Tsinghua Science and Technology

Volume 23 | Issue 2 Article 8

2018

Greedy Optimization for K-Means-Based Consensus Clustering

Xue Li
the School of Economics and Management, Tsinghua University, Beijing 100084, China.

Hongfu Liu
the Department of Electrical and Computer Engineering, Northeastern University, Boston MI 02115, USA.

Recommended Citation

Xue Li, Hongfu Liu. Greedy Optimization for K-Means-Based Consensus Clustering. Tsinghua Science and Technology 2018, 23(2): 184-194.

TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 04/09 pp184–194
DOI: 10.26599 / TST.2018.9010063
Volume 23, Number 2, April 2018

Greedy Optimization for K-Means-Based Consensus Clustering

Xue Li* and Hongfu Liu

Abstract: Consensus clustering aims to fuse several existing basic partitions into an integrated one; this has been widely recognized as a promising tool for multi-source and heterogeneous data clustering. Owing to robust and high-quality performance over traditional clustering methods, consensus clustering attracts much attention, and much efforts have been devoted to develop this field. In the literature, the K-means-based Consensus Clustering (KCC) transforms the consensus clustering problem into a classical K-means clustering with theoretical supports and shows the advantages over the state-of-the-art methods. Although KCC inherits the merits from K-means, it suffers from the initialization sensitivity. Moreover, the current consensus clustering framework separates the basic partition generation and fusion into two disconnected parts. To solve the above two challenges, a novel clustering algorithm, named Greedy optimization of K-means-based Consensus Clustering (GKCC) is proposed. Inspired by the well-known greedy K-means that aims to solve the sensitivity of K-means initialization, GKCC seamlessly combines greedy K-means and KCC together, achieves the merits inherited by GKCC and overcomes the drawbacks of the precursors. Moreover, a 59-sampling strategy is conducted to provide high-quality basic partitions and accelerate the algorithmic speed. Extensive experiments on 36 benchmark datasets demonstrate the significant advantages of GKCC over KCC and KCC++ in terms of the objective function values and standard deviations and external cluster validity.

Key words: K-means; consensus clustering; initialization; greedy optimization

1 Introduction

Cluster analysis aims to separate a set of data points into several groups so that the points in the same group are more similar than those in different groups[1], which is a crucial and fundamental technique in machine learning and data mining. It has been widely used in information retrieval, recommendation systems, biological analysis, health care, supply chain management, marketing, business, etc. Much efforts have been devoted to this research field, and many clustering algorithms have been proposed based on different assumptions. For example, K-means is the archetypal clustering method that aims to find K centers to represent the entire data[2]. Agglomerative hie-rarchy clustering merges the nearest two points or clusters at each time until all the points are in the same cluster[3]; Density-Based Spatial Clustering of Applications with Noise (DBSCAN) separates the points by high-density regions[4]. Since cluster analysis is an unsupervised task and different algorithms provide different clustering results, it is difficult to select the best algorithm for a given application. Moreover, certain algorithms have many parameters to tune, and their performance is prone to large volatility.

Consensus clustering, also known as ensemble clustering, has been regarded as a robust meta-clustering

• Xue Li is with the School of Economics and Management, Tsinghua University, Beijing 100084, China. E-mail: lix2.11@sem.tsinghua.edu.cn.
• Hongfu Liu is with the Department of Electrical and Computer Engineering, Northeastern University, Boston MI 02115, USA. E-mail: liu.hongf@husky.neu.edu.
* To whom correspondence should be addressed.
Manuscript received: 2017-02-06; revised: 2017-04-25; accepted: 2017-05-02

algorithm[5]. This algorithm fuses several diverse basic partitions generated by traditional clustering algorithms into an integrated partition. It has been widely recognized that consensus clustering is effective to generate robust clustering results, detect bizarre clusters, handle noise, outliers, and sample variations, and integrate solutions from multiple distributed sources of data or attributes[6]. Unlike the traditional clustering methods which use the original data, the input of consensus clustering is a set of basic partitions. Given a dataset, basic partition generation strategies are used to produce various diverse basic partitions. For example, Random Parameter Selection (RPS) applies clustering methods with varied parameters, while Random Feature Selection (RFS) conducts clustering on the partial data with traditional clustering method. Consensus clustering is a fusion problem in essence, rather than a traditional clustering problem. It can be roughly divided into two categories: The first category designs a utility function that measures the similarity between basic partitions and the final partition, and solves a combinatorial optimization problem by maximizing the utility function[6, 7]. The second category employs a co-association matrix to calculate the number of times a pair of instances co-occurring in the same cluster, and then runs a graph partition method for the final consensus result[8].

Although consensus clustering has several merits compared to traditional clustering methods, it has several challenges. First, clustering is an unsupervised task, i.e., no label information can be used to guide the fusion process. Second, the nonorder property makes it difficult to align the clusters in different partitions. Third, the basic partitions might have different cluster numbers. Liu et al. addressed the above challenges in a unified framework in their K-means-based Consensus Clustering (KCC) algorithm that transforms consensus clustering to a (weighted) K-means clustering problem[9, 10]. Although such transformations provide large benefits in terms of efficiency and theoretical support, the performance of KCC can still be unstable because K-means is sensitive to initialization. Besides, the algorithm fails to generate the basic partition set.

In this paper, we propose a novel clustering algorithm by considering the KCC problem, namely Greedy optimization of K-means-based Consensus Clustering (GKCC) that is based on greedy center allocation in an augmented partition feature space. We aim to solve the sensitivity of K-means initialization and basic partition generation in a unified framework. Inspired by greedy K-means, a highly efficient variant of K-means[11] that initializes the K centers with the previous $K-1$ centers and greedily searches the rest one, greedy K-means is employed for K-means initialization and basic partition generation. However, greedy K-means generates n partitions with one certain cluster number, and only one partition is selected for the next-step optimization, a type of waste. Moreover, the time complexity becomes expensive when n is very large.

Therefore, the intermediate partitions generated by greedy K-means are further used as the basic partitions for later consensus fusion. To overcome the high time complexity, a 59-sampling strategy is used to accelerate the speed to avoid the brute-force global search. Interestingly, these intermediate partitions not only can be used for the consensus fusion, but also provide rich information for the next-step greedy K-means. Figure 1 shows the framework of GKCC. The entire process is similar to greedy K-means. In each phase, the centroids in the previous phase are used to greedily search one extra center, and then K-means is conducted to adjust the current centroids. In contrast, in the proposed GKCC, the intermediate partitions are also used to enrich the feature space by concatenating the original features and these partitions. Therefore, the original data and basic partitions are combined as the new data for generating subsequent basic partitions. Thus, we provide a new basic partition generation strategy that strongly couples the later fusion and builds an end-to-end process for ensemble clustering.

The benefits of GKCC are three-fold. First, GKCC seamlessly combines greedy K-means and KCC for a robust and high-quality clustering. Second, GKCC involves the original data and basic partitions to generate subsequent basic partitions and makes the consensus cluster a one-step process. Third, GKCC overcomes the sensitivity issue of K-means initialization and provides a robust and high-quality performance. Extensive experiments on 36 benchmark datasets show significant performance improvements of GKCC over rival algorithms in terms of stability and quality. In summary, the major contributions of this study are as follows:

- GKCC makes full use of intermediate partitions generated by greedy K-means to generate subsequent basic partitions and later fusion.
- Using greedy dynamic search, GKCC incrementally adds new centers and overcomes the sensitivity of K-means to initialization. Besides, a random-sampling strategy is used for the global search of new good centers, and the randomly generated partitions are used as the basic partitions in the final fusion stage.
- Extensive experimental results on 36 benchmark datasets

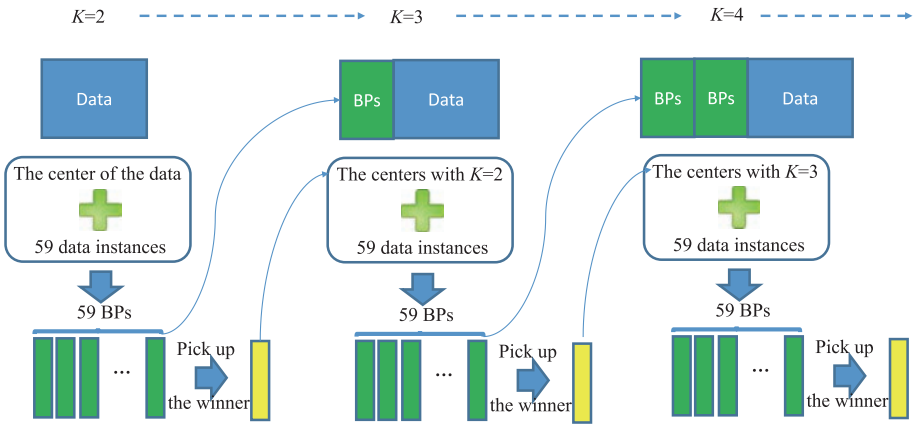


Fig. 1 Framework of GKCC.

show that GKCC outperforms other state-of-the-art clustering methods in terms of objective function values and external metrics.

2 Related Studies

The related studies in terms of consensus clustering and K-means initialization are described in this section.

2.1 Consensus clustering

Consensus clustering aims to find a single partition that agrees with several existing partitions as much as possible. Usually a utility function is designed to measure the agreements between basic partitions and the final consensus partition at the partition-level. In that case, consensus clustering can be formalized as a (combinational) optimization problem with a given objective function, and it typically uses heuristics to find approximate solutions. Many algorithms have been proposed to solve different objective functions including the Expectation-Maximization (EM) algorithm[12], non-negative matrix factorization[13], kernel-based methods[14], and simulated annealing[15]. Among these methods, a pioneering work attracted much attention[16], which uses K-means clustering to find the solution based on quadratic entropy[17]. Along this line, Ref. [9] provided a theoretical framework for KCC. Recently, the variants of KCC were proposed to enrich this area, such as DIsassemble-ASsemble (DIAS)[7], Spectral Ensemble Clustering (SEC)[10, 18], Entrpoy-based Consensus Clustering (ECC)[19], and Infinite Ensemble Clustering (IEC)[20, 21]. Although these methods achieved promising results, they all suffer from the sensitivity of K-means initialization. Another family of methods measures the similarity at the instance level. They define a co-association matrix to count the number of times two instances co-occurring in the same cluster; this can be regarded as a new similarity matrix. Then, any graph partition method can be applied on the co-association matrix to obtain the final result. Some of these methods include graph-based algorithms[5, 22], co-association matrix-based methods[8], relabeling and voting methods[23], locally adaptive cluster-based methods[24], genetic algorithm-based methods[25], spectral ensemble clustering[10], and many other methods. Notably, SEC[10, 18] combines these two types of consensus clustering methods and indicates that the similarity at different levels can be interconvertible. More details can be found in a survey reported in Ref. [26].

2.2 K-means initialization

Since K-means is sensitive to initialization, much efforts have been made to solve this challenge. The simplest way is to run multiple K-means algorithms with random initialization and return the one with the minimum objective function. K-means++ applies an adaptive sampling strategy to seed K initial centers; this provides an algorithm that performs $O(\log K)$-competitive algorithm with the optimal clustering[27]. Similarly, K-means|| samples several points each time with multiple runs and conducts a weighted K-means on these sampled points to produce K clusters for initialization[28]. Greedy K-means is an incremental approach to dynamically add one center at a time through a deterministic global search[11]. Other methods include nearest-neighbor based method[29], affinity propagation based method[30], and recursive K-means[31].

Although some studies have been conducted on the K-means initialization of consensus clustering, this is the first study to fully solve the initialization sensitivity of KCC. Inspired by the greedy allocation strategy of greedy K-means[11], we combine the virtues of greedy K-means and KCC to design our new algorithm, GKCC. The main idea

is to use the intermediate partitions discovered by greedy K-means to define the set of basic partitions and for the final fusion step. Moreover, the greedy strategy of greedy K-means completely resolves the initialization problem that plagues K-means-based approaches. Finally, some preliminary theoretical support is provided in the form of error bounds.

3 Preliminary Knowledge and Problem Definition

In this part, first the preliminary knowledge of KCC and greedy K-means is provided, and then the problem solved in this study is defined. Table 1 shows several key notations used in the following sections.

3.1 KCC

The goal of ensemble clustering is to find a single partition that agrees with the existing basic partitions as much as possible. It has been widely recognized that ensemble clustering generates robust partitions, finds bizarre clusters, handle noise, outliers, and sample variations, and integrates solutions from multiple distributed or incomplete sources of data or attributes[5, 16].

Unlike traditional clustering methods, which separate a group of data instances into different groups where the instances in the same group are much more similar to each other, ensemble clustering fuses several different partitions into a consensus partition. The input of traditional clustering methods is the data matrix, whereas the input of ensemble clustering is a set of basic partitions. Here basic partitions might be generated by the same clustering algorithm with different parameters, or by the same clustering algorithm with different features or even by several different clustering algorithms. Thus, ensemble clustering is a fusion problem, rather than a clustering problem.

Given a set of r basic partitions $\mathcal{H} = \{\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}, \cdots, \boldsymbol{H}^{(r)}\}$ of the data matrix $\boldsymbol{X}$, where the cluster number of $\boldsymbol{H}^{(i)}$ is K_i, the aim of consensus clustering is to fuse all the basic partitions into a consensus partition $\boldsymbol{H}^*$. It can be divided into two categories in terms of measuring the similarity in different levels.

Table 1 Notations.

Notation	Domain	Description
n	R	Instance number
m	R	Feature number
K	R	Cluster number
r	R	Basic partition number
$\boldsymbol{X}$	$\mathbf{R}^{n\times m}$	Data matrix
Π		Set of r basic partitions
$\boldsymbol{H}^{(i)}$	$\{0,1\}^{n\times K_i}$	i-th basic partition
$\boldsymbol{H}^*$	$\{0,1\}^{n\times K}$	Consensus partition
$\mathcal{C}$		Set of centers

The first category designs the utility function to measure the similarity between the final consensus partition and basic partitions. Usually the following formula is maximized to solve ensemble clustering.

$$\Gamma(\boldsymbol{H}^*, \mathcal{H}) = \sum_{i=1}^{r} U(\boldsymbol{H}^*, \boldsymbol{H}^{(i)}) \tag{1}$$

where $\Gamma : \mathcal{N}^{nK} \times \mathcal{N}^{nKr} \mapsto \mathrm{R}$ is a *consensus function*, and $U : \mathcal{N}^{nK} \times \mathcal{N}^{nK} \mapsto \mathrm{R}$ is a *utility function*, $i = 1, 2, \cdots, r$.

The selection of the utility function is critical for the success of a consensus clustering. In the literature, many external measures originally proposed for cluster validity have been used as the utility functions for consensus clustering, such as normalized mutual information[5], category utility function[32], quadratic mutual information[16], and Rand index[15]. These utility functions together with the consensus function mainly determine the quality of consensus clustering.

A *contingency matrix* is often used for computing the difference between two partitions. In Table 2, $n_{kj}^{(i)}$ denotes the number of data objects present in both cluster $C_j^{(i)}$ in $\boldsymbol{H}^{(i)}$ and cluster C_k in $\boldsymbol{H}^*$, $n_{k+} = \sum_{j=1}^{K_i} n_{kj}^{(i)}$, and $n_{+j}^{(i)} = \sum_{k=1}^{K} n_{kj}^{(i)}$, $1 \leqslant k \leqslant K$, $1 \leqslant j \leqslant K_i$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n$, $p_{k+} = n_{k+}/n$, and $p_{+j}^{(i)} = n_{+j}^{(i)}/n$. Then, we have the *normalized contingency matrix* for utility computation. For instance, the well-known category utility function[32] can be computed as follows:

$$U_c(\boldsymbol{H}^*, \boldsymbol{H}^{(i)}) = \sum_{k=1}^{K} p_{k+} \sum_{j=1}^{K_i} (p_{kj}^{(i)}/p_{k+})^2 - \sum_{j=1}^{K_i} (p_{+j}^{(i)})^2 \tag{2}$$

Thanks to the KCC[6], we can exactly transform the consensus clustering with categorical utility function into classical K-means clustering by introducing the following binary matrix. Let $\boldsymbol{B} = \{b(x)\}$ be a binary dataset derived from the set of r basic partitions $\mathcal{H}$ as follows:

Table 2 Contingency matrix.

		$\boldsymbol{H}^{(i)}$				
		$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	$\sum$
$\boldsymbol{H}^*$	C_1	$n_{11}^{(i)}$	$n_{12}^{(i)}$	$\cdots$	$n_{1K_i}^{(i)}$	n_{1+}
	C_2	$n_{21}^{(i)}$	$n_{22}^{(i)}$	$\cdots$	$n_{2K_i}^{(i)}$	n_{2+}
	$\cdot$	$\cdot$	$\cdot$	$\cdots$	$\cdot$	$\cdot$
	C_K	$n_{K1}^{(i)}$	$n_{K2}^{(i)}$	$\cdots$	$n_{KK_i}^{(i)}$	n_{K+}
	$\sum$	$n_{+1}^{(i)}$	$n_{+2}^{(i)}$	$\cdots$	$n_{+K_i}^{(i)}$	n

X	π_1	π_2	π_3	π_4
x_1	1	2	1	1
x_2	1	2	1	1
x_3	1	2	2	1
x_4	2	3	2	1
x_5	2	3	2	2
x_6	3	1	3	2
x_7	3	1	3	2

⇨

X	π_1			π_2			π_3			π_4		$w_{b(x)}$
x_1	1	0	0	0	1	0	1	0	0	1	0	12
x_2	1	0	0	0	1	0	1	0	0	1	0	12
x_3	1	0	0	0	1	0	0	1	0	1	0	13
x_4	0	1	0	0	0	1	0	1	0	1	0	11
x_5	0	1	0	0	0	1	0	1	0	0	1	10
x_6	0	0	1	1	0	0	0	0	1	0	1	9
x_7	0	0	1	1	0	0	0	0	1	0	1	9

Fig. 2 Illustration of the binary matrix in Eq. (3).

$$b(x) = \langle b(x)_1, \cdots, b(x)_r \rangle,$$
$$b(x)_i = \langle b(x)_{i1}, \cdots, b(x)_{iK_i} \rangle,$$
$$b(x)_{ij} = \begin{cases} 1, \text{ if } \boldsymbol{H}^{(i)}(x) = j; \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

Figure 2 shows an example of the binary matrix in Eq. (3). Then, K-means is conducted on $\boldsymbol{B}$ for the final consensus partition. KCC builds the connection between the fusion problem and clustering problem and solves the complex consensus clustering by the simplest K-means optimization[6].

3.2 Greedy K-means

The greedy K-means algorithm[11] is a highly effective variant of K-means and does not involve any random initialization. It starts with one center, trivially the centroid of all points, and centers are added incrementally until a desired maximum number of centers is reached. Each new center allocation involves a global search over all the points in the dataset to find the point that would maximally decrease the objective function if it were added as a new center. Faster versions of the global search are available by randomization (see the next section) or specialized data structures[11]. Greedy K-means completely solves the problem of initialization of traditional K-means variants and achieves exceptional performance in practice, typically performing much better than other popular algorithms such as K-means++[27].

A recent study[33] provides theoretical justification for greedy algorithms such as greedy K-means. Let f be the objective function of K-means for a given set S of cluster centers:

$$f(S) = \sum_{x \in \mathcal{X}} \min_{c \in S} ||x - c||^2 \tag{4}$$

When $S \subset \mathcal{X}$, the function f is weakly-1-supermodular; in this case, the following holds[33],

$$f(S_t) - f(S^*) \leqslant \left(1 - 1/k\right)^t \left(f(S_0) - f(S^*)\right) \tag{5}$$

where $S_t \subset \mathcal{X}$ is the set of cluster centers returned by a greedy algorithm in the $t+1$ iteration; S_0 is the initial set with only one element; $S^* \subset \mathcal{X}$ is the unknown optimal center set with k elements. Additional results are provided in Ref. [33] for the case of unconstrained S.

3.3 Problem definition

Inspiringly, KCC solves consensus clustering using classical K-means with theoretical supports. However, KCC inherits the simple and efficient character from K-means, while suffering from the sensitivity of K-means initialization. Moreover, KCC treats the basic partition generation and fusion as two separated processes. For greedy K-means, it produces too many intermediate partitions without further use and the time complexity of greedy K-means is $O(n^2)$. This prevents itself from handling large-scale datasets.

The above disadvantages of KCC and greedy K-means motivated us to propose a novel consensus clustering algorithm to seamlessly combine the merits and overcome the drawbacks of these two powerful tools. In this study, GKCC is proposed to solve the sensitivity of K-means initialization, high time complexity of greedy K-means, and separate process of consensus clustering into a unified framework.

4 GKCC

In this section, we introduce GKCC to solve the abovementioned challenges simultaneously. GKCC has three major features: (1) GKCC uses greedy K-means to generate basic partitions and initialize the centroids of KCC fusion. (2) Unlike greedy K-means, which generates partitions in the original feature space, GKCC concatenates the partitions from the previous stage with the original data for the next-stage basic partition generation. (3) The 59-sampling strategy is used to avoid the brute-force global search of greedy K-means.

Figure 1 shows the framework of GKCC. Because the optimal centroid for $K = 1$ is the center of the data, we start from $K = 2$ with the two centroids containing the center of the data and one randomly selected data instance. This process is repeated 59 times to generate 59 basic partitions (the reason for the selection of 59 will be illustrated by Lemma 1). For $K = 3$, the original data are augmented with the 59 basic partitions generated in the last stage. If the basic partitions and original data are directly connected, some problems arise because the data lie in the continuous feature space, whereas partitions exist in the discrete partition space. To solve this problem, the 1-of-K coding is used to transform the basic partitions into a binary matrix using Eq. (3). Recall that greedy K-means picks up the winner among the 59 basic partitions according to the objective function of K-means and then uses the centroid of the winner partition

to initialize the next stage. Here, the clustering is run in the augmented space, i.e., the dimensionalities of centroids with different cluster numbers are different. Although the original feature space and partition space are different, they both explore the cluster structure of the data. Notably, a one-to-one mapping relationship between partitions and centroids exists in different spaces. In the proposed GKCC method, the dimensionality gradually increases. At the very beginning, the data lie in the original space; then the original feature space is concatenated with the partition space to provide rich information. In the consecutive phase, we aim to initialize the centroids by that present in the last phase. However, the dimensionalities are different. Owing to the one-to-one mapping relationship between partitions and centroids, the partition is used in the last phase to initialize the centroids in the current space. Therefore, the winner partition obtained from the last stage ($K=2$) is used to calculate the two centroids and sample a new centroid in the augmented space for the initialization with $K=3$. The above process is repeated until the predefined user cluster number is achieved. Finally, all the basic partitions with different cluster numbers are collected.

A few remarks about the 59-sampling strategy are mentioned above. In the greedy K-means, when the datapoint is incrementally selected to add to the existing set of K-means centers, the datapoint that decreases the objective function the most is sought. However, a naive implementation of this scheme would require trying each datapoint to find the winner; this is clearly not possible for large-scale datasets (it scales as $O(n^2)$ for n datapoints). The 59-sampling strategy simply involves the sampling of 59 points out of the total n points and retaining the best point. This is motivated by the following (known) result.

Lemma 1 Given a global ranking of all the datapoints according to how much the objective function would decrease if a datapoint was added as a new center, at least $\lceil \log(1-\delta)/\log(1-p) \rceil$ datapoints should be selected without replacement to guarantee that; with a probability of at least δ, the best sampled datapoint is within the top p in the global ranking.

Proof Let A denote the event where at least one of the sampled datapoints is within the top p in the global ranking. If ν is the number of sampled datapoints, we have

$$\delta \leqslant P(A) = 1 - P(\overline{A}) \leqslant 1 - (1-p)^{\nu} \tag{6}$$

where the last inequality is due to the sampling without replacement. Hence, at least $\nu \geqslant \log(1-\delta)/\log(1-p)$ sampled datapoints are needed. The proof is thus completed. ■

Note that the above lemma holds for any value of n (the total number of points). Applying the lemma with $p = 0.05$ and $\delta = 0.95$ shows that we need to sample $\nu = 59$ different datapoints. During the basic partition generation of GKCC, the cluster number is varied from 2 to K to increase the diversity of basic partitions. Moreover, for a certain cluster number k, the $k-1$ centers are used from the previous stage plus the newly added center after the convergence of K-means from the randomly selected added point. In this manner, a meaningful partition is obtained for further fusion rather than some random partition. The 59-sampling strategy guarantees that in each step, the optimal point is selected with a probability of at least 95%.

Overall, during the first phase of GKCC, $59 \times (K-1)$ basic partitions are generated, a large enough number to obtain a robust consensus clustering in the second phase. By greedily adding the new centers, GKCC completely solves the sensitivity of K-means. Besides, the 59-sampling avoids the brute-force global search, and the discovered 59 partitions in each center allocation are used for consensus clustering without waste.

The time complexity of GKCC to generate the basic partitions is $O(InK^2m)$, where I is the average number of iterations, n is the number of points, K is the cluster number, and m is the number of features. The time complexity for consensus clustering is $O(InK^3)$. Note that $K \ll n$ and $m \ll n$. Therefore, the overall time complexity of GKCC is linear to n, i.e., which means that GKCC is suitable for large-scale clustering.

GKCC is summarized in Algorithm 1 that organizes the basic partition generation and KCC initialization in a unified framework.

5 Experimental Results

In this section, the effectiveness of GKCC on numerous benchmark datasets is demonstrated in terms of the objective function values of K-means and their standard deviations as well as two widely used external measurements. Besides, the basic partitions generated by our method are shown to outperform those derived from the RPS strategy.

5.1 Experimental setting

Datasets. 36 datasets from different domains were used for evaluation. Table 3 shows the characteristics of these datasets. Datasets 1–16 are gene expression data[19], datasets 17–26 are image datasets[21]; the last five

Algorithm 1: GKCC

Input: $\mathcal{X}$: training data,
K: the cluster number.

% Generate basic partitions
Let $\Pi = \emptyset$ be the set of basic partitions;
Let $\mathcal{C} = \emptyset$ be the set of centers;
$\mathcal{C} = \mathcal{C} \bigcup$ the center of $\mathcal{X}$;
Set $k = 2$;
while $k \leqslant K$ **do**
  Step 1. Sampling 59 points from $\mathcal{X}$, d_i with $1 \leqslant i \leqslant 59$.
  Step 2. Generate basic partitions and update centers.
    for $i = 1, \cdots, 59$ **do**
      $\mathcal{S} = \emptyset$;
      $\mathcal{C}' = \mathcal{C} \bigcup d_i$;
      Run K-means on $\mathcal{X}$ with the initial center $\mathcal{C}'$ and return the partition π' and the objective function value $objC_i$;
      $\mathcal{S} = \mathcal{S} \bigcup \pi'$;
    end
    $\Pi = \Pi \bigcup \mathcal{S}$;
    Build the binary matrix $\mathcal{S}_b$ of $\mathcal{S}$ by Eq. 3;
    $\mathcal{X} = [\mathcal{X}\ \ \mathcal{S}_b]$;
    Run K-means on $\mathcal{X}$ with the initial partition according to $\min_i objC_i$ and return the new centers as $\mathcal{C}$.
  Step 3. $k = k + 1$.
end
% Obtain consensus clustering
Build the binary matrix $\boldsymbol{B}$ of Π by Eq. 3;
Run K-means on $\mathcal{X}$ with the initial partition according to $\min_i objC_i$ and return the final partition π.

Output: Π and π.

datasets were obtained from the UCI Machine Learning Repository[18].

Tools. Here, our algorithm was tested in a consensus clustering framework. The compared algorithms include KCC[6] using standard K-means clustering with random initialization on binary matrix $\boldsymbol{B}$ and KCC++ using K-means++[27] for initialization. For fair comparison, same basic partitions derived from GKCC for other algorithms were used, and the cluster number K was set as the true cluster number. Note that each algorithm is run 20 times, and the average and standard deviations are reported.

Validity measurements. Both the objective function of K-means and two external measurements, Normalized Mutual Information (NMI) and normalized Rand index R_n, were used to evaluate the clustering performance[34].

The NMI measures the mutual information between the resulting cluster labels and ground truth labels, followed by a normalization operation to ensure that NMI ranges from 0 to 1. NMI is defined as follows:

$$\mathrm{NMI} = \frac{\sum_{i,j} n_{ij} \log \frac{n \cdot n_{ij}}{n_{i+} \cdot n_{+j}}}{\sqrt{(\sum_i n_{i+} \log \frac{n_{i+}}{n})(\sum_j n_{j+} \log \frac{n_{+j}}{n})}} \tag{7}$$

The R_n measures the similarity between two partitions in a statistical way as follows:

$$R_n = \frac{2\sum_{i,j} \binom{n_{ij}}{2} - 2\sum_i \binom{n_{i+}}{2} \cdot \sum_j \binom{n_{+j}}{2} / \binom{n}{2}}{\sum_i \binom{n_{i+}}{2} + \sum_j \binom{n_{+j}}{2} - 2\sum_i \binom{n_{i+}}{2} \cdot \sum_j \binom{n_{+j}}{2} / \binom{n}{2}} \tag{8}$$

Each variable is defined in Table 2. Both NMI and R_n are positive measurements, i.e., a better partition has a larger NMI or R_n value.

Table 3 36 benchmark datasets from different domains.

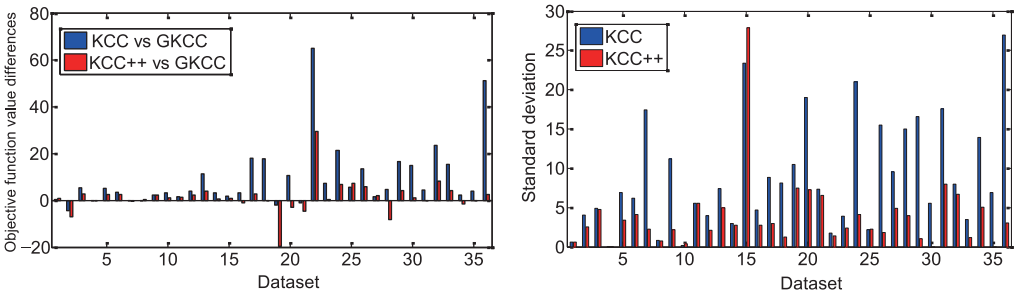
No.	Name	Instance	Feature	Cluster	No.	Name	Instance	Feature	Cluster
1	Alizadeh-2000-v1	42	1095	2	19	Caltech101	1415	4096	5
2	Alizadeh-2000-v1	62	2093	3	20	dslr_surf	157	800	10
3	Alizadeh-2000-v1	62	2093	4	21	ImageNet	7341	4096	5
4	Armstrong-2002-v1	72	1081	3	22	ORL	400	1024	40
5	Armstrong-2002-v1	72	2194	3	23	SUN09	3282	4096	5
6	Bhattacharjee-2001	203	1543	5	24	USPS	9298	256	10
7	Bredel-2005	50	1739	3	25	VOC2007	3376	4096	5
8	Dyrskjot-2003	40	1203	3	26	web_surf	295	800	10
9	Golub-1999-v1	72	1868	2	27	balance	625	4	3
10	Golub-1999-v2	72	1868	3	28	BreastTissue	106	9	6
11	Lapointe-2004	69	1652	3	29	ecoli	336	7	8
12	Risinger-2003	42	1771	4	26	glass	214	9	6
13	Tomlins-2006-v1	104	2315	5	31	iris	150	4	3
14	Tomlins-2006-v2	92	1288	4	32	pendigits	10 192	16	10
15	Yeoh-2002-v1	248	2526	2	33	satimage	4435	36	6
16	Yeoh-2002-v2	248	2526	7	34	waveform	5000	21	3
17	amazon_surf	958	800	10	35	wine	178	13	3
18	caltech_surf	1123	800	30	36	yeast	1484	6	10

5.2 Clustering performance

Here, the clustering performance is evaluated in terms of the objective function values and their standard deviations as well as external measurements.

Figure 3a shows the differences in objective function values among KCC, KCC++, and GKCC. For better visualization, the objective function values are condensed as $\sum_{k=1}^{K}\sum_{x\in\mathcal{C}_k}||x-m_k||^2/n^2$. The positive values in Fig. 3a indicate that GKCC has smaller objective function values of K-means, whereas the negative values indicate KCC or KCC++ outperforms GKCC. In most cases, GKCC excels KCC and KCC++ by a large margin even with a condensed objective function value. Figure 3b shows the standard deviation of KCC and KCC++. Two points should be mentioned: (i) Because the objective function values are adjusted by $1/n^2$, the standard deviations are adjusted by $1/n^4$ accordingly. Therefore, KCC and KCC++ suffer from large violability. (ii) The standard deviation of GKCC is zero. Although the basic partitions have some random factors, the final clustering is deterministic. Note that the core clustering method of KCC is the standard K-means, whereas the bound of KCC++ is based on the expectation. Consequently, KCC or KCC++ are run multiple times for a robust solution. For GKCC, running for one time is enough, and tremendous computation costs can be saved when dealing with large-scale datasets. Note that the deterministic solution is crucial for practical use.

Next, the performance of GKCC is evaluated in terms of external measurements, R_n and NMI. Figure 4 shows the clustering performance of different clustering methods on 36 benchmark datasets. Because R_n and NMI are both positive measurements, the higher values of R_n and NMI indicate a better clustering performance. GKCC almost achieved the best performance on 36 benchmark datasets. Notably, GKCC outperforms KCC++ over 40% on dataset-2 in terms of R_n and exceeds KCC over 50% on dataset-15 in terms of NMI, exhibiting the benefit of the clustering solution with a lower K-means objective function. These experimental results verify the motivation of our study. By



(a) Differences in objective function values (b) Standard deviation of objective function values

Fig. 3 Differences in objective function values and standard deviations among different clustering methods on 36 benchmark datasets.

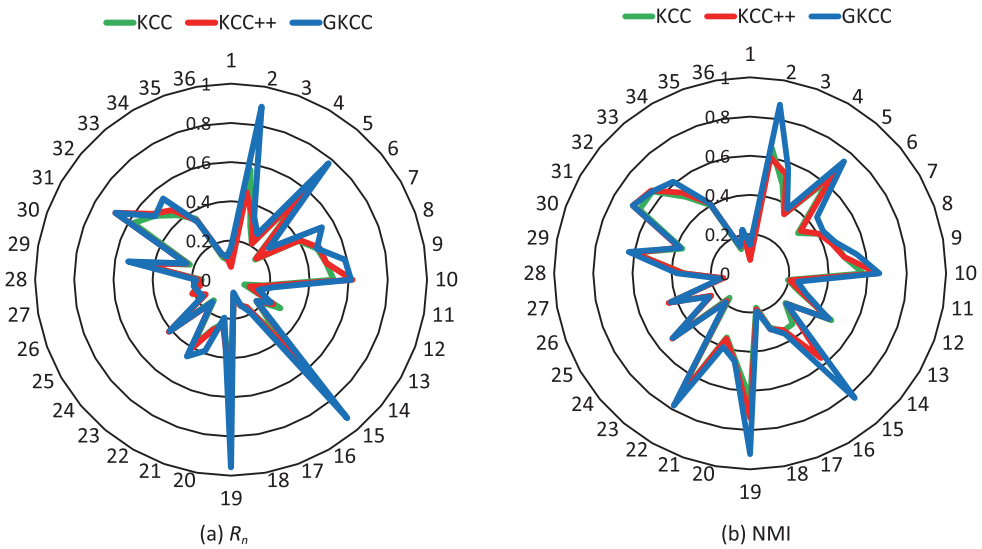


Fig. 4 Clustering performance of different clustering methods on 36 benchmark datasets in terms of R_n and NMI.

Table 4 Average clustering performance of three clustering methods on 36 benchmark datasets.

Measurement	KCC	KCC++	GKCC
R_n	0.2986	0.3139	0.3710
NMI	0.3965	0.4147	0.4560

combining KCC and greedy K-means, GKCC can deliver high-quality partitions for practical use. Table 4 shows the average clustering performance of three clustering methods on 36 benchmark datasets. GKCC has significant advantages over KCC and KCC++ in the average level.

Finally, the benefit of greedy basic partition generation over the widely used RPS generation is demonstrated. RPS uses a single clustering method such as K-means with different cluster numbers to generate a set of basic partitions. Here, the cluster numbers were varied from 2 to K, and the same number of basic partitions was generated as the greedy strategy for fair comparison. Figure 5 shows the clustering performance with different basic partition generation strategies on the first 10 datasets for limited space concern. These two different strategies are competitive. For example, on dataset-1, RPS outperforms the greedy strategy by over 30% in terms of R_n. This can be attributed to the small cluster number of dataset-1. The greedy strategy exceeds RPS by over 30% on dataset-2. Thus, this our greedy strategy delivers high-quality basic partitions for later fusion. Table 5 shows the average performance of these two strategies on 36 benchmark datasets. Although RPS can provide a diverse set of basic partitions by varying the clustering number, the greedy strategy might generate high-quality basic partitions owing

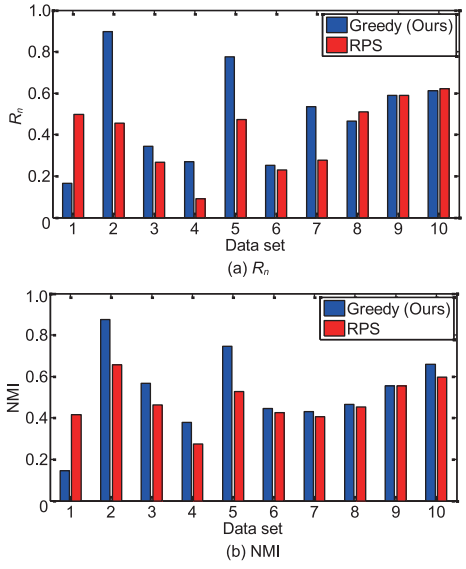


Fig. 5 Clustering performance with different basic partition generation strategies on the first 10 datasets.

Table 5 Average clustering performance with different basic partition generation strategies on 36 benchmark datasets.

Measurement	RPS	Greedy (Ours)
R_n	0.3135	0.3710
NMI	0.4151	0.4560

to better initialization. Moreover, the 59-sampling strategy also provides diverse basic partitions.

6 Conclusion

In this paper, a novel consensus clustering method GKCC was proposed. GKCC solved the sensitivity of KCC and combined the basic partition generation and fusion in a unified framework. Derived by greedy K-means and KCC, GKCC inherited the merits and overcomed the drawbacks of its precursors. Moreover, a 59-sampling strategy was used to further accelerate the speed. Extensive experiments on 36 benchmark datasets demonstrated the effectiveness of GKCC over KCC and KCC++ in terms of objective function values and their standard deviations as well as external measurements.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (No. 71471009).

References

[1] A. K. Jain, Data clustering: 50 years beyond K-means, *Patt. Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[2] J. MacQueen, Some methods for classification and analysis of multivariate observations, in *Proc. 5^{th} Berkeley Symp. Mathematical Statistics and Probability*, Berkeley, CA, USA, 1967.

[3] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. India: Pearson Education, 2006.

[4] M. Ester, H. P. Kriegel, J. Sander, and X. W. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in *Proc. 2^{nd} Int. Conf. Knowledge Discovery and Data Mining*, München, Germany, 1996.

[5] A. Strehl and J. Ghosh, Cluster ensembles— A knowledge reuse framework for combining multiple partitions, *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2003.

[6] J. J. Wu, H. F. Liu, H. Xiong, J. Cao, and J. Chen, K-means-based consensus clustering: A unified view, *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 155–169, 2015.

[7] H. F. Liu, J. J. Wu, D. C. Tao, Y. C. Zhang, and Y. Fu, Dias: A disassemble-assemble framework for highly sparse text

clustering, in *Proc. 2015 SIAM Int. Conf. Data Mining*, 2015.

[8] A. L. N. Fred and A. K. Jain, Combining multiple clusterings using evidence accumulation, *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, 2005.

[9] J. J. Wu, H. F. Liu, H. Xiong, and J. Cao, A theoretic framework of K-means-based consensus clustering, in *Proc. 23^{rd} Int. Joint Conf. Artificial Intelligence*, Beijing, China, 2013.

[10] H. F. Liu, T. L. Liu, J. J. Wu, D. C. Tao, and Y. Fu, Spectral ensemble clustering, in *Proc. 21^{th} ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Sydney, NSW, Australia, 2015.

[11] A. Likas, N. Vlassis, and J. J. Verbeek, The global K-means clustering algorithm, *Patt. Recognit.*, vol. 36, no. 2, pp. 451–461, 2003.

[12] A. Topchy, A. K. Jain, and W. Punch, A mixture model for clustering ensembles, in *Proc. 4^{th} SIAM Int. Conf. Data Mining*, Lake Buena Vista, FL, USA, 2004.

[13] T. Li, C. Ding, and M. I. Jordan, Solving consensus and Semi-supervised clustering problems using nonnegative matrix factorization. in *Proc. 7^{th} IEEE Int. Conf. Data Mining*, Omaha, NE, USA, 2007.

[14] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper, Weighted partition consensus via kernels, *Patt. Recognit.*, vol. 43, no. 8, pp. 2712–2724, 2010.

[15] Z. W. Lu, Y. X. Peng, and J. G. Xiao, From comparing clusterings to combining clusterings, in *Proc. 23^{rd} National Conf. Artificial Intelligence*, Chicago, IL, USA, 2008, pp. 665–670.

[16] A. Topchy, A. K. Jain, and W. Punch, Combining multiple weak clusterings, in *Proc. 3^{rd} IEEE Int. Conf. Data Mining*, Melbourne, FL, USA, 2003.

[17] A. Gionis, H. Mannila, and P. Tsaparas, Clustering aggregation, *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 4, 2007.

[18] H. F. Liu, J. J. Wu, T. L. Liu, D. C. Tao, and Y. Fu, Spectral ensemble clustering via weighted K-means: Theoretical and practical evidence, *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1129–1143, 2017.

[19] H. F. Liu, R. Zhao, H. S. Fang, F. X. Cheng, Y. Fu, and Y. Y. Liu, Entropy-based consensus clustering for patient stratification, *Bioinformatics*, vol. 33, no. 17, pp. 2691–2698, 2017.

[20] H. F. Liu, M. Shao, S. Li, and Y. Fu, Infinite ensemble for image clustering, in *Proc. 22^{nd} ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016.

[21] H. F. Liu, M. Shao, S. Li, and Y. Fu, Infinite ensemble clustering, *Data Min. Knowl. Discov.*, vol. 32, no. 2, pp. 385–416, 2018.

[22] X. L. Z. Fern and C. E. Brodley, Random projection for high dimensional data clustering: A cluster ensemble approach, in *Proc. 20^{th} Int. Conf. Machine Learning*, Washington, DC, USA, 2013.

[23] H. G. Ayad and M. S. Kamel, Cumulative voting consensus method for partitions with variable number of clusters, *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 30, no. 1, pp. 160–173, 2008.

[24] C. Domeniconi and M. Al-Razgan, Weighted cluster ensembles: Methods and analysis, *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 4, p. 17, 2009.

[25] H. S. Yoon, S. Y. Ahn, S. H. Lee, S. B. Cho, and J. Kim, Heterogeneous clustering ensemble method for combining different cluster results, in *Proc. 2006 Int. Conf. Data Mining for Biomedical Applications*, 2006, pp. 82–92.

[26] S. Vega-Pons and J. Ruiz-Shulcloper, A survey of clustering ensemble algorithms, *Int. J. Patt. Recogn. Artif. Intell.*, vol. 25, no. 3, pp. 337–372, 2011.

[27] D. Arthur and S. Vassilvitskii, K-means++: The advantages of careful seeding. in *Proc. 18^{th} Annual ACM-SIAM Symp. Discrete Algorithms*, New Orleans, LA, USA, 2007.

[28] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, Scalable K-means++, *Proc. VLDB Endow.*, vol. 5, no. 7, pp. 622–633, 2012.

[29] O. Kettani, B. Tadili, and F. Ramdani, A deterministic K-means algorithm based on nearest neighbor search, *Int. J. Comput. Appl.*, vol. 63, no. 15, pp. 33–37, 2013.

[30] Y. Zhu, J. Yu, and C. Y. Jia, Initializing K-means clustering using affinity propagation, in *Proc. 9^{th} IEEE Int. Conf. Hybrid Intelligent Systems*, Shenyang, China, 2009.

[31] M. Capó, A. Pérez, and J. A. Lozano, A recursive K-means initialization algorithm for massive data, in *Proc. Spanish Association for Artificial Intelligence*, 2015.

[32] B. Mirkin, Reinterpreting the category utility function, *Mach. Learn.*, vol. 45, no. 2, pp. 219–228, 2001.

[33] C. Boutsidis, E. Liberty, and M. Sviridenko, Greedy minimization of weakly supermodular set functions, arXiv preprint arXiv: 1502.06528, 2015.

[34] J. J. Wu, H. Xiong, and J. Chen, Adapting the right measures for K-means clustering, in *Proc. 15^{th} ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Paris, France, 2009.

Xue Li received the B.E. degree in 2011 from Beihang University, and earned the PhD degree in 2018 from Tsinghua University. Her research interests generally focus on optimization and operations research, and supply chain management.

Hongfu Liu received the bachelor and master degrees in management information systems from Beihang University, in 2011 and 2014, respectively. He is currently pursuing the PhD degree in Northeastern University, Boston. His research interests generally focus on data mining and machine learning, with special interests in ensemble learning.