I.J. Intelligent Systems and Applications, **2021, 1, 58-68**
Published Online February 2021 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2021.01.05

Genetic-based Summarization for Local Outlier Detection in Data Stream

Mohamed Sakr, Walid Atwa and Arabi Keshk
Computer Science Dept. Faculty of Computers and Information, Menoufia University, Egypt
E-mail: mssakr@ymail.com, walid.atwa@ci.menofia.edu.eg, arabikeshk@yahoo.com

Received: 03 February 2020; Revised: 08 March 2020; Accepted: 16 March 2020; Published: 08 February 2021

Abstract: Outlier detection is one of the important tasks in data mining. Detecting outliers over streaming data has become an important task in many applications, such as network analysis, fraud detections, and environment monitoring. One of the well-known outlier detection algorithms called Local Outlier Factor (LOF). However, the original LOF has many drawbacks that can't be used with data streams: 1- it needs a lot of processing power (CPU) and large memory to detect the outliers. 2- it deals with static data which mean that in any change in data the LOF recalculates the outliers from the beginning on the whole data. These drawbacks make big challenges for existing outlier detection algorithms in terms of their accuracies when they are implemented in the streaming environment. In this paper, we propose a new algorithm called GSILOF that focuses on detecting outliers from data streams using genetics. GSILOF solve the problem of large memory needed as it has fixed memory bound. GSILOF has two phases. First, the summarization phase that tries to summarize the past data arrived. Second, the detection phase detects the outliers from the new arriving data. The summarization phase uses a genetic algorithm to try to find the subset of points that can represent the whole original set. our experiments have been done over real datasets. Our experiments confirming the effectiveness of the proposed approach and the high quality of approximate solutions in a set of real-world streaming data.

Index Terms: Outlier detection, data streams, local outlier factor, genetics.

1. Introduction

Outlier detection is also known as anomaly detection has gained a lot of importance and attention in the field of data mining. It has been used in many applications such as credit card fraud detection and intrusion detection in web apps. A lot of algorithms have been developed to detect outliers in static data in which the number of points are determined and doesn't change over time. However, detecting outliers on streamed data is difficult because the size of the data set is infinite, and the data is changing over time thus can't be stored in memory for processing [1].

One of the techniques that are used in outlier detection is density-based techniques. Density-based techniques have a great ability to detect outliers in different densities and dealing with nonhomogeneous densities datasets.

One of the well-known algorithms for outlier detection that is density based is Local Outlier Factor LOF. LOF has been used in data sets with heterogeneous densities [2, 3, 4]. However, LOF deals with static data that don't change over time as its calculations are done over the whole data one time. Because LOF did its calculations one time on the whole data it needs a huge amount of memory to store the data to process. Specifically, LOF has $O(n^2)$ space complexity to detect outliers as it stores all the points of the data and its distances between the all points. Also in any change in data by adding or deleting any points the LOF needs to be recalculated on the whole data set. Such these limitations of LOF, it can't be used with data streams as data streams size are infinite and data are changing over time as new points arrive [5].

A data stream is a continuous data records ordered by timestamps and the data points are available partially at any given point in time. Thus, when working on applications with streaming data, their temporal contexts need to be considered. In addition, the processing needs additional requirement on computational and memory resources. There are many applications that detect outlier detection over streaming data, such as network detection, fraud detections, and environmental monitoring. Thus, we need to find abnormal data over data streams in real-time.

Researchers have proposed different solutions to this problem. One of those solutions works by using sliding-window in the application and performing learning only on those windowed data. This solution performs well in some applications and also makes real-time results. However, the correctness of its results depends largely on the size of window that is not considered. There are other existing solutions but most of them fail to address those properties of streaming data, and thus produce results exhibiting poor accuracy [6].

In this paper, we aim to propose new algorithm called GSILOF (Genetic Summarizing Incremental LOF). that overcome aforementioned challenges in streaming data. The GSILOF algorithm consists of two phases 1) detection

phase 2) summarization phase. in the detection phase GSILOF algorithm tries to detect the outliers from the arrived data points. In the detection phase every new point is checked for being an outlier or not by calculating the LOF for each point. To calculate the LOF for each new arriving point the detection phase uses a modified version of LOF called Incremental LOF (ILOF) [6]. ILOF overcome the problem of LOF as it doesn't require the recalculation of the whole data points when the data set changes by insertion or deletion. In the ILOF it detects when a new point is added or deleted and update the points which may be affected.

In the summarization phase, we develop a novel density-based sampling algorithm that samples the past data while preserving the density of the remaining points. We use genetic algorithms to try to optimize the summarization function to select the optimal subset that represents the whole set and minimize the density change between the two sets (whole and sub).

The rest of the paper is organized as follows: section 2 gives a review on the related work. Section 3 describes the GSILOF. Section 4 views the experimental results over real data sets. Finally, the conclusion is presented in section 5.

2. Related Work

Many surveys of outlier detection are mentioned. However, most of them focus on the problem of finding outliers in static data in which all the data points are available as a whole and its size is pre-calculated [7, 8]. Streaming environment, a large amount of data is being generated at high speed and volume. So, outliers need to be detected in limited time at one pass. Many categories for outlier detection in data streams have been proposed [6, 9, 10]: distance based, a distribution based, and clustering based.

In distance-based category outliers are detected by measuring the distance from a point to the other points in the data set. The detected outlier can be in one of two types global or local. A data point p is a global outlier if the number of points within a given distance is less than k point [11].

Many adaptions to this approach have been done to use it in data streams. In [12] the authors proposed an algorithm deal with the sliding window model, where outlier queries are performed in order to detect anomalies in the current window. However, it has limited memory requirements and returns an approximate answer based on accurate estimations with a statistical guarantee.

In [13] the authors introduced a method where an outlier is defined by taking into account the sum of the distances from 1^{st} up to the k^{th} nearest neighbors. The authors in [14] enhanced the algorithm in [12] by reducing the time complexity and memory consumption needed. They first propose a continuous algorithm which has two versions. The first version requires the radius R to be fixed but it can handle multiple values of K neighbors. The second version can handle multiple values of R and K. second they propose an algorithm which is based on micro clusters to reduce the number of distance computations. However, the time complexity of this algorithm is guaranteed to be $O(n \log k)$ while maintaining the space complexity to be $O(nk)$, where n is the number of data points and k refers to the parameter of KNN (K-Nearest Neighbourhood).

For the local outliers, a point is considered a local outlier if it is far by d distance with respect to its k neighbors. Authors in [15] proposed a new factor that calculates the degree of oulierness for each point called the Local Outlier Factor (LOF). It uses the concept of reachability to define the density of data points: the density of each data point is measured by considering the reachability of this data point, in regards to the reachabilities of its neighbours. One of the main features of the LOF technique is that it can detect outliers in non-homogeneous densities with high accuracy. Also, it doesn't need any assumptions regarding the underlying distribution of the dataset. Authors in reference [6], presented an incremental version of LOF over streaming data. The authors gave theoretical evidence to show that the insertion of new data points as well as deletion of an old data point affects only a limited number of neighbors.

In distribution-based model outliers are detected by having a priori knowledge of the distribution of the data set and compare it with the new incoming data. The detection model is applied to fit the data with the distribution of data using a mixture of distributions. In references [16, 17] authors used the Gaussian mixture model (GMM), where the dataset is equipped to a certain number of Gaussian distributions and the model is trained using Expectation-Maximization (EM) method. These models are usually having low computational resources, but most of them require parameters as inputs and they also assume a fixed distribution in dataset, that is not appropriate with streaming data.

Clustering based categories try first to divide the data into clusters depend on the distribution of the data. Then, some algorithms mark the clusters with a small number of points as outliers. other algorithms mark the points that are far from the clusters by a threshold as outliers. However, most of these algorithms were proposed to cluster the data sets rather than detecting outliers, e.g., [18, 19, 20].

Authors in [21] proposed another clustering algorithm to outlier detection over streaming data, they used the idea of sliding window and clusters the data in each window. the detected outliers are not represented immediately but rather considered as candidate outliers. They maintained the mean value of each cluster and carried over to the next window in the stream to further compare with other windows. If the candidate outlier passed a given number of windows, it is then identified as true outlier.

Authors in [22] propose an algorithm that extended affinity propagation to handle evolving data steam with dynamic distribution and capable of performing fast and incremental processing of data objects, generate histograms for

the clusters in the streamed data which used later for mining and also in detecting outliers.

In [23] authors proposed an algorithm that works on the problem of determining within any period of time whether an object in a data stream is outliers. They used a cluster-based approach to represent data in a hierarchical fashion to determine an outlier score based on the deviation between object and cluster. In [24] authors proposed a cluster-based approach that is used in outlier detection. they focus on data streams with varying arrival rates. They try to learn from the history of the data stream to determine the normal behavior of the current period. The advantage of this approach is that it can improve the accuracy of detection by using relevant history, while remaining computationally.

3. Genetic Summarizing Incremental LOF (GSILOF)

In this section, we present how to solve the problem of detecting outliers in a data stream over limited memory environment. The proposed algorithm GSILOF consists of two phases: 1) the detection phase 2) the summarization phase. The detection phase uses the ILOF to detect the outliers. The summarization phase called Genetic Summarization (GS) which uses a Genetic algorithm (GA) that work parallel to find the best solution. GAs work on multiple solutions in many directions. It tries to find the best solution from many solutions, so it doesn't stick in local optima.

3.1. Detection phase

when new point arrives the detection phase starts to calculate the LOF score for this point. And check if this score is below the predefined threshold, if yes this means that this point is inlier. If the score is above the threshold, then this point is considered an outlier. To calculate the LOF for each point a modified version of the LOF is used called ILOF which handles the drawbacks of the LOF to work with streaming data. The ILOF doesn't require the recalculation for the all previously entered data but only the affected points by the new point. For the LOF algorithm, it must loop through all the data points when a new point is inserted to update the LOF score, in contrast, the ILOF algorithm loops through the affected points only by the insertion of the new points. when a new point is inserted the ILOF searches for the data points whose neighbor are affected. Then the ILOF selectively updates lrd_k and LOF_k of them. For more details about the incremental insertion of iLOF, readers are referred to [6].

3.2. Summarization phase

GA is based on Darwin's theory of natural selection and "survival of the fittest" [25]. GA is used in searching among many solutions and finding the optimal one. It depends on randomly selecting a number of solutions and calculate the degree of how good these solutions are. Each solution is represented by a chromosome. To calculate the solution degree and rank, a fitness function is used. A generation consists of number of chromosomes. After calculating the fitness function for each suggested solution (chromosome) a generation is finished. Part of the chromosomes is selected to be part of the next generation. By applying crossover and mutation functions new chromosomes are generated (best generate the best). After that the old selected chromosomes and the newly generated ones are combined to construct the new generation. These iterations are done until a pre-defined threshold. in the end the best solution is selected for the minimization or maximization function. The solution consists of the best chromosome that achieves the best results for the objective function.

The GS algorithm summarizes the points in the window by selecting a small number of points that represent the total points and minimizing the density difference between the old set and the new one. The GS algorithm depends on the Nonparametric Density Summarization in [26] except that the authors in [26] use gradient descent to select the subset and minimize the density difference the GS uses Genetic algorithm to apply the minimization function. Genetic algorithm is better than the gradient descent as it tries to find many optimal solutions and solves the local minima not like the gradient descent that can be stuck in the local minima.

The GS algorithm uses the objective function to evaluate each chromosome as shown in equation 1. For given two datasets $Z = \{z_1,z_2,...,z_N\}$ and $X = \{x_1,x_2,...,x_M\}$ where Z is a proper subset of X , $\rho k(x_n)$ is the Euclidean distances between data point x_n and its k^{th} nearest neighbor in Z, and $vk(x_n)$ is the distance between x_n and its k^{th} nearest neighbor in X .

$$\sum_{X \in C_{k,n}} \frac{\rho k(x)}{vk(x)} + \frac{\rho k(x_n)}{vk(x_n)} - e^{LOF_k(x_n)} + \psi_{0,1}(y_n) + \lambda\left(\sum_{i=1}^{\frac{w}{2}} y_i - \frac{w}{4}\right) \tag{1}$$

where, w is the window size, y_n is the decision variable $\in \{0,1\}$, λ regularization constant

$$\psi_{0,1} = \begin{cases} (y_n - 1)^2, & if\ y_n > 1 \\ {y_n}^2, & if\ y_n < 0 \\ 0, & otherwise \end{cases}$$

where $C_{k,n}$ is a set of data points that have x_n as their k^{th} nearest neighbor in Z. y_n is decision variables with a value of 1 when the data point x_n is selected, used in order to transform the combinatorial optimization problem in [27] to a solvable form.

3.3. GSILOF Steps

The *GSILOF* algorithm takes the maximum window size *ws* which represent the total number of points that will remain in memory. The threshold θ of the LOF to mark any point that has LOF more than θ to be an outlier. The number of chromosomes *nc* and number of Generations *ng* that are given to the Genetic algorithm to summarize the points in the memory. When new point arrives the GSILOF algorithm will calculate the LOF for this point and check the LOF against the threshold θ to determine if this point is an outlier or not. Then this point will be added to the current window line (4-8). This process is repeated when new points arrive. When the number of the points in the window reaches *ws* the summarization phase starts line (9-11).

The core step in the summarization phase is to take the oldest *w/2* points in the current window which is *A* and *B* and summarize them by selecting only half of them which is *w/4* and removing the remain *w/4* from the window. The GSILOF initiate the population and add *nc* chromosome in it. The number of genes in every chromosome is *w/2*. then evaluate every chromosome using the fitness function shown in equation 1 line (12-17) the fitness function will process on the oldest *w/2* points. After initiating the population, the GSILOF starts a number of generations *ng* looping over them, and in each generation the GSILOF line (18-24). after the population generation and evaluation is finished the GSILOF starts to transform the best chromosome into *Y* to use it to select the most unimportant points to remove the line (25). The GSILOF loop through *Y* and check if the value of the Y_n is 1. Then, it adds the points in the current half of the window in *n* index to *WD'* line (25-32). At the end the GSILOF removes the old *w/2* points from the current window and adds the new selected *points* to the window line (33-34). Since GS summarizes the oldest *W/2* data points to *W/4* data points whenever the number of data points reaches *ws*, the number of data points to be processed by GSILOF is always bounded by *ws*. The size *ws* is typically determined by the memory limitation.

Algorithm 1 GSILOF

input: infinite data stream $P=\{p_1,p_2,...,p_t,...\}$, maximum window size *ws*, threshold of LOF scores θ, number of Generations *ng*, number of chromosomes *nc*

```
1   WD ← {} // current points in the window
2   DO←{}   /current detected outliers
3   Foreach p_t ∈P do
4       LOF_k(p_t) ← ILOF(p_t, DO,θ)
5       If LOF_k(p_t) >= θ
6           Add p_t to DO
7       End
8       Add p_t to WD
9       if | X | != ws then
10          Continue;
11      End
12      P ← {} // populations
13      Create Population P consists of nc chromosomes
14      Foreach ch ∈ P do
15          Initiate chromosome ch in p
16          Evaluate ch using the Objective Function (Equation 1)
17      End
18      Foreach g in ng
19          Apply selection to P using roulette wheel selection
20          elitism P // saving best chromosomes in new population, making a copy of each elite chromosome
21          Apply cross-over method over P // one-point random cross-over of 2 chromosomes
22          Apply mutation to P  //single point mutation: flipping a chromosome bit
23          Save best result
24      end
25      Transform the resulted chromosomes to Y
26      Project Y into binary domain
27      WD' ← {} // set of points after genetic summarization finish
28      for n=1:W/2 do
29          if Y_n = 1 then
30              WD' ← WD' ∪ { WD_n }
31          End
32      End
33      Remove oldest W/2 data points in WD
34      WD ← WD ∪ WD'
35  End
```

For example, as shown in fig 1 in time *t1* when new points *A* arrive the GSILOF algorithm will calculate the LOF for this point and check the LOF against the threshold θ to determine if this point is an outlier or not after that it will be added to the current window. Repeating again in time *t2* when new points *B* arrives. Now the current window contains points *A* and *B*. The same for time *t3* when new points C arrive. Now the current window contains points *A, B* and *C*. When new points *D* arrive in time *t4* the GSILOF algorithm repeats again and now the current window will contain *A,B,C* and *D*. then the GSILOF checks if the current size of the points in the window reaches *ws* which is yes, Then the summarization phase in GSILOF starts. the fitness function will process on the points *A* and *B* only (the oldest *w/2*). After the summarization phase ends *A* and *B* now becomes *AB'* and the size of *AB'* is half the size of *AB.* At the end the GSILOF removes the old *w/2* points (*A* and *B*) from the current window and adds the new selected *AB'* to the window line.

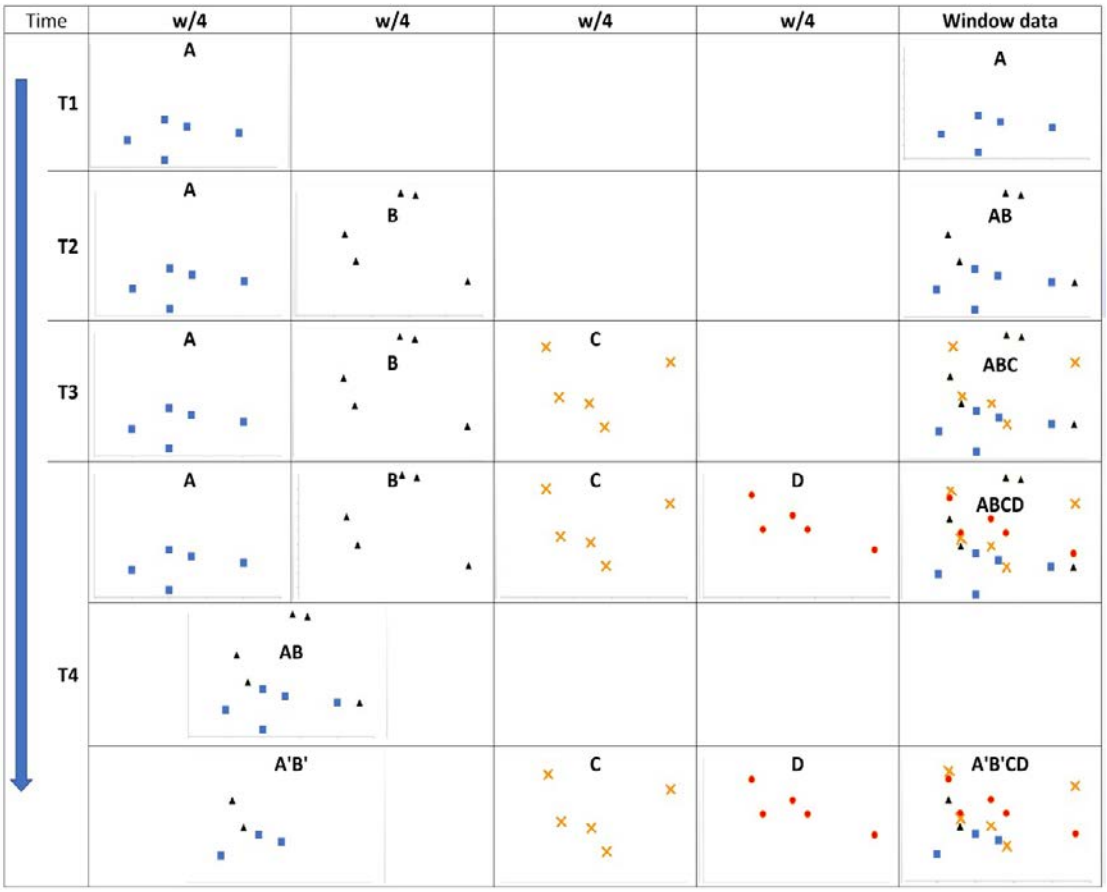

Fig.1. How Genetic summarization works when new points arrive

4. Experiment

This section presents the experimental results of the GSILOF algorithm. In addition, a comparison between the GSILOF algorithm and the DILOF algorithm in [26] are presented. The GSILOF is implemented in C++ and is based on the source code for DILOF (http://di.postech.ac.kr/DILOF). Both of the algorithms are deployed over a machine has a 2.5 GHz Intel Core i7 CPU, 16G memory DDR3, and 512G SSD hard disk.

We compare GSILOF with the DILOF algorithm in terms of outlier detection accuracy and execution time. The accuracy is measured using Area Under the Curve (AUC). Execution time is the time taken to finish processing all the data points in the data stream.

Three real-world datasets were used in the evaluation of the GSILOF algorithm. All the three datasets are obtained from the Machine Learning database repository at UCI (http://archive.ics.uci.edu/ml/) [28]. The datasets are:

1- UCI Vowel : this dataset is modified to be suitable for data stream format like in [29]. The class 1 is down sampled to 50 outliers. this dataset consists of 1,456 data points, 12 dimension and 11 classes.
2- KDD Cup 99 smtp: for this dataset we set the network attacks to be outliers as in [16]. The dataset consists of 95,156 data points and 3 dimensions.
3- KDD Cup 99 http: this dataset consists of 567,479 data points and 3 dimensions.

For the experimental setup, the hyper-parameters of DILOF, η and λ are fixed to 0.3 and 0.001 for all datasets. And the number of the iteration is set to be 20. We set *k* to be 19 for the UCI Vowel dataset and 8 for the both KDD Cup 99 smtp and http datasets. For the summarization phase we set the window limit *w* to be *W* = {100, 120, 140, 160, 180, 200} for the UCI Vowel dataset and *W* = {100, 200, 300, 400} for both KDD Cup 99 smtp and http datasets.

4.1. The outlier detection accuracy using Area Under the ROC Curve (AUC)

We evaluate the AUC for the two algorithms GSILOF and DILOF. Fig 2, Fig 3 and Fig 4 shows the AUC for the three datasets UCI Vowel, KDD Cup 99 smtp and KDD Cup 99 http respectively. For all the datasets the GSILOF has higher AUC than DILOF. Also, for all the datasets the AUC percentage increases when the window size increase this is for GSILOF algorithm. This is due to that when we increase the window size the number of processed points becomes large which increases the detection accuracy for the outliers [30-37]. for the KDD CUP 99 smtp dataset we notice that the AUC for the DILOF isn't guaranteed to become bigger when we increase the window size as noted in window size 100, 200 and 300. Which changes from high to low then high again. This is because the DILOF depends on gradient descent which may start solving the problem of the summarization using local minima not the global minima and stuck in this solution up and high. In contradiction the GSILOF depends on the genetics for the summarization which random many solutions and overrides the problem of stuck in local minima.

We notice that for all the datasets the difference between the AUC for the two algorithms starts to be big and become smaller as the window size increases but don't reach zero or negative value as the AUC for the GSILOF are always higher than the AUC of DILOF.

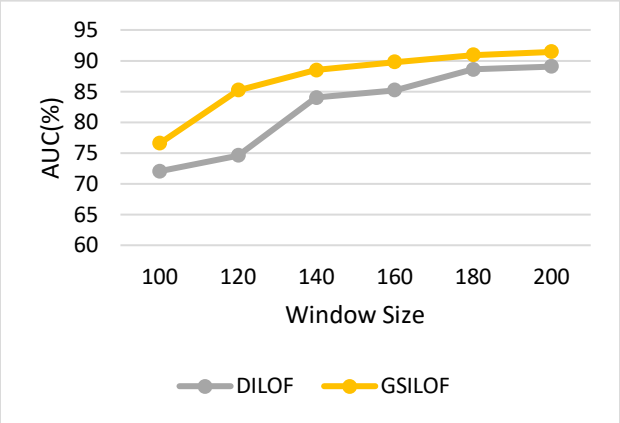

Fig.2. UCI Vowel Dataset Outlier Detection Accuracy using AUC

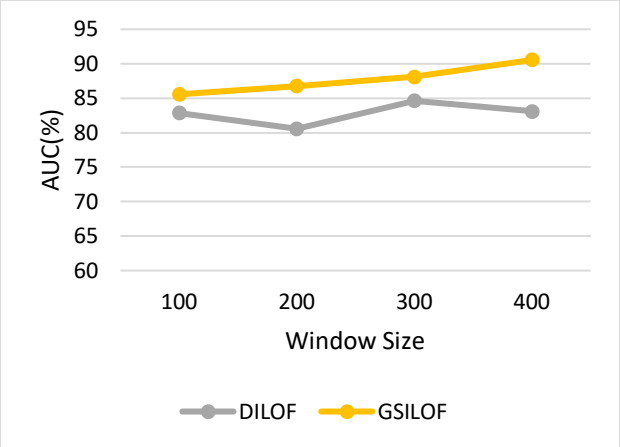

Fig.3. KDD Cup 99 smtp Dataset Outlier Detection Accuracy using AUC

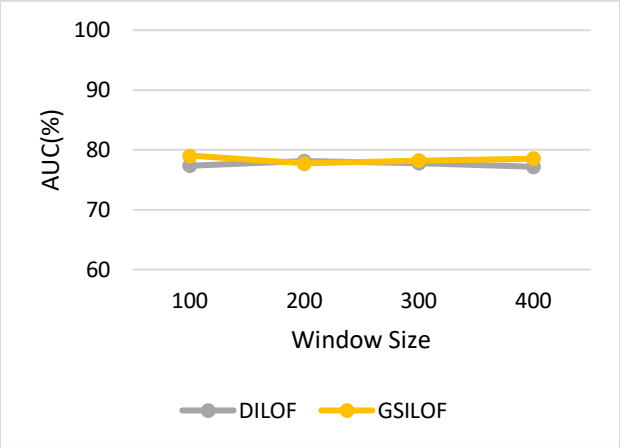

Fig.4. KDD Cup 99 http Dataset Outlier Detection Accuracy using AUC

4.2. Execution time

Fig 5, Fig 6 and Fig 7 shows the AUC for the three datasets UCI Vowel, KDD Cup 99 smtp and KDD Cup 99 http respectively. As shown the execution time for the GSILOF algorithm is always higher than the DILOF algorithm. This is because that GSILOF searches for many solutions and tries to find the best one while the DILOF works on one solution and start to update it to find the near solution that is better than the found one. Also, when we analyze the datasets as streams and divide the total execution time over the iterations of new points arrival, we found that the delay for the GSILOF algorithm is too small and doesn't affect the systems that will use it. Also, we can in the future try to make the GSILOF run parallel in finding the best solution using GPU or multi-core architecture. But to achieve this the window size must be big to overcome the time consumed in partitioning the data to be processed between cores or GPUs. We note that for the KDD Cup 99 http dataset when we increase the window size the execution time becomes almost the same for the two algorithms.

A big question was raised when we analyzed the execution time what if we increase the number of iterations for the DILOF algorithm? We tested a different number of iterations 20, 50, 100 and 200. Fig 8 and Fig 9 show the AUC and execution time for the different iterations for the UCI Vowel dataset. We notice that when we increase the number of iterations the execution time increases which is normal as it increases the solutions updates in the gradient descent. The execution time increases when the window size increase as noted before. For the AUC we notice that it isn't right that when we increase the number of iterations the AUC doesn't guarantee to be bigger. For example, at the window size 140 the ACU for 20 iterations is bigger than the AUC for 200 iterations. This indicates that in any case the AUC for the DILOF algorithm won't reach the AUC for the GSILOF.

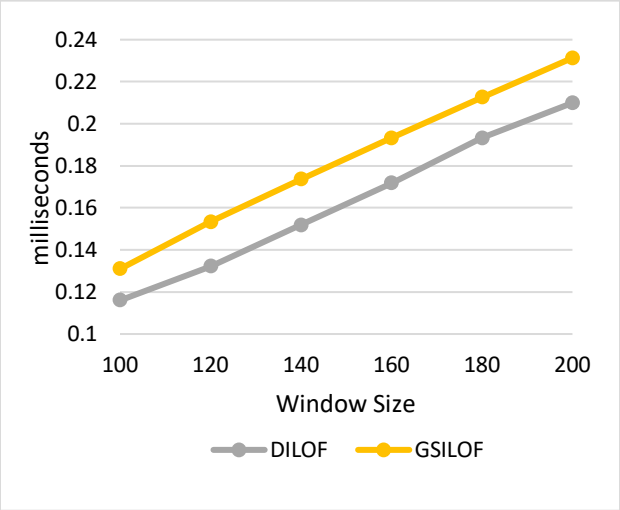

Fig.5. UCI Vowel Dataset Execution Time

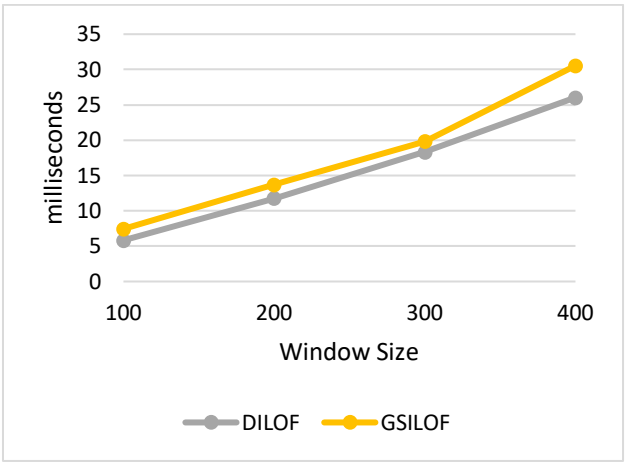

Fig.6. KDD Cup 99 smtp Dataset Execution Time

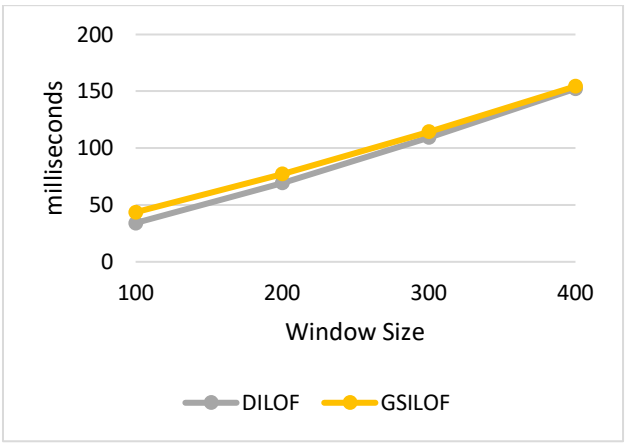

Fig.7. KDD Cup 99 http Execution Time

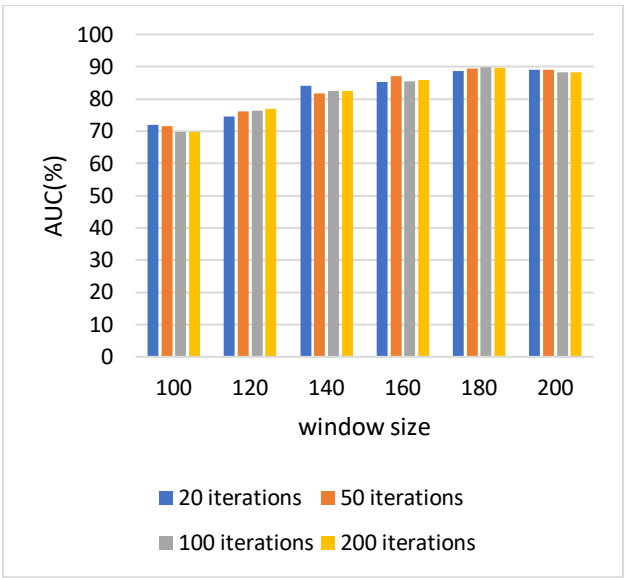

Fig.8. UCI Vowel Dataset Outlier Detection Accuracy using AUC For different iterations

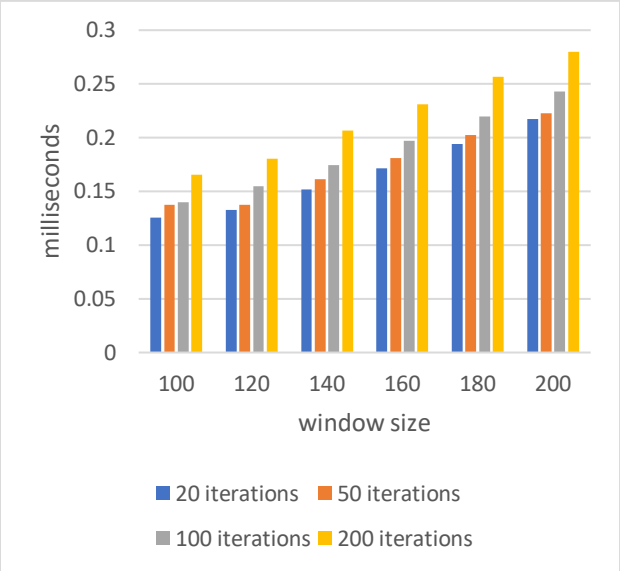

Fig.9. UCI Vowel Dataset Execution Time for different iterations

5. Conclusion

We propose an outlier detection algorithm called GSILOF for data streams. GSILOF consists of two phases the detection and summarization phases. The detection phase uses the ILOF to detect the outliers. While the summarization phase uses a Genetic algorithm (GA) that work parallel to find the best solution. GSILOF effectively and efficiently overcomes two fundamental limitations of LOF as it can detect the outliers in streaming data with limited memory environment. Our comprehensive experimental evaluations demonstrate that GSILOF significantly outperforms DILOF in terms of both detection accuracy and execution time. We are investigating further improvements and open research directions. In particular: The execution time of GSILOF is still high, especially as dimension of data increases. An efficient algorithm for clustering is to be developed in order to decrease the overall time of GSILOF. Also, algorithms for detecting Collective outliers are barely found in the literature and therefore this area has much to be researched.

References

[1] S. Sadik and L. Gruenwald, "Research issues in outlier detection for data streams," ACM SIGKDD Explorations Newsletter, vol. 15, pp. 33-40, 2014.
[2] M. Salehi, C. Leckie, J. C. Bezdek, T. Vaithianathan and X. Zhang, "Fast memory efficient local outlier detection in data streams," IEEE Transactions on Knowledge and Data Engineering, vol. 28, pp. 3246-3260, 2016.
[3] Y. Yan, L. Cao, C. Kulhman and E. Rundensteiner, "Distributed local outlier detection in big data," in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.
[4] M. Sakr, W. Atwa, and A. Keshk. "Research Article Distributed Anomaly Detection Over Big Data." *Research Journal of Applied Sciences, Engineering and Technology* 16, no. 2 (2019): 77-87.
[5] A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern recognition letters, vol. 31, pp. 651-666, 2010.
[6] D. Pokrajac, A. Lazarevic and L. J. Latecki, "Incremental local outlier detection for data streams," in Computational intelligence and data mining, 2007. CIDM 2007. IEEE symposium on, 2007.
[7] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey," ACM computing surveys (CSUR), vol. 41, p. 15, 2009.
[8] M. Sakr, W. Atwa, and A. Keshk. "Sub-Grid Partitioning Algorithm for Distributed Outlier Detection on Big Data." In *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, pp. 252-257. IEEE, 2018.
[9] M. Gupta, J. Gao, C. C. Aggarwal and J. Han, "Outlier detection for temporal data: A survey," IEEE Transactions on Knowledge and Data Engineering, vol. 26, pp. 2250-2267, 2014.
[10] W. Atwa, and L. Kan. "Clustering evolving data stream with affinity propagation algorithm." In International Conference on Database and Expert Systems Applications, pp. 446-453. Springer, Cham, 2014.
[11] E. M. Knox and R. T. Ng, "Algorithms for mining distancebased outliers in large datasets," in Proceedings of the international conference on very large data bases, 1998.
[12] F. Angiulli and F. Fassetti, "Detecting distance-based outliers in streams of data," in Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, 2007.
[13] D. Yang, E. A. Rundensteiner and M. O. Ward, "Neighbor-based pattern detection for windows over streaming data," in Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, 2009.
[14] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas and Y. Manolopoulos, "Continuous monitoring of distance-based outliers over data streams," in Data Engineering (ICDE), 2011 IEEE 27th International Conference on, 2011.

[15] M. M. Breunig, H.-P. Kriegel, R. T. Ng and J. Sander, "LOF: identifying density-based local outliers," in ACM sigmod record, 2000.
[16] K. Yamanishi, J.-I. Takeuchi, G. Williams and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," Data Mining and Knowledge Discovery, vol. 8, pp. 275-300, 2004.
[17] K. Yamanishi and J.-i. Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002.
[18] W. Atwa, and L. Kan. "Constraint-based clustering algorithm for multi-density data and arbitrary shapes." In *Industrial Conference on Data Mining*, pp. 78-92. Springer, Cham, 2017.
[19] F. Cao, M. Estert, W. Qian and A. Zhou, "Density-based clustering over an evolving data stream with noise," in Proceedings of the 2006 SIAM international conference on data mining, 2006.
[20] W. Atwa, and L. Kan. "Semi-supervised Clustering Method for Multi-density Data." In *International Conference on Database Systems for Advanced Applications*, pp. 313-319. Springer, Cham, 2015.
[21] Elahi, M.; Li, K.; Nisar, W.; Lv, X.; Wang, H. Efficient Clustering-Based Outlier Detection Algorithm for Dynamic Data Stream. In Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Shandong, China, 18–20 October 2008; Volume 5, pp. 298–304.
[22] W. Atwa, and L. Kan. "Affinity propagation-based clustering for data streams." *Applied Mathematics & Information Sciences* 9, no. 4, 2015.
[23] I. Assent, P. Kranen, C. Baldauf and T. Seidl, "Anyout: Anytime outlier detection on streaming data," in International Conference on Database Systems for Advanced Applications, 2012.
[24] M. Salehi, C. A. Leckie, M. Moshtaghi and T. Vaithianathan, "A relevance weighted ensemble model for anomaly detection in switching data streams," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2014.
[25] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," Machine learning, vol. 3, pp. 95-99, 1988.
[26] G. S. Na, D. Kim and H. Yu, "DILOF: Effective and Memory Efficient Local Outlier Detection in Data Streams," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018.
[27] B. Póczos, L. Xiong and J. Schneider, "Nonparametric divergence estimation with applications to machine learning on distributions," arXiv preprint arXiv:1202.3758, 2012.
[28] D. Dheeru and E. Karra Taniskidou, UCI Machine Learning Repository, 2017.
[29] C. C. Aggarwal and S. Sathe, "Theoretical foundations and algorithms for outlier ensembles," ACM SIGKDD Explorations Newsletter, vol. 17, pp. 24-47, 2015.
[30] J. Tang, Z. Chen, A. W.-C. Fu and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2002.
[31] S. Ramaswamy, R. Rastogi and K. Shim, "Efficient algorithms for mining outliers from large data sets," in ACM Sigmod Record, 2000.
[32] E. Lozano and E. Acufia, "Parallel algorithms for distance-based and density-based outliers," in Data Mining, Fifth IEEE International Conference on, 2005.
[33] W. Jin, A. K. H. Tung, J. Han and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2006.
[34] D. M. Hawkins, Identification of outliers, vol. 11, Springer, 1980.
[35] M. Bai, X. Wang, J. Xin and G. Wang, "An efficient algorithm for distributed density-based outlier detection on big data," Neurocomputing, vol. 181, pp. 19-28, 2016.
[36] C. C. Aggarwal and P. S. Yu, "Outlier detection with uncertain data," in Proceedings of the 2008 SIAM International Conference on Data Mining, 2008.
[37] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in ACM Sigmod Record, 2001.

Authors' Profiles

Arabi keshk received the B.Sc. in Electronic Engineering and M.Sc. in Computer Science and Engineering from Menoufia University, Faculty of Electronic Engineering in 1987 and 1995, respectively and received his Ph.D. in Electronic Engineering from Osaka University, Japan in 2001. His research interest includes software testing, software engineering, distributed system, database, data mining, and bioinformatics

Walid Atwa received the B.Sc. and M.Sc. in Computer Science from Menoufia University, Faculty of computers and information in 2006 and 2010, respectively, received his Ph.D. in Computer Science from Beijing Institute of Technology, China. His research interests are data mining and machine learning.

Mohamed Sakr received the B.Sc, M.Sc. and Ph.D. in Computer Science from Menoufia University, Faculty of computers and information. His research interests are data mining and machine learning.

How to cite this paper: Mohamed Sakr, Walid Atwa, Arabi Keshk, "Genetic-based Summarization for Local Outlier Detection in Data Stream", International Journal of Intelligent Systems and Applications(IJISA), Vol.13, No.1, pp.58-68, 2021. DOI: 10.5815/ijisa.2021.01.05