Effective Density-Based Concept Drift Detection for Evolving Data Streams

Zelin Cui[1], Hui Tian[2], and Hong Shen[3,4(✉)]

[1] Institute of Information Security Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Information and Communication Technology, Griffith University, Brisbane, Australia
[3] Faculty of Applied Sciences, Macao Polytechnic University, Macao, China
hong.shen@adelaide.edu.au
[4] School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou, China

Abstract. Concept drift is a common phenomenon appearing in evolving data streams of a wide range of applications including credit card fraud protection, weather forecast, network monitoring, etc. For online data streams it is difficult to determine a proper size of the sliding window for detection of concept drift, making the existing dataset-distance based algorithms not effective in application. In this paper, we propose a novel framework of Density-based Concept Drift Detection (DCDD) for detecting concept drifts in data streams using density-based clustering on a variable-size sliding window through dynamically adjusting the size of the sliding window. Our DCDD uses XGBoost (eXtreme Gradient Boosting) to predict the amount of data in the same concept and adjusts the size of the sliding window dynamically based on the collected information about concept drifting. To detect concept drift between two datasets, DCDD calculates the distance between the datasets using a new detection formula that considers the attribute of time as the weight for old data and calculates the distance between the data in the current sliding window and all data in the current concept rather than between two adjacent windows as used in the exiting work DCDA [2]. This yields an observable improvement on the detection accuracy and a significant improvement on the detection efficiency. Experimental results have shown that our framework detects the concept drift more accurately and efficiently than the existing work.

Keywords: Data Mining · Machine Learning · Data-Stream Clustering · Concept-Drift Detection

1 Introduction

Data stream clustering has been successfully applied for detection of concept drift [11] which is an important problem arising in a wide range of applications including credit card fraud protection, the weather forecast, network intrusion detection, etc.. The concept of interest may depend on some hidden context, not given explicitly in the form of predictive features [3]. In other words, the concepts drift with time from what we analyze from current data. For example, the buying preferences of customers may change with

J. S. Park et al. (Eds.): PDCAT 2023, LNEE 1112, pp. 190–201, 2024.
https://doi.org/10.1007/978-981-99-8211-0_18

time, depending on the day of the week, availability of alternatives, discount rate, etc. [3]. If we do not detect the concept drift in time, we may end up with taking the wrong concept, which not only decreases the quality of clusters but also can lead to unexpected clustering results. Hence, dealing with concept drift is crucial in many applications.

Typically, concept drift detection is done by calculating the distance of two datasets between adjacent sliding windows of fixed size using the rough-set theory. The detection effectiveness depends heavily on the size of sliding windows. Both too small and too large windows are undesirable, because the former may be unable to capture a single concept and the latter may contain multiple concepts. However, because of the fast evolution property of online data streams, it is difficult to determine a proper size of the sliding window for effective detection of concept drift. This makes the existing algorithms based on this approach difficult to be used in real application.

In this paper, we present a new framework for concept drift detection, named density-based concept drift detection (DCDD). It is based on density-based clustering [15] with a variable-size sliding window which is formed by dynamically adjusting the size of the sliding window based on the prediction model trained by XGBoost(eXtreme Gradient Boosting) [4] to adapt to the changes of the data stream. We extend the formula used in the existing drift detection algorithm DCDA [2] by incorporating the time attribute in calculating the distance of two datasets to find the concept drift.

2 Related Work

Concept drift, which was first introduced by Schlimmer and Granger in [11], refers to the phenomenon that data points are subject to different distribution models in different time periods. There is a rich literature on concept drift detection in which many algorithms are based on classification relying on error rate of classification prediction, such as [17]. While the classification-based algorithms are simple and efficient, they need data with class labels as training data set which is difficult to obtain for time-evolving data streams.

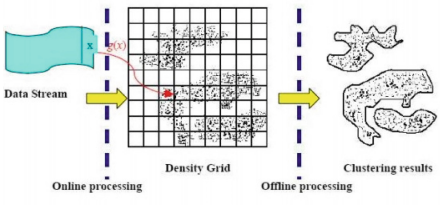
To address this issue, concept drift detection based on clustering was proposed. Stream-detect [10], detects concept drift through analyzing the clustering results to identify changes in data streams by measuring deviation of clustering result online. Chen et al. [3] proposed a framework to perform clustering on the categorical time-evolving data by comparing the distribution of the clusters and the outliers from the last and the current sliding windows. Because the window size is fixed, it does not adapt to the change of data streams. [2], proposed a concept drift detection algorithm (DCDA) based on the rough-set theory [13] and sliding window technique to improve the efficiency, which calculates the distance between the last and the current windows to detect concept drift before starting the clustering process.

To find arbitrarily shaped clusters and handle noises efficiently, numerous density-based clustering algorithms have been developed, such as D-Stream I [5], DD-Stream [12], D-Stream II [18], GDC-Stream [7] and PKS-Stream [14], and Relative Density-Based [9], These algorithms process the raw data only once and do not need to set the number of clusters. Recently, concept drift detection has been applied to high-speed streams [16] and also finds application for multi-label classification of IoT data steams [19]. They suffer from the difficulty of effectively adapting to dynamic changes of online data streams.

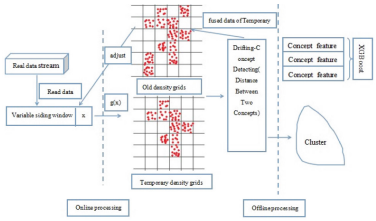
3 Preliminaries

3.1 Density-Based Clustering

From the Fig. 1(a), we can see the framework of density-based clustering. It uses a two-phase scheme [1], which consists of an online component and an offline component. In the online component, density-based clustering maps each input data record into a corresponding grid and updates the density of grid which is the sum of all data points in the gird. In the offline component, it uses an incomplete partitioning strategy to cluster the density grids. We take advantage of this process and design our framework for concept drift detection that is shown in Fig. 1(b).



(a) Density-based clustering [5]



(b) Density-based concept drift detection

Fig. 1. Density-based Clustering and Concept Drift Detection

3.2 Definitions

In this section, we introduce the relevant concepts used in our framework. We assume that the input data stream has d dimensions and define the data space $S = S_1 \times S_2 \times \cdots \times S_d$, where S_i is the definition space for the i^{th} dimension.

Definition 3.2.1 (Grid Cell). For data space $S = S_1 \times S_2 \times \cdots \times S_d$, each $S_i (1 \le i \le d)$ is divided into p_i parts evenly, we define the intersection of $S_i (1 \le i \le d)$ as the grid cell g. That is, $g_{j_1 j_2 \ldots j_d} = S_{1,j_1} \cap S_{2,j_2} \cap \ldots \bigcap S_{d,j_d}$, $1 \le j_t \le p_t$, $1 \le t \le d$.

When a data record $X = (x_1, x_2, \ldots, x_d)$ arrives, it can be mapped to a density grid $g(x)$ as follows: $g(x) = (j_1, j_2, \cdots, j_d)$, where $x_i \in S_{ij_i}$, $1 \le i \le d$, $1 \le j_t \le pt$, $1 \le t \le d$;

The grid density of a grid cell g at t is:

$$D(g, t) = \sum_{x \in g} D(x, t)$$

At time t, the average density of the non-empty grid cells is $Den_{\text{avg}} = \frac{\sum_{i=1}^{K} D(g,t)}{K}$, where the $D(g, t)$ is the grid density of the non-empty grid cell g and K is the number of non-empty grid cells.

Definition 3.2.2 (Dense Grid and Sparse Grid). Grid cell g is a dense grid if $D(g, t) \geq \alpha Den_{avg}$, and a sparse grid if $D(g, t) < \alpha Den_{avg}$, where α is a parameter controlling the threshold.

In paper [5], the boundary between dense grid and sparse grid is defined as a fixed value that is not flexible and hard to set. In comparison, our above definition on the boundary can effectively adapt to the unknown data stream.

Definition 3.2.3 (Grid Characteristic Vector). The characteristic vector of grid cell g is defined as (*D, label, status, t*), where D is the last updated density of g, *label* is the class of g, t is the time that the last data came in, and *status* (either SPORADIC or DENSE) is used to mark the status of g.

In our framework, in order to get the distance between two data sets, we use two density grids: temporary density grids and old density girds. The temporary density grids store the grid characteristic vectors of the data in the current sliding window. The old density girds store the grid characteristic vectors of all data in the same concept.

In order to predict the next concept and the amount of data stream in the next concept, we need to collect and store the feature vector of the concept that we call concept-feature when the concept drift is detected. We use the XGBoost (eXtreme Gradient Boosting) [4] to train the concept-features and the trained model to predict. How to extract the attributes of the concept-feature will be explained in Sect. 4.4.

3.3 Concept Drifting Detection

Concept drift detection algorithm for data streams (DCDA) was proposed in [2] that works by calculating the distance between the current sliding window and the last sliding window based on the rough membership function and the sliding-window technique [1, 6, 8, 10].

The distance between two datasets is measured as follows:

For the current subset S^{T_i} and the last subset S^{T_j}, the distance between S^{T_i} and S^{T_j} is defined as

$$d_A\left(S^{T_i}, S^{T_j}\right) = \frac{1}{|A|}\sum_{a\in A} d_{\{a\}}\left(S^{T_i}, S^{T_j}\right) = \frac{\sum_{a\in A}\sum_{x\in S^{[T_i,T_J]}}\left|\mu_{S^{T_i}}^{\{a\}}(x) - \mu_{S^{T_j}}^{\{a\}}(x)\right|}{\left|S^{[T_i,T_j]}\right||A|},$$

where A is a non-empty set of attributes, and $\mu_{S^{T_i}}^{\{a\}}(x)$ is a rough membership function.

If the distance between two datasets is larger than the threshold, the data in the current sliding window will perform re-clustering to capture the emerging new concept. In contrast, if the concept is steady, each object of the current window will be allocated into the corresponding cluster according to distance comparison [2].

4 The Proposed Algorithm

4.1 Overall Framework

Our density-based concept drift detection (DCDD) framework follows the density-based clustering framework [15] composed of an online component and an offline component as illustrated in Fig. 1(b). In the online component, we use a variable sliding window to read new data records. When the variable sliding window is full, the data stream is mapped into the temporary density grids and the characteristic vector of the corresponding grid cells is updated. Then we calculate the distance between the old density grids and the temporary density grids to detected concept drift (initially the old density grids was empty). If the distance is smaller than a certain threshold, no concept drift is detected, and the temporary density grids are merged into the old density grids and cleared. Then the variable sliding window is adjusted by our strategy described in Sect. 4.3. Otherwise, if the distance is greater than the threshold, concept drift is detected, the old density grids are copied and clustered in the offline component and cleared. The temporary density grids are copied to the old density grids and cleared. In the offline component, our DCDD forms clusters based on the copy of the old density grids. Besides, it extracts the concept-feature of this concept and adds it into the concept-list. In addition, when the size of concept-list is enough large, it uses the XGBoost (eXtreme Gradient Boosting) [4] to train the dataset of concept-list and then uses the trained model to predict the message of the next concept to adjust the variable sliding window.

4.2 Time-Weighted Concept Drift Detection

For the online component, to detect concept drift in a data stream, we apply an extended concept drift detection model of DCDA [2] to calculate the distance between the old density grid and the temporary density grid as follows, observing the deficiencies of DCDA:

We assign all grids in the old density grids a weight $\frac{1}{\delta^{t_{now}-t-1}}$, where t_{now} is present time, t is in the Grid Characteristic Vector and $\delta \in (0, 1)$ is a constant called the weight factor.

For dense grids in the temporary density grids T and dense grids in the old density grids O, the distance between T and O with respect to S is defined as

$$d_A(T, O) = \frac{1}{|S|} \sum_{s \in S} d_{\{s\}}(T, O) = \frac{\sum_{s \in S} \sum_{g \in T \cup O} \left| \mu_T^{\{s\}}(g) - \frac{1}{\delta^{t_{now}-t-1}} \mu_O^{\{s\}}(g) \right|}{|T \cup O||S|}, \quad (1)$$

where $\delta \in (0, 1)$ is a constant, t_{now} is the present time, t is in the Grid Characteristic Vector of g, $\frac{1}{\delta^{t_{now}-t-1}}$ is the weight for grid in the old density grids, S is the dimension of the defined data space, $\mu_G^{\{s\}}(g)$ is a rough membership function.

4.3 Sliding Window Size Calculation

To improve efficiency, our detection scheme uses a variable-size sliding window whose size is dynamically adjusted according to the framework in Fig. 2. Firstly, we initialize

a sliding window size $N = N_{init}$ and set a maximum size N_{MAX} based on the memory capacity. We detect the concept drift between the temporary density grids and the old density grids based on the detection formula of Eq. (1). If the old and temporary density grids present the same concept, we determine whether the temporary density grids have new dense grids that are not in the old density grids, and calculate the density of all these dense grids M. We then adjust the sliding window size to $N = N + M$ ($N = N_{MAX}$ if $N + M$ exceeds N_{MAX}); If the old and temporary density grids present concept drift, we revert the size of sliding window to the initial value in the next step ($N = N_{init}$). If our prediction model that is described in Sect. 4.4 has created, we set N based on the predicted value. Using the above strategy, our algorithm is described in algorithm 1.

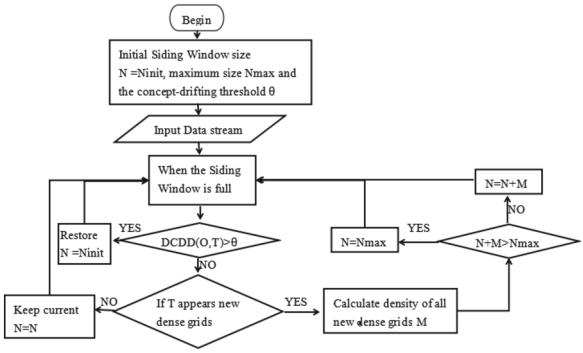


Fig. 2. Dynamic Adjustment of Sliding Window Size

4.4 Prediction on Concept-Feature Classification

When we collect certain amount of concept-feature, we use XGBoost [4] to train concept-feature to obtain a prediction model. Before running XGBoost, three parameters are set for xGboost: general parameters, booster parameters and task parameters. General parameters control the booster which are either tree model (tree) or linear model (linear) commonly.

We set 5 types of attributes for concept-feature, our prediction model with 5 trees is defined below:

$$model \ : \hat{y}_i = \sum_{k=1}^{5} f_k(x_i), f_k \in F$$

The objective is defined as:

$$Obj = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right) + \sum_{k=1}^{5} \Omega(f_k),$$

where $\sum_{i=1}^{n} l\left(y_i, \hat{y}_i\right)$ is the training loss.

This prediction model is based on concept-feature of the current concept drift to predict the amount of data stream in the next concept. When the current concept is over, we get the concept-feature and add concept-feature into concept-list. When the size of concept-list reaches β, the concept-list is trained by XGBoost. Using the model, we obtain the predicted value PN. So at the fastest detection speed, PN is divided into two parts, which effectively sets the size of variable sliding window to $\frac{PN}{2}$.

Algorithm 1 Adjust_Sliding_Window

Input: Old density girds $O =< G_{o1}, G_{o2}, \cdots, G_{on} >$;
Temporary density grids $T =< G_{t1}, G_{t2}, \cdots, G_{tn} >$;
Current sliding window size N;
Output: size of Sliding Window N.

```
(1) M = 0;
(2) for j=1 to |T|, T_j ∈ T
(3)     if (T_j is not in O)
(4)         M = M+|T_j|;
(5)     end if
(6) end for
(7) N = N+M
(8) if (N>Nmax)
(9)     N = Nmax;
(10) end if
(11) return N;
```

4.5 Algorithm Description

The whole algorithm of our density-based concept drift detection (DCDD) is presented in Algorithm 3. The algorithm for computing grid density distance is presented in Algorithm 2. Our DCDD algorithm can be simply summarized below: we use the detection formula of Eq. (1) to detect concept drift. If concept drift occurs, we use the prediction model to adjust the size of the sliding window. Otherwise, we use the strategy in Sect. 4.3 to adjust the size of the sliding window.

The time complexity of our detection algorithm is $O(|T \cup O|\,|S|)$, where T is the number of dense grids in the temporary density grids, O is the number of dense grids in the old density grids and S is the dimensions of the defined data space. Compared with DCDA [2], the time complexity of our detection is greatly reduced. It is easy to see that the time complexity of our detection algorithm is linear with respect to the number of dense girds.

Algorithm 2 Compute_Grid_Density_Distance

Input: Old density girds $O =< G_{o1}, G_{o2}, \cdots, G_{on} >$;
Temporary density grids $T =< G_{t1}, G_{t2}, \cdots, G_{tn} >$;
Output: Distance between T and O.
(1) $G = O \cup T$;
(2) distance = 0;
(3) **for** s=1 to |S| **do**
(4) $G/IND(S_s) = \{g_1, g_2, \cdots, g_m\}$, $S_s \epsilon S$;
(5) **for** j=1 to m_p **do**
(6) distance = distance + $||g_j \cap O| - \frac{1}{\delta^{tnow - t}} \times |g_j \cap T||$;
(7) **end for**
(8) **end for**
(9) **return** distance;

5 Experimental Results

We evaluate the precision, recall and efficiency of our DCDD and compare it with DCDA [2]. We use the synthetic data that contains 15% noisy data and a real data set KDD CUP-99 that is network intrusion detection data set and has been cited by many articles of data stream clustering. It collected 9 weeks of TCPdump (*) network connection and system audit data by the MIT Lincoln laboratory which contains the simulation of various types of users, a variety of network traffic and attack means, and it like a real network environment and the network intrusion detection data stream. It contains a total of 41 dimensional properties, of which 34 are continuous attributes. Each data stream of network connection is marked as normal or abnormal, and the abnormal type is subdivided into 4 main categories that are DOS, R2L, U2R and PROBING. We test the DCDD on KDD CUP-99 data set one hundred times persistently and test the accuracy of clustering results.

5.1 Performance Evaluation

In order to compare our proposed DCDD with the existing DCDA, we use the popular evaluation metrics of precision and recall. If a is the number of drifting concepts in the data set, b is the number of drifting concepts that we detect and c is the number of drifting concepts that are correctly detected. The precision and recall of the detection are defined as $Precision = \frac{c}{b}$ and $Recall = \frac{c}{a}$, respectively.

Firstly, we test DCDD under our detection formula Eq. (1) and DCDA with different initial sizes of the sliding window on the same synthetic dataset, and we set parameters $\theta = 0.3$, $\alpha = 0.5$ and set $\delta = 0.5$. From the result of the 3, although the recall of DCDD is slightly worse than the recall of DCDA, DCDD gives a much better precision than the DCDA (Fig. 3).

Algorithm 3 Density-based Concept Drift Detection (DCDD)

Input: Data Stream $S =< X_1, X_2, \cdots, X_d >$;
Output: Cluster Result.

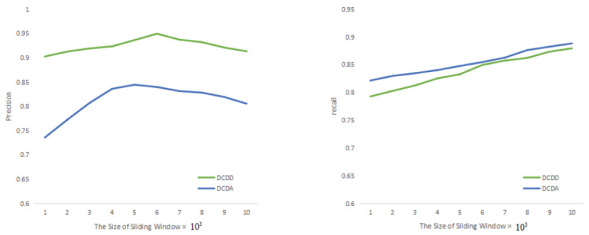
```
(1)  variable-size sliding window size N = Ninit;
(2)  initialize an empty concept_list;
(3)  old density girds O;
(4)  temporary density grids T;
(5)  while data stream is active do
(6)      add data stream X = (X1, X2, ···, Xd) into sliding window;
(7)      if (variable-size sliding window is FULL)
(8)        map the data into T;
(9)        if DCDD (O,T) ≥ θ
(10)           Cluster(O);
(11)           Cluster_Feature(O) add into Cluster_List;
(12)           if No Train(Cluster_List) and size of Cluster_List ≥β
(13)             obj = Train(Cluster_List);
(14)           end if;
(15)           clean O;
(16)           O=T;
(17)           if size of Cluster_List ≥β
(18)             N = obj(Cluster_Feature(O))/2;
(19)           end if
(20)           else
(21)             N = Ninit;
(22)         end if
(23)         else
(24)           N = Adjust_Sliding_Window(O,T);
(25)           clean O;
(26)         end else
(27)      end if
(28) end while
```

Then we test our DCDD using the real data set KDD CUP-99, we set the parameters of XGBoost: booster is gbtree and others are default and set $\beta = 10000$, $\delta = 0.5$. In our experiment, we use the strategy in Sect. 4.3 to adjust the size of the variable sliding window in the first ten thousand concept drifts that are used to train and we use the prediction model to adjust it afterwards. From the result in Fig. 4(a), it is obvious that the number of detected drifting concepts by our DCDD decreases with the increase of α. The precision and recall of DCDD are presented in Fig. 4(b). In these experiment, the threshold value θ is set to 0.1 and the size of the sliding window is initialized to 100. The parameter α is set from 0.2 to 1 with a step length of 0.2.

With the increase in the number of detected concept drifts, the precision and recall of DCDD with model 1 are shown in Fig. 4(c), where the threshold value θ is set to 0.1.

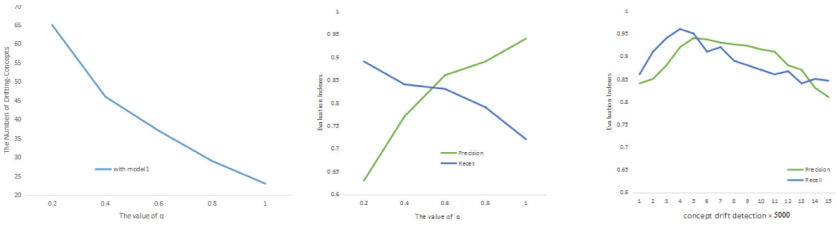
5.2 Result Comparison

We compare the experimental results of our DCDD with DCDA on F1-measure, where $F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$, and time consumption wrt the number of concept drifts. Firstly, the threshold value θ is set to 0.1 and α is set to 0.5. The results are shown in Fig. 5. We



(a) Precision (b) Recall

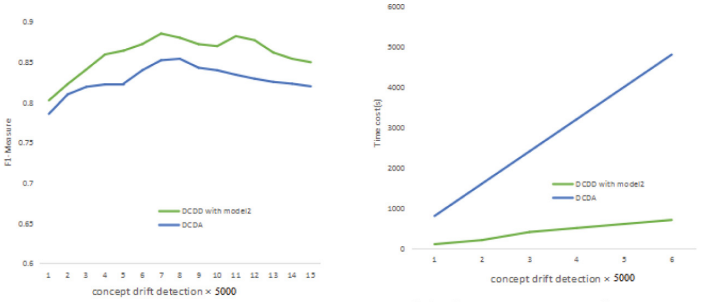
Fig. 3. Precision and recall of our DCDD and DCDA on synthetic dataset.



(a) (b) (c)

Fig. 4. Number of drifting concepts, precision and recall wrt to α and drift direction

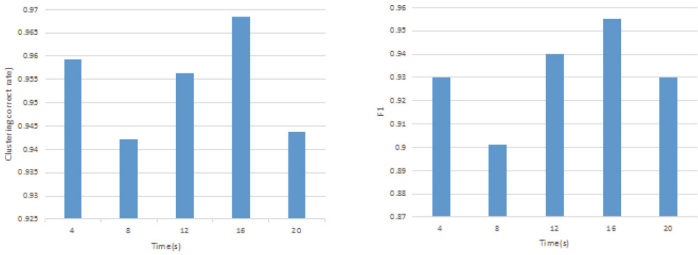
can see from Fig. 5(a) that F1-measure of DCDD is slightly better than DCDA at the first ten thousand concept drift, and then it grows to a more significant level as the number of concept drifts increases. For comparison of time consumption shown in Fig. 5(b), it is clear that our DCDD has a much lower time cost than DCDA and runs about 8 times faster than DCDA.



(a) Comparison of F1-measure (b) Comparison of time cost for Concept Drift Detection

Fig. 5. Comparison between DCDD with DCDA on KDD-CUP dataset.

The accuracy of clustering results of our DCDD on KDD-CUP dataset is shown in Fig. 6 (a) and the F1-measure of clustering results in Fig. 6 (b). From the clustering results



(a) Clustering accuracy t (b) F1 measure on

Fig. 6. Performance of DCDD on KDD-CUP dataset

at different times, DCDD achieves not only good accuracy but also good F1-measure at different times that are better than the results of DCDA [2].

6 Conclusion

In this paper, we proposed DCDD, a new framework for concept drift detection based on density-based clustering [15]. It improves the DCDA algorithm [2] both in terms of the F1-measure and computational cost (quite significantly). Our algorithm depends only on the number of grids rather than the number of data points in the grids, which makes it much more efficient. In addition, we proposed a strategy and prediction model to adjust the variable-size sliding window to adapt to the changes of data streams and to further improve the efficiency. In the future we will address the periodicity of concepts in data streams to gain improvement in detection accuracy.

Acknowledgement. This work is supported by Macao Polytechnic University Research Grant RP/FCA- 13/2022. The corresponding author is Hong Shen.

References

1. Aggarwal, C.C., Yu, P.S., Han, J., Wang, J.: A framework for clustering evolving data streams. In: International Conference on Very Large Data Bases, pp. 81–92 (2003)
2. Cao, F., Liang, J., Bai, L., Zhao, X., Dang, C.: A framework for clustering categorical time-evolving data. IEEE Trans. Fuzzy Syst. **18**(5), 872–882 (2010)
3. Chen, H.L., Chen, M.S., Lin, S.C.: Catching the trend: a framework for clustering concept-drifting categorical data. IEEE Trans. Knowl. Data Eng. **21**(5), 652–665 (2009)
4. Chen, T., He, T., Benesty, M.: Xgboost: extreme gradient boosting (2015)
5. Chen, Y., Tu, L.: Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 133–142 (2007)
6. Chi, Y., Song, X., Zhou, D., Hino, K., Tseng, B.L.: Evolutionary spectral clustering by incorporating temporal smoothness. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, pp. 153–162 (2007)

7. Cai, B., Hu, C., Ren, J.: Clustering over an evolving data stream based on grid density and correlation. ICIC Exp. Lett. **4**5(A), 1603–1609 (2010)
8. Corne, D., Handl, J., Knowles, J.: Evolutionary clustering. In: Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, pp. 332–337 (2006)
9. Cui, Z., Shen, H.: The framework of relative density-based clustering. In: Chen, G., Shen, H., Chen, M. (eds.) PAAP 2017. CCIS, vol. 729, pp. 343–352. Springer, Singapore (2017). https://doi.org/10.1007/978-981-10-6442-5_31
10. Gaber, M.M., Yu, P.S.: Detection and classification of changes in evolving data streams. Int. J. Inf. Technol. Decis. Mak. **05**(5), 659–670 (2006)
11. Granger, R.H., Schlimmer, J.C.: Beyond incremental processing: tracking concept drift. In: Proceeding of the Twenty-Second International Conference on Very Large Databases, pp. 502–507 (1986)
12. Jia, C., Tan, C.Y., Yong, A.: A grid and density-based clustering algorithm for processing data stream. In: International Conference on Genetic and Evolutionary Computing, pp. 517–521 (2008)
13. Pawlak, Z.: Rough sets. Int. J. Comput. Inform. Sci. **11**(5), 341–356 (1982)
14. Ren, J., Cai, B., Hu, C.: Clustering over data streams based on grid density and index tree. J. Converg. Inf. Technol. **6**(1), 83–93 (2011)
15. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: the algorithm gdbscan and its applications. Data Min. Knowl. Disc. **2**(2), 169–194 (1998)
16. Souza, V.M.A., Chowdhury, F.A., Mueen, A.: Unsupervised drift detection on high-speed data streams. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 102–111 (2020)
17. Tsymbal, A., Pechenizkiy, M., Cunningham, X.: Dynamic integration of classifiers for handling concept drift. Information Fusion **9**(1), 56–68 (2008)
18. Tu, L., Chen, Y.: Stream data clustering based on grid density and attraction. ACM Trans. Knowl. Discov. Data **3**(3), 167–176 (2009)
19. Wang, P., Jin, N., Fehringer, G.: Concept drift detection with false positive rate for multi-label classification in iot data stream. In: 2020 International Conference on UK-China Emerging Technologies (UCET), pp. 1–4 (2020)