftarrow R \cup \{max(R)\}$; `//` $R = \{r_1, ..., r_\kappa\}$
$\mathcal{E} \longleftarrow \mathcal{E} \cup \{C_\kappa\}$;
`// test each observation in` W_κ `to check if it is an anomaly`
foreach $X_n \in W_\kappa$ **do**
  `// test if X belongs to any cluster in each clustering` C_j
  `// belongs is defined in Definition 1 using Equation 1 and 2`
  foreach $C_j \in \mathcal{E}$ **do**
    if $\exists c_{j,i} \in C_j$ — *belongs* $(X, c_{j,i})$ **then**
      $b_j = 0$; `//` X `is normal in clustering` C_j
    else
      $b_j = 1$; `//` X `is anomalous in clustering` C_j
    end
  end
  $a_n = \frac{\sum_{j=1}^{\kappa} b_j r_j}{\sum_{j=1}^{\kappa} r_j}$; [1]
  $A \longleftarrow A \cup \{a_n\}$;
end
return A;

Algorithm 1. Ensemble Anomaly Detection

2. *Weighting Step*: According to our windowing step, all of the previous w sized observation windows have already been clustered. The history is used to better estimate the underlying distribution of the current window. However, not all of the previous clusterings have the same level of importance, and some of them might be more relevant to the current window, as the data stream switches between different states. Our solution, which is called the *Weighting step*, is to assign weights to previous clustering models based on the similarity between the current and previous clustering models.

In this step, the distance between each previous clustering model (C_j) and the current clustering model (C_κ) is computed. As time complexity is a major issue for anomaly detection on data streams, a *greedy* approach is proposed to compute this distance. In this approach, the distance between each pair of cluster boundaries in C_κ and C_j is computed based on their *focal distance* [20]. Focal distance is a measure of the distance between two hyperellipsoids considering their shapes, orientations and locations. According to a recent work [20], this measure works well for computing the similarity between hyperellipsoids.

Input : $C_j = \{c_{j,1}, ..., c_{j,|C_j|}\}$: previous clustering
$C_\kappa = \{c_{\kappa,1}, ..., c_{\kappa,|C_\kappa|}\}$: current clustering
Output: r_j: the similarity between C_j and C_κ
// λ and γ are smallest/equal and largest clustering respectively
if $|C_j| \leq |C_\kappa|$ **then**
  $\lambda \longleftarrow j$; $\gamma \longleftarrow \kappa$;
else
  $\lambda \longleftarrow \kappa$; $\gamma \longleftarrow j$;
end
$\mathcal{D} \longleftarrow \{\}$; // compute distance between every pair of hyperellipsoids
foreach $c_{\lambda,r} \in C_\lambda$ **do**
  foreach $c_{\gamma,c} \in C_\gamma$ **do**
    $d_{r,c} \longleftarrow FocalDistance(c_{\lambda,r}, c_{\gamma,c})$; // Focal Distance[20]
    // $\mathcal{D} = \{D_1, ..., D_{|C_\lambda|}\}$ and $D_i = \{d_{i,1}, ..., d_{i,|C_\gamma|}\}$
  end
end
$dist = 0$; // compute minimum distance based on a Greedy approach
while $\mathcal{D} \neq \varnothing$ **do**
  $d_{r,c}^{min} \longleftarrow$ find minimum element in $\mathcal{D}$;
  $dist \longleftarrow dist + d_{r,c}^{min}$
  foreach $D_i \in \mathcal{D}$ **do**
    $D_i \longleftarrow D_i - \{d_{i,c}\}$; // removing relevant assigned clusters
  end
  $\mathcal{D} \longleftarrow \mathcal{D} - \{D_r\}$;
end
$r_j = \frac{1}{dist}$; return r_j;

Algorithm 2. Relevance Function

After determining the focal distance, the algorithm finds the minimum distance among all computed distances, and assigns the relevant cluster boundaries as a matching pair. It continues this process, until all of the clusters in at least one of the two clustering models are each matched to a corresponding cluster in the other model. The sum of the found minimum distances is used as the distance between two clustering models. Obviously, if the models are less distant, they would be more similar. Therefore, the distances are reversed at the end of the algorithm to show the similarity. The relevant pseudo code is described in Algorithm 2. Finally, in order to use the current clustering model in the ensemble step, the maximum assigned similarity among all previous clusterings is assigned to the current clustering. In this way, we can find a balance between the current window and previous windows for anomaly detection.

3. *Ensemble Step*: In this step, anomaly detection is performed based on the current and previous clusterings. As discussed in the previous subsection, not all historical models are useful, due to the changing environment. Hence, for each current observation, we check if it belongs to a hyperellipsoid in each of the clustering models according to the following definition:

Definition 1. *The observation X* ***belongs*** *to hyperellipsoid $c_{j,i}$, if the Mahalanobis distance between them is less than a threshold t^2, where the Mahalanobis distance is:*

$$\mathcal{M}(X, c_{j,i}) = (X - \hat{\mu}_{j,i})^T \hat{\Sigma}_{j,i}^{-1} (X - \hat{\mu}_{j,i}) \tag{1}$$

where $\hat{\mu}_{j,i}$ and $\hat{\Sigma}_{j,i}$ are the mean and covariance of hyperellipsoid $c_{j,i}$ respectively, and the threshold is:

$$t^2 = (\chi_P^2)_\rho^{-1} \tag{2}$$

where t^2 is the inverse of chi-square statistic, ρ is the percentage of data covered by the ellipsoid, and P is the dimensionality of the observations.

Thereafter, the probability of being outlier is computed for the current observation according to the formula in Algorithm 1 (line 16). The formula is based on a relevance voting. The algorithm computes this probability for all data points X in the current window.

3.3 Time Complexity

The total number of observations (data points) N is divided into κ windows of size w. Let l be the average of number of ellipsoids in each C_j. Therefore l is a function of $K_d, d = \{1, .., D\}$ and $K_d \ll N$. The time complexity for the *Windowing Step* is $O(N)$ as the complexity of HyCARCE is near linear with respect to the number of data points. The *Weighting Step* can be computed in $O(\frac{N}{w} l^3)$, considering the 'Relevance' can be computed in $O(l^3)$ in the worst case. In the last step of the algorithm, 'belongs' can be computed in $O(l)$ so the time complexity of the *Ensemble Step* is $O(w\kappa l)$. As a result, the time complexity of Algorithm 1 is $O(N(\frac{l^3}{w} + l))$, which is near linear with respect to the number of data points. Hence, our ensemble approach is computationally efficient. Time complexity of iLOF is also near to linear depending on the parameter k.

4 Evaluation

In this section we aim to compare the accuracy, sensitivity and specificity of our ensemble model with the iLOF algorithm on several dynamic environments.

4.1 Data Sets

We use three different data sets to evaluate our approach.

Synthetic Data Set: In order to generate synthetic data sets, we consider a state machine that can simulate a switching data stream from a changing environment. We assume there are only two different states (S_1, S_2) and changing from one state to the other occurs periodically. The first state (s_1) has three underlying component distributions $\theta_{1,1}$, $\theta_{1,2}$ and $\theta_{1,3}$ with corresponding mixture

weights $\phi_{1,1}$, $\phi_{1,2}$ and $\phi_{1,3}$. The second state (s_2) has two underlying component distributions $\theta_{2,1}$ and $\theta_{2,2}$ with mixture weights $\phi_{2,1}$ and $\phi_{2,2}$. Altogether the environment consists of five component distributions. The states are changing periodically based on a constant rate (every $m = 100$ samples). In addition, the distributions are hyperellipsoids and the initial mean and covariance is chosen randomly. As discussed earlier, from one instance of a state to the other, for the same state, we have some perturbations in the mean and covariance of the underlying distribution, i.e.,the parameters of $\theta_{d,i}$ can be slightly perturbed between different occurrences of state s_d. By using this state machine, we have generated 50,000 2-dimensional records. We generated 10 different data sets and averaged the final results. The inserted noise was 2%.

Real Data Sets: In order to evaluate our algorithm over a real data set, we used two public available data sets. These data sets contain periodic measurements over day (state s_1) and night (state s_1). The first is the IBRL (Intel Berkeley Research Lab) data set[1]. A group of 54 sensors were deployed to monitor an office environment, from Feb. 28th until Apr. 5th, 2004. Moreover, by visualising the data collected by all the sensors, we observed that sensor number 45 stopped working in the last two days causing a drift which is labeled as anomalies (4%). The sensors were collecting weather information. We chose two features, humidity and temperature. The measurements were taken every 31 seconds and there are about 50,000 records.

The second real data set is from the TAO (Tropical Atmosphere Ocean) project[2], by the Pacific Marine Environmental Lab of the U.S. National Oceanic and Atmospheric Administration. This monitors the atmosphere in the tropical Pacific ocean. We have used the period of time from Jan. 1st until Sep. 1st, 2006 which is used in [13]. We chose three features, precipitation, relative humidity and sea surface temperature. Among the different monitoring sites, we chose site:(2 °N,165 °E), since this site has all three features available for the mentioned period of time. The measurements were taken every 10 minutes and there are about 37,000 records. This data set has some labels on the quality of measurements and we have used them for our evaluation. After visualization of the low quality data points (2%) and good quality ones, we find that in this data set the noise is a dense separate region from the normal observations.

4.2 Performance Measures

In order to evaluate our algorithm in comparison to iLOF, we have used three performance measures: (1) Area under the ROC curve (AUC). (2) The *accuracy* ($\frac{TP+TN}{P+N}$) of the ROC curve's optimal point, where P is the number of positives (anomalies) in the data set, N is the number of negatives (normal observations), TP is the number of true positives (correctly reported anomalies) and TN is the number of true negatives (correctly reported normal observations).

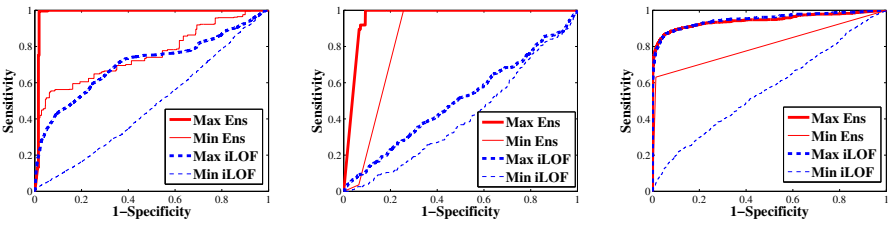
[1] `http://db.lcs.mit.edu/labdata/labdata.html`
[2] `www.pmel.noaa.gov/tao`

The optimal point on a ROC curve is the point with the maximum distance from diagonal line using the Youden Index ($max_i \frac{SE_i + SP_i - 1}{\sqrt{2}}$), where SE_i is the sensitivity for the ith threshold and SP_i is the specificity for the ith threshold. (3) The ratio of correctively detected anomalies *sensitivity* ($\frac{TP}{P}$) and correctly detected normal observations *specificity* ($\frac{TN}{N}$) for the ROC curve's optimal points.

4.3 Results and Discussion

We have compared our ensemble approach with iLOF on the three test data sets. For iLOF we have used the implementation provided in ELKI[21]. All measurements were normalized based on min-max normalization and we set $\rho = 0.99$ in Equation 2 [19]. Moreover, we have studied how the performance of our approach varies over different window sizes. The window size w is initially set to 100 observations, and then it is increased by increments of 500 observations until w is $\frac{1}{3}$ of the whole data set length (because we need at least two window sizes for voting). We also studied the effect of changing the number of nearest neighbours k in the iLOF algorithm, where $k \in \{3, 5, 10, .., 200\}$. Since the computational time of the iLOF algorithm increases with k, we set the upper bound to 200 to obtain a reasonable runtime.

We have computed the ROC curves for different window sizes w and number of neighbours k. Figure 1 depicts only the best ROC curves (thicker graph) and worst ones (thinner graphs) for simplicity for both approaches over three different data sets. The results show that in the two real data sets with dense outliers, our approach is better than iLOF by a large margin for both the best and worst curves (Figure 1a and 1b). In addition in Figure 1b our worst curve is even better than iLOF's best curve. However, in the synthetic data set in which the outliers are uniformly distributed over the data stream, both approaches have approximately the same best curves, and our approach has better AUC in comparison to iLOF in the worst case (Figure 1c). In order to make a more detailed comparison, we computed the optimal points of different ROC curves



(a)IBRL (b)TAO (c)Synthetic

Fig. 1. ROC curves for three data sets: Ensemble (Ens) in red, iLOF in blue

Table 1. Optimal Point Comparison: Our Ensemble Method vs iLOF

Data Set	Algorithm	Accuracy			Specificity			Sensitivity		
		Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
IBRL	Ensemble	98.23	90.62	74.90	98.18	90.22	73.60	100	98.39	55.45
	iLOF	93.68	76.46	07.73	98.06	78.36	03.30	95.81	39.74	03.28
TAO	Ensemble	90.81	86.57	74.62	90.73	86.45	74.40	100	100	100
	iLOF	88.42	84.98	53.38	88.92	76.09	53.37	54.72	39.89	26.06
Synthetic	Ensemble	98.17	97.80	94.42	98.87	98.27	94.56	87.06	72.61	48.66
	iLOF	94.49	90.82	84.74	94.64	91.04	85.51	88.98	79.09	21.82

as we described in Section 4.2. The results of the optimal point's accuracy, sensitivity and specificity are shown in Table 1.

The accuracy and specificity of the optimal points in our approach are higher than iLOF in all data sets. This means that iLOF produces more false negatives. In the IBRL and TAO data sets which are the real data sets, we have dense outliers (either drift or a separate distribution) which yields the false negatives. As can be seen the minimum specificity of iLOF in both real data sets is extremely low. Moreover, since in all three data sets we have switching states our ensemble algorithm can perform better in terms of specificity, whereas iLOF keeps all the history of the data points, and it could not differentiate between different states, which again results in false negatives. The last three columns of the table show the optimal point's sensitivity. The results show that we have much higher sensitivity in comparison to iLOF in the IBRL and TAO data sets and iLOF fails to detect anomalies with a considerably lower rate. However, in the synthetic data set with uniformly distributed anomalies, iLOF performs better in terms of finding anomalies in the average and maximum cases. However, the difference is small, and our ensemble method performs better in terms of overall accuracy. Finally, considering the max, average and min in all measures in Table 1, our method performs more consistently, which shows the results are less dependent on choosing the window size w. Also, for iLOF the range between the max and min cases is larger, indicating that it is sensitive to the choice of the parameter k.

In summary, our approach outperforms iLOF on the two real data sets with dense anomalies. Moreover, it is better than iLOF in accuracy and specificity on the synthetic data set, while iLOF only performs better in terms of sensitivity in the synthetic data set with uniformly distributed anomalies.

5 Conclusion

In this paper we proposed a novel approach to the problem of anomaly detection in data streams where the environment changes intermittently. We introduce an ensemble based approach to construct a robust model of normal behaviour in switching data streams. Although there have been many supervised techniques based on ensemble models for outlier detection in data streams, to the best of our knowledge there is no similar work that tackles the problem of using ensemble

techniques for anomaly detection in data streams. The empirical results show the strength of our approach in terms of accuracy. This highlights several interesting directions for future research. First, we can explore alternatives to our greedy approach for comparing clusterings. Second, we will investigate approaches to minimize memory consumption. Third, we will investigate different methods for selecting appropriate window boundaries.

Acknowledgments. The authors would like to thank National ICT Australia (NICTA) for providing funds and support.

References

1. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: SIGKDD, pp. 226–235. ACM (2003)
2. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New ensemble methods for evolving data streams. In: SIGKDD, pp. 139–148. ACM (2009)
3. Rajasegarar, S., Leckie, C., Palaniswami, M.: Anomaly detection in wireless sensor networks. IEEE Wireless Communications 15(4), 34–40 (2008)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Computing Surveys 41(3), 1–58 (2009)
5. Gupta, M., Gao, J., Aggarwal, C.C., Han, J.: Outlier detection for temporal data: A survey. Knowledge and Data Eng. 25(1), 1–20 (2013)
6. Pokrajac, D., Lazarevic, A., Latecki, L.J.: Incremental local outlier detection for data streams. In: CIDM, pp. 504–515. IEEE (2007)
7. Yamanishi, K., Takeuchi, J.I., Williams, G., Milne, P.: On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In: SIGKDD, pp. 320–324. ACM (2000)
8. Yamanishi, K., Takeuchi, J.I.: A unifying framework for detecting outliers and change points from non-stationary time series data. In: SIGKDD, pp. 676–681. ACM (2002)
9. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92. VLDB Endowment (2003)
10. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SIAM Conf. on Data Mining, pp. 328–339 (2006)
11. Aggarwal, C.C.: A segment-based framework for modeling and mining data streams. Knowledge and Inf. Sys. 30(1), 1–29 (2012)
12. Knox, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: VLDB, pp. 392–403. Citeseer (1998)
13. Angiulli, F., Fassetti, F.: Detecting distance-based outliers in streams of data. In: CIKM, pp. 811–820. ACM (2007)
14. Yang, D., Rundensteiner, E.A., Ward, M.O.: Neighbor-based pattern detection for windows over streaming data. In: Advances in DB Tech., pp. 529–540. ACM (2009)
15. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: ACM SIGMOD, vol. 29, pp. 93–104. ACM (2000)
16. Vu, N.H., Gopalkrishnan, V., Namburi, P.: Online outlier detection based on relative neighbourhood dissimilarity. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 50–61. Springer, Heidelberg (2008)

17. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: SIGKDD, pp. 157–166. ACM (2005)
18. Aggarwal, C.C.: Outlier ensembles: Position paper. SIGKDD Explorations Newsletter 14(2), 49–58 (2013)
19. Moshtaghi, M., Rajasegarar, S., Leckie, C., Karunasekera, S.: An efficient hyperellipsoidal clustering algorithm for resource-constrained environments. Pattern Recognition 44(9), 2197–2209 (2011)
20. Moshtaghi, M., Havens, T.C., Bezdek, J.C., Park, L., Leckie, C., Rajasegarar, S., Keller, J.M., Palaniswami, M.: Clustering ellipses for anomaly detection. Pattern Recognition 44(1), 55–69 (2011)
21. Achtert, E., Goldhofer, S., Kriegel, H.P., Schubert, E., Zimek, A.: Evaluation of clusterings–metrics and visual support. In: ICDE, pp. 1285–1288. IEEE (2012)