Weighted Numerical and Categorical Attribute Clustering in Data Streams

Wen-Bin Liang, Chang-Dong Wang* and Jian-Huang Lai
School of Data and Computer Science, Sun Yat-sen University, Guangzhou, P. R. China.
Guangdong Key Laboratory of Information Security Technology, Guangzhou, China.
Email: wenbin_liang@foxmail.com, changdongwang@hotmail.com, stsljh@mail.sysu.edu.cn

***Abstract*—In recent years, combining multiple attributes of data sets including numerical and categorical attributes for data stream clustering has been a popular practice for improving clustering accuracy. In this paper, we propose a novel data stream clustering algorithm called WKStream which can cluster numerical and categorical data stream with different shapes efficiently. As different attributes have different significances in clustering process, different weights should be assigned for them. Accordingly, we design a new concept called $Nonuniformity$ to assign weight to numerical attributes; While for the categorical attribute, we design a novel method to measure the distance between two values in the same categorical attribute. To discover clusters of different shapes, the concepts of micro-cluster and macro-cluster are utilized and extended to model complex cluster structures. Experimental results conducted on numerical, categorical and mixture real-world data sets have validated the effectiveness of the proposed method.**

I. Introduction

In the past few years, a large volume of data has been generated, which contain lots of valuable information. Therefore analyzing the large volume of data has been a hot and significant topic [1] [2] [3] [4]. The techniques of clustering data stream in numerical attributes have been studied for a relative long time, such as CluStream and SVStream. As for the categorical domain, a clustering model designed by Bai et al. [2] uses the rough set theory to cluster categorical data. But the data stream in real world usually contains both numerical and categorical attributes. In this paper we propose an effective and efficient data stream clustering algorithm handling both numerical and categorical mixture data.

A. Related Work

In the numerical domain, the CluStream framework [1] is one representative method. Its main process contains online component and offline component. The online component is used for real-time clustering and the offline component is used to construct data stream summary statics. SVStream [3] is another representative method for numerical domain, which is based on support vector domain description (SVDD) [5] and support vector clustering (SVC) [6]. It does well in the case of different shapes of data stream and evolving structure. But it is time-consuming in calculating Lagrangian multiplier.

*Corresponding author

In the categorical domain, the traditional way to measure distance in categorical attributes is Jaccard coefficient, which is used in k-modes [7]. In the k-modes method, it regards the distance between each pair of values same, which can not reflect the real-world task. For example, in a color attribute, the domain is $\{faint\ yellow,\ bright\ yellow,\ crimson,\ light\ red\}$. The difference between $faint\ yellow$ and $crimson$ is larger than $faint\ yellow$ and $bright\ yellow$. To address this problem, recently, Bai et al. [2] [8] used the rough set theory to measure distance in categorical values.

B. Our Work

In this paper, we present a novel data stream clustering algorithm which considers both numerical and categorical attributes. We first design a self-adaptive method to assign weight to numerical attributes. Secondly we design a way to measure the difference between two categorical values in the same attribute. Based on these, a formula is designed to calculate the distance between two objects. And then we utilize the concepts of micro-cluster and macro-cluster and design the formulas to calculate the distance between two macro (micro)-clusters. Based on above main ideas, we propose a data stream algorithm for numerical and categorical mixture data called WKStream. The main algorithm process is summarized as follows. Firstly, user inputs a fixed integer k. We map k into a much larger number k'. And we develop an algorithm, Weighted k-means+ (Wk-means+) to get k' micro-clusters in the first data chunk, and then we merge the nearest micro-clusters to get k macro-clusters so as to construct clusters of different shapes approximately. In the following data chunks, we will cluster k' micro-clusters, which will be absorbed by the nearest macro-cluster. In the current chunk, the micro-clusters in the same macro-cluster will be regarded as in the same cluster as the corresponding macro-cluster. In this way, the clustering result for the current chunk can be obtained.

The rest of this paper will be organized as follows. In Section II, we will describe in detail the proposed method, including some new concepts and the overall algorithm. In Section III, experiments will be conducted to confirm the effectiveness of the proposed method, which cover three kinds of data stream clustering, namely pure numerical, pure categorical, and numerical and categorical mixture data. And we will report the comparison result with SVStream [3],

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

CluStream [1] and Bai's [2] algorithm. Finally, Section IV draws the conclusion of this paper.

II. The Proposed Algorithm

A. Distance between Two Objects

In this paper, we regard the data stream arrives by chunk, each of which has s objects. The t-th chunk is denoted by

$$\mathcal{X}^t = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \mathbf{x}_3^t, \ldots, \mathbf{x}_s^t\}. \tag{1}$$

The attributes are $\mathcal{A} = \{A_1, A_2 ... A_r, A_{r+1}, \ldots, A_d\}$, where $A_1, A_2, \ldots, A_r$ are numerical attributes and $A_{r+1}, \ldots, A_d$ are categorical attributes.

Definition 1 (Object distance): The distance of two objects $\mathbf{x}$ and $\mathbf{y}$ is defined by combining the distance from the numerical attributes and that from the categorical attributes as follows

$$D(\mathbf{x}, \mathbf{y}) = (1-\lambda) D_1(\mathbf{x}, \mathbf{y}) + \lambda D_2(\mathbf{x}, \mathbf{y}) \tag{2}$$

where λ is a parameter used to balance the weight between numerical distance and categorical distance. $D_1(\mathbf{x}, \mathbf{y})$ is the distance of numerical data and $D_2(\mathbf{x}, \mathbf{y})$ is the distance of categorical data, which are defined respectively as follows

$$D_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{r} \omega_i (x_i - y_i)^2 \tag{3}$$

$$D_2(\mathbf{x}, \mathbf{y}) = \sum_{i=r+1}^{d} D_{A_i}(x_i, y_i) \tag{4}$$

where x_i and y_i are the i-th attribute of data points $\mathbf{x}$ and $\mathbf{y}$ respectively. The weights ω_i in $D_1(\mathbf{x}, \mathbf{y})$ are determined based on the *Nonuniformity*, which measures how similar is between distribution of data and the uniform distribution. And $D_{A_i}(x_i, y_i)$ is the distance between categorical label x_i and y_i in attribute A_i. In what follows, we will describe in detail how to calculate ω_i and $D_{A_i}(x_i, y_i)$.

1) Numerical Weight:

Definition 2 (Nonuniformity): Given a one-dimension data set, which is arranged in the ascending order and denoted as follows,

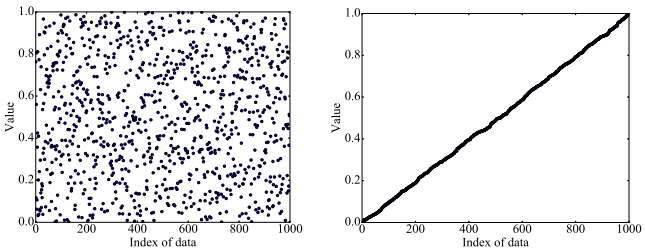
$$X = \{x_1, x_2, \ldots, x_n\} \tag{5}$$

for each $i > j$, $x_i \geq x_j$. The *Nonuniformity* of this dimension is defined as

$$Nonuniformity = \sum_{i=1}^{n} \left(x_i - \left(x_1 + (i-1)\frac{x_n - x_1}{n-1} \right) \right)^2 \tag{6}$$

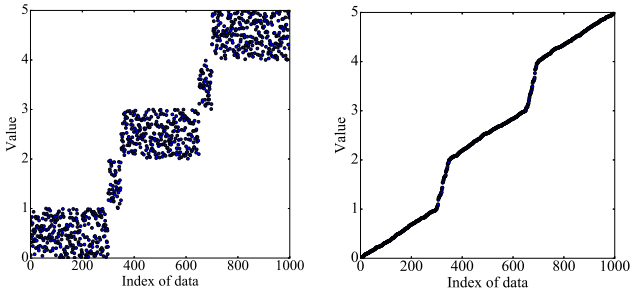
We will construct an arithmetic progression to show the meaning of *Nonuniformity*. We pick the smallest element x_1 and the largest element x_n to construct an arithmetic progression with n elements. This arithmetic progression is $\{x_1, x_1 + 1 \times \frac{x_n - x_1}{n-1}, \ldots, x_1 + (i-1) \times \frac{x_n - x_1}{n-1}, \ldots, x_n\}$. *Nonuniformity* measures the distance between the origin sequence and the arithmetic progression above.

In the traditional k-means or k-modes algorithms, attributes are regarded as the same weight in calculating distance (e.g. Euclidean distance formula). But in real applications, different attributes have different significances [9]. So in this part, we design a way to measure the weight of attributes. If the distribution of values in the same attribute is similar to the uniform distribution, this attribute has little discriminability for the objects, because uniform distribution is random in an interval and it can't show the boundaries among clusters. When we calculate the distance between two objects, we will assign a lower weight to this attribute as will be shown soon.



(a) Unsorted (b) Sorted

Fig. 1: Illustration of *Nonuniformity*: 1000 random values in $(0, 1)$, unsorted and sorted.



(a) Unsorted (b) Sorted

Fig. 2: Illustration of *Nonuniformity*: 1000 random values in three intervals, unsorted and sorted.

In Definition 2, we regard the uniform distribution sequence as an arithmetic progression approximately. To show this approximation, we generate 1000 random values in $(0, 1)$, as shown in Fig. 1(a). After sorting the random value, as shown in Fig. 1(b), we can find the values and indexes of data are linearly dependent, which illustrates that the uniform distribution data can be regarded arithmetic progression approximately. So *Nonuniformity* measures the dissimilarity between the arithmetic progression and the given data attribute.

To further demonstrate how the distribution of data attribute affects the discriminability of data objects, we use the second data set, which contains random values in three intervals, namely $(0, 1)$, $(2, 3)$ and $(4, 5)$, as shown in Fig. 2 and we add some noise among their boundaries. By using Eq. (6), we can get the *Nonuniformity* of the two datasets shown in Fig. 1(a) and Fig. 2(a), which are 2.77 and 24.71 respectively. With lower *Nonuniformity*, it is more difficult to find the boundaries among clusters as compared in Fig. 1 and Fig. 2. So in clustering process we will assign lower weights to attributes

TABLE I: Illustration of categorical distance: Proportion of Each Hotel Class.

Hotel types	Business hotel	Resort hotel	Budget hostel
A	65%	15%	20%
B	60%	20%	20%
C	30%	10%	50%
D	30%	5%	65%

TABLE II: Illustration of categorical distance: Difference between Areas.

	A	B	C	D
A	0	0.1	0.7	0.9
B	0.1	0	0.7	0.9
C	0.7	0.7	0	0.2
D	0.9	0.9	0.2	0

with lower $Nonuniformity$. Based on the above analysis, we assign the weight for the i-th numerical attribute as

$$\omega_i = \frac{N_i}{\sum_{j=1}^{r} N_j} \tag{7}$$

where N_i denotes the $Nonuniformity$ of the i-the numerical attribute.

2) Categorical Distance: In what follows, we will talk about how to measure the distance for categorical attributes.

For a data set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, assume that we are given the clustering label for each data point, i.e. $\{z_1, z_2, \ldots, z_n\}$, where

$$z_i \in \mathcal{Z},\ \mathcal{Z} = \{1, 2, \ldots, k\} \tag{8}$$

We denote the u-th value of categorical attribute A_i by a_{iu}. We construct a set S_{iu}, which contains all the elements in attriute A_i with value a_{iu}. We denote the number of elements in S_{iu} with cluster label z_j by $count(z_j, S_{iu})$. $D_{A_i}(a_{iu}, a_{iv})$ is calculate by the following

$$D_{A_i}(a_{iu}, a_{iv}) = \sum_{z_j \in \mathcal{Z}} \left| \frac{count(z_j, S_{iu})}{|S_{iu}|} - \frac{count(z_j, S_{iv})}{|S_{iv}|} \right| \tag{9}$$

To further demonstrate the meaning of $D_{A_i}(a_{iu}, a_{iv})$, we will use an example of hotel classification. Suppose that there are four area namely {A,B,C,D} in a city. A and B are in city center. C and D are in suburb of the city. A hotel object has many attributes and the location area is one of the categorical attributes. We divide all the hotels into three classes, namely business hotel, resort hotel and budget hostel. Table I shows the proportion of each class in each area. By using Eq. (9), we can get the distance table, as shown in Table I. We can find that the differences of the same kind area are smaller than that of the distinct areas. For example, the difference of A and B is smaller than the difference of A and C, because the distribution of hotel class of A and B is similar. So Eq. (9) measures the difference of two categorical distributions.

We can use a distance table like Table II to record the distance between two values in an attribute. In the first chunk, we will assign the same values for them. Then we will cluster the data in the first chunk. Based on the first clustering result, we use Eq. (9) and all the historical clustering results to update the distance table of each attribute. Then we use the new distance table for the second chunk data clustering. We repeat these steps until no new data chunks arrive.

B. Distance between Two Micro-Clusters

In our algorithm, we design a way to construct complex cluster structures. In particular, a cluster is a set of some micro-clusters defined as follows.

Definition 3 (Micro-cluster): A micro-cluster MiC is a triple

$$MiC = \{\mathbf{c}, \bar{r}, p\} \tag{10}$$

where $\mathbf{c}$ is the center of cluster, $\bar{r}$ is the radius of cluster and p is the number of the points in this micro-cluster.

We will use an example to show how to calculate a micro-cluster. Suppose $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$ is a set of objects, which are in the same micro-cluster, MiC. Object $\mathbf{x}_i = [x_{i1}\ x_{i2}\ \ldots\ x_{ir}\ x_{i(r+1)}\ x_{i(r+2)}\ \ldots\ x_{id}]$, where the first r attributes are numerical and the rest are categorical attributes. The cluster center is $\mathbf{c} = [c_1\ c_2\ \ldots\ c_r\ c_{r+1}\ldots\ c_d]$ with c_j defined as

$$c_j = \begin{cases} \frac{\sum_{i=1}^{p} x_{ij}}{p} & 1 \leq j \leq r \\ x_{aj} & r+1 \leq j \leq d \end{cases} \tag{11}$$

where x_{aj} is the j-th attribute appearing the most times in all the p objects. Using Eq. (2), we can get the distance between each object and the center. And $\bar{r}$ is calculated by

$$\bar{r} = \frac{\sum_{i=1}^{p} D(\mathbf{c}, \mathbf{x}_i)}{p} \tag{12}$$

The distance between micro-cluster can be defined as follows.

Definition 4 (Distance between micro-clusters): Given two micro-clusters, $MiC_1 = \{\mathbf{c}_1, \bar{r}_1, p_1\}$ and $MiC_2 = \{\mathbf{c}_2, \bar{r}_2, p_2\}$, the distance between these micro-clusters is defined by the following

$$D_{MiC} = \frac{D(\mathbf{c}_1, \mathbf{c}_2)}{(\bar{r}_1 + \bar{r}_2)} \tag{13}$$

where $D(\mathbf{c}_1, \mathbf{c}_2)$ denotes the distance between two cluster centers.

C. Distance between Macro-Cluster and Micro-Cluster

Based on the micro-cluster, the macro-cluster can be defined as follows.

Definition 5 (Macro-cluster): A macro-cluster is a set of micro-cluster.

$$MaC = \{MiC_1, MiC_2, \ldots, MiC_q\} \tag{14}$$

where q is the number of micro-clusters in this macro-cluster.

Definition 6 (Distance between macro-cluster and micro-cluster): Given a macro-cluster MaC and a micro-cluster MiC, the distance between them is given by the following

$$D_{AI}(MaC, MiC) = \min_{\forall MiC_i \in MaC} \{D_{MiC}(MiC_i, MiC)\} \tag{15}$$

where D_{MiC} is given by Eq. (13).
In fact, a micro-cluster can be regarded as a special macro-cluster which contains only one micro-cluster. So based on this, we can get the distance between macro-cluster and micro-cluster accordingly.

D. The Overall Algorithm

Based on the above defined concepts, this subsection will describe the overall algorithm.

1) Seeds Generation: In k-means, user should input the number of clusters k. In our algorithm, we need to specify the number of seeds at first, which is denoted as k' and calculated by the following

$$k' = f(k) = 2k. \tag{16}$$

At the first step of k-means, we should generate k' seeds. But in this way, the clustering result will be different based on different seeds. So we use the step in k-means++ [10] to select cluster seeds, which is summarized in Algorithm 1. In this section, we will use distances defined in the previous sections.

2) Weighted k-means+: For each chunk, we will get s data objects. First we create k' seeds by use the seed generation algorithm described above. Then we will calculate $Nonuniformity$ of each attribute for assigning weights in the next step. In particular, a damping factor α is introduced in updating the weights ω in each chunk. For one thing, we want to update the weight by using the lastest chunk information. For another, we want to prevent the weight changing a lot and make the weight keep the feature of the past chunks because the data in the past chunks contains more information than the current chunk. So we usually set the damping factor α more than 0.5. In our experiments, α is set to be 0.75. For each data object, we calculate distance between the data objects and each center (in the first step, centers are seeds) by using Eq. (2) and assign it to the nearest center. And then we update the centers. We will repeat these two steps until convergence. After that we can get k' micro-clusters. The main procedure is summarized in Algorithm 2.

3) Streaming Clustering: In the initial stage, we will fill the distance table for each categorical attribute by value 1. And In the initial stage, we can get k macro-clusters (some macro-clusters may have only one micro-cluster) in the first chunk. When a new data chunk arrives, we will use the Weighted k-means+ process to produce k' micro-clusters. For each micro-cluster, we will assign it to the nearest macro-cluster we have obtained previously. So we divide the objects in this chunk into k macro-clusters. And we regard the objects in the same macro-cluster as objects in the same cluster. Then we will merge the two macro-clusters with the distance lower than a threshold, m. The overall procedure is summarized in Algorithm 3.

Algorithm 1 Generate_Seeds(k',D)

Input: k': the number of seeds, D: data set
Output: SS: the set of seeds
Select a object $\mathbf{x}_1$ in D randomly
$SS \leftarrow \{\mathbf{x}_1\}$
repeat
  Calculate $\sum_{a \in SS} D(\mathbf{a}, \mathbf{x}_i)$ for each $x_i \in \bar{SS}$, where $\bar{SS} = D - SS$
  Assign the probability based on the sum distance
  Select the next seed $\mathbf{x}_j$ based on their probability
  $SS \leftarrow SS \cup \{\mathbf{x}_j\}$
until $|SS| = k'$
return SS

Algorithm 2 Weighted_k-means+(k,D,ω)

Input: k: the number of result clusters (micro-clusters), D: data set
$k' = f(k)$
Calculate $Nonuniformity$ of each attribute by Eq. (6)
Calculate $\omega' = [\omega'_1\ \omega'_2\ \ldots\ \omega'_r]$ by Eq. (7)
Update by $\omega = \alpha * \omega + (1-\alpha) * \omega'$
$SS = Generate\_Seeds(k', d)$
repeat
  Assgin each point in D into the nearest centers
  Calculate the k' centers
until Convergence
return Micro-cluster set MIS and weights ω

III. Experiment

In order to show the effectiveness of our proposed method, extensive experiments have been conducted on some large-scale real world data sets from the following three perspectives, namely pure numerical data stream, pure categorical data stream and mixture numerical and categorical data stream. First of all, we analyze the effect of the parameters, including m, the threshold of merging two macro-clusters, and λ, which balances the weight of numerical and categorical attributes. Then comparison experiments have been conducted to compare our algorithm with some existing data stream clustering algorithms, namely SVStream [3], CluStream [1] and categorical data stream clustering algorithm designed by Bai et al [2]. For evaluation metrics, since the ground-truth class labels are available, we use the most widely used measure namely normalized mutual information (NMI) [11]. Given the clustering labels π of c clusters and the actual class labels θ of $\hat{c}$ classes, we construct a confusion matrix where entry (i, j) defines the number $N_i^{(j)}$ of data points in cluster i and class j. Then NMI can be computed by the following equation.

$$NMI = \frac{2\sum_{l=1}^{c}\sum_{h=1}^{\hat{c}} \frac{N_l^{(h)}}{N} \log \frac{N_l^{(h)} N}{\sum_{i=1}^{c} N_i^{(h)} \sum_{i=1}^{\hat{c}} N_l^{(i)}}}{H(\pi) + H(\theta)}, \tag{17}$$

where $H(\pi) = -\sum_{i=1}^{c} \frac{N_i}{N} \log \frac{N_i}{N}$ and $H(\theta) = -\sum_{j=1}^{\hat{c}} \frac{N^(j)}{N} \log \frac{N^(j)}{N}$. A high NMI indicates the clustering and class labels match well.

Algorithm 3 The Proposed WKStream Algorithm

Input: k: the number of result clusters, $DS = X_1, X_2, X_3 \ldots$: data stream
Initialize macro-cluster set $MAS = \varnothing$, $t = 1$
Initialize $\omega = [\omega_1\ \omega_2\ \ldots\ \omega_r] = [\frac{1}{r}\frac{1}{r}\ldots\frac{1}{r}]$ where r is the number of numerical attributes
Initialize the distance table of each categorical attribute by value 1
tmp=Weighted_K-means+(k', X_1, ω)
Create a micro-cluster set MIS=$tmp.MIS$
Update weights $w = tmp.w$
Create a macro-cluster for each micro-cluster in MIS
Add these macro-clusters into MAS, where $|MAS| = |MIS|$
repeat
  Merging two nearest macro-clusters in MAS
until $|MAS| = k$
repeat
  $t \leftarrow t + 1$
  Use Algorithm 2 to get k' micro-clusters
  repeat
    Calculate the distance between a micro-cluster and the nearest macro-cluster in MAS
    if {distance $\geq m$ and $|MAS|$ <MAX_NUMBER_MAC} **then**
      Create a new macro-cluster for it
    else
      Assign it to nearest macro-cluster
    end if
  until All the micro-clusters are handled
  Update the macro-cluster in MAS
  repeat
    Calculate distance between each macro-cluster
    Find the smallest distance
    if (distance $\leq$ threshold) **then**
      Merge the two macro-clusters
    end if
  until No macro-clusters need to be merged
  Assign the same label to the micro-clusters in the same macro-cluster
  Save clustering result and update distance table
until No data chunk arrives

A. Data Description

In our experiments, two testing datasets are used. The first one is KDD-CUP 99 and the other one consists of database operations recorded from a server.

1) KDD-CUP 99: This is a data set obtained from the third International Knowledge Discovery and Data Mining Tools Competition. The data set contains 494,021 records with 34 numerical attributes and 7 categorical. The records are divided into 23 clusters. One of the clusters is normal connection and other 22 clusters are network attack types. This data set will be used in the case of both pure numerical data test and mixture data test.

2) Database Operation Record: Each record contains 27 fields including id, server_ip, client_ip, SQL statement and so on. The SQL statement contains table name and conditions. The operations in this data set can be categorized into 10 types, which are divided by people according to SQL statements. For experimental purpose, we first divide a SQL statement into serval parts, namely operation type, operation object and serval conditions. Then we will map these parts into to integer sets for clustering. This data set will be used in the case of pure categorical data test.

B. Parameter Analysis

The parameter analysis is conducted on the numerical and categorical mixture data set, KDD-CUP 99.

1) The Balancing Parameter λ: The balancing parameter is used to balance the weight of numerical and categorical attributes. The test values are between 0.2 to 0.8 with step 0.1. In Fig. 3 we can find that the best λ is 0.4. When λ is smaller than 0.4, NMI increases with λ increasing. When λ is larger than 0.4, NMI decreases with λ increasing in the most cases. Because in this data set, the number of numerical attributes is as large as approaching five times of that of categorical attributes. So numerical attributes contain more information, and it should have a larger weight.

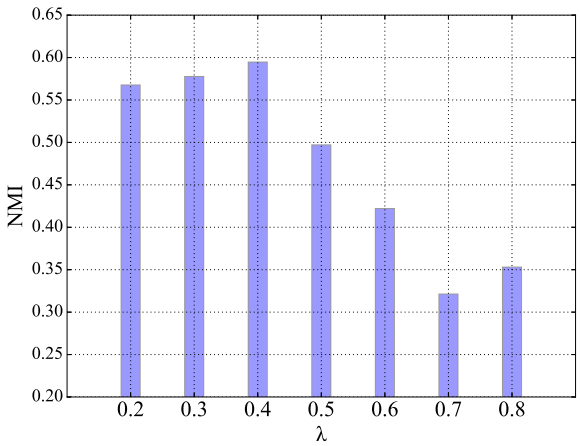


Fig. 3: Parameter analysis: the NMI value as a function of λ on the KDD-CUP 99 dataset.

2) The Threshold of Merging Macro-Cluster: The threshold m is used to judge whether two macro-cluster should be merged. The test values are between 0.25 to 1.5 with step 0.25. In Fig. 4 we can find that the best m is 0.5. If we set m too large, quite different kinds of macro-clusters will be merged. If we set m too small, the macro-clusters in the same kind can't be merged, which would waste the quota of macro-cluster, resulting the case that the new kind of macro-cluster can't be assigned individual macro-cluster index. In our algorithm, if two macro-clusters have been merged, they can't be separated anymore. So we should set m to a smaller value to avoid excessively merging.

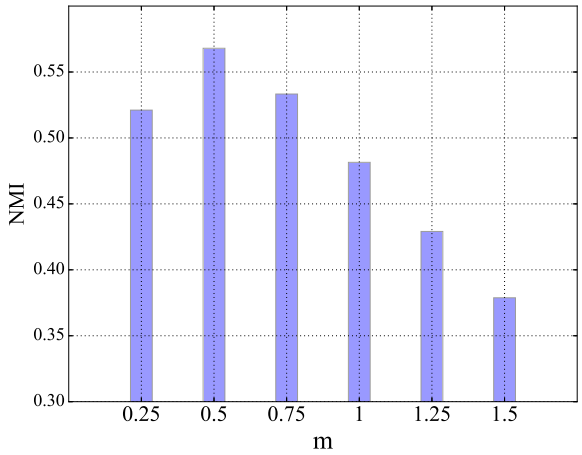


Fig. 4: Parameter analysis: the NMI value as a function of m on the KDD-CUP 99 dataset.

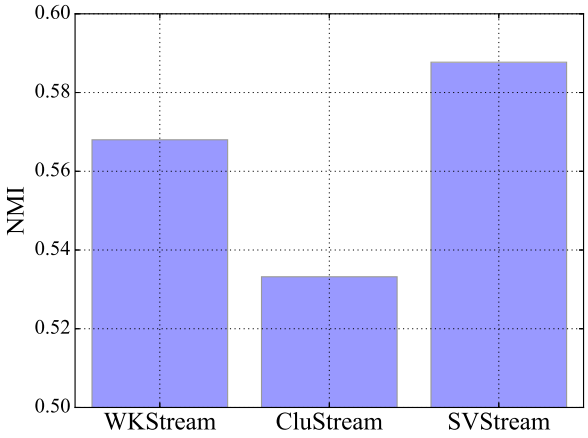


Fig. 5: Comparison on numerical data stream.

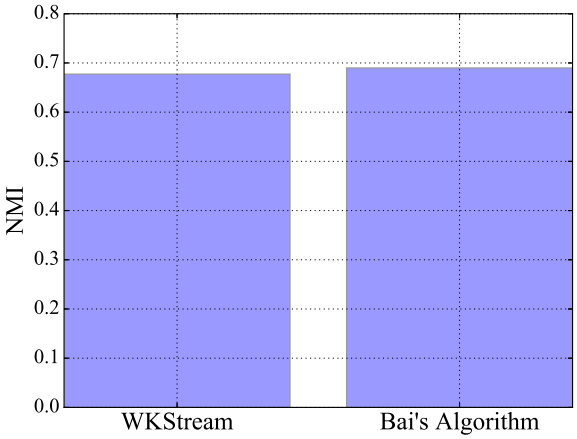


Fig. 6: Comparison on categorical data stream.

C. Comparison Experiments

In this subsection, we compare the performance of our proposed algorithm with other existing algorithms in terms of NMI, namely SVStream [3], CluStream [1] and the algorithm designed by Bai et al. [2]. The comparison are from the aspects, namely pure numerical data and pure categorical data.

1) Numerical Data Comparison: This test is based on the data set, KDD-CUP 99. The results are reported in Fig. 5. We can find SVStream has the best NMI in this test. Our algorithm performs slightly worse than SVStream, but SVStream uses much more time than our algorithm to achieve this. In SVStream process, it can detect and delete the noise points from data stream because it is based on SVDD (support vector domain description), which does well in noise handling. In our algorithm, if a noise point is picked to be a seed, this micro-cluster will be very small because the seed is far from the other points. This will waste an index of micro-cluster, which makes our algorithm generating NMI lower than SVStream. On the other hand, though SVDD is good at handling noise points, it has to use lots of time to solve Lagrangian multiplier. Our algorithm is based on k-means with much lower time cost and has a pretty good performance. Comparing to SVStream, our algorithm is good at handling largescale data stream.

2) Categorical Data Comparison: This test is based on the data set from a server, which consists of only categorical attributes. In Fig. 6 we can find that the performance of our algorithm is similar to that of Bai's algorithm. In Bai's algorithm, it considers drifting concepts [12] in data stream. If the algorithm detects concepts drifting, it will update the cluster model to adapt the new structure in data stream. Our algorithm will use all the clustering history to measure the distance of two values in the same categorical attribute. In the beginning of our algorithm, the history information is not enough, which cause our algorithm makes a little mistakes. With data growing, our algorithm gets more and more information, and it can measure the distance more accurately.

IV. Conclusions

In this paper, we have proposed a novel numerical and categorical data stream clustering algorithm. In this algorithm, an objective function is proposed which simultaneously considers the numerical and categorical attributes. And we design an iterative method to balance the weight of each numerical attribute. For categorical attributes, we design a method to measure the distance between two categorical values. We have evaluated the performance of the proposed algorithm by conducting experiments. The results have shown that our algorithm is effective in clustering mixture types of data stream.

Acknowledgment

This work was supported by National Key Research and Development Program of China (2016YFB1001003), NSFC

(No. 61502543), and Guangdong Natural Science Funds for Distinguished Young Scholar (2016A030306014).

REFERENCES

[1] Aggarwal, C. Charu, Han, Jiawei, Wang, Jianyong, Yu, and S. Philip, "A framework for clustering evolving data streams," *VLDB*, vol. 29, pp. 81–92, 2003.

[2] L. Bai, X. Cheng, J. Liang, and H. Shen, "An optimization model for clustering categorical data streams with drifting concepts," pp. 2871–2883, 2016.

[3] C. D. Wang, J. H. Lai, D. Huang, and W. S. Zheng, "SVStream: A support vector-based algorithm for clustering data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1410–1424, 2013.

[4] M. Ghesmoune, M. Lebbah, and H. Azzag, *Clustering Over Data Streams Based on Growing Neural Gas*. Springer International Publishing, 2015.

[5] D. M. J. Tax and R. P. W. Duin, "Support vector domain description," *Pattern Recognition Letters*, vol. 20, no. 11–13, pp. 1191–1199, 1999.

[6] A. Ben-Hur, D. Horn, H. T. Siegelmann, and V. Vapnik, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, no. 2, pp. 125–137, 2002.

[7] A. Chaturvedi, P. E. Green, and J. D. Caroll, "K-modes clustering," *Journal of Classification*, vol. 18, no. 1, pp. 35–55, 2001.

[8] B. Liang, "Improved k-modes clustering algorithm based on rough sets," *Computer Science*, 2009.

[9] Y.-M. Xu, C.-D. Wang, and J.-H. Lai, "Weighted multi-view clustering with feature selection," *Pattern Recognition*, vol. 53, pp. 25–35, 2016.

[10] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," pp. 1027–1035, 2007.

[11] A. Strehl and J. Ghosh, "Cluster ensembles − a knowledge reuse framework for combining multiple partitions," *JMLR*, vol. 3, pp. 583–617, 2002.

[12] F. Cao, J. Liang, L. Bai, X. Zhao, and C. Dang, "A framework for clustering categorical time-evolving data," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 5, pp. 872–882, 2010.