Knowledge-Based Systems 163 (2019) 416–428

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A stratified sampling based clustering algorithm for large-scale data

Xingwang Zhao [a,b], Jiye Liang [a,*], Chuangyin Dang [b]

[a] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan, 030006, Shanxi, China
[b] Department of Systems Engineering and Engineering Management, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 23 May 2018
Received in revised form 3 September 2018
Accepted 6 September 2018
Available online 10 September 2018

Keywords:
Large-scale data
Fuzzy *c*-means algorithm
Stratified sampling
Data labeling

ABSTRACT

Large-scale data analysis is a challenging and relevant task for present-day research and industry. As a promising data analysis tool, clustering is becoming more important in the era of big data. In large-scale data clustering, sampling is an efficient and most widely used approximation technique. Recently, several sampling-based clustering algorithms have attracted considerable attention in large-scale data analysis owing to their efficiency. However, some of these existing algorithms have low clustering accuracy, whereas others have high computational complexity. To overcome these deficiencies, a stratified sampling based clustering algorithm for large-scale data is proposed in this paper. Its basic steps include: (1) obtaining a number of representative samples from different strata with a stratified sampling scheme, which are formed by locality sensitive hashing technique, (2) partitioning the chosen samples into different clusters using the fuzzy *c*-means clustering algorithm, (3) assigning the out-of-sample objects into their closest clusters via data labeling technique. The performance of the proposed algorithm is compared with the state-of-the-art sampling-based fuzzy *c*-means clustering algorithms on several large-scale data sets including synthetic and real ones. The experimental results show that the proposed algorithm outperforms the related algorithms in terms of clustering quality and computational efficiency for large-scale data sets.

1. Introduction

Clustering is an exploratory data analysis tool for discovering the underlying groups in the data. Its aim is to divide a set of unlabeled objects into natural groups so that the data objects within each group share some similarity and the data objects across different groups are different [1]. Clustering analysis has numerous applications in such areas as customer segmentation, target marketing, bioinformatics, social network analysis, and scientific data analysis. In the past five decades, various clustering algorithms have been developed in the literature in many different scientific disciplines. We refer the reader to surveys of clustering algorithms and applications [2–4].

With the development of the internet of things, cloud computing, and social networks, there has been a rapid increase in the volume of data [5]. The analysis of this large-scale data necessitates highly scalable clustering techniques [6–8]. However, the traditional clustering algorithms cannot be directly applied to large-scale data because of their high computation time. Therefore, the biggest and most important challenge for large-scale data clustering is how to improve the computational efficiency of the clustering algorithms while maintaining the clustering quality. In the last few decades, researchers have proposed some accelerated clustering algorithms to cope with increasing scale of the data set [9,10]. At the high level, there are two kinds of pervasive solutions for large-scale data clustering: parallel/distributed computation and data reduction schemes. This paper will mainly focus on the latter solution.

In data reduction schemes, sampling is a systematic and cost-effective way of reducing the data size while maintaining the essential properties of data. A general framework for performing large-scale data clustering based on sampling is shown in Fig. 1. Concretely speaking, the framework includes the following procedures: (1) sampling a number of representative objects from the original large-scale data, (2) generating the partial clustering results with traditional clustering algorithms on the sampled data, and (3) attaining the total clustering results by labeling remaining data objects with the partial clustering results. The key points of sampling-based clustering algorithms are how to design an appropriate sampling scheme for choosing representative objects. They need to maintain the distribution characteristics of the data set as much as possible. In order to achieve this goal, many clustering methods have been developed by combining traditional algorithms with sampling. Based on the difference of the sampling scheme, these methods can be classified as clustering algorithms

* Corresponding author.
E-mail addresses: zhaoxw84@163.com (X. Zhao), ljy@sxu.edu.cn (J. Liang), mecdang@cityu.edu.hk (C. Dang).

https://doi.org/10.1016/j.knosys.2018.09.007
0950-7051/

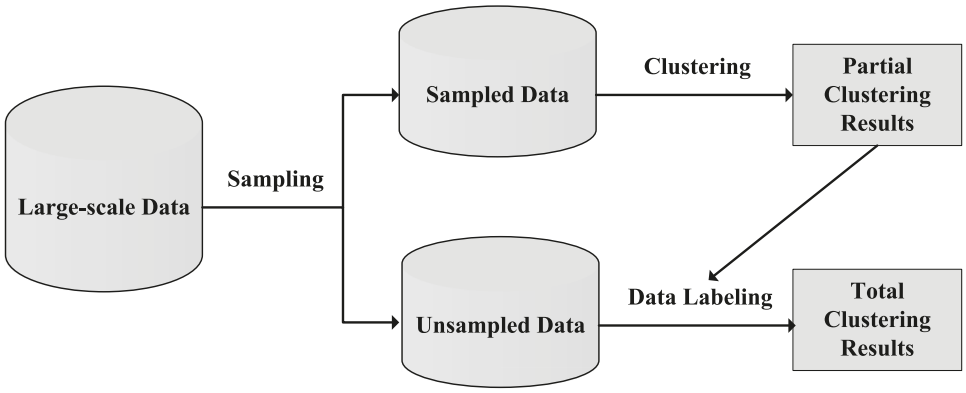

Fig. 1. A general framework of sampling-based clustering algorithm for large-scale data.

with uniform random sampling, progressive sampling, biased sampling and stratified sampling.

In uniform random sampling, every interesting object has the same probability to be sampled to form a representative subsample [11]. This sampling technique has been widely used in clustering algorithms to accelerate large data analysis. For example, the CURE (*Clustering Using REpresentatives*) clustering algorithm is a hierarchical technique which uses uniform random sampling to improve the computational efficiency [12]. This method firstly draws a sample of the data set randomly and the new hierarchical clustering is carried on the sampled data. The CLARANS (*Clustering Large Applications based on Randomized Sampling*) clustering algorithm also relies on uniform sampling technique to decrease the search space and further speed up the computation [13]. The RSEFCM (*Random Sampling plus Extension Fuzzy C-Means*) algorithm forms a subsample of the original data with random uniform sampling firstly and then uses the fuzzy c-means clustering algorithm [14]. After yielding the partial cluster centers, each unsampled object can be classified according to the cluster membership. The empirical evaluation of the above mentioned uniform sampling-based algorithms shows them to outperform clustering on the original entire data in terms of both time and space efficiency. Unfortunately, uniform random sampling is not always practical, due to a poor match between the sampling design and the structure of data [15].

As an important sampling scheme, progressive sampling has been applied to clustering analysis. In progressive sampling, an initial small sample of data is used to form a clustering result. The size of the subsample is increased gradually, with a new clustering result created each time and the subsample grows in size. When a stopping criterion is meet, the technique terminates. Domingos et al. [16] used Hoeffding bounds in a progressive sampling technique to both estimate the initial sample size and judge if the sample size at any point in the progression was sufficient. Wang et al. [17] used progressive sampling to select a subsample that accurately represents the data set. A divergence test was used to assess if the subsample matches the distribution of the data set. If the test failed, progressively larger subsamples were taken until the test passed. Finally a clustering algorithm was run on the chosen subsample. Parker and Holl [18] proposed a geometric progressive fuzzy c-means (GOFCM) that leverages progressive sampling, Thompson's method and a different stopping criterion. Concretely speaking, the initial subsample size is estimated using Thompson's method. The size of subsequent subsamples are calculated using a geometric schedule. The calculated size of the subsample stops growing once it exceeds a user provided value. For large-scale data clustering, one of the key principles of progressive sampling is to design a reasonable termination condition.

Different from the above sampling method, the process of biased sampling takes into account the probability information that a given point will be included in the sample. When probability information is not taken into account the sampling, it can be reduced to the uniform random sampling. Therefore, it may be considered as a generalization of uniform random sampling [19]. Since the large-scale data usually have multi-scale densities and arbitrary shape-based clusters, density-biased sampling is an appropriate method to preserve the density. Biased sampling has been used extensively to clustering analysis, in which the sampling probability is related to the neighborhood density [15,20]. Kollios et al. developed a clustering algorithm based on density-biased sampling [15]. In this algorithm, whether a data point is sampled or not depends on its kernel density. Palmer et al. proposed the biased sampling-based algorithm using the hash technique [20]. This method firstly divides the data space into equal-sized cells, and stores the data points in the hash tables. Thereafter, sampling is biased by cell density. The cells with few data points are over-sampled, whereas the cells with many data points are under-sampled. Experimental results in [15,20] show that the above methods improve the clustering quality compared to uniform random sampling. However, the computation of the probability information requires high execution time, and may become infeasible for large-scale data clustering.

As a well-known sampling technique, stratified sampling conducted in two steps, dividing the whole data set into disjoint groups called strata, and then randomly sampling within each strata to select the representative objects [21]. In order to generate diversity and informative feature subsets for dimensional data ensemble clustering, the authors firstly cluster the high-dimensional features into a few feature groups called feature strata, and then use stratified sampling to randomly sample features separately from feature strata to form the feature subsets [22]. The experimental results have shown that the ensemble clustering results with stratified sampling outperform the results with random sampling. In all, compared with uniform random sampling, progressive sampling and biased sampling, stratified sampling has both higher computational efficiency and better clustering quality, which can be considered as a generalization of biased sampling [23].

In order to increase the efficiency and effectiveness of clustering algorithms, a fuzzy c-means clustering algorithm via stratified sampling plus extension (abbr. SSEFCM) for large-scale data is developed in this paper. Concretely speaking, the basic steps of the proposed method include: (1) dividing the large-scale data into some groups called strata roughly using locality sensitive hashing technique, and randomly sampling objects separately from different strata to form the representative samples, (2) partitioning the chosen samples into different clusters with the fuzzy c-means clustering algorithm, (3) assigning the out-of-sample objects into their closest clusters via data labeling technique. Different from most of the existing sampling-based clustering algorithms, the

main innovation of the SSEFCM algorithm is it takes into account the distribution of data sets in sampling. In the process of stratified sampling, a data stratum containing a large fraction of the objects or with large variance, should be sampled more objects to represent the original data. This difference have the benefits of generating more representative sample subsets and better partial clustering results. In experimental analysis, a series of experiments have been carried on both synthetic data and real data to evaluate the effectiveness and efficiency of the proposed algorithm in terms of different indices. These experiments show that the performance of the proposed algorithm is better than that of the state-of-the-art sampling-based fuzzy c-means clustering algorithms for large-scale data set.

The rest of this paper is organized as follows. Section 2 reviews the fuzzy c-means clustering algorithm and related work on large-scale data clustering. In Section 3, the details of the proposed algorithm based on stratified sampling are presented. Experimental results are given and analyzed in Section 4. Finally, concluding remarks and future work are discussed in Section 5.

2. Fuzzy c-means algorithm and related work

In this section, the fuzzy c-means clustering algorithm [24] and related work on large-scale data clustering are reviewed.

2.1. Fuzzy c-means clustering algorithm

Among the various fast clustering algorithms for large-scale data, one of the most popular algorithm is the fuzzy c-means algorithm (FCM), which is easy to implement, simple and efficient. Suppose that $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is a set of N objects. Each object $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$ is characterized by a set of d attributes or features. For a set of data objects $\mathbf{x}_j \in R^d, j = 1, \ldots, N$, the FCM algorithm aims to find a partition represented as k fuzzy clusters, while minimizing the cost function F, which is defined as the sum of the squared distances between the data points and the corresponding centers [3]. This can be posed as follows:

$$F(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{k} \sum_{j=1}^{N} u_{i,j}^m D_{i,j}^2, \tag{1}$$

subject to

$$u_{ij} \in [0, 1], 1 \le i \le k, 1 \le j \le N, \tag{2}$$

$$\sum_{i=1}^{k} u_{i,j} = 1, 1 \le j \le N, \tag{3}$$

and

$$0 < \sum_{j=1}^{N} u_{i,j} < N, 1 \le i \le k, \tag{4}$$

where

- $\mathbf{U} = [u_{i,j}]$ is a k-by-N fuzzy matrix, and $u_{i,j} \in [0, 1]$ denotes the membership degree of the jth object to the ith cluster;
- $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$ is the cluster center matrix and $\mathbf{v}_i = [v_{i,1}, \ldots, v_{i,d}]$ is the ith cluster center with d features;
- $m \in [1, \infty)$ is the fuzzy index. When $m = 1$, the FCM algorithm will become the hard k-means clustering algorithm;
- $D_{i,j} = \|x_j - v_i\|_2$ represents the Euclidean distance between the jth object and the ith cluster center.

The above cost function F is normally optimized with an iterative procedure. The membership U and prototype matrix V are obtained by the alternative optimization method. A more detailed discussion about the convergence properties can be found in [24]. Its basic steps are described in the following:

Step 1: Set the fuzzy parameter m, the number of clusters k, and termination criterion ε. Initialize the cluster centers matrix $\mathbf{V}$ randomly. Set step variable $t = 0$;

Step 2: Update the membership matrix $\mathbf{U}$ by

$$u_{i,j}^{(t+1)} = 1 / \sum_{h=1}^{k} \left[\frac{D_{h,j}}{D_{i,j}} \right]^{(2/(1-m))}, \tag{5}$$

for $i = 1, \ldots, k$, and $j = 1, \ldots, N$;

Step 3: Update the prototype matrix $\mathbf{V}$ by

$$\mathbf{v}_i^{(t+1)} = \frac{\sum_{j=1}^{N} \left[u_{i,j}^{(t+1)} \right]^m \mathbf{x}_j}{\sum_{j=1}^{N} \left[u_{i,j}^{(t+1)} \right]^m}, \tag{6}$$

for $i = 1, \ldots, k$;

Step 4: Repeat steps 2–3 until $\|\mathbf{V}^{(t+1)} - \mathbf{V}^{(t)}\| < \varepsilon$.

Similar to most of the state-of-the-art clustering algorithms, the FCM algorithm is a linear clustering algorithm, i.e., it assumes that the clusters are linearly separable in the input space. It suffers from the issues of non-linear separability and high-dimensional feature space. In order to tackle these drawbacks, some improved fuzzy c-means algorithms, called kernel FCM algorithms, are developed in the fields of machine learning and data mining [14,25].

2.2. Clustering algorithms for large-scale data

When talking about clustering analysis for large-scale data, the scalability seems to be a "never-ending" challenge. In the following, the related work on large-scale data clustering are briefly reviewed. Most of these methods involve a preprocessing phase to compress or distribute the data, before clustering is performed. According to preprocessing approaches, these studies can be classified as follows:

- **Sampling-based methods**: These methods reduce the computation time by first choosing a subset of the given data set, using only the sampled subset to find the clusters, and then assigning the remaining data points to the closest cluster. The methods mentioned in the introduction section all belong to this category. The success of these techniques depends on the premise that the selected representative objects maintain the important structural information of the data. This paper focuses on this kind methods for large-scale data clustering problem.
- **Incremental methods**: In order to cluster the data quickly, these algorithms only scan the data points once. Bagirov et al. developed a fast modified global k-means algorithm for incremental cluster construction [26]. Wang et al. designed an incremental multiple medoids based on fuzzy clustering algorithm for large relational data [27].
- **Condensation-based methods**: These methods speed up the efficiency by encapsulating the data set into special data structures like trees or graphs. After obtaining the data structures of large data, the storage and the time of frequent operations are greatly reduced. For instance, the BIRCH algorithm clusters the large-scale data set by defining a clustering-feature tree structure [28].
- **Divide-and-conquer methods**: This methodology firstly divides the large data into different subsets that can fit the main memory and then the clustering algorithms are implemented on these subsets separately. The final clustering results are yield by merging the partial clusters of different subsets. The typical algorithms include [29,30].

- **Parallel methods**: With the development of cloud computing, parallel processing techniques for clustering have gained popularity [31]. These techniques firstly divide the clustering task into a number of independent sub-tasks which can be performed simultaneously. Then, these solutions of sub-tasks are merged into the final clustering results. For example, Ene et al. applied the Map-Reduce framework to speed up the k-means and the k-medians clustering algorithms [31]. Recently, a survey about big data clustering algorithms based on Map-Reduce can be found in [32]. Although the parallel methods have a good potential for clustering large-scale data, the implementation complexities remain a great challenge.

3. The proposed algorithm

In this section, we present a stratified sampling-based clustering algorithm, which is named Stratified Sampling plus Extension FCM (abbr. SSEFCM). Our aim is to improve the computational efficiency while maintaining the clustering quality. Initially we give a process of the stratified sampling method for large-scale data in details, and then describe the labeling scheme. Finally, the runtime complexity of the proposed algorithm will be discussed.

3.1. Stratified sampling

In stratified sampling, data objects are grouped into relatively homogeneous subsets called strata, according to one certain property, and then a representative set is selected from each stratum [22]. As a commonly used technique for large-scale data analysis, stratification and sample allocation are two key components in stratified sampling. The entire data set is divided into disjoint strata with stratification schemes. In the process of sample allocation, the sample size is determined and the sample set is drawn from each stratum.

In the following, our stratification and sample allocation methods are presented separately.

3.1.1. Stratification

In order to use stratified sampling, a large data set needs to be divided into isolated strata that present certain level of homogeneity and the objects within a stratum are similar to each other. As is well known, clustering analysis is one kind of unsupervised learning methods. The difficulty lies in how to find the appropriate stratification variable for clustering analysis tasks. Thus, a simple and efficient technique should be used to achieve this goal.

Locality-sensitive hashing (LSH) proposed in [33], is a randomized algorithm that have been intensively studied and widely used in many different fields due to its promising performance in both efficiency and accuracy [34]. Different from the conventional hashing algorithm in computer science that avoids collisions (i.e., avoids mapping two objects into the same bucket), the LSH method aims to maximize the probability of collision of similar objects in the original metric space. Its basic idea is to use a set of hash functions which store similar points in the same bucket with high probability, and dissimilar points are stored in the same bucket with low probability. Therefore, we employ the LSH method for data stratification due to its effective proximity preserving properties.

Formally, the stratified scheme using the LSH method to generate different groups can be described as follows. Given a data set $\mathbf{X}$ with N objects in a d-dimensional features space $\mathbb{R}^d$, we use a set of hash functions $H = \{h_1, \ldots, h_M\}$ to compute a M-bit binary code $y = \{y_1, \ldots, y_M\}$ for $\mathbf{x} \in \mathbf{X}$ as

$$y = \{h_1(\mathbf{x}), \ldots, h_M(\mathbf{x})\}, \tag{7}$$

where the gth bit is computed as $y_g = h_g(\mathbf{x})$. The hash function performs the mapping as $h_g : \mathbb{R}^d \longrightarrow \mathbb{B}$. Such a binary encoding process can also be viewed as mapping the original data point to a binary valued space. For a data point $\mathbf{x} \in \mathbf{X}$, the hash function $h_g \in H$ based on p-stable distributions is defined as [35]

$$h_g(\mathbf{x}) = \left\lfloor \frac{\mathbf{w_g}^T\mathbf{x} + b_g}{r_g} \right\rfloor \mod 2, \tag{8}$$

where $\mathbf{w_g} \in \mathbb{R}^d$ is a random vector with each value chosen independently from a Gaussian distribution, and $b_g \in R$ is a real number chosen uniformly from the range $[0, r_g)$ where r_g is the window size. Note that the hash function given by Eq. (8) generates the code as $h_g(\mathbf{x}) \in \{0, 1\}$. That is to say, after the hashing process, each object will obtain M-bit binary codes. Therefore, one can have at most $L = 2^M$ different labels of original data, i.e., L strata. After such transformation, two similar points in the Euclidean space, e.g., $\mathbf{x}$ and $\mathbf{y}$, have a high probability to be located into the same stratum. The strata could be used as approximate clustering results. In order to balance the tradeoff between accuracy and computational efficiency, the number of strata L is set to be the same as the number of clusters k in the experiments, i.e., $M = \lceil \log_2(k) \rceil$.

Table 1
An example of LSH scheme.

$\mathbf{X}$	h_1	h_2	h_3
$\mathbf{x}_1$	0	0	1
$\mathbf{x}_2$	0	1	0
$\mathbf{x}_3$	0	0	1
$\mathbf{x}_4$	1	0	0
$\mathbf{x}_5$	1	0	0
$\mathbf{x}_6$	1	0	0
$\mathbf{x}_7$	0	1	1
$\mathbf{x}_8$	0	1	1

Table 2
Stratification created by H for data $\mathbf{X}$.

Stratification label	Objects
001	$\{x_1, x_3\}$
010	$\{x_2\}$
011	$\{x_7, x_8\}$
100	$\{x_4, x_5, x_6\}$

In the following, taking the data in Table 1 as an example, we briefly illustrate the LSH working scheme. Let $\mathbf{X}$ be a data set containing eight objects, and $H = \{h_1, h_2, h_3\}$ be a set of hash functions. According to Eq. (8), after the hashing process, suppose that each object will obtain 3-bit binary codes shown in Table 1. Therefore, after such transformation, the data $\mathbf{X}$ will be divided into four strata shown in Table 2.

3.1.2. Sample allocation

The methods of sample allocation in stratified sampling can be classified into uniform, proportional, and optimal [36]. In the following, the optimal approach will be used to determine the size of each stratum, because it considers the stratum size and the objects variability among each stratum at the same time. After stratifying objects by locality-sensitive hashing, we assume that the data set $\mathbf{X}$ with N objects is divided into several disjoint subsets or strata, denoted by $\{S_1, \ldots, S_L\}$, where $\bigcup_{l=1}^{L} S_l = \mathbf{X}$, $S_l \bigcap S_h = \varnothing (1 \le l, h \le L, l \ne h)$.

Firstly, according to some related concepts and formulas in sampling theory, the number of objects needed to be sampled, n, can be calculated from [37]

$$n = \frac{(\sum_{l=1}^{L} N_l \sigma_l)^2}{\sigma^2 + \sum_{l=1}^{L} N_l \sigma_l^2}, \tag{9}$$

where N_l and σ_l are the number and standard deviation of objects in the stratum S_l , respectively; σ^2 is the variance of the total data set. In fact, if sample size n is larger than 5% of the total data size N,

the sample size n needs to be adjusted. In [37], the above formula is adjusted to reduce the sample size with the finite population corrections, which is defined as follows

$$n' = \frac{n}{1 + n/N}. \tag{10}$$

For the sake of simplicity, the sample size is still denoted as n in the following discussion. Then, the question is how to determine the sample size of each stratum so as to minimize the variance of sampled objects $\mathbf{X}_s$, subject to the constraint $\sum_{l=1}^{L} n_l = n$. Formally, the problem of sample allocation is solved by

$$\begin{aligned} \min \mathrm{Var}(\overline{\mathbf{X}_s}) &= \sum_{l=1}^{L} \frac{W_l^2 \sigma_l^2}{n_l}, \\ \text{s.t.} \sum_{l=1}^{L} n_l &= n, \end{aligned} \tag{11}$$

where $W_l = \frac{N_l}{N}$ is the proportion of objects in the stratum S_l to the whole data set X; σ_l denotes the standard deviation of objects in the lth stratum; n_l denotes the number of data objects drawn from the lth stratum. An application of the Lagrangian method yields the optimal sample allocation strategy

$$n_l = \frac{n W_l \sigma_l}{\sum_{l=1}^{L} W_l \sigma_l}. \tag{12}$$

This formula implies that the strata with large $W_l\sigma_l$ should be sampled heavily. A large W_l indicates that a stratum contains a large fraction of the objects, so more objects should be selected to represent the lth stratum. If σ_l is large, the values of objects in this stratum are quite spread. And in order to reflect the variability of objects in the lth stratum, a relatively large number of objects should be used.

To see the advantages of stratified sampling via the optimal allocation scheme, we compare its variance with that of sampled data under uniform random sampling. In terms of variance, it is well known that the stratified sampling with optimal allocation performs better than that with proportional allocation [36], in which the sample size of each stratum is proportional to the number of objects in this stratum. That is to say, the variance of sampled data with proportional allocation is larger than that with optimal allocation [36]. So, it suffices to just compare the variance under uniform random sampling with that under stratified sampling via proportional allocation scheme. Let μ and σ^2 be the mean and variance of the whole data, and $\mathbf{X}_r$ be the data set sampled from $\mathbf{X}$ using uniform random sampling. The variance under uniform random sampling can be formulated as

$$\begin{aligned} \mathrm{Var}(\overline{\mathbf{X}_r}) &= \frac{\sigma^2}{n} \\ &= \frac{1}{n}\frac{1}{N}\sum_{l=1}^{L}\sum_{i=1}^{N_l}(x_{i,l} - \mu)^2 \\ &= \frac{1}{n}\sum_{l=1}^{L} W_l \sigma_l^2 + \frac{1}{n} W_l (\mu_l - \mu)^2. \end{aligned} \tag{13}$$

And the variance of the sampled data under stratified sampling with proportional allocation can be formulated as

$$\begin{aligned} \mathrm{Var}(\overline{\mathbf{X}_s}) &= \sum_{l=1}^{L} W_l^2 \frac{\sigma_l^2}{n_l} \\ &= \sum_{l=1}^{L} W_l^2 \frac{\sigma_l^2}{n W_l} \\ &= \frac{1}{n}\sum_{l=1}^{L} W_l \sigma_l^2. \end{aligned} \tag{14}$$

From these equations, one observe that stratified sampling with proportional allocation always yields a smaller variance than uniform random sampling does. Thus, stratified sampling with optimal allocation attains a smaller variance than uniform random sampling does. Therefore, the sampled data set obtained by stratified sampling is more representative to the whole data set than that obtained by uniform random sampling.

3.2. Out-of-sample extension

In the sampling-based clustering algorithms, one of the most important issues is how to assign the out-of-sample data into one of the previously determined clusters. Among the different extension methods, the nearest neighbor labeling is one of the most popular techniques due to its simplicity and high efficiency. Suppose that the clustering results of the sampled data set $\mathbf{X}_s$ generated by the standardized FCM algorithm are $C_s = \{\hat{c}_1, \ldots, \hat{c}_k\}$ with cluster centers $V_s = \{\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_k\}$. Given these partial clustering results, the out-of-sample data point $\mathbf{x}_i \in \mathbf{X} - \mathbf{X}_s$ is assigned to the corresponding cluster $\hat{c}_{l_i}$ by the following criterion

$$l_i = \arg \min_{j=1,\ldots,k} D(\mathbf{x}_i, \hat{\mathbf{v}}_j), \tag{15}$$

where $D(\mathbf{x}_i, \hat{\mathbf{v}}_j) = \sqrt{\sum_{l=1}^{d}(x_{i,l} - \hat{v}_{j,l})^2}$ denotes the Euclidean distance between the data object $\mathbf{x}_i$ and the cluster center $\hat{\mathbf{v}}_j$, and d is the number of features.

3.3. Algorithm description

Based on the stratified sampling, a new clustering algorithm for large-scale data is developed, which is described in Algorithm 1.

Algorithm 1 The SSEFCM Algorithm

Input:
  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, $\mathbf{x}_j \in \mathbb{R}^d$: N d-dimensional data points;
  k: the number of clusters;
  $fPDA$: maximum fraction of data to sample;
  m: the fuzzy parameter in FCM;
  ε: the termination criterion in FCM.
Output:
  $C = \{c_1, \cdots, c_k\}$: the clustering results of the data set $\mathbf{X}$.
Method:
According to Eq. (8), partition the data into L strata using locality-sensitive hashing;
According to Eq. (9), determine the sample size n;
if $n > N \times fPDA$ **then**
  $n = \lceil N \times fPDA \rceil$
end if
According to Eq. (12), determine the number of objects n_l drawn from the lth stratum;
Sample n_l objects from the lth stratum, respectively; and denote the sampled data as $\mathbf{X}_s$;
Cluster the sampled data $\mathbf{X}_s$ with the FCM algorithm, i.e., $(\mathbf{U}_s, \mathbf{V}_s) = \mathrm{FCM}(\mathbf{X}_s, k, m, \varepsilon)$;
for $\mathbf{x} \in \mathbf{X} - \mathbf{X}_s$ **do**
  Assign each object $\mathbf{x}$ to the corresponding cluster using Eq. (15);
end for

The time complexity of the SSEFCM algorithm is basically determined by three parts: data stratifying, the sampled data clustering with the FCM algorithm and data labeling for the out-of-sample data. In the first part, the cost of applying the LSH technique on the input data set to obtain L strata is $T_1 = O(Nd \log_2 L)$ [33], where N is the number of data points, d is the size of features, and L

Table 3
Characteristics of the data sets.

Data sets	# Objects	# Attributes	# Classes
5K2D15 [14]	5,000	2	15
3M2D5 [17]	3,000,000	2	5
Electricity [38]	45,312	8	2
MNIST [14]	70,000	784	10
Person Activity [39]	164,860	8	11
Skin Segmentation [39]	245,057	3	2
Covtype [14]	581,012	54	7
KDD99 [40]	4,898,431	41	2

is the number of strata. And the random sampling in strata can be implemented in parallel. Then, the computational complexity in this step is $T_2 = \max_{l=1,\ldots,L} O(N_l)$, where N_l is the number of objects in the lth stratum. In the second part, the time complexity of partitioning the sampled data X_s with the FCM algorithm is $T_3 = O(ndk^2t)$ [24], where n is the sample size, k is the number of clusters, and t is the number of iterations of the FCM algorithm on X_s. Finally, the computational complexity of data labeling is $T_4 = O((N-n)k)$. Therefore, the total time complexity of the SSEFCM algorithm is $O(Nd\log_2 L)+\max_{l=1,\ldots,L} O(N_l)+O(ndk^2t)+O((N-n)k)$. Thus, the algorithm has linear time complexity with the size of the data set.

4. Experimental analysis

In the experiments, we compare the performance of our proposed algorithm with that of the state-of-the-art sampling-based fuzzy c-means clustering algorithms. Below we first explain the experimental setups for our evaluation and then present and analyze the experimental results.

4.1. Experimental setups

4.1.1. Data sets

A number of experiments are carried on eight data sets, including two synthetic data sets and six real data sets. The characteristics of the eight data sets are shown in Table 3.

- **5K2D15:** This is a synthetic data set, including 5000 two dimensional data objects, with 15 clusters [14]. The scatter plot of the data points is shown in Fig. 2.
- **3M2D5:** This data set was used in [17]. It is composed of 3, 000, 000 objects, which are drawn from a mixture of $k = 5$ bivariate normal distributions. The components are as given by

$$\begin{aligned}&\frac{1}{5}\text{Gaussian}\begin{pmatrix}-3\\-3\end{pmatrix}\begin{pmatrix}1&0\\0&0.2\end{pmatrix}\\&+\frac{1}{5}\text{Gaussian}\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1&0\\0&1\end{pmatrix}\\&+\frac{3}{10}\text{Gaussian}\begin{pmatrix}4\\3\end{pmatrix}\begin{pmatrix}0.1&0\\0&1\end{pmatrix}\\&+\frac{1}{10}\text{Gaussian}\begin{pmatrix}4\\3\end{pmatrix}\begin{pmatrix}0.5&0\\0&1\end{pmatrix}\\&+\frac{1}{5}\text{Gaussian}\begin{pmatrix}5\\-2\end{pmatrix}\begin{pmatrix}0.2&0\\0&1\end{pmatrix}.\end{aligned}$$

- **Electricity:** This data was collected from the Australian New South Wales Electricity Market. In this market, prices are not fixed and are affected by demand and supply of the market. The data set contains 45,312 objects. The class label identifies the change of the price relative to a moving average of the last 24 h.
- **MNIST:** This data set consists of 70,000 28 × 28 pixel handwritten digits images, which is divided into 10 classes. Each pixel is described by an integer value between 0 and 255. We normalize their values into [0, 1] by dividing 255, and each image is represented with a 784-dimensional features [14].
- **Person Activity:** This data set contains the recordings of five people performing different activities. There are 164,860 objects and 11 classes. Each object is described by 8 features.
- **Skin Segmentation:** This data set is collected by randomly sampling B,G,R values from face images of various age groups (young, middle, and old), race groups (white, black, and asian), and genders obtained from FERET database and PAL database. It consists of 245,057 samples divided into 50,859 skin samples and 194,198 is non-skin samples.
- **Covtype:** This data set consists of 54 cartographic attributes obtained from U.S. Geological Survey and U.S. Forest Service. It has 7 classes and 581,012 objects. It contains 10 quantitative features, 4 binary wilderness area designation features, and 40 binary features [14].
- **KDD99:** This data set was used for the Third International Knowledge Discovery and Data Mining Tools Competition. The KDD99 data set consists of 4,898,431 objects of 41 dimensional vectors and is labeled data that specifies the attack type (normal or attack). Note that there are many duplicated objects in this data. Before clustering the data, for each group of duplicated objects, only one object is retained, and the rest are deleted.

4.1.2. Compared clustering algorithms and parameters settings

The proposed SSEFCM algorithm is compared with the following state-of-the-art sampling-based fuzzy c-means clustering algorithms on the above data sets.

- **Single Pass FCM (SPFCM)** [41] firstly breaks the large data into equally sized small groups by uniform random sampling, sequentially processes these manageable chunks by the fuzzy c-means algorithm, and then combines the clustering results from each group.
- **Random Sampling plus Extension FCM (RSEFCM)** [14] obtains a random sample of the large data and the fuzzy c-means algorithm is applied to the sampled data. Once cluster centers are returned from the fuzzy c-means algorithm, the cluster membership μ_{ik} can be used for out-of-sample data to obtain the full data clustering results.
- **Geometric Progressive FCM (GOFCM)** [18] follows a similar pattern of the SPFCM algorithm to handle with the large-scale data. Compared to the SPFCM algorithm, the improvements consist of determining the initial subsample size using Thompson's method, forming subsequent subsamples with progressive sampling, and terminating the algorithm with stopping criterion.
- **Minimum Sample Estimate Random FCM (MSERFCM)** [18] is designed as an improvement to the RSEFCM algorithm. Specifically speaking, the MSEERFCM algorithm firstly derive the initial subsample size using the Thompson's method and find better initial clustering centers for further clustering. Then, a second subsample is progressively sampled from the original data and clustered with the clustering centers obtained from the first subsample.
- **Biased Sampling plus Extension FCM (BSEFCM)** [15] uses biased sampling technology to obtain a subsample according to the local density. The fuzzy c-means algorithm is carried on the subsample to get partial clustering results. And the clustering results of out-of-sample data are obtained via data labeling.

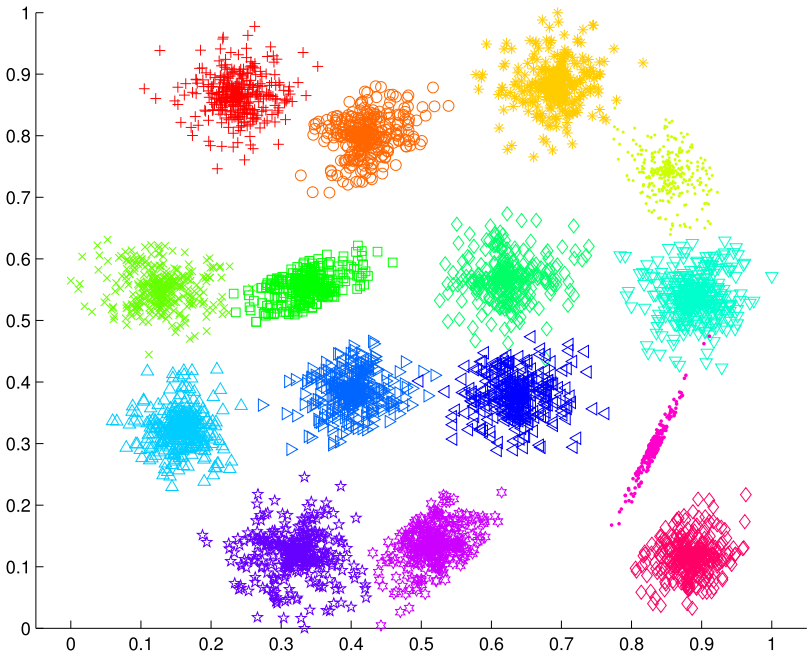

Fig. 2. Scatter plot of 5K2D15 data set.

According to the preceding brief descriptions, we find that the SPFCM and RSEFCM algorithms are based on uniform random sampling; the GOFCM and MSERFCM algorithms are based on progressive sampling; the BSEFCM algorithm is based on biased sampling. In the BSEFCM algorithm, before conducting biased sampling, 10% objects are first randomly sampled to compute the local density of the data set.

Obviously, the clustering result depends on both the sampling strategy and the baseline clustering algorithm. Herein, for a fair comparison, the traditional fuzzy c-means clustering algorithm was used to partition the sampled data. And the data labeling technology is used to classify the remaining data.

In all the experiments, only the sample size (fPDA) varies among 1%, 2.5%, 5%, 10% and 25%; the other parameters are kept fixed. For the fuzzy c-means algorithm, we set the fuzzifier $m = 2.0$ and termination criterion $\varepsilon = 0.000001$. All the parameters required by the GOFCM and MSERFCM algorithms are set to be default as in [18]. How to automatically determine the number of clusters k in a data set is a hard problem [42], which is beyond the scope of this paper. Thus, we generally set it to the number of ground truth classes in the data sets. Each experiment is composed of 20 trials, and k data objects are chosen randomly as initial cluster centers for the first subsample. The reported experimental results are the average values across these 20 trials. All the experiments were carried on a workstation with Intel Xeon CPU E5-2650@2.60 GHz and 128 GB of main memory, running Microsoft Windows 7 Professional. All codes were written in the MATLAB computing environment.

4.2. Evaluation criteria

Two kinds of metrics are used to assess the effectiveness and efficiency of the algorithms. Firstly, three popular external criteria: Clustering Accuracy (CA), Adjusted Rand Index (ARI), and Normalized Mutual Information (NMI), are used to evaluate the effectiveness of the clustering algorithms, which measure the agreement of the clustering results produced by an algorithm and the ground truth.

Suppose that $C = \{c_1, c_2, \ldots, c_k\}$ and $P = \{p_1, p_2, \ldots, p_{k'}\}$ represent the clustering results and pre-defined classes of the data set with N objects, respectively. k and k' are the number of clusters C and classes P; $N_{i,j}$ is the number of common objects of cluster c_i and pre-defined class p_j; N_i^c is the number of data points in cluster c_i ; and N_j^p is the number of data points in class p_j. Then the three popular external criteria are as follows.

• *Clustering Accuracy (CA).* CA measures the percentage of correctly classified data points in the clustering solution compared to pre-defined class labels. The CA is defined as

$$CA = \frac{\Sigma_{i=1}^{k} \max_{j=1}^{k'} N_{i,j}}{N}. \tag{16}$$

• *Adjusted Rand Index (ARI).* ARI takes into account the number of objects that exist in the same cluster and different clusters [43]. The ARI is defined as

$$ARI = \frac{\binom{N}{2}\sum_{i=1}^{k}\sum_{j=1}^{k'}\binom{N_{i,j}}{2} - [\sum_{i=1}^{k}\binom{N_i^c}{2}\sum_{j=1}^{k'}\binom{N_j^p}{2}]}{\frac{1}{2}\binom{N}{2}[\sum_{i=1}^{k}\binom{N_i^c}{2} + \sum_{j=1}^{k'}\binom{N_j^p}{2}] - [\sum_{i=1}^{k}\binom{N_i^c}{2}\sum_{j=1}^{k'}\binom{N_j^p}{2}]}. \tag{17}$$

• *Normalized Mutual Information (NMI).* This is one of the common external clustering validation metrics, which estimates the extent of the clustering structure with the external classification information of the data [44]. Thus, NMI is defined as

$$NMI = \frac{\sum_{i=1}^{k}\sum_{j=1}^{k'} N_{i,j} \log \frac{N \cdot N_{i,j}}{N_i^c \cdot N_j^p}}{\sqrt{\sum_{i=1}^{k} N_i^c \cdot \log \frac{N_i^c}{N} \cdot \sum_{j=1}^{k'} N_j^p \cdot \log \frac{N_j^p}{N}}}. \tag{18}$$

The maximum value of the three external criteria is 1. If the clustering structure is close to the true class structure, then the values of them are high. The higher the values of the three measures for a clustering result, the better the clustering performance is.

On the other hand, the efficiency of the algorithms is measured in terms of running time and speedup ratio. The running time is recorded by CPU time. And, speedup ratio is described as follows.

• *Speedup Ratio (SR).* This criterion computes the ratio of running times between the sampled-based algorithms and the traditional fuzzy c-means clustering algorithm. If T_{FCM} is the runtime of fuzzy c-means clustering algorithm and T_{SA} is the time for the

sampled-based algorithm. Then, the SR is calculated as

$$SR = \frac{T_{FCM}}{T_{SA}}. \quad (19)$$

Note that the reported values of three external criteria for each algorithm are based on full data partitions. Unless otherwise noted, the reported CPU time and speedup ratio comparisons below include the running time for sampling, the fuzzy c-means clustering algorithm on subsample, and data labeling for out-of-sample objects.

4.3. Results on effectiveness analysis

Tables 4–6 show the comparative results in terms of CA, ARI and NMI on the five data sets, respectively. For each data set, the maximum and second maximum values are highlighted in boldface. In addition, a value is marked '-' when the clustering result of the BSEFCM algorithm is not obtainable. Note that the CA values of different algorithms on Electricity, Skin Segmentation, and KDD99 are same. Thus, the maximum and second maximum values are not highlighted in boldface for these three data. Firstly, the results of clustering accuracy (CA) on the eight data sets are shown in Table 4. The results show that the SSEFCM algorithm outperforms SPFCM, RSEFCM, and MSERFCM algorithms on most of the data sets under different sampling ratios and has comparable results with the GOFCM algorithm on 5K2D15. These results indicate that the SSEFCM algorithm achieves higher or comparable clustering accuracy than the other sample-based algorithms. The results of the BSEFCM algorithm on the Person Activity data set is better than that of other algorithms. Again, we see that the SSEFCM algorithm usually has smaller standard deviations than that of SPFCM, RSEFCM, and MSERFCM algorithms. This reflects that compared with all the other algorithms, the SSEFCM algorithm has more robustness for large-scale data clustering. Furthermore, it can be seen that the accuracy values on MNIST, Person Activity and Covtype data sets are lower than the values on 5K2D15 and 3M2D15. This phenomenon also appears in some prior work on the same data set [14,27]. Surprisingly, even under a low sampling ratio, the SSEFCM algorithm generates better results than under a higher sampling ratio. For example, on the Covtype data set, the CA value is 0.5190 when the sampling ratio is equal to 2.5%, whereas the CA values are 0.5149, 0.5127, and 0.5152 for sampling ratios of 5%, 10%, and 25%, respectively. It may be due to that a small portion of representative sample may be enough to effectively reveal the inherent clustering structure, while the introduction of more objects will have a slightly negative impact on the clustering algorithm. The results of ARI and NMI in Tables 5 and 6 possess a similar pattern as that of CA. For the clustering results on Electricity, Skin Segmentation, and KDD99, the advantages of these algorithms cannot be distinguished from CA index. However, the effectiveness of the SSEFCM algorithm with the other algorithms is obvious in term of ARI and NMI indices.

4.4. Results on efficiency analysis

The following experiments is used to show the time efficiency of the proposed algorithm for large-scale data clustering. Firstly, the CPU times (20 run averages) of all the algorithms are listed in Table 7. The minimum and second minimum values for each data are shown in bold. As evident from Table 7, the SSEFCM algorithm is the fastest algorithm on most of the data sets. As previously mentioned the BSEFCM algorithm is the most time-consuming algorithm. In addition, the running time of each of these algorithms increases with the increasing of the sampling ratio for the same data set. However, this relationship is not strictly monotonic. And some sampling-based algorithms take more time than the FCM algorithm on some data sets. For example, the GOFCM algorithm takes more time on MNIST and Person Activity data set when the sampling ratio is equal to 10% or 25% than that of the FCM algorithm on the full data set.

On the other hand, the results of speedup ratio on the eight data sets for the different algorithms are shown in Fig. 3. It is easy to note that, except the SPFCM algorithm on 5K2D15 and 3M2D5 data sets, the speedup ratio of each of these algorithms decreases with the increase of sampling ratio. For the eight data sets, the SSEFCM algorithm typically has the highest speedup, and the RSEFCM algorithm is often the second and faster than all the others.

4.5. Statistical test

In order to give an overall evaluation of the different algorithms, we apply the Friedman test [45,40] to the results in Tables 4–7. Since the BSEFCM algorithm has no clustering results on some data sets, the other algorithms SPFCM, RSEFCM, GOFCM, MSERFCM, and SSEFCM will be compared in the following. Therefore, there are $A = 5$ algorithms, $B = 160$ cases (i.e., 8 data sets, 5 kinds of sampling ratio and 4 evaluation indices). Let r_i^j be the rank of the jth of the A algorithms on the ith of the B cases. For CA, ARI and NMI indices, the larger the value, the smaller the rank value. On the contrary, the smaller the value of running time, the smaller the rank value. Based on the mean performance of the five algorithms for each data set in Tables 4–7, the average aligned-ranks of each algorithm is computed with the Friedman test. The lower the average rank, the better the corresponding algorithm. In order to check whether the algorithm with the smallest rank value is significantly better than the others, the p-value of the Friedman test is computed in practice, which represents the lowest level of significance of a hypothesis. The null hypothesis for this test assumes that the results of the algorithms are equivalent and their rankings are also similar. If the returned p-value is less than the specified significance level, the null hypothesis is rejected.

Under the null hypothesis, the Friedman statistic

$$\mathcal{X}_F^2 = \frac{12B}{A(A+1)} \sum_{j=1}^{A} R_j^2 - 3B(A+1), \quad (20)$$

is distributed according to $\mathcal{X}_F^2$ with $A-1$ degrees of freedom, being $R_j = (\frac{1}{B}) \sum_{i=1}^{B} r_i^j$ the average rank of the jth algorithm for all the cases, and B the number of cases of the problem considered.

The average ranks of the five algorithms over all 100 cases are calculated to be 4.07, 2.98, 3.33, 3.14 and 1.47 for SPFCM, RSEFCM, GOFCM, MSERFCM, and SSEFCM, respectively. According to the Friedman test, a p-value is 5.1793×10^{-53}, which indicates that the null hypothesis can be rejected with high confidence. One can observe that the proposed algorithm SSEFCM is statistically better than four competitors (SPFCM, RSEFCM, GOFCM, and MSERFCM) at the 95% confidence level. That is to say, the proposed algorithm is the "winner" from this perspective.

4.6. Scalability test

To further examine the performance of the SSEFCM algorithm, scalability is evaluated by measuring the running time with respect to data size and dimension size. For this test, a synthetic data generator [46] is used to generate a group of synthetic data sets with different numbers of data points and attributes. The number of data points varies from 100,000 to 500,000, and the dimensionality is varies among 10, 20, 30, 40, and 50. There are 11 classes for each synthetic data set. Note that the running time of all the clustering algorithms is composed of the time of sampling, sampled data clustering with the fuzzy c-means algorithm and

Table 4
CA values (means ± std) of different algorithms on eight data sets.

Data sets	fPDA	SPFCM	RSEFCM	GOFCM	MSERFCM	BSEFCM	**SSEFCM**
5K2D15	1%	0.6083 ± 0.0783	0.8442 ± 0.0571	0.9391 ± 0.0356	**0.9746 ± 0.0316**	0.9377 ± 0.0368	**0.9727 ± 0.0379**
	2.5%	0.9737 ± 0.0340	0.9121 ± 0.0429	**0.9785 ± 0.0344**	**0.9770 ± 0.0309**	0.9317 ± 0.0297	0.9729 ± 0.0560
	5%	0.9712 ± 0.0298	0.9343 ± 0.0421	**0.9735 ± 0.0316**	0.9640 ± 0.0383	0.9494 ± 0.0439	**0.9731 ± 0.0428**
	10%	0.9725 ± 0.0272	0.9589 ± 0.0413	**0.9845 ± 0.0255**	0.9749 ± 0.0312	0.9328 ± 0.0421	**0.9772 ± 0.0359**
	25%	**0.9763 ± 0.0303**	0.9741 ± 0.0344	0.9744 ± 0.0314	0.9700 ± 0.0363	0.9378 ± 0.0374	**0.9878 ± 0.0310**
3M2D5	1%	0.9913 ± 0.0020	0.9454 ± 0.0153	**0.9916 ± 0.0001**	0.9894 ± 0.0157	–	**0.9916 ± 0.0001**
	2.5%	0.9894 ± 0.0157	0.9511 ± 0.0261	**0.9916 ± 0.0001**	0.9895 ± 0.0150	–	**0.9924 ± 0.0003**
	5%	0.9816 ± 0.0174	0.9520 ± 0.0257	**0.9916 ± 0.0000**	0.9916 ± 0.0011	–	**0.9935 ± 0.0002**
	10%	0.9895 ± 0.0151	0.9895 ± 0.0151	**0.9916 ± 0.0001**	0.9873 ± 0.0213	–	**0.9941 ± 0.0425**
	25%	0.9895 ± 0.0152	0.9831 ± 0.0293	0.9948 ± 0.0151	0.9895 ± 0.0151	–	**0.9971 ± 0.0437**
Electricity	1%	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	–	0.5755 ± 0.0000
	2.5%	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	–	0.5755 ± 0.0000
	5%	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	–	0.5755 ± 0.0000
	10%	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	–	0.5755 ± 0.0000
	25%	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	0.5755 ± 0.0000	–	0.5755 ± 0.0000
MNIST	1%	0.1531 ± 0.0099	**0.2634 ± 0.0268**	0.2110 ± 0.0016	0.2442 ± 0.0234	–	**0.2518 ± 0.0248**
	2.5%	0.1674 ± 0.0180	0.2543 ± 0.0209	0.2119 ± 0.0021	0.2609 ± 0.0209	–	**0.2779 ± 0.0368**
	5%	0.2052 ± 0.0026	**0.2560 ± 0.0234**	0.2121 ± 0.0032	0.2485 ± 0.0211	–	**0.2649 ± 0.0305**
	10%	0.2292 ± 0.0150	**0.2627 ± 0.0190**	0.2138 ± 0.0092	0.2573 ± 0.0224	–	**0.2642 ± 0.0315**
	25%	0.2408 ± 0.0193	**0.2576 ± 0.0169**	0.2111 ± 0.0015	0.2406 ± 0.0194	–	**0.2793 ± 0.0368**
Person activity	1%	0.3309 ± 0.0007	0.3307 ± 0.0007	0.3314 ± 0.0012	0.3308 ± 0.0008	**0.3337 ± 0.0040**	**0.3316 ± 0.0005**
	2.5%	0.3306 ± 0.0001	0.3308 ± 0.0006	0.3314 ± 0.0012	0.3306 ± 0.0004	**0.3325 ± 0.0031**	**0.3316 ± 0.0003**
	5%	0.3307 ± 0.0004	0.3306 ± 0.0002	0.3308 ± 0.0005	0.3308 ± 0.0004	**0.3331 ± 0.0028**	**0.3317 ± 0.0006**
	10%	0.3308 ± 0.0003	0.3306 ± 0.0002	0.3306 ± 0.0001	0.3307 ± 0.0005	**0.3323 ± 0.0029**	**0.3328 ± 0.0006**
	25%	0.3307 ± 0.0003	0.3306 ± 0.0002	0.3307 ± 0.0002	0.3307 ± 0.0004	**0.3331 ± 0.0034**	**0.3325 ± 0.0001**
Skin Segmentation	1%	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000
	2.5%	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000
	5%	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000
	10%	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000
	25%	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000	0.7925 ± 0.0000
Covtype	1%	0.4886 ± 0.0012	0.5101 ± 0.0072	0.5040 ± 0.0064	**0.5111 ± 0.0077**	–	**0.5112 ± 0.0070**
	2.5%	0.4928 ± 0.0034	**0.5111 ± 0.0068**	0.5053 ± 0.0061	0.5101 ± 0.0064	–	**0.5190 ± 0.0058**
	5%	0.5014 ± 0.0028	**0.5115 ± 0.0062**	0.5047 ± 0.0052	0.5097 ± 0.0067	–	**0.5149 ± 0.0062**
	10%	0.5027 ± 0.0033	0.5113 ± 0.0061	0.5028 ± 0.0046	**0.5119 ± 0.0081**	–	**0.5127 ± 0.0065**
	25%	0.5097 ± 0.0064	0.5114 ± 0.0060	0.5039 ± 0.0065	**0.5134 ± 0.0057**	–	**0.5152 ± 0.0071**
KDD99	1%	0.9845 ± 0.0000	0.9845 ± 0.0000	0.9847 ± 0.0000	0.9846 ± 0.0000	–	0.9846 ± 0.0000
	2.5%	0.9846 ± 0.0000	0.9847 ± 0.0000	0.9845 ± 0.0000	0.9846 ± 0.0000	–	0.9846 ± 0.0000
	5%	0.9846 ± 0.0000	0.9846 ± 0.0000	0.9847 ± 0.0000	0.9846 ± 0.0000	–	0.9846 ± 0.0000
	10%	0.9846 ± 0.0000	0.9846 ± 0.0000	0.9846 ± 0.0000	0.9845 ± 0.0000	–	0.9848 ± 0.0000
	25%	0.9846 ± 0.0000	0.9846 ± 0.0000	0.9846 ± 0.0000	0.9846 ± 0.0000	–	0.9848 ± 0.0000

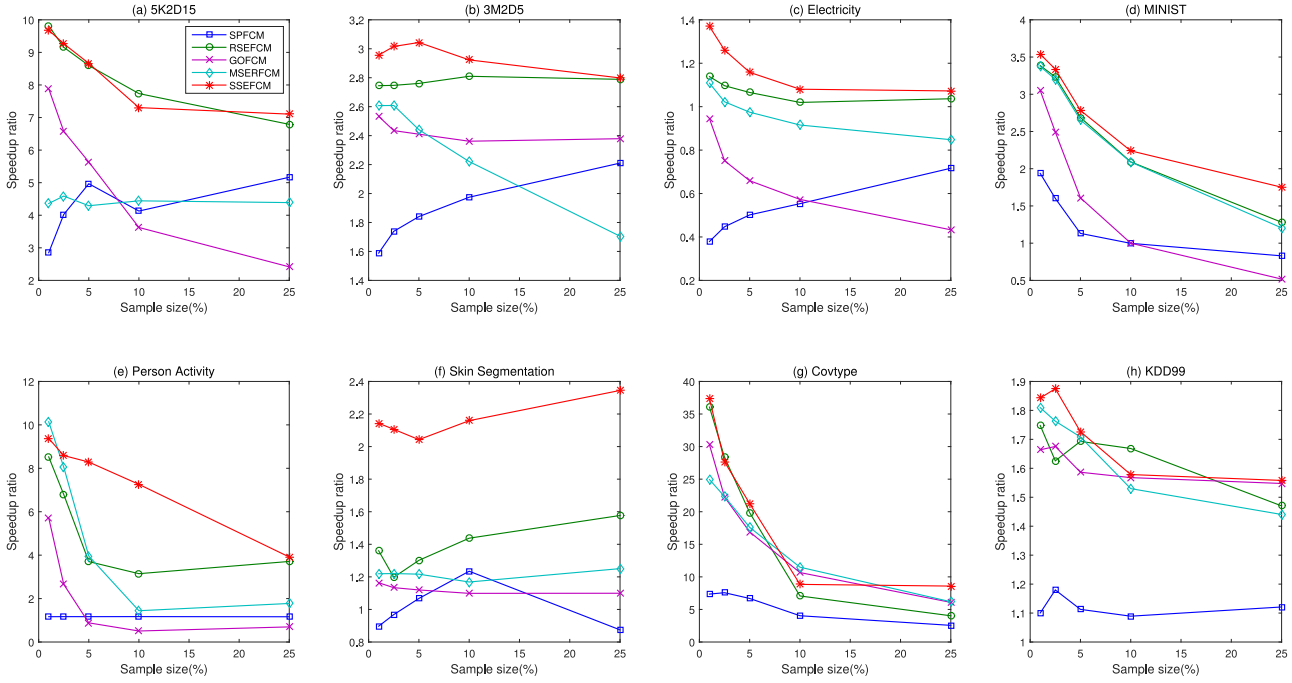

Fig. 3. Speedup ratio of the different algorithms on eight data sets.

Table 5
ARI values (means ± std) of different algorithms on eight data sets.

Data sets	fPDA	SPFCM	RSEFCM	GOFCM	MSERFCM	BSEFCM	**SSEFCM**
5K2D15	1%	0.8645 ± 0.0891	0.8014 ± 0.0662	0.9123 ± 0.0468	**0.9326 ± 0.0439**	0.9142 ± 0.0473	**0.9410 ± 0.0438**
	2.5%	0.9537 ± 0.0476	0.8846 ± 0.0519	**0.9663 ± 0.0490**	**0.9464 ± 0.0423**	0.9075 ± 0.0379	0.9527 ± 0.0658
	5%	0.9520 ± 0.0408	0.9109 ± 0.0547	**0.9611 ± 0.0429**	0.9484 ± 0.0526	0.9289 ± 0.0579	**0.9546 ± 0.0563**
	10%	0.9634 ± 0.0421	0.9426 ± 0.0531	**0.9757 ± 0.0360**	0.9634 ± 0.0428	0.9095 ± 0.0557	**0.9648 ± 0.0476**
	25%	**0.9639 ± 0.0423**	0.9622 ± 0.0465	0.9617 ± 0.0443	0.9567 ± 0.0493	0.9100 ± 0.0483	**0.9654 ± 0.0431**
3M2D5	1%	0.9799 ± 0.0037	0.9767 ± 0.0263	**0.9804 ± 0.0002**	0.9767 ± 0.0265	–	**0.9803 ± 0.0003**
	2.5%	0.9773 ± 0.0224	0.9694 ± 0.0443	**0.9804 ± 0.0001**	0.9768 ± 0.0260	–	**0.9804 ± 0.0006**
	5%	0.9754 ± 0.0000	0.9694 ± 0.0441	**0.9804 ± 0.0001**	**0.9804 ± 0.0001**	–	**0.9805 ± 0.0005**
	10%	0.9768 ± 0.0260	0.9768 ± 0.0260	**0.9804 ± 0.0001**	0.9731 ± 0.0365	–	**0.9780 ± 0.0008**
	25%	0.9763 ± 0.0260	0.9657 ± 0.0504	0.9768 ± 0.0260	**0.9768 ± 0.0260**	–	**0.9769 ± 0.0008**
Electricity	1%	0.0002 ± 0.0001	0.0024 ± 0.0014	0.0007 ± 0.0004	**0.0029 ± 0.0034**	–	**0.0025 ± 0.0015**
	2.5%	0.0002 ± 0.0002	0.0004 ± 0.0006	0.0008 ± 0.0018	**0.0029 ± 0.0025**	–	**0.0026 ± 0.0026**
	5%	0.0002 ± 0.0001	0.0003 ± 0.0012	0.0006 ± 0.0002	**0.0013 ± 0.0021**	–	**0.0028 ± 0.0021**
	10%	0.0000 ± 0.0002	**0.0011 ± 0.0010**	0.0007 ± 0.0008	0.0010 ± 0.0011	–	**0.0026 ± 0.0024**
	25%	−0.0003 ± 0.0002	**0.0007 ± 0.0012**	0.0003 ± 0.0008	0.0001 ± 0.0006	–	**0.0027 ± 0.0008**
MNIST	1%	0.0087 ± 0.0038	**0.0817 ± 0.0210**	0.0503 ± 0.0017	0.0681 ± 0.0159	–	**0.0837 ± 0.0251**
	2.5%	0.0180 ± 0.0121	0.0750 ± 0.0174	0.0501 ± 0.0011	**0.0812 ± 0.0225**	–	**0.0947 ± 0.0418**
	5%	0.0403 ± 0.0013	0.0725 ± 0.0179	0.0501 ± 0.0009	**0.0715 ± 0.0204**	–	**0.0856 ± 0.0225**
	10%	0.0568 ± 0.0089	**0.0802 ± 0.0182**	0.0506 ± 0.0040	0.0740 ± 0.0172	–	**0.0857 ± 0.0330**
	25%	0.0636 ± 0.0136	**0.0723 ± 0.0117**	0.0496 ± 0.0004	0.0654 ± 0.0144	–	**0.0931 ± 0.0378**
Person activity	1%	0.0032 ± 0.0003	0.0035 ± 0.0005	0.0030 ± 0.0009	0.0032 ± 0.0008	**0.0051 ± 0.0014**	**0.0045 ± 0.0011**
	2.5%	0.0032 ± 0.0005	0.0034 ± 0.0006	0.0035 ± 0.0009	0.0034 ± 0.0007	**0.0055 ± 0.0013**	**0.0045 ± 0.0007**
	5%	0.0034 ± 0.0005	0.0034 ± 0.0006	0.0034 ± 0.0005	0.0034 ± 0.0008	**0.0052 ± 0.0010**	**0.0045 ± 0.0010**
	10%	0.0034 ± 0.0005	0.0033 ± 0.0006	0.0034 ± 0.0007	0.0036 ± 0.0004	**0.0058 ± 0.0010**	**0.0046 ± 0.0005**
	25%	0.0035 ± 0.0006	0.0036 ± 0.0003	0.0035 ± 0.0008	0.0036 ± 0.0005	**0.0057 ± 0.0010**	**0.0047 ± 0.0007**
Skin segmentation	1%	−0.0384 ± 0.0000	−0.0383 ± 0.0015	−0.0384 ± 0.0008	**−0.0378 ± 0.0016**	−0.0446 ± 0.0019	**−0.0347 ± 0.0014**
	2.5%	−0.0384 ± 0.0000	−0.0386 ± 0.0010	**−0.0383 ± 0.0007**	−0.0388 ± 0.0008	−0.0444 ± 0.0016	**−0.0341 ± 0.0047**
	5%	−0.0384 ± 0.0000	**−0.0381 ± 0.0007**	−0.0386 ± 0.0004	−0.0384 ± 0.0005	−0.0447 ± 0.0018	**−0.0343 ± 0.0084**
	10%	−0.0384 ± 0.0000	−0.0383 ± 0.0004	−0.0386 ± 0.0004	**−0.0382 ± 0.0004**	−0.0443 ± 0.0016	**−0.0341 ± 0.0024**
	25%	−0.0383 ± 0.0000	−0.0383 ± 0.0003	**−0.0382 ± 0.0004**	−0.0384 ± 0.0002	−0.0442 ± 0.0020	**−0.0342 ± 0.0055**
Covtype	1%	0.0071 ± 0.0123	**0.0360 ± 0.0024**	0.0281 ± 0.0147	0.0357 ± 0.0024	–	**0.0359 ± 0.0024**
	2.5%	0.0091 ± 0.0167	**0.0361 ± 0.0026**	0.0322 ± 0.0114	0.0356 ± 0.0035	–	**0.0364 ± 0.0023**
	5%	0.0218 ± 0.0115	**0.0363 ± 0.0025**	0.0304 ± 0.0113	0.0355 ± 0.0026	–	**0.0367 ± 0.0022**
	10%	0.0275 ± 0.0082	0.0364 ± 0.0025	0.0287 ± 0.0104	**0.0365 ± 0.0035**	–	**0.0368 ± 0.0025**
	25%	0.0357 ± 0.0038	0.0363 ± 0.0025	0.0250 ± 0.0144	**0.0371 ± 0.0023**	–	**0.0368 ± 0.0028**
KDD99	1%	0.9341 ± 0.0002	0.9341 ± 0.0003	**0.9346 ± 0.0000**	0.9344 ± 0.0006	–	**0.9346 ± 0.0004**
	2.5%	0.9343 ± 0.0001	**0.9347 ± 0.0001**	0.9341 ± 0.0000	0.9343 ± 0.0006	–	**0.9348 ± 0.0004**
	5%	0.9343 ± 0.0000	0.9344 ± 0.0001	**0.9348 ± 0.0000**	0.9342 ± 0.0002	–	**0.9348 ± 0.0000**
	10%	0.9343 ± 0.0000	**0.9345 ± 0.0001**	**0.9345 ± 0.0000**	0.9342 ± 0.0001	–	**0.9848 ± 0.0000**
	25%	0.9343 ± 0.0000	0.9343 ± 0.0000	**0.9345 ± 0.0000**	0.9345 ± 0.0000	–	**0.9348 ± 0.0000**

labeling the remaining data points. The sample size is set to 5%. And each value in Fig. 4 is the average time of 20 runs of each algorithm.

The scalability with respect to data size of the different clustering algorithms is shown in Fig. 4(a). In this study, we set the dimensionality to 10, and the cluster number to 11, and also vary the number of objects from 100,000 to 500,000. From Fig. 4(a), it can be seen that all sampling-based clustering algorithms are linear with respect to the number of objects. The increasing rate of the running time of the proposed algorithm SSEFCM is very close to the RSEFCM and MSERFCM algorithms, and much slower than the SPFCM and GOFCM algorithms. Therefore, the SSEFCM algorithm ensures efficient execution when the data size is large.

The scalability with data dimensionality of the different algorithms is shown in Fig. 4(b). We fix the data size to 100,000, and the cluster number to 11, and also vary the number of dimensions from 10 to 100. Similar to Fig. 4(a), one can see that the runtime of all the sampling-based clustering algorithms increases linearly with the increasing of the data dimensionality. Compared with the RSEFCM and MSERFCM algorithms, the proposed algorithm SSEFCM takes more time. This is because the locality-sensitive hashing technique is needed for data stratification before performing stratified sampling. However, the SSEFCM algorithm performs significantly better than the SPFCM and GOFCM algorithms. Thus, the SSEFCM algorithm is also scalable to large-scale data sets.

In summary, the experimental results show that the proposed algorithm not only obtains higher or comparable quality of clustering results, but also has high computational efficiency. This means that the SSEFCM algorithm is more suitable for handling large-scale data sets. The reasons about the effectiveness and efficiency of the proposed algorithm are analyzed in the following. The time complexity of the FCM is $O(Ndk^2t)$, where N is the data size, d is the number of attributes, k is the number of clusters, and t is the number of iterations. Thus, given a data set with k clusters, the FCM algorithm can be accelerated by reducing the sample size N, attribute size d or iterations t. Except the SPFCM algorithm, the other sampling-based FCM algorithms can reduce the runtime complexity via running the FCM on subsample. One of the key differences between the proposed algorithm and the other algorithms is the SSEFCM algorithm takes into account the distribution of data sets in sampling. A data stratum containing a large fraction of the objects or with large variance, should be sampled more objects to represent the original data. This difference have the benefits of generating more representative sample subsets. The better partial clustering results on sampled data will result in better final clustering results. And the better subsample will generating better initial cluster centers. Better cluster center estimates reduce the number of iterations to reach the termination criterion.

5. Conclusion and future work

To overcome the limitations of the existing clustering algorithms with sampling scheme, a new large-scale data fuzzy c-means clustering algorithm based on stratified sampling, named Stratified Sampling plus Extension FCM (abbr. SSEFCM), has been

Table 6
NMI values (means ± std) of different algorithms on eight data sets.

Data sets	fPDA	SPFCM	RSEFCM	GOFCM	MSERFCM	BSEFCM	**SSEFCM**
5K2D15	1%	0.6543 ± 0.0613	0.9106 ± 0.0295	0.9549 ± 0.0202	**0.9612 ± 0.0172**	0.9622 ± 0.0183	**0.9559 ± 0.0212**
	2.5%	**0.9669 ± 0.0246**	0.9492 ± 0.0215	**0.9803 ± 0.0204**	0.9626 ± 0.0166	0.9603 ± 0.0153	0.9639 ± 0.0267
	5%	0.9691 ± 0.0194	0.9601 ± 0.0229	**0.9794 ± 0.0167**	0.9750 ± 0.0214	0.9668 ± 0.0232	**0.9771 ± 0.0237**
	10%	0.9772 ± 0.0201	0.9725 ± 0.0204	**0.9855 ± 0.0140**	0.9813 ± 0.0170	0.9601 ± 0.0229	**0.9804 ± 0.0179**
	25%	0.9799 ± 0.0179	0.9803 ± 0.0185	**0.9805 ± 0.0171**	0.9787 ± 0.0197	0.9592 ± 0.0191	**0.9819 ± 0.0159**
3M2D5	1%	0.9686 ± 0.0050	0.9665 ± 0.0199	**0.9693 ± 0.0002**	0.9665 ± 0.0201	–	**0.9691 ± 0.0004**
	2.5%	0.9665 ± 0.0201	0.9609 ± 0.0337	**0.9694 ± 0.0001**	0.9665 ± 0.0197	–	**0.9688 ± 0.0008**
	5%	0.9634 ± 0.0000	0.9610 ± 0.0335	**0.9693 ± 0.0001**	0.9633 ± 0.0001	–	**0.9686 ± 0.0006**
	10%	**0.9666 ± 0.0197**	**0.9666 ± 0.0197**	**0.9693 ± 0.0001**	0.9638 ± 0.0277	–	0.9603 ± 0.0006
	25%	0.9666 ± 0.0197	0.9582 ± 0.0382	0.9665 ± 0.0197	**0.9668 ± 0.0197**	–	**0.9673 ± 0.0006**
Electricity	1%	0.0001 ± 0.0002	0.0008 ± 0.0031	0.0007 ± 0.0028	**0.0012 ± 0.0025**	–	**0.0016 ± 0.0011**
	2.5%	0.0003 ± 0.0002	0.0008 ± 0.0008	0.0007 ± 0.0013	**0.0014 ± 0.0036**	–	**0.0015 ± 0.0022**
	5%	0.0005 ± 0.0002	0.0007 ± 0.0006	**0.0019 ± 0.0013**	0.0013 ± 0.0016	–	**0.0017 ± 0.0013**
	10%	0.0005 ± 0.0002	**0.0012 ± 0.0012**	0.0010 ± 0.0008	0.0011 ± 0.0014	–	**0.0015 ± 0.0022**
	25%	0.0007 ± 0.0003	**0.0016 ± 0.0015**	0.0014 ± 0.0011	0.0009 ± 0.0010	–	**0.0016 ± 0.0010**
MNIST	1%	0.0173 ± 0.0053	**0.1821 ± 0.0314**	0.1218 ± 0.0038	0.1593 ± 0.0286	–	**0.1861 ± 0.0370**
	2.5%	0.0359 ± 0.0155	0.1713 ± 0.0279	0.1216 ± 0.0021	**0.1819 ± 0.0345**	–	**0.1952 ± 0.0506**
	5%	0.0745 ± 0.0024	**0.1678 ± 0.0294**	0.1215 ± 0.0024	0.1640 ± 0.0313	–	**0.1856 ± 0.0304**
	10%	0.1322 ± 0.0153	**0.1816 ± 0.0273**	0.1228 ± 0.0099	0.1711 ± 0.0297	–	**0.1859 ± 0.0414**
	25%	0.1504 ± 0.0233	**0.1693 ± 0.0216**	0.1204 ± 0.0013	0.1553 ± 0.0268	–	**0.1969 ± 0.0462**
Person activity	1%	0.0083 ± 0.0004	0.0091 ± 0.0048	0.0091 ± 0.0037	0.0094 ± 0.0032	**0.0158 ± 0.0049**	**0.0105 ± 0.0036**
	2.5%	0.0083 ± 0.0003	0.0095 ± 0.0026	0.0095 ± 0.0022	0.0084 ± 0.0018	**0.0167 ± 0.0036**	**0.0118 ± 0.0027**
	5%	0.0084 ± 0.0003	0.0090 ± 0.0012	0.0092 ± 0.0018	0.0095 ± 0.0015	**0.0135 ± 0.0031**	**0.0105 ± 0.0021**
	10%	0.0083 ± 0.0005	0.0081 ± 0.0014	0.0082 ± 0.0010	0.0094 ± 0.0013	**0.0183 ± 0.0046**	**0.0102 ± 0.0021**
	25%	0.0083 ± 0.0008	0.0086 ± 0.0008	0.0082 ± 0.0008	0.0083 ± 0.0008	**0.0151 ± 0.0031**	**0.0102 ± 0.0012**
Skin segmentation	1%	0.0223 ± 0.0001	0.0223 ± 0.0014	0.0222 ± 0.0007	0.0219 ± 0.0012	**0.0261 ± 0.0020**	**0.0257 ± 0.0012**
	2.5%	0.0223 ± 0.0001	0.0226 ± 0.0011	0.0223 ± 0.0007	0.0226 ± 0.0007	**0.0260 ± 0.0016**	**0.0258 ± 0.0012**
	5%	0.0222 ± 0.0001	0.0222 ± 0.0006	0.0225 ± 0.0004	0.0222 ± 0.0005	**0.0264 ± 0.0018**	**0.0259 ± 0.0014**
	10%	0.0222 ± 0.0000	0.0222 ± 0.0004	0.0224 ± 0.0003	0.0221 ± 0.0003	**0.0259 ± 0.0015**	**0.0256 ± 0.0017**
	25%	0.0222 ± 0.0000	0.0222 ± 0.0003	0.0221 ± 0.0004	0.0222 ± 0.0002	**0.0258 ± 0.0021**	**0.0261 ± 0.0015**
Covtype	1%	0.0475 ± 0.0102	0.1052 ± 0.0048	0.0910 ± 0.0154	**0.1058 ± 0.0053**	–	**0.1056 ± 0.0040**
	2.5%	0.0576 ± 0.0151	**0.1044 ± 0.0036**	0.0934 ± 0.0135	0.1034 ± 0.0032	–	**0.1069 ± 0.0044**
	5%	0.0777 ± 0.0193	**0.1042 ± 0.0028**	0.0914 ± 0.0141	0.1035 ± 0.0029	–	**0.1062 ± 0.0035**
	10%	0.0818 ± 0.0157	**0.1040 ± 0.0028**	0.0911 ± 0.0138	0.1036 ± 0.0028	–	**0.1049 ± 0.0031**
	25%	0.0974 ± 0.0106	0.1040 ± 0.0025	0.0842 ± 0.0173	**0.1050 ± 0.0024**	–	**0.1049 ± 0.0035**
KDD99	1%	0.8683 ± 0.0003	0.8687 ± 0.0004	0.8683 ± 0.0000	**0.8684 ± 0.0017**	–	**0.8689 ± 0.0006**
	2.5%	0.8688 ± 0.0002	0.8695 ± 0.0001	0.8681 ± 0.0007	**0.8690 ± 0.0007**	–	**0.8689 ± 0.0000**
	5%	0.8689 ± 0.0001	0.8690 ± 0.0001	**0.8692 ± 0.0000**	0.8688 ± 0.0003	–	**0.8692 ± 0.0001**
	10%	0.8689 ± 0.0001	0.8692 ± 0.0002	**0.8692 ± 0.0002**	0.8688 ± 0.0002	–	**0.8692 ± 0.0003**
	25%	0.8689 ± 0.0000	0.8690 ± 0.0000	0.8687 ± 0.0010	**0.8691 ± 0.0002**	–	**0.8693 ± 0.0001**

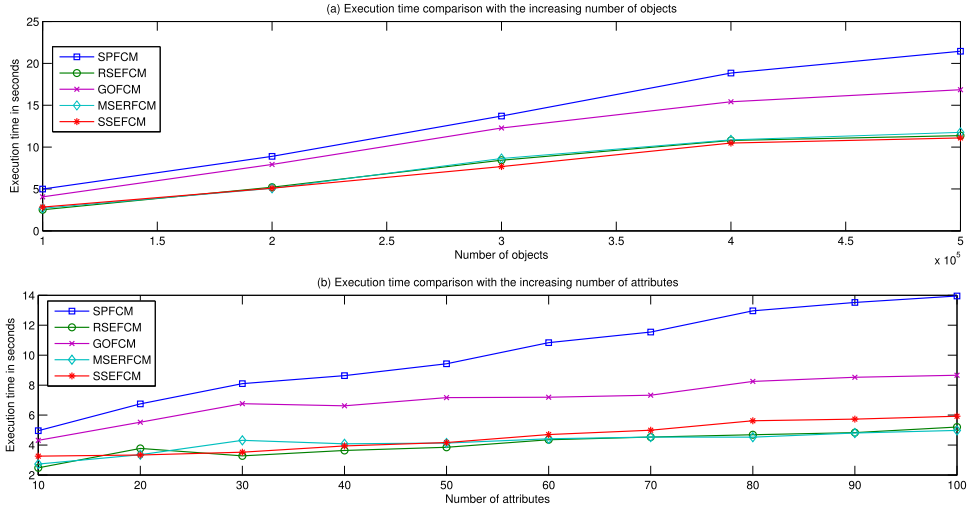

Fig. 4. Running time comparison of different clustering algorithms with the increasing number of objects and attributes.

proposed in this study. In the developments, locality sensitive hashing technique has been used to divide the original data into some strata. The partial clustering results of representative objects chosen by stratified sampling have been obtained with the FCM algorithm. The out-of-sample objects are assigned to their closest clusters via data labeling technique. To demonstrate the performance of the SSEFCM algorithm, five representative fuzzy c-means clustering algorithms for large-scale data with sampling scheme have been employed as references or baselines. Experimental results on the synthetic and real-world data sets show that

Table 7
Average running times (seconds) of different algorithms on eight data sets.

Data sets	FCM	fPDA	SPFCM	RSEFCM	GOFCM	MSERFCM	BSEFCM	**SSEFCM**
5K2D15	1.6631	1%	0.5838	**0.1695**	0.2109	0.3814	14.4499	**0.1718**
		2.5%	0.4134	**0.1816**	0.2529	0.3625	14.6386	**0.1795**
		5%	0.3348	**0.1935**	0.2952	0.3875	15.1941	**0.1920**
		10%	0.4028	**0.2150**	0.4588	0.3744	15.2461	**0.2277**
		25%	0.3219	**0.2449**	0.6885	0.3790	15.2202	**0.2341**
3M2D5	160.8236	1%	101.2333	**58.5885**	63.4912	61.6825	–	**54.4018**
		2.5%	92.4848	**58.5379**	66.0536	61.6553	–	**53.3239**
		5%	87.3707	**58.2549**	66.7317	65.8855	–	**52.8421**
		10%	81.4449	**57.2347**	68.1051	72.4159	–	**55.0167**
		25%	72.7613	**57.6662**	67.6074	94.3177	–	**57.4687**
Electricity	1.5744	1%	4.1538	**1.3828**	1.6665	1.4205	–	**1.1496**
		2.5%	3.5229	**1.4359**	2.0959	1.5397	–	**1.2489**
		5%	3.1389	**1.4778**	2.3892	1.6154	–	**1.3595**
		10%	2.8455	**1.5441**	2.7475	1.7198	–	**1.4573**
		25%	2.1972	**1.5189**	3.6359	1.8566	–	**1.4679**
MNIST	58.8222	1%	30.3626	**17.3804**	19.2809	17.4374	–	**16.6234**
		2.5%	36.7322	**18.1886**	23.5950	18.3776	–	**17.6282**
		5%	51.9991	**21.8893**	36.5623	22.1608	–	**21.1283**
		10%	58.9648	**28.1189**	58.8393	28.1534	–	**26.2598**
		25%	70.8996	**45.9129**	113.5843	48.8118	–	**33.5821**
Person activity	38.5009	1%	33.0408	4.5147	6.7297	**3.7956**	853.1262	**4.1060**
		2.5%	32.9488	5.6675	14.5002	**4.7799**	847.9988	**4.4834**
		5%	32.9770	10.3987	43.6366	**9.7218**	850.6041	**4.6456**
		10%	32.9051	**12.2334**	75.9068	26.7228	844.2221	**5.3023**
		25%	33.1079	**10.3770**	55.2864	21.6855	874.4470	**9.8359**
Skin segmentation	10.4617	1%	11.6602	**7.6906**	8.9887	8.5815	2698.0212	**4.8813**
		2.5%	10.8046	8.7356	9.2260	**8.5726**	2710.6388	**4.9670**
		5%	9.8005	**8.0374**	9.3434	8.5989	2692.9827	**5.1214**
		10%	8.4737	**7.2770**	9.5206	8.9529	2829.4008	**4.8455**
		25%	11.9611	**6.6344**	9.5160	8.3653	2202.2446	**4.4617**
Covtype	618.57275	1%	84.0651	**17.1506**	20.4236	24.7981	–	**16.5517**
		2.5%	81.7189	**21.8120**	27.7859	27.6978	–	**22.3622**
		5%	92.5196	**31.1592**	36.6117	35.1420	–	**29.1113**
		10%	153.3395	87.5048	**58.0041**	**53.8501**	–	69.9118
		25%	242.0357	153.5883	101.6747	**100.0634**	–	**72.0472**
KDD99	56.7401	1%	51.5675	32.4625	34.1005	**31.3545**	–	**30.7895**
		2.5%	48.0950	34.9000	33.8670	**32.1880**	–	**30.2640**
		5%	50.9920	33.5175	35.7670	**33.2165**	–	**32.8640**
		10%	52.0940	**34.0085**	**36.1898**	37.0895	–	35.9485
		25%	50.6165	38.5935	**36.6630**	39.3895	–	**36.4170**

the proposed algorithm have much better clustering performance than the other sampling-based algorithms in term of efficiency and effectiveness.

Sampling-based methods significantly reduce the time for clustering large-scale data sets, but it may lower the clustering quality and performance for some cases. Therefore, we will study the problem of large-scale data clustering and develop scalable clustering solutions from the perspective of incremental learning in the future.

Acknowledgments

The authors are very grateful to the anonymous reviewers and editor. Their many helpful and constructive comments and suggestions helped us significantly improve this work. This work was supported by National Natural Science Fund of China (Nos. 61603230, 61432011, 61876103, U1435212, 61573229), the Natural Science Foundation of Shanxi Province, China (No. 201601D202039), and CityU 11301014 of Hong Kong SAR Government, and the 1331 Engineering Project of Shanxi Province, China.

References

[1] J.W. Han, M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann, San Francisco, 2001.
[2] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: A review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[3] R. Xu, II. D. Wunsch, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005) 645–678.
[4] A.K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[5] X. Wu, X. Zhu, G.Q. Wu, W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (1) (2014) 97–107.
[6] L. Bai, J.Y. Liang, C.Y. Dang, F.Y. Cao, The impact of cluster representatives on the convergence of the *K*-modes type clustering, IEEE Trans. Pattern Anal. Mach. Intell. 35 (6) (2013) 1509–1522.
[7] Y. Wang, L. Chen, K-MEAP: Multiple exemplars affinity propagation with specified k clusters, IEEE Trans. Neural Netw. Learn. Syst. 27 (12) (2016) 2670–2682.
[8] Y.H. Qian, F.J. Li, J.Y. Liang, B. Liu, C.Y. Dang, Space structure and clustering of categorical data, IEEE Trans. Neural Netw. Learn. Syst. 27 (10) (2016) 2047–2059.
[9] Y. Yang, J.M. Jiang, Hybrid sampling-based clustering ensemble with global and local constitutions, IEEE Trans. Neural Netw. Learn. Syst. 27 (5) (2016) 952–965.
[10] Y. Zhang, S. Chen, G. Yu, Efficient distributed density peaks for clustering large data sets in mapreduce, IEEE Trans. Knowl. Data Eng. 28 (12) (2016) 3218–3230.
[11] S.K. Thompson, Sampling, John Wiley & Sons, New York, 2012.
[12] S. Guha, R. Rastogi, K. Shim, CURE: An efficient clustering algorithm for large databases, in: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, (1998), pp. 73-84.
[13] R.T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, in: Proceedings of the 20th International Conference on Very Large Data Bases, (1994), pp. 144-155.
[14] T.C. Havens, J.C. Bezdek, C. Leckie, L.O. Hall, M. Palaniswami, Fuzzy *c*-means algorithms for very large data, IEEE Trans. Fuzzy Syst. 20 (6) (2012) 1130–1146.

[15] G. Kollios, D. Gunopulos, N. Koudas, S. Berchtold, Efficient biased sampling for approximate clustering and outlier detection in large datasets, IEEE Trans. Knowl. Data Eng. 15 (5) (2003) 1170–1187.
[16] P. Domingos, G. Hulten, P. Edu, C. Edu, A general method for scaling up machine learning algorithms and its application to clustering, in: Proceedings of the Eighteenth International Conference on Machine Learning, (2001), pp. 106-113.
[17] L. Wang, J.C. Bezdek, C. Leckie, R. Kotagiri, Selective sampling for approximate clustering of very large data sets, Int. J. Intell. Syst. 23 (3) (2008) 313–331.
[18] J.K. Parker, L.O. Hall, Accelerating fuzzy *c*-means using an estimated subsample size, IEEE Trans. Fuzzy Syst. 22 (5) (2014) 1229–1244.
[19] A. Nanopoulos, Y. Theodoridis, Y. Manolopoulos, Indexed-based density biased sampling for clustering applications, Data Knowl. Eng. 57 (1) (2006) 37–63.
[20] C. Palmer, C. Faloutsos, Density biased sampling: An improved method for data mining and clustering, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, (2000), pp. 82-92.
[21] Y. Ye, Q. Wu, J.Z. Huang, M. Ng, X. Li, Stratified sampling for feature subspace selection in random forests for high dimensional data, Pattern Recognit. 46 (2013) 769–787.
[22] L. Jing, K. Tian, J.Z. Huang, Stratified feature sampling method for ensemble clustering of high dimensional data, Pattern Recognit. 48 (11) (2015) 3688–3702.
[23] D. Freedman, R. Pisani, R. Purves (Eds.), Statistics, fourth ed., W. W. Norton Company, NewYork, 2007.
[24] J. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, NY, 1981.
[25] X. Yang, G. Zhang, J. Lu, J. Ma, A kernel fuzzy c-means clustering based fuzzy support vector machine algorithm for classification problems with outliers or noises, IEEE Trans. Fuzzy Syst. 19 (1) (2011) 105–115.
[26] A.M. Bagirov, J. Ugon, D. Webb, Fast modified global k-means algorithm for incremental cluster construction, Pattern Recognit. 44 (4) (2011) 866–876.
[27] Y. Wang, L. Chen, J.P. Mei, Incremental fuzzy clustering with multiple medoids for large data, IEEE Trans. Fuzzy Syst. 22 (6) (2014) 1557–1568.
[28] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: An efficient data clustering method for very large databases, ACM SIGMOD Record 25 (2) (1996) 103–114.
[29] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. 100 (1) (1971) 68–86.
[30] D. Cheng, R. Kannan, S. Vempala, G. Wang, A divide-and-merge methodology for clustering, ACM Trans. Database Syst. 31 (4) (2006) 1499–1525.
[31] A. Ene, S. Im, B. Moseley, Fast clustering using MapReduce, in: Proceedings of the International Conference on Knowledge Discovery and Data Mining, (2011), pp. 681-689.
[32] A. Mohebi, S. Aghabozorgi, T.Y. Wah, T. Herawan, R. Yahyapour, Iterative big data clustering algorithms: A review, Softw. - Pract. Exp. 46 (1) (2016) 107–129.
[33] P. Indyk, R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, in: Proceedings of the 13th Annual ACM Symposium on Theory of Computing, (1998), pp. 604-613.
[34] J. Wang, W. Liu, S. Kumar, S.F. Chang, Learning to hash for indexing Big Data: A survey, Proc. IEEE 104 (1) (2016) 34–57.
[35] M. Datar, N. Immorlica, R. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the Symposium on Computational Geometry, (2004), pp. 253-262.
[36] J.A. Rice, Mathematical Statistics and Data Analysis, third ed., Wadsworth Publishing Co Inc, USA, 2007.
[37] W.G. Cochran, Sampling Techniques, John Wiley & Sons, New York, 1977.
[38] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 27–39.
[39] A. Frank, A. Asuncion, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2010.
[40] D. Kumar, J.C. Bezdek, M. Palaniswami, S. Rajasegarar, C. Leckie, T.C. Havens, A hybrid approach to clustering in big data, IEEE Trans. Cybern. 46 (10) (2016) 2372–2385.
[41] P. Hore, L.O. Hall, D.B. Goldgof, Single pass fuzzy *c*-means, in: Proceeding of the IEEE International Conference on Fuzzy Systems, (2007), pp. 1-7.
[42] J.Y. Liang, X.W. Zhao, D.Y. Li, F.Y. Cao, C.Y. Dang, Determining the number of clusters using information entropy for mixed data, Pattern Recognit. 45 (6) (2012) 2251–2265.
[43] L. Hubert, P. Arabie, Comparing partitions, J. Classification 2 (1) (1985) 193–218.
[44] X.W. Zhao, J.Y. Liang, C.Y. Dang, Clustering ensemble selection for categorical data based on internal validity indices, Pattern Recognit. (69) (2017) 150–168.
[45] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Amer. Statist. Assoc. 32 (200) (1937) 675–701.
[46] Y. Lai, R. Orlandic, W.G. Yee, S. Kulkarni, Scalable clustering for large high-dimensional data based on data summarization, in: Proceedings of IEEE Symposium on 2007 Computational Intelligence and Data Mining, (2007), pp. 456-461.