K-means-based Consensus Clustering: A Unified View

Junjie Wu, *Member, IEEE,* Hongfu Liu, Hui Xiong, *Senior Member, IEEE,* Jie Cao, Jian Chen, *Fellow, IEEE*

Abstract—The objective of consensus clustering is to find a single partitioning which agrees as much as possible with existing basic partitionings. Consensus clustering emerges as a promising solution to find cluster structures from heterogeneous data. As an efficient approach for consensus clustering, the K-means based method has garnered attention in the literature, however the existing research efforts are still preliminary and fragmented. To that end, in this paper, we provide a systematic study of K-means-based Consensus Clustering (KCC). Specifically, we first reveal a necessary and sufficient condition for utility functions which work for KCC. This helps to establish a unified framework for KCC on both complete and incomplete data sets. Also, we investigate some important factors, such as the quality and diversity of basic partitionings, which may affect the performances of KCC. Experimental results on various real-world data sets demonstrate that KCC is highly efficient and is comparable to the state-of-the-art methods in terms of clustering quality. In addition, KCC shows high robustness to incomplete basic partitionings with many missing values.

Index Terms—Consensus Clustering, K-means, Utility Function

1 Introduction

Consensus clustering, also known as *cluster ensemble* or *clustering aggregation*, aims to find a single partitioning of data from multiple existing basic partitionings [1]. It has been widely recognized that consensus clustering can help to generate robust clustering results, find bizarre clusters, handle noise, outliers and sample variations, and integrate solutions from multiple distributed sources of data or attributes [2].

Consensus clustering is a combinatorial optimization problem in essence, and in some special cases, e.g., using median partition with the Mirkin distance, it has been proven to be NP-complete [3]. In the literature, many algorithms have been proposed to address the computational challenges, such as the co-association matrix-based methods [4], the graph-based methods [1], the prototype-based methods [5], and other heuristic approaches [6], [7], [8]. Among these research efforts, the K-means-based method proposed in [5] is of particular interests, for its simplicity and high efficiency inherited from the classic K-means clustering method [27]. However, the existing studies are still preliminary and fragmented. Indeed, the general theoretic framework of utility functions suitable for *K-means-based consensus clustering* (KCC) is yet not available. Also, the understanding of key factors, which have significant impact on the performances of KCC, is still limited.

To fulfill this crucial void, in this paper, we provide a systematic study of K-means-based consensus clustering. The major contributions are summarized as follows. First, we formally define the concept of KCC, and provide a necessary and sufficient condition for utility functions which are suitable for KCC. Based on this condition, we can easily derive a *KCC utility function* from a continuously differentiable convex function, which helps to establish a unified framework for KCC, and makes it a systematic solution. Second, we adjust the computations of the utility functions and distance functions so as to extend the applicable scope of KCC to the cases where there exist severe data incompleteness. Third, we empirically explore the major factors which may affect the performances of KCC, and obtain some practical guidance from specially designed experiments on various real-world data sets.

Extensive experiments on various real-world data sets demonstrate that: (a) KCC is highly efficient and is comparable to the state-of-the-art methods in terms of clustering quality; (b) Multiple utility functions indeed improve the usability of KCC on different types of data, while we find that the utility functions based on Shannon entropy generally have more robust performances; (c) KCC is very robust even if there exist very few high-quality basic partitionings or severely incomplete basic partitionings; (d) The choice of the generation strategy for basic partitionings is critical to the success of KCC; (e) The number, quality and diversity of basic partitionings are three major factors that affect the performances of KCC, although the impact from them are different.

Overview. The remainder of this paper is organized as follows. In Section 2, we present the relate work. Section 3 gives some preliminaries and proposes the problem to study. In Section 4, we introduce the necessary and sufficient condition

Junjie Wu and Hongfu Liu are with the Department of Information Systems, School of Economics and Management, Beihang University, China. E-mail: wujj@buaa.edu.cn.
Hui Xiong is with the Department of Management Science and Information Systems, Rutgers University, USA. E-mail: hxiong@rutgers.edu.
Jie Cao is with Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, China. E-mail: caojie690929@163.com.
Jian Chen is with the Department of Management Science and Engineering, Tsinghua University, China. E-mail: chenj@sem.tsinghua.edu.cn.
This work was partially supported by the National Natural Science Foundation of China (NSFC) (Grant Nos. 71171007, 70901002, 71031001 and 71072172), and by the Foundation for the Author of National Excellent Doctoral Dissertations of China (Grant No. 201189). The third and fifth authors were partially supported by NSFC (Grant No. 71028002).

for the utility functions of KCC. Section 5 addresses the challenges caused by missing data. We show experimental results in Section 6, and finally conclude the work in Section 7.

2 Related Work

Tremendous research efforts have been devoted to consensus clustering (CC). These studies can be roughly divided into two categories: CC with implicit Objectives (CCIO) and CC with explicit Objectives (CCEO).

The methods in CCIO do not set any global objective functions for CC. Rather, they employ heuristics to find approximate solutions. In the pioneer work, [1] developed three graph-based algorithms for CC. Although an objective function was defined on the normalized mutual information measure, the proposed algorithms actually do not address this optimization problem directly [9]. Following this idea, Refs. [10] and [11] built different types of graphs to improve the clustering quality. Another class of solutions in CCIO is based on the similarity matrix. For instance, [4] summarized the information of basic partitionings into a co-association matrix, based on which the agglomerative hierarchical clustering was used to find the final clustering. Some work along this line has been proposed subsequently, with the focus either on improving hierarchical clustering [12], or on building more informative co-association matrix [13]. Other CCIO methods include Relabeling and Voting [6], [14], Locally Adaptive Cluster based methods [15], fuzzy clustering based methods [16], genetic algorithm based methods [17], and still many more.

The methods in CCEO have explicit global objective functions for consensus clustering. For instance, to find the Median Partition based on Mirkin distance [18], [19] proposed three simple heuristics. Along this line, [20] further proposed some new heuristics for enhancement. The comparative studies on some of these heuristics can be found in [21]. In the inspiring work, [5] proposed an objective function based on the Category Utility Function for consensus clustering, and used K-means clustering to find the solution. This elegant idea could be traced back to the work by Mirkin [22]. They further extended their work to using the expectation-maximization algorithm with a finite mixture of multinomial distributions for consensus clustering [23]. In addition, there are some other interesting objective functions for consensus clustering, such as the ones which can be solved by nonnegative matrix factorization [7], kernel-based methods [24], simulated annealing [8], and genetic algorithms [25], respectively.

There are still many other algorithms for consensus clustering. Readers with interests can refer to some survey papers [9], [26]. In general, compared with CCIO methods, CCEO methods might offer better interpretability and greater robustness to clustering results, via the guidance of the objective functions. However, it is often very hard for CCEO methods to make a balance between the high execution efficiency and high clustering quality. More importantly, each CCEO method typically works for one objective function, which seriously restricts the applicability to real-life applications in different domains. These indeed motivate our study in this paper, which attempts to build a general theoretic framework for K-means-based consensus clustering using multiple utility functions.

3 Preliminaries and Problem Definition

Here we briefly introduce the basic concepts of consensus clustering and formulate the research problem of this paper.

3.1 Consensus Clustering

Let $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ denote a set of data objects/points/instances. A partitioning of $\mathcal{X}$ into K crisp clusters is represented as a collection of K subsets of objects in $\mathcal{C} = \{C_k | k = 1, \cdots, K\}$, with $C_k \bigcap C_{k'} = \emptyset$, $\forall k \neq k'$, and $\bigcup_{k=1}^{K} C_k = \mathcal{X}$, or as a label vector $\pi = \langle L_\pi(x_1), \cdots, L_\pi(x_n)\rangle$, where $L_\pi(x_i)$ maps x_i to one of the K labels in $\{1, 2, \cdots, K\}$. We also use some conventional math notations as follows. For instance, $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{R}_{++}$, $\mathbb{R}^d$, and $\mathbb{R}^{nd}$ denote the sets of reals, non-negative reals, positive reals, d-dimensional real vectors, and $n \times d$ real matrix, respectively. $\mathbb{Z}$ denotes the set of integers, and $\mathbb{Z}_+$, $\mathbb{Z}_{++}$, $\mathbb{Z}^d$ and $\mathbb{Z}^{nd}$ are defined analogously. For a d-dimensional real vector x, $\|x\|_p$ denotes the L_p norm of x, i.e., $\|x\|_p = \sqrt[p]{\sum_{i=1}^{d} x_i^p}$, $|x|$ denotes the cardinality of x, i.e., $|x| = \sum_{i=1}^{d} x_i$, and x^T denotes the transposition of x. The gradient of a single variable function f is denoted as ∇f, and the logarithm of based 2 is denoted as $\log$.

In general, the existing consensus clustering methods can be categorized into two classes, i.e., the methods with or without global objective functions [9]. In this paper, we are concerned with the former methods, which are typically formulated as a combinatorial optimization problem as follows. Given r basic partitionings of $\mathcal{X}$ (a *basic partitioning* is a crisp partitioning of $\mathcal{X}$ by some clustering algorithm) in $\Pi = \{\pi_1, \pi_2, \cdots, \pi_r\}$, the goal is to find a consensus partitioning π such that

$$\Gamma(\pi, \Pi) = \sum_{i=1}^{r} w_i U(\pi, \pi_i) \tag{1}$$

is maximized, where $\Gamma : \mathbb{Z}_{++}^n \times \mathbb{Z}_{++}^{nr} \mapsto \mathbb{R}$ is a *consensus function*, $U : \mathbb{Z}_{++}^n \times \mathbb{Z}_{++}^n \mapsto \mathbb{R}$ is a *utility function*, and $w_i \in [0, 1]$ is a user-specified *weight* for π_i, with $\sum_{i=1}^{r} w_i = 1$. Sometimes a distance function, e.g., the well-known Mirkin distance [18], rather than a utility function is used in the consensus function. In that case, we can simply turn the maximization problem into a minimization problem without changing the nature of the problem.

Consensus clustering as a combinatorial optimization problem is often solved by some heuristics and/or some metaheuristics. Therefore, the choice of the utility function in Eq. (1) is crucial for the success of a consensus clustering, since it largely determines the heuristics to employ. In the literature, some external measures originally proposed for cluster validity have been adopted as utility functions for consensus clustering, e.g., the Normalized Mutual Information [1], Category Utility Function [22], Quadratic Mutual Information [5], and Rand Index [8]. These utility functions of different math properties pose computational challenges to consensus clustering.

3.2 K-means Clustering

K-means [27] is a prototype-based partitional technique to find user-specified K crisp clusters. These clusters are represented

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Sample Instances of the Point-to-Centroid Distance

$\phi(x)$	dom(ϕ)	$f(x,y)$	Distance
$\|x\|_2^2$	$\mathbb{R}^d$	$\|x-y\|_2^2$	squared Euclidean distance
$-H(x)$	$\{x \mid x \in \mathbb{R}^d_{++}, \|x\|=1\}$	$\sum_{j=1}^d x_j \log \frac{x_j}{y_j}$	KL-divergence
$\|x\|_2$	$\mathbb{R}^d_{++}$	$\|x\|_2(1-\cos(x,y))$	cosine distance
$\|x\|_p$, $p>1$	$\mathbb{R}^d_{++}$	$\phi(x) - \frac{\sum_{j=1}^d x_j y_j^{p-1}}{\phi(y)^{p-1}}$	L_p distance

Note: (1) H: Shannon entropy; (2) cos: cosine similarity.

by their centroids, e.g., the arithmetic means of data points in the respective clusters. K-means can be also viewed as a heuristic to optimize the following objective function:

$$\min \sum_{k=1}^{K} \sum_{x \in C_k} f(x, m_k), \tag{2}$$

where m_k is the centroid of the kth cluster C_k, and f is the distance from a data point to a centroid. The clustering process of K-means is a two-phase iterative heuristic, with *data assignment* and *centroid update* staggering successively.

It is interesting to note that the choice of distance functions in Eq. (2) is closely related to the choice of centroid types in K-means, given that the convergence of the two-phase iteration must be guaranteed [29]. Since the arithmetic mean has higher computational efficiency and better analytical properties, we hereby limit our study to the *classic K-means* with arithmetic centroids. We call a distance function *fits* K-means if this function corresponds to the centroids of arithmetic means.

It has been shown that the Bregman divergence [30] as a family of distances fits the classic K-means [31]. That is, let $\phi : \mathbb{R}^d \mapsto \mathbb{R}$ be a differentiable, strictly-convex function, then the Bregman loss function $f : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ defined by

$$f(x,y) = \phi(x) - \phi(y) - (x-y)^T \nabla\phi(y) \tag{3}$$

fits K-means clustering. More importantly, under some assumptions including the *unique minimizer* assumption for the centroids, the Bregman divergence is the only distance that fits K-means [32]. Nevertheless, the strictness of the convexity of ϕ can be further relaxed, if the unique minimizer assumption reduces to the non-unique case. This leads to the more general "point-to-centroid distance" derived from convex but not necessarily strictly convex ϕ [33], although it has the same mathematical expression as the Bregman divergence in Eq. (3). As a result, we can reasonably assume that the distance function f in Eq. (2) is an instance of the point-to-centroid distance. Table 1 lists some important instances of the point-to-centroid distance.

3.3 Problem Definition

In general, there are three key issues for a consensus clustering algorithm: accuracy, efficiency, and flexibility. *Accuracy* means the algorithm should be able to find a high-quality partitioning from a large combinatorial space. The simple heuristics, such as the Best-of-r algorithm that only selects the best partitioning from the r basic partitionings, usually cannot guarantee the quality. *Efficiency* is another big challenge to consensus clustering, especially for the algorithms that employ some complicated meta-heuristics, such as the genetic algorithm, simulated annealing, and the particle swarm algorithm. *Flexibility* requires the algorithm to serve as a framework open to various types of utility functions. This is important for the applicability of the algorithm to a wide range of application domains. However, the flexibility issue has seldom been addressed in the literature, since most of the existing algorithms either have no objective functions, or are designed purposefully for one specific utility function.

Then, here is the problem: *Can we design a consensus clustering algorithm that can address the three problems simultaneously?* The forementioned K-means algorithm indeed provides an interesting clue. If we can somehow transform the consensus clustering problem into a K-means clustering problem, we can then make use of the two-phase iteration heuristic of K-means to find a good consensus partitioning in an efficient way. Moreover, as indicated by Eq. (3), the point-to-centroid distance of K-means is actually a family of distance functions derived from different convex functions (ϕ). This implies that if the distance function of K-means can be mapped to the utility function of consensus clustering, we can obtain multiple utility functions for consensus clustering in a unified framework, which will provide great flexibility.

In light of this, in this paper, we focus on building a general framework for consensus clustering using K-means, referred to as the *K-means-based Consensus Clustering* (KCC) method. We are concerned with the following questions:

- How to transform a consensus clustering problem to a K-means clustering problem?
- What is the necessary and sufficient condition for this transformation?
- How to adapt KCC to the situations where there exist incomplete basic partitionings?

Here "incomplete basic partitioning" means a basic partitioning that misses some of the data labels. It may due to the unavailability of some data objects in a distributed or time-evolving system, or simply the loss of some data labels in a knowledge reuse process [1].

Note that few studies have investigated consensus clustering from a KCC perspective except for Ref. [5], which demonstrated that using the special Category Utility Function, a consensus clustering can be equivalently transformed to a K-means clustering with the squared Euclidean distance [22]. However, it did not establish a unified framework for KCC, neither did it explore the general property of utility functions that fit KCC. Moreover, it did not showcase how to handle incomplete basic partitionings, which as shown in a later section is indeed a big challenge to KCC. We therefore attempt to fill these crucial voids.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
The Contingency Matrix

π \ π_i	$C_1^{(i)}$	$C_2^{(i)}$	$\cdots$	$C_{K_i}^{(i)}$	$\sum$
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

4 Utility Functions for K-means-based Consensus Clustering

In this section, we establish the general framework of K-means-based consensus clustering (KCC). Specifically, we propose the necessary and sufficient condition for a utility function suitable for KCC, and link it to K-means clustering.

4.1 Consensus Clustering to K-means Clustering

We begin by introducing the notion of *contingency matrix*. A contingency matrix is actually a co-occurrence matrix for two discrete random variables. It is often used for computing the difference or similarity between two partitionings in cluster validity. Table 2 shows an example.

In Table 2, we have two partitionings: π and π_i, containing K and K_i clusters, respectively. In the table, $n_{kj}^{(i)}$ denotes the number of data objects belonging to both cluster $C_j^{(i)}$ in π_i and cluster C_k in π, $n_{k+} = \sum_{j=1}^{K_i} n_{kj}^{(i)}$, and $n_{+j}^{(i)} = \sum_{k=1}^{K} n_{kj}^{(i)}$, $1 \le j \le K_i$, $1 \le k \le K$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n$, $p_{k+} = n_{k+}/n$, and $p_{+j}^{(i)} = n_{+j}^{(i)}/n$, we then have the *normalized contingency matrix* (NCM), based on which a wide range of utility functions can be defined accordingly. For instance, the well-known Category Utility Function [22] can be computed as:

$$U_c(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+} \sum_{j=1}^{K_i} (\frac{p_{kj}^{(i)}}{p_{k+}})^2 - \sum_{j=1}^{K_i} (p_{+j}^{(i)})^2. \quad (4)$$

We then introduce the *binary data set*. Let $\mathcal{X}^{(b)} = \{x_l^{(b)} | 1 \le l \le n\}$ be a binary data set derived from the set of r basic partitionings Π as follows:

$$x_l^{(b)} = \langle x_{l,1}^{(b)}, \cdots, x_{l,i}^{(b)}, \cdots, x_{l,r}^{(b)} \rangle, \text{ with} \quad (5)$$

$$x_{l,i}^{(b)} = \langle x_{l,i1}^{(b)}, \cdots, x_{l,ij}^{(b)}, \cdots, x_{l,iK_i}^{(b)} \rangle, \text{ and} \quad (6)$$

$$x_{l,ij}^{(b)} = \begin{cases} 1, & \text{if } L_{\pi_i}(x_l) = j \\ 0, & \text{otherwise} \end{cases}, \quad (7)$$

where "$\langle\ \rangle$" indicates a transversal vector. Therefore, $\mathcal{X}^{(b)}$ is an $n \times \sum_{i=1}^{r} K_i$ binary data matrix with $|x_{l,i}^{(b)}| = 1,\ \forall\ l, i$.

Now suppose we have a partitioning π with K clusters generated by running K-means on $\mathcal{X}^{(b)}$. Let m_k denote the centroid of the kth cluster in π, which is a $\sum_{i=1}^{r} K_i$-dimensional vector as follows:

$$m_k = \langle m_{k,1}, \cdots, m_{k,i}, \cdots, m_{k,r} \rangle, \text{ with} \quad (8)$$

$$m_{k,i} = \langle m_{k,i1}, \cdots, m_{k,ij}, \cdots, m_{k,iK_i} \rangle, \quad (9)$$

$1 \le j \le K_i$, $1 \le i \le r$, and $1 \le k \le K$. We then link $\mathcal{X}^{(b)}$ to the contingency matrix by formalizing a lemma as follows:

Lemma 1: For K-means clustering on $\mathcal{X}^{(b)}$, the centroids

$$m_{k,i} = \left\langle \frac{p_{k1}^{(i)}}{p_{k+}}, \cdots, \frac{p_{kj}^{(i)}}{p_{k+}}, \cdots, \frac{p_{kK_i}^{(i)}}{p_{k+}} \right\rangle, \forall\ k, i. \quad (10)$$

Proof: According to the definition of centroids in K-means clustering, we have

$$m_{k,ij} = \frac{\sum_{x_l \in C_k} x_{l,ij}^{(b)}}{|C_k|}. \quad (11)$$

Recall the contingency matrix in Table 2, we have $|C_k| = n_{k+}$, and $\sum_{x_l \in C_k} x_{l,ij}^{(b)} = |C_k \bigcap C_j^{(i)}| = n_{kj}^{(i)}$. As a result,

$$m_{k,ij} = \frac{n_{kj}^{(i)}}{n_{k+}} = \frac{p_{kj}^{(i)}}{p_{k+}}, \quad (12)$$

and Eq. (10) thus follows. □

Remark 1: While Lemma 1 is very simple, it unveils important information critical to the construction of KCC framework. That is, using the binary data set $\mathcal{X}^{(b)}$ as the input for K-means clustering, the resulting centroids can be computed upon the elements in the contingency matrices, from which a consensus function can be also defined. In other words, the contingency matrix and the binary data set together serve as a bridge that removes the boundary between consensus clustering and K-means clustering.

We then have the following important definition:

Definition 1 (KCC Utility Function): A utility function U is a KCC utility function, if $\forall\ \Pi = \{\pi_1, \cdots, \pi_r\}$ and $K \ge 2$, there exists a distance function f such that

$$\max_{\pi \in F} \sum_{i=1}^{r} w_i U(\pi, \pi_i) \Longleftrightarrow \min_{\pi \in F} \sum_{k=1}^{K} \sum_{x_l \in C_k} f(x_l^{(b)}, m_k) \quad (13)$$

holds for any feasible region F.

Remark 2. Definition 1 specifies the key property of a KCC utility function; that is, it should help to transform a consensus clustering problem to a K-means clustering problem. In other words, KCC is a solution to consensus clustering, which takes a KCC utility function to define the consensus function, and relies on the K-means heuristic to find the consensus partitioning.

4.2 The Derivation of KCC Utility Functions

Here we derive the KCC utility functions, and give some examples that can be commonly used in real-world applications. We first give the following lemma:

Lemma 2: A utility function U is a KCC utility function, if and only if $\forall\ \Pi$ and $K \ge 2$, there exist a differentiable convex function ϕ and a strictly increasing function g_Π such that

$$\sum_{i=1}^{r} w_i U(\pi, \pi_i) = g_\Pi(\sum_{k=1}^{K} p_{k+} \phi(m_k)). \quad (14)$$

Proof: We begin the proof by giving one important fact as follows. Suppose f is a distance function that fits K-means clustering. Then according to Eq. (3), f is a point-to-centroid

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

distance that can be derived by a differentiable convex function ϕ. Therefore, if we substitute f in Eq. (3) into the right-hand-side of Eq. (13), we have

$$\begin{aligned}
&\sum_{k=1}^{K}\sum_{x_l\in C_k} f(x_l^{(b)}, m_k)\\
&=\sum_{k=1}^{K}\sum_{x_l\in C_k} \phi(x_l^{(b)}) - \phi(m_k) - (x_l^{(b)} - m_k)^T\nabla\phi(m_k)\\
&\overset{(\alpha)}{=}\sum_{k=1}^{K}\sum_{x_l\in C_k} \phi(x_l^{(b)}) - \sum_{k=1}^{K}\sum_{x_l\in C_k} \phi(m_k)\\
&=\sum_{x_l^{(b)}\in\mathcal{X}^{(b)}} \phi(x_l^{(b)}) - n\sum_{k=1}^{K} p_{k+}\phi(m_k),
\end{aligned} \tag{15}$$

where (α) holds since m_k is the arithmetic mean for all $x_l^{(b)} \in C_k$, i.e., $\sum_{x_l\in C_k} x_l^{(b)} - m_k = \mathbf{0}$. Since both $\sum_{x_l^{(b)}\in\mathcal{X}^{(b)}} \phi(x_l^{(b)})$ and n are constants given Π, we have

$$\min_{\pi}\sum_{k=1}^{K}\sum_{x_l\in C_k} f(x_l^{(b)}, m_k) \Longleftrightarrow \max_{\pi}\sum_{k=1}^{K} p_{k+}\phi(m_k). \tag{16}$$

Now let us turn back to the proof of the sufficient condition. As g_Π is strictly increasing, we have

$$\max_{\pi}\sum_{i=1}^{r} w_i U(\pi,\pi_i) \Longleftrightarrow \max_{\pi}\sum_{k=1}^{K} p_{k+}\phi(m_k). \tag{17}$$

So we finally have Eq. (13), which indicates that U is a KCC utility function. The sufficient condition holds.

We then prove the necessary condition. Suppose the distance function f in Eq. (13) is derived from a differentiable convex function ϕ. By Eq. (13) and Eq. (16), we have Eq. (17).

Let $\Upsilon(\pi)$ denote $\sum_{i=1}^{r} w_i U(\pi,\pi_i)$ and $\Psi(\pi)$ denote $\sum_{k=1}^{K} p_{k+}\phi(m_k)$ for the convenience of description. Note that Eq. (17) holds for any feasible region F since Eq. (16) is derived from the equality rather than equivalence relationship. Therefore, for any two consensus partitionings π' and π'', if we let $F = \{\pi', \pi''\}$, we have

$$\Upsilon(\pi') > (=, \text{or} <)\ \Upsilon(\pi'') \iff \Psi(\pi') > (=, \text{or} <)\ \Psi(\pi''), \tag{18}$$

which indicates that Υ and Ψ have the same ranking over all the possible partitionings in the universal set $F^* = \{\pi | L_\pi(x_l^{(b)}) \in \{1,\cdots,K\}, 1 \le l \le n\}$. Define a mapping g_Π that maps $\Psi(\pi)$ to $\Upsilon(\pi)$, $\pi \in F^*$. According to Eq. (18), g_Π is a function, and $\forall\ x > x',\ g_\Pi(x) > g_\Pi(x')$. This implies that g_Π is strictly increasing. So the necessary condition holds, and the whole lemma thus follows. □

Remark 3. Compared with the KCC utility function in Definition 1, the value of Lemma 2 is to replace "⇔" by "=", which sheds light on deriving the detailed expression of KCC utility functions. We here use Π as the subscript for g, because it directly affects the ranking of π in F^* given by Υ or Ψ. That is, different mapping functions may exist for different settings of Π.

We then take a further step to analyze Eq. (14). Recall the contingency matrix in Table 2. Let $P_k^{(i)}$ denote $\langle p_{k1}^{(i)}/p_{k+}, \cdots, p_{kj}^{(i)}/p_{k+}, \cdots, p_{kK_i}^{(i)}/p_{k+}\rangle$. According to Lemma 1, $P_k^{(i)} \doteq m_{k,i}$, but $P_k^{(i)}$ is defined from a contingency matrix perspective. We then have an important theorem as follows:

Theorem 1: U is a KCC utility function, if and only if $\forall\ \Pi = \{\pi_1,\cdots,\pi_r\}$ and $K \ge 2$, there exists a set of continuously differentiable convex functions $\{\mu_1,\cdots,\mu_r\}$ such that

$$U(\pi,\pi_i) = \sum_{k=1}^{K} p_{k+}\mu_i(P_k^{(i)}), \forall\ i. \tag{19}$$

The convex function ϕ for the corresponding K-means clustering is given by

$$\phi(m_k) = \sum_{i=1}^{r} w_i\nu_i(m_{k,i}), \forall\ k, \tag{20}$$

where

$$\nu_i(x) = a\mu_i(x) + c_i, \forall\ i, a \in \mathbb{R}_{++}, c_i \in \mathbb{R}. \tag{21}$$

Proof: We first prove the sufficient condition. If we substitute $U(\pi,\pi_i)$ in Eq. (19) into the left-hand-side of Eq. (14), we have

$$\begin{aligned}
\sum_{i=1}^{r} w_i U(\pi,\pi_i) &= \sum_{k=1}^{K} p_{k+}\sum_{i=1}^{r} w_i\mu_i(P_k^{(i)})\\
&\overset{(\alpha)}{=} \frac{1}{a}\sum_{k=1}^{K} p_{k+}\sum_{i=1}^{r} w_i\nu_i(m_{k,i}) - \frac{1}{a}\sum_{i=1}^{r} w_i c_i\\
&\overset{(\beta)}{=} \frac{1}{a}\sum_{k=1}^{K} p_{k+}\phi(m_k) - \frac{1}{a}\sum_{i=1}^{r} w_i c_i,
\end{aligned} \tag{22}$$

where (α) holds due to Eq. (21), and (β) holds due to Eq. (20). Let $g_\Pi(x) = a_\Pi x + b_\Pi$, where $a_\Pi = \frac{1}{a}$ and $b_\Pi = -\frac{1}{a}\sum_{i=1}^{r} w_i c_i$. Apparently, g_Π is a strictly increasing function for $a > 0$. We then have $\sum_{i=1}^{r} w_i U(\pi,\pi_i) = g_\Pi(\sum_{k=1}^{K} p_{k+}\phi(m_k))$, which indicates that U is a KCC utility function . The sufficient condition thus holds. It remains to prove the necessary condition.

Recall Lemma 2. Due to the arbitrariness of Π, we can let $\Pi \doteq \Pi_i = \{\pi_i\}$ $(1 \le i \le r)$. Accordingly, m_k reduces to $m_{k,i}$ in Eq. (14), and $\phi(m_{k,i})$ reduces to $\phi_i(m_{k,i})$, i.e., the ϕ function defined only on the ith "block" of m_k without involving the weight w_i. Then by Eq. (14), we have

$$U(\pi,\pi_i) = g_{\Pi_i}(\sum_{k=1}^{K} p_{k+}\phi_i(m_{k,i})),\ 1 \le i \le r, \tag{23}$$

where g_{Π_i} is the mapping function when Π reduces to Π_i. By summing up $U(\pi,\pi_i)$ from $i = 1$ to r, we have

$$\sum_{i=1}^{r} w_i U(\pi,\pi_i) = \sum_{i=1}^{r} w_i\ g_{\Pi_i}(\sum_{k=1}^{K} p_{k+}\phi_i(m_{k,i})),$$

which, according to Eq. (14), indicates that

$$g_\Pi(\sum_{k=1}^{K} p_{k+}\phi(m_k)) = \sum_{i=1}^{r} w_i g_{\Pi_i}(\sum_{k=1}^{K} p_{k+}\phi_i(m_{k,i})). \tag{24}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

$[\pi, \Gamma, F] =$**KCC**(Π, w, μ, K)

Input: Π: the set of basic partitionings
w: the set of weights for basic partitionings
μ: the convex function defined on $P_k^{(i)}$
K: the number of clusters
Output: π: the consensus partitioning
Γ: the optimal consensus-function value

Procedure

1. Let $\nu \equiv \mu$, get ϕ by Eq. (20), and then f by Eq. (3);
2. Construct the binary data set $\mathcal{X}^{(b)}$ from Π;
3. Call K-means to cluster $\mathcal{X}^{(b)}$ into K clusters and get π;
4. Compute Γ by Eq. (1) and Eq. (19);
5. **return** π and Γ.

Fig. 1. KCC Algorithm

If we take the partial derivative with respect to $m_{k,ij}$ on both sides, we have

$$\underbrace{g'_{\Pi}(\sum_{k=1}^{K} p_{k+}\phi(m_k))}_{(\alpha)} \underbrace{\frac{\partial \phi(m_k)}{\partial m_{k,ij}}}_{(\beta)} = w_i \underbrace{g'_{\Pi_i}(\sum_{k=1}^{K} p_{k+}\phi_i(m_{k,i}))}_{(\gamma)} \underbrace{\frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}}}_{(\delta)}. \tag{25}$$

As (γ) and (δ) do not contain any weight parameters w_l, $1 \le l \le r$, the right-hand-side of Eq. (25) has one and only one weight parameter: w_i. This implies that (α) is a constant, otherwise the left-hand-side of Eq. (25) would contain multiple weight parameters other than w_i due to the existence of $\phi(m_k)$ in $g'_{\Pi,K}$. Analogously, since (β) does not contain all p_{k+}, $1 \le k \le K$, (γ) must also be a constant. These results imply that $g_{\Pi}(x)$ and $g_{\Pi_i}(x)$, $1 \le i \le r$, are all linear functions. Without loss of generality, we let

$$g_{\Pi}(x) = a_{\Pi}x + b_{\Pi},\ \forall\ a_{\Pi} \in \mathbb{R}_{++}, b_{\Pi} \in \mathbb{R},\ \text{and} \tag{26}$$

$$g_{\Pi_i}(x) = a_i x + b_i,\ \forall\ a_i \in \mathbb{R}_{++}, b_i \in \mathbb{R}. \tag{27}$$

As a result, Eq. (25) turns into

$$a_{\Pi}\frac{\partial \phi(m_k)}{\partial m_{k,ij}} = w_i a_i \frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}},\ \forall\ i, j. \tag{28}$$

$\partial\phi_i(m_{k,i})/\partial m_{k,ij}$ is the function of $\{m_{k,i1}, \cdots, m_{k,iK_i}\}$ only, which implies that $\partial\phi(m_k)/\partial m_{k,ij}$ is not the function of $m_{k,l}$, $\forall\ l \ne i$. As a result, given the arbitrariness of i, we have $\phi(m_k) = \varphi(\phi_1(m_{k,1}), \cdots, \phi_r(m_{k,r})))$, which indicates

$$\frac{\partial \phi(m_k)}{\partial m_{k,ij}} = \frac{\partial \varphi}{\partial \phi_i}\frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}}.$$

Accordingly, Eq. (28) turns into $\partial\varphi/\partial\phi_i = w_i a_i / a_{\Pi}$, so

$$\phi(m_k) = \sum_{i=1}^{r}(\frac{w_i a_i}{a_{\Pi}}\phi_i(m_{k,i}) + d_i), \forall\ d_i \in \mathbb{R}. \tag{29}$$

Let $\nu_i(x) = \frac{a_i}{a_{\Pi}}\phi_i(x) + \frac{d_i}{w_i}$, $1 \le i \le r$, Eq. (20) thus follows.

Moreover, according to Eq. (23) and Eq. (27), we have

$$U(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+}(a_i\phi_i(P_k^{(i)}) + b_i), \forall\ i. \tag{30}$$

Let $\mu_i(x) = a_i\phi_i(x) + b_i$, $1 \le i \le r$, Eq. (19) follows. Further let $a = 1/a_{\Pi}$ and $c_i = d_i/w_i - b_i/a_{\Pi}$, we also have Eq. (21). The necessary condition holds. We complete the proof. □

Remark 4. Theorem 1 gives the necessary and sufficient condition for being a KCC utility function. That is, a KCC utility function must be a weighted average of a set of convex functions defined on $P_k^{(i)}$, $1 \le i \le r$, respectively. Theorem 1 also indicates the way to conduct KCC. That is, we first design or designate a set of convex functions μ_i defined on $P_k^{(i)}$, $1 \le i \le r$, from which the utility function as well as the consensus function can be derived by Eq. (19) and Eq. (1), respectively; then after setting a and c_i, $1 \le i \le r$, in Eq. (21), we can determine the corresponding ϕ for K-means clustering by Eq. (20), which is further used to derive the point-to-centroid distance f using Eq. (3); the K-means clustering is finally employed to find the consensus partitioning π.

Some practical points regarding to μ_i and ν_i, $1 \le i \le r$, are noteworthy here. First, according to Eq. (3), different settings of a and c_i in Eq. (21) will lead to different distances f but the same K-means clustering in Eq. (2), given that μ_i, $1 \le i \le r$, are invariant. As a result, we can simply let $\nu_i \equiv \mu_i$ by having $a = 1$ and $c_i = 0$ in practice, which are actually the default settings in our work. Second, it is more convenient to unify μ_i, $1 \le i \le r$, to a same convex function μ in practice, although they are treated separately above to keep the generality of the theorem. This also becomes the default setting in our work. Third, it is easy to show that the linear extension of μ to $\mu'(x) = c\mu(x) + d$ ($c \in \mathbb{R}_{++}, d \in \mathbb{R}$) will change the utility function in Eq. (19) proportionally but again leads to the same K-means clustering and thus the same consensus partitioning. Therefore, there is a many-to-one correspondence from utility function to K-means clustering, and we can use the simplest form of μ without loss of accuracy. We summarize the KCC procedure using the default settings ($a = 1, c_i = 0, \mu \equiv \mu_i \equiv \nu_i, \forall i$) in Fig. 1.

The computational complexity of KCC on clustering $\mathcal{X}^{(b)}$ is $O(InrK)$, where I is the number of iterations for the convergence of K-means clustering, and n is the number of data instances. Note that although $\mathcal{X}^{(b)}$ has $\sum_{i=1}^{r} K_i$ dimensions in total, only the r non-zero dimensions will be recorded for each data point, and thus we have r rather than $\sum_{i=1}^{r} K_i$ in the complexity formulation. Since I is usually smaller than 20, r is often set to 100 for robustness, and K is often very small, KCC on $\mathcal{X}^{(b)}$ is roughly linear to n, which is the key factor for the high efficiency of KCC. If we further take the preparation of basic partitionings into consideration, there will be additional runtime cost of $O(r \times IndK)$, where d is the number of data dimensions, and r indicates the number of repetitions of K-means clustering for r basic partitionings. As a result, the overall complexity of KCC will be $O((d+1)InrK)$, roughly r times of the runtime cost of K-means without ensembles. If we make use of parallel computing, the runtime cost for basic partitionings can be lowered to $O(rIndK/w)$, and the overall

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

TABLE 3
Sample KCC Utility Functions

	$\mu(m_{k,i})$	$U_\mu(\pi, \pi_i)$	$f(x_l^{(b)}, m_k)$
U_c	$\|m_{k,i}\|_2^2 - \|P^{(i)}\|_2^2$	$\sum_{k=1}^K p_{k+}\|P_k^{(i)}\|_2^2 - \|P^{(i)}\|_2^2$	$\sum_{i=1}^r w_i \|x_{l,i}^{(b)} - m_{k,i}\|_2^2$
U_H	$(-H(m_{k,i})) - (-H(P^{(i)}))$	$\sum_{k=1}^K p_{k+}(-H(P_k^{(i)})) - (-H(P^{(i)}))$	$\sum_{i=1}^r w_i D(x_{l,i}^{(b)} \| m_{k,i})$
$U_{\cos}$	$\|m_{k,i}\|_2 - \|P^{(i)}\|_2$	$\sum_{k=1}^K p_{k+}\|P_k^{(i)}\|_2 - \|P^{(i)}\|_2$	$\sum_{i=1}^r w_i(1 - \cos(x_{l,i}^{(b)}, m_{k,i}))$
U_{L_p}	$\|m_{k,i}\|_p - \|P^{(i)}\|_p$	$\sum_{k=1}^K p_{k+}\|P_k^{(i)}\|_p - \|P^{(i)}\|_p$	$\sum_{i=1}^r w_i(1 - \frac{\sum_{j=1}^{K_i} x_{l,ij}^{(b)}(m_{k,ij})^{p-1}}{\|m_{k,i}\|_p^{p-1}})$

Note: D – KL-divergence; H – Shannon entropy; L_p – L_p norm.

complexity of KCC will be $O((d/w+1)InrK)$, where w is the number of parallel workers.

Example. Hereinafter, we denote the KCC utility function derived by μ in Eq. (19) as U_μ for convenience. Table 3 shows some examples of KCC utility functions derived from various μ, and their corresponding point-to-centroid distances f, where $P^{(i)} \doteq \langle p_{+1}^{(i)}, \cdots, p_{+j}^{(i)}, \cdots, p_{+K_i}^{(i)} \rangle$, $1 \le i \le r$. Note that U_c is the well-known Category Utility Function [22], but the other three U_μ have hardly been mentioned in the literature. For instance, U_H is derived from the Shannon entropy H, and the corresponding distance is the well-known KL-divergence D, which has been widely used for information-theoretic text clustering [33]. Since the binary data set $\mathcal{X}^{(b)}$ is also in high dimensionality as the text data, we could expect the excellent performance of U_H in KCC. U_{L_p} is derived from the L_p-norm, with $U_{\cos}$ being one of the special case at $p = 2$. We highlight $U_{\cos}$ here because its corresponding distance is the reversed cosine similarity, another widely adopted distance in text clustering [33]. Therefore, it is interesting to compare the performances between U_H and $U_{\cos}$ in the context of consensus clustering.

4.3 Two Forms of KCC Utility Functions

Theorem 1 indicates how to derive a KCC utility function from a convex function μ. However, it does not guarantee that the obtained KCC utility function is explainable. Therefore, we here introduce two special forms of KCC utility functions which are meaningful to some extent.

4.3.1 Standard Form of KCC Utility Functions

Suppose we have a utility function U_μ derived from μ. Recall $P^{(i)} = \langle p_{+1}^{(i)}, \cdots, p_{+K_i}^{(i)} \rangle$, $1 \le i \le r$, which are actually constant vectors given the basic partitionings Π. If we let

$$\mu_s(P_k^{(i)}) = \mu(P_k^{(i)}) - \mu(P^{(i)}), \tag{31}$$

then by Eq. (19), we can obtain a new utility function as follows:

$$U_{\mu_s}(\pi, \pi_i) = U_\mu(\pi, \pi_i) - \mu(P^{(i)}). \tag{32}$$

As $\mu(P^{(i)})$ is a constant given π_i, μ_s and μ will lead to a same corresponding point-to-centroid distance f, and thus a same consensus partitioning π. The advantage of using μ_s rather than μ roots in the following proposition:

Proposition 1: $U_{\mu_s} \ge 0$.

Proposition 1 ensures the non-negativity of U_{μ_s}. Indeed, U_{μ_s} can be viewed as the *utility gain* from a consensus clustering, by calibrating U_μ to the benchmark: $\mu(P^{(i)})$. Here, we define the utility gain as the *standard form* of a KCC utility function. Accordingly, all the utility functions listed in Table 3 are in the standard form. It is also noteworthy that the standard form has invariance; that is, if we let $\mu_{ss}(P_k^{(i)}) = \mu_s(P_k^{(i)}) - \mu_s(P^{(i)})$, we have $\mu_{ss} \equiv \mu_s$ and $U_{\mu_{ss}} \equiv U_{\mu_s}$. Therefore, a convex function μ can derive one and only one KCC utility function in the standard form.

4.3.2 Normalized Form of KCC Utility Functions

It is natural to take a further step from the standard form U_{μ_s} to the *normalized form* U_{μ_n}. Let

$$\mu_n(P_k^{(i)}) = \frac{\mu_s(P_k^{(i)})}{|\mu(P^{(i)})|} = \frac{\mu(P_k^{(i)}) - \mu(P^{(i)})}{|\mu(P^{(i)})|}. \tag{33}$$

Since $\mu(P^{(i)})$ is a constant given π_i, it is easy to know that μ_n is also a convex function, from which a KCC utility function U_{μ_n} can be derived as follows:

$$U_{\mu_n}(\pi, \pi_i) = \frac{U_{\mu_s}(\pi, \pi_i)}{|\mu(P^{(i)})|} = \frac{U_\mu(\pi, \pi_i) - \mu(P^{(i)})}{|\mu(P^{(i)})|}. \tag{34}$$

From Eq. (34), $U_{\mu_n} \ge 0$, which can be viewed as the *utility gain ratio* to the constant $|\mu(P^{(i)})|$. Note that the ϕ functions corresponding to U_{μ_n} and U_{μ_s}, respectively, are different due to the introduction of $|\mu(P^{(i)})|$ in Eq. (33). As a result, the consensus partitionings by KCC are also different for U_{μ_n} and U_{μ_s}. Nevertheless, the KCC procedure for U_{μ_n} will be exactly the same as the procedure for U_{μ_s} or U_μ, if we let $w_i = w_i/|\mu(P^{(i)})|$, $1 \le i \le r$, in Eq. (20). Finally, it is noteworthy that the normalized form U_{μ_n} also has the invariance property.

In summary, given a convex function μ, we can derive a KCC utility function U_μ, as well as its standard form U_{μ_s} and normalized form U_{μ_n}. While U_{μ_s} leads to a same consensus partitioning as U_μ, U_{μ_n} results in a different one. Given clear physical meanings, the standard form and normalized form will be adopted as two major forms of KCC utility functions in the experimental section below.

5 Handling Incomplete Basic Partitionings

Here, we introduce how to exploit K-means-based consensus clustering for handling incomplete basic partitionings (IBPs). We begin by formulating the problem as follows.

5.1 Problem Description

Assume a basic partitioning π_i is obtained by clustering a data subset $\mathcal{X}_i \subseteq \mathcal{X}$, $1 \le i \le r$, with the constraint that $\bigcup_{i=1}^r \mathcal{X}_i = \mathcal{X}$. Here the problem is, given r IBPs in $\Pi = \{\pi_1, \cdots, \pi_r\}$, how to cluster $\mathcal{X}$ into K crisp clusters using KCC?

The value of solving this problem lies in two folds. First, from the theoretical perspective, IBPs will generate incomplete

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

TABLE 4
Adjusted Contingency Matrix

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

binary data set $\mathcal{X}^{(b)}$ with missing values, and how to deal with missing values has long been the challenging problem in the statistical field. Moreover, how to guarantee the convergence of K-means on incomplete data is also very interesting in theory. Second, from the practical perspective, it is not unusual in real-world applications that part of data instances are unavailable in a basic partitioning due to a distributed system or the delay of data arrival. Knowledge reuse is also a source for IBPs, since the knowledge of various basic partitionings may be gathered from different research or application tasks [1].

To meet this challenge, in what follows, we propose a new solution to K-means-based consensus clustering on IBPs.

5.2 Solution

We first adjust the way for utility computation on IBPs. We still have maximizing Eq. (1) as the objective of consensus clustering, but the contingency matrix for $U(\pi, \pi_i)$ computation is modified to the one in Table 4. In the table, $n_{k+}^{(i)}$ is the number of instances assigned from $\mathcal{X}_i$ to cluster C_k, $1 \leq k \leq K$, and $n^{(i)}$ is the total number of instances in $\mathcal{X}_i$, i.e., $n^{(i)} = |\mathcal{X}_i|$, $1 \leq i \leq r$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n^{(i)}$, $p_{k+}^{(i)} = n_{k+}^{(i)}/n^{(i)}$, $p_{+j}^{(i)} = n_{+j}^{(i)}/n^{(i)}$, and $p^{(i)} = n^{(i)}/n$.

We then adjust K-means clustering to handle the incomplete binary data set $\mathcal{X}^{(b)}$. Let the distance f be the sum of the point-to-centroid distances in different basic partitionings, i.e.,

$$f(x_l^{(b)}, m_k) = \sum_{i=1}^{r} I(x_l \in \mathcal{X}_i) f_i(x_{l,i}^{(b)}, m_{k,i}), \tag{35}$$

where f_i is f on the ith "block" of $\mathcal{X}^{(b)}$. $I(x_l \in \mathcal{X}_i) = 1$ if $x_l \in \mathcal{X}_i$, and 0 otherwise.

We then obtain a new objective function for K-means clustering as follows:

$$F = \sum_{k=1}^{K} \sum_{x_l \in C_k} f(x_l^{(b)}, m_k) \tag{36}$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_l \in C_k \bigcap \mathcal{X}_i} f_i(x_{l,i}^{(b)}, m_{k,i}), \tag{37}$$

where the centroid $m_{k,i} = \langle m_{k,i1}, \cdots, m_{k,iK_i} \rangle$, with

$$m_{k,ij} = \frac{\sum_{x_l \in C_k \bigcap \mathcal{X}_i} x_{l,ij}^{(b)}}{|C_k \bigcap \mathcal{X}_i|} = \frac{n_{kj}^{(i)}}{n_{k+}^{(i)}} = \frac{p_{kj}^{(i)}}{p_{k+}^{(i)}}, \ \forall\ k, i, j. \tag{38}$$

Note that Eq. (37) and Eq. (38) indicate the specialty of K-means clustering on incomplete data; that is, the centroid of cluster C_k $(1 \leq k \leq K)$ has no longer existed physically, but rather serves as a "virtual" one just for the computational purpose. It is replaced by the loose combination of r sub-centroids computed separately on r IBPs.

For the convergence of the adjusted K-means, we have:

Theorem 2: For the objective function given in Eq. (36), K-means clustering using f in Eq. (35) as the distance function and using $m_{k,i}, \forall\ k, i$, in Eq. (38) as the centroid, is guaranteed to converge in finite two-phase iterations.

The proof is omitted for space concern. Let $P_k^{(i)} \doteq \langle p_{k1}^{(i)}/p_{k+}^{(i)}, \cdots, p_{kK_i}^{(i)}/p_{k+}^{(i)} \rangle = m_{k,i}$. We then extend KCC to the IBP case as follows:

Theorem 3: U is a KCC utility function, if and only if $\forall\ \Pi = \{\pi_1, \cdots, \pi_r\}$ and $K \geq 2$, there exists a set of continuously differentiable convex functions $\{\mu_1, \cdots, \mu_r\}$ such that

$$U(\pi, \pi_i) = p^{(i)} \sum_{k=1}^{K} p_{k+}^{(i)} \mu_i(P_k^{(i)}), \forall\ i. \tag{39}$$

The convex function ϕ_i $(1 \leq i \leq r)$, for the corresponding K-means clustering is given by

$$\phi_i(m_{k,i}) = w_i \nu_i(m_{k,i}), \forall\ k, \tag{40}$$

where

$$\nu_i(x) = a\mu_i(x) + c_i, \forall\ i, a \in \mathbb{R}_{++}, c_i \in \mathbb{R}. \tag{41}$$

Remark 5. The proof is similar to the one for Theorem 1, so we omit it here. Eq. (39) is very similar to Eq. (19) except for the appearance of the parameter $p^{(i)}$ $(1 \leq i \leq r)$. This parameter implies that the basic partitioning on a larger data subset will have more impact on the consensus clustering, which is considered reasonable. Also note that when the incomplete data case reduces to the normal case, Eq. (39) reduces to Eq. (19) naturally. This implies that the incomplete data case is a more general scenario in essence. Finally, the KCC procedure for incomplete data is very similar to the algorithm in Fig. 1, although the computation of some variables is slightly different.

6 Experimental Results

In this section, we present experimental results of K-means-based consensus clustering on real-world data sets. Specifically, we first demonstrate the execution efficiency and clustering quality of KCC, and then explore the major factors that affect the performance of KCC. We finally showcase the effectiveness of KCC on handling incomplete basic partitionings.

6.1 Experimental Setup

Experimental data. We used a testbed consisting of a number of real-world data sets obtained from both the UCI and TREC repositories. Table 5 shows some important characteristics of these data sets, where "CV" is the Coefficient of Variation statistic [34] that characterizes the degree of class imbalance. A higher CV value indicates a more severe class imbalance.

Validation measure. Since the class labels were provided for each data set, we adopted the *normalized Rand index* (R_n), a long-standing external measure for objective cluster validation.

TABLE 5
Some Characteristics of Real-World Data Sets

Data Sets	Source	#Objects	#Attributes	#Classes	CV
breast_w	UCI	699	9	2	0.439
ecoli†	UCI	332	7	6	0.899
iris	UCI	150	4	3	0.000
pendigits	UCI	10992	16	10	0.042
satimage	UCI	4435	36	6	0.425
dermatology	UCI	358	33	6	0.509
wine‡	UCI	178	13	3	0.194
mm	TREC	2521	126373	2	0.143
reviews	TREC	4069	126373	5	0.640
la12	TREC	6279	31472	6	0.503
sports	TREC	8580	126373	7	1.022

†: two clusters containing only two objects were deleted as noise.
‡: the values of the last attribute were normalized by a scaling factor 100.

In the literature, it has been recognized that R_n is particularly suitable for K-means clustering evaluation [35]. The value of R_n typically varies in [0,1] (might be negative for extremely poor results), and a larger value indicates a higher clustering quality. More details of R_n can be found in Ref. [29].

Clustering tools. Three types of consensus clustering methods, namely the K-means-based algorithm (KCC), the graph partitioning algorithm (GP), and the hierarchical algorithm (HCC), were employed for the comparison purpose. GP is actually a general concept of three benchmark algorithms: CSPA, HGPA and MCLA [1], which were coded in MATLAB and provided by Strehl[1]. HCC is essentially an agglomerative hierarchical clustering algorithm based on the so-called co-association matrix. It was implemented by ourselves in MATLAB following the algorithmic description in Ref. [4]. We also implemented KCC in MATLAB, which includes ten utility functions, namely U_c, U_H, $U_{\cos}$, U_{L_5} and U_{L_8}, and their corresponding normalized versions (denoted as NU_x).

To generate basic partitionings (BPs), we used the *kmeans* function of MATLAB with squared Euclidean distance for UCI data sets and with cosine similarity for text data sets. Two strategies, i.e., Random Parameter Selection (RPS) and Random Feature Selection (RFS) proposed in Ref. [1], were used to generate BPs. For RPS, we randomized the number of clusters within an interval for each basic clustering. For RFS, we randomly selected partial features for each basic clustering.

Unless otherwise specified, the default settings for the experiments are as follows. The number of clusters for KCC is set to the number of true clusters (namely the clusters indicated by the known class labels). For each data set, 100 BPs are typically generated for consensus clustering (namely $r = 100$), and the weights of these BPs are exactly the same, i.e., $w_i = w_j,\ \forall\, i, j$. RPS is the default generation strategy for BPs, with the number of clusters for *kmeans* being randomized within $[K, \sqrt{n}]$, where K is the number of true clusters and n is the total number of data objects. When RFS is used instead, we typically select two features randomly for each BP, and set the number of clusters to K for *kmeans*. For each Π, KCC and GP are run ten times to obtain the average result, whereas HCC is run only once due to its deterministic nature. In each run of KCC, K-means subroutine is called ten times to return the best result. Similarly, in each run of GP, CSPA, HGPA and MCLA are called simultaneously to find the best result.

All experiments were run on a Windows 7 platform of SP2 32-bit edition. The PC has an Intel Core i7-2620M 2.7GHz*2 CPU with a 4MB cache, and a 4GB DDR3 664.5MHz RAM.

1. Available at: http://www.strehl.com.

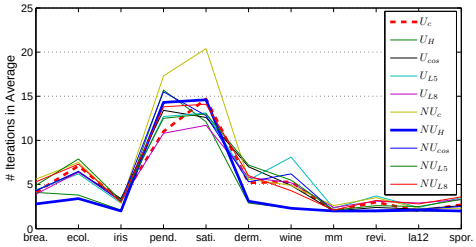
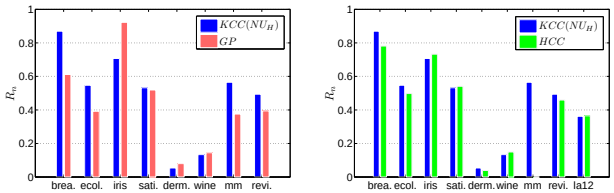


Fig. 2. Illustration of KCC Convergence



(a) KCC vs. GP (b) KCC vs. HCC

Fig. 3. Comparison of Clustering Quality

6.2 Clustering Efficiency of KCC

The primary concern about a consensus clustering method is the efficiency issue. So we first examine the convergence of KCC, and then its efficiency compared with other methods.

We generated one set of basic partitionings for each data set, and then ran KCC on each Π using different utility functions. The average numbers of iterations for convergence are shown in Fig. 2. As can be seen, KCC generally converges within 15 iterations regardless of utility functions used, with the only exceptions on *pendigits* and *satimage* data sets. Among the ten utility functions, NU_H exhibits the fastest speed of convergence on nearly all data sets except *pendigits* and *satimage*, as indicated by the blue solid-line in bold.

We then compare KCC with GP and HCC in terms of execution efficiency. Note that the three methods were run with default settings, and U_c was selected for KCC since it showed a moderate convergence speed in Fig. 2 (as indicated by the red dashed-line in bold). Table 6 shows the runtime comparison of the three methods, where the fastest one is in bold for each data set. As can be seen, although KCC was run $10 \times 10 = 100$ times for each data set, it is still the fastest one on nine out of eleven data sets. For the large-scale data sets, such as *satimage* and *reviews*, the advantage of KCC is particularly evident. HCC seems more suitable for small data sets, such as *iris* and *wine*, and becomes struggled for large-scale data sets for its n-squared complexity. GP consumes much less time than HCC on large-scale data sets, but it suffers from the high space complexity — that is why it failed to deliver results for three data sets in Table 6 (marked as "N/A"), even one more time than HCC. Note that the execution time for generating basic partitionings was listed in the bottom of the table.

To sum up, KCC shows significantly higher clustering efficiency than the other two popular methods. This is particularly important for real-world applications with large-scale data sets.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

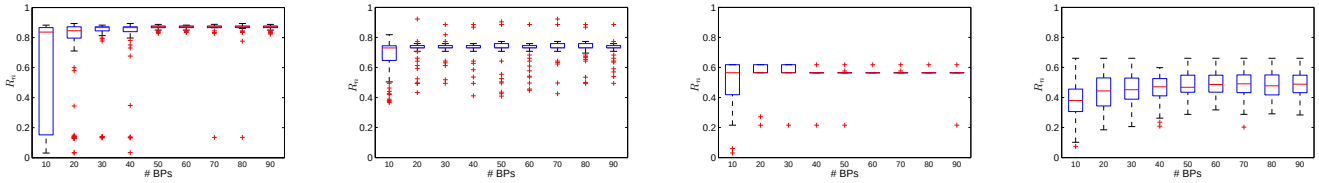
TABLE 6
Comparison of Execution Time (in seconds)

	brea.	*ecol.*	*iris*	*pend.*	*sati.*	*derm.*	*wine*	*mm*	*revi.*	*la12*	*spor.*
$KCC(U_c)$	**1.95**	**1.40**	0.33	**81.19**	**32.47**	**1.26**	0.56	**2.78**	**4.44**	**8.15**	**11.33**
GP	8.80	6.79	4.08	N/A	54.39	6.39	3.92	15.40	32.35	N/A	N/A
HCC	18.85	2.33	**0.18**	N/A	2979.48	2.78	**0.28**	535.63	2154.45	6486.87	N/A
K-means (100 times)	6.75	1.67	0.62	55.78	27.84	0.69	2.45	643.95	1455.98	1405.90	3873.26

(1) **N/A:** failure due to the out-of-memory error; (2) **K-means (100 times):** runtime for basic-partitioning generation.

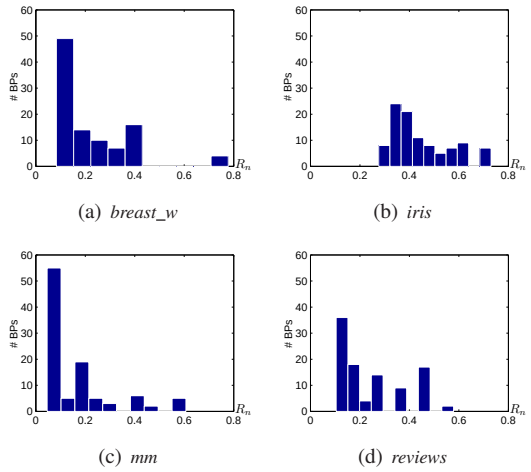
TABLE 7
KCC Clustering Results (by R_n)

	U_c	U_H	U_{cos}	U_{L5}	U_{L8}	NU_c	NU_H	NU_{cos}	NU_{L5}	NU_{L8}
breast_w	0.0556	0.8673	0.1111	0.1212	0.1333	0.0380	**0.8694**	0.1173	0.1329	0.1126
ecoli	0.5065	0.4296	0.4359	0.4393	0.4284	0.5012	**0.5470**	0.4179	0.4174	0.4281
iris	0.7352	0.7338	0.7352	0.7352	**0.7455**	0.7325	0.7069	**0.7455**	**0.7455**	0.7352
pendigits	0.5347	0.5596	**0.5814**	0.5692	0.5527	0.5060	0.5652	0.5789	0.5684	0.5639
satimage	0.4501	0.4743	0.5322	0.4738	0.4834	0.3349	**0.5323**	0.5318	0.4691	0.4797
dermatology	0.0352	**0.0661**	0.0421	0.0274	0.0223	0.0386	0.0537	0.0490	0.0259	0.0309
wine	0.1448	**0.1476**	0.1448	0.1397	0.1379	0.1448	0.1336	0.1449	0.1447	0.1379
mm	0.5450	0.5702	0.5674	0.5923	**0.6184**	0.4841	0.5648	0.6023	0.6131	**0.6184**
reviews	0.3767	0.4628	0.4588	0.4912	0.4648	0.3257	0.4938	0.5200	0.4817	**0.5323**
la12	0.3455	**0.3878**	0.3647	0.3185	0.3848	0.3340	0.3619	0.3323	0.3451	0.4135
sports	0.3211	0.4039	0.3429	0.3720	0.3312	0.2787	**0.4093**	0.3407	0.3053	0.3130
score	8.4645	10.337	9.0279	8.7187	8.6803	7.8909	**10.355**	9.2036	8.6247	8.9366



(a) *breast_w* (b) *iris* (c) *mm* (d) *reviews*

Fig. 4. Impact of the Number of Basic Partitionings to KCC



(a) *breast_w* (b) *iris* (c) *mm* (d) *reviews*

Fig. 5. Distribution of Clustering Quality of Basic Partitionings

6.3 Clustering Quality of KCC

Here, we demonstrate the cluster validity of KCC by comparing it with the well-known GP and HCC algorithms. The normalized Rand index R_n was adopted as the cluster evaluation measure.

We took RPS as the strategy for basic partitioning generation, and employed KCC, GP and HCC with default settings for all the data sets. The clustering results of KCC with different utility functions are shown in Table 7. As can be seen, seven out of ten utility functions performed the best on at least one data set, as highlighted in bold. This implies that the diversity of utility functions is very important to the success of consensus clustering. KCC thus achieves an edge by providing a flexible framework that can incorporate various utility functions for different applications. For instance, for the *breast_w* data set, using U_H or NU_H can obtain an excellent result with $R_n > 0.85$; but the clustering quality will drop sharply to $R_n < 0.15$ if other functions are used instead.

In real-world applications, however, it is hard to know which utility function is the best to a given data set without providing external information. One solution is to rank the utility functions empirically on a testbed, and then set the one with the robust performance as the default choice. To this end, We score a utility function U_i by $score(U_i) = \sum_j \frac{R_n(U_i,D_j)}{\max_i R_n(U_i,D_j)}$, where $R_n(U_i, D_j)$ is the R_n score of the clustering result generated by applying U_i on data set D_j. The row "score" at the bottom of Table 7 shows the final scores of all the utility functions. As can be seen, NU_H won the highest score, closely followed by U_H, and then $NU_{\cos}$ and $U_{\cos}$. Since NU_H also showed fast convergence in the previous section, we hereby take it as the default choice for KCC. It is also interesting to note that though being the first utility function proposed in the literature, U_c and its normalized version generally perform the worst among all the listed utility functions.

We then compare the clustering qualities between KCC and the other two methods, where NU_H was selected for KCC. Fig. 3 shows the comparative results. Note that in the left (right) sub-graph we omitted three (two) data sets, on which GP (HCC) failed due to the out-of-memory error. As can be seen, compared with GP and HCC, KCC generally shows comparable clustering performance. Indeed, KCC outperformed GP on five out of eight data sets, and beat HCC on five out of nine data sets. Another two observations are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering

also noteworthy. First, although HCC seems closer to KCC in terms of clustering quality, its robustness is subject to doubt — it performed extremely poorly (with a near-to-zero R_n value) on *mm*. Second, the *wine* and *dermatology* data sets are really big challenges to consensus clustering — the clustering quality measured by R_n is always below 0.2 no matter what method is used. We will show how to handle this below.

In summary, KCC is competitive to GP and HCC in terms of clustering quality. Among the ten utility functions, NU_H is more robust and thus becomes the default choice of KCC.

6.4 Exploration of Impact Factors

In this section, we explore the factors that might affect the clustering performance of KCC. We are concerned with the following characteristics of basic partitionings (BPs) in Π: the number of BPs (r), the quality of BPs, the diversity of BPs, and the generation strategy for BPs. Four data sets, i.e., *breast_w*, *iris*, *mm* and *reviews*, were used here for illustration.

6.4.1 I: Number of Basic Partitionings

To test the impact of the number of BPs, we did random sampling on Π (containing 100 BPs for each data set), and generated the subset Π^r, with $r = 10, 20, \cdots, 90$. For each r, we repeated sampling 100 times, and ran KCC on each sample for clustering results, as shown by the boxplot in Fig. 4.

As can be seen from Fig. 4, the volatility of the clustering quality tends to be reduced with the increase of r. When $r \geq 50$, the volatility seems to be fixed to a very small interval. This implies that $r = 50$ might be a rough critical point for obtaining robust KCC results in real-world applications. To validate this, we further enlarged the size of the complete set Π to 500, and iterated the above experiments by setting $r = 10, 20, \cdots, 490$, respectively. Again we found that the clustering quality of KCC became stable when $r \geq 50$. It is worth noting that this is merely an empirical estimation; the critical point might be raised as the scale of data sets (i.e., n) gets significantly larger. Nevertheless, it is no doubt that increasing the number of BPs can effectively suppress the volatility of KCC results.

6.4.2 II&III: Quality and Diversity of Basic Partitionings

In the field of supervised learning, researchers have long recognized that both the quality and diversity of single classifiers are crucial to the success of an ensemble classifier. Analogously, one may expect that the quality and diversity of basic partitionings might affect the performance of consensus clustering. While there have been some initial studies along this line, few research has clearly justified the impact of these two factors based on real-world data experiments. Hardly can we find any research that addressed how they interact with each other in consensus clustering. These indeed motivated our experiments below.

Fig. 5 depicts the quality distribution of basic partitionings of each data set. As can be seen, the distribution generally has a long right tail, which indicates that there is only a very small portion of BPs that are in relatively high quality. For example, *breast_w* has four BPs with R_n values over 0.7, but the rests are all below 0.4 and lead to an average: $\overline{R}_n = 0.2240$. Fig. 6 then illustrates the pair-wise similarity in R_n between any two BPs. Intuitively, a more diversified Π will correspond to a darker similarity matrix, and vice versa. In this sense, the BPs of *breast_w* and *iris* are better in diversity than the BPs of *mm* and *reviews*. Note that the BPs in each subplot were sorted in the increasing order of R_n.

Based on the above observations, we have the following conjecture: (1) Quality factor: The clustering quality of KCC is largely determined by the small number of BPs in relatively high quality (denoted as HQBP); (2) Diversity factor: The diversity of BPs will become the dominant factor when HQBPs are unavailable. We adopted a *stepwise deletion* strategy to verify the conjecture. That is, for the set of BPs of each data set, we first sorted BPs in the decreasing order of R_n, and then removed BPs gradually from top to bottom to observe the change of clustering quality of KCC. The red solid-lines in Fig. 7 exhibit the results. For the purpose of comparison, we also sorted BPs in the increasing order of R_n and repeated the stepwise deletion process. The results are represented by the blue dashed-lines in Fig. 7.

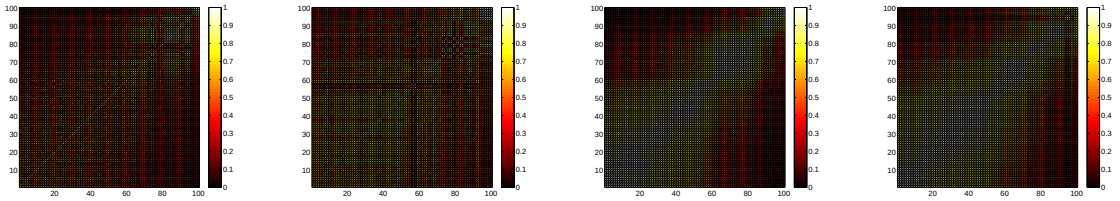
As can be seen from the red solid-lines in Fig. 7, the KCC quality suffers a sharp drop after removing the first few HQBPs from each data set. In particular, for the three data sets showing more significant long tails in Fig. 5, i.e., *breast_w*, *mm* and *reviews*, the quality deterioration is more evident. This implies that it is the small portion of HQBPs rather than the complete set of BPs that determines the quality of KCC. We can verify this point by further watching the variation of the blue dashed-lines, where the removal of BPs in relatively low quality (denoted as LQBP) shows hardly any influence to the KCC quality. Since the removal of LQBPs also means the shrinkage of diversity, we can understand that the quality factor represented by the few HQBPs is actually more important than the diversity factor. Therefore, our first conjecture holds. This result also indicates that KCC is capable of taking advantage of a few HQBPs to deliver satisfactory results, even if the whole set of BPs is generally in poor quality.

We then explore the diversity factor by taking a closer look at Fig. 7. It is interesting to see that the qualityof *breast_w* drops after removing roughly the first twenty HQBPs, among which only four HQBPs have $R_n > 0.7$ (as indicated by Fig. 5). This implies that it is the power of the diversity factor that keeps the KCC quality staying in a certain level until too many HQBPs are gone. Furthermore, it is noteworthy from Fig. 7 that *mm* and *reviews* experience quality drops earlier than *breast_w* and *iris*. To understand this, let us recall Fig. 6, where the bottom-left areas in much lighter colors indicate that *mm* and *reviews* have much poorer diversity than *breast_w* and *iris* on LQBPs. This further illustrates the existence and the importance of the diversity factor, especially when HQBPs are hard to attain.

Fig. 7 illustrates the impact of the diversity factor but in an indirect manner. We here go a further step to measure this factor directly. An index Div is first introduced to measure the diversity of basic partitionings. That is,

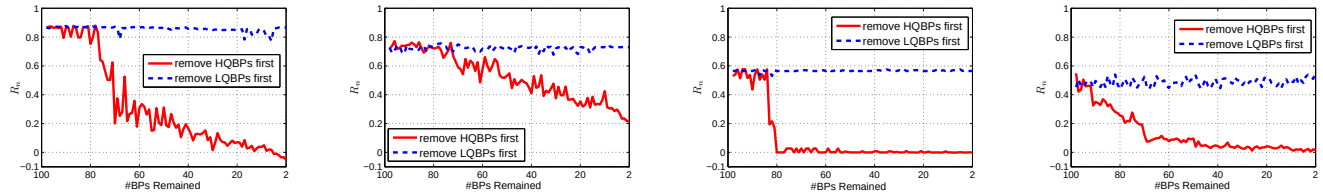
$$Div = 1 - \frac{1}{r}\left(\sum_{i=1}^{r}\sum_{j=1}^{r} R_n(\pi_i, \pi_j)^2\right)^{1/2}, \tag{42}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering



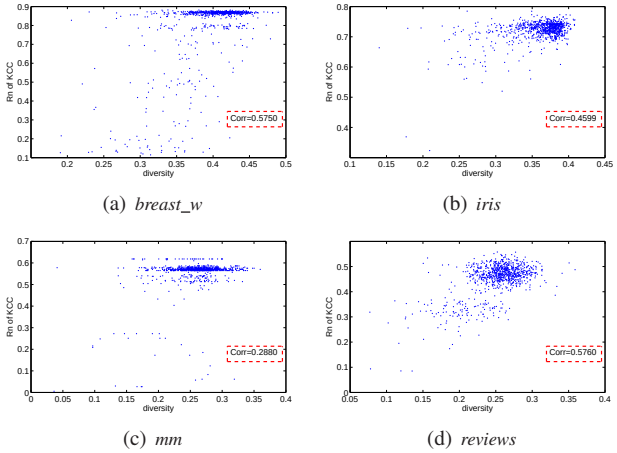
(a) *breast_w* (b) *iris* (c) *mm* (d) *reviews*

Fig. 6. Sorted Pair-wise Similarity Matrix of Basic Partitionings



(a) *breast_w* (b) *iris* (c) *mm* (d) *reviews*

Fig. 7. Performance of KCC Based on Stepwise Deletion Strategy



(a) *breast_w* (b) *iris* (c) *mm* (d) *reviews*

Fig. 8. Impact of Diversity

where $R_n(\pi_i, \pi_j)$ is the normalized Rand index between π_i and π_j. A larger Div indicates a better diversity. We then unveil the direct impact of the diversity factor to the consensus clustering quality. Two data sets, namely *breast_w* and *reviews*, were employed for this experiment. For each data set, we change the number of basic partitionings r from 10 to 90, and with each r we repeat the basic-partitioning generation 100 times. As a result, we finally obtain 900 sets of experimental results for each data set, upon which the relationship between Div and R_n is found, as shown in Fig. 8. As can be seen, while the correlation coefficient ($Corr$) values are not very large, the diversity of the basic partitionings generally shows positive correlation with the quality of the consensus clustering.

In summary, the quality and diversity of basic partitionings are both critical to the success of KCC. As the primary factor, the quality level usually depends on a few BPs in relatively high quality. The diversity will become a dominant factor instead when HQBPs are not available.

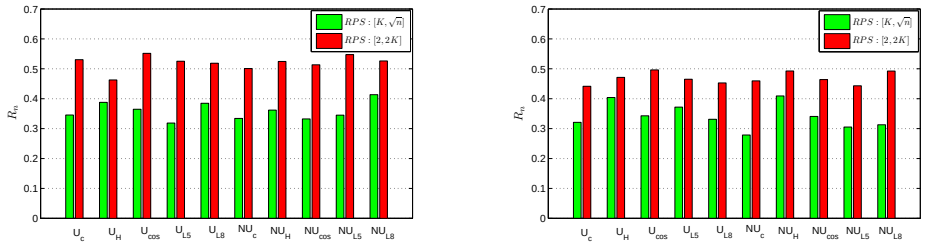
6.4.3 IV: The Generation Strategy of Basic Partitionings

So far we relied solely on RPS to generate basic partitionings, with the number of clusters K_i $(1 \leq i \leq r)$ varied in $[K, \sqrt{n}]$, where K is the number of true clusters and n is the number of data objects. This led to some poor clustering results, such as on UCI data sets *wine* and *dermatology* with $R_n < 0.15$ and on text data sets *la12* and *sports* with $R_n \approx 0.4$ (as shown by Table 7 and Fig. 3). Here we demonstrate how to use other generation strategies to improve clustering quality.

We first consider data sets *la12* and *sports*. These are two text data sets with relatively large scales, which means the interval $[K, \sqrt{n}]$ might be too large to generate good-enough BPs. To address this, we still use RPS, but narrow the interval of K_i to $[2, 2K]$. Fig. 9 shows the comparative result: The clustering performances of KCC were improved substantially after the interval adjustment. This clearly demonstrates that KCC can benefit from adjusting RPS when dealing with large-scale data sets. The reason for these improvements is nothing else but the quality improvement of the BPs. Indeed, for *la12* the mean of the quality of BPs increased from 0.16 to 0.43 after adjusting RPS, whereas the variation of the quality remained unchanged. Similar situation holds for *sports*, with the quality of BPs increasing from 0.17 to 0.45.

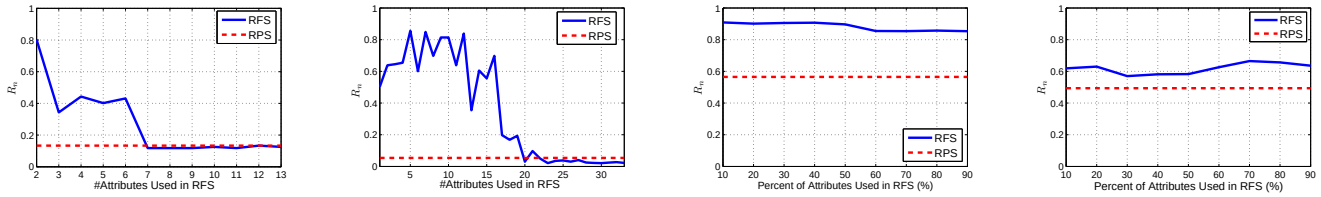
We then illustrate the benefits from using RFS instead of RPS. We employed KCC with RFS on four data sets: *wine*, *dermatology*, *mm*, and *reviews*. For each data set, we gradually increased the number of attributes used to generate basic parititionings (denoted as d), and traced the trend of the clustering performance with the increase of d. Fig. 10 shows the results, where the red dashed-line serves as a benchmark that indicates the original clustering quality using RPS. As can be seen, RFS achieves substantial improvements on *wine* and *dermatology* when d is very small. For instance, the clustering quality on *wine* reaches $R_n = 0.8$ when $d = 2$; it then suffers a sharp fall as d increases, and finally deteriorates to the poor level as RPS, i.e., $R_n < 0.2$, when $d \geq 7$. Similar situation holds for *dermatology*, where KCC with RFS obtains the best clustering results when $5 \leq d \leq 12$. We also tested the performance of RFS on two high-dimensional text data sets *mm* and *reviews*, as shown in Fig. 10, where the ratio of the selected attributes increases gradually from 10% to 90%. The results are still very positive — KCC with RFS leads to consistently higher clustering qualities than KCC with RPS. The reason for the above improvements is similar to the case

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering



(a) *la12* (b) *sports*

Fig. 9. Quality Improvements of KCC by Adjusting RPS



(a) *wine* (b) *dermatology* (c) *mm* (d) *reviews*

Fig. 10. Improvements of KCC by Using RFS

of adjusting RPS; that is, the quality of BPs was improved substantially, e.g., from 0.19 to 0.33 for *wine* when $d = 2$.

Given the above experiments, we can understand that the generation strategy of basic partitionings has great impact to the clustering performance of KCC. RPS with default settings can serve as the primary choice for KCC to gain better diversity, but should be subject to adjustments when dealing with large-scale data sets. RFS is a good alternative to RPS, especially for data sets on which RPS performs poorly, such as the ones with severe noise in features.

6.5 Performances on Incomplete Basic Partitionings

Here, we demonstrate the effectiveness of KCC on handling basic partitionings with missing data. To this end, we adopted two strategies to generate incomplete basic partitionings (IBPs). Strategy-I is to randomly remove some data instances from a data set first, and then employ *kmeans* on the incomplete data set to generate an IBP. Strategy-II is to employ *kmeans* on the whole data set first, and then randomly remove some labels from the complete basic partitioning to get an incomplete one. Data sets *breast_w*, *iris*, *mm* and *reviews* were used with default settings. The removal ratio, denoted as rr, was set from 0% to 90% to watch the variation trend.

Fig. 11 shows the result. To our surprise, IBPs generally exert little negative impact to the performance of KCC until $rr > 70\%$. For the *mm* data set, the clustering quality of KCC is even improved from around 0.6 to over 0.8 when $10\% \leq rr \leq 70\%$! This result strongly indicates that KCC is very robust to the incompleteness of basic partitionings; or in other words, it can help to recover the whole cluster structure based on the cluster fragments provided by IBPs. It is also interesting to see that the effect of Strategy-I seems to be comparable to the effect of Strategy-II. In some cases, Strategy-I even leads to better performance than Strategy-II, e.g., on the *mm* data set, or on the *breat_w* and *iris* data sets when rr is sufficiently large. This observation is somewhat unexpected, since Strategy-II was thought to better reserve the information about the whole cluster structure. This observation is however more valuable, since Strategy-I gets closer to the real-life situations KCC will face in practice.

In summary, KCC shows its robustness in handling incomplete basic partitionings. This further validates its effectiveness for real-world applications.

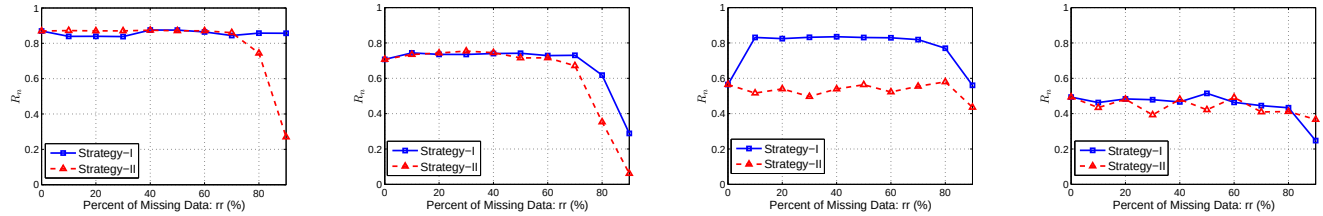
7 Concluding Remarks

In this paper, we established the general theoretical framework of K-means-based consensus clustering (KCC) and provided the corresponding algorithm. We also extended the scope of KCC to the cases where there exist incomplete basic partitionings. Experiments on real-world data sets have demonstrated that KCC has high efficiency and the comparable clustering performance with state-of-the-art methods. In particular, KCC has shown robust performances even if only a few high-quality basic partitions are available or the basic partitionings have severe incompleteness. In the future, we plan to take more research into Theorem 1 from a practical perspective. Specifically, we will study how to derive μ from U, which is useful in some real-world applications.

References

[1] A. Strehl and J. Ghosh, "Cluster ensembles — a knowledge reuse framework for combining partitions," *JMLR*, vol. 3, pp. 583–617, 2002.
[2] N. Nguyen and R. Caruana, "Consensus clusterings," in *ICDM*, 2007.
[3] A. Topchy, A. Jain, and W. Punch, "Clustering ensembles: Models of consensus and weak partitions," *PAMI*, vol. 27, no. 12, 2005.
[4] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *PAMI*, vol. 27, no. 6, pp. 835–850, 2005.
[5] A. Topchy, A. Jain, and W. Punch, "Combining multiple weak clusterings," in *ICDM*, 2003, pp. 331–338.
[6] R. Fischer and J. Buhmann, "Path-based clustering for grouping of smooth curves and texture segmentation," *PAMI*, vol. 25, no. 4, 2003.
[7] T. Li, D. Chris, and I. J. Michael, "Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization," *ICDM*, 2007.
[8] Z. Lu, Y. Peng, and J. Xiao, "From comparing clusterings to combining clusterings," in *AAAI*, 2008, pp. 361–370.
[9] S. Vega-Pons and J. Ruiz-shulcloper, "A survey of clustering ensemble algorithms," *IJPRAI*, vol. 25, no. 3, pp. 337–372, 2011.
[10] Fern and C. E. Brodley, "A survey of clustering ensemble algorithms," *ICML*, 2004.
[11] Abdala, D. Duarte, P. Wattuya, and X. Jiang, "Ensemble clustering via random walker consensus strategy," *ICPR*, 2010.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2014.2316512, IEEE Transactions on Knowledge and Data Engineering



(a) *breast_w* (b) *iris* (c) *mm* (d) *reviews*

Fig. 11. Performances of KCC on Basic Partitionings with Missing Data

[12] Y. Li, J. Yu, P. Hao, and Z. Li, "Clustering ensembles based on normalized edges," *PAKDD*, pp. 664–671, 2007.

[13] X. Wang, C. Yang, and J. Zhou, "Clustering aggregation by probability accumulation," *PR*, vol. 42, no. 5, pp. 668–675, 2009.

[14] H. G. Ayad and M. S. Kamel, "Cumulative voting consensus method for partitions with variable number of clusters," *PAMI*, vol. 30, no. 1, pp. 160–173, 2008.

[15] C. Domeniconi and M. Al-Razgan, "Weighted cluster ensembles: Methods and analysis," *TKDD*, vol. 2, no. 4, 2009.

[16] K. Punera and J. Ghosh, "Consensus-based ensembles of soft clusterings," *AAI*, vol. 22, no. 7-8, pp. 780–810, 2008.

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

[18] B. Mirkin, "The problems of approximation in spaces of relationship and qualitative data analysis," *Information and Remote Control*, vol. 35, p. 1424–1431, 1974.

[19] V. Filkov and S. Steven, "Integrating microarray data by consensus clustering," *IJAIT*, vol. 13, no. 4, pp. 863–880, 2004.

[20] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *TKDD*, vol. 1, no. 1, pp. 1–30, 2007.

[21] A. Goder and V. Filkov, "Consensus clustering algorithms: Comparison and refinement," in *SIAM Workshop on Algorithm Engineering and Experiments*, 2008.

[22] B. Mirkin, "Reinterpreting the category utility function," *ML*, vol. 45, no. 2, pp. 219–228, November 2001.

[23] A. Topchy, A. Jain, and W. Punch, "A mixture model for clustering ensembles," in *SDM*, Florida, USA, 2004.

[24] S. Vega-Pons, J. Correa-Morris, and J. Ruiz-Shulcloper, "Weighted partition consensus via kernels," *PR*, vol. 43, no. 8, 2010.

[25] H. Luo, F. Jing, and X. Xie, "Combining multiple clusterings using information theory based genetic algorithm," *International Conference on Computational Intelligence and Security*, vol. 1, pp. 84–89, 2006.

[26] T. Li, M. M. Ogihara, and S. Ma, "On combining multiple clusterings: an overview and a new perspective," *Applied Intelligence*, vol. 32, no. 2, pp. 207–219, 2010.

[27] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *BSMSP*, L. L. Cam and J. Neyman, Eds., vol. 1, Statistics. University of California Press, 1967.

[28] Teboulle, "A unified continuous optimization framework for center-based clustering methods," *JMLR*, vol. 8, pp. 65–102, 2007.

[29] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison-Wesley, 2005.

[30] L. Bregman, "The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, pp. 200–217, 1967.

[31] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *JMLR*, vol. 6, pp. 1705–1749, 2005.

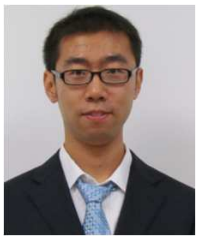
[32] A. Banerjee, X. Guo, and H. Wang, "On the optimality of conditional expectation as a bregman predictor," *TIT*, vol. 51, no. 7, 2005.

[33] J. Wu, H. Xiong, C. Liu, and J. Chen, "A generalization of distance functions for fuzzy c-means clustering with centroids of arithmetic means," *TFS*, vol. 20, no. 3, pp. 557–571, 2012.

[34] M. DeGroot and M. Schervish, *Probability and Statistics (3rd Edition)*. Addison Wesley, 2001.

[35] J. Wu, H. Xiong, and J. Chen, "Adapting the right measures for k-means clustering," in *KDD*, Paris, France, 2009, pp. 877–886.

Junjie Wu received his Ph.D. degree in Management Science and Engineering from Tsinghua University in 2008. He also holds a B.E. degree in Civil Engineering from the same university. He is currently an Associate Professor in Information Systems Department, School of Economics and Management, Beihang University, the director of Social Computing and Sentimental Analysis Center, and the vice director of Beijing Key Laboratory of Emergency Support Simulation Technologies for City Operations. His general area of research is data mining and complex networks.



Hongfu Liu received his B.E. degree in Management Information Systems from the School of Economics and Management, Beihang University, in 2011. He is currently a master student in the same department. His research interests generally focus on data mining and machine learning, with special interests in ensemble learning.

Hui Xiong is currently an Associate Professor and the Vice Chair of the Management Science and Information Systems department at Rutgers University. He received the B.E. degree from the University of Science and Technology of China (USTC), China, the M.S. degree from the National University of Singapore (NUS), Singapore, and the Ph.D. degree from the University of Minnesota (UMN), USA. His general area of research is data and knowledge engineering, with a focus on developing effective and efficient data analysis techniques for emerging data intensive applications.

Jie Cao received the Ph.D. degree from Southeast University, China, in 2002. He is currently a professor and the director of Jiangsu Provincial Key Laboratory of E-Business at Nanjing University of Finance and Economics. He has been selected in the Program for New Century Excellent Talents in University (NCET) and awarded with Young and Mid-aged Expert with Outstanding Constribution in Jiangsu province. His main research interests include cloud computing, business intelligence and data mining.

Jian Chen received the B.Sc. degree in Electrical Engineering from Tsinghua University, China, in 1983, and the M.Sc. and the Ph.D. degree both in Systems Engineering from the same University in 1986 and 1989, respectively. He is Lenovo Chair Professor and Chairman of Management Science Department, Director of Research Center for Contemporary Management, Tsinghua University. His main research interests include supply chain management, E-commerce, decision support systems.