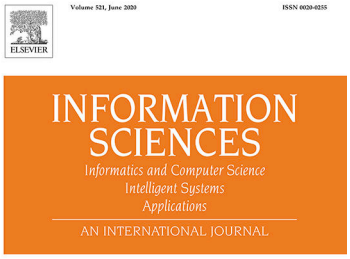
Journal Pre-proofs



k-Mnv-Rep: a *k*-type clustering algorithm for matrix-object data

Liqin Yu, Fuyuan Cao, Xiao-Zhi Gao, Jing Liu, Jiye Liang

PII: S0020-0255(20)30661-7
DOI: https://doi.org/10.1016/j.ins.2020.06.071
Reference: INS 15634

To appear in: *Information Sciences*

Available online at www.sciencedirect.com
ScienceDirect

Received Date: 8 December 2018
Accepted Date: 29 June 2020

Please cite this article as: L. Yu, F. Cao, X-Z. Gao, J. Liu, J. Liang, *k*-Mnv-Rep: a *k*-type clustering algorithm for matrix-object data, *Information Sciences* (2020), doi: https://doi.org/10.1016/j.ins.2020.06.071

k-Mnv-Rep: a k-type clustering algorithm for matrix-object data

Liqin Yu[a], Fuyuan Cao[a,*], Xiao-Zhi Gao[b], Jing Liu[a], Jiye Liang[a]

[a]*School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China*
[b]*School of Computing, University of Eastern Finland, Kuopio 70211, Finland*

Abstract

In matrix-object data, an object (or a sample) is described by more than one feature vector (record) and all of those feature vectors are responsible for the observed classification of the object. A task for matrix-object data is to cluster it into a set of groups by analyzing and utilizing the information of feature vectors. Matrix-object data are widespread in many real applications. Previous studies typically address data sets that an object is generally represented by a feature vector, which may be violated in many real-world tasks. In this paper, we propose a k-multi-numeric-values-representatives (abbr. k-Mnv-Rep) algorithm to cluster numeric matrix-object data. In this algorithm, a new dissimilarity measure between two numeric matrix-objects is defined and a new heuristic method of updating cluster centers is given. Furthermore, we also propose a k-multi-values-representatives (abbr. k-Mv-Rep) algorithm to cluster hybrid matrix-object data. The two proposed algorithms break the limitations of the previous studies, and can be applied to address matrix-object data sets that exist widely in many real-world tasks. The benefits and effectiveness of the two algorithms are shown by some experiments on real and synthetic data sets.

Keywords: Matrix-object, Clustering, Numeric, Hybrid

1. Introduction

Consider the following learning problem. Each user has more than one visited record about web pages in a fixed period, which are described by three features: Visited URL (Uniform Resource Locator), Visited time point and Visited time. These records carry behavior characteristics of each

*Corresponding author
Email addresses: 1367545521@qq.com (Liqin Yu), cfy@sxu.edu.cn (Fuyuan Cao), xiao-zhi.gao@uef.fi (Xiao-Zhi Gao), jingliu_sxu@hotmail.com (Jing Liu), ljy@sxu.edu.cn (Jiye Liang)

user. How to discover different user groups according to behavior characteristics has an important significance in recommend system and other applications.

Clustering is a widely used method to find different user groups in real-world applications [40]. It aims to partition the data of objects into a set of clusters in which objects in the same cluster are close to each other whereas objects in different clusters are far from each other [16, 20]. In traditional clustering problems, each object is typically represented as a fixed-length vector of features (usually called a feature vector). For our web page grouping problem, each record is a feature vector. Each object consists of more than one feature vector. We call this kind of grouping problem as matrix-object data clustering, which arises in complex applications of machine learning where each object is called as a matrix-object.

In matrix-object data clustering, each matrix-object is a r-by-m matrix that can be regarded as r feature vectors described by m attributes. Here, r varies from matrix-object to matrix-object. This kind of data extensively exist in many fields, such as banking, insurance, telecommunication, retails and medical science. Their characteristics are listed as follows.

- **Correlation**: Each matrix-object and its feature vectors have some correlations. For example, different users may have different interests about web pages.
- **One-many**: Each matrix-object corresponds with more than one feature vector. In the web page grouping problem, each user has many visited records, and the number of records varies from user to user.
- **Mixed**: In most cases, a matrix-object is described by both categorical and numeric attributes together. For example, Visited URL is a categorical attribute while Visited time is a numeric attribute.
- **Evolution**: Some attribute values may change as time goes on. For example, a user visits one URL repeatedly in this month, but in the next month the URL may be not visited. In other words, the change of a user's behavior is a dynamic evolution process with time.

If traditional clustering algorithms are used to solve the above problem, matrix-object data need to be transformed. One way is to compress a matrix-object into a vector, by which most of information is lost. A commonly used compression method is to represent a matrix-object by modes and means for categorical and numeric attributes, respectively. In the web page grouping

task, for one user, only the web page with the highest visited frequency is considered on Visited URL. For Visited time point and Visited time, only the average values are considered whereas the visited information of each web page for a user has not been reflected. Another way is to consider each attribute value as a new attribute, by which matrix-object data will be sparse and high dimensional. Especially for Visited time point and Visited time, data may become more sparse and high dimensional because of their continuous values. Therefore, traditional algorithms have some difficulties in dealing with matrix-object data clustering problem [24, 15, 18, 4, 6, 3, 19, 23].

In this paper, we propose two k-type algorithms to handle the matrix-object data clustering problem. A k-Mnv-Rep algorithm is developed to cluster the numeric matrix-object data, in which a new dissimilarity measure and a new update way of cluster centers are studied. Furthermore, we propose a k-Mv-Rep algorithm to cluster hybrid matrix-object data. The main contributions of our paper are given as follows.

- We define a novel dissimilarity measure to calculate the difference between two numeric matrix-objects.
- We provide an update policy of cluster centers after each iteration for clustering numeric matrix-object data.
- We propose the k-Mnv-Rep clustering algorithm to cluster numeric matrix-object data.
- We propose the k-Mv-Rep clustering algorithm to cluster hybrid matrix-object data, in which we define the dissimilarity between two hybrid matrix-objects and the update policy of cluster centers.

The rest of this paper is organized as follows. The related research work is reviewed in Section 2. The clustering algorithms for matrix-object data are proposed in Section 3. The experimental results and the convergence test of our algorithms are reported and carried out in Section 4. The scalability study on numeric synthetic matrix-object data is provided in Section 5. This paper is concluded and some future work is pointed out in Section 6.

2. Related work

A comparable situation is multi-instance (MI) learning. It is a variation of the standard supervised machine learning. In MI learning, each example (a bag) may have many alternative feature

vectors (instances), where every bag has a class label. However, the instances themselves are not explicitly labeled. The learning target is to build a model based on the given examples that can accurately predict the class labels of the future bags.

MI learning was first proposed in [37]. It is assumed that a bag has many feature vectors, but only one of those feature vectors may be responsible for the observed classification of the object. A strong assumption is made regarding the relationship between instances inside the bag and the label of the bag, which is generally referred to as the standard MI assumption. Under this assumption, each instance has a hidden class label identifying it as either a positive or a negative instance. A bag is considered to be positive if and only if it contains at least one positive instance. Nevertheless, for many applications, the standard MI assumption may not be directly applicable. Therefore, there has been a trend toward the relaxation of this assumption, e.g., other interactions between instances and the class labels of bags are possible.

Weidmann et al. [39] a total of proposed three types of generalized MI assumptions and formulated a hierarchy. Under the count-based MI assumption (the most generalized assumption), MI learning is no longer a binary classification problem, and a bag is positive if and only if the number of instances it contains is within a certain range for each positive concept. Scott et al. [31] proposed the GMIL assumption, in which a bag is positive if and only if its instances are sufficiently close to at least r concepts of k target concepts. Combing the two assumptions, Tao et al. [36] proposed the count-based GMIL assumption. The DD-SVM/MILES assumption [8, 7] is related to the GMIL assumption, in that the distance from a set of target concepts is used to determine bag labels. The BARTMIP method [44] assumes that bag labels are related to the distances from target bags instead of target concepts. In the collective assumption [41], the bag-level class label is determined by the average class value of the instances of that bag. By considering the weight of each instance, Foulds proposed two MI assumptions [14].

In MI learning, each MI assumption defines a relationship between the instances in a bag and the bag-level class label, i.e., each instance of bags has a hidden class label. Given a bag, its label is obtained by the labels of its instances. Notably, the assumption view of MI learning is useful from a practical machine learning perspective [21].

MI learning may not perform satisfactorily in case when training example bags do not have class labels. For example, under the standard assumption, a positive bag has many instances but perhaps only one of those instances is positive. Since instances in bags have a hidden class label

without being explicitly labeled, it is difficult to guarantee that the distance between two positive bags is based on the label of the positive instance. Therefore, unsupervised learning-based MI is indeed challenging. As we know, the BAMIC algorithm is an unsupervised MI learning algorithm [44]. However, it is only used to prepare for predicting labels of future bags in the BARTMIP method and in the BAMIC algorithm, the hidden class label of an instance is not considered.

Matrix-objects learning focuses on behavior analysis. In the matrix-object data clustering problem, each feature vector does not have a hidden class label and all feature vectors are responsible for the observed classification of the matrix-object. Furthermore, each feature vector should correspond with a time point. In other words, the change of a user's behavior is a dynamic evolution process with time.

3. Matrix-object data clustering

In this section, we propose two k-type algorithms to cluster matrix-object data. The k-type clustering algorithm [26, 17] consists of three components: (1) representation of cluster centers, (2) allocation of objects into clusters and (3) update of cluster centers. Thus, these two new algorithms are described from two aspects: the dissimilarity measure, and the representation and update policy of cluster centers.

3.1. Problem statement and notations

Let $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be a matrix-object data set described by m attributes $\{A_1, A_2, \ldots, A_m\}$, where $X_i (1 \leq i \leq n)$ is the ith matrix-object with r_i feature vectors and is described as

$$X_i = \begin{pmatrix} v_{i11} & v_{i12} & \cdots & v_{i1m} \\ v_{i21} & v_{i22} & \cdots & v_{i2m} \\ \vdots & \vdots & \ddots & \vdots \\ v_{ir_i1} & v_{ir_i2} & \cdots & v_{ir_im} \end{pmatrix}.$$

It can also be written as $X_i = (X_{i1}; X_{i2}; \ldots; X_{im})$ representing m r_i-dimensional column vectors, where r_i may be different for each matrix-object and $X_{is} = [v_{i1s}, v_{i2s}, \ldots, v_{ir_is}]' \ (1 \leq s \leq m)$ is the sth r_i-dimensional column vector. Moreover, let V^s and $V_{X_i}^{A_s}$ be the set of domain values of $\mathbf{X}$ and X_i on A_s respectively. The task of matrix-object clustering is to group $\mathbf{X}$ into clusters. The k-mw-modes algorithm has been proposed for categorical matrix-object data clustering problem [5]. In this

algorithm, the computation of the distance between two matrix-objects and the update mechanism of cluster centers are based on the frequency of attribute values. For numeric matrix-object data, its attribute values are continuous not discrete. Therefore, the k-mw-modes algorithm is not efficient in coping with the numeric matrix-object data clustering problem. In this paper, we mainly consider numeric matrix-object data clustering problem. In addition, numeric and categorical attributes are usually mixed in many real-world matrix-object data sets. With the k-prototypes algorithm as a reference, we also propose a new algorithm to attack the hybrid matrix-object data clustering problem.

3.2. Proposed algorithm for numeric matrix-object data

In this section, a k-Mnv-Rep algorithm is developed to cluster numeric matrix-object data, in which a dissimilarity between two numeric matrix-objects is defined in order to allocate objects into clusters. Moreover, a heuristic method of updating cluster centers is proposed.

3.2.1. Dissimilarity between two numeric matrix-objects

Each matrix-object is a r_i-by-m ($r_i \geq 1$) matrix, and r_i varies from matrix-object to matrix-object. To measure the dissimilarity between two numeric matrix-objects, the Hausdorff distance [13] and its variants[38, 44] can be applied. The dissimilarity is actually determined by the Euclidean distance of two feature vectors from two matrix-objects, which violates the property that all feature vectors are responsible for the observed classification of matrix-objects. In this paper, we propose a new dissimilarity measure for numeric matrix-objects. It is known that each distance must be obtained under the same feature space. Suppose features (attributes) are independent, the dissimilarity between two matrix-objects can be measured by their difference on each feature. Because each matrix-object can be viewed as m r_i-dimensional vectors, the problem is converted to acquire the dissimilarity between two vectors with different lengths in the same feature space. Due to the continuity of numeric attribute values, in the new measure we make use of their neighbors. The details are described as follows.

Definition 1. *Suppose that* $\mathbf{X}$ *is a given numeric matrix-object data set,* V^s *denotes the set of domain values of* $\mathbf{X}$ *on the attribute* A_s. $X_{is} = [v_{i1s}, v_{i2s}, \ldots, v_{ir_is}]'$ *is the* s*th* r_i*-dimensional column vector of* $X_i (X_i \in \mathbf{X})$. $\forall v \in V^s$*, the number of its neighbors in* X_{is} *for a given parameter*

ε_s is defined as

$$n\_f_i(v) = \sum_{p=1}^{r_i} n\_g(v, v_{ips}), \tag{1}$$

where

$$n\_g(x, y) = \begin{cases} 1, & if \quad |x - y| \leq \varepsilon_s. \\ 0, & otherwise. \end{cases} \tag{2}$$

Definition 2. *Given two numeric matrix-objects X_i, X_j described by m attributes $\{A_1, A_2, \ldots, A_m\}$, the dissimilarity measure between them can be defined as*

$$n\_d(X_i, X_j) = \frac{1}{m} \sum_{s=1}^{m} n\_\delta(X_{is}, X_{js}), \tag{3}$$

where

$$n\_\delta(X_{is}, X_{js}) = \frac{1}{2} \sum_{v \in V} \left| \frac{n\_f_i(v)}{\sum_{v \in V} n\_f_i(v)} - \frac{n\_f_j(v)}{\sum_{v \in V} n\_f_j(v)} \right|, \tag{4}$$

and $V = X_{is} \cup X_{js}$. $|\cdot|$ represents the absolute value of a value. In addition, a normalization factor $\frac{1}{2}$ is added to make $0 \leq n\_\delta(X_{is}, X_{js}) \leq 1$. And $n\_\delta(X_{is}, X_{js}) = 1$ if and only if $X_{is} \cap X_{js} = \emptyset$.

Given a parameter ε_s on the attribute A_s, each attribute value has some neighbors in the vector X_{is}. According to the definitions, $n\_f_i(v)$ is the number of neighbors of the attribute value v in X_{is}. Given two matrix-objects X_i, X_j, $\frac{n\_f_i(v)}{\sum_{v \in V} n\_f_i(v)}$ reflects the importance of v in X_{is}. The higher $n\_f_i(v)$ is, the more important v in X_{is} is. For X_{is} and X_{js}, if the closer the importance of v in X_{is}, X_{js} are, the more similar X_{is}, X_{js} are. Therefore, the dissimilarity between X_{is} and X_{js} can be measured by the difference of importance of their attribute values.

As an example, the dissimilarity between two numeric matrix-objects is given as follows.

Example 1. *Suppose that X_i, X_j, X_k are three numeric matrix-objects, they are described by m attributes. On the attribute A_s, $X_{is} = [2, 3, 6]'$, $X_{js} = [3, 9]'$, $X_{ks} = [3, 7]'$. We can obtain $X_{is} \cup X_{js} = \{2, 3, 6, 9\}$. Suppose that $\varepsilon_s = 1$, according to Eq.(1),*

$$n\_f_i(2) = 2, n\_f_i(3) = 2, n\_f_i(6) = 1, n\_f_i(9) = 0,$$

$$n\_f_j(2) = 1, n\_f_j(3) = 1, n\_f_j(6) = 0, n\_f_j(9) = 1.$$

The dissimilarity between X_{is}, X_{js} is $|\frac{2}{2+2+1+0} - \frac{1}{1+1+0+1}| = \frac{1}{15}$ for the attribute value 2. In the same way, we compute the dissimilarities between X_{is}, X_{js} about the other three values as $\frac{1}{15}, \frac{1}{5}, \frac{1}{3}$, respectively. So, the dissimilarity between X_{is}, X_{js} can be computed as $n\_\delta(X_{is}, X_{js}) = \frac{1}{2}(\frac{1}{15} + \frac{1}{15} +$

$\frac{1}{5}+\frac{1}{3})=\frac{1}{3}$. *Similarly, the dissimilarity between* X_{is}, X_{ks} *can be computed as* $n\_\delta(X_{is}, X_{ks})=\frac{1}{6}$. *Apparently,* $n\_\delta(X_{is}, X_{js}) > n\_\delta(X_{is}, X_{ks})$, *which confirms the facts. The dissimilarities on other attributes can be obtained as well.*

3.2.2. A heuristic updating cluster center method for numeric matrix-object data

In the k-type algorithm, the clusters are updated iteratively by allocating the object into the closest cluster. In each iteration, the distances between each object and each cluster need to be computed. However, the distance computation is very consuming, because each cluster contains many objects and algorithms need multiple iterations. Therefore, the cluster representation is critical. Usually, the k-type clustering algorithm can find the best representation by traversing all possible cluster centers, but it is very time consuming. In our paper, a heuristic algorithm is proposed to discover cluster centers.

For traditional data, each object is a feature vector, and usually its cluster representation is also a vector. That is, the representation is an attribute value for each attribute. For matrix-object data, each matrix-object has many feature vectors. In order to obtain a better representation, multiple representative values should be selected to represent a cluster for each attribute. In this section, we define a weight for each attribute value to measure its representative.

Definition 3. *Given a numeric matrix-object data set* $\mathbf{X}=\{X_1, X_2, ..., X_n\}$, *it is described by* m *attributes. Let* V^s *represent the set of domain values of* $\mathbf{X}$ *on the attribute* A_s. $\forall v \in V^s$, *its weight in* $\mathbf{X}$ *on* A_s *is defined as*

$$n\_\omega(v)=\frac{1}{n}\sum_{i=1}^{n}\frac{n\_f_i(v)}{r_i}. \tag{5}$$

Given the matrix-object $X_i (X_i \in \mathbf{X})$, according to Eq.(1), $n\_f_i(v)$ is the number of neighbors of the attribute value v in X_{is}. So $\frac{n\_f_i(v)}{r_i}$ reflects the importance of v in X_{is}. The higher $n\_f_i(v)$ is, the more important v in X_{is}. If an attribute value is selected to represent the cluster center, it is supposed to be important in each matrix-object. In Eq.(5), $0 \leq \frac{n\_f_i(v)}{r_i} \leq 1$, $0 \leq n\_\omega(v) \leq 1$. And $n\_\omega(v)=1$ if and only if $\frac{n\_f_i(v)}{r_i}=1$ $(\forall i \in \{1, 2, \ldots, n\})$. That is, an attribute value has a higher weight if and only if it is important in each matrix-object.

By Eq.(5), we can obtain the weight of each attribute value and arrange them in the descending order of the weight. For the attribute A_s, the first u_s values are selected as the cluster center. We set $u_s = round(\frac{1}{n}\sum_{i=1}^{n}|V_{X_i}^{A_s}|)$. In this way, the cluster centers on other attributes can be found as

well. In the next iteration, the new cluster centers are used to compute the distances instead of the clusters. The heuristic algorithm is described in **Algorithm 1**.

Algorithm 1 A heuristic updating cluster center algorithm for numeric matrix-object data.

Input:
- **X**: a set of n numeric matrix-objects described by m attributes;
-ε: $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m\}$ is the set of parameters for computing the neighbors;
Output: - Q: A cluster center of **X**;
Method:
for $s = 1$ to m **do**
  sum=0, $Q = \emptyset$;
  for $i = 1$ to n **do**
    $sum = sum + |V_{X_i}^{A_s}|$;
  end for
  $u_s = round(sum/n)$;
  for $t = 1$ to $|V^s|$ **do**
    Calculate the weight $n\_\omega(v_t^s)$ of $v_t^s \in V^s$ by Eq.(5);
  end for
  Let $\{v_1^s, v_2^s, \ldots, v_{|V^s|}^s\}$ be the set after the descending order by the weight;
  $Q_{A_s} = \{v_1^s, v_2^s, \ldots, v_{u_s}^s\}$;
  $Q = Q \bigcup Q_{A_s}$;
end for
return Q.

An example is given to illustrate the process of updating cluster centers of the numeric matrix-object data sets as follows.

Example 2. *Given a numeric matrix-object data set* $\mathbf{X} = \{X_1, X_2, X_3\}$*, it is described by* m *attributes. On the attribute* A_s*,* $X_{1s} = [2, 2, 3, 6]'$*,* $X_{2s} = [1, 3, 7]'$*,* $X_{3s} = [4, 4, 5]'$*. We can obtain the domain values* V^s *of* $\mathbf{X}$ *on* A_s*, and* $V^s = \{1, 2, 3, 4, 5, 6, 7\}$*. Suppose that* $\varepsilon_s = 1$*, we can compute the weight of each attribute value in* $\mathbf{X}$ *by Eq.(5). For the value 1,* $n\_\omega(1) = \frac{1}{3}(\frac{2}{4} + \frac{1}{3} + \frac{0}{3}) = \frac{5}{18}$*. Similarly, we can compute* $n\_\omega(2) = \frac{17}{36}, n\_\omega(3) = \frac{7}{12}, n\_\omega(4) = \frac{19}{36}, n\_\omega(5) = \frac{5}{12}, n\_\omega(6) = \frac{11}{36}, n\_\omega(7) = \frac{7}{36}$*. Obviously,* $n\_\omega(3) > n\_\omega(4) > n\_\omega(2) > n\_\omega(5) > n\_\omega(6) > n\_\omega(1) >$

$n\_\omega(7)$. *The number of attribute values is computed as* $u_s = round(\frac{3+3+2}{3}) = 3$, *so we select* $\{3, 4, 2\}$ *as the representative values of the data set* $\mathbf{X}$.

3.2.3. The k-Mnv-Rep algorithm

Given a numeric matrix-object set $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ clustered into k ($\ll n$) clusters, the goal of the k-Mnv-Rep algorithm is to minimize the following objective function

$$\begin{aligned} F(W, Q) &= \sum_{l=1}^{k}\sum_{i=1}^{n} \omega_{li} \times n\_d(X_i, Q_l), \\ s.t. \quad & \omega_{li} \in \{0, 1\}, 1 \le l \le k, 1 \le i \le n, \\ & \sum_{l=1}^{k} \omega_{li} = 1, 1 \le i \le n, \\ & 0 < \sum_{i=1}^{n} \omega_{li} < n, 1 \le l \le k, \end{aligned} \tag{6}$$

where $W = [\omega_{li}]$ is a k-by-n $\{0, 1\}$ matrix in which $\omega_{li} = 1$ indicates that object X_i is allocated to cluster l, and $Q = [Q_1, Q_2, \ldots, Q_k]$ in which $Q_l \in Q$ is the multi-numeric-values representatives of cluster l.

Minimizing this objective function is known as a NP-hard problem [12]. In general, $F(W, Q)$ can be solved with an iterative process to solve two subproblems iteratively until the process converges. The first step is to fix $Q = Q^t$ at iteration t and solve the reduced problem $F(W, Q^t)$ with Eq.(3) to find W^t that minimizes $F(W, Q^t)$, then the second step is to fix W^t and solve the reduced problem $F(W^t, Q)$ by using **Algorithm 1** to find Q^{t+1} that minimizes $F(W^t, Q)$. The k-Mnv-Rep algorithm is described in **Algorithm 2**.

The computational complexity of the k-Mnv-Rep algorithm is analyzed as follows.

- Computing dissimilarity. The complexity for calculating the dissimilarity between two numeric matrix-objects on A_s is $\mathcal{O}(|V^s|)$. The computational complexity of the dissimilarity between two numeric matrix-objects on m attributes is $\mathcal{O}(m|V'|)$, where $|V'| = max\{|V^s|, 1 \le s \le m\}$.

- Updating cluster centers. The complexity for computing the weight of attribute values on A_s is $\mathcal{O}(n|V^s|)$. The computational complexity of updating k cluster centers on m attributes is $\mathcal{O}(kmn|V'|)$, where $|V'| = max\{|V^s|, 1 \le s \le m\}$.

Algorithm 2 The k-Mnv-Rep Algorithm.

Input:
- **X**: a data set of n numeric matrix-objects described by m attributes;
- k: the number of clusters need to be clustered;
- ϵ: a threshold;

Output: - *cid*: the labels of all objects after clustering;

Method:

Generate k random numbers and obtain k initial centers by the indexes;
Let $Q = \{Q_1, Q_2, \ldots, Q_k\}$ be the initial centers, and $value = 0$, $num = 0$;
while $num \leq 100$ **do**
  $value1 = 0$;
  for $i = 1$ *to* n **do**
    Allocate X_i into the lth cluster if $l = \arg\min_{l=1}^{k}\{n\_d(X_i, Q_l)\}$.
    $value1 = value1 + \min_{l=1}^{k}\{n\_d(X_i, Q_l)\}$;
  end for
  If $|value1 - value| \leq \epsilon$, *break*; Else $value = value1$ and $num = num + 1$;
  for $l = 1$ to k **do**
    Update the cluster center Q_l by **Algorithm 1**;
  end for
end while

If the clustering process needs t iterations to converge, the total computational complexity of the k-Mnv-Rep algorithm is $\mathcal{O}(tmnk|V'|)$, where $|V'| = max\{|V^s|, 1 \leq s \leq m\}$. It is obviously that the time complexity of the proposed algorithm increases linearly as the number of matrix-objects, attributes, clusters and attribute values increases.

3.3. Proposed algorithm for hybrid matrix-object data

In the section 3.2, the k-Mnv-Rep algorithm has been proposed to solve numeric matrix-object data clustering problem, but it is limited to numeric matrix-object data. In real world, it is common that numeric and categorical attributes are mixed in many data sets. In this section, with the k-prototypes algorithm as a reference, we extend the k-Mnv-Rep to domains with mixed numeric and

categorical values by combining the k-mw-modes algorithm [5]. The new algorithm is also described from two processes: dissimilarity measure and updating cluster centers.

3.3.1. Dissimilarity between two hybrid matrix-objects

Distance measures are usually meaningful only in the same feature space. For example, in the traditional data representation, in order to measure the distance between two hybrid objects, different measures are usually used on numeric and categorical attributes [17]. For hybrid matrix-object data, we also use different distances to measure data on different type of attributes. The details are given as follows.

Definition 4. *Let* $\mathbf{X}$ *be a hybrid matrix-object data set described by* m *attributes* $\{A_1^r, \ldots, A_p^r, A_{p+1}^c, \ldots, A_m^c\}$*, where the first* p *attributes describe numeric data and the rest describes categorical data.* $\forall X_i, X_j \in \mathbf{X}$*, the dissimilarity measure between them can be defined as*

$$h\_d(X_i, X_j) = \sum_{s=1}^{p} n\_\delta(X_{is}, X_{js}) + \gamma \sum_{s=p+1}^{m} c\_\delta(X_{is}, X_{js}), \tag{7}$$

where the weight γ *is used to avoid favouring either type of attribute.* $c\_\delta(X_{is}, X_{js})$ *is a distance measure on* A_s *about categorical matrix-objects as follows [5].*

$$c\_\delta(X_{is}, X_{js}) = \frac{1}{2} \sum_{v \in V} \left| \frac{\sum_{p=1}^{r_i} c\_g(v, v_{ips})}{r_i} - \frac{\sum_{q=1}^{r_j} c\_g(v, v_{jqs})}{r_j} \right|, \tag{8}$$

where $V = X_{is} \cup X_{js}$. $c\_g(\cdot, \cdot)$ is a function whose value is 1 if two parameter values are equal, otherwise its value is 0. A normalization factor $\frac{1}{2}$ is added to make $0 \leq c\_\delta(X_{is}, X_{js}) \leq 1$. And $c\_\delta(X_{is}, X_{js}) = 1$ if and only if $X_{is} \cap X_{js} = \emptyset$.

In Eq.(7), as $0 \leq n\_\delta(X_{is}, X_{js}) \leq 1$ and $0 \leq c\_\delta(X_{is}, X_{js}) \leq 1$, so the weight can be set to $\gamma = 1$. If ε_s in Eq.(1) is set to 0, $n\_g(v, v_{ips}) = c\_g(v, v_{ips})$ and $\sum_{v \in V} n\_f_i(v) = r_i$. i.e., $c\_\delta(X_{is}, X_{js}) = n\_\delta(X_{is}, X_{js})$. Therefore, the dissimilarity between two hybrid matrix-objects can be unified as follows.

Definition 5. *Given a hybrid matrix-object data set* $\mathbf{X}$*,* V^s *denotes the set of its domain values on the attribute* A_s*.* $\forall v \in V^s$*, the number of its neighbors in* X_{is} *for a given parameter* ε_s' *is defined as*

$$h\_f_i(v) = \sum_{p=1}^{r_i} h\_g(v, v_{ips}), \tag{9}$$

where

$$h\_g(x,y) = \begin{cases} 1, & if \quad |x-y| \le \varepsilon'_s, \\ 0, & otherwise, \end{cases} \tag{10}$$

and

$$\varepsilon'_s = \begin{cases} \varepsilon_s, & numeric. \\ 0, & categorical. \end{cases} \tag{11}$$

Definition 6. *Let* $\mathbf{X}$ *be a hybrid matrix-object data set described by* m *attributes.* $\forall X_i, X_j \in \mathbf{X}$*, the dissimilarity measure between them can be defined as*

$$h\_d(X_i, X_j) = \frac{1}{m}\sum_{s=1}^{m} h\_\delta(X_{is}, X_{js}), \tag{12}$$

where

$$h\_\delta(X_{is}, X_{js}) = \frac{1}{2}\sum_{v\in V}\left|\frac{h\_f_i(v)}{\sum_{v\in V} h\_f_i(v)} - \frac{h\_f_j(v)}{\sum_{v\in V} h\_f_j(v)}\right|, \tag{13}$$

and $V = X_{is} \cup X_{js}$*.* $|\cdot|$ *represents the absolute value of a value.*

For categorical attribute, $\varepsilon'_s = 0$, $h\_f_i(v)$ denotes the frequency of the attribute value v. For numeric attribute, $\varepsilon'_s = \varepsilon_s$, $h\_f_i(v)$ denotes the number of neighbors of v. For a given data set and a given attribute, the frequency and the number of neighbors all can reflect the importance of an attribute value. So $h\_d(X_i, X_j)$ can measure the dissimilarity.

3.3.2. A heuristic updating cluster center method for hybrid matrix-object data

The cluster centers are updated respectively for numeric and categorical attributes. For each kind of attribute, the updating method aims to find some values with higher weight. The definition of weight is the key of updating algorithm. According to Eq.(9), for a given value v, $h\_f_i(v)$ can be unified for numeric and categorical attributes. Therefore, for a given hybrid data set, the weight of an attribute value can be written as follows.

Definition 7. *Let* $\mathbf{X}$ *be a data set of* n *hybrid matrix-objects described by* m *attributes. Suppose that* V^s *represents the set of domain values of* $\mathbf{X}$ *on the attribute* A_s*,* $\forall v \in V^s$*, its weight in* $\mathbf{X}$ *on* A_s *is defined as*

$$h\_\omega(v) = \frac{1}{n}\sum_{i=1}^{n}\frac{h\_f_i(v)}{r_i}. \tag{14}$$

where $h\_f_i(v)$ *is obtained by Eq.(9).*

For categorical attribute, $h\_f_i(v)$ denotes the frequency of the attribute value v. For numeric attribute, $h\_f_i(v)$ denotes the number of neighbors of v. No matter which attribute, $\frac{h\_f_i(v)}{r_i}$ can reflect the importance of v in X_i. If a value has higher importance in each matrix-object, its weight $h\_\omega(v)$ is higher. We select some values having higher weight as the center. The number of values as the new center is set to $u = round(\frac{1}{n}\sum_{i=1}^{n}|V_{X_i}^{A_s}|)$.

3.3.3. The k-Mv-Rep algorithm

Let $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be a hybrid matrix-object set. The k-Mv-Rep algorithm for clustering $\mathbf{X}$ into k ($\ll n$) clusters aims to minimize the following objective function

$$\begin{aligned} & F'(W', Q') = \sum_{l=1}^{k}\sum_{i=1}^{n}\omega'_{li} \times h\_d(X_i, Q'_l), \\ & s.t. \quad \omega'_{li} \in \{0,1\}, 1 \le l \le k, 1 \le i \le n, \\ & \qquad \sum_{l=1}^{k}\omega'_{li} = 1, 1 \le i \le n, \\ & \qquad 0 < \sum_{i=1}^{n}\omega'_{li} < n, 1 \le l \le k, \end{aligned} \tag{15}$$

where $W' = [\omega'_{li}]$ is a k-by-n $\{0,1\}$ matrix in which $\omega'_{li} = 1$ indicates that X_i is allocated to cluster l, and $Q' = [Q'_1, Q'_2, \ldots, Q'_k]$ in which $Q'_l \in Q'$ is the multi-values representatives of cluster l.

Similar to the k-Mnv-Rep algorithm, we still apply the new heuristic updating center method for hybrid matrix-object data to find a local minimum of $F'(W', Q')$. The process of finding the local minimum is the same as the k-Mnv-Rep algorithm. The details of the k-Mv-Rep algorithm are described in **Algorithm 3**.

The computational complexity of the k-Mv-Rep algorithm is $\mathcal{O}(tmnk|V'|)$, where t represents the iterations and $|V'| = max\{|V^s|, 1 \le s \le m\}$. It is obviously that the time complexity of the proposed algorithm increases linearly as the number of matrix-objects, attributes, clusters and attribute values increases.

4. Experiments

To evaluate the effectiveness of the proposed algorithms, we conduct a series of experiments on real data sets. Firstly, five external indices are introduced. Then, we conduct experiments

Algorithm 3 The k-Mv-Rep Algorithm.

Input:
- **X**: a data set of n hybrid matrix-objects described by m attributes;
- k: the number of clusters need to be clustered;
- ϵ: a threshold;

Output: - *cid*: the labels of all objects after clustering;

Method:

Generate k random numbers and obtain k initial centers by the indexes;
Let $Q = \{Q_1, Q_2, \ldots, Q_k\}$ be the initial centers, and $value = 0, num = 0$;
while $num \leq 100$ **do**
  $value1 = 0$;
  for $i = 1$ *to* n **do**
    Arrange X_i to the lth cluster if $l = \arg\min_{l=1}^{k}\{h\_d(X_i, Q_l)\}$.
    $value1 = value1 + \min_{l=1}^{k}\{h\_d(X_i, Q_l)\}$;
  end for
  If $|value1 - value| \leq \epsilon$, *break*; Else $value = value1$ and $num = num + 1$;
  for $l = 1$ to k **do**
    Update the cluster center Q_l by Eq.(14);
  end for
end while

on numeric matrix-object data to evaluate the k-Mnv-Rep algorithm. Next, some experiments on hybrid matrix-object data are made to evaluate the k-Mv-Rep algorithm. Finally, we discuss the impact of the parameter ϵ on the effectiveness of the two proposed algorithms, and test the convergence of the two algorithms.

4.1. Evaluation indexes

We introduce five external indices to evaluate the proposed algorithms. They are accuracy (AC), Precision(PE), recall (RE), adjusted rand index (ARI) [25] and normalized mutual information (NMI) [35] respectively.

Let **X** be a matrix-object data set, $C = \{C_1, C_2, \ldots, C_{k'}\}$ be a clustering result of **X**, $P = \{P_1, P_2, \ldots, P_k\}$ be a real partition in **X**. The overlap between C and P can be summarized in a

contingency table shown in Table 1, where n_{ij} denotes the number of matrix-objects in common between P_i and C_j, $n_{ij} = |P_i \bigcap C_j|$. p_i and c_j are the number of matrix-objects in P_i and C_j, respectively.

Table 1: The contingency table.

	C_1	C_2	$\cdots$	$C_{k'}$	$Sums$
P_1	n_{11}	n_{12}	$\cdots$	$n_{1k'}$	p_1
P_2	n_{21}	n_{22}	$\cdots$	$n_{2k'}$	p_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
P_k	n_{k1}	n_{k2}	$\cdots$	$n_{kk'}$	p_k
$Sums$	c_1	c_2	$\cdots$	$c_{k'}$	n

The five evaluation indexes are defined as follows:

$$AC = \frac{1}{n} \max_{j_1 j_2 \cdots j_k \in S} \sum_{i=1}^{k} n_{ij_i}, PE = \frac{1}{k} \sum_{i=1}^{k} \frac{n_{ij_i^*}}{p_i}, RE = \frac{1}{k} \sum_{i=1}^{k} \frac{n_{ij_i^*}}{p_i},$$

$$ARI = \frac{\sum_{ij} C_{n_{ij}}^2 - [\sum_i C_{p_i}^2 \sum_j C_{c_j}^2]/C_n^2}{\frac{1}{2}[\sum_i C_{p_i}^2 + \sum_j C_{c_j}^2] - [\sum_i C_{p_i}^2 \sum_j C_{c_j}^2]/C_n^2},$$

$$NMI = \frac{\sum_{i=1}^{k} \sum_{j=1}^{k'} n_{ij} log(\frac{n_{ij} n}{p_i c_j})}{\sqrt{\sum_{i=1}^{k} p_i log(\frac{p_i}{n}) \sum_{j=1}^{k'} c_j log(\frac{c_j}{n})}},$$

where $n_{1j_1^*} + n_{2j_2^*} + \ldots + n_{kj_k^*} = \max_{j_1 j_2 \ldots j_k \in S} \sum_{i=1}^{k} n_{ij_i}$ $(j_1^* j_2^* \ldots j_k^* \in S)$ and $S = \{j_1 j_2 \ldots j_k | j_1, j_2, \cdots, j_k \in \{1, 2, \ldots, k\}, j_i \neq j_t \text{ for } i \neq t\}$ is a set of all permutations of $1, 2, \ldots, k$. We consider that the higher the values of AC PE, RE, ARI and NMI are, the better the clustering solution is.

4.2. Experiments on numeric matrix-object data

We evaluate the k-Mnv-Rep algorithm by conducting experiments on numeric matrix-object data sets. Two parts are included into this subsection, the introduction of real data sets and the comparison of experimental results by k-Mnv-Rep algorithm and the other seven algorithms.

4.2.1. The real data sets

As the lack of the public numeric matrix-object data sets, we applied some multi-instance data sets to evaluate the k-Mnv-Rep algorithm. Some experiments on nine real data sets are made.

These data sets are ever processed by Veronika Cheplygina [9] and can be found on their web site [1]. The details are listed in Table 2.

Table 2: The details of the nine data sets.

Datasets	*Objects*	*Attributes*	*Records*	*Clusters*
Messidor	1200	689	12352	2
Elephant	200	232	1391	2
Web2	75	6521	2219	2
Mutagenesis1	188	9	10486	2
Mutagenesis2	42	9	2132	2
Musk1	92	167	476	2
Component	718	200	13129	2
Process	1240	200	21306	2
Function	770	200	13498	2

Messidor [10, 22] is from the image classification problem. Each image from the original data is split up into patches. Patches which do not have a sufficient amount of foreground are discarded. Each image can be seen as a matrix-object and each patch is a feature vector. The original data can be found from two web sites [2] [3].

Elephant is a data set about images [1]. Each image can be taken as a matrix-object, and each image segment is a feature vector of the matrix-object. Web2 is from the problem that aims to classify webpage as interesting or not [45]. A webpage can be viewed as a matrix-object, and the links on the webpage can be taken as the feature vectors.

Mutagenesis [34] and Musk1 [37] are from the drug activity prediction problem. In these data sets, each molecule is seen as a matrix-object, and each shape of each molecule is viewed as a feature vector. The Mutagenesis data have two versions, easy (1) and hard (2). For the Musk1, the original data can be downloaded from UCI [2].

Component, Process and Function are from the text categorization problem [29]. Each document corresponds to a matrix-object and each paragraph to a feature vector in it. Features used are word occurrence frequencies and some statistics about the nature of the protein-GO code interaction for

[1] `http://www.miproblems.org`

[2] `http://messidor.crihan.fr/download.php`

[3] `http://www.bioimage.ucsb.edu/research/biosegmentation`

each paragraph. The original data can be found from a web site [4].

4.2.2. Comparison results

We compared the k-Mnv-Rep algorithm with the BAMIC algorithm with three distances [44], the CAN, PCAN algorithms [27], the CLR-L_1, CLR-L_2 algorithms [28] by conducting some experiments on the nine numeric data sets. For the last four algorithms, their inputs are a data set in which each object is described by a vector. Therefore, we take the mean value of each matrix-object as the attribute value on each attribute.

In this experiment, we run the eight algorithms 30 times respectively and take the mean value as the final results. Furthermore, the parameter ε_s is set to the half of standard deviation of $\mathbf{X}$ on the sth attribute and the parameter ϵ is set to 0.2 in the k-Mnv-Rep algorithm.

The comparison results on the nine data sets are shown in Table 3. The left of the signal "$\pm$" denotes the mean value while the right of it represents the standard deviation. The values of evaluation indexes of these algorithms are ranked for each data set, the highest value getting the rank 1, the second higher value getting rank 2, ..., as shown in the parentheses in Table 3. AvgR represents the average rank of these algorithms on nine data sets.

From Table 3, we can see that AvgR of the k-Mnv-Rep algorithm ranks the first on all evaluation indexes. That is to say, the k-Mnv-Rep algorithm outperforms other compared algorithms in general. In detail, the k-Mnv-Rep algorithm is better than the other algorithms on six data sets for each evaluation index. For the rest of three data sets, two of which have the second higher value on all evaluation indexes. For the data set Function, the k-Mnv-Rep algorithm is only worse than the PCAN algorithm. However, the PCAN algorithm can not be applied in some data sets because of inverse matrix needs to be computed.

In addition, for AC, we can find that the BAMIC-avgH is better than the BAMIC-minH and the BAMIC-maxH on three data sets while the BAMIC-maxH is better than the BAMIC-minH and the BAMIC-avgH on four data sets. On the rest two data sets, the BAMIC-minH algorithm performs the best. For the other evaluation indexes, the same problem appears in the BAMIC algorithm. Therefore, it is difficult for the BAMIC algorithm to find the most appropriate distance from the three distances.

[4] http://www.biocreative.org/tasks/biocreative-i/task-2-functional-annotations/

Table 3: Comparison results of different algorithms on the nine numeric data sets.

Index	*Datasets*	*BAMIC-minH*	*BAMIC-maxH*	*BAMIC-avgH*	*CAN*	*PCAN*	*CLR-L_1*	*CLR-L_2*	*k-Mnv-Rep*
	Messidor	0.5194±0.0092(3)	0.5153±0.0138(4)	0.5223±0.0149(2)	———————(6.5)	———————(6.5)	———————(6.5)	———————(6.5)	**0.5264±0.0325**(1)
	Elephant	0.5868±0.0385(4)	0.6005±0.0311(2)	0.5996±0.0412(3)	0.5250±1.11E-16(5)	———————(8)	0.5150±1.11E-16(6.5)	0.5150±1.11E-16(6.5)	**0.6035±0.0584**(1)
	Web2	0.6662±0.1051(2)	**0.7106±0.0397**(1)	0.6600±0.0451(3)	0.5466±1.11E-16(6.5)	———————(8)	0.5600±1.11E-16(5)	0.5466±1.11E-16(6.5)	0.6262±0.1021(4)
	Muta1	0.6053±0.0449(2)	0.5317±0.0171(6)	0.5264±0.0081(7)	0.5744±1.11E-16(4)	0.5797±1.11E-16(3)	0.5212±1.11E-16(8)	0.5691±0.0000(5)	**0.6778±0.0464**(1)
AC	Muta2	0.6142±0.0465(7)	0.5317±0.0296(8)	0.6182±0.1086(6)	0.6428±1.11E-16(3.5)	0.6428±1.11E-16(3.5)	0.6428±1.11E-16(3.5)	0.6428±1.11E-16(3.5)	**0.6826±0.1033**(1)
	Musk1	0.5278±0.0162(3)	**0.5615±0.0347**(1)	0.5119±0.0186(4)	0.5000±0.0000(6.5)	0.5000±0.0000(6.5)	0.5000±0.0000(6.5)	0.5000±0.0000(6.5)	0.5398±0.0338(2)
	Compon	0.7527±0.0062(3)	0.7227±0.1004(5)	0.7268±0.1066(4)	0.4902±0.0000(8)	0.8537±0.0000(2)	0.5069±1.11E-16(6.5)	0.5069±1.11E-16(6.5)	**0.8606±0.0026**(1)
	Process	0.8038±0.0012(4)	0.8341±0.1047(3)	0.8705±0.0624(2)	0.4967±0.0000(8)	0.5016±0.0000(7)	0.5040±0.0000(5.5)	0.5040±0.0000(5.5)	**0.8993±0.0046**(1)
	Function	0.6922±0.1224(5)	0.7466±0.1038(3)	0.7057±0.1202(4)	0.4948±0.0000(8)	**0.9000±0.0000**(1)	0.5012±0.0000(6.5)	0.5012±0.0000(6.5)	0.8873±0.0023(2)
	AvgR	3.6667	3.6667	3.8889	6.2222	5.0556	6.0556	5.8889	**1.5556**

Index	*Datasets*	*BAMIC-minH*	*BAMIC-maxH*	*BAMIC-avgH*	*CAN*	*PCAN*	*CLR-L_1*	*CLR-L_2*	*k-Mnv-Rep*
	Messidor	0.5199±0.0095(3)	0.5153±0.0148(4)	0.5208±0.0131(2)	———————(6.5)	———————(6.5)	———————(6.5)	———————(6.5)	**0.5282±0.0362**(1)
	Elephant	0.5885±0.0399(5)	0.6010±0.0311(4)	0.6058±0.0467(3)	**0.7564±0.0000** (1)	———————(8)	0.5721±1.11E-16(6.5)	0.5721±1.11E-16(6.5)	0.6361±0.0723 (2)
	Web2	0.5899±0.1104(2)	0.4746±0.1070(6)	0.4206±0.0404(7)	0.4944±0.0000 (4.5)	———————(8)	0.5011±1.11E-16(3)	0.4944±0.0000 (4.5)	**0.6526±0.0726**(1)
	Muta1	0.5821±0.0450(6)	0.4847±0.0694(8)	0.5377±0.0205(7)	0.6649±0.0000 (3)	0.6675±0.0000 (2)	0.6569±0.00E+00(5)	0.6623±1.11E-16(4)	**0.6782±0.0448**(1)
PE	Muta2	0.6006±0.0264(3)	0.6345±0.0452(2)	0.5490±0.1336(8)	0.5593±1.11E-16(5.5)	0.5593±1.11E-16(5.5)	0.5593±1.11E-16(5.5)	0.5593±1.11E-16(5.5)	**0.6531±0.1317**(1)
	Musk1	0.5289±0.0168(3)	**0.5632±0.0370**(1)	0.5121±0.0200(4)	0.5097±1.11E-16(6.5)	0.5097±1.11E-16(6.5)	0.5097±1.11E-16(6.5)	0.5097±1.11E-16(6.5)	0.5511±0.0609 (2)
	Compon	0.7758±0.0163(4)	0.7765±0.0578(3)	0.7731±0.0665(5)	0.0759±1.39E-17(8)	0.8539±0.0000 (2)	0.7517±1.11E-16(6.5)	0.7517±1.11E-16(6.5)	**0.8677±0.0017**(1)
	Process	0.8454±0.0007(3)	0.8427±0.0903(4)	0.8775±0.0516(2)	0.0414±0.0000 (8)	0.7504±0.0000 (7)	0.7510±1.11E-16(5.5)	0.7510±1.11E-16(5.5)	**0.9130±0.0033**(1)
	Function	0.7073±0.1326(5)	0.7605±0.1068(3)	0.7183±0.1237(4)	0.0588±6.94E-18(8)	**0.9002±1.11E-16**(1)	0.5836±1.11E-16(6.5)	0.5836±1.11E-16(6.5)	0.8979±0.0015 (2)
	AvgR	3.7778	3.8889	4.6667	5.6667	5.1667	5.7222	5.7778	**1.3333**

Index	*Datasets*	*BAMIC-minH*	*BAMIC-maxH*	*BAMIC-avgH*	*CAN*	*PCAN*	*CLR-L_1*	*CLR-L_2*	*k-Mnv-Rep*
	Messidor	0.5200±0.0095(2)	0.5153±0.0149(4)	**0.5208±0.0130**(1)	———————(6.5)	———————(6.5)	———————(6.5)	———————(6.5)	0.5193±0.0241(3)
	Elephant	0.5868±0.0385(4)	0.6005±0.0311(2)	0.5996±0.0412(3)	0.5250±1.11E-16(5)	———————(8)	0.5150±1.11E-16(6.5)	0.5150±1.11E-16(6.5)	**0.6035±0.0584**(1)
	Web2	0.5403±0.0709(2)	0.4960±0.0374(4)	0.4563±0.0286(7)	0.4926±1.11E-16(5.5)	———————(8)	0.5014±0.00E+00(3)	0.4926±1.11E-16(5.5)	**0.6717±0.0795**(1)
	Muta1	0.5898±0.0492(6)	0.4999±0.0484(8)	0.5417±0.0226(7)	0.6563±0.0000 (3)	0.6603±1.11E-16(2)	0.6242±1.11E-16(5)	0.6523±1.11E-16(4)	**0.6881±0.0517**(1)
RE	Muta2	0.6124±0.0260(3)	0.6234±0.0427(2)	0.5523±0.1304(4)	0.5503±0.0000 (6.5)	0.5503±0.0000 (6.5)	0.5503±0.00000 (6.5)	0.5503±0.0000(6.5)	**0.6360±0.1037**(1)
	Musk1	0.5284±0.0162(3)	**0.5596±0.0338**(1)	0.5115±0.0186(4)	0.5063±1.11E-16(6.5)	0.5063±1.11E-16(6.5)	0.5063±1.11E-16(6.5)	0.5063±1.11E-16(6.5)	0.5355±0.0333(2)
	Compon	0.7527±0.0062(3)	0.7227±0.1004(5)	0.7268±0.1066(4)	0.4902±0.0000 (8)	0.8537±0.0000 (2)	0.5069±1.11E-16(6.5)	0.5069±1.11E-16(6.5)	**0.8606±0.0026**(1)
	Process	0.8038±0.0012(4)	0.8341±0.1047(3)	0.8705±0.0624(2)	0.4967±0.0000 (8)	0.5016±0.0000 (7)	0.5040±0.0000 (5.5)	0.5040±0.0000(5.5)	**0.8993±0.0046**(1)
	Function	0.6922±0.1224(5)	0.7466±0.1038(3)	0.7057±0.1202(4)	0.4948±0.0000 (8)	**0.9000±0.0000** (1)	0.5012±0.0000 (6.5)	0.5012±0.0000(6.5)	0.8873±0.0023(2)
	AvgR	3.5556	3.5556	4.0000	6.3333	5.2778	5.8333	6.0000	**1.4444**

Index	*Datasets*	*BAMIC-minH*	*BAMIC-maxH*	*BAMIC-avgH*	*CAN*	*PCAN*	*CLR-L_1*	*CLR-L_2*	*k-Mnv-Rep*
	Messidor	0.0010±0.0015(3)	0.0008±0.0017(4)	0.0020±0.0032(2)	———————(6.5)	———————(6.5)	———————(6.5)	———————(6.5)	**0.0052±0.0091**(1)
	Elephant	0.0313±0.0258(4)	0.0394±0.0214(3)	0.0419±0.0358(2)	0.0020±4.34E-19(5)	———————(8)	-0.0001±0.0000(6.5)	-0.0001±0.0000(6.5)	**0.0526±0.0507**(1)
	Web2	**0.0544±0.0707**(1)	-0.0078±0.0578(3)	-0.0521±0.0218(7)	-0.0118±1.73E-18(5.5)	———————(8)	-0.0090±1.73E-18(4)	-0.0118±1.73E-18(5.5)	0.0539±0.1256(2)
	Muta1	0.0454±0.0255(2)	-0.0123±0.0212(7)	-0.0028±0.0011(6)	0.0013±2.17E-19(4)	0.0055±8.67E-19(3)	-0.0347±6.94E-18(8)	-0.0026±4.34E-19(5)	**0.1240±0.0617**(1)
ARI	Muta2	0.0342±0.0424(3)	-0.0405±0.0274(8)	0.0657±0.1282(2)	0.0282±3.47E-18(5.5)	0.0282±3.47E-18(5.5)	0.0282±3.47E-18(5.5)	0.0282±3.47E-18(5.5)	**0.1435±0.1449**(1)
	Musk1	-0.0067±0.0071(3)	**0.0095±0.0176**(1)	-0.0072±0.0041(4)	-0.0074±0.0000(6.5)	-0.0074±0.0000(6.5)	-0.0074±0.0000(6.5)	-0.0074±0.0000(6.5)	0.0021±0.0171(2)
	Compon	0.2548±0.0126(3)	0.2379±0.1796(5)	0.2505±0.1898(4)	0.0025±0.0000(6)	0.4998±0.0000(2)	0.0001±0.0000(7.5)	0.0001±0.0000(7.5)	**0.5197±0.0075**(1)
	Process	0.3689±0.0031(4)	0.4901±0.2247(3)	0.5643±0.1344(2)	0.0086±0.0000(5)	5.21E-06±0.0000(8)	5.21E-05±6.78E-21(6.5)	5.21E-05±6.78E-21(6.5)	**0.6375±0.0148**(1)
	Function	0.2068±0.1434(5)	0.2855±0.1506(3)	0.2261±0.1744(4)	0.0059±0.0000(6)	**0.6395±0.0000**(1)	-1.34E-05±0.0000(7.5)	-1.34E-05±0.0000(7.5)	0.5997±0.0072(2)
	AvgR	3.1111	4.1111	3.6667	5.5556	5.3889	6.5000	6.3333	**1.3333**

Index	*Datasets*	*BAMIC-minH*	*BAMIC-maxH*	*BAMIC-avgH*	*CAN*	*PCAN*	*CLR-L_1*	*CLR-L_2*	*k-Mnv-Rep*
	Messidor	0.0014±0.0011(3)	0.0013±0.0012(4)	0.0017±0.0018(2)	———————(6.5)	———————(6.5)	———————(6.5)	———————(6.5)	**0.0047±0.0063**(1)
	Elephant	0.0271±0.0197(5)	0.0324±0.0156(4)	0.0376±0.0304(3)	**0.0620±6.94E-18**(1)	———————(8)	0.0057±0.0000(6.5)	0.0057±0.0000(6.5)	0.0586±0.0465(2)
	Web2	0.0379±0.0560(2)	0.0263±0.0189(3)	0.0244±0.0191(4)	0.0001±0.0000(5.5)	———————(8)	5.42E-06±8.47E-22(7)	0.0001±0.0000(5.5)	**0.1350±0.0644**(1)
	Muta1	0.0290±0.0135(6)	0.0138±0.0199(7)	0.0062±0.0038(8)	0.0936±1.39E-17(3)	0.0972±1.39E-17(2)	0.0781±1.39E-17(5)	0.0901±1.39E-17(4)	**0.1165±0.0441**(1)
NMI	Muta2	0.0372±0.0118(4)	0.0713±0.0188(2)	0.0645±0.0632(3)	0.0099±1.73E-18(6.5)	0.0099±1.73E-18(6.5)	0.0099±1.73E-18(6.5)	0.0099±1.73E-18(6.5)	**0.1147±0.1134**(1)
	Musk1	0.0031±0.0052(3)	**0.0149±0.0131**(1)	0.0015±0.0036(4)	0.0002±0.0000(6.5)	0.0002±0.0000(6.5)	0.0002±0.0000(6.5)	0.0002±0.0000(6.5)	0.0140±0.0317(2)
	Compon	0.2255±0.0215(5)	0.2264±0.1216(4)	0.2281±0.1337(3)	0.0771±1.39E-17(6)	0.4001±0.0000(2)	0.0285±3.47E-18(7.5)	0.0285±3.47E-18(7.5)	**0.4367±0.0048**(1)
	Process	0.3752±0.0021(4)	0.4092±0.1795(3)	0.4759±0.0959(2)	0.0842±0.0000(5)	0.0122±0.0000(8)	0.0207±0.0000(6.5)	0.0207±0.0000(6.5)	**0.5914±0.0114**(1)
	Function	0.1823±0.1261(5)	0.2439±0.1280(3)	0.1935±0.1483(4)	0.0719±0.0000(6)	**0.5318±0.0000**(1)	0.0016±2.17E-19(7.5)	0.0016±2.17E-19(7.5)	0.5307±0.0045(2)
	AvgR	4.1111	3.4444	3.6667	5.1111	5.3889	6.6111	6.3333	**1.3333**

To give a comprehensive comparison, we use the Friedman test and Bonferroni-Dunn test [11] to analyze the differences of the eight compared algorithms. According to AvgR we can get the average rank of these algorithms for all cases. Suppose that r_i^j represents the rank of the jth algorithm on the ith case, the average rank of the jth algorithm $R_j = \frac{1}{B}\sum_{i=1}^{B} r_i^j$, where B represent the number of cases. The Friedman test compares the average ranks of algorithms. As there are $A = 8$ algorithms and $B = 5$ cases (i.e., 5 external criterions), the average rank R_j $(1 \leq j \leq 8)$ can be computed shown in Table 4. According to the average ranks, we still know the k-Mnv-Rep algorithm is better than the other seven algorithms.

Table 4: The average rank of these algorithms on five cases.

	BAMIC-minH	*BAMIC-maxH*	*BAMIC-avgH*	*CAN*	*PCAN*	*CLR-L_1*	*CLR-L_2*	*k-Mnv-Rep*
R_j	3.6445	3.7333	3.9778	5.7778	5.2556	6.1444	6.0667	**1.4**

Under the null-hypothesis, all algorithms are equivalent and so their ranks should be equal (i.e., $R_j = 4.5$ for eight algorithms). The Friedman test aims to check whether the measured average ranks are significantly different from the mean rank $R_j = 4.5$ expected under the null-hypothesis:

$$\begin{aligned}\chi_F^2 &= \frac{12B}{A(A+1)}\left[\sum_{j=1}^{A} R_j^2 - \frac{A(A+1)^2}{4}\right] \\ &= \frac{12\cdot 5}{8\cdot(8+1)}\left[3.6445^2 + 3.7333^2 + 3.9778^2 + 5.7778^2\right. \\ &\quad \left. +5.2556^2 + 6.1444^2 + 6.0667^2 + 1.4^2 - \frac{8\cdot(8+1)^2}{4}\right] \\ &\approx 15.47.\end{aligned}$$

With the eight algorithms, the Friedman statistic is distributed according to the χ_F^2 distribution with $A - 1 = 7$ degrees of freedom. The critical value is $14.07 < 15.47$, so we reject the null-hypothesis and we think the compared eight algorithms have significant differences.

Then, we use the Bonferroni-Dunn test to reveal the differences. The critical value is 2.450 when we use $\alpha = 0.1$ according to [11]. So the critical difference for nine data sets can be computed as $CD = q_\alpha\sqrt{\frac{A(A+1)}{6N}} = 2.450\cdot\sqrt{\frac{8\cdot(8+1)}{6\cdot 9}} \approx 2.82$. N is the number of data sets. By the critical difference we can identify the algorithms for all cases. If the difference of the average rank is larger than the critical difference CD for two algorithms, we think they are significantly different. Fig.1 shows the Bonferroni-Dunn test of eight algorithms in terms of five evaluation indexes. The circles

represent the average ranks of algorithms and the length of the bar is the critical difference CD.

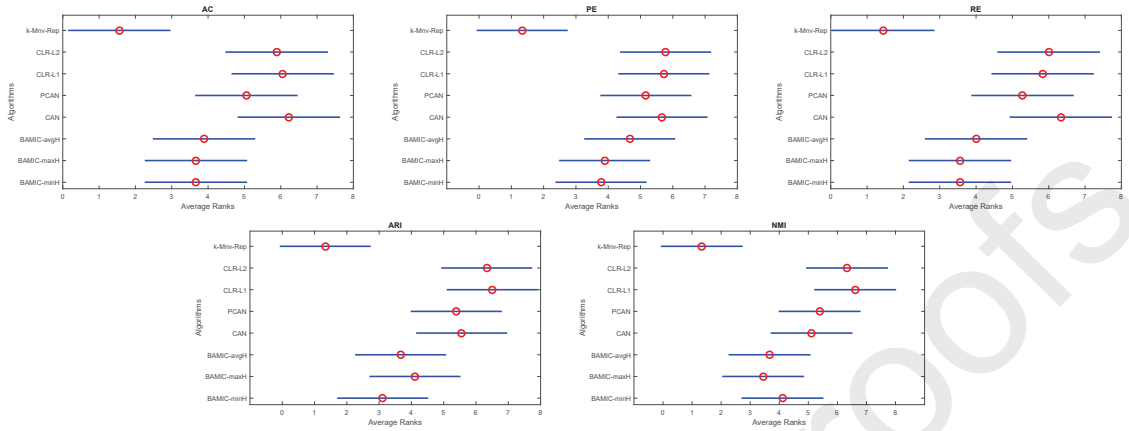


Fig. 1: The Bonferroni-Dunn test of the k-Mnv-Rep algorithm in terms of five indexes.

From Fig.1, we can find that the k-Mnv-Rep algorithm has significant difference with the CAN, PCAN, CLR-L_1, CLR-L_2 algorithms on all indexes, and it can be differentiated with the BAMIC-minH, BAMIC-maxH, BAMIC-avgH algorithms. However, the BAMIC-minH, BAMIC-maxH, BAMIC-avgH algorithms almost have not differences, and the CAN, PCAN, CLR-L_1, CLR-L_2 algorithms are almost not differentiated. Furthermore, according to the average ranks, we can learn about the k-Mnv-Rep performs the best. Above all, the k-Mnv-Rep algorithm outperforms other compared algorithms because it has higher ranks and bigger differences with those algorithms.

4.3. Experiments on hybrid matrix-object data

Some experiments on hybrid matrix-object data are conducted to evaluate the effectiveness of the k-Mv-Rep algorithm. Firstly, the real hybrid matrix-object data set is introduced. Then, the comparison results of the k-Mv-Rep algorithm and the k-prototypes algorithm are given.

4.3.1. The real hybrid data set

As the lack of the public hybrid matrix-object data sets, we applied the data set Papers [5] to evaluate the k-Mv-Rep algorithm. The data set Papers provided by Aminer describes the information of the papers from some authors before 2016. It is described by 6 attributes, *Index*, *Title*,

[5] https://biendata.com/user/login/?next=/competition/scholar/data

Authors, *Year*, *Journal*, *Index1*. *Index1* represents the papers that cites the paper *Index*. Obviously, the first author for each paper may publish more than one paper. Suppose that we consider each the first author as an object, we use the detail records of all his or her papers to describe the object. Furthermore, we use the number of the papers citing each paper to replace the last attribute *Index1*. Thus, each author is described by five attributes, *Index*, *Title*, *Year*, *Journal*, *Cites*, and has more than one records. We delete the attribute *Index* because the values on it are different each other. As the attributes *Title*, *Year*, *Journal* are three categorical attributes while *Cites* is a numeric attribute, the new data set can be considered as a hybrid matrix-object data set and is named as Authors.

Although the data set transformed is a matrix-object data set, it has not label information. To evaluate the proposed algorithm, we need to preprocess the data set to obtain its structure. The multidimensional scaling technique [30] is applied to visualize the data. The main goal of the technique is to obtain a configuration of n points (rows) in P dimensions (cols) by passing the n-by-n distance matrix obtained by Eq.(12) to the function *mdscale* from MATLAB. The Euclidean distances between n points approximate a monotonic transformation of the corresponding dissimilarities in the n-by-n distance matrix. Therefore, we can visualize n points to reflect the distribution of the data. To visualize the data, we set $P = 2$. In most cases, the distribution of a real data set is generally disordered. By the visualization technology, we can delete some points to get the relative clear structure of the data.

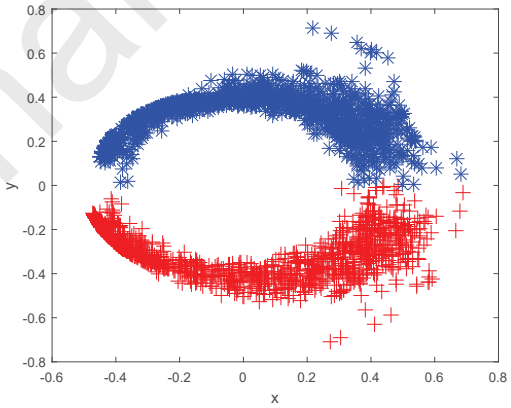


Fig. 2: The distribution of the data set Authors.

For the data set Authors, we select objects that are in the range of $x < 0.55, y > 0.16$ or $x < 0.55, y < -0.16$ in the coordinate system as a new set after the visualization of initial data set, then visualize the new data in Fig.2. From the visual figure, we can intuitively find the number of clusters and obtain the label information of each matrix-object. The Authors is listed in Table 5.

Table 5: The details of the data set Authors.

Objects	*Attributes*		*Records*	*Clusters*
	Categorical	*Numeric*		
2533	3	1	40624	2

4.3.2. Comparison results

To our best knowledge, some existed clustering algorithms can not process the hybrid matrix-object data directly. The k-prototypes algorithm [17] is applied in this subsection. As each matrix-object is a matrix instead of a vector, we need to transform the matrix-object data set into the required form of the k-prototypes algorithm. For a hybrid matrix-object, we use the mode to represent the attribute values on the categorical attributes and use the mean value to denote the attribute values on the numeric attributes. Thus, a hybrid matrix-object is transformed as a vector and the matrix-object data set can be clustered by the k-prototypes algorithm.

We execute the k-prototypes and the k-Mv-Rep algorithms 50 times respectively, then take the mean value of the experimental results as the final results. In the k-Mv-Rep algorithm the parameter ϵ is set to 0.2 while in the k-prototypes algorithm γ is set to the average standard deviation of all numeric attributes according to [17]. Table 6 shows the comparison results of the two algorithms on the data set Authors.

Table 6: Comparison results of different algorithms on the data set Authors.

Algorithms	*AC*	*PE*	*RE*	*ARI*	*NMI*
k-prototypes	0.5349±0.0159	0.5436±0.0176	0.5344±0.0157	0.0055±0.0049	0.0055±0.0041
k-Mv-Rep	0.6534±0.1080	0.6826±0.1131	0.6541±0.1076	0.1405±0.1616	0.1340±0.1382

We can see from Table 6 that the k-Mv-Rep algorithm is better than the k-prototypes algorithm on the five evaluation indexes. Furthermore, the AC of the proposed algorithm is approximately

13% more than it of the k-prototypes algorithm. Therefore, it can be seen that the k-Mv-Rep algorithm is better than the k-prototypes algorithm.

4.4. Impact of ϵ

In the two proposed algorithms, the parameter ϵ is given as a control condition of the program termination. The program will be stop when the value of the objective function changes less than ϵ. Therefore, different ϵ may result in different clustering results. How to decide the size of ϵ is a very important problem. In this subsection, we executed the two algorithms 30 times respectively with different ϵ, from 0.05 to 0.35 with step 0.05, on the corresponding experimental data sets and recorded the means of accuracy (AC) and iterations. The details are shown in Figs.3-4. The bold

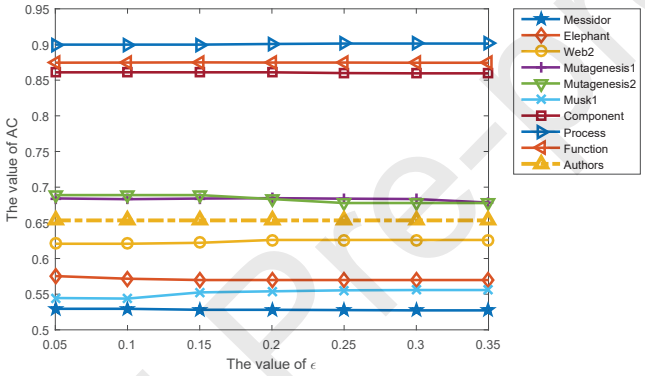


Fig. 3: The AC of the ten data with different ϵ.

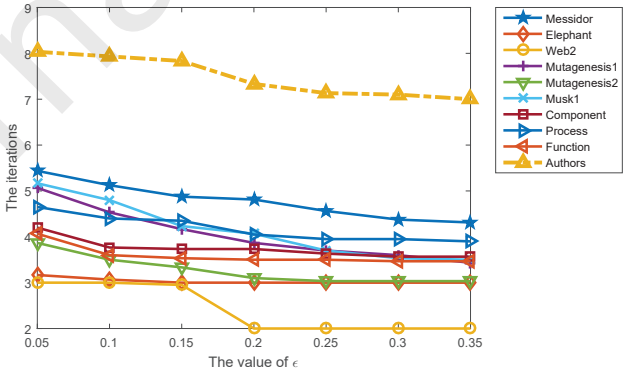


Fig. 4: The iterations of the ten data with different ϵ.

dash dot line depicts the test results of the k-Mv-rep algorithm on the data set Authors. Those solid lines are obtained by the k-Mnv-rep algorithm on the nine data sets.

From Fig.3, we can observe that the AC on the ten data sets has some slight fluctuations with the increasing of ϵ, but overall they are relatively stable. Meanwhile, from Fig.4, we can see clearly that the iterations on the ten data all show a decreasing trend on the whole with the increasing of ϵ. Mostly, the iterations decrease slowly relatively when $\epsilon > 0.2$. In the two proposed algorithms, higher AC and fewer iterations are expected in each experiment. Combining the AC and the iterations, we set $\epsilon = 0.2$.

4.5. Test of convergence

To test the convergence of the proposed algorithms, we refer to some algorithms [32], [33], [43], [42]. As a heuristic algorithm is proposed to update cluster centers in this paper, we give the convergence test of the k-Mnv-rep algorithm and k-Mv-rep algorithm to ensure the reasonability of the heuristic algorithm. In each experiment, we capture the objective function value of each iteration and draw them into a line. Figs.5-6 show the convergence test of algorithms on ten experimental data sets. We can see that the objective function value is decreasing with the iterations increasing.

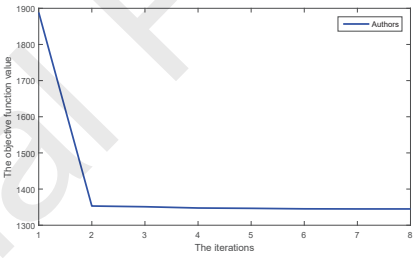


Fig. 5: The changes of the objective function value against the iterations for the k-Mv-rep algorithm.

5. Scalability study on synthetic data

In this section, we make a scalability test of the k-Mnv-Rep algorithm on synthetic data. The algorithm used to generate numeric matrix-object synthetic data is proposed, and the scalability of the k-Mnv-Rep algorithm on synthetic data sets is demonstrated.

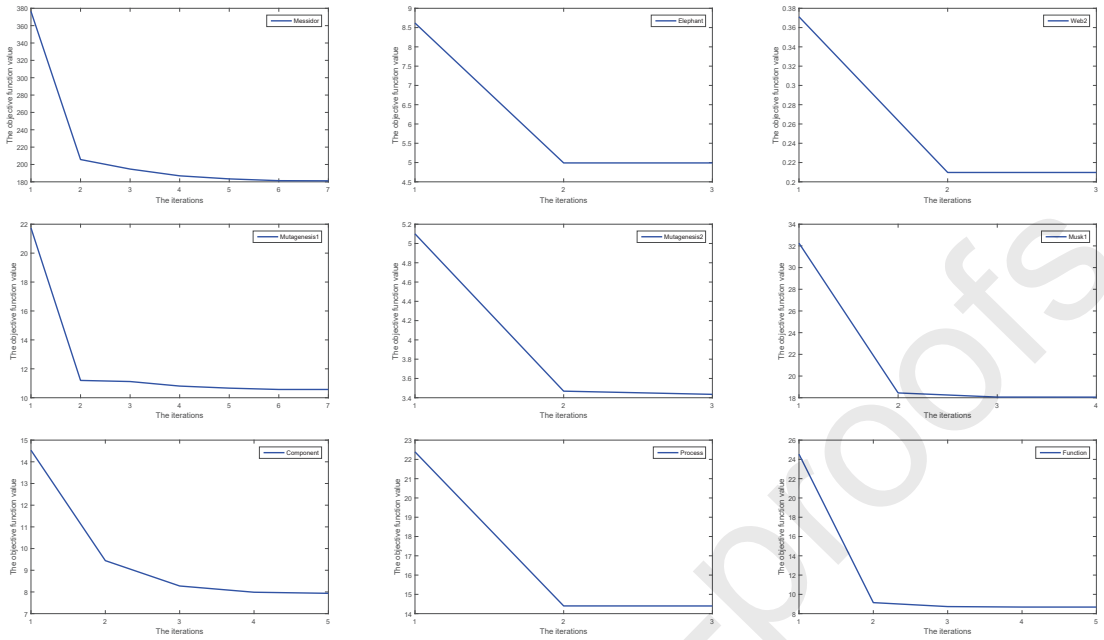


Fig. 6: The changes of the objective function value against the iterations for the k-Mnv-rep algorithm.

5.1. The generating method of synthetic matrix-object data sets

Let $\mathbf{X}$ denote a set of n matrix-objects $\{X_1, X_2, \ldots, X_n\}$ to be generated with m attributes $\{A_1, A_2, \ldots, A_m\}$. We use the following parameters to generate the synthetic data set $\mathbf{X}$ with k clusters $C = \{C_1, C_2, \ldots, C_k\}$, and we make each cluster identifiable from other clusters:

- k: the number of clusters to be generated in $\mathbf{X}$;
- L: the maximum number of records in each matrix-object;
- n_l: the number of matrix-objects in $C_l (1 \leq l \leq k)$;
- m_{ls}, s_{ls}: the mean value and standard deviation of C_l on the sth attribute;

To simplicity, we set n_l to be the same value when $1 \leq l \leq k$. Then the records in C_l can be generated to satisfy the distribution of mean value m_{ls} and standard deviation s_{ls} on the sth attribute. In the same way, the records in $C_{l'} (1 \leq l' \leq k, l' \neq l)$ can be generated. The number of records is set to the maximum value $n_l \times L$ for each cluster. In addition, the ratio 8:2 is applied to select records in each matrix-object that makes the distribution of data become more reasonable in real world. In this way, most of records in an object meet the same distribution while a small

number of records may be from other clusters. To generate a matrix-object X_i in the cluster C_l, we perform the following steps.

1) Randomly select a value in $\{1, 2, \ldots, L\}$ as the number of records of X_i (written as r_i);

2) Randomly select $80\% \times r_i$ records from C_l as a part of records of X_i;

3) Randomly select $20\% \times r_i$ records from $\{C_1, C_2, \ldots, C_k\}$ as the rest of records of X_i.

4) Repeat Step 1, 2, 3 to generate all matrix-objects for the cluster C_l and assign the cluster label to all objects in the cluster.

Repeat the above steps to generate objects for other clusters with different parameters l, m_{ls}, s_{ls}.

5.2. Scalability study

We test the scalability of the k-Mnv-Rep algorithm with the changing of the number of objects, attributes, clusters, and records of the data set. The number of records can be described by the maximum number of records in each matrix-object. The higher the maximum value is, the more the number of records is. A total of 20 synthetic data sets were generated. In each scalability experiment, the same data set was utilized 20 times, and the execution time was the average of 20 runs. The experiments were conducted on a PC with an Intel Core i7 CPU (3.6 GHz) and 8 GB of memory. The experimental results are reported based on four experiments below.

Experiment 1: In this experiment, we set n=1000, m=10, k=2, and L varies from 10 to 50 with a step length of 10. In addition, attribute values are almost different for each record. Fig.7 shows the scalability of the k-Mnv-Rep algorithm against the number of records. We can see that this algorithm was linearly scalable to the number of records. Therefore, the k-Mnv-Rep algorithm can efficiently cluster data sets with a large number of records.

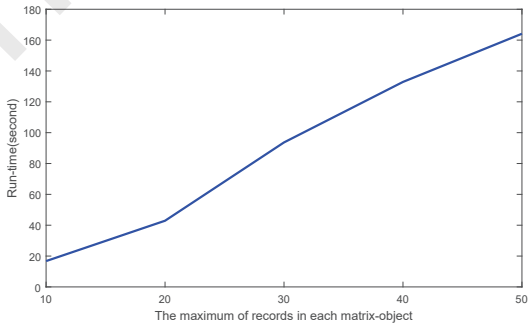


Fig. 7: The scalability of the k-Mnv-Rep algorithm against the maximum of records in each object.

Experiment 2: In this experiment, we set n=1000, L=10, k=2, and m varies from 10 to 50 with a step length of 10. Fig.8 shows the scalability of the k-Mnv-Rep algorithm against the number of attributes. We can see that this algorithm was linearly scalable to the number of attributes. Therefore, the k-Mnv-Rep algorithm can efficiently cluster high-dimensional data sets.

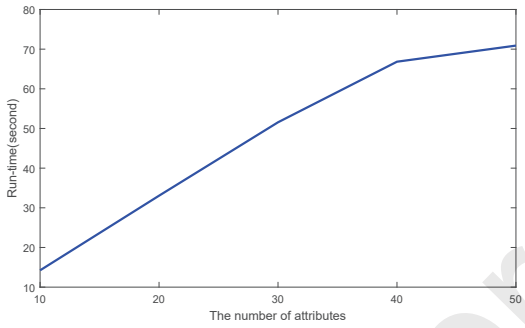


Fig. 8: The scalability of the k-Mnv-Rep algorithm against the number of attributes.

Experiment 3: In this experiment, we set n=1000, L=10, m=10, and k varies from 2 to 10 with a step length of 2. Fig.9 shows the scalability of the k-Mnv-Rep algorithm against the number of clusters. We can see that this algorithm was linearly scalable to the number of clusters. Therefore, the k-Mnv-Rep algorithm scaled well with the number of clusters.

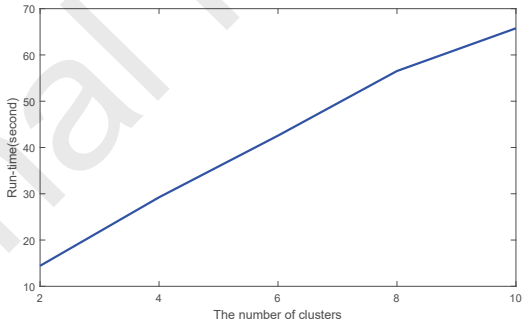


Fig. 9: The scalability of the k-Mnv-Rep algorithm against the number of clusters.

Experiment 4: In this experiment, we set L=10, m=10, k=2, and n varies from 1000 to 5000 with a step length of 1000. Fig.10 shows the scalability of the k-Mnv-Rep algorithm against the number of matrix-objects. We can see that this algorithm was linearly scalable to the number of matrix-objects. Therefore, the k-Mnv-Rep algorithm can efficiently cluster a large number of

matrix-objects.

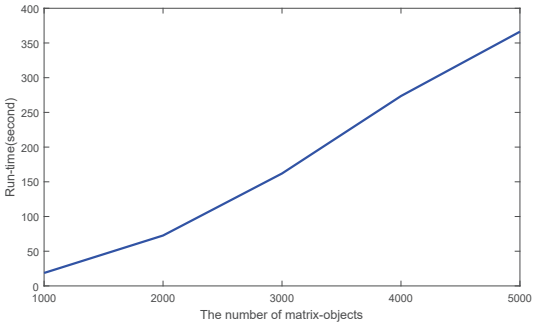


Fig. 10: The scalability of the k-Mnv-Rep algorithm against the number of matrix-objects.

6. Conclusions and future work

The matrix-object data sets extensively exist in numerous engineering applications. How to discover the structure of this kind of data sets has a significance on mining and analyzing their behavior characteristics. In this paper, we propose a new dissimilarity measure between two numeric matrix-objects, design a heuristic cluster center identification method and develop the k-Mnv-Rep algorithm to cluster numeric matrix-object data sets. Furthermore, we investigate the k-Mv-Rep algorithm to implement the clustering of the hybrid matrix-object data sets. The proposed algorithms are compared with some existing algorithms and their convergence is tested on real-world data sets. The experimental results have demonstrated the benefits of these two novel algorithms. We also make some scalability study on numeric synthetic matrix-object data, which shows that the k-Mnv-Rep algorithm can efficiently process data sets with a large number of records, matrix-objects, attributes and clusters. Note that the parameter ε_s for computing neighbors of attribute values is usually obtained by one half of standard deviation of $\mathbf{X}$ on the sth attribute. Our future work focuses on how to automatically acquire its best value according to the distribution of data.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (under grants 61976128, 61773247), the Innovation Project of excellent postgraduates in Shanxi Province (under grant 2019BY002), the Key Research and Development Projects of Shanxi Province (under grant

201803D31022), the Fund Program for the Scientific Activities of Selected Returned Overseas Professionals in Shanxi Province (under grant 2016-001), the Research Project Supported by Shanxi Scholarship Council of China (under grant 2016-003) and the 1331 Engineering Project of Shanxi Province, China.

References

[1] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. *Advances in neural information processing systems*, 15:561–568, 2002.

[2] K. Bache and M. Lichman. UCI machine learning repository, 2014. `http://archive.ics.uci.edu/ml`.

[3] F. Cao, Z. Huang, J. Liang, X. Zhao, Y. Meng, and Y. Qian. An algorithm for clustering categorical data with set-valued features. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4593–4606, 2018.

[4] F. Cao, J. Liang, D. Li, and X. Zhao. A weighting k-modes algorithm for subspace clustering of categorical data. *Neurocomputing*, 108(5):23–30, 2013.

[5] F. Cao, L. Yu, Z. Huang, and J. Liang. k-mw-modes: An algorithm for clustering categorical matrix-object data. *Applied Soft Computing*, 57:605–614, 2017.

[6] L. Chen, S. Wang, K. Wang, and J. Zhu. Soft subspace clustering of categorical data with probabilistic distance. *Pattern Recognition*, 51:322–332, 2016.

[7] Y. Chen, J. Bi, and Z. Wang. Miles: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):1931–1947, 2006.

[8] Y. Chen and Z. Wang. Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, 5:913–939, 2004.

[9] V. Cheplygina, D. Tax, and M. Loog. Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48(1):264–275, 2015.

[10] E. Decencière, X. Zhang, G. Cazuguel, B. Lay, B. Cochener, C. Trone, P. Gain, R. Ordonez, P. Massin, A. Erginay, B. Charton, and J. Klein. Feedback on a publicly distributed image database: the Messidor database. *Image Analysis and Stereology*, pages 231–234, 2014.

[11] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006.

[12] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1-3):9–33, 2004.

[13] G. Edgar. *Measure, Topology, and Fractals Geometry*. Springer New York, 2008.

[14] J. Foulds. Learning instance weights in multi-instance learning. *Master¡¯s thesis, University of Waikato*, 2009.

[15] Y. Gao and K. Xu. Prankaggreg: A fast clustering based partial rank aggregation. *Information Sciences*, 478:408–421, 2019.

[16] J. Han, M. Kamber, and J. Pei. Data mining: Concepts and techniques. *Data Mining Concepts Models Methods and Algorithms Second Edition*, 5(4):1–18, 2011.

[17] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[18] Z. Huang and M. K. Ng. A fuzzy k-modes algorithm for clustering categorical data. *IEEE Transactions on Fuzzy Systems*, 7(4):446–452, 1999.

[19] Z. Huang, M. K. Ng, H. Rong, and Z. Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):657–668, 2005.

[20] A. K. Jain and R. C. Dubes. Algorithms for clustering data. *Technometrics*, 32(2):227–229, 1988.

[21] F. James and F. Eibe. A review of multi-instance learning assumptions. *Knowledge Engineering Review*, 25(1):1–25, 2010.

[22] M. Kandemir and F. A. Hamprecht. Computer-aided diagnosis from weak supervision: A benchmarking study. *Computerized Medical Imaging and Graphics*, 42:44–50, 2015.

[23] L. Kaufmann and P. Rousseeuw. Clustering by means of medoids. In *Statistical Data Analysis Based on the L_1-norm Related Methods*, pages 405–416, 1987.

[24] K. M. Kumar and A. R. Reddy. An efficient k-means clustering filtering algorithm using density based initial cluster centers. *Information Sciences*, 418-419:286–301, 2017.

[25] J. Liang, L. Bai, C. Dang, and F. Cao. The k-means-type algorithms versus imbalanced data distributions. *IEEE Transactions on Fuzzy Systems*, 20(4):728–745, 2012.

[26] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[27] F. Nie, X. Wang, and H. Huang. Clustering and projected clustering with adaptive neighbors. In *Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, pages 977–986, 2014.

[28] F. Nie, X. Wang, M. I. Jordan, and H. Huang. Constrained laplacian rank algorithm for graph-based clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1969–1976, 2016.

[29] S. Ray and M. Craven. Learning statistical models for annotating proteins with function information using biomedical text. *BMC bioinformatics*, 6(Suppl 1):S18, 2005.

[30] S. Schiffman, L. Reynolds, and F. Young. *Introduction to Multidimensional Scaling: Theory, Methods, and Applications*. Academic Press, 1981.

[31] S. Scott, J. Zhang, and J. Brown. On generalized multiple-instance learning. *International Journal of Computational Intelligence and Applications*, 5(01):21–35, 2005.

[32] R. Shang, W. Wang, R. Stolkin, and L. Jiao. Non-negative spectral learning and sparse regression-based dual-graph regularized feature selection. *IEEE Transactions on Cybernetics*, 48(2):793–806, 2018.

[33] R. Shang, Z. Zhu, L. Jiao, W. Wang, and S. Yang. Global discriminative-based nonnegative spectral clustering. *Pattern Recognition*, 55:172–182, 2016.

[34] A. Srinivasan, S. Muggleton, and R. King. Comparing the use of background knowledge by inductive logic programming systems. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, pages 199–230, 1995.

[35] A. Strehl and J. Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2003.

[36] Q. Tao, S. Scott, N. V. Vinodchandran, and T. T. Osugi. Svm-based generalized multiple-instance learning via approximate box counting. In *International Conference on Machine Learning*, page 101, 2004.

[37] G. D. Thomas and H. L. Richard. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

[38] J. Wang and J. D. Zucker. Solving the multiple-instance problem: A lazy learning approach. In *Seventeenth International Conference on Machine Learning*, pages 1119–1126, 2000.

[39] N. Weidmann, E. Frank, and B. Pfahringer. A two-level learning method for generalized multi-instance problems. In *Proceedings of the 14th European Conference on Machine Learning*, pages 468–479, 2003.

[40] R. Xu and D. Wunsch. *Clustering*. John Wiley and Son, 2008.

[41] X. Xu. Statistical learning in multiple instance problems. *Masterj¯s thesis, University of Waikato*, 2003.

[42] M. Yang, R. Shang, L. Jiao, W. Zhang, and S. Yang. Dual-graph regularized non-negative matrix factorization with sparse and orthogonal constraints. *Engineering Applications of Artificial Intelligence*, 69:24–35, 2018.

[43] M. Yang, R. Shang, L. Jiao, W. Zhang, Y. Yuan, and S. Yang. Feature selection based dual-graph sparse non-negative matrix factorization for local discriminative clustering. *Neurocomputing*, 290:87–99, 2018.

[44] M. Zhang and Z. Zhou. Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, 31(1):47–68, 2009.

[45] Z. Zhou, K. Jiang, and M. Li. Multi-instance learning based web mining. *Applied Intelligence*, 22(2):135–147, 2005.