SLDPC: Towards Second Order Learning for Detecting Persistent Clusters in Data Streams

Ammar Al Abd Alazeez Sabah Jassim Hongbo Du
Department of Applied Computing
The University of Buckingham, Buckingham, MK18 1EG, UK
{1405097,sabah.jassim, hongbo.du}@buckingham.ac.uk

Abstract—**The main attention of research on data stream clustering algorithms so far has been focused on the adaptation of the algorithms for static datasets to the data streams and improvements of the existing adapted algorithms. Such algorithms fulfil the purpose of the first-order learning from data to clusters. This paper prompts a new question on second-order learning of cluster models from data streams and presents a learning algorithm that detects persistent clusters from consecutive clustering snapshots in data streams. In this work, we first collect a sequence of cluster snapshots as the output clusters at selected query points and then identify the persistent clusters within a given timeframe. The algorithm is evaluated on collections of synthetic datasets. The experimental results have demonstrated the effectiveness of the algorithm in detecting such persistent clusters.**

Keywords-Data Stream Clustering Algorithms, Persistent clusters, Clustering of Clusters, Second-order Learning

I. INTRODUCTION

Data stream clustering is defined as a grouping of data in light of frequently arriving new data chunks for gaining understanding about underlying group patterns that may change over time [1]. Depending on the approaches taken, either incremental learning or two-phase learning, existing algorithms for data stream clustering either present an up-to-current-time view of clusters [2] or generate a view of clusters at a user query point [3]. However, there is no keeping of a historic trail of the output cluster models over time. If such a historic trail is maintained, the persistence of certain clusters can be analysed through a second-order learning, i.e. learning persistent clusters by mining the clustering outputs collected over a sequence of time points. Stability of output clusters in static datasets has been well researched, and has a lot of advantages including predicting a correct number of clusters, detecting the lack of structure in the dataset, and assessing the quality of clustering algorithms [4][5]. However, identifying the persistent clusters in data streams has not been properly studied.

There are a lot of potential applications that can benefit from finding persistent clusters in the data streams. Tracking social media events such as birthdays, tracking objects such as cars and rockets from video footages, using CCTV cameras in identifying abnormal objects like unattended bags against a stable background, and monitoring patients in hospitals are only a few of many possible examples.

In this paper, we first define the problem of second-order learning of persistent clusters in data streams. In the problem context, we argue that data stream clustering can happen on two levels of processing. At the first level, also known as the online layer, the first-order learning of clusters is performed by using existing algorithms such as the prototype-based algorithm EINCKM presented in [6], and the clustering results are saved. At the second level, also known as the offline layer, second-order learning algorithms can be deployed to detect the patterns of persistent clusters from the saved clustering results produced at the first level. The paper then presents a simple second-order learning (SLDPC) algorithm for this purpose. The algorithm was evaluated on a selected collection of datasets using various measurements. Experimental results show that the proposed algorithm is capable of detecting correct persistent clusters. The modular structure of the proposed algorithm makes it easy to accommodate future improvements and parallelisations.

The rest of this paper is organised as follows. Section 2 explains the state of the art of the related work in the current literature. Section 3 formally defines the problem of persistent cluster detection, and illustrates it with examples. Section 4 describes the proposed SLDPC algorithm. Section 5 presents an evaluation of the algorithm performance through experiments using synthesised datasets. Section 6 concludes the work and outlines the possible future directions of this research.

II. RELATED WORK

A. Clustering of Clusters and Clustering Ensemble

In many pattern recognition problems, we are dealing with cluster analysis of existing clusters [7], such as multi-resolution granularity in hierarchical clustering [8]. Existing clusters are groups of pattern samples which, due to a priori knowledge, are known to belong to the same cluster. Kandt [9] presented a system named SEISMO for detecting seismic activity by several sensor networks. The characteristics of each detection and the time interval between these detections are used to group a subset of detections together (subclusters), which correspond to an event (cluster). Although clustering of clusters is a kind of second-order learning, it is more about grouping data in different granularity of abstraction than finding persistent clusters among existing ones.

Cluster ensemble has emerged as a prominent way of improving robustness, stability, and accuracy of clusters [10]. It is a process of merging multiple clustering models into a single consolidated clustering [11]. Fathzadeh and Mokhtari [12] presented an ensemble fuzzy C-Means (SEFCM) algorithm for data streams. The divide-and-conquer method comprised of three stages; 1) divide data streams into smaller blocks; 2) cluster every block using ensemble clustering (EFCM) algorithm; and 3) combine the concluding clusters using single linkage to find global clusters across the block clusters. Cluster ensemble is more about the consensus of different clustering models, and not really about discovering persistent and stable clusters although their inputs can be seen as clusters.

B. *Stable Clusters*

Stable clusters in static data mean when multiple datasets are sampled from the same distribution, the clustering algorithm is expected to behave in the same way and produce similar results [13]. In other words, to find stable clusters in static data, we need to further analyse the clustering results from each sample to identify stable ones. There is some degree of similarity but also a big difference between stable clusters in static data clustering and persistent clusters in data stream clustering [14][15]. In static clustering, the stability of clusters is defined over different versions of clustering, i.e. the stability is not defined over a time period. However, in data stream clustering algorithms, the data chunk could be evolved over time, i.e. some old clusters may disappear, and some new clusters may emerge (concept drift principle [16]). Our objective is therefore different: we have already a number of versions of clusterings depend on query points and the aim is to discover the clusters that stay relatively fixed.

C. *Clustering Learning Approaches*

Learning approach in the data streams mining is a wide area of research. Generally, learning approaches could be divided into two categories [17]: instance-incremental (or incremental learning) methods that learn from each example as it arrives and batch-incremental (or two-phase leaning) methods that gather examples in batches to train models. He *et al.* [18] proposed a general adaptive incremental learning framework that is capable of learning from continuous raw data, accumulating experience over time, and using such knowledge to improve future learning and prediction performance for classification purpose. However, this work is focusing on improving the output model to adapt to new incoming data chunks whereas we are aiming to identify persistent clusters through consecutive clustering snapshots.

There is a big difference between two-phase leaning [3] and second-order learning for data stream clustering. The two-phase learning algorithms try to discover final clusters from many prototype micro-clusters. It is still a first-order learning from data to clusters. The second-order learning algorithms, on the other hand, try to find persistent clusters that exist through a sequence of consecutive clustering results. In other words, it takes as inputs a sequence of clustering results and identifies as outputs the clusters that persist over the whole time period.

D. *Data Stream Clustering Algorithm EINCKM*

EINCKM is an incremental prototype-based algorithm for clustering data streams [6]. It consists of a generic modular framework that comprises three main steps *Build Clusters*, *Merge*, and *Prune*. *Build Clusters* applies the K-Means method to identify the clusters from input data chunks, *Merge* may combine the newly discovered clusters with some existing ones, and *Prune* identifies outlier objects and removes out of date data points. The algorithm applies a simple heuristic-based strategy to estimate the number of clusters, a radius-based scheme to combine overlapped clusters, and a variance-based technique to detect the outliers. However, this algorithm is designed to perform first-order clusterings. In other words, it gives the up-to-current-time snapshot of clustering results.

III. Problem Description

Informally, persistent clusters are those that do not *change much* and persist over a period throughout a series of clustering results with respect to the coming data chunks. More precisely, persistent clusters can be defined in the following way. Let $C_i = \left\{C_1^{(i)}, C_2^{(i)}, \dots, C_n^{(i)}\right\}$ represent a cluster clustering result at a time point *i* known as a *snapshot* where $C_j^{(i)}$ is a cluster. Let $C = [C_1, C_2, \dots, C_m]$ represent a sequence of clustering snapshots. Given a time frame $[\mathrm{t}_1, \mathrm{t}_2]$, where $t_1 \geq 1$, $t_2 \leq m$ and $t_1 < t_2$, and user-defined thresholds on:

- Centroid change margin CCC, representing the maximum distance allowed for the centroids of clusters $C_j^{(i)}$ and $C_j^{(i+1)}$.
- Size change margin CSC, representing the maximum amount of change in cluster sizes between clusters $C_j^{(i)}$ and $C_j^{(i+1)}$.
- Variance (radius) change margin CRC, representing the maximum amount of change in cluster radius between $C_j^{(i)}$ and $C_j^{(i+1)}$.

Then the persistent clusters are those $C_i^{(j)}$s which exist within the given time frame $[t_1, t_2]$ and the cross-snapshot differences (i.e. changes) of their centroids, sizes, and variances are less than or equal to CCC, CSC, and CRC respectively. Mining such *persistent clusters* is an automatic process of discovering all persistent clusters as defined.

Fig. 1 shows an example of persistent cluster discovery. In the first cluster snapshot (Fig. 1(a)), there are three initial clusters. In the second snapshot (Fig. 1(b)), the three clusters from the snapshot one persist with little changes, but the snapshot shows the creation of a new cluster (cluster 4). In the third snapshot (Fig. 1(c)), the three persistent clusters still remain with little changes to the centroids, sizes and variances. However, cluster 4 disappears, indicating that it is only a temporary cluster. At the same time, a new cluster (i.e. the new cluster 4) emerges. In the final snapshot (Fig. 1(d)), the three persistent clusters still remain in place, but cluster 4 in the previous snapshot disappears because it is a temporary cluster. So, finally, three persistent clusters are obtained as the output of the discovery process. Table 1 describes the snapshots summary and indicates the persistent clusters (*Persistency-Tag* with the highest value in the last snapshot). Note that the *N* represents the number of data points in each cluster, *μ* is the vector value of the centroids, *R* is the radius of each cluster, and *Persistency-Tag* is the counter of repeated clusters.

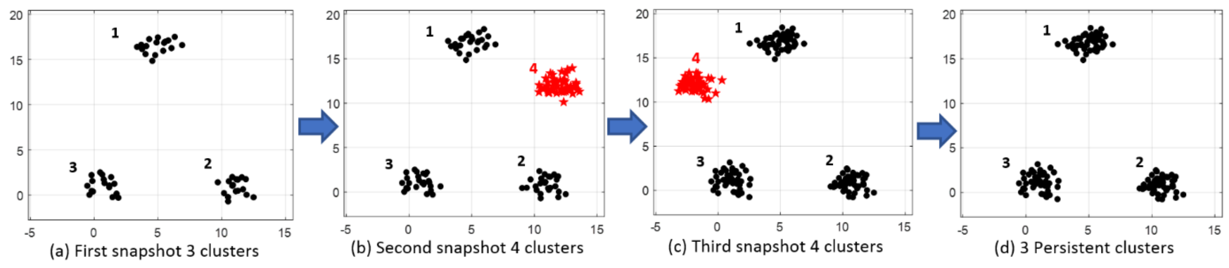


Fig. 1. Example of persistent clusters

IV. The Proposed SLDPC Algorithm

The general contextual framework of the basic proposed SLDPC algorithm is depicted in Fig. 2. Against the snapshots produced by online EINCKM algorithm, the main stages of the SLDPC algorithm are described as follows:

- Receive consecutive clustering snapshots and user parameters as inputs.
- Define the persistent clusters for each consecutive pair of clustering snapshots using the merging strategy.
- Find the final output persistent clusters through the consecutive clustering snapshots.

Algorithm 1 presents the pseudo-code description of the basic SLDPC algorithm. The inputs are mainly a sequence of consecutive snapshots of existing clusters summary *C*. Each cluster summary is a tuple (N, LS, LSS, μ, R), where N is the number of data points, *LS* is the linear sum of the data points and *LSS* is the sum of squared data points. The inputs also include the user definition thresholds which include CCC, CSC, and CRC (cf. Sec. 3). The output is *PS* persistent clusters.

For the user convenience, the algorithm takes the first threshold parameter CCC in terms of how many standard deviation from the mean of a cluster. Therefore, the user will decide a real number (e.g. 1, 1.5, 2, etc.), and the algorithm multiplies the number with the standard deviation of cluster $C_j^{(i)}$ to determine the absolute distance threshold between the two centroids. For the same purpose user convenience, the size change margin threshold parameter CSC is normally represented as a percentage of change in relation to the size of cluster $C_j^{(i)}$. Similarly, the radius change margin threshold parameter CRC is also represented as a percentage of change in relation to the radius of cluster $C_j^{(i)}$.

TABLE 1. Snapshot description

Snap-shot Number	No. of Clus-ters	Cluster Description				
		Seq.	N	μ	R	Persis-tency Tag
1	3	1	17	(5,16)	2	1
		2	15	(12,1)	2.5	1
		3	16	(1,2)	3	1
2	4	1	25	(5.2,17)	3	2
		2	24	(13,2)	3.5	2
		3	26	(2,3)	4	2
		4	40	(13,14)	2	1
3	4	1	42	(5.5,18)	4	3
		2	37	(12,2.5)	4.5	3
		3	40	(3,4)	5	3
		4	35	(-2,12)	3	1

Persistent Clusters

V. Evaluation and Experimental Results

The SLDPC algorithm is meant to work with any data stream clustering solutions and for any scenarios of any dataset with any number of dimensions as long as the output clusters can be represented in a summary form as expected by the algorithm. However, in order to ease the verification of the results, we decided to use EINCKM algorithm for clustering data streams because the algorithm describes the clustering

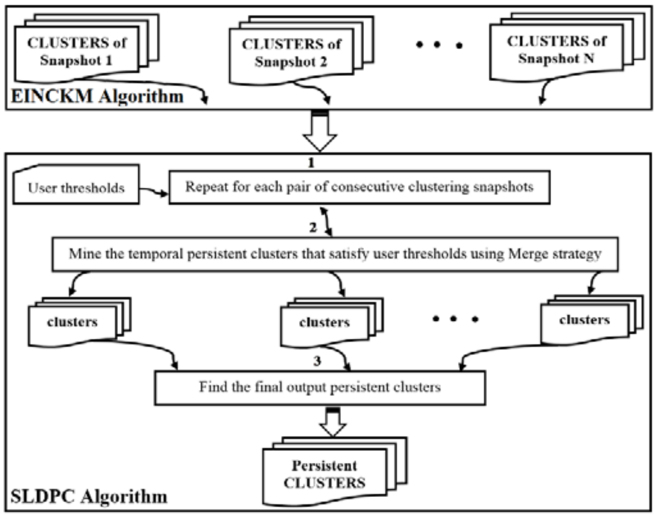

CLUSTERS of Snapshot 1
CLUSTERS of Snapshot 2
• • •
CLUSTERS of Snapshot N
EINCKM Algorithm
User thresholds
1
Repeat for each pair of consecutive clustering snapshots
2
Mine the temporal persistent clusters that satisfy user thresholds using Merge strategy
clusters
clusters
• • •
clusters
3
Find the final output persistent clusters
Persistent CLUSTERS
SLDPC Algorithm

Fig. 2. Outline of the SLDPC algorithm

Algorithm 1 Second Order Learning

```
Inputs:
 - Consecutive accumulated cluster snapshots summary $(N, LS, LSS, \mu, R)$.
 - User parameters
   $[t_1, t_2]$: Time frame
   $CCC$: Threshold of moving the centroids // by default $CCC = 2$
   $CSC$: Threshold of cluster size change // by default $CSC = \pm 10\%$
   $CRC$: Threshold of cluster radius change // by default $CRC = \pm 10\%$
Outputs:
  $PS$: Persistent Clusters;
Algorithm Steps:
  1.Repeat for each pair of consecutive snapshots
    $If\ dist(\mu_i, \mu_j) \le (CCC * STD)$ && // dist is a distance function such as Euclidean
      $|size(C(i)) - size(C(j))| / size(C(i)) \le CSC$ && // calculate the percentage differences of sizes
      $|R(i) - R(j)| / R(i) \le CRC\ then$ // calculate the percentage differences of radii
        $Persistency\_Tag(j) = Persistency\_Tag(j) + 1$
    $else$
      $Persistency\_Tag(j) = 1$
    $end$
  2. Repeat for each cluster
    $If\ Persistency\_Tag(i) = (t_2 - t_1), add\ C(i)\ into\ PS;$
    $end$
```

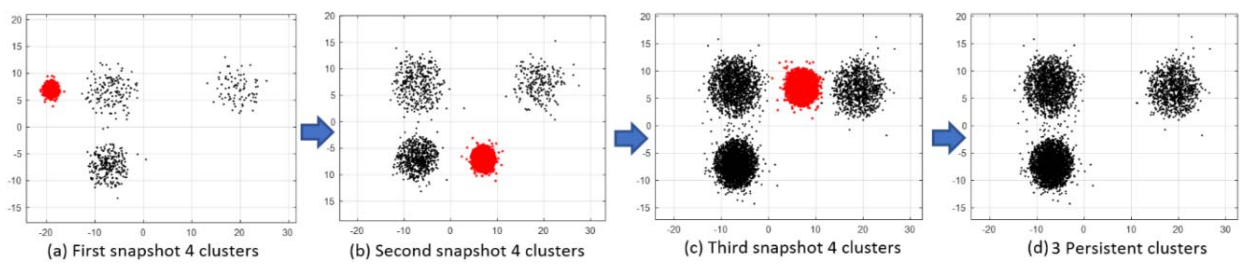


Fig. 5. DS1-Scenario_1

output in terms of cluster summaries. All the experiments have done with the three synthetic datasets generated from multi-variant Gaussian distribution (DS1, DS2, and DS3) of 100,000, 500,000, and 1,000,000 data points respectively of two dimensions. The DS1 contains six clusters; DS2 contains fifteen clusters, and DS3 contains thirty clusters. Clusters in each of the three datasets have different sizes. Each cluster is generated randomly by following a normal distribution of a different mean and variance. Fig. 3 shows the scatterplots of DS1. The details of the normal distributions used for generating the datasets are given in Appendix 1. Table 2 summarises four scenarios of different numbers of persistent and non-persistent clusters (known as *temporary clusters*). In the first three scenarios, there are both temporary and persistent clusters. Fig. 4 shows that for DS1 there are three persistent and three temporary clusters (first scenario), and Fig. 5 illustrates the evolution of the clusters through a sequence of four snapshots. In the fourth scenario, however, all clusters are temporary; an extreme case of concept drift where nothing is persistent.

To evaluate correctness, we used three commonly used evaluators: purity, entropy, and the sum of squared errors (SSE). Purity was used in [19], entropy in [20], and SSE in [3]. Purity refers to the proportion of the data points belonging to a known cluster that are assigned as members of a cluster by the algorithm. The higher the proportion of purity (between [0, 1]) is, the more certain that the algorithm has found the original clusters and the better the algorithm is [21]. Entropy reflects the number of the data points from different known clusters in the original dataset that are assigned to a cluster by the algorithm. The value of this measure is between $[0, Log_2N]$

TABLE 2. Suggested scenarios

		Scenario 1		Scenario 2		Scenario 3		Scenario 4	
Dataset	#clusters	#Persist	#Temporary	#Persist	#Temporary	#Persist	#Temporary	#Persist	#Temporary
DS1	6	3	3	2	4	1	5	0	6
DS2	15	6	9	3	12	1	14	0	15
DS3	30	10	20	5	25	1	29	0	30

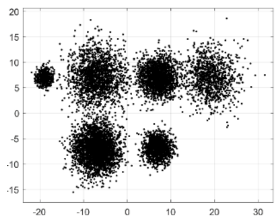


Fig. 3. Clusters in DS1

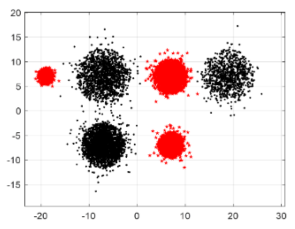


Fig. 4. Temporary/Persistent clusters in DS1

where N is the number of known clusters involved. The smaller value of the entropy is, the fewer members of the known clusters are mixed in the clusters discovered by the algorithm, and the better the clustering algorithm is [22]. SSE is a commonly used cluster quality measure. It evaluates the compactness of the resulting clusters. Low scores of SSE indicates better clustering results as the clusters contain less internal variations [21]. The efficiency of an algorithm was measured by the amount of time in seconds taken for the algorithm in completing the clustering task.

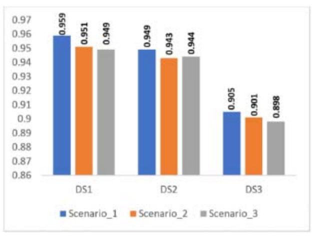
MATLAB 2017b was used to implement the SLDPC algorithm and the experiment framework. For the first, second, and third scenarios as mentioned, we split a given dataset into two parts: the persistent clusters and the temporary clusters. We selected data chunks randomly from the persistent clusters and snapshot-wise data points from the temporary clusters. The idea behind the random selection of the data points is to investigate the behaviour of the algorithm when there is no control on the sequence of data points, i.e. we did not select specific data points from particular groups in the original datasets. In order to minimise the effect of the random choice of data points, the experiments were repeated 100 times, and the average was calculated.

All the experiments were run on a machine equipped with 2.30 GHz 4 cores Intel(R) Core(TM) i5-4590 CPU and 16 GB memory. The operating system was Windows7.

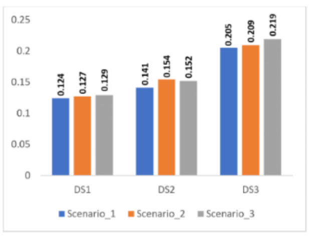
A. Experimental Results

Fig. 6 illustrate the performance evaluation of SLDPC algorithm. As shown in Fig. 6, differences between the scenarios across the synthetic datasets are only marginal, the algorithm performs consistently across the synthesised datasets in all scenarios.

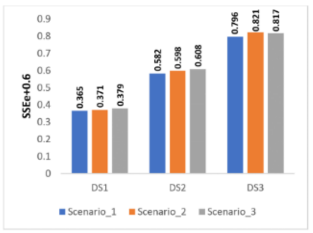
Fig. 6(a) shows that the level of purity is high across all scenarios when comparing the persistent output clusters from the SLDPC algorithm against the known persistent clusters in the ground truth (the synthesised datasets with known clusters). This is caused by the stringent merge strategy deployed in both EINCKM and SLDPC algorithms and the exclusion of some data points as outliers by using the filtering technique in EINCKM. With a small number of persistent clusters, e.g. in scenario 3 the level of purity is lower than those for the scenarios with more persistent clusters. Both entropy measurements and SSE measurement as shown in Fig. 6(b) and (c) are relatively low deu to the effective pruning strategy of the



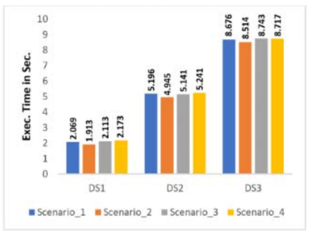
(a) The purity measurement



(b) The entropy measurement



(c) The SSE measurement



(d) The efficiency measurement

Fig. 6. Performance measurements

EINCKM algorithm. Removing the outliers prevent including them into different persistent clusters. The execution time for the SLDPC algorithm to find the final persistent clusters is also very short. The empirical results show a linear growth of time in relation to the dataset size (see Fig. 6(d)).

B. *Discussion*

The most noticeable nature of the SLDPC algorithm is its simplicity and efficiency in discovering persistent clusters. The main principle behind the algorithm is to maintain a vote to each cluster. Only the clusters with sufficient votes remain as persistent clusters. The constraint of the basic algorithm is that it assumes the input clusters are represented as cluster summaries which tend to be applied only to spherical shaped clusters. Therefore, the algorithm works well with prototype-based and model-based algorithms. Currently, the algorithm might not apply to cluster inputs that are represented in other forms of structures (such as data point based representation of clusters by density-based algorithms).

Regarding parameters representing thresholds, we set the default values $CCC = 2$, CRC and CSC to $\pm 10\%$. Deciding CCC parameter is not trivial. There are number of ways to define it. For instance, we could use absolute distance between two centroids, but this number is very hard for the user to find. By refering to the normal distribution and statistic theory regarding the significant difference we decid to rely on the number of STDs to determine this particular threshold. Setting this threshold is challenging, therefore, need further investigation. We set the default value of cluster size change CSC and the cluster radius change CRC depending on the heuristics. However, such default values may not apply to a certain dataset, and hence we leave the user to define the appropriate thresholds for the parameters.

We understand the importance of the threshold values to the final outputs of persistent clusters in the problem definition. To further this consideration, we can introduce two more threshold parameters. The first additional parameter is the number of snapshots t within the time frame $[t_1, t_2]$. This parameter allows the discovery of persistent clusters not in all the snapshots in the snapshot sequence C, but rather among t snapshots of the sequence. Another addtional threhsold parameter we can introduce is the persistency rate PR that specifies the rate of persistency across the snapshots; the persistent clusters does not have to appear in every snapshot, but PR percent of the snapshots. Both parameters are meant to increase the flexibility of the algorithm in producing the persistent clusters that are variants from the standard definition.

VI. Conclusion and Future Works

This paper prompted a problem of second-order learning for persistent clusters in data streams, and presented the SLDPC algorithm for detecting such persistent clusters by analysing a sequence of snapshots of clustering results. The key ideas of the algorithm is to assign a vote to clusters that do not change much, and then collect those clusters. The evaluation results have shown that the algorithm produces correct and good quality clusters with low time complexity. The algorithm emphasises on simplicity and adaptivity for future improvement.

Our future work will focus on enhancing the algorithm. Firstly, we will work towards tailoring the algorithm to suit other cluster input representations. Secondly, we will investigate introducing degrees of fuzziness in user-defined thresholds and reducing the needs for user-defined thresholds if possible. Finally, we will further investigate discovering the patterns of periodic changes in cluster models besides persistency. In discovering periodic changes and persistent hidden group patterns can have a wide range of applications such as climate changes.

Acknowledgement

The first author wishes to thank the University of Mosul and Government of Iraq/Ministry of Higher Education and Research (MOHESR) for funding him to conduct this research at the University of Buckingham.

References

[1] Yogita and D. Toshniwal, "Clustering Techniques for Streaming Data – A Survey," *3rd IEEE International Advance Computing Conference (IACC)*, pp. 951–956, 2012.

[2] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams," *IEEE FOCS Conference*, pp. 359–366, 2000.

[3] C. Aggarwal, J. Han, J. Wang, and P. Yu, "A Framework for Clustering Evolving Data Streams," *Proceedings of the 29th VLDB Conference, Germany*, pp. 1–12, 2003.

[4] U. Luxburg, "Clustering Stability: An Overview," *arXiv:1007.1075v1, now-the essence of knowldge*, pp. 1–41, 2010.

[5] S. Saha and S. Bandyopadhyay, "A New Measure of Stability of Clustering Solutions: Application to Data Partitioning," *2009 International Conference on Adaptive and Intelligent Systems*, 2009.

[6] A. Al Abd Alazeez, S. Jassim, and H. Du, "EINCKM: An Enhanced Prototype-based Method for Clustering Evolving Data Streams in Big Data," *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, no. Icpram, pp. 173–183, 2017.

[7] T. Gao, A. Li, and F. Meng, "Research on Data Stream Clustering Based on FCM Algorithm," *Scince Direct, Elsvier, Procedia Computer Science*, vol. 122, pp. 595–602, 2017.

[8] J. de Andrade Silva, E. R. Hruschka, and J. Gama, "An evolutionary algorithm for clustering data streams with a variable number of clusters,"

Expert Systems with Applications, vol. 67, pp. 228–238, 2017.

[9] K. Kandt and W. H. Drive, “Second-order conceptual clustering of temporal events,” *1990 IEEE International Conference on Systems, Man, and Cybernetics Conference Proceedings*, vol. 000, pp. 599–604, 1990.

[10] R. Ghaemi, N. Sulaiman, H. Ibrahim, and N. Mustapha, “A Survey : Clustering Ensembles Techniques,” *Engineering and Technology*, vol. 38, no. February, pp. 636–645, 2009.

[11] T. M. Alqurashi, “Clustering Ensemble Method,” Thesis, University of East Angglia, School of Computer Science, 2017.

[12] P. Zhang, X. Zhu, J. Tan, and L. Guo, “Classifier and cluster ensembles for mining concept drifting data streams,” *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 1175–1180, 2010.

[13] N. Sauvageot *et al.*, “Stability-based validation of dietary patterns obtained by cluster analysis,” *Nutrition Journal*, vol. 16, no. 1, p. 4, 2017.

[14] A. Rinaldo and R. Nugent, “Stability of Density-Based Clustering,” *Journal of Machine Learning Research*, vol. 13, pp. 905–948, 2012.

[15] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann, “Stability-Based Validation of Clustering Solutions,” *Neural Computation*, vol. 16, no. 6, pp. 1299–1323, 2004.

[16] P. Chaovalit, “Clustering Transient Data Streams By Example And By Variable,” Thesis, University of Maryland, Department of Information Systems, 2009.

[17] D. Puschmann, P. Barnaghi, and R. Tafazolli, “Adaptive Clustering for Dynamic IoT Data Streams,” *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 64–74, 2017.

[18] H. He, S. Chen, K. Li, and X. Xu, “Incremental learning from stream data,” *Neural Networks, IEEE Transactions …*, vol. 22, no. 12, pp. 1901–1914, 2011.

[19] F. Cao, M. Ester, W. Qian, and A. Zhou, “Density-based clustering over an evolving data stream with noise,” *Proceedings of the Sixth SIAM International Conference on Data Mining*, vol. 2006, pp. 328–339, 2006.

[20] Y. Zhao and G. Karypis, “Technical Report Criterion Functions for Document Clustering: Experiments and Analysis,” *University of Minnesota, Department of Computer Science / Army HPC Research Center/ Technical Report*, pp. 1–30, 2001.

[21] J. Silva, E. Faria, R. Barros, E. Hruschka, and A. Carvalho, “Data Stream Clustering : A Survey,” *ACM Computing Surveys (CSUR)*, pp. 1–37, 2013.

[22] H. L. Nguyen, Y. K. Woon, and W. K. Ng, “A survey on data stream clustering and classification,” *Knowledge and Information Systems, Springer*, pp. 535–569, 2015.

APPENDIX I

The following tables show the details and specify the distribution of each of the three synthesized datasets.

TABLE 1: Parameters details of DS1.

C	**Mean**		**STD**	**Size**
	X	**Y**		
C1	5	7	2	20000
C2	16	7	3	11000
C3	5	-7	1.5	15000
C4	-5	7	3	18000
C5	-16	7	1	5000
C6	-5	-7	2.5	31000

TABLE 2: Parameters details of DS2.

C	**Mean**		**STD**	**Size**
	X	**Y**		
C1	20	20	2	20000
C2	5	40	1.5	26000
C3	30	19	1.1	34000
C4	12	40	2.5	9000
C5	25	30	2.4	31000
C6	-15	37	3.5	66000
C7	-25	43	1.8	29000
C8	1	67	1.2	15000
C9	15	55	2.9	17000
C10	-2	54	2.3	40000
C11	-20	55	3.9	13000
C12	15	75	3.8	60000
C13	20	65	0.9	50000
C14	-7	80	4.1	20000
C15	-25	75	2.7	70000

TABLE 3: Parameters details of DS3.

C	**Mean**		**STD**	**Size**
	X	**Y**		
C1	7	7	2.5	80000
C2	30	7	3.5	11000
C3	-11	7	1.9	350000
C4	-35	7	3.9	29000
C5	45	7	1.1	12900
C6	7	30	3.4	3000
C7	32	30	2.2	50000
C8	-15	30	3.1	7000
C9	-30	30	1.2	6000
C10	50	30	3.3	17000
C11	-15	60	1.2	9000
C12	45	60	5.9	20000
C13	10	60	6.4	10000
C14	-50	60	7.9	1000
C15	75	60	1.7	43000
C16	1	105	7.5	40000
C17	25	105	1.3	16500
C18	-35	105	4.4	35000
C19	-60	105	2.4	4000
C20	60	105	7.7	19000
C21	5	150	6.4	50000
C22	30	150	0.9	12000
C23	-30	150	4.4	80000
C24	-60	150	2.4	5000
C25	60	150	5.5	8000
C26	7	190	3.6	25000
C27	25	190	0.8	7100
C28	-20	190	4.2	2500
C29	-50	190	2.8	3500
C30	50	190	5.6	43500