ORIGINAL RESEARCH

A statistical approach for clustering in streaming data

Niloofar Mozafari [*], Sattar Hashemi, Ali Hamzeh

School of Electrical Computer Engineering, Shiraz University, Shiraz, Iran.

Received: June 30, 2013 **Accepted:** November 26, 2013 **Online Published:** January 9, 2014
DOI: 10.5430/air.v3n1p38 **URL:** http://dx.doi.org/10.5430/air.v3n1p38

Abstract

Recently data stream has been extensively explored due to its emergence in large deal of applications such as sensor networks, web click streams and network flows. Vast majority of researches in the context of data stream mining are devoted to supervise learning, whereas, in real word human practice label of data are rarely available to the learning algorithms. Hence, clustering as the most important unsupervised learning has been in the gravity of focus of quite a lot number of the researchers in data stream community. Clustering paradigms basically place the similar objects together and separate the dissimilar ones into different clusters.

In this paper, we propose a Statistical framework for data Stream Clustering, which abbreviated as *StatisStreamClust* that makes use of two components to find clusters in data stream. The first component especially designed to detect concept change where data underlying distributions change from time to time. Upon detection of concept change by the first component, the second component is triggered to update the whole clustering model. *StatisStreamClust* brings great benefits to data stream clustering including no sensitivity to the number of clusters and dimensions, reasonable complexity and in the meantime desirable performance, and finally no need to determine window size a priori. To explore the advantages of our approach, quite a lot of experiments with different settings and specifications are conducted. The obtained results are very promising.

Key Words: Data stream, Trend, Concept change, Precision, Recall, F1 measure, Mean delay time

1 Introduction

Recently, data streams have been extensively investigated due to the large amount of applications such as sensor networks, web click streams and network flows.[1–6] Data stream is an ordered sequence of data with huge volumes arriving at a high throughput that must be analyzed in a single pass.[7–9] Vast majority of researches in the context of data stream mining are devoted to supervise learning, whereas, in real word human practice label of data are rarely available to the learning algorithms. Clustering as the most important unsupervised learning has been in the gravity of focus of quite a lot numbers of researchers in data stream community. It inputs the similar objects together and separates dissimilar ones into different clusters.[10]

An intuitive approach for clustering data streams is to recluster them periodically. At the predetermined time point, clusters of data streams are updated with an existing clustering algorithm. This approach has two major difficulties. First, due to huge data volume of streaming data, reclustering of them is very costly. Second, in which time the data must be reclustered. If the reclustering of data is done from time to time, most of the reclustering tasks are unnecessary. On the other hand, if the distance of reclustering tasks is partly far, some cluster information may be loosed. So, we need a solution that is able to perform clustering whenever it is necessary or whenever the nature of streaming data changes.

In this paper, we propose StatisStreamClust (Statistical framework for data Stream Clustering) that partitions the input instances whenever it is needed and involves two components to find clusters in data stream. First, it detects concept change where data underlying distributions change from time to time. Then, after detecting a change, the underlying clustering model would be updated. Briefly, the con-

[*] **Correspondence:** Dr. Niloofar Mozafari; Email: mozafari@cse.shirazu.ac.ir; Address: Department of Computer Science and Engineering and Information Technology, School of Electrical Computer Engineering, Zand Avenue, Shiraz University, Iran.

tributions of our approach are multiple folds. First, it does not require a sliding window on the data stream whose size is a well-known challenging issue; second, it works well in multi-dimensional data stream and also it is not sensitive to the number of dimension or number of clusters. To explore the advantages of our approach, quite a lot of experiments with different settings and specifications are conducted. The obtained results suggest that StatisStreamClust is the method of choice by offering a reasonable complexity and desirable performance.

The rest of this paper is organized as follows: Section 2 reviews the related works, and in Section 3 the proposed algorithm for data stream clustering is explained. The experimental results are given in Section 4 and Section 5 brings the paper to the end by giving conclusion and future work.

2 Related work

The problem of concept change detection in unlabeled time-evolving data is formulated as follows: we are given a series of unlabeled data points D = $z_1, z_2, \ldots, z_n$. D can be divided into s segments Di where i=1, 2 , .., s that follow different distributions. Change in the distribution of data causes some difficulties in data stream learning problems. For example, performing clustering on the entire data streams uninterested in concept change decreases the quality of clusters. Hence, there has been developed some methods to work along concept change. Some of these methods can handle concept change using clustering and the others focus exclusively on concept change detection. We classify them into two groups: 1) Model driven concept change detection: the main aim of these systems is clustering and update clusters whenever the accuracy of clustering is decreased. 2) Data driven concept change detection: these methods focus exclusively on concept change detection according to the nature of data. In the following, we review two types of these systems.

2.1 Model driven concept change detection

The main goal of these methods is clustering and update clusters whenever the accuracy of clustering is decreased. Thus, they handle concept change in data stream by update clusters when the accuracy is decreased. This idea is first introduced by Aggarwal[11] that introduced a framework for data stream clustering, namely cluStream. That framework is separated into two components: 1) Online component that partitions data stream into clusters and adopts a very efficient process to storage the appropriate summary statistics for each cluster. 2) The offline component that applies whenever a user required. This component uses the summary statistics that are stored in the online component and the other user input, to provide the user with a quit understanding of the clusters. However, cluStream has the property of interpretable ability to track evolving clusters, but not designed to handle outliers.

In[12] a method for clustering data streams with arbitrary shape and the ability to handle outliers is proposed. That method is also followed two processes of online and offline in order to partition data stream with considering concept change. Nasraoui et al. proposed a density based method for clustering data streams into one step.[8] That algorithm adopts sliding window on the data stream so that partitions instances according to a density based clustering in the first window. It stores a representation for each cluster that involves the center of cluster and whose weight of instances. The weight of instances in each cluster determines according to both the distance of each instance to the center of cluster and also the time of entering instance to that cluster. Instances in each window belong to appropriate cluster according to the representation of each cluster and then the parameters of the clusters are updated. Also in that algorithm, outliers are removed using the statistical tests.

Another view of clustering in numerical data stream with considering concept change, namely evolutionary clustering is proposed in.[13, 14] Evolutionary clustering optimizes two potentially conflicting criteria: 1) Clustering in current window should be similar to clustering in previous window (without considering concept change). 2) Clustering in current window should be able to partition instances accurately (with considering concept change). Chakrabarti et al.[13] formulated the concept of evolutionary clustering and extended k-means[15] and agglomerative hierarchical[45] to evolutionary clustering. The idea of temporal smoothness is proposed in.[14] Temporal smoothness implicate that the current clustering do not deviate dramatically from the most recent history clustering. According to this idea, two frameworks for evolutionary spectral clustering[16] are proposed, so that temporal smoothness is incorporated into the overall clustering quality. Although that those method have good performance for data stream clustering with considering concept change, cannot handle sudden concept change because they consider sudden concept change as noise.

Dai et al. proposed a framework for clustering in data streams with considering concept change that unlike of previous works partitions data streams rather than their data points into clusters.[17] This type of clustering data streams not their data points has vast applications such as in sensor networks and stock markets. That framework involves two phases: 1) Online maintenance phase that devotes an efficient algorithm to store a summary for data stream with multiple resolutions. 2) Offline phase that employs an adaptive clustering algorithm to retrieve the approximation of the desired sub streams, based on the clustering queries specified by the user, form the summary which stored in online phase. In[18] an online clustering without providing details in offline phase was proposed for clustering data streams not their data points. It continuously reports clusters within the given distance threshold. Also, Beringer et al. presented an algorithm for clustering over parallel data streams.[19] That method summarized data streams using Discrete Fourier Transform and reports clustering results by applying a sliding window on streams. Another work[20] constructs incrementally a hierarchy of clusters form a divisive point of view to provide a time series system for whole clustering. That method performs clustering whenever a number of data points of each time series are received. Yeh et al. proposed an algorithm for clustering data streams not their data points that reports dynamically cluster evolutions with efficient cluster split and merge processes which trigged by events.[22]

In[21] an algorithm for mining evolving user profiles in the web is proposed. For each cluster, a series of information such as birth, persistence, atavism and the death are defined to model the profiling web usages. In order to track evolving user profiles, the results of clustering are analyzed. In 2009, Chen et al. presented a framework that is able to detect concept in categorical domain change and show the trend of evolving clusters.[23]

2.2 Data driven concept change detection

These methods focus exclusively on concept change detection according to concept change definition and the nature of data. In general, change is defined as moving from one state to another state.[24] There are some important works to detect changes where some of them detect changes with statistical hypothesis testing and multiple testing problems.[25] In the statistical literature, there are some works for change point detection.[26] However, most of the statistical tests are parametric and also needs the whole data to run.[27,28] These methods are not applicable in the data stream area, because they require storing all data in memory to run their employed tests.[28] Popular approaches for the concept change detection uses three techniques including (1) Sliding window which is adopted to select data points for building a model.[29,30] (2) Instance weighting which assumes that recent data points in window is more important than the other.[31,32] (3) Ensemble learning which is created with multiple models with different window sizes or parameter values or weighting functions. Then, the prediction is based on the majority vote of the different models.[33–36] Both sliding window and instance weighting families suffer from some issues: First, they are parametric methods; the sliding window techniques require determining window size and instance weighting methods need to determine a proper weighting function. Second, when there is no concept change in the data stream for a long period of time, both of sliding window and instance weighting methods would not work well because they do not take into account or give low weights to the ancient instances.[37] The ensemble methods try to overcome the problems that sliding window and instance weighting are faced with by deciding according to the reaction of multiple models with different window sizes or parameter values or weighting functions. However, these techniques need to determine the number of models in the ensemble technique.

Another family of concept change detection methods is based on density estimation. For example, Aggarwal's method[38] uses velocity density estimation which is based on some heuristics instead of classic statistical changes detectors to find changes. In order to provide an intuitive understanding of the rate of change in the density at a spatial location over a window, concept of velocity density which is determined according to forward time slice density estimate and reverse time slice density estimate is defined. Aggarwal's method visualizes the rate of flowing in data stream. The user decides whether concept change occurred or not. As another major works in this family, we could mention Kifer's[39] and Dasu's works[40] which try to determine the changes based on comparing two probability distributions from two different windows.[39,40] For example, in[39] the change detection method based on KS test determines whether the two probability density estimations obtained from two consequent different windows are similar or not. That approach uses two windows, namely reference window and current window over data streams and determines whether the two samples in the windows created by same distribution function or not. Although that It assumes the instances are generated independently, but does not make any assumption on the type of distribution function that generated instances. That approach for change detection is impractical for high dimensional data streams. Dasu et al. propose a method for change detection which is related to Kulldorff's test. This method is practical for multi-dimensional data streams.[40] However, this method relies on a discretization of the data space, thus it suffers from the curse of dimensionality.

Another major work is proposed by Ho et al.[24] In Ho's method, upon arrival of new data point, a hypothesis test takes place to determine whether a concept change has been occurred or not. This hypothesis test is driven by a family of martingales[24] which is based on Doob's Maximal Inequality.[41] Although Ho's method detects changes points accurately, it can only detect some types of changes to be more detailed in.[42]

3 StatisStreamClust

The proposed method partitions instances into k clusters after arriving n instances. Whenever an instance is received, it belongs to the proper cluster according to its distance to the mean of the clusters. In other words, the distance of the new received instance to the mean of the existing clusters is calculated and the instance belongs to the cluster with the shortest distance and the mean of that cluster is updated. Suppose that mean of n received instances in the cluster k is $m_{k,n}$. The instance x_{n+1} belong to the proper cluster using

Formula (1):

$$x_{n+1} \in cl_k : k = \min_k dist(x_{n+1}, m_{k,n}) \tag{1}$$

where cl_k is k^{th} cluster and dist(x_{n+1},$m_{k,n}$) is the distance of instance x_{n+1} to the mean of cluster k. After belonging instance x_{n+1} to the cluster k, the new mean of the cluster is calculated using the following formula:

$$m'_{k,n+1} = \frac{n.m_{k,n} + x_{n+1}}{n+1} \tag{2}$$

After that, a hypothesis testing takes place to determine whether concept change is occurred or not. This test is according to exchangeability condition and defined as follow:
H_0=$Thereisnoconceptchageindatastream$
H_0=$Thereisaconceptchageindatastream$
In order to take place the hypothesis test, *statisStreamClust* investigates the status of the new received instance in comparison of the other instances existing in the cluster k. If there exists a change in the trend of instances existing in every cluster, it means a concept change occurred and the clustering model must be updated. To do that, the distance of the all instances existing in the cluster k to $m'_{k,n}$ is calculated and the trend of instances was calculated using the following formula:

$$p - value = \frac{\#\{i : SM_{k,t} > SM_{k,n}\} + \theta_n \#\{i : SM_{k,i} = SM_{k,n}\}}{n} \tag{3}$$

where θ_n is chosen from [0,1] randomly. $SM_{k,i}$ is the distance of the i^{th} instance existing to the cluster k to its mean. It determines how much a data point is different from the others. SM is high, when data is farther from the mean of the data points.
The changes of *p_value* toward higher values can be deemed as data points are running away from their mean. In contrast, having data close to their mean conveys the meaning that p_values are approaching smaller values. In order to decide whether H0 must be accepted or not, a martingale[24] is defined based on the sequence of p_values.

$$M_i^{(\varepsilon)} = \varepsilon p_i^{\varepsilon - 1} M_{i-1}^{(\varepsilon)} \tag{4}$$

According to Doob's Maximal Inequality,[41] it is unlikely for Mk to have a high value. Thus, we can detect changes when martingale value is greater than λ.[42]
When a change is occurred in any cluster, all the previous information is removed. To be more illustrative, we presented the outline of our method for clustering data stream as follow:
StatisStreamClustStatisStreamClust
Partitions instances after arriving 50 instances using k-means algorithm Loop A new unlabeled data stream z_i is received. Find the proper cluster Compute the distance of z_i to the mean of instances of that cluster. Compute p_values and p-values'=1-p_values using (1). Compute Mi and Mi' using (2) according to p_values (Mi) and the 1 – p_values (Mi'). If $\frac{M_i + M_{i'}}{2} > \lambda$ then Update Clustering Model Delete all previous information End if End loop

4 Experimental Results and Discussion

This section composes of two subsections, precisely covering our observation and analysis. The first subsection presents the data sets and evaluation measures. The latter one presents and analyses the obtained results.

4.1 Experimental Setup

This section introduces the examined data sets and evaluation measures respectively.

4.1.1 Data sets

To explore advantages of *statisStreamClust*, we conduct our experiments on data set which was used previously in Aggarwal's work.[11] The instances of this data set follow a series of Gaussian distributions. In order to reflect the concept change, we change the mean and variance of the current Gaussian distribution after arriving 1050 instances.

4.1.2 Evaluation measures

There are many evaluation measures to assess the performance of clustering algorithms. They can be categorized into two groups: internal and external criteria. An internal criterion formulates the quality of clustering model as a function of the instances and the similarities between them. External criteria use external information not given to the clustering algorithm, such as label of the instances. In this paper, we evaluate our method with the both groups.
As an internal criterion, we use a famous cluster validation technique –Silhouette validation.[43] This measure first assigns a quality measure to each instance,

$$sil(d_i) = \frac{b(d_i) - a(d)}{\max\{a(d_i), b(d_i)\}} \tag{5}$$

where $a(d_i)$ is the average of the Euclidian distance of instance di to all other instances in the same cluster and $b(s_i)$ is the average of the Euclidian distance of instance d_i to all other instances in the closest cluster. A cluster silhouette for a cluster C_k who has m instances is defined as follow:

$$sil(c_k) = \frac{\sum_{d_i \in C_k} sil(d_i)}{m} \tag{6}$$

Finally, the Global Silhouette, GS, which is used to evaluate clustering quality as an internal criterion is defined as Formula (7).

$$GS = \frac{1}{p} \sum_{1}^{p} sil_{C_k} \tag{7}$$

where p is the number of clusters. Silhouette validation takes into account the compactness of the instances in each cluster and the separation of clusters.

To evaluate the accuracy of the proposed method with an external criterion, we focus on Normalized Mutual Information (NMI) which is an information theoretic measure that was previously used in.[44]

$$NMI(\lambda^a, \lambda^b) = \frac{\sum_{h=1}^{k^{(a)}} \sum_{l=1}^{k^{(b)}} \log(\frac{n.n_{h,l}}{n_h^{(a)} n_l^{(b)}})}{\sqrt{(\sum_{h=1}^{k^{(a)}} n_h^{(a)} \log \frac{n_h^{(a)}}{n})(\sum_{l=1}^{k^{(b)}} n_l^{(b)} \log \frac{n_l^{(b)}}{n})}} \tag{8}$$

In this formula, λ^a is the true label of instances and λ^b is the result of clustering using the proposed method. K(a) and K(b) are the number of clusters in λ^a and λ^b respectively. $n_h^{(a)}$ is the number of data in hth cluster. $n_{h,l}$ defines the set of common instances in cluster h and l.

The Normalized Mutual Information measures the degree of similarity between two clustering. If two clustering have much information in common, this measure gets high value; i.e. close to 1 and vice versa.

4.2 Results and discussion

To assess the performance of *StatisStreamClust*, we compare it with two other methods: one is called Basic and the other is Window based algorithm. The Basic algorithm does clustering once total instances are collected and the Window based algorithm partitions streaming data after arriving 2000 instances. To have fair comparison, *statisStreamClust* and the other two algorithms use k-means algorithm. Figures 2, 3 and 4 visualize the ability of Basic, Window based and our method in partitioning streaming data respectively. Horizontal axis shows the time and the vertical one illustrates the value of instances in each time. The instances of this data set come from five Gaussian distributions that mean of them changes after 1050 instances. Figure 1 illustrates this data set. Each colour indicates the instances that are generated from a Gaussian distribution.

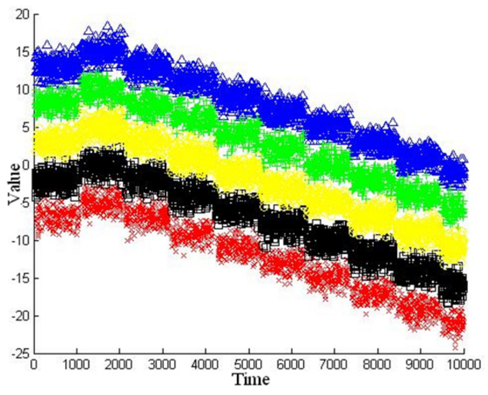

Figure 1: The instances of this data set come from five Gaussian distributions that mean of them changes after 1050 instances. Each color indicates the instances that generates from a Gaussian distribution

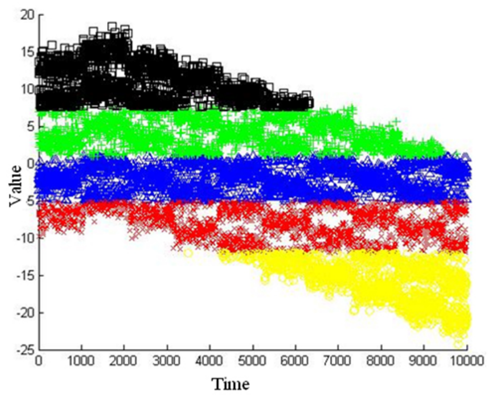

Figure 2: The ability of Basic method to partition streaming data. The basic algorithm does clustering once total examples are collected. Horizontal axis shows the time and the vertical one illustrates the value of instances in each time

As Figure 2 shows Basic window algorithm has poor performance to partition instances over time because it does not consider the concept change where the distribution of instances changes. Window based algorithm has the better performance in partitioning streaming data because it partitions data after arriving 2000 instances and then removes all the instances existing in the previous window. Although the performance of Window based algorithm to partition streaming data is partly good, it is not comparable to *statisStreamClust*. The proposed algorithm partitions instances into k clusters using k-means algorithm after arriving 50 instances. Whenever an instance is received, it belongs to the proper cluster according to its distance to the mean of the clusters and then a martingale is run for that cluster. If the martingale of each cluster is greater than λ, the clustering model is updated and all the previous information is deleted. To conclude, statisStreamClust assigns instances into the proper cluster and updates clustering model whenever needed.

Our method takes into account the trend of data behaviour which can be close or away from the centre of data and after detecting change, removes all the previous information. Basic algorithm does not consider concept change and run clustering algorithm once all instances are collected. Window based algorithm partitions instances after arriving 2000 instances. So it has the better performance in comparison of Basic. Although the performance of Window based is not comparable to our algorithm because our method updates clustering model whenever it is needed. Otherwise it assigns each instance to the proper cluster according to its distance to the mean of cluster.

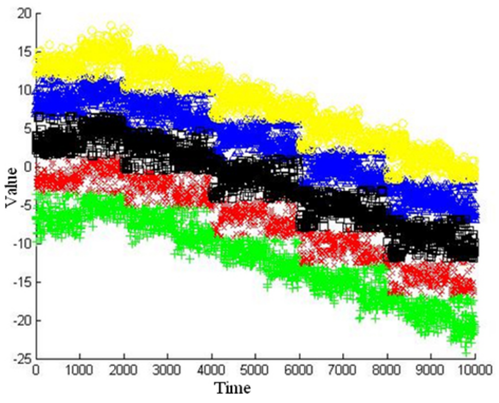

Figure 3: The ability of Window based algorithm to partition streaming data. Window based algorithm does clustering after arriving a fixed number of instances (2000 instances). Horizontal axis shows the time and the vertical one illustrates the value of instances in each time

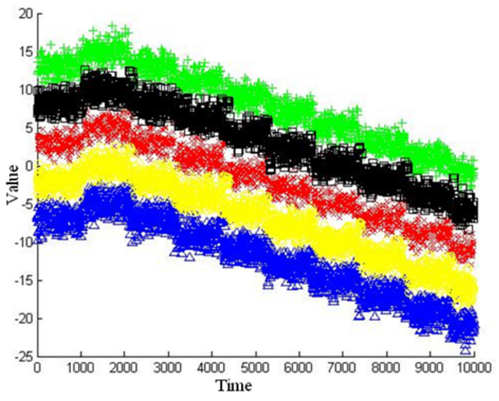

Figure 4: The ability of statisStreamClust to partition streaming data. The proposed approach for clustering streaming data reclusters instances whenever a change is detected. Otherwise the clustering model is updated. Horizontal axis shows the time and the vertical one illustrates the value of instances in each time

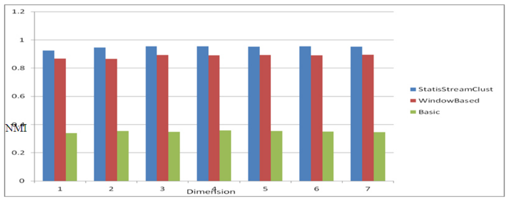

Figure 5: The comparison of the proposed method with two algorithms of Basic and Window based in the different dimensions with NMI.

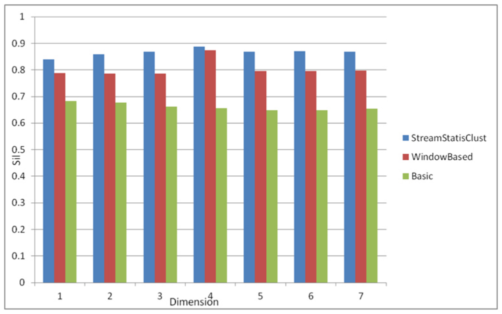

Figure 6: The comparison of the proposed method with two algorithms of Basic and Window based in the different dimensions with Silhouette.

Our experimental observations and theoretical analysis clearly reveal that statisStreamClust is able to partition streaming data in a robust manner. This method is robust to both the number of clusters and number of dimensions. To be more illustrative, we draw the readers' attention to the fact that part of our evaluations are carried out on data stream with different number of clusters (Table 1). From Figures 5 and 6, one may come up with the fact that robustness to the number of dimensions is an intrinsic nature of *statisStreamClust*.

The complexity of statisStreamClust is O(nlogn). Upon arrival of new data point, a proper cluster for that instance is found according to it distance to the mean of clusters by complexity of O(k), where k is the number of clusters. After that, a hypothesis test takes place to determine whether a concept change has been occurred in that cluster or not. This hypothesis test is driven by a family of martingales which is based on Doob's Maximal Inequality. To do this hypothesis test, statisStreamClust gets USM which ranks data points according to their mean with the complexity of O(n). In next step, p_value statistic is defined to rank USM of new data point with respect to the other USM. This step can be done by sorting the values and it would be done in O(nlogn) when heap-sort is used. In order to decide whether to accept H0 or not, a martingale is defined based on the sequence of p_value. The complexity of this step is O(1). Therefore the complexity of our algorithm is O(k)+O(n)+O(nlogn)+O(1)=O(nlogn).

Table 1: comparison of StatisStreamClust with algorithms of Basic and Window based in different number of clusters with two evaluation measures.

		2 clusters	3 clusters	4 clusters	5 clusters	6 clusters	7 clusters
Our Approach	NMI	0.8861 ± 5.6460e-016	0.9241 ± 0.0024	0.9398 ± 0.0037	0.9367 ± 0.0015	0.9348 ± 0.0024	0.9331 ± 0.0025
	Silhouette	0.8759 ± 5.6460e-016	0.8513 ± 0.0062	0.8291 ± 0.0072	0.7723 ± 0.0072	0.7725 ± 0.0094	0.7525 ± 0.0162
Basic	NMI	0.1449 ± 2.0639e-004	0.3070 ± 3.9502e-004	0.3912 ± 3.1308e-005	0.4405 ± 1.6552e-004	0.4851 ± 3.2827e-004	0.5122 ± 2.7958e-004
	Silhouette	0.7277 ± 1.5197e-005	0.7092 ± 3.6691e-004	0.7087 ± 2.7655e-005	0.7136 ± 1.3144e-005	0.6979 ± 8.2859e-005	0.7026 ± 4.9147e-005
Window Based	NMI	0.7250 ± 3.3876e-016	0.7794 ± 1.1292e-016	0.806 ± 3.4237e-004	0.8076 ± 4.5168e-016	0.8238 ± 0.0013	0.7740 ± 0.0670
	Silhouette	0.8150 ± 1.1292e-016	0.7904 ± 1.0000e-007	0.7724 ± 1.1052e-005	0.7652 ± 3.3876e-016	0.7562 ± 3.1007e-005	0.7544 ± 0.0084

5 Conclusion

Clustering as the most important unsupervised learning has been in the gravity of focus of quite a lot number of researchers in data stream community. It inputs the similar objects together and separates dissimilar ones into different clusters. Although data stream communities have recently focused on unsupervised domain and clustering as the most important unsupervised learning, the proposed approaches are not yet matured to the point to be relied on. In other words, most of them provide merely a mediocre performance specially when applied on multi-dimensional data streams. In this paper, we propose a statistical algorithm that partitions streaming data whenever it is needed. To do that, our algorithm first detects where the nature of data changes over the time and then recluster instances in those times. The advantages of our approach are the following. First, it does not require a sliding window on the data stream whose size is a well-known challenging issue; second, it works well in multi-dimensional data stream. To explore the advantages of our approach, quite a lot of experiments with different settings and specifications are conducted. The obtained results are very promising. In the future work, we will investigate clustering data stream where the number of clusters changes over the time.

References

[1] Babcock, B., Babu, S., Datar, R., Motwani, R. and Widom, J.: Models and Issues in Data Stream Systems, in proceedings of ACM Symp, Principles of Databases Systems (PODS), pp. 1-16, 2002.

[2] Fan, W.: Systematic Data Selection to Mine Concept Drifting Data Streams, in Proceedings of ACM SIGKDD, pp. 128-137, 2004. PMid:15079794

[3] Liu, X., Guan, J., Hu, P.: Mining Frequent Closed Item Sets from a Landmark Window Over Online Data Streams, in journal of computers and mathematics with applications, 57: 927-936, 2009.

[4] Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: On Demand Classification of Data Streams, in proceedings of ACM SIGKDD, pp. 503-508, 2004.

[5] Jiang, T., Feng, Y., Zhang, B.: Online Detecting and Predicting Special Patterns over Financial Data Streams, Journal of Universal Computer Science, 15(13): 2566-2585, 2009.

[6] Hashemi, S., Yang, Y.: Flexible decision tree for data stream classification in the presence of concept change, noise and missing values, Data Mining and Knowledge Discovery, 19(1): 95-131 (2009). `http://dx.doi.org/10.1007/s10618-009-0130-9`

[7] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, Morgan Kaufmann, (2001).

[8] Nasraoui, O., Rojas, C.: Robust Clustering for Tracking Noisy Evolving Data Streams, Proceedings of Sixth SIAM International Conference of Data Mining (SDM), 2006.

[9] Hashemi, S., Yang, Y., Mirzamomen, Z., Kangavari, M.: Adapted One-versus-All Decision Trees for Data Stream Classification, in IEEE Transaction on Knowledge and Data Engineering, 21(5): 624-637 (2009). `http://dx.doi.org/10.1109/TKDE.2008.181`

[10] Mirking, B. G.,: Mathematical Classification and Clustering, 1996.

[11] Aggarwal, C., Han, J., Wang, J., and Yu, P.: A Framework for Clustering Evolving Data Stream, VLDB conference, 2003.

[12] Cao, F., Ester, M., Qian, W., and Zhou, A.: Density-Based Clustering over an Evolving Data Stream with Noise, in proceedings of sixth SIAM international conference of Data Mining (SDM), 2006.

[13] Chakrabarti, D., Kumar, R. and Tomkins, A.: Evolutionary Clustering, Proceedings of ACM SIGKDD '06, pp. 554-560, 2006.

[14] Chi, Y., Song, X -D., Zhou, D.-Y., Hino, K. and Tseng, B. L.: Evolutionary Spectral Clustering by Incorporating Temporal Smoothness, in proceeding of ACM SIGKDD '07, pp. 153-162, 2007.

[15] Lloyd, S. P.: Least square quantization in PCM", in IEEE Transactions on Information Theory, vol. 28, pp.: 129–137, 1957. `http://dx.doi.org/10.1109/TIT.1982.1056489`

[16] Ng, A., Jordan, M. and Weiss, Y.: On spectral clustering: Analysis and an Algorithm, Advances in Neural Information Processing Systems 14: Proceedings of the 2001.

[17] Dai, B. -R., Huang, J. -W., Yeh, M. -Y. and Chen, M. -S.: Adaptive Clustering for Multiple Evolving Streams, IEEE transactions on knowledge and data engineering, Vol. 18, No. 9, 2006.

[18] Yang, J.: Dynamic Clustering of Evolving Streams with a Single Pass, in proceeding of international conference on Data Engineering, pp. 695-697, 2003.

[19] Beringer, J. and Hullermeier, E.: Online Clustering of Parallel Data Streams, in Data and Knowledge Engineering, vol. 58, no. 2, pp.

180-204, 2005. http://dx.doi.org/10.1016/j.datak.2005.05.009

[20] Rodrigues, P.P., Gama, J. and Pedroso, J.P.: ODAC: Hierarchical Clustering of Time Series Data Streams, in proceeding of 6th international conference on Data Mining, pp. 499-503, 2006.

[21] Nasraoui, O., Soliman, M., Saka, E., Badia, A. and Germain, R.: A Web Usage Mining Framework for Mining Evolving User Profiles in Dynamic Web Sites, in IEEE Transactions on Knowledge and Data Engineering, VOL. 20 , Issue 2, PP. 202-215, 2008.

[22] Yeh, M. -Y., Dai, B. -R., and Chen, M. -S.: Clustering over Multiple Evolving Streams by Events and Correlations, IEEE transactions on knowledge and data engineering, VOL.19, NO.10, 2007.

[23] Chen, H., Chen, M., Lin, S.: Catching the Trend: a Framework for Clustering Concept-Drifting Categorical Data, in IEEE transaction on knowledge and data engineering, VOL. 21, NO. 5, 2009.

[24] Ho, S.-S., Wechsler, H.: A martingale framework for detecting changes in data streams by testing exchangeability, in IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, 2010.

[25] Bickel, P.J. and Doksum, K.: Mathematical statistics: basic ideas and selected topics Holden-Day, Inc., 1977.

[26] Carlstein, E., Muller, H.-G. and Siegmund, D. Editors: Change point problems, Institute of Mathematical Statistics, Hayward, California, 1994. PMid:8034709

[27] Glaz, J., Balakrishnan, N., editors: Scan statistics and applications, Boston, 1999. http://dx.doi.org/10.1007/978-1-4612-1578-3

[28] Glaz, J., Naus, J. and Wallenstein, S.: Scan statistics, springer, New York, 2001. http://dx.doi.org/10.1007/978-1-4757-3460-7

[29] Klinkenberg, R. and Joachims, T.: Detecting concept drift with support vector machines. in proceedings of 17th international conference on Machine Learning, P. Langley, Ed. Morgan Kaufmann, pp. 487–494, 2000.

[30] Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts, journal of Machine Learning, 23, pp. 69-101, 1996.

[31] Klinkenberg, R.: Learning drifting concepts: examples selection vs example weighting, Intelligent Data Analysis, Special Issue on Incremental Learning Systems capable of dealing with concept drift, vol. 8, no. 3, pp. 281–300, 2004.

[32] Chu, F., Wang, Y. and Zaniolo, C.: An adaptive learning approach for noisy data streams. in proceedings of 4th IEEE internaional conference on Data Mining. IEEE Computer Society, pp. 351–354, 2004

[33] Kolter, J. Z. and Maloof, M. A.: Dynamic weighted majority: A new ensemble method for tracking concept drift. in proceedings of 3th IEEE international conference on Data Mining. IEEE Computer Society, pp. 123–130, 2003.

[34] Wang, H., Fan, W., Yu, P. S. and Han, J.: Mining conceptdrifting data streams using ensemble classifiers, in proceedings of 9th ACM SIGKDD internaional conference on Knowledge Discovery and Data Mining, L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, Eds. ACM, pp. 226–235, 2003.

[35] Scholz, M. and Klinkenberg, R.: Boosting classifiers for drifting concepts, in Intelligent Data Analysis, Vol. 11 No. 1, pp. 3-28, 2007.

[36] Bousquet, O. and Warmuth, M.: Tracking a Small Set of Experts by Mixing Past Posteriors, in Journal of Machine Learning Research, Vol. 3, pp. 363-396, 2002.

[37] Ho, S.-S. and Wechsler, H.: Detecting changes in unlabeled data streams using martingale, in Proceeding 20th International Joint Conference on Artificial Intelligence, M. Veloso, pp. 1912–1917, 2007.

[38] Aggarwal, C. C.: A framework for change diagnosis of data streams, in proceedings of ACM SIGMOD internaional conference on Management of Data, pp. 575–586, 2003.

[39] Kifer, D., Ben-David, S. and Gehrke, J.: Detecting change in data streams, in proceedings of 13th internaional conference on Very Large Data Bases, M. A. Nascimento, M. T. O zsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, Eds. Morgan Kaufmann, pp. 180–191, 2004.

[40] Dasu, T., Krishnan, S., Venkatasubramanian, S. and Yi, K.: An information-theoretic approach to detecting changes in multidimensional data streams, in Interface, 2006.

[41] Wetherill, G. B. and Glazebrook, K. D.: Sequential methods in statistics, 3rd ed. Chapman and Hall, 1986.

[42] Mozafari, N., Hashemi, S., Hamzeh, A.: A Precise Statistical Approach for Concept Change Detection in Unlabeled Data Streams, to be appeared in Journal of Computers and Mathematics with Applications, Elsevier, 2011.

[43] Rousseeuw, P.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis, J. Computational and Applied Math., vol. 20, pp. 53-65, 1987. http://dx.doi.org/10.1016/0377-0427(87)90125-7

[44] Strehl, A., Ghosh, J.: Cluster ensembles – a knowledge reuse framework for combining partitionings, in the journal of machine learning research (JMLR), 2002.

[45] Ward, J.: Hierarchical Grouping to Optimize an Objective Function, Journal of the American Statistical Association 58(301): 236-244, 1963. http://dx.doi.org/10.1080/01621459.1963.10500845