Data Editing Techniques to Allow the Application of Distance-Based Outlier Detection to Streams

Vit Niennattrakul
Dept. of Computer Engineering
Chulalongkorn University
Bangkok 10330, Thailand
g49vnn@cp.eng.chula.ac.th

Eamonn Keogh
Dept. of Computer Science & Engineering
University of California, Riverside
Riverside, CA 92502, USA
eamonn@cs.ucr.edu

Chotirat Ann Ratanamahatana
Dept. of Computer Engineering
Chulalongkorn University
Bangkok 10330, Thailand
ann@cp.eng.chula.ac.th

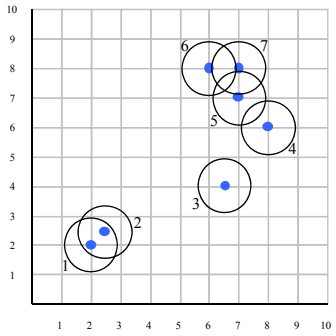
***Abstract*— The problem of finding outliers in data has broad applications in areas as diverse as data cleaning, fraud detection, network monitoring, invasive species monitoring, etc. While there are dozens of techniques that have been proposed to solve this problem for *static* data collections, very simple distance-based outlier detection methods are known to be competitive or superior to more complex methods. However, distance-based methods have time and space complexities that make them impractical for streaming data and/or resource limited sensors. In this work, we show that simple data-editing techniques can make distance-based outlier detection practical for very fast streams and resource limited sensors. Our technique generalizes to produce two algorithms, which, relative to the original algorithm, can guarantee to produce no false positives, or guarantee to produce no false negatives. Our methods are independent of both data type and distance measure, and are thus broadly applicable.**

***Keywords* - Data editing; Anomaly detection; Data stream**

I. INTRODUCTION

Finding outliers in data has broad applications in areas as diverse as data cleaning, fraud detection, telemedicine, invasive species monitoring, etc. Given the ubiquity of this problem, there have been a significant number of solutions proposed, particularly in *static data* collections. However, there is an increasing appreciation of the need to detect outliers in real time on data streams. For the static case, it is well known that very simple distance-based outlier detection methods can be competitive or superior to more complex methods [3]. For example, just in the context of time series data, a recent extensive empirical study compared nine different methods on nineteen different problems, and found that "*[distance-based outlier detection] is the best overall technique among all techniques*" [5]. Furthermore, distance-based methods typically do not require careful settings of many parameters, a notable weakness of many of the more complex or domain specific solutions. Unfortunately, however, distance-based methods have time and space complexities that make them impractical for direct application to streaming data and/or resource limited sensors.

In this work, we show that simple data-editing techniques that were introduced in the context of classification can make distance based outlier methods traceable, both in terms of time and space, for streaming and resource limited applications. To help the reader understand distance-based outlier detection and our modifications to it, consider the simple dataset in Figure 1.



Dataset **D**

	X	Y	*t*
d_1	2	2	0.9
d_2	2.5	2.5	0.9
d_3	6.5	4	0.9
d_4	8	6	0.9
d_5	7	7	0.9
d_6	6	8	0.9
d_7	7	8	0.9

Figure 1. A small dataset **D** containing 7 items. Note that the ordering of points in the database is arbitrary. The variable ***t*** refers to the threshold, the radius of the circles.

In this toy example, we have been given a database of 7 normal objects and a single parameter, the *threshold* of 0.9. If we are asked if a new data point *Q* is normal or an outlier, then we simply scan the database to see if the new object is within 0.9 unit of at least one object in the dataset. Once *Q* encounters a near enough object, we say that *Q* has been *dismissed*. Visually, this corresponds to asking if the new point falls within one or more circles shown in Figure 1. While this algorithm is very simple, and, apart from the *threshold* value, is completely parameter free. It is surprisingly effective, as previous works (with minor differences [5][16]) and later experiments will show.

In this work, we introduce a solution to this problem using a data editing technique [9]. A simple search-based technique uses some heuristic functions to reduce the size of a database, thereby guaranteeing that when elements from a stream are examined with this edited database, the result will be returned to users within a guaranteed time. The penalty we must pay for this guarantee is either (our choice) slightly increasing the false positive rate *or* slightly increasing the false negative rate. In this work, we show that a simple heuristic, iteratively removing one of the current nearest

neighbor pairs, produces excellent results across a wide variety of domains.

This paper is organized as follows. Related work, background material, and the necessary notation are provided in Section II. In Section III, we introduce our main idea, a cost-based editing algorithm, which can efficiently reduce the size of the database while guaranteeing either no false negatives or no false positives. Section IV demonstrates the utility of our proposed technique for various domains. Finally, we offer conclusions and directions for future work in Section V.

II. Related Work and Background

Outlier detection (deviation detection, exception mining, novelty detection, etc.) is an important problem that has attracted wide interest and numerous solutions. These solutions can be broadly classified into several major ideas:

- **Model-Based [2]**: An explicit model of the domain is built (i.e., a model of the heart, or of an oil refinery), and objects that do not fit the model are flagged.
- **Connectedness [11]**: In domains where objects are linked (social networks, biological networks), objects with few links are considered potential anomalies.
- **Density-Based [4]**: Objects in low-density regions of space are flagged.
- **Classification-Based [10]**: Many classification algorithms can be cast as one-class classifiers, which return a binary decision that recognizes the object as either belonging to the same set as the (single-class) training dataset or not.
- **Distance-Based [1]**: Given any distance measure, objects that have distances to their nearest neighbors that exceed a specific threshold are considered potential anomalies.

Model-based methods require the building of a model, which is often an expensive and difficult enterprise, requiring the input of a domain expert. Connectedness approaches are only defined for datasets with linkage information. Density based models require the careful settings of several parameters, have quadratic time complexity, and may have difficulties in datasets that have localized pockets with differing densities.

A. Background on Numerosity Reduction

Numerosity Reduction refers to any technique that reduces the number of instances in a database while retaining some essential properties [15]. It is also called various other names, including prototype selection [9], instance pruning [13], condensing, and data editing. Virtually all numerosity reduction techniques have been proposed in the context of nearest-neighbor classification, not on anomaly detection. The objective functions of these two tasks are different, i.e., maximizing training accuracy for nearest-neighbor classification and maximizing coverage areas for anomaly detection. In addition, the classification task has two or more defined classes, so we just need to define a *decision boundary*, whereas in anomaly detection we typically have just one type of data annotation, "normal data," and we need to define a *decision area*. Unfortunately, the goal of maintaining the best decision area is not computable in polynomial time, because this problem can be mapped to either the minimum set cover problem or the maximum coverage problem; both are NP problems [8].

B. Notations

Table I summarizes the notation used in this paper; we expand on the definitions below.

TABLE I. SYMBOL TABLE

Symbol	Explanation
D	A reference/training database consisting *only* of *normal* objects
d_i	The i^{th} data object in the training database **D**
N	The size of the database **D**, i.e., $N = \|\mathbf{D}\|$
S	A reduced-size training database consisting of some data objects in the database **D**
s	A data object in the reduced-size database **S**
q	A query object from a data stream
K	The size of the database **S**, i.e., $K = \|\mathbf{S}\|$
$d_i.t$	The threshold for i^{th} data object in the training database **D**
$s_i.t$	The threshold for i^{th} data object in the reduced-size training database **S**
b	A user-defined parameter for sensitivity adjustment between having no false negatives and no false positives
$dist_{o1,o2}$	The distance between two data objects o_1 and o_2

We begin by defining the key terms in this paper. Note that all outliers must be outlying with reference to some model or example of normal data:

Definition 1: A *Reference/Training Database* **D** is a collection of N objects which are assumed to be normal. A threshold $d_i.t$ is associated with each data object d_i in D. The initial value of the threshold $d_i.t$ is defined by the user or learned directly from the training database.

As noted above, **D** may be too large to handle streaming data at the required rate, so we need to reduce its size.

Definition 2: A *Reduced-Size Training Database* **S** is a collection of K objects which are all assumed to be normal. All members of the reduced-size database are from the reference database. However, the thresholds $d_i.t$ of data objects in the reduced-size database can be different.

Our hope is that the reduced-size database **S** maintains the properties of the training database **D** with the greatest possible fidelity. The most important property to us is the region implicitly defined as normal.

Definition 3: A *Normal Region* is a volume of space in which all data objects are implicitly assumed to be normal. The normal region contains any data objects which have nearest neighbor distances to objects in the training database less than or equal to thresholds $d_i.t$.

Definition 4: An *Anomalous Region* is an area in which any data objects in this area are considered anomalous. An anomalous region contains any data objects which have distances to all objects in the training database greater than its threshold $d_i.t$. In set notation, it is the complement of the normal region in the Universe.

Definition 5: *Sensitivity* is the measurement of performance in a binary test. It is the number of true

positives, over the sum of the number of true positives and the number of false negatives.

C. Problem Definition

Recall our basic scenario: we have a training set and we learned or were given a threshold T. We are completely satisfied with our anomaly detection system, except that we cannot use it if the data arrives too quickly. More formally:

Given a training database **D**, a computational time α (seconds) for calculating the distance between a pair of objects in our domain, and an arrival rate β (data objects per second) of a data stream, we find a reduced database **S** which allows us to handle this arrival rate, and: Option 1) guarantees no false negatives and minimizes false positives, or Option 2) guarantees no false positives and minimizes false negatives. The size K of the reduced database **S** can be determined by the following equation $K = 1/(\alpha \cdot \beta)$.

Having defined the problem, all we need to do is create an algorithm to reduce the set of objects in **D** to the set of objects in **S**, adjusting the thresholds when necessary. This is the topic of the next section.

III. Cost-based Editing Algorithm

We will begin by showing a simple obvious greedy algorithm for data reduction. This algorithm, unfortunately, is too slow for some of the larger datasets we wish to consider. This motivates us to consider a much faster approximation algorithm in Section III(B). As we will show empirically, the faster algorithm produces near identical results.

A. Simple Obvious Greedy Algorithm

Recall that our task is to take as input a dataset **D**, and the desired final size of K, and produce as output a dataset **S**. We begin by showing Option 1: guaranteeing no false negatives and minimizing false positives.

As shown in Table II, we iteratively remove objects from **D** until it reaches the desired size and then relabel it **S**. Rather than deleting randomly, we delete one object, d_A, which has the smallest nearest neighbor distance, as found in line 3.

TABLE II. Simple Obvious Greedy Algorithm

```
[S] = SimpleObviousGreedyAlgorithm_Option_1 (D, K)
1  while sizeof(D) > K
2     [dA, dB, distdA,dB] = Find_the_Smallest-NN-Dist_Pair in D;
3     Delete dA from D;
4     // dB.t = distdA,dB + dA.t; only if all regions of dA are not in dB
5  end
6  S = D;
```

The Option 2 version of the algorithm is also simple; we just uncomment line 4 in Table II. Instead of just deleting d_A, we expand the threshold of its nearest neighbor (d_B) (line 3) just enough so that it completely envelopes the region formally enclosed by the deleted item. Therefore, a threshold of d_B is only reassigned when all regions of d_A are not in d_B; otherwise the threshold of d_B remains the same.

The *Find_the_Smallest-NN-Distance_Pair* subroutine in line 2 is the most costly part of the algorithm, requiring in the worst case time quadratic in N. We can attempt to mitigate this by using an index of some kind to achieve O(Nlog$N$), but for high-dimensional data, the constants hidden in this are large, and the overall algorithm is too slow to be practical. For example, the Robotics experiment shown in Section III(C) would require 16 hours to reduce the dataset to one-quarter of its size. In the next section, we show an algorithm that can give almost identical solutions, but is orders of magnitude faster.

B. A Faster Greedy Algorithm

We propose to mitigate the costs of the greedy editing algorithm by reducing the search space. The intuition of our idea is very simple to state. Instead of starting with the whole database **D** and using a search to pare it down to database **S**, we randomly take a subset of K+1 items from **D** to create a *temporary* version of **S**. This temporary version of **S** is too large, being of size K+1. So we use the *Find_the_Smallest-NN-Distance_Pair* subroutine to find one object to remove. Note that running this quadratic subroutine is much faster on **S** than it is with **D**. At this point, we are not done; we randomly take another item from **D** and place it into **S**, once again making **S** slightly too large, and once again pruning it down by deleting one of the closest pairs. We continue doing this until we have exhausted all objects from **D**.

Table III formalizes these ideas. We move the first K+1 data objects of the database **D** to the database **S** (line 1-3). For each iteration, we find the closest pair in **S** by running a subroutine *Find_a_Smallest-NN-Distance_Pair*, delete the smallest nearest-neighbor-distance object (s_A), and then replace the deleted object with another data object from the database **D**. The algorithm will terminate when no data object is left in the database **D**.

TABLE III. Faster Greedy Algorithm

```
[S] = FasterAlgorithm_Option_1 (D, K)
1  for 1 to K+1
2     Randomly move a data object in D to S;
3  end
4  while true
5     [sA, sB, distsA,sB] = Find_a_Smallest-NN-Dist_Pair in S;
6     Delete sA from S;
7     // sB.t = distsA,sB + sA.t; only if all regions of sA are not in sB
8     if D is empty then break; end
9     Randomly move another data object in D to S;
10 end
```

As before, the Option 2 version of this algorithm is easy to extend from the Option 1 version, i.e., by just uncommenting line 7.

The threshold $s_B.t$ will be updated with the distance between the data s_A and the data s_B, plus the threshold $s_A.t$ if all regions of the data s_A are not in s_B; otherwise, the threshold $s_B.t$ remains the same.

Calling the subroutine *Find_a_Smallest-NN-Dist_Pair* once in the database **S** consumes much less time than in **D**, i.e., only O(K^2) (or O(Klog$K$) if some indexing techniques can be implemented). Concretely, the algorithm shown

in Table II requires O(N) distance calculations in the first iteration, O(N-1) in the next, etc. Its total time complexity is therefore O(N^3). In contrast, the algorithm in Table III requires a O(K^2) distance calculation to be performed N-K times, and its total time complexity is therefore O((N-K)K^2). When $K << N$, which is the situation we are interested in, this difference can be on the scale of orders of magnitude.

Obviously, this approximation algorithm is ordering dependent. Therefore, in the experimental evaluation, we will demonstrate that this faster algorithm achieves near identical results compared with the simple, but much slower algorithm.

C. *Extended Variable Sensitivity Algorithm*

Since Option 1 and Option 2 are for editing the database with no false negatives and with no false positives, respectively, in this section we propose an extension of the faster greedy algorithm, Option 3, which allows the user to trade off between false negatives and some false positives by using a sensitivity parameter b.

The difference is that a threshold $s_B.t$ is adjusted according to a sensitivity parameter b, where b is a real number between 0 and 1. If b is close to 0, this means a little false negative is tolerated; otherwise, if b is close to 1, this means a little false positive is tolerated. Therefore, the parameter b is used to adjust how large the threshold $s_B.t$ should be. We demonstrate the influence of the parameter b in Figure 2, when different b values, 0, 0.5, and 1, are applied.

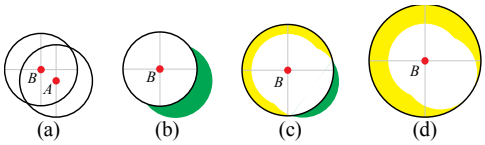


Figure 2. An object A is removed from (a) an original database, and an object B is object A's nearest neighbor, where different parameters b, i.e., (b) 0, (c) 0.5, and (d) 1, lead the difference in false positive (light yellow) and false negative (dark green) areas.

Specifically, the new threshold $s_B.t$ ranges from the original $s_B.t$ in Figure 2b) to the maximum threshold covering all regions of s_A in Figure 2d), which is $dist_{sA,sB}$ + $s_A.t$. As shown in Table IV, in line 7, the parameter b is used to determine the new threshold. However, this new threshold $s_B.t$ will be updated only if all regions of s_A are not in s_B; otherwise, no change will occur in the threshold $s_B.t$.

TABLE IV. VARIABLE SENSITIVITY GREEDY ALGORITHM

	[**S**] = VariableSensitivityAlgorithm (**D**, K, b)
1	**for** 1 to K+1
2	Randomly move a data object in **D** to **S**;
3	**end**
4	**while** true
5	[s_A, s_B, $dist_{sA,sB}$] = Find_a_Smallest-NN-Dist_Pair in **S**;
6	Delete s_A from **S**;
7	$s_B.t$=$s_B.t$+($dist_{sA,sB}$+$s_A.t$-$s_B.t$)*b; if all regions of s_A are not in s_B
8	**if D** is empty **then** break; **end**
9	Randomly move another data object in **D** to **S**;
10	**end**

IV. EXPERIMENTAL EVALUATION

We begin by stating our experimental philosophy. To ensure that our experiments are easily reproducible, we have built a website which contains all data and code [17]. In addition, this website contains additional experiments that are omitted here for brevity. Nevertheless, we note that this paper is self-contained.

In this section, we will empirically demonstrate that our proposed ideas can significantly improve the efficiency of distance-based outlier detection while maintaining its high effectiveness. We evaluate our Option 1, which guarantees no false negatives by varying the reduced size (K). Naturally, we evaluate by calculating the increase in the false positive rate as we reduce the size of the dataset, and the proposed sensitivity parameter b for Option 3 is also evaluated. The false positive rate and false negative rate of each parameter b is reported against the reduced-database size.

To measure the effectiveness of our editing heuristics, we compare our algorithm with a random-based editing technique. For the random-based editing technique, given a reduced database size K, K data objects from the database **D** are uniformly selected. Note that we do not make comparisons to other anomaly detection algorithms because distance based outliers for time series have already been forcefully shown to be the best for time series in extensive empirical tests ([5][16]).

For each technique, the mean values of ten runs are reported. Our algorithm and the rival method are implemented using Matlab R2010a. All experiments were conducted on a Windows Vista Ultimate SP1 64-bit desktop with 2.83 GHz Intel Core 2 Quad CPU, 4 GB of RAM, and 250 GB of Hard Disk.

A. *Threshold Initialization*

The distance-based anomaly detection method used in this work requires only one parameter, the initial threshold t of the training dataset. In some domains, this may be given by the domain expert, but here we use a simple approach to determine it for the ground truth of results. The threshold $d_i.t$ for each data object d_i is set to be the nearest neighbor distance of d_i itself in the training dataset. With this simple threshold initialization method, our distance-based anomaly detection can produce very effective results. Note that this value t is a parameter of distance-based anomaly detection in general, *not* of our extension to it, which requires no additional parameters. Further note that we are explicitly avoiding changing parameters after seeing the *test* data

B. *The SmartCane System*

The SmartCane system [14] is a device developed by researchers at UCLA to enable training and monitoring usages of canes for the elderly and infirm. The goal of this system is to reduce falls, which are one of the leading causes of death in the elderly. Specifically, the SmartCane system records walking behavior from embedded sensors at 300 Hz, and sends data via Bluetooth back to a personal device such as a tablet PC or a PDA to record usage activities. This system allows physicians to review histories of cane usage and can suggest future customization and training which may

reduce falls. In addition to offline analysis, this system is general enough to allow for the possibility of *real time anomaly detection*, which, in the best case scenario, may warn that the user is disorientated and at risk of an imminent fall.

The device, as shown in Figure 3, is embedded with six sensors, two contact pressure sensors (at the handle and bottom), three single-axis gyros, and one 3-axis accelerometer; therefore, eight channels of signals are simultaneously collected over time. Figure 3 shows an example subsequence of these eight signals.

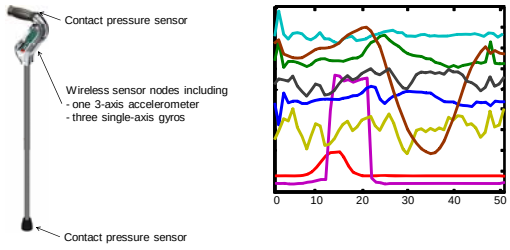


Figure 3. *left)* SmartCane *right)* sample data of all eight channels.

We divide a sequence of length 3244 into two equal parts, i.e., a training database and a test database. Figure 4 shows the first 500 data points of the two databases. To achieve our results in this particular domain, we had to set exactly one free parameter, with the length of the subsequence to consider. For simplicity, we decided on a length of 50. However, it is important to note that a change in this value only very slightly affects results.

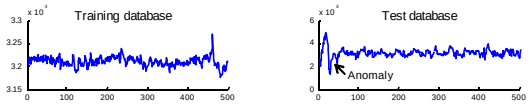


Figure 4. A time series sequence of Gyro3, where the test database contains six anomalies.

In Figure 5, we show that we can throw away more than three-fourths of the data, and still have comparable results to the approach in [12]. Our cost-based editing techniques can significantly reduce a large portion of data, while the false positive rate slightly increases it. In addition, the false negative rate is guaranteed to not increase since Option 1 has been applied.

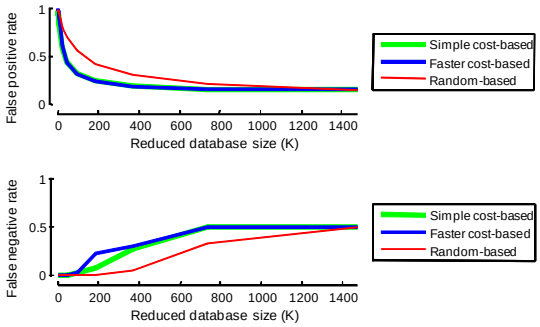


Figure 5. Our editing technique can significantly reduce database size with a slight increase in false positive rates.

With the greater time complexity of the simple cost-based method, training time (reducing database time) is much longer than the faster cost-based method. Generally, when the database size is reduced to 10% of its original size, the faster cost-based method uses only 1 second, while the simple cost-based method requires 42 seconds.

The sensitivity parameter b is utilized to trade off between the false positive and false negative rates. Therefore, users can choose the parameter b according to their applications. We demonstrate the effect of false positive/negative rate when the parameter b is varied in Figure 6.

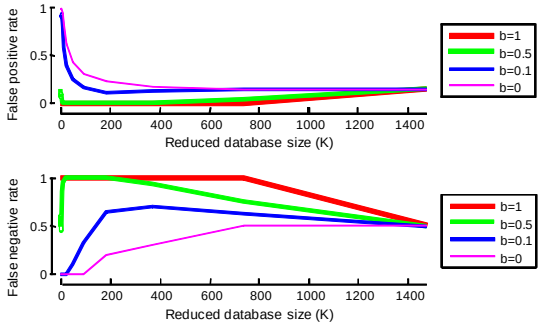


Figure 6. *top*) False positive rates for the faster algorithm when b is varied; *bottom*) False negative rates for the faster algorithm when b is varied.

C. *Robotics*

The Sony AIBO shown in Figure 7 is a small quadruped robot that comes equipped with a tri-axial accelerometer [7]. This accelerometer measures data at a rate of 125 Hz. We can easily obtain a large normal reference, Database **D**, for the AIBO by allowing the robot to walk unobstructed in normal conditions. However, this tiny robot has limited space[1] in which to store these normal examples. Once again, this is exactly the scenario for which our algorithms are designed for.

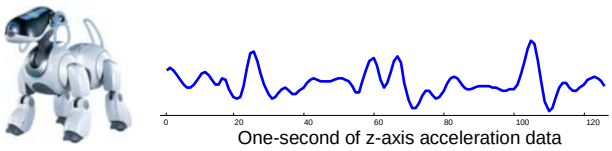


Figure 7. *left*) A Sony AIBO robot; *right*) An on-board sensor can measure X/Y/Z acceleration at 125 Hz. Here, just a snippet of the Z-axis is shown.

We created a training dataset consisting of unobstructed walking, and a carefully annotated test dataset consisting of unobstructed walking interspersed with occasions where the robot walked into a wall (labeled as anomalous). Each dataset consists of 6000 data points, where Figure 8 shows the first 3000 data points. Subsequences were extracted with a window length of 50 data points and, as is common

[1] The AIBO has 64MB of memory, but this figure is for the entire system, so space efficient algorithms are critical.

practice for time series, we normalized all subsequences with z-normalization [6][15].

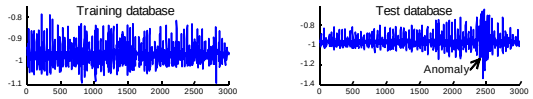


Figure 8. Illustration of robotic dataset, *left*) sample training dataset; *right*) sample test dataset. In this test dataset, five anomaly sequences were randomly inserted.

In Figure 9, we show how our algorithm's performance gracefully degrades as we use smaller and smaller values of *K*. As we can see, we can more than halve the required space, with only a tiny increase in the false positive rate. In this experiment, the two cost-based approaches are demonstrated, the simple obvious greedy one and our faster greedy algorithm. As we can see in Figure 9, our faster algorithm can achieve a performance nearly equal to that of the simple greedy algorithm, our faster algorithm requires only a fraction of the time required by the simple algorithm.

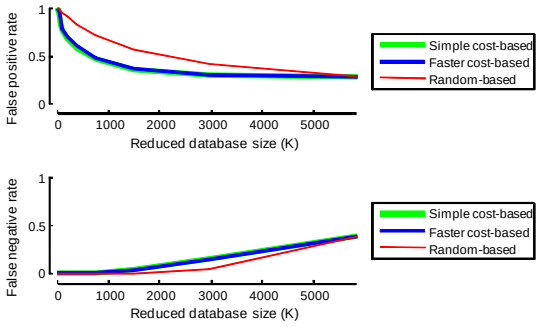


Figure 9. The false positive rates of three approaches (simple obvious cost-based, faster cost-based, and random-based editing techniques) against reduced database size.

V. CONCLUSION

In this work, we have introduced a framework to *allow efficient distance-based* anomaly detection in data streams. Our method borrows from a well-known idea in classification, which is to use data editing to reduce the size of the database, while maintaining critical properties. Our proposed framework guarantees either no false positives or no false negatives relative to the performance of the full dataset, or allows the user to choose sensitivity/specificity based on his/her tolerance to each kind of risk.

ACKNOWLEDGEMENT

This research was funded by the Thailand Research Fund given through the Royal Golden Jubilee Ph.D. Program (PHD/0141/2549 to V. Niennattrakul and C.A. Ratanamahatana) and NSF awards 0803410 and 0808770.

REFERENCES

[1] F. Angiulli and F. Fassetti, "Detecting Distance-based Outliers in Streams of Data," In Proceedings of CIKM'07, pp. 811-820, November 6-10 2007.

[2] F. J. Anscombe and I. Guttman, "Rejection of Outliers," Technometrics, vol. 2, pp. 123-147, May 1960.

[3] S. D. Bay and M. Schwabacher, "Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule," In Proceedings KDD'03, pp. 29-38, August 24-27 2003.

[4] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander, "OPTICS-OF: Identifying Local Outliers," In Proceedings of PKDD'99, pp. 262-270, September 15-18 1999.

[5] V. Chandola, D. Cheboli, and V. Kumar, "Detecting Anomalies in a Timeseries Database," CS Technical Report 09-004, Computer Science Department, University of Minnesota, January, 2009.

[6] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures," Proceedings of the VLDB Endowment, vol. 1, pp. 1542-1552, August 2008.

[7] G. S. Hornby, M. Fujita, S. Takamura, T. Yamamoto, and O. Hanagata, "Autonomous Evolution of Gaits with the Sony Quadruped Robot," In Proceedings of GECCO'99, pp. 1297-1304, July 13-17 1999.

[8] R.M. Karp, "Reducibility among Combinatorial Problems," Complexity of Computer Computations, 1972, pp. 85-103.

[9] E. Pękalska, R. P. W. Duin, and P. Paclík, "Prototype Selection for Dissimilarity-Based Classifiers," Pattern Recognition, vol. 39, pp. 189-208, February 2006.

[10] V. Roth, "Kernel Fisher Discriminants for Outlier Detection," Neural Computation, vol. 18, pp. 942-960, April 2006.

[11] J. Tang, Z. Chen, A. W.-C. Fu, and D. W.-L. Cheung, "Enhancing Effectiveness of Outlier Detections for Low Density Patterns," In Proceedings of PAKDD'02, pp. 535-548, May 6-8 2002.

[12] A. Vahdatpour and M. Sarrafzadeh, "Unsupervised Discovery of Abnormal Activity Occurrences in Multi-dimensional Time Series, with Applications in Wearable Systems," In Proceedings of SDM'10, pp. 641-652, April 29 - May 1 2010.

[13] D. R. Wilson and T. R. Martinez, "Instance Pruning Techniques," In Proceedings of ICML'97, pp. 403-411, July 8-12 1997.

[14] W. Wu, L. Au, B. Jordan, T. Stathopoulos, M. Batalin, W. Kaiser, A. Vahdatpour, M. Sarrafzadeh, M. Fang, and J. Chodosh, "The SmartCane System: An Assistive Device for Geriatrics," In Proceedings of the ICST 3rd International Conference on Body Area Networks, pp. 1-4, 2008.

[15] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast Time Series Classification using Numerosity Reduction," In Proceedings of ICML'06, pp. 1033-1040, June 25-29 2006.

[16] D. Yankov, E. Keogh, and U. Rebbapragada, "Disk Aware Discord Discovery: Finding Unusual Time Series in Terabyte Sized Datasets," Knowledge and Information Systems, vol. 17, pp. 241-262, November 2008.

[17] All code and data used in this paper: http://www.cp.eng.chula.ac.th/~g49vnn/ICDM2010