Detecting Outliers on Arbitrary Data Streams using Anytime Approaches

Ira Assent
Department of Computer Science
Aalborg University, Denmark
ira@cs.aau.dk

Philipp Kranen, Corinna Baldauf, Thomas Seidl
Data Management and Exploration Group
RWTH Aachen University, Germany
{kranen,baldauf,seidl}@cs.rwth-aachen.de

ABSTRACT

Data streams are gaining importance in many sensoring and monitoring environments. Frequent mining tasks on data streams include classification, modeling and outlier detection. Since often the data arrival rates vary, anytime algorithms have been proposed for stream clustering and classification, which can deliver a fast first result and improve their result if more time is available.

In this work, we propose the novel concept of anytime outlier detection and introduce an algorithm for anytime outlier detection based on a hierarchical cluster representation. We show promising results in preliminary experiments and discuss future research for anytime outlier detection.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

1. INTRODUCTION

Data streams are prevalent in many application domains, ranging from web traffic over surveillance data to sensor networks and monitoring tasks. Analysis of such streaming data requires special focus on the properties of streaming contexts, i.e. infinite data, limited memory, limited time and often varying time allowances. Especially for monitoring applications, data mining needs to produce results in near real-time in order to be meaningful. For example, in medical applications, monitoring a patient's heart rate is only helpful if any relevant patterns can be communicated to the doctors or nurses immediately such that they can take appropriate counter-measures on time. Similarly, assisted living that relies on supervision of elderly or chronically ill people requires that people receive assistance if anything suspicious occurs.

These examples are situations where we are interested in determining deviating signals from the continuous data stream. Outlier detection is the data mining task that aims at discovering such deviating data. In the literature, different notions of outliers exist. A popular definition is that of Hawkins, who states that an outlier is "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism" [10]. Existing work on outlier detection has approached this task from different perspectives, e.g. statistically by analyzing the data distribution [4], or based on distances by defining outliers as objects who do not have a minimum number of objects within a certain distance [15]. Cluster-based techniques use clustering to describe the underlying structure of the data, i.e. the prevailing patterns, and define outliers as those objects that do not belong to any cluster or do not fit their cluster well [7, 11].

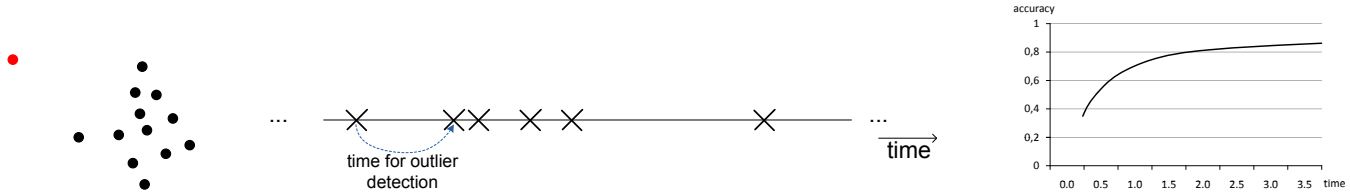
Recently, some approaches for outlier detection on data streams have been proposed [1, 3, 24, 26]. However, all of these approaches assume streams of fixed arrival rates, while in many data streams the assumption of a stream with a fixed data arrival rate does not hold. Consider for example sensor networks. In order to reduce the communication overhead and to save battery power, modern sensors send data only when necessary, e.g. when measurements differ significantly. Likewise, when monitoring a patient's health status, little data needs to be sent to a central analysis system when the measured values are within normal ranges, but additional measures might be taken and sent otherwise.

Generally, stream mining tasks cannot be performed offline. This is inherent to the scenario: for any algorithm that cannot process the incoming data items at their speed, any buffering of the points (to process them offline) would fail, because new data items are constantly arriving. The algorithm could never catch up again and hence any buffer is doomed to overflow. This yields the well known requirement for stream algorithms to be able to keep up with the data stream. For streams with constant arrival rate, the algorithm is simply tailored to the specific amount of time (budget), that is available between two objects.

The varying data streams discussed above pose additional challenges for outlier detection. A straightforward solution would be to resort to a very efficient, but maybe less effective outlier detection algorithm. Such a "worst case" outlier detection would be able to keep up with the stream even in very bursty times, but would be idle most of the remaining time. Trying to estimate an average time budget does still incur idle times and, more critically, incurs the danger of a buffer overflow, since the amount of data to come is unforeseeable. Thus, outlier detection needs to adapt to the stream speed.

Making as much use of the time available for data mining on streams has been discussed in the literature as anytime

StreamKDD'10, July 25, 2010, Washington, DC, USA



(a) Outlier

(b) Varying stream arrival rates correspond to varying time allowance for outlier detection

(c) Prototypical anytime algorithm: good initially, improving with more time available.

Figure 1: Anytime outlier detection.

algorithms [25, 23, 6, 16] for classification or clustering. The general idea is that an anytime algorithm should be capable of delivering an initial result as soon as possible, but be able to improve the mining result if more time is available. In this manner, the algorithm refines its initial solution until interrupted by the next data in the stream.

In this work, we propose the problem of anytime outlier detection. We claim that especially for monitoring applications in medicine, assisted living, sensor networks etc. outlier detection should be capable of keeping up with streams of varying arrival rates, while refining the initial decision on the outlierness of an object if time permits. Moreover, we introduce our approach for anytime outlier detection, show preliminary experiments and discuss future research directions for anytime outlier detection.

2. RELATED WORK

In the literature, different paradigms for outlier detection exist. In supervised learning, the problem is to learn to detect outliers based on labeled historical data, similar in spirit to an unbalanced classification problem [28, 8]. As validated outlier data is typically not available, unsupervised approaches search for outliers as those objects that deviate considerably from the remainder of the data [10].

Distance-based outlier detection searches objects where at least a certain fraction of all objects lies at a distance greater than some threshold [15]. Statistical techniques assume a certain data distribution to identify data that deviates significantly from it [4]. Cluster-based techniques use clustering to describe the underlying structure of the data, i.e. the prevailing patterns, and define outliers as those objects that do not belong to any cluster or do not fit their cluster well [7, 11]. The assumption that deviating behavior is uniform within a dataset was overcome in the local outlier factor (LOF) approach [5], where the notion of an outlierness degree was introduced. An extension to top-n outlier detection was proposed in [14]. Outlier detection for high dimensional data using angle spectrum diversity is studied in [20]. Harmonizing local outlier scores to probabilities is proposed in [19].

Some approaches for stream mining focus on detecting outliers in time series data, e.g. [22]. The goal is identifying patterns in the temporal ordering, which is a different task from detecting outliers in a stream of data objects.

Recently, some approaches for outlier detection on data streams have been proposed [1, 3, 24, 26]. Anguilli et al. [3] compute distance-based outliers within a fixed window over the stream. A statistical approach for outlier detection in streaming data is based on learning a model of the data that accounts for temporal decay of older data in the stream [26]. [1] present a supervised outlier detection approach for multidimensional data streams with a special focus of learning the difference between different types of rare events. For sensor networks, [24] propose a framework for distributed data approximation of multi-dimensional data that can be used to analyze different data properties, such as distance- or density-based outliers, by distributing the computational effort within the network.

However, all of these approaches assume streams of fixed arrival rates and do not meet the requirement of anytime outlier detection that more time leads to better accuracy.

3. ANYTIME OUTLIER DETECTION

In this section, we introduce the problem of anytime outlier detection for data streams of varying interarrival rates. This is followed by an algorithmic solution to the problem.

3.1 Problem definition

Outlier detection is the data mining task concerned with identifying deviating, abnormal data. As illustrated in Figure 1(a), the idea is to compare objects with others in the data collection to identify whether they fit the general pattern of the data.

The goal in anytime outlier detection is to make best use of the time available, as dictated by the stream arrival rate (cf. Figure 1(b)). This means that during busy times, when the stream arrival rate is very high, the detection of outliers needs to be very fast. Conversely, when the stream speed is low, detection of outliers should be capable of using the additional time to determine the outlierness more reliably. In this manner, all available time up to the arrival of the next objects should be used to perform outlier detection as good as possible. The performance of a prototypical anytime algorithm is illustrated in Figure 1(c): good outlier detection accuracy early on, and increasing performance as more time is available.

We formalize the notion of anytime outlier detection in the following:

DEFINITION 1. ***Anytime outlier detection.***

Given a data stream of data objects o_i arriving at unknown interarrival rates, the anytime outlier detection problem is

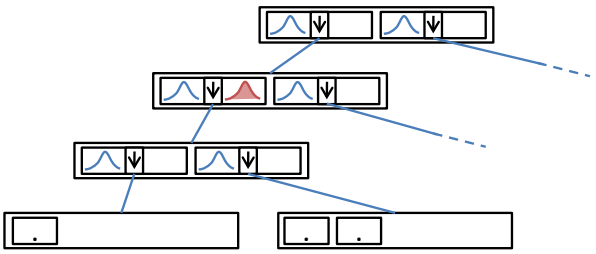


Figure 2: ClusTree structure.

to compute an outlier score $s(o_i)$ in the time t_i between the arrival of o_i and its successor o_{i+1}. The larger t_i, the better the outlier score $s(o_i)$ should reflect the outlier degree of o_i.

Please note that the assessment of the quality of the outlier score $s(o_i)$ is not straightforward. In a controlled experiment, we may use synthetic data or manually label data to compute a ground truth that can be used to evaluate the scores obtained. For real application scenarios, however, only domain experts can assess the outlierness of objects returned with high scores.

3.2 Algorithmic approach

As reviewed in the related work section, different approaches have been discussed in the literature. An approach we will follow here is the clustering-based paradigm [7, 11]: the idea is to use clustering to identify the prevailing patterns in the data and thereby those data items that do not follow these detected patterns. In Figure 1(a), the majority of (black) objects could be detected as a cluster, whereas the single (red) object at the top left is not part of this cluster, and is thereby easily identified as an outlier. Naturally, in many practical applications, the decision boundary between "normal" data and outliers is less clear-cut. We thus use an outlier score to assess the degree of deviation, as in other outlier degree approaches (e.g. LOF [5]).

As stated in Definition 1, the key property of anytime outlier detection is the ability to compute an initial outlier score very efficiently, and then refine it with time. This means that we need to be able to compare an object with the remainder of the data at different granularities.

We propose making use of a hierarchical clustering structure we introduced for anytime clustering in [16]. The idea of the ClusTree is a tree which contains cluster information at various levels. It is a balanced index like the R-tree, R*-tree, etc. [9, 23]. The information stored in each node is a cluster feature that summarizes its subtree as used in [2, 27, 16]. A cluster feature $CF = (n, LS, SS)$ contains the number n of represented objects, their linear sum LS, and their squared sum SS. This tuple suffices for computing mean and variance, and can be incrementally updated. Any cluster feature (CF) then represents a micro-cluster, i.e. a set of objects, and the main characteristics of its distribution. The tree is created and updated like any multidimensional index structure. In addition to these cluster features, nodes also store buffers that are used to temporarily store objects before they are assigned to a cluster. We will not go into details about the anytime clustering algorithm here, for more information please refer to [16]. The general structure of the ClusTree is illustrated in Figure 2: in this example, each inner node stores two entries, which contain a cluster feature (depicted as a small curve), and possibly also a buffer entry (second small curve or empty). Following the pointers to the subtrees, we can descend to lower levels, where the same data is represented at a more fine grained level.

The ClusTree thus provides different representations of the data at different levels of granularity. In the root node, few cluster features summarize the entire data. We can thus easily compare the object to these cluster features to determine whether the object agrees with them or not. If there is more time available, we can descend down the tree to determine the outlierness of the object more closely. At lower levels of the tree, we have more detailed cluster features which is likely to improve the reliability of the outlier score.

We propose an outlier score that is based on how well the object fits its closest entry at the time of interruption by the anytime algorithm. This means that we descend down the tree, and stop when the next object in the stream arrives. At this point, we use the distance between the object and its closest entry as the outlier score for this object.

We define this as the mean outlier score in the following:

DEFINITION 2. ***Mean outlier score.***

Given a data object o_i, its mean outlier score $s_m(o_i)$ is computed as $s_m(o_i) := dist(o_i, \mu(e_s))$, where $\mu(e_s)$ is the mean of the entry where o_i is inserted when o_{i+1} arrives.

A second way of deciding the outlierness is based on the interpretation of the cluster features as parameters of a Gaussian distribution for the subtree. The Gaussian probability density of an object o_i on an entry e_s with mean μ_{e_s} and covariance matrix Σ_{e_s} is given by the following equation:

$$g(o_i, e_s) = \frac{1}{(2\pi)^{d/2} \cdot det(\Sigma_{e_s})^{1/2}} e^{\left(-\frac{1}{2}(o_i - \mu_{e_s})^T \Sigma_{e_s}^{-1} (o_i - \mu_{e_s})\right)}$$

where $det(\Sigma_{e_s})$ is the determinant and $\Sigma_{e_s}^{-1}$ the inverse of Σ_{e_s}. The important part is that we can estimate the probability density of o_i on e_s based on e_s's cluster feature vector to determine its outlierness.

We define this density outlier score as follows:

DEFINITION 3. ***Density outlier score.***

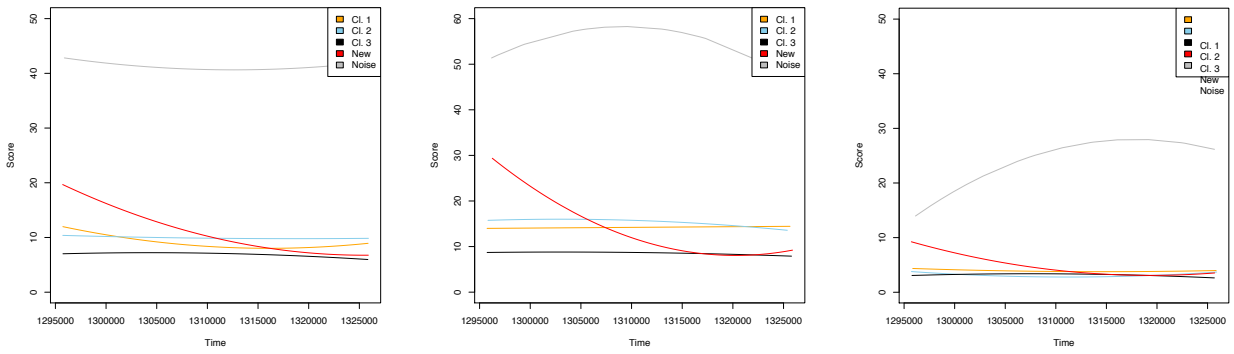
Given a data object o_i, its density outlier score $s_d(o_i)$ is computed as $s_d(o_i) := g(o_i, e_s)$, where e_s is the entry where o_i is inserted when o_{i+1} arrives.

These two scores differ in how they assess the similarity of the object to the nearest entry. In the following section, we will evaluate the performance of these two approaches in preliminary experiments.

4. PRELIMINARY EXPERIMENTS

We evaluate our anytime outlier detection approach on synthetic data to validate that we are capable of successfully identifying hidden outliers, and on real data to demonstrate the practical usefulness of the approach.

As discussed before, our approach is based on the ClusTree structure that provides a cluster representation of the data. We therefore start by building up the tree, and then analyze the outlier detection. The synthetic data (500,000 objects) therefore consists of 210,000 four-dimensional objects that belong to 3 classes that form the "normal" data, plus another 290,000 objects which either belong to the 3 original classes or to a newly introduced class. There is always 5% noise present that forms a class of its own (i.e. 3



(a) Anytime stream results (b) Early interruption (c) Late interruption

Figure 3: Mean outlier score.

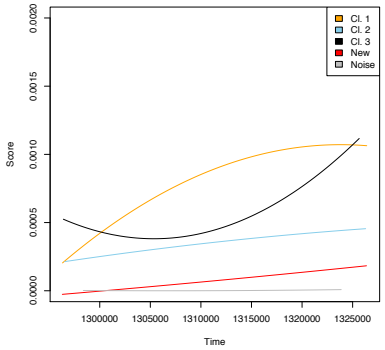


Figure 4: Density outlier score on synthetic data.

+ 1 + noise classes). For all figures, we show the outlier scores (as smoothed trendlines for better readability) in a time window starting from the last "normal" object of the first part of the data, until 5000 objects in the data stream have been analyzed.

Figure 3(a) shows the outlier scores for the different classes. The scores were computed using the mean outlier score (cf. Def. 2), hence low values mean higher similarity. Noise (in gray, top line) is clearly indicated by values that are four times as high as those of consistent data. The novel class (in red, second from top) does not start with scores as high as noise objects, but with values only twice as high as the normal data. This is probably due to the generation that allows noise to be placed in a larger data space radius than the fourth class, making this class more similar to existing clusters. As we can see, the more data from the new class is seen, the lower the scores of other objects that fit with this new cluster structure. This initial result means that our approach shows very promising behavior in terms of identifying outliers (noise), and also of adapting to new concepts over time.

For a more detailed evaluation of the anytime behavior of our outlier detection approach, we have studied the outlier scores with respect to the time that was available until interruption. Generally, we found that the outlier scores are very useful already on the second level below the root (Figure 3(b)), and that the differentiation between the new class and existing classes improved when the algorithm had more time available and could go to leaf level (Figure 3(c)).

We now turn to evaluating the second outlier score, the density outlier score (cf. Def. 3). In this case, a high score indicates similarity. Figure 4 gives an example of the experimental results at level three below the root: the outlier score returned for noise objects is reliably close to 0. The red line – representing the new class – indicates, that the new class is also initially identified as outliers, and later recognized as being more similar to existing data. However, with the density outlier score, this learning process is slower than for the mean outlier score.

Finally, we evaluate the usefulness of the anytime outlier score for a real-world scenario where outlier detection is used for intrusion detection in network traffic data (KDD Cup 1999 [12]). In this data, a peculiar property is that the classes usually arrive in larger stretches. We examined various points in the data, where a change of classes occurs to see if the outlier score reflects this. Results for mean outlier score are given in Figure 5. The scores of 10,000 objects are given: The first half is the end of a stretch of nearly 2 million objects belonging to one class (represented by the line at the bottom). The second half are objects of a new class. The scores of the novel class are considerably higher independent of the interrupt level.

5. OUTLOOK AND FUTURE WORK

The new anytime outlier problem and our proposed technique open up further research questions. We could explore the usage of the different results we obtain for one object as we descend the tree. The individual results at the different levels constitute a vector of scores, which we name the *outlier profile*. In an anytime setting we obtain a dynamic vector that grows with time, i.e. while we descend. This outlier profile can be returned to the user to provide a more detailed reasoning on the outlier decision. Instead of just using the latest score we could combine the scores in an (unweighted or weighted) ensemble approach. The different levels can also be exploited in traditional settings, i.e. where anytime properties are not required. In a training phase the best level or the optimal outlier profile weights can be determined for the given domain.

Another aspect regards the employed tree structure, which

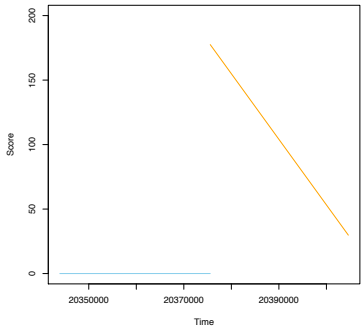


Figure 5: Network traffic: new data type.

so far uses the incremental insertion proposed in [16]. Bulk loading uses larger (entire) sets of data at a time to build up data structures and has been shown to improve the performance in various applications [21, 17]. For anytime outlier detection in a streaming context bulk loading can be used in two different ways. In rather static domains, i.e. where large amounts of training data are available and normal concepts do not change (e.g. health monitoring), bulk loading can be employed once to build up the tree structure for improved anytime outlier detection. In domains with evolving data distributions, as e.g. sensor networks or customer data, bulk loading can be employed in a chunk based approach where new trees are regularly build and employed.

So far we only used the points seen so far to determine the outlierness of a new object. With more objects arriving the outlier degree might decrease due to a newly emerging concept, i.e. the score can also be considered as depending on future observations. In this setting we have to keep a reference to (or a buffer of) outlier points to be able to inspect their score over time. Moreover, a strategy is required that allows the transition from noise or outliers to regular points to reflect novel clusters in the data structure, i.e. the update and insertion process has to be revised.

Finally, the problem of anytime outlier detection can be solved through different outlier paradigms (cf. Section 2). Moreover, recent results show that anytime algorithms are also beneficial in other contexts, e.g. for a pool of instances [13] or on constant data streams [18], by distributing the computation time according to confidences in the individual decisions. In this spirit, anytime outlier algorithms could stop early for credible outliers and spend more time on objects where the outlierness is less clear.

6. CONCLUSIONS

Data streams are ubiquitous and often the amount of data arriving on the stream varies over time. In this paper we proposed the new concept of anytime outlier detection, which handles these varying arrival rates by providing a first result very efficiently and using additional time to improve the reliability of its decision. We sketched an anytime outlier algorithm based on a recent stream clustering approach and showed promising preliminary results. Finally we discussed future research directions for the newly proposed topic.

7. ACKNOWLEDGMENTS

This work has been supported by the UMIC Research Centre, RWTH Aachen University, Germany.

8. REFERENCES

[1] C. C. Aggarwal. On abnormality detection in spuriously populated data streams. In *SDM*, 2005.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[3] F. Angiulli and F. Fassetti. Detecting distance-based outliers in streams of data. In *CIKM*, 2007.

[4] V. Barnett and T. Lewis. *Outliers in Statistical Data*. Wiley, 3rd ed., 1994.

[5] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *ACM SIGMOD*, pages 93–104, 2000.

[6] D. DeCoste. Anytime query-tuned kernel machines via cholesky factorization. In *SDM*, 2003.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *KDD*, 1996.

[8] A. Foss, O. Zaïane, and S. Zilles. Unsupervised Class Separation of Multivariate Data through Cumulative Variance-Based Ranking. In *ICDM*, 2009.

[9] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD*, 1984.

[10] D. Hawkins. *Identification of outliers*. Chapman and Hall New York, 1980.

[11] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 2003.

[12] S. Hettich and S. Bay. UCI ML repository, 2008.

[13] B. Hui, Y. Yang, and G. I. Webb. Anytime classification for a pool of instances. *ML Journal*, 2009.

[14] W. Jin, A. Tung, and J. Han. Mining top-n local outliers in large databases. *KDD*, 2001.

[15] E. Knorr, R. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *VLDBJ*, 2000.

[16] P. Kranen, I. Assent, C. Baldauf, and T. Seidl. Self-adaptive anytime stream clustering. In *ICDM*, 2009.

[17] P. Kranen, R. Krieger, S. Denker, and T. Seidl. Bulk loading hierarchical mixture models for efficient stream classification. In *PAKDD*, 2010.

[18] P. Kranen and T. Seidl. Harnessing the strengths of anytime algorithms for constant data streams. *DMKD Journal*, 19(2):245–260, 2009.

[19] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Loop: Local outlier probabilities. In *CIKM*, 2009.

[20] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *KDD*, 2008.

[21] S. T. Leutenegger, J. M. Edgington, and M. A. Lopez. Str: A simple and efficient algorithm for R-tree packing. In *IEEE ICDE*, pages 497–506, 1997.

[22] S. Muthukrishnan, R. Shah, and J. Vitter. Mining deviants in time series data streams. *SSDBM*, 2004.

[23] T. Seidl, I. Assent, P. Kranen, R. Krieger, and J. Herrmann. Indexing density models for incremental learning and anytime classification on data streams. In *EDBT*, 2009.

[24] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online outlier detection in sensor data using non-parametric models. In *VLDB*, pages 187–198, 2006.

[25] K. Ueno, X. Xi, E. J. Keogh, and D.-J. Lee. Anytime classification using the nearest neighbor algorithm with applications to stream mining. In *ICDM*, 2006.

[26] K. Yamanishi, J. Takeuchi, G. Williams, and P. Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *DMKD Journal*, 8(3):275–300, 2004.

[27] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD*, 1996.

[28] C. Zhu, H. Kitagawa, and C. Faloutsos. Example-Based Robust Outlier Detection in High Dimensional Datasets. In *ICDM*, 2005.