International Journal of Database Theory and Application
Vol.8, No.1 (2015), pp.55-62
http://dx.doi.org/10.14257/ijdta.2015.8.1.07

# Study of Data Stream Clustering Based on MSF

Yingmei Li, Min Li, Jingbo Shao and Gaoyang Wang

*College of Computer Science and Information Engineering, Harbin Normal University, 150025 Harbin, China*
*yingmei_li2013@163.com*

***Abstract***

*Nowadays with the rapid development of wireless sensor networks, and network traffic monitoring, stream data gradually becomes one of the most popular data models. Stream data is different from the traditional static data. Clustering analysis is an important technology for data mining, so that many researchers pay their attention to the clustering of stream data. In this paper, MSFS algorithm is proposed. By means of the experimental verification analysis, based on biologically inspired computational model, higher clustering purity on both the real dataset and the simulation datasets existence is demonstrated for the proposed algorithm. In other words, the cluster result of MSFS algorithm is advantageous over previous method.*

***Keywords:*** *stream data; clustering analysis; the model of (Multiple Species Flocking on Stream)MSF; cluster purity*

## 1. Introduction

Recently, with advances in communication and data collection techniques, people receive a large number of real time data at very high rates. These fields all receive real time data continuously: sensor networks, network traffic control, and web log monitoring and processing centers. In data mining area, there are many techniques but they should be tuned and changed to work in data stream mining. The data stream mining is different from the regular static data mining. The difference can be described as Table 1.

**Table 1. Traditional Data Mining and Stream Data Mining**

| project | Regular data mining | Stream data mining |
|---|---|---|
| Times of data scanning | Many times | Only one |
| Processing time | Unlimited | Extremely limited |
| Memory space | Unlimited | Extremely limited |
| Result | Accurate | Approximate |

These distinguishing features bring new challenge to stream data processing. It also becomes one of the hot issues in the area of data mining, how to dig out the information is of interest. Clustering analysis is an important technology for data mining, and many researchers pay their attention to the clustering of stream data [1].

In this paper, MSFS algorithm is proposed. It combines MSF model and the DenStream clustering algorithm that is based on density. MSF model is a kind of swarm intelligence model for text clustering, and we make use of the feature similarity rule to make MSFS suitable for data stream clustering.

This paper is organized as follows. The second section describes the related work with the proposed algorithm, *e.g.*, the DenStream algorithm and the MSF (Multiple Species Flocking) model. Section 3 describes the proposed algorithm. In Section 4, the results of

ISSN: 2005-4270 IJDTA

the method on synthetic and real life data sets are presented. Finally, we discuss the advantages of the approach and conclude this article.

## 2. Related Work

In recently years, many special attentions have been paid towards searching efficient and efficacious methods for clustering data streams.

In 2000, Guha *et al.*, proposed a data stream clustering algorithm based on k-means [2]. Callaghan *et al.*, proposed an algorithm for real-time data streams called STREAM [3]. STREAM employs the ideology of dividing and conquering and the local search technology for the multi-level clustering so that the performance and clustering effect has been greatly improved. The approach based on partition represents the early stages of real-time data stream clustering research, and these methods can reflect the characteristics of the data stream in terms of creating memory and maintaining summary data structure. But there also exist some common problems such as producing spherical cluster, noise-sensitive character and disability to analyze the evolution. In 2003, CluStream was proposed in [4]. It treats the data stream clustering as a dynamic process changing by the time series. And in next year, HPStream was proposed [5].

RDF-CluStream is another data stream clustering algorithm which is based on the relative density. It employs the grid method to deal with the data stream. However, there also exist some problems. When the grid is very finely divided, it requires larger memory space instead, if the grid dividing is not fine, the clustering accuracy may be affected.

TS-Stream is a clustering algorithm based on decision tree [6]. It uses the time series generating functions to extract features and produce partitions in better accordance. The experiment results confirm that TS-Stream can help in creating a more diversified investment portfolio, maximizing gains.

EMicro algorithm is put forward in 2010 [7]. This algorithm not only takes into account the distance between tuples and the attribute-level uncertainty of data tuples, but also emphasizes tuples' own characteristics - the existence-level uncertainty. Compared with the previous uncertain data stream clustering algorithms, EMicro algorithm has a lot of improvements and performs well for data streams.

Cao et al. raised a density-based clustering algorithm called DenStream for evolving data streams that captures synopsis information about the nature of the data stream by using summary statistics [8]. The clustering process is divided into online clustering and offline clustering such like CluStream. In online clustering part, if the density of a cluster is greater than a certain threshold, the algorithm will think of the cluster as potential micro-clusters (p-micro-cluster).On the contrary, the cluster will be treated as an outlier micro-cluster (o-micro-cluster). In the offline part, when the query request arrives, it will deal with the p-micro-cluster and the o-micro-cluster. Then the result will be output. The process of offline part essentially follows the methods of DBSCAN [9].

## 3. MSF Model Introduction

In this paper, MSF model is based on a Flocking clustering algorithm, and Flocking model is a bionic model. Flocking model [10] was developed by Reynolds and others through the study of birds, group behaviors; it can also be seen as the prototype of PSO proposed in 1995 [11]. Cui studied the Flocking model and propose a MSF model that has been applied to text clustering [12].

But in FlockStream algorithm, the authors Agostino Forestiero et al have also proposed a rule that does not refer to the rule modified by a fourth principle [13].

Assuming that $R^d$ is the D-dimensional feature space of the data point in the data stream, $R_v^2$ is the two-dimensional Cartesian virtual space, we will deploy the agent to this space and make it move according to the rules of flocking model.

 

Here the virtual space is assumed to be discrete and not continuous, and is implemented as a fixed-size two-dimensional grid, and in each cell of the grid, only one proxy agent can be deployed and the position is determined $P = (x, y)$. Each data point $p = (x_1, x_2, ..., x_d)$ in $R^d$ is associated with the A proxy of the virtual space $R_v^2$. Here agent $A = (P, \vec{v})$, $P = (x, y)$ is the position in virtual space, $\vec{V} = (m, \theta)$ which means the velocity vector of the agent (*m* is the middleweight; $\theta$ is the angle between the positive x-axle). We assume that the middleweight of all the boid is 1, which means that each boid in virtual space can move only one unit at one time.

Assuming that *pc* is the data points of feature space, and *Ac* is its agent in the virtual space. *R1* is the region radius of *Ac* is virtual space, *dv* is the Euclidean distance of agent in $R_v^2$, dist represents the Euclidean distance of data point in the feature space, $\varepsilon$ is the maximum threshold. If $F_1$, $F_2$, ... $F_n$ are in its visual range , $d_v(A_c, F_i) \le R_1$ is true. If the data point *pi* corresponds with the agent *Fi* , and $dist\ (p_i, p_c) \le \varepsilon$ , then we say that the agent *Ac* is similar to $F_i$.

According to the calibration principle, velocity vector of the current agent Ac can be calculated as the average of all agents speeds within its visible area. Here calibration rules can be described as follows:

for $i \in \{1,2,... n\}$ ,if $dist\ (p_i, p_c) \le \varepsilon \wedge d_v(A_c, F_i) \le R_1 \wedge d_v(A_c, F_i) \ge R_2 \Rightarrow$

$$var = \frac{1}{n}\sum_{n=1} v_i$$

Cohesion principle enables the agent to move to the similar boid within the visible area, let *Pc* and *Pi* be the location of *Ac* and its nearby agent *Fi*, where *i = 1, ..., n*, then the cohesion vector can be expressed as the following formula, *Cnb* represents the center of all agents within the visible area of agent *Ac*.

For $i \in \{1, 2...n\}$, if $dist(p_i, p_c) \le \varepsilon \wedge d_v(A_c, F_i) \le R_1 \wedge d_v(A_c, F_i) \ge R_2 \Rightarrow$

$$\vec{v}_{cr} = \overrightarrow{C_{nb} P_c}$$

Separation rules avoid *Ac* agent moving towards the location of the agents dissimilar to it. If *Cdb* represents the center of all the agents within its visible range, $\overrightarrow{C_{db} P_c}$ represents the vector of line from *Pc* to *Cdb*, the direction of separating speed is opposite to this direction. When there are two groups of agents too close, the priority of separation rules will be higher than the rest of the rules, until the distance between them reaching *R2* or more. Separation vector can be expressed by the following reasoning:

for $i \in \{1,2,... n\}$ ,if $dist\ (p_i, p_c) > \varepsilon \vee d_v(\overrightarrow{A_c, F_i}) \le R_2 \Rightarrow$

$$\vec{v}_{sr} = \overrightarrow{C_{db} P_c}$$

The group behavior vector of each agent *Ac* can be combined linearly through these three principles:

$$\vec{v}_A = \vec{v}_{ar} + \vec{v}_{cr} + \vec{v}_{sr}$$

The Advantages of MSF model algorithm is based on the principle of heuristic search mechanism, heuristic search can help boid in group to form a small group rapidly. Each boid is constantly moving in the virtual space $R_v^2$, and they are looking for boids similar to them to form a group, when add new boid or delete boid at runtime, new clustering results will be quickly produced. And these characters can be used in stream clustering. Therefore, this paper combines the MSF model with DenStream algorithm to propose MSFS algorithm.

# 4. MSFS Algorithm

MSFS algorithm references the model of MSF rules in FlockingStream on the basis of DenStream algorithm. In this article, four types of agents are aroused out: data agents, p micro-cluster agent, o micro-cluster agent, c clustering agent.

## 4.1. Related Concepts

In this algorithm, in addition to the use of the rules of MSF model, taking the difference of agent models into account, we expanded four different agent models: data agent (on behalf of data points), p micro-cluster Agent (on behalf of the potential core of micro-clusters, that is, potential c-micro-cluster), o micro-cluster agent (on behalf of outlier micro-cluster), c clustering agent (representative of the final cluster). During the execution of the algorithm, according to the relevant constraints, change agent type, respond clustering request, generate clustering results.

Agent *p*, that is, the agent who meets the relative definition of *P* micro clusters the concepts $\{\overrightarrow{CF^1}, \overrightarrow{CF^2}, \omega\}$ mentioned in DenStream, Agent *o* is empathy defined $\{\overrightarrow{CF^1}, \overrightarrow{CF^2}, \omega, t_0\}$. Data agent is on behalf of the agent of data point, we define *C* clustering agent as the clustering results generated after responding the request.

During the initialization of the algorithm, each multidimensional data point is associated with one data gent; then, randomly deploy the agents which meet the data collection to two-dimensional virtual grid. The location of each agent *A=(P,* $\vec{v}$ *)* in the grid is randomly generated, and its velocity vector is defined as $\vec{v} = (m, \theta)$, init *m* as 1 and $\theta \in [0, 2\pi]$. After the parameters of data agent are predefined, data agent will move according to MSF rules.

## 4.2. The Specific Process of the Algorithm

After the initial definition, coming to the phases of maintenance and clustering, a piece of data points and data stream associated with some agents has already flow into virtual space at a flow rate, we fix a maximum number of iteration performed to maintain the p agents and o agents, finish the clustering process through generating c agents. The specific process can be represented by the following algorithm.

MSFS ( *DS, ε, β,μ,λ*){

For *i*=1,2,3……Max(iteration){

Init();

AgentsMerging();

$T_p = \left\lceil \frac{1}{\lambda} \log\left(\frac{\beta\mu}{\beta\mu - 1}\right) \right\rceil$;

If (*t* mod *Tp*==0) {

For each p-agents

If ( $\omega_p < \beta\mu$ )

Change p-agents to o-agents;

$$\xi = \frac{2^{-\lambda(t - t_o + t_p)} - 1}{2^{-\lambda Tp} - 1}$$

For each o-agents{

 

If ( $\omega_o > \beta\mu$ )

Change o-agents to p-agents;

Else if ( $\omega o < \xi$ )

Delete the cluster *Co* that o-agent represents;}}}

If a request of a clustering arrives

Return the cluster that c-agent represents;

}

AgentsMerging(){

For each agent *A* {

If agent *A* is a data agent, another agent *B* is in *A'* neighbor area{

If *B* is a data agent and $dist\ (P_A, P_B) \le \varepsilon$

Join *A* and *B* to form an o-agent;

Else (*B* is a p-agent or an o-agent) {

{Compute the new radius $r_p$ (or $r_o$ ) of $C_p^B$ (or $C_o^B$ );

If $r_p < \varepsilon$ ( or $r_o < \varepsilon$ )

*A* is merged with *B*; } }

}

Else the agent *A* is a p-agent or an o-agent, another agent *B* is in *A*' neighbor area{

If *B* is a data agent and $dist\ (c_p^A, p_B) \le \varepsilon$ (or $dist\ (c_o^A, p_B) \le \varepsilon$ )

*B* is joined with *A*;

Else *B* is another p-agent or o-agent and $dist\ (c_p^A, c_p^B) \le \varepsilon$ (or $dist\ (c_o^A, c_o^B) \le \varepsilon$ )

*A* is merged with *B* to form c-agent;

c-agent is transformed into a cluster of similar agents;

}

Compute the velocity vector of agent *A* by applying the MSF rules;

Move the agent;}

}

The related interpretations of AgentsMerging () algorithms are as follows:

(1) When a data agent *A* on behalf of data *PA* comes across another data agent *B* on behalf of *PB*, if it satisfies $dist\ (P_A, P_B) \le \varepsilon$ , that is, the Euclidean distance between them is less than or equal $\varepsilon$, then *A* and *B* are combined into one o-agent.

(2) When a data agent *A* comes across a p agent *B* on behalf of micro-cluster $C_p^B$ (or an o agent on behalf of micro-cluster $C_o^B$ ), if the radius of the new micro cluster generated by *A* and *B* is less than or equal to $\varepsilon$, then *A* combines with *B*.

(3) If *A* is not a data agent, but a *p* or *o* agent or agency, when it encounters another *P* or *O* agent, if the distance between the corresponding micro-clusters is less than $\varepsilon$, then we can merge them into clustering agent *C* which has certain similarity.

(4) If a *P* or *O* agent comes across a data agent *B*, the same to (2), analyze if agent *B* can be combined with *A*.

(5)Finally, once having done a merge operation, velocity vector of the agent will be calculated according to MSF rules, and then the agent will be adjusted according to four principles.

### 4.3. Performance Analysis

MSFS algorithm proposed in this paper uses a local multi-agency random searching strategy; each agent maintains relative independence with other agents, the exchange of information between agents merely relies on an asynchronous manner with neighbors for instant communication. It is the dispersion and asynchronous nature of MSFS algorithm that makes it can be better used to cluster large data sets.

MSFS algorithm takes MSF model, as described in Chapter III, each agent has a fixed rate, there is a minimum distance between one agent and others, and agents update their velocity direction according to the principle of the calibration to keep accordance with other agents, the position and velocity of the agents who are similar to each other will be clustered to one value by cohesion principle, and avoid conflict between one agent and others by the principle of separation. The initialization time complexity of MSF model is $O(n^2)$, where n represents the number of using agents in virtual space.

In the actual process, each agent needs to calculate the degree of similarity to compare with other agents, as a basis to decide whether they belong to the same cluster, which may need to perform a certain number of comparisons, but we give each agency a fixed spatial position and visible range, so the agent only needs to access the members within the area and rapidly gathers into categories, so the time of instructions of comparing calculation is very low in the algorithm. In the next chapter, we will verify the algorithm by experiments

## 5. Experimental Results

We employ Java to achieve MSFS algorithm's experimental result. And the computer configuration parameters are like this: the processor is Intel (R) core i3-2120, operating system is Windows 7, and the system memory is 4.00GB.

The experiment is divided into two parts: real data sets and synthetic data sets with some noise data. Real data set is called as KDD CUP99 which is used in KDD (Knowledge Discovery) contest in 1999. It is always employed to analyze the real-time detection of computer attacks in the stream of data clustering mining areas.

In the experiment, we take the advantage of the average purity (purity) to compare clustering quality of clustering algorithm clusters. The clustering purity is defined as follows:

$$purity = \frac{\sum_{i=1}^{K} \frac{\left|C_i^d\right|}{\left|C_i\right|}}{K} \times 100\ \%$$

Where *K* denotes the number of clusters, $\left|C_i^d\right|$ indicates the number of points with the dominant class label in cluster *i*, and $\left|C_i\right|$ indicates the number of points in cluster *i*.

 

### 5.1. Real Data Sets

Experimental data shows that MSFS' clustering purity is always better than DenStream on the network intrusion dataset-KDD Cup99. The results are shown as Figure 1 and Figure 2.

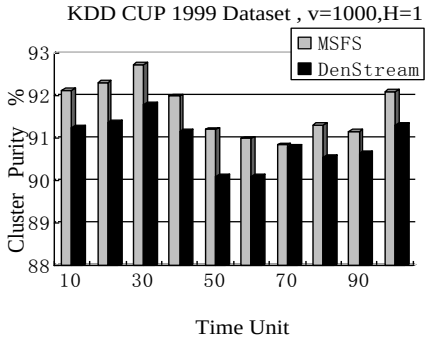


**Figure 1. The Cluster Purity of MSFS and DenStream with H=1**

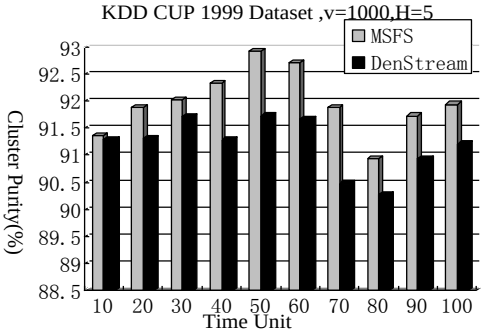


**Figure 2. Comparison between MSFS and DenStream with H=5**

### 5.2. Synthetic Data Sets

In this paper, three artificial datasets DS1, DS2, DS3 are selected for more equitable comparing. New evolutionary data sets, EDS is produced by the method of random selection. In real applications, some unavoidable noise data is generated due to some unexpected reasons. Therefore, we added 5% noise data to the EDS and observed experimental results. The Figure 3 shows the experimental results.

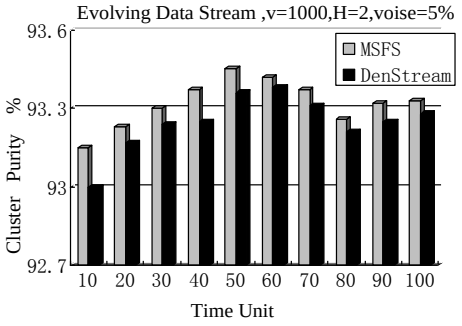


**Figure 3. The Experiment Result on EDS with noise=5%**

As shown in the figure, MSFS's clustering purity performs significantly better than DenStream algorithm.

## 6. Discussion and Conclusions

MSFS can produce better clustering effect than DenStream algorithm in experimental comparison. When experiments is performed based on real data sets, MSFS algorithm achieves higher clustering purity. What's more, MSFS algorithm is more outstanding when it deals with the data whith some noise. However, because the parameters are pre-defined, proposed algorithm has high parameter sensitivity. In the future, this issue will be concerned and its solution is going to be proposed.

## Acknowledgments

This work is supported by the Heilongjiang Provincial Department of Education Science Research Project (No. 12541239).

## References

[1] S. Ding, F. Wu, J. Qian, H. Jia and F. Jin, "Research on data stream clustering algorithms", Journal of Theoretical and Applied Information Technology, vol. 44, no. 2, **(2012)**.
[2] S. Guha, A. Meyerson and N. Mishra, "Clustering data streams", Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Washington, USA, **(2000)** November 12-14.
[3] L. O'Callaghan, "Streaming data algorithms for high quality clustering", Proceedings of the 18th International Conference on Data Engineering, San Jose, USA, **(2002)** February 26-March 1.
[4] C. C. Aggarwal, J. Han, J. Wang and P. Yu, "A framework for clustering evolving data streams", Proceedings of the 29th international conference on Very large data bases; Berlin, Germany, **(2003)** September 9-12.
[5] C. C. Aggarwal, J. Han, J. Wang and P. S. Yu, "A framework for projected clustering of high dimensional datastreams", Proceedings of the 30th international conference on very large data bases, Toronto, Canada, **(2004)** August 31-September 3.
[6] C. M. M. Pereira and R. F. de Mello, "TS-stream: clustering time series on data streams", Journal of Intelligent Information Systems, vol. 42, no. 1, **(2014)**.
[7] C. Zhang, C. Jin and A. Zhou, "Clustering algorithm over uncertain data stream", Journal of Software, (in Chinese), vol. 21, no. 9, **(2010)**.
[8] F. Cao, M. Ester, W. Qian and A. Zhou, "Density-based clustering over evolving data stream with noise", Proceedings of the sixth SIAM international conference on data mining, **(2006)** Apr 20-22; Bethesda, USA.
[9] M. Ester, H.-P. Kriegel, S. Jrg and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", Proceedings of the second ACM SIGKDD international conference on knowledge discovery and data mining, Portland, USA, **(1996)**.
[10] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioral model", Proceedings of the 14th annual conference on computer graphics and interactive techniques, **(1987)**.
[11] J. Kennedy and R. C. Eberhart, "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, **(1995)** November 27-December 1.
[12] X. Cui and T. E. Potok, "A distributed agent implementation of multiple species flocking model for document partitioning clustering", Proceedings of 10th International Workshop cooperative Information Agents, Edinburgh, UK, **(2006)** September 11-13.
[13] A. Forestiero, C. Pizzuti and G. Spezzano, "A single pass algorithm for clustering evolving data streams based on swarm intelligence", Journal of Data Mining and Knowledge Discovery, vol. 26, no. 1, **(2013)**.

## Author

**Yingmei Li**, received the M. E. degree from Harbin Engineering University, Harbin, She is currently a professor with the Department computer science and information engineering, Harbin Normal University, Harbin, China. Her current research interests include database, software engineering.

 