Information Sciences 571 (2021) 418–442

Contents lists available at ScienceDirect

Information Sciences

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

Clustering mixed numerical and categorical data with missing values

Duy-Tai Dinh [a,*], Van-Nam Huynh [a,*], Songsak Sriboonchitta [b]

[a] Japan Advanced Institute of Science and Technology, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
[b] The Centre of Excellence in Econometrics, Faculty of Economics, Chiang Mai University, Thailand

ARTICLE INFO

Article history:
Received 12 February 2019
Received in revised form 17 April 2021
Accepted 20 April 2021
Available online 24 April 2021

Keywords:
Clustering
Partitional clustering
Mixed data
Missing values

ABSTRACT

This paper proposes a novel framework for clustering mixed numerical and categorical data with missing values. It integrates the imputation and clustering steps into a single process, which results in an algorithm named Clustering Mixed Numerical and Categorical Data with Missing Values (k-CMM). The algorithm consists of three phases. The initialization phase splits the input dataset into two parts based on missing values in objects and attributes types. The imputation phase uses the decision-tree-based method to find the set of correlated data objects. The clustering phase uses the mean and kernel-based methods to form cluster centers at numerical and categorical attributes, respectively. The algorithm also uses the squared Euclidean and information-theoretic-based dissimilarity measure to compute the distances between objects and cluster centers. An extensive experimental evaluation was conducted on real-life datasets to compare the clustering quality of k-CMM with state-of-the-art clustering algorithms. The execution time, memory usage, and scalability of k-CMM for various numbers of clusters or data sizes were also evaluated. Experimental results show that k-CMM can efficiently cluster missing mixed datasets as well as outperform other algorithms when the number of missing values increases in the datasets.

1. Introduction

Data clustering is a method of creating groups of objects in such a way that objects in the same cluster are very similar, but objects in different clusters are quite distinct [18]. In other words, with a set of data instances, the fundamental problem of clustering is to partition it into a set of groups that are as similar as possible [1]. Clustering has been applied in many areas of science and engineering, such as natural science, social science, life science, earth science, information science, medical science, policy, and decision making [3]. Further, it can be adapted into the intermediate steps of other research topics and applications such as bioinformatics, collaborative filtering, customer segmentation, data exploration, data summarization, dynamic trend detection, information retrieval, market basket analysis, medical diagnostics, multimedia data analysis, social network analysis, text mining, and web analysis [4]. Fig. 1 graphically depicts the classification of clustering algorithms. It consists of two categories: hard clustering and fuzzy clustering (or soft clustering). In the first category [8,16,19,20,27], a data point belongs to one and only one cluster, whereas in the second category [11,25,35,36,41,47,49], a data point can belong to two or more clusters with some probability. In each category, clustering algorithms can belong

* Corresponding author.
E-mail addresses: taidinh@jaist.ac.jp (D.-T. Dinh), huynh@jaist.ac.jp (V.-N. Huynh).

https://doi.org/10.1016/j.ins.2021.04.076
0020-0255/© 2021 The Author(s). Published by Elsevier Inc.

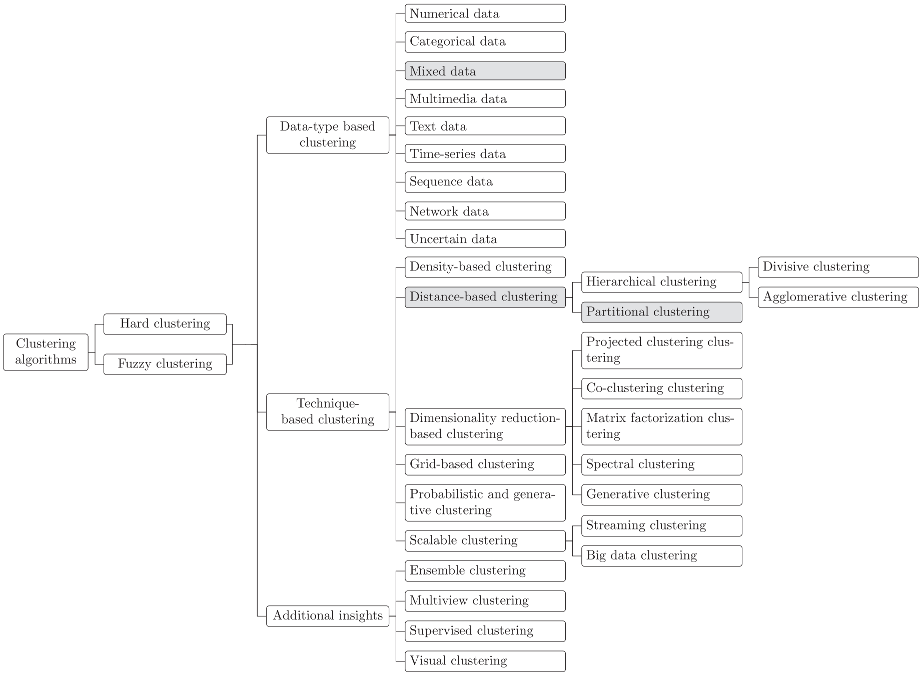


Fig. 1. Classification of clustering algorithms.

to several groups such as data-type-based clustering [8,14–16,19–21,27,29,30], technique-based clustering [9,28,31–33,41–43,49], and additional insights [14,31,44]. In technique-based clustering, the distance-based method is commonly used for wide-ranging applications because it can handle almost all data types as long as the algorithm uses a suitable distance function for an input dataset. As a result, the clustering task can be reduced to the task of finding a distance measure for a certain data type [1]. The two types of distance-based clustering are hierarchical and partitional. Hierarchical clustering is a set of nested clusters presented in a dendrogram, in which lower-level clusters are sub-clusters of higher-level clusters, whereas partitional (or partitioning) clustering creates a one-level non-overlapping partitioning of data instances [7,12,13]. In a dataset with n data instances, the standard algorithms of hierarchical clustering have time complexities of $\mathcal{O}(n^3)$ and require $\mathcal{O}(n^2)$ for memory spaces, which makes them too slow for even medium datasets. Partitional clustering algorithms, which do not need to store large proximity matrices and often converge at a local optimum with the time and space complexity of $\mathcal{O}(n)$ [50], are computationally more efficient than hierarchical methods.

Clustering algorithms have been designed for certain types of data. The numerical data are represented by continuous values, whereas the categorical data, which is a special case of the discrete type, can have only a finite number of values. Categorical data appear frequently in many real-life applications, such as name, gender, age group, educational level, and blood type. The mixed datasets contained both numerical and categorical values. Real-life data are often mixed types. Medical data, for example, include categorical values such as nationality, gender, job, education, marital status, and smoking or nonsmoking, as well as numerical values such as age, height, weight, and salary. Retail purchase transactions consist of categorical values, such as categories of items, types, and customers' locations, as well as numerical values such as quantity, unit profit, and price. During the last two decades, many attempts have been made to solve the problem of mixed data clustering. This study focuses on the design of a clustering algorithm for mixed data with missing values. The mechanisms of the proposed algorithm are based on the following observations.

Observation 1. Clustering is one of the most popular research topics in data mining and knowledge discovery for databases. As mentioned earlier, its applications have been used in a wide range of areas and can be adapted to other research topics. Although the literature on clustering abounds, no perfect model exists that can solve all clustering tasks. Additionally, from both research and application viewpoints, the interest in clustering seems to be unwaning. Extensive research for missing values can be found in the literature. However, few studies have focused on the combination of clustering in the presence of missing values, especially for missing data in mixed numerical and categorical features.

Observation 2. Most clustering algorithms handle either categorical or numerical data. Before such algorithms are used, data preprocessing such as discretization or one-hot encoding is performed to convert the numerical data to categorical data and vice versa. However, the discretization process leads to information loss because the membership degree of a value to discretized values is not considered, although assigning correct numerical values to categorical values is challenging if these values have no intrinsic order. For example, encoding the color attribute that contains categorical values such as red, green, and blue to integer values of one, two, and three will not be appropriate because these values have the same order. An inappropriate transformation can impair the information inherent in the values, resulting in misleading outcomes [23].

Observation 3. In this era of information technology, data can be collected from various sources, such as sensors, digital devices, machines, and humans. These sources generate large amounts of data in a short time. Collecting data is not always an easy task, however, and can lead to missing values in the data (simply called missing data or missing values) due to different mechanisms: equipment malfunctions, human error, data not entered due to misunderstanding, data deletion due to inconsistency between recorded data, system generated errors, certain data not be considered significant at the time of entry, and changes in the data. Missing data can also occur due to observation conditions, instrument sensitivity limitations, and other real-life considerations. Unfortunately, they may hide the correct answers underlying in the data. They can also reduce the performance of the algorithms. To overcome these problems, researchers need to impute or discard missing values in advance when using statistical analysis methods that require complete data. However, simply discarding missing data is not a reasonable practice, because valuable information can be lost and inferential power can be compromised. This can cause selection bias in some cases. Additionally, deleting observations with missing values can result in very few observations remaining in the data when a large number of predictive variables contain missing values. As a result, imputing the data before performing any analysis is advised.

Observation 4. Many datasets from the UCI Machine Learning Repository contain missing values. Fig. 2 shows the Hepatitis and Horse colic datasets with the missing rates of approximately 6% and 20%, respectively. Further, existing frameworks and packages for clustering categorical and mixed data[1] strongly recommend that datasets should be filled in a way that makes sense for the problem at hand, especially in the case of many missing values.

These observations motivated the design of an algorithm that can cluster mixed numerical and categorical data with missing values. Generally, existing algorithms use two approaches to cluster the missing data:

1. Datasets are preprocessed so that they do not contain any missing values and are then passed as an input to the clustering algorithms.
2. Clustering algorithms are designed to work directly with incomplete datasets.

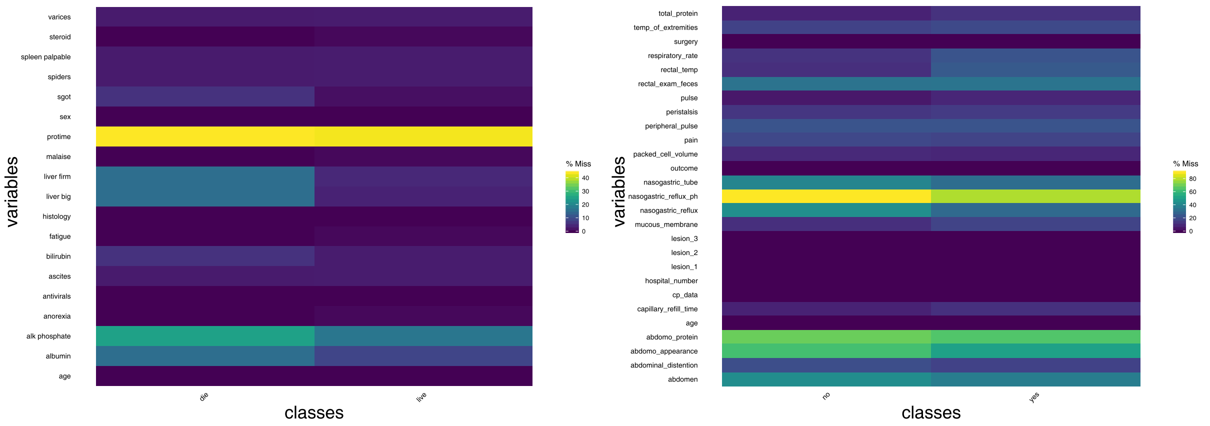
This study focused on the second approach for the proposed algorithm. Specifically, we integrated the imputation process into the clustering process so that users do not need to preprocess datasets such as filling missing values and discretizing the data beforehand. The major contributions and innovations of this study are as follows:

- To the best of our knowledge, this is the first study that considers combining partitional clustering and missing value imputation for mixed numerical and categorical data with missing values. The purpose is to design a new clustering framework that takes advantage of missing values imputation to improve the quality of the clustering results. The proposed framework does not require users to preprocess the input dataset with missing values. Additionally, it can be used to perform clustering tasks for any real mixed dataset as long as the formats match the input requirement of the algorithms.
- We extended the decision-tree-based imputation method proposed in [10] to impute missing values occurring in categorical attributes. In particular, we designed a new similarity measure, MCS, to estimate the similarity between complete and incomplete objects. Compared to the weighted similarity measure (WSM) used in the previous method, the proposed similarity can reduce the complexity of the imputing method.
- We proposed an algorithm named Clustering Mixed Numerical and Categorical Data with Missing Values (k-CMM). The algorithm consists of three phases: initialization, imputation, and clustering. The three phases are integrated, with the output of the current phase being the input of the next phase. The second and third phases are performed to impute missing values within objects and assign them to k clusters.

+

For clustering, k-CMM uses the kernel density estimation method (kernel-based method) and the means for the formation of a cluster center. The kernel-based method, which can be regarded as probability-based modeling, allows the interpretation of cluster centers for categorical attributes to be consistent with the statistical interpretation of the cluster means for numerical attributes. It also supports the algorithm for describing cluster structures inside the data. Moreover, it also facilitates the formulation of the information-theoretic-based distance (ITBD) measure for categorical attributes. Along with the use of the squared Euclidean for numerical attributes, the ITBD uses the logarithmic function and the relative frequencies of categories in the attributes to determine which pairs of values are more or less similar; as a result, it is applicable for categorical attributes whose domains have probabilistic models.

[1] https://github.com/nicodv/kmodes.



(a) Hepatitis dataset

(b) Horse colic dataset

Fig. 2. Datasets with missing values.

+

For imputation, we extended the ITDB measure to estimate the similarity between complete and incomplete mixed objects: namely, MCS measure. k-CMM applies the decision-tree-based imputation method, taking advantage of the decision tree to find the set of correlated objects for each incomplete object. It then determines the candidate values for missing positions by examining the correlation between missing and complete attributes in an incomplete object by using the IS measure, and the similarity between the incomplete and complete objects in the correlated set by using the MCS measure. The decision-tree-based imputation approach provides an interpretable method that considers all possible outcomes and traces each path to a decision.

– An extensive experiment conducted on real-life datasets evaluated the performance of k-CMM in terms of clustering quality, computational complexity, and scalibility. Experimental results show that both imputation and clustering steps can enhance clustering results. Further, the proposed algorithm outperforms the previous algorithm for mixed data with missing values.

The rest of this paper is organized as follows. Section 2 reviews previous works related to clustering and missing-value imputation. Section 3 provides preliminaries and definitions. Section 4 describes the proposed k-CMM algorithm. Section 5 discusses the experimental results. Section 6 draws conclusions and outlines directions for future work.

2. Related work

The k-means algorithm [27] splits a numerical dataset into a predetermined number of k clusters. This algorithm consists of two phases: initialization and iteration. In the initialization phase, the algorithm randomly assigns objects to k clusters. In the iteration phase, the algorithm uses the Euclidean distance to compute the distance between each object and each cluster and then assigns the object to the nearest cluster. The main steps of the k-means algorithm are as follows: First, it randomly selects k objects from dataset X, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. The algorithm then iteratively improves within-cluster variation. For each cluster, the new mean is computed by using the objects assigned to the cluster in the previous iteration. All objects are then reassigned by using the updated means as the new cluster centers. The assignment and update steps are performed in turn until all clusters are stable, which means that the clusters formed in the current round are the same as those formed in the previous round. K-means has some remarkable properties. It efficiently clusters large datasets because its computational complexity is linearly proportional to the size of the datasets. It also often terminates at a local optimum, with its performance depending on the initialization of the centers [18]. However, only working on numerical data prohibits some applications of the k-means algorithm in which categorical and mixed data are involved. The traditional approach to converting categorical data into numerical values does not produce meaningful results when categorical domains are not ordered [20]. Several attempts have been made to remove this limitation while retaining the advantages of k-means, so-called k-means-like algorithms [8,14–16,19,29,30,40]. K-modes [19] can be considered as pioneering work for clustering categorical data. This algorithm first initializes k initial modes and then allocates every object to the nearest mode. It uses modes to represent clusters and a frequency-based method to update the modes in the clustering process. The mode of a cluster is a data point whose attribute values are assigned by the most frequent values of the attribute domain set appearing in the cluster. Because k-modes originates from k-

means, it can also be treated as an optimization problem. The objective function of the k-modes can be formulated by changing the Euclidean distance to a simple matching distance as follows:

$$F(U,Z) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m} u_{i,l} \times \delta(x_{i,j}, z_{l,j}) \tag{1}$$

Here, k, n, and m are the number of clusters, categorical objects, and categorical attributes in a dataset, respectively; $U = [u_{i,l}]$ is an $n \times k$ partition matrix; and $Z = \{Z_1, Z_2, \ldots, Z_k\}$ is a set of mode vectors. Each mode vector Z_i consists of m categorical values $(z_{i1}, z_{i2}, \ldots, z_{im})$, with each being the mode of an attribute. With x and y being two categorical values, the simple matching distance between x and y is provided by

$$\delta(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise} \end{cases}$$

The k-modes has the same properties as those of k-means, except that it works only on categorical data.

In 2004, San et al. proposed a k-representatives algorithm [40]. Cartesian product and union operations were used to form cluster centers. This algorithm uses the relative frequencies of categorical values within the cluster and a simple matching measure between categorical values to quantify the distance between objects. The main procedure of the k-representatives was performed in the same manner as the k-modes. In 2013, Chen et al. proposed the k-centers algorithm [8], which uses the kernel density estimation approach to form cluster centers. Furthermore, to measure the individual contribution of each attribute to clusters, it uses a built-in feature in which categorical attributes are assigned weights. The dissimilarity is measured by using simple matching as an indicator function to represent each data object by a set of vectors and the Euclidean norm. Recently, Nguyen et al. proposed a new extension of the k-means for categorical data by combining a modified concept of cluster centers based on the kernel-based estimation method and the information-theoretic-based dissimilarity measure [29]. More specifically, it extends the kernel function for cluster centers (proposed in the k-centers algorithm) rather than use the attribute's entire domain with regard to the cardinality of an attribute subdomain of a certain cluster.

For clustering mixed numerical and categorical data, Huang proposed the k-prototypes algorithm [20]. The clustering process of the k-prototypes algorithm is similar to the k-means algorithm, except that it uses the k-modes approach to update the categorical attribute values of cluster prototypes. To quantify the distance between objects, it uses the squared Euclidean distance and simple matching for numerical attributes and categorical attributes, respectively. Several studies [21,23] have been proposed to improve the k-prototypes algorithm. A set of n mixed objects are described by attributes $A_1^r, A_2^r, \ldots, A_p^r, A_{p+1}^c, \ldots$, and A_m^c. The objective function for the k-prototypes is

$$F(U,Z) = \sum_{l=1}^{k}(F_l^r + F_l^c) \tag{2}$$

where $F_l^r = \sum_{i=1}^{n} u_{i,l} \sum_{j=1}^{p}(x_{i,j} - z_{l,j})^2$ and $F_l^c = \gamma \sum_{i=1}^{n} u_{i,l} \sum_{j=p+1}^{m} \delta(x_{i,j}, z_{l,j})$, in which γ is a balance weight used to avoid favoring either type of attribute.

Several studies can be found for the task of clustering missing data. In 2015, Wilson [48] developed a marginal multivariate Gaussian density method for incomplete continuous data. This method falls into the group of model-based clustering, which estimates the structure of clusters by using the likelihood to infer cluster information of a certain dataset based on the assumption that the data comes from a joint multivariate Gaussian distribution. The simulation compared the marginal density-based method with other approaches, such as multiple imputation and the complete case method that deletes incomplete observations. As mentioned in [48], the marginal method only works well in a simple setting; when the data structure is more complicated, multiple imputation is preferable. Further, the complete case is more suitable than the marginal density method for large-sized data with low dimensions and few missing values. Furthermore, the datasets used in the simulation are assumed to be continuous, quantitative, and multivariate data making them suitable for Gaussian mixture modeling clustering methods. In this way, the scope of these methods is restricted to quantitative continuous data with random missing.

In 2016, Pattanodom et al. [34] proposed an ensemble clustering framework for data with missing values. Specifically, the framework first creates different variations in the input dataset by randomly filling missing values in the data with candidates selected from existing domains. It then uses a binary cluster association matrix that indicates the membership between any pairs of objects and clusters to summarize the ensemble information. Finally, it uses k-means to cluster the resulting matrix. The main target of this framework is to take advantage of generating various versions of the input data in combination with other generation strategies, such as using different k to promote diversity within an ensemble. The experiment was conducted on several numerical datasets obtained from the UCI repository and compared BA-Matrix with several methods such as linear regression, KNN, and expectation–maximization methods in terms of accuracy. However, the framework proposed in this paper can only be applied to numerical data. The performance also depends on randomly filling for missing values.

In 2019, Boluki et al. [5] proposed a new Random Labeled Point Process (RLPP) by incorporating the generation of missing values with the original generating RLPP. The optimal clusters in the context of missing values can be obtained by marginalizing the missing attributes in the new RLPP. The proposed framework was derived for different scenarios in which attributes were followed by multivariate Gaussian distributions. The simulations, performed on synthetic data and RNA-seq data, evaluated the performance of RLPP in comparison with other methods such as fuzzy c-means, k-means, and hierarchical clustering. However, as mentioned in this paper, conducting the marginalization can be computationally intractable in the case of missing values with general patterns, and thus approximation methods such as Monte Carlo integration are required.

Several previous studies have considered the combination of clustering and missing-value imputation, but they also have several limitations, as discussed above. Moreover, some of algorithms were designed mainly for pure numerical data and, thus, cannot be applied to mixed data without any modification. The following sections define the problem of clustering mixed data with missing values and then propose an algorithm called k-CMM.

3. Preliminaries and problem statement

The problem of clustering mixed data has been the subject of several prior studies [20,23]. Table 1 shows the list of notations used in this paper. Use $A^r = \{A^r_1, A^r_2, \ldots, A^r_p\}$ as the set of p numerical attributes, in which the domain of A^r_α $(1 \leqslant \alpha \leqslant p)$, denoted by $DOM(A^r_\alpha)$, is represented by continuous values. Use $A^c = \{A^c_1, A^c_2, \ldots, A^c_q\}$ as the set of q categorical attributes, in which the domain of A^c_β $(1 \leqslant \beta \leqslant q)$, denoted by $DOM(A^c_\beta)$, is represented by a finite and unordered set that contains only singletons. Use $A = \{A_1, A_2, \ldots, A_m\}$ such that $A = A^r \cup A^c$ and $m = p + q$ as the set of m distinct attributes, in which the d^{th} attribute A_d $(1 \leqslant d \leqslant m)$ is either a numerical or categorical attribute. A mixed numerical and categorical object (mixed object or mixed record) is a tuple of the form $\langle id, x \rangle$, in which id is its unique identifier and x is represented by a tuple $t \in DOM(A_1) \times DOM(A_2) \times \ldots \times DOM(A_m)$. For simplicity, a mixed object x with $id = i$ is denoted as x_i. A mixed dataset $S = \{x_1, x_2, \ldots, x_n\}$ is a set of n mixed objects, in which $x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ is a set of m mixed numerical and categorical values at the i^{th} element of S. In contrast, S is an $m \times n$ matrix $(n \gg m)$, in which m and n comprise the number of attributes and objects in dataset S, respectively. The element at position (i, d) $(1 \leqslant i \leqslant n, 1 \leqslant d \leqslant m)$ of the matrix stores the value of the object i^{th} at the attribute d^{th} so that $x_{id} \in DOM(A) = DOM(A^r) \cup DOM(A^c)$. We also designed a framework that allows for clustering a pure categorical dataset containing missing values—that is, a dataset with the attributes $A = A^c$. Note that if a categorical value in S is a missing value, it is represented as "?" or " " (empty). For example, Table 2 shows a mixed dataset with six attributes; the first four attributes are categorical, whereas the last two attributes are numerical. It contains 15 objects with eight missing values and is used for the running example. For the sake of brevity, an object with and without missing values is denoted as an incomplete and complete object, respectively, whereas a dataset with and without missing values is denoted as the incomplete and complete dataset, respectively.

Definition 1 (*Clusters*). A mixed dataset S and set $C = \{C_1, C_2, \ldots, C_k\}$ contains k disjoint subsets. C_j $(1 \leqslant j \leqslant k)$ is called a cluster of S iff for every $C_i \in C$ $(1 \leqslant i \leqslant k \wedge i \neq j), C_j \cap C_i = \varnothing$, and $S = \bigcup_{j=1}^{k} C_j$. The number of data objects in cluster C_j is denoted by n_j.

Example 1. Assume that three subsets of $S: C_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}, C_2 = \{x_4, x_6, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}\}, C_3 = \{x_1, x_2, x_3\}$ appear in the dataset shown in Table 2. Then, $\{C_1, C_2\}$ are clusters of S, whereas $\{C_1, C_3\}, \{C_2, C_3\}$ and $\{C_1, C_2, C_3\}$ are not.

Definition 2 (*Relative frequency of a categorical value*). If a cluster C_j and a categorical value o^j_{ld} appear in C_j at the d^{th} categorical attribute, the relative frequency of o^j_{ld} in C_j is denoted as

$$f_j(o^j_{ld}) = \frac{\#_j(o^j_{ld})}{n_j} \tag{3}$$

in which $\#_j(o^j_{ld})$ is the number of categorical values o^j_{ld} appearing in cluster C_j at the d^{th} attribute. The relative frequency of o^j_{ld} in dataset S at the d^{th} categorical attribute is denoted and defined as follows:

$$f(o^j_{ld}) = \frac{\#(o^j_{ld})}{|S|} \tag{4}$$

Here, $\#(o^j_{ld})$ is the number of categorical values o^j_{ld} appearing in dataset S at the d^{th} attribute.

Example 2. In the dataset shown in Table 2, assume that if cluster $C_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$, then the relative frequency of the categorical value b in the attribute A_1 is $f_1(b) = \frac{3}{6} = 0.5$.

Partitional clustering algorithms use a center to represent each cluster during the clustering process. In this study, to represent the cluster centers, we used the *mean* for numerical attributes and the variation in Aitchison & Aitken's kernel function [2] to estimate the probability density function of each categorical attribute in the center.

Table 1
Table of notations.

Symbol	Description
k	Number of clusters
x_i	Mixed object with index i
x_i^r	Numerical part of object x_i
x_i^c	Categorical part of object x_i
X_d	Random variable
S	Mixed dataset
A_d	d^{th} Attribute of dataset S
A_d^r	d^{th} Numerical attribute of dataset S
A_d^c	d^{th} Categorical attribute of dataset S
C_j	j^{th} Cluster
Z_j	Center of cluster C_j
x_{id}	Value appears at the i^{th} element and d^{th} attribute of dataset S
$\bar{o}_{ld}^j$	Numerical value appears at the l^{th} element and d^{th} attribute of cluster C_j
o_{ld}^j	Categorical value appears at the l^{th} element and d^{th} attribute of cluster C_j
O_d	Set of categorical values appears at the d^{th} attribute of dataset S
O_d^j	Set of categorical values appears at the d^{th} attribute of cluster C_j
z_d^{rj}	Value of the d^{th} numerical attribute in the center Z_j
z_d^{cj}	Value of the d^{th} categorical attribute in the center Z_j

Table 2
Mixed numerical and categorical dataset with missing values.

	Attrs					
Objs	A_1	A_2	A_3	A_4	A_5	A_6
x_1	d	b	f	e	7	14
x_2	b	b	c	e	6	13
x_3	b	b	c	e	2	13
x_4	a	b	a	b	5	13
x_5	c	a	f	d	2	14
x_6	d	b	f	?	?	14
x_7	a	a	c	e	1	14
x_8	b	a	?	?	?	12
x_9	c	b	a	c	5	14
x_{10}	b	?	d	e	5	7
x_{11}	d	a	d	c	10	13
x_{12}	d	b	d	d	3	12
x_{13}	?	b	a	e	2	?
x_{14}	a	a	d	d	2	18
x_{15}	b	a	f	e	2	14

Definition 3 (*The mean of numerical attributes in a cluster*). A cluster C_j contains p numerical attributes $A^r = \{A_1^r, A_2^r, \dots, A_p^r\}$ and Z_j is the center of C_j. The mean of each attribute A_d^r $(1 \leqslant d \leqslant p, p < m)$ in cluster C_j is defined as follows:

$$z_d^{rj} = \frac{1}{n_j}\sum_{l=1}^{n_j} \bar{o}_{ld}^j \tag{5}$$

Example 3. In the dataset shown in Table 2, assuming that cluster $C_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$, then the mean value of attribute A_6 is $z_6^{r1} = \frac{(14+13+13+14+14+14)}{6} = 13.67$.

In a numerical setting, a cluster center can be considered as the expectation of a continuous random variable associated with the observations inside the cluster, in which the variable is assumed to follow a Gaussian distribution. To retain the advantages of k-means in a categorical setting, the cluster centers can be estimated using the kernel density estimation (KDE) approach. This guarantees consistency in the statistical interpretation of the cluster centers for categorical data as the mean for numerical data. This method estimates the probability distribution of a random variable based on a random sample. This study also uses KDE to define cluster centers, called probabilistic centers. Recall that a density estimator is an algorithm that takes a d-dimensional dataset and produces an estimate of the d-dimensional probability distribution from which the data are drawn.

Definition 4 (*Kernel density estimation for categorical data [29]*). Let cluster C_j exist. Let X_d be a random variable associated with observations o^j_{ld} $(1 \leqslant l \leqslant n_j)$ appearing in C_j at the d^{th} attribute, and let $p(X_d)$ denote the probability density of X_d. Let O^j_d be the set of categorical values in C_j at the d^{th} attribute so that $O^j_d = \bigcup_{l=1}^{n_j} o^j_{ld}$ and $\lambda_j \in [0,1]$ be the unique smoothing bandwidth for cluster C_j. For each value o^j_{ld} in O^j_d, the variation in Aitchison & Aitken's kernel function is denoted and defined as

$$K(X_d, o^j_{ld}, \lambda_j) = \begin{cases} 1 - \frac{|O^j_d|-1}{|O^j_d|}\lambda_j & \text{if } X_d = o^j_{ld} \\ \frac{1}{|O^j_d|}\lambda_j & \text{otherwise} \end{cases} \tag{6}$$

The parameter λ is a unique smoothing bandwidth that uses least-squares cross-validation to minimize the total error of the resulting estimation over all data objects. The optimal smoothing bandwidth parameter for cluster C_j is defined as

$$\lambda_j = \frac{1}{n_j - 1} \frac{\sum_{d=1}^{q}\left(1 - \sum_{o^j_{ld} \in O^j_d} [f_j(o^j_{ld})]^2\right)}{\sum_{d=1}^{q}\left(\sum_{o^j_{ld} \in O^j_d} [f_j(o^j_{ld})]^2 - \frac{1}{|O_d|}\right)} \tag{7}$$

Note that the kernel function of a categorical value at the d^{th} attribute is defined in terms of the cardinality of the domain O^j_d of the cluster C_j instead of the cardinality of the entire domain O_d. The kernel density estimation of $p(X_d)$ is denoted and defined as

$$\hat{p}(X_d, \lambda_j, C_j) = \sum_{o^j_{ld} \in O^j_d} f_j(o^j_{ld}) K(X_d, o^j_{ld}, \lambda_j) \tag{8}$$

Definition 5 (*Center of cluster*). With cluster $C_j = \{x_1, x_2, \ldots, x_{n_j}\}, x_i = (x_{i1}, x_{i2}, \ldots, x_{im}), m = |A|$. The center of C_j is then defined as

$$Z_j = \{z^j_1, z^j_2, \ldots, z^j_m\} \tag{9}$$

in which the d^{th} attribute z^j_d $(1 \leqslant d \leqslant m)$ is either z^{rj}_d or z^{cj}_d. Specifically, if the d^{th} attribute of Z_j is a numerical attribute, its representative is calculated by using Definition 3 (Eq. 5). Otherwise, the representative of a categorical attribute of Z_j is the probability distribution on O^j_d estimated by the kernel density estimation method by using Eq. (8), defined as

$$z^{cj}_d = [P^j_d(o^j_{1d}), P^j_d(o^j_{2d}), \ldots, P^j_d(o^j_{|O^j_d|d})] \tag{10}$$

Here, the probabilistic value of a categorical value o^j_{ld} $(1 \leqslant l \leqslant n_j)$ can be estimated based on Eqs. (3), (6), and (8) as follows:

$$P^j_d(o^j_{ld}) = \begin{cases} \lambda_j \frac{1}{|O^j_d|} + (1-\lambda_j) f_j(o^j_{ld}) & \text{if } o^j_{ld} \in O^j_d \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Example 4. In the dataset shown in Table 2, assume that cluster $C_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$, the smoothing bandwidth parameter of this cluster is $\lambda_1 = 0.4827$. Then the center Z_1 of C_1 at the categorical attributes A_1, A_2, A_3, A_4 and numerical attributes A_5, A_6 are {'d': 0.2069, 'b': 0.3793, 'c': 0.2069, 'a': 0.2069}, {'b': 0.5, 'a': 0.5}, {'f': 0.5, 'c': 0.5}, {'e': 0.6724, 'd': 0.3275}, 3.5556, and 14.1111, respectively.

Previous studies used several methods to quantify the dissimilarity between a mixed object and its center [20,23]. In particular, distance measures such as Euclidean, Manhattan, Minkowski, and Mahalanobis [18] can be applied for numerical attributes, whereas the simple matching dissimilarity measure [19,20,40], the Euclidean norm [8], and the information-theoretic-based dissimilarity measure [29] can be applied to categorical attributes. In this study, we used the squared Euclidean and the information-theoretic-based dissimilarity measure to quantify the dissimilarity between numerical and categorical attributes in mixed objects. The information-theoretic definition of similarity [24] is applicable to domains with probabilistic models.

Theorem 1 (*Similarity theorem for the probabilistic model [24]*). *The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are:*

$$\mathrm{sim}(A,B)=\frac{\log P(\mathrm{common}(A,B))}{\log P(\mathrm{description}(A,B))} \tag{12}$$

Here, $P(s)$ is the probability of a statement s.

Based on the Theorem 1, Nguyen et al. [29] proposed an information-theoretic-based dissimilarity measure for categorical data. In this study, we also used this measure to quantify the dissimilarity between a mixed object and a cluster.

Definition 6 (*Information-theoretic-based dissimilarity measure [29]*). The similarity between two categorical values o^j_{ld} and $o^j_{l'd}$ at the d^{th} attribute is defined as

$$\mathrm{sim}_d(o^j_{ld},o^j_{l'd})=\frac{2\log f(o^j_{ld},o^j_{l'd})}{\log f(o^j_{ld})+\log f(o^j_{l'd})} \tag{13}$$

in which $f(o^j_{ld},o^j_{l'd})=\frac{\#(o^j_{ld},o^j_{l'd})}{|S|}$ with $\#(o^j_{ld},o^j_{l'd})$ is the number of mixed objects in dataset S that receives the categorical values belonging to $\{o^j_{ld},o^j_{l'd}\}$ at the d^{th} attribute, whereas $f(o^j_{ld})$ is measured by the Eq. 4.

The dissimilarity measure between two categorical values o^j_{ld} and $o^j_{l'd}$ at the d^{th} attribute can be defined as

$$\mathrm{dsim}_d(o^j_{ld},o^j_{l'd)})=1-\mathrm{sim}_d(o^j_{ld},o^j_{l'd})=1-\frac{2\log f(o^j_{ld},o^j_{l'd})}{\log f(o^j_{ld})+\log f(o^j_{l'd})} \tag{14}$$

Example 5. In the dataset shown in Table 2, we omitted four incomplete objects from the dataset. The dissimilarity of categorical values d and b in the attribute A_1 is $\mathrm{dsim}_1(d,b)$ = 1 $-\frac{2\times\log(\frac{6}{11})}{\log(\frac{3}{11})+\log(\frac{3}{11})}$ = 0.5335.

Definition 7 (*Dissimilarity between a mixed object and a cluster*). Consider a cluster C_j and a mixed object $x_i = (x_{i1},x_{i2},\ldots,x_{im})$. The dissimilarity between x_i and the center $Z_j=\{z^j_1,z^j_2,\ldots,z^j_m\}$ at the d^{th} attribute is defined as

$$\mathrm{dis}_d(x_i,Z_j)=\begin{cases}\left(x_{id}-z^{rj}_d\right)^2 & \text{if the } d^{th} \text{ attribute is a numerical attribute}\\ \sum\limits_{o^j_{ld}\in O^j_d} P^j_d(o^j_{ld})\mathrm{dsim}_d(x_{id},o^j_{ld}) & \text{if the } d^{th} \text{ attribute is a categorical attribute}\end{cases} \tag{15}$$

Specifically, the dissimilarity between x_i and Z_j at the d^{th} attribute is measured based on the attribute type. For numerical attributes, the squared Euclidean distance quantifies the proximity between the mean of the clusters and the numerical value of mixed objects. For categorical attributes, proximity is measured by accumulating the probability distribution on O^j_d and the dissimilarity between the d^{th} component x_{id} of the object x_i and the d^{th} component z^{cj}_d of the center Z_j. Finally, the dissimilarity between the mixed object x_i and cluster center Z_j is defined as

$$\mathrm{dis}(x_i,Z_j)=\sum_{d=1}^{m}\mathrm{dis}_d(x_i,Z_j) \tag{16}$$

Example 6. In the dataset shown in Table 2, assume that cluster $C_1=\{x_1,x_2,x_3,x_5,x_7,x_{15}\}$ and its cluster Z_1 is {'d': 0.2069, 'b': 0.3793, 'c': 0.2069, 'a': 0.2069}, {'b': 0.5, 'a': 0.5}, {'f': 0.5, 'c': 0.5}, {'e': 0.6724, 'd': 0.3275}, 3.5556, and 14.1111. The dissimilarity between x_1 = {d, b, f, e, 7, 14} and Z_1 is $\mathrm{dis}(x_1,Z_1)=0.3818+0.5+0.3155+0.2489+11.8642+0.0123$ = 13.3227.

In 2016, Deb et al. [10] proposed a DSMI algorithm for imputing missing values in traffic accident data. The DSMI has two limitations:

- Because this algorithm focuses mainly on categorical attributes, how to handle numerical attributes is not clear.
- The computational cost of constructing graphs and calculating the weighted similarity measure is very high when dealing with high-dimensional datasets or datasets containing multiple missing values.

Our imputation method was based on the DSMI algorithm. However, we modified it to address the above limitations and then integrated the imputation step into the clustering step for the design of an algorithm that can cluster categorical or mixed data with missing values. To this end, we used two measures for the imputation step.

The first, the IS measure (ISM), is used to evaluate the degree of association between two sets of categorical values in a data object. The ISM was first introduced by Tan et al. [46]. It contains the product of two quantities (interest factor and support count) which compute the correlations between the values of different attributes in an object. In other words, the ISM considers both the interestingness and significance of a pattern. Another interpretation of the ISM is the geometric mean of the confidence of rules that can be generated from two items: $IS(A,B) = \sqrt{\text{Confidence}(A \Rightarrow B) \times \text{Confidence}(B \Rightarrow A)}$, in which $\text{Confidence}(A \Rightarrow B) = P(A,B)/P(A), P(A) = \#A/n, \#A$, and n are the number of data objects that contain A and the number of data objects in the dataset, respectively.

Definition 8 (*IS measure [10]*). A set T contains both complete and incomplete categorical objects. Let $A' = \{A'_1, A'_2, \ldots, A'_{m'}\}$ and $A'' = \{A''_1, A''_2, \ldots, A''_{m''}\}$ $(A', A'' \subset A; m', m'' < m)$ be two sets of categorical attributes that contain missing values and non-missing values in T, respectively. For all $a' = \{a'_1, a'_2, \ldots, a'_n\} \in A'_1 \times A'_2 \times \ldots \times A'_{m'}$ and $a'' = \{a''_1, a''_2, \ldots, a''_n\} \in A''_1 \times A''_2 \times \ldots \times A''_{m''}$, the IS measure between a' and a'' is defined as

$$\text{IS}(a', a'') = \frac{\text{Support}(a', a'')}{\sqrt{\text{Support}(a') \times \text{Support}(a'')}} \tag{17}$$

in which $\text{Support}(a', a'') = \frac{\#(a', a'')}{|T|}$, $\#(a', a'')$ is the number of mixed objects that contain both a' and a''.

The second measure is the missing-complete similarity measure (MCS). We extended the information-theoretic-based similarity measure in Eq. (13) to make it applicable for measuring the proximity of complete and incomplete objects.

Definition 9 (*Missing-complete similarity measure (MCS measure)*). The T set contains both complete and incomplete objects. Two categorical values o^j_{ld} and $o^j_{l'd}$ appear in T at the d^{th} attribute. The similarity between them is defined as

$$\text{sim}^{mis}_d(o^j_{ld}, o^j_{l'd}) = \begin{cases} \frac{2\log f_T(o^j_{ld}, o^j_{l'd})}{\log f_T(o^j_{ld}) + \log f_T(o^j_{l'd})} & \text{if } o^j_{ld} \neq ? \text{ and } o^j_{l'd} \neq ?, \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

in which $f_T(o^j_{ld}) = \frac{\#_T(o^j_{ld})}{n_T}$ and $\#_T(o^j_{ld})$ denotes the number of o^j_{ld} appearing in T, and n_T denotes the number of objects in T.

Let $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and $x_{i'} = (x_{i'1}, x_{i'2}, \ldots, x_{i'm})$ be the complete mixed object and incomplete mixed object, respectively. The MCS between x_i and $x_{i'}$ is then defined as

$$\text{MCS}(x_i, x_{i'}) = \sum_{d=1}^{m} \text{sim}^{mis}_d(x_{id}, x_{i'd}) \tag{19}$$

in which the d^{th} attributes of x_i and $x_{i'}$ are categorical attributes.

Example 7. This example illustrates how to impute missing values in categorical attributes using the IS and MCS measures. Table 3 shows the subset that contains the complete categorical objects extracted from Table 2.

Assume that the incomplete categorical object $x^c_6 = \langle d, b, f, ? \rangle$ in Table 2 is chosen for imputation. The DT that uses the attribute A^c_4 as the class attribute is built for x^c_6 based on the dataset in Table 3 (Fig. 3). The object x^c_6 is then assigned to leaf 7. The set of complete categorical objects that are correlated with x^c_6 are $\{x^c_1, x^c_2, x^c_3, x^c_7, x^c_{15}\}$. The set of categorical values in the complete attributes A^c_1, A^c_2, and A^c_3 contains $\{d, b, f\}, \{b, b, c\}, \{a, a, c\}$, and $\{b, a, f\}$. The set of categorical values in the incomplete attribute A^c_4 contains $\{e\}$. The possible imputed value is only e. Thus, it is chosen to impute the missing value in x^c_6—that is, $x^c_6 = \langle d, b, f, e \rangle$.

Next, assume that the incomplete categorical object $x^c_8 = \langle b, a, ?, ? \rangle$ in Table 2 is selected for imputation. Two values are missing in x^c_8 for attributes A^c_3 and A^c_4. Therefore, two DTs that use the attribute A^c_3 and A^c_4 as the class attributes are built for x^c_8 based on the dataset in Table 3 (Fig. 4), respectively. For the missing value in the attribute A^c_3, the object x^c_8 is assigned to leaf 8 of the tree 4a. The set of complete categorical objects that correlate with x^c_8 contains $\{x^c_5, x^c_{15}\}$. For the missing value in the attribute A^c_4, the object x^c_8 is assigned to leaf 1 of the tree 4b. The set of complete categorical objects that correlate with x^c_8 contains $\{x^c_2, x^c_3, x^c_{15}\}$. Because x^c_8 falls into multiple leaves, the objects from all these leaves are grouped into one collection; thus, the set of correlated objects is $\{x^c_2, x^c_3, x^c_5, x^c_{15}\}$. The set of categorical values in the complete attributes A^c_1 and A^c_2 contains $\{b, b\}, \{c, a\}$, and $\{b, a\}$, whereas the set of categorical values in the incomplete attribute A^c_3 and A^c_4 contain $\{c, e\}, \{f, d\}$, and $\{f, e\}$. The IS and MCS measures for each pair of categorical values in the complete attributes and incomplete attributes are IS ({b,b},{c,e}) = $\frac{0.5}{\sqrt{0.5 \times 0.5}}$ = 1, IS({c,a}, {f,d}) = $\frac{0.25}{\sqrt{0.25 \times 0.25}}$ = 1, IS({b,a}, {f,e}) = $\frac{0.25}{\sqrt{0.25 \times 0.25}}$ = 1, MCS(x^c_2, x^c_8) = MCS(x^c_3, x^c_8) = $\frac{2\log 0.8}{\log 0.8 + \log 0.8}$ + $\frac{2\log 1}{\log 0.4 + \log 0.6}$ + 0 + 0 = 1, MCS(x^c_5, x^c_8) = $\frac{2\log 1}{\log 0.2 + \log 0.8}$ + $\frac{2\log 0.6}{\log 0.6 + \log 0.6}$ + 0 + 0 = 1, and MCS(x^c_{15}, x^c_8) = $\frac{2\log 0.8}{\log 0.8 + \log 0.8}$ + $\frac{2\log 0.6}{\log 0.6 + \log 0.6}$ + 0 + 0 = 2. The affinity degree of possible imputed values was calculated by the average of the IS and MCS measures for each pair of categorical values in the complete and incomplete attributes: $\delta(\{c, e\})$ = (1 + 1)/2 = 1, $\delta(\{f, d\})$ = (1 + 1)/2 = 1, and $\delta(\{f, e\})$ = (1 + 2)/2 = 1.5. The actual imputed values were chosen by random sampling according to the affinity degrees. Specifically,

Table 3
The complete categorical objects.

	Attrs			
Objs	A_1^c	A_2^c	A_3^c	A_4^c
x_1^c	d	b	f	e
x_2^c	b	b	c	e
x_3^c	b	b	c	e
x_4^c	a	b	a	b
x_5^c	c	a	f	d
x_7^c	a	a	c	e
x_9^c	c	b	a	c
x_{11}^c	d	a	d	c
x_{12}^c	d	b	d	d
x_{14}^c	a	a	d	d
x_{15}^c	b	a	f	e

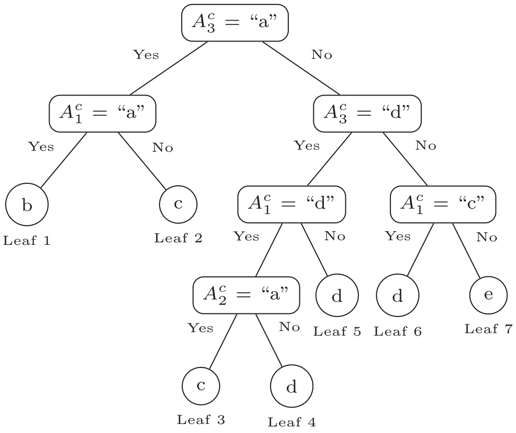


Fig. 3. Tree for the missing attribute A_4^c in x_6^c.

the sampling probabilities of $\{c, e\}, \{f, d\}$, and {f,e} are $0.2857, 0.2857$, and 0.4286, respectively. Thus, {f,e} has a probability of being chosen as the actual imputed values for x_8^c.

Based on these definitions, the problem of clustering for mixed numerical and categorical datasets with missing values aims to minimize the following objective function:

$$F(U, Z) = \sum_{j=1}^{k} \sum_{i=1}^{n} u_{i,j} \times dis(x_i, Z_j) \tag{20}$$

subject to

$$\begin{cases} \sum_{j=1}^{k} u_{i,j} = 1 & 1 \leqslant i \leqslant n \\ u_{i,j} \in \{0, 1\} & 1 \leqslant j \leqslant k,\ 1 \leqslant i \leqslant n \end{cases} \tag{21}$$

In this function, $U = [u_{i,j}]_{n \times k}$ is the partition matrix and $u_{i,j}$ takes a value of 1 if object x_i is in cluster C_j and 0 otherwise. By using the distance measure, we computed the partition matrix in Eq. (20) as

$$\begin{aligned} &\text{if } \text{dis}(x_i, Z_j) \leqslant \text{dis}(x_i, Z_{j'}) \text{ then} \\ u_{i,j} = 1,\ &\text{and } u_{i,j'} = 0,\ \text{for } 1 \leqslant j' \leqslant k, j \neq j' \end{aligned} \tag{22}$$

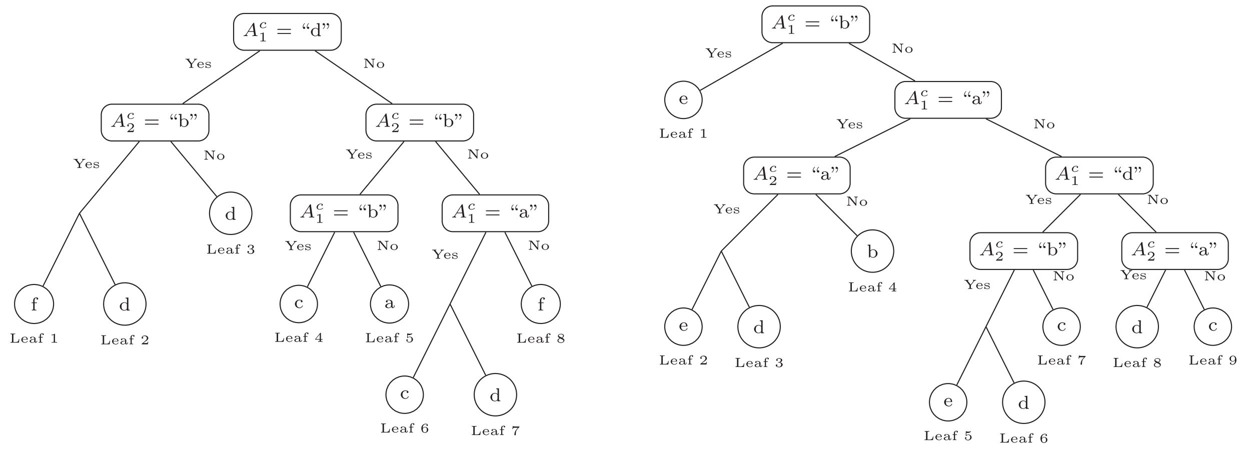
Further, cluster center Z_j is updated by using Eq. (9). Specifically, the cluster center at a numerical attribute is computed by the mean of this attribute for all objects in the cluster, whereas the center at a categorical attribute is computed by using the kernel-based method, as presented in Eq. (10).

4. Proposed *k*-CMM algorithm

Fig. 5 shows the general framework of the proposed *k*-CMM algorithm. In this flowchart, the *k*-CMM algorithm performs three phases: *initialization phase*, *imputation phase*, and *clustering phase*. In the initialization phase, a fully mixed dataset with missing values is divided into two subdatasets: complete dataset and missing dataset, denoted as S_1 and S_2, respectively. Each complete object in S_1 is then split into two subobjects based on the attribute type. Specifically, the algorithm separately extracts numerical values and categorical values from mixed objects and puts them into two subsets: S_1^r and S_1^c, respectively. The same step was applied to the missing objects in S_2. The numerical and categorical values are put into two subsets S_2^r and S_2^c, respectively. Next, the algorithm randomly initializes k clusters from the complete S_1 set. For every incomplete categorical object in the set S_2^c, denoted as x_i^c, the algorithm uses a decision-tree-based imputation approach to build a DT for each missing attribute of x_i^c. In particular, each missing attribute is considered as a class attribute to construct a DT from the complete categorical dataset S_1^c. The incomplete categorical object x_i^c is then assigned to the corresponding tree leaf that contains a set of correlated objects with x_i^c. When x_i^c is assigned to a leaf node, the missing values in this object are imputed using the possible imputed values from the correlated objects found in the leaf node. Also, the missing numerical object that has the same *id* as x_i^c, that is x_i^r in S_2^r, is imputed by using the *means* of values in the numerical attributes of the complete numerical objects in S_1^r that have the corresponding *ids* with those of correlated objects obtained in the previous step. Subsequently, the imputed objects x_i^r and x_i^c are merged to form a completely mixed numerical and categorical object, denoted as x_i. Note that the order of attributes of x_i after merging is the same as that of mixed objects in the original dataset S. The object x_i is then added to S_1, and x_i^r and x_i^c are eliminated from S_2^r and S_2^c, respectively. Next, the algorithm performs the clustering phase by assigning objects in S_1 into appropriate clusters and updating the centers of clusters. The imputation phase is repeated until S_2^c is empty. If the termination condition is not met, the algorithm performs the clustering phase until all the clusters are stable. Finally, it returns k clusters.

Algorithm 1. The *k*-CMM Algorithm

```
input  : $S$: a mixed dataset, $k$: a user-specified integer number to specify the number of clusters
output: $k$ clusters of $S$
Split $S$ into two subdatasets, $S = S_1$ (complete dataset) + $S_2$ (incomplete dataset)
Extract complete objects in $S_1$ into $S_1^r$ and $S_1^c$ that contain numerical and categorical objects,
  respectively
Extract missing objects in $S_2$ into $S_2^r$ and $S_2^c$ that contain numerical and categorical objects,
  respectively
Randomly initiate $k$ cluster centers from $S_1$ $Z^{(0)} = \{Z_1^{(0)}, \ldots, Z_k^{(0)}\}$
$U \leftarrow \emptyset$, $t = 0$
foreach object $x_i^c$ in $S_2^c$ do
    $IDList$, $x_i^c$ = Categorical_Imputation($x_i^c$, $S_1^c$)
    $x_i^r$ = Numerical_Imputation($x_i^r$, $S_1^r$, $IDList$)
    $x_i$ = merge $x_i^r$ and $x_i^c$
    $S_1 = S_1 \cup \{x_i\}$, $S_2^r = S_2^r \setminus \{x_i^r\}$, $S_2^c = S_2^c \setminus \{x_i^c\}$
    Keep $Z^{(t)}$ fixed, generate $U^{(t)}$ to minimize the distance between objects and cluster centers
      using Eq. (16)
    Keep $U^{(t)}$ fixed, update $Z^{(t)}$ using Eq. (9)
    $t = t + 1$
end
while clusters are not convergent do
    Keep $Z^{(t)}$ fixed, generate $U^{(t)}$ to minimize the distance between objects and cluster mode
      using Eq. (16)
    Keep $U^{(t)}$ fixed, update $Z^{(t)}$ using Eq. (9)
    $t = t + 1$
end
return $k$ clusters;
```



(a) Tree for the missing attribute A_3^c in x_8^c

(b) Tree for the missing attribute A_4^c in x_8^c

Fig. 4. Trees for the incomplete categorical object x_8^c.

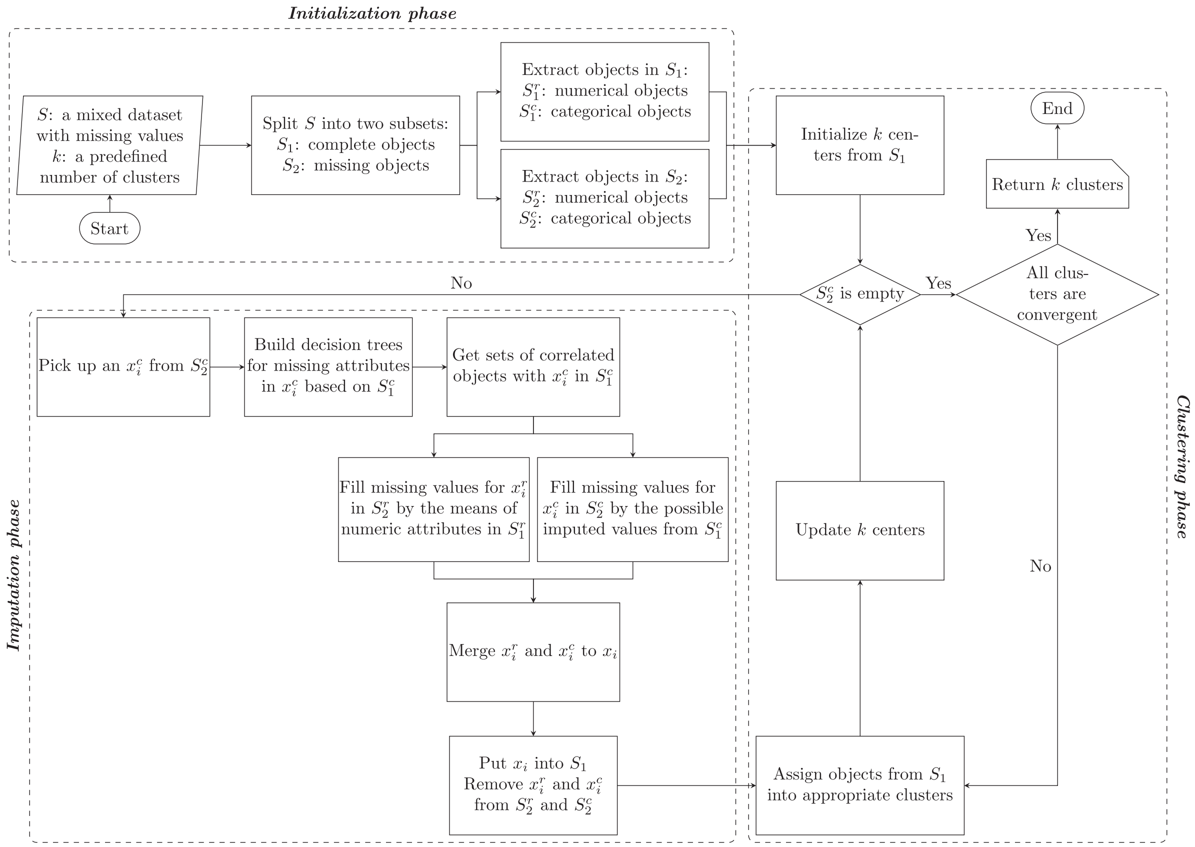

Initialization phase
S: a mixed dataset with missing values k: a predefined number of clusters
Start
Split S into two subsets: S_1: complete objects S_2: missing objects
Extract objects in S_1: S_1^r: numerical objects S_1^c: categorical objects
Extract objects in S_2: S_2^r: numerical objects S_2^c: categorical objects
Initialize k centers from S_1
End
Return k clusters
Yes
S_2^c is empty
Yes
All clusters are convergent
No
Pick up an x_i^c from S_2^c
Build decision trees for missing attributes in x_i^c based on S_1^c
Get sets of correlated objects with x_i^c in S_1^c
Fill missing values for x_i^r in S_2^r by the means of numeric attributes in S_1^r
Fill missing values for x_i^c in S_2^c by the possible imputed values from S_1^c
Merge x_i^r and x_i^c to x_i
Put x_i into S_1 Remove x_i^r and x_i^c from S_2^r and S_2^c
Update k centers
No
Assign objects from S_1 into appropriate clusters
Imputation phase
Clustering phase

Fig. 5. Flowchart of the *k*-CMM algorithm for mixed data with missing values.

Algorithm 2. Categorical_Imputation Procedure

```
input  : $x_i^c$: a missing categorical object, $S_1^c$: the set of complete categorical objects
output: $IDList$: the list of $id$ numbers of correlated objects with $x_i^r$, the imputed categorical
         object $x_i^c$
1  $A^c \leftarrow \emptyset$, $DTSet \leftarrow \emptyset$, $NodeSet \leftarrow \emptyset$, $O_i \leftarrow \emptyset$
2  Put attributes having missing values in $x_i^c$ to $A^c$
3  foreach attribute $A_i^c$ in $A^c$ do
4      if $DTSet$ does not contain decision tree $DT$ for $A_i^c$ then
5          Build a decision tree $DT$ that uses $A_i^c$ as class attribute from $S_1^c$
6          $DTSet \leftarrow DT$
7      end
8  end
9  foreach decision tree $DT$ in $DTSet$ do
10     Assign $x_i^c$ into leaf nodes of corresponding decision tree $DT$
11     $NodeSet \leftarrow$ chosen leaf nodes
12 end
13 Group objects in $NodeSet$ into one collection $T$
14 Find objects in $T$ that match with the maximum number of complete attribute(s) in $x_i^c$, and let
     N be the number of such objects, put their ids into $IDList$
15 for $i$ = 1 to $N$ do
16     $O_i \leftarrow$ possible imputed value(s) from the $i^{th}$ matched object
17     Calculate the IS measure by Eq. (17) for $O_i$
18     Calculate the MCS measure by Eq. (18) between the $i^{th}$ matched object and $x_i^c$
19     Calculate the affinity degree $\delta_i$ for $O_i$ based on IS and MCS
20 end
21 Impute missing value(s) for $x_i^c$ by using random sampling from the set of possible imputed
     values $\{O_1, \ldots, O_N\}$ based on the sampling probabilities specified by the set of affinity degrees
     $\{\delta_1, \ldots, \delta_N\}$
22 return $IDList$, $x_i^c$;
```

Algorithm 3. Numerical_Imputation Procedure.

```
input  : $x_i^r$: a missing numerical object, $S_1^r$: the set of complete numerical objects, $IDList$: the
         list of $id$ numbers of correlated objects with $x_i^r$
output: the imputed numerical object $x_i^r$
1  $A^r \leftarrow \emptyset$
2  Put attributes having missing values in $x_i^r$ to $A^r$
3  Get the set of correlated numerical objects in $S_1^r$ that have the $id$ in $IDList$, denoted as
     $CorrSet$
4  foreach attribute $A_i^r$ in $A^r$ do
5      Replace the missing value by mean of numerical values in $A_i^r$ of $CorrSet$, using Eq. (5)
6  end
7  return $x_i^r$;
```

Algorithm 1 shows the pseudo code of the k-CMM algorithm. It first scans and divides the mixed dataset S into two sub-datasets: namely, S_1 and S_2, which are the complete and incomplete datasets, respectively (line 1). It then extracts separately mixed objects in S_1 and S_2 to single-type objects. More specifically, complete numerical objects, complete categorical objects, missing numerical objects, and missing categorical objects are stored in S_1^r, S_1^c, S_2^r, and S_2^c, respectively (lines 2 and 3). Next, the algorithm randomly initiates k cluster centers from S_1 (line 4). Each center is represented by Eq. (9). Two variables, U and t, are created to store clusters and count the number of iterations of the clustering process (line 5). In the next step, k-CMM scans all objects in S_2^c to impute missing values inside these objects and assign them to clusters (lines 6–14). For each object x_i^c in S_2^c, the algorithm calls the Categorical_Imputation and Numerical_Imputation procedures to replace missing values in missing categorical objects and missing numerical objects, respectively.

Algorithm 2 shows the pseudo code of the Categorical_Imputation procedure. The input of this procedure is a missing categorical object x_i^c and a set of complete categorical objects S_1^c. First, it creates four variables—A^c, $DTSet$, $NodeSet$, and O_i to temporally store missing attributes, decision trees, sets of correlated objects, and sets of possible imputed values during the imputation process, respectively (line 1). It then searches for attributes that contain missing values (missing attributes for shortly) and puts them into set A^c (line 2). For every missing attribute A_i^c in A^c, the procedure checks whether a DT exists that uses A_i^c as the class attribute. If no such DT exists, the procedure builds a DT by using the missing attribute A_i^c as the class attribute and complete categorical objects from S_2^c to generate decision rules. In this study, we used $C4.5$ [38] to generate DTs. This process is performed until all missing attributes in A_i^c have their corresponding DTs (lines 3–8). The resulting DTs were then placed into the *DTSet*. After constructing the DTs, object x_i^c is assigned to the leaf node of the tree with decision rule(s) corresponding to the values of x_i^c. When x_i^c is assigned to the appropriate leaves, each leaf node consists of complete categorical objects from S_1^c that correlate with x_i^c. The resulting nodes are put into set *NodeSet* (lines 9–12). Each leaf node in *NodeSet* is represented as a list of objects. If x_i^c falls into multiple leaves, the procedure merges these leaves and group objects into one collection, T (line 13). For the next step, the procedure chooses complete objects in T that have the maximum number of complete attributes in common with x_i^c. Additionally, the *id* of the most closely correlated objects with x_i^c are placed in *IDList*. This list is later used in the Numerical_Imputation procedure (line 14). The categorical values in these selected objects corresponding to the missing attributes in x_i^c are then considered as the possible imputed values. The procedure then calculates the *IS* and *MCS* measures for possible imputed values in each complete object using Eqs. (17) and (18), respectively. Next, each list of possible imputed values is associated with the affinity degree given by the average of the *IS* and *MCS* values (lines 15–20). When the affinity degrees of possible imputed values are determined, the procedure assigns actual imputed values by using random sampling from the list of possible imputed values based on their affinity degrees (line 21). Finally, it returns *IDList* and complete x_i^c, in which all missing values are imputed (line 22).

Algorithm 3 shows the pseudo code of the Numerical_Imputation procedure. This procedure input is a missing numerical object x_i^r, the set of complete numerical objects S_1^r, and list of *ids* of correlated objects with x_i^r that is obtained in Algorithm 2. The procedure first finds attributes that contain missing values and puts them into set A^r (line 2). It then extracts a list of correlated objects with x_i^r based on the *ids* in *IDList* and puts these objects into *CorrSet* (line 3). For each missing attribute A_i^r in A^r, the procedure replaces the missing values in this attribute by using the mean of numerical values appearing at the same attribute of complete objects in *CorrSet*, through Eq. (5) (lines 4 to 6). Finally, it returns the complete object x_i^r, in which all missing values are imputed (line 7). After all missing values are imputed in the first algorithm, k-CMM merges x_i^r and x_i^c into a mixed complete object x_i so that the order of attributes of x_i is the same as that of objects in the original dataset (line 9). It then adds x_i to S_1 and removes x_i^r and x_i^c from S_2^r and S_2^c, respectively (line 10). In the next step, the k-CMM assigns objects in S_1 into appropriate clusters and updates the cluster centers (lines 11–13). The algorithm works in the same manner for all incomplete objects in S_2^c and S_2^r. If the termination condition is not met, the k-CMM performs the clustering phase until all clusters are stable (lines 15–19). Finally, it returns k clusters as the desired output (line 20).

We also designed a particular case of the k-CMM algorithm to make it applicable for clustering a missing categorical data. The framework for this task, shown in Fig. 6, was used to design the k-CCM algorithm [15]. Generally, the main flow of clustering missing categorical data resembles clustering missing mixed data, except that we changed methods to represent the centers of clusters, quantify the distances, and impute only for missing categorical values. Note that clustering for missing mixed data is nontrivial and more challenging than clustering for missing categorical data because different attributes in mixed data need to be treated heterogeneously. Thus, the framework for mixed data needs to be designed in a way that explicitly accounts for the underlying heterogeneity. It handles four main tasks: missing numerical data imputation, missing categorical data imputation, numerical data clustering, and categorical data clustering, whereas the framework for categorical data handles two tasks of clustering and imputation for categorical data.

4.1. Correctness of k-CMM

This section analyzes the correctness of the k-CMM algorithm to determine whether it accurately solves the problem of clustering a dataset with missing values after performing a finite number of processing steps. We use a *loop invariant* with three properties—initialization, maintenance, and termination—to verify the correctness of k-CMM. These properties are very similar to the concept of *mathematical induction*; to prove that a property holds, we need to define a base case and an inductive step. The specification of the k-CMM algorithm is as follows:

Precondition: S is a mixed dataset, k is a positive number.

Postcondition: The algorithm returns k clusters that satisfies the Def. 1 and all objects are imputed after a finite number of iterations.

Theorem 2. *k-CMM meets its specification.*

Proof. Assume that the precondition holds initially and that the Categorical_Imputation and Numerical_Imputation procedures return correct outputs; that is, categorical and numerical parts of a missing object are imputed correctly. We first show that this implies that when k-CMM terminates, each object in S is partitioned into one cluster and missing objects in S are imputed by either Algorithm 2 or 3, and thus, the postcondition is satisfied. In k-CMM (Algorithm 1), we consider $\sum_{j=1}^{k} u_{i,j} = 1$

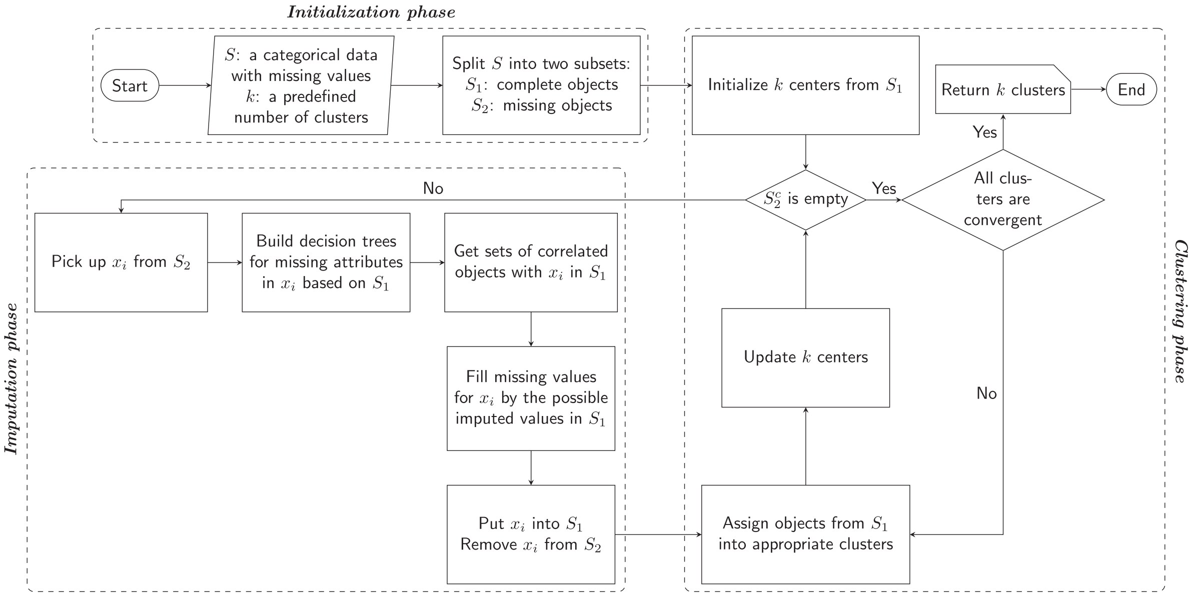

Initialization phase
Start
S: a categorical data with missing values
k: a predefined number of clusters
Split S into two subsets:
S_1: complete objects
S_2: missing objects
Initialize k centers from S_1
Return k clusters
End
Yes
S_2^c is empty
Yes
All clusters are convergent
No
Pick up x_i from S_2
Build decision trees for missing attributes in x_i based on S_1
Get sets of correlated objects with x_i in S_1
Fill missing values for x_i by the possible imputed values in S_1
Put x_i into S_1
Remove x_i from S_2
Update k centers
Assign objects from S_1 into appropriate clusters
No
Imputation phase
Clustering phase

Fig. 6. Flowchart of the k-CMM* algorithm for categorical data with missing values.

($1 \leqslant i \leqslant n$) of the partition matrix U is the loop invariant of the *for loop* (lines 6–14) and the *while loop* (lines 15–19), which are denoted as C_1 and C_2, respectively.

Initialization: We show that the loop invariant holds before the first loop iteration of C_1 and C_2. If the instructions in loops C_1 and C_2 are not executed, all objects are classified as belonging to the input dataset S, which means that the membership value of each object x_i in S is $\sum_{j=1} u_{i,j} = 1$. Therefore, the loop invariant holds before the first iteration of the loop.

Maintenance: Next, we show that each iteration preserves the loop invariant. If S_2^c is not empty, the instructions in loop C_1 are executed. Each missing object x_i is imputed and then assigned to its nearest cluster based on the distance between x_i and k cluster centers so that it satisfies the condition shown in Eq. (22). Each object in loop C_2 is assigned to the nearest cluster in the same manner until all clusters are stable; that is, the clusters formed in the current stage are the same as those formed in the previous stage. By contrast, if S_2^c is empty, instructions in loop C_2 are executed, and each instance is expected to belong to one and only one cluster. Thus, the loop invariant remains true before the next iteration of loops C_1 and C_2 for both cases.

Termination: In the *for* loop C_1, k-CMM traverses all objects in set S_2^c. Because the cardinality of S is finite, the cardinality of S_2^c is also finite; as a result, loop C_1 terminates after a finite number of iterations ($\leqslant n$). For the *while* loop C_2, similar to the k-means algorithm that almost always converges to a local minimum because the local variations of the loss function are conveniently bounded [6], k-CMM also terminates after a finite number of iterations. From observing that $\sum_{j=1}^{k} u_{i,j} = 1$ for all objects ($1 \leqslant i \leqslant n$), we conclude that Algorithm 1 is correct. The accuracy of the main workflow of the Categorical_Imputation procedure (Algorithm 2) has been discussed in [10]. Next, we prove the correctness of the Numerical_Imputation procedure (Algorithm 3) in the same manner as the k-CMM. Notice that the number of missing values in x_i^r is $m^r; m^r \geqslant 0$ is the loop invariant for the *for* loop.

Initialization: If object x_i contains no missing value in the numerical attributes, then $m^r = 0$; otherwise, $m^r > 0$. Therefore, the loop invariant holds before the first iteration of the loop.

Maintenance: Next, each iteration of the *for* loop preserves the loop invariant. If $m^r > 0$ in each iteration, the missing value in a numerical attribute is replaced by the *mean* of the corresponding numerical attribute of the set of correlated objects (lines 3–6). Therefore, at the end of the imputation, $m^r = m^r - 1$. The loop invariant holds after each iteration of the *for* loop until $m^r = 0$.

Termination: Each iteration decreases the value of m^r. Because the number of attributes in S is finite, the number of attributes with missing values is also finite. As a result, if the loop continues to iterate, m^r must eventually be no greater than 0. At this point, the loop breaks.

[2] https://www.jaist.ac.jp/iscenter/en/mpc/vpcc/.

4.2. Complexity analysis

In this section, we measure the computational complexity of the proposed algorithm with respect to Algorithms 1–3. The complexity of the entire process is the sum of the complexities of the three phases: initialization, imputation, and clustering. To approximate the complexity, we consider only the computation of the main steps in each phase. In the first phase, Algorithm 1 takes a mixed dataset as the input and then scans it once to divide the full data set into two subsets and extract mixed-type objects in each subset into single-type objects. The time requirement for this task is $\mathcal{O}(n \times m)$, in which n and m are the number of instances and features in the dataset, respectively. In the second phase, k-CMM performs Algorithms 2 and 3 to fill in the missing values for categorical and numerical attributes in each single-type missing object. m_{objs} denotes the number of missing objects in the dataset so that the number of complete objects is n -m_{objs}. The main task for Algorithm 2 is to generate a decision tree by using the $C4.5$ algorithm [38] for each missing attribute inside an incomplete object. The complexity of building a tree is $\mathcal{O}((n - m_{objs}) \times m^2)$ [45]. $\bar{m}_c$ and $\bar{m}_n$ denotes the average number of missing categorical and numerical attributes in the incomplete dataset, respectively. The complexity of Algorithm 2 is $\mathcal{O}(\bar{m}_c \times (n - m_{objs}) \times m^2)$. Algorithm 3 has a linear complexity with respect to the number of missing numerical attributes for each incomplete object, $\mathcal{O}(\bar{m}_n)$. Therefore, the complexity of the second phase is $\mathcal{O}(m_{objs} \times (\bar{m}_c \times (n - m_{objs}) \times m^2 + \bar{m}_n))$. For the clustering phase, the algorithm requires $\mathcal{O}(k \times n)$ to assign n instances to k clusters. For each assignment, $\mathcal{O}(m \times t)$ is required to estimate the dissimilarity between each object and a cluster center, in which t is the average number of categories in each attribute. The complexity of the assignment step is $\mathcal{O}(k \times n \times m \times t)$. Further, to update k cluster centers, the algorithm requires $\mathcal{O}(k \times m \times t)$. Thus, the complexity of the clustering phase for an iteration is $\mathcal{O}(k \times n \times m \times t)$ + $\mathcal{O}(k \times m \times t)$ = $\mathcal{O}(k \times m \times t \times (n+1)) \approx \mathcal{O}(k \times n \times m \times t)$. Assume that k-CMM requires $\bar{n}$ iterations to satisfy the convergence; then, it takes $\mathcal{O}(\bar{n} \times k \times n \times m \times t)$. In general, the complexity of k-CMM is $\mathcal{O}(n \times m + m_{objs} \times (\bar{m}_c \times (n - m_{objs}) \times m^2 + \bar{m}_n) + \bar{n} \times k \times n \times m \times t)$ = $\mathcal{O}(m_{objs} \times (\bar{m}_c \times (n - m_{objs}) \times m^2 + \bar{m}_n) + n \times m \times (\bar{n} \times k \times t + 1)) \approx \mathcal{O}(m_{objs} \times (\bar{m}_c \times (n - m_{objs}) \times m^2 + \bar{m}_n) + \bar{n} \times k \times n \times m \times t)$.

5. Comparative experiment

Experiments were performed to evaluate the performance of the proposed k-CMM on a PC Cluster.[2] Each node is equipped with an Intel Xeon Gold 6130 2.1 GHz (16 cores ×2), with 64 GB of RAM, running CentOS 7.2. The proposed algorithm was implemented in Python. The source code and datasets are provided at https://goo.gl/twPGZX. The performance of the k-CMM algorithm is compared with that of the k-prototypes algorithm [20] and two other tandem versions, denoted as k-prototypes^{tdm} and k-CMM^{tdm}, respectively. The two tandem versions preprocess the input dataset by replacing the missing values in the numerical and categorical attributes by the *mean* and *mode*, respectively, before performing the clustering step. Additionally, to evaluate the effectiveness of the imputation method, the particular case of the k-CMM algorithm for clustering pure categorical data with missing values, namely, k-CMM*, is compared with five partitional clustering algorithms: k-modes [19], k-representatives [40], Modified 1, Modified 2, and Modified 3 [29]. Generally, the performances of partitional clustering algorithms and the proposed k-CMM algorithm can be influenced by two factors: initialization methods used in the algorithms and the choice of the number of clusters (k). For the first factor, the above algorithms use a random initialization method to generate the initial clusters. This simple method is widely discussed in other studies. However, it can produce different results for different algorithms. Among these results, low-clustering quality can occur [16]. For this reason, each algorithm ran 10 trials for each dataset. The overall performance was then calculated by averaging the results of all the trials. For the second factor, the inaccurate estimation of k can affect the quality of the clustering results [14]. For this reason, in the experiment, we set k as the number of classes in the ground-truth labels.

5.1. Datasets

Algorithm performances were compared on 12 datasets, including 4 pure categorical datasets and 8 mixed datasets with missing values. Tables 4 and 5 lists the characteristics of these datasets, while Fig. 7 shows the cumulative sum of missing values in each dataset. These real-life datasets have various characteristics obtained from the UCI Machine Learning Repository [3]. Note that the number of attributes of each dataset shown in Table 2 is the sum of the number of numerical attributes and the number of categorical attributes (excluding the class attribute). By using these datasets, the performance of the k-CMM algorithm was evaluated for different types of data encountered in real-life applications. We used four external metrics—purity, normalized mutual information (NMI), homogeneity, and completeness measures—to evaluate the quality of the clustering results. These metrics use class information in the original datasets as the ground truth, as well as clustering results generated by an algorithm to measure how well the output clusters match the ground truth [26]. In particular, we omitted the class attributes in the datasets during the clustering process and used them only when evaluating the clustering results. Consider a mixed dataset S with n data objects. Let $C = \{C_1, \ldots, C_k\}$ be the set of clusters generated by a clustering algorithm from S, in which k is

[2] https://www.jaist.ac.jp/iscenter/en/mpc/vpcc/.

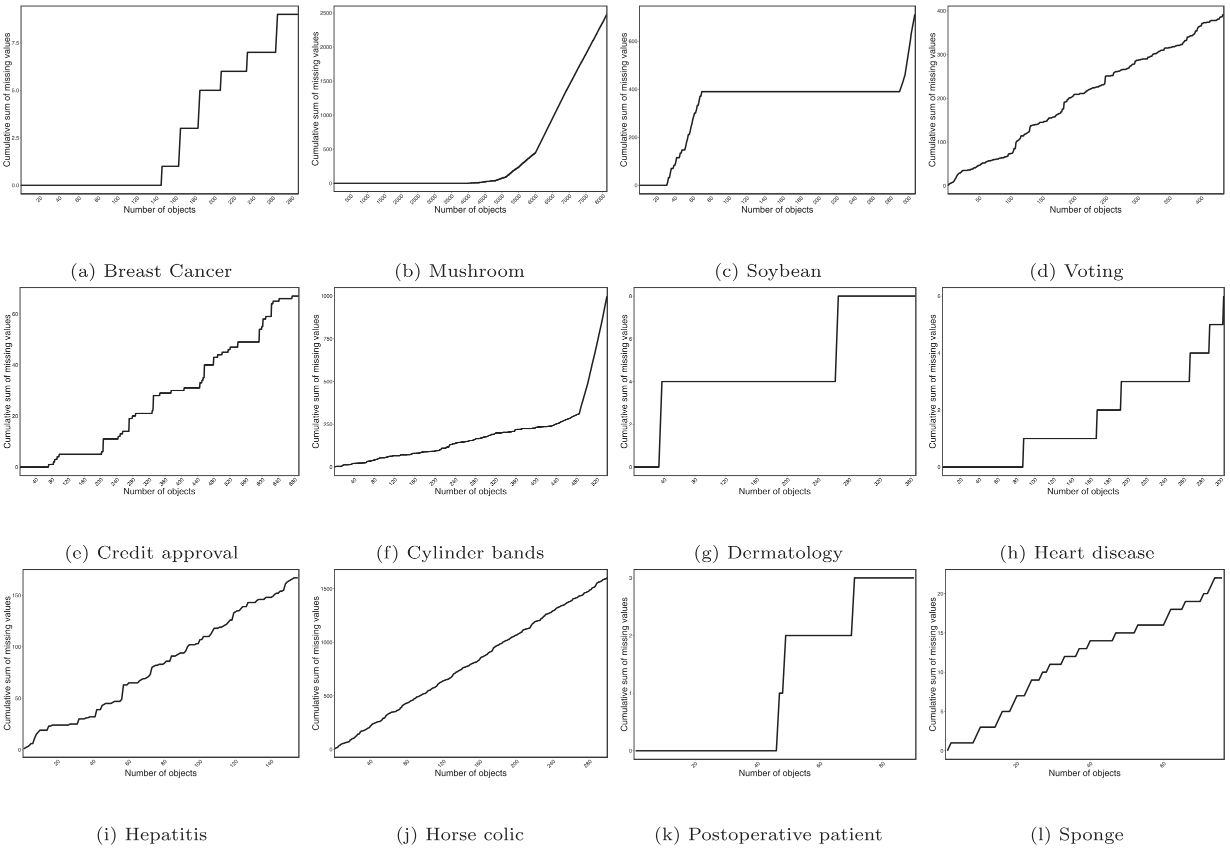
[3] http://archive.ics.uci.edu/ml

Table 4
List of notations for dataset.

#objs	≜	number of objects
#nAttrs	≜	number of numerical attributes
#cAttrs	≜	number of categorical attributes
#mObjs	≜	number of missing objects
#mAttrs	≜	number of missing attributes
#mValues	≜	number of missing values

Table 5
Characteristics of the datasets.

#	Dataset	#objs	#nAttrs	#cAttrs	#mObjs	#mAttrs	#mValues	#classes	type
1	Breast cancer	286	0	9	9	2	9	2	cat
2	Mushroom	8,124	0	22	2,480	1	2,480	2	cat
3	Soybean	307	0	35	41	34	712	19	cat
4	Voting	435	0	16	203	16	392	2	cat
5	Credit approval	690	6	9	37	7	67	2	mixed
6	Cylinder bands	540	20	19	263	28	999	2	mixed
7	Dermatology	366	33	1	8	1	8	6	mixed
8	Heart disease	303	5	8	6	2	6	5	mixed
9	Hepatitis	155	6	13	75	16	167	2	mixed
10	Horse colic	299	11	293	19	15	1,602	2	mixed
11	Post patient	90	1	7	3	1	3	3	mixed
12	Sponge	76	3	42	22	1	22	2	mixed



(a) Breast Cancer
(b) Mushroom
(c) Soybean
(d) Voting
(e) Credit approval
(f) Cylinder bands
(g) Dermatology
(h) Heart disease
(i) Hepatitis
(j) Horse colic
(k) Postoperative patient
(l) Sponge

Fig. 7. Cumulative sum of missing values in datasets.

Table 6
Comparing the average purity results between k-CMM and k-prototypes.

#	Dataset	k-prototypes	k-prototypes$^{\text{tdm}}$	k-CMM	k-CMM$^{\text{tdm}}$
1	Credit approval	0.5620	0.5620	**0.6581**	0.5620
2	Cylinder bands	**0.6940**	0.6840	0.6678	0.5780
3	Dermatology	**0.3590**	0.3490	0.3060	0.3540
4	Heart disease	0.5410	0.5410	**0.5480**	0.5410
5	Hepatitis	**0.7940**	**0.7940**	**0.7940**	**0.7940**
6	Horse colic	0.6350	0.6350	**0.6850**	0.6350
7	Postoperative patient	0.7176	**0.7209**	0.7130	0.7110
8	Sponge	**0.9210**	**0.9210**	**0.9210**	**0.9210**

Table 7
Comparing the average purity results between k-CMM* and five other algorithms for categorical data.

#	Dataset	k-modes	k-representatives	Modified 1	Modified 2	Modified 3	k-CMM*
1	Breast Cancer	0.7028	0.7028	0.7098	0.7028	0.7028	**0.7133**
2	Mushroom	0.8578	**0.8876**	0.8861	0.7078	0.7365	0.8858
3	Soybean	0.6227	0.7095	0.7091	0.7293	0.7202	**0.7687**
4	Voting	0.8581	0.8764	0.8713	0.8760	0.8775	**0.8805**

the number of clusters in C. Let $P = \{P_1, \ldots, P_{k'}\}$ be the set of partitions inferred by the original class information in S, in which k' is the number of classes in P. The boldfaced numbers in the following tables show the best performance among the compared algorithms for each dataset.

5.2. Purity

Eq. (23) shows the purity formulation. Specifically, we first assign each cluster to the class, which is the most common cluster. The accuracy of this assignment, in other words, the purity value, is calculated by counting the number of correctly assigned objects and dividing by n. The purity values range from zero (worst clustering) to one (best clustering).

$$\text{Purity}(C,P) = \frac{1}{n}\sum_{j}\max_{j'}|C_j \cap P_{j'}|(1 \leqslant j \leqslant k, 1 \leqslant j' \leqslant k') \tag{23}$$

Here, C_j and $P_{j'}$ are interpreted as the set of data objects in cluster C_j and in class $P_{j'}$, respectively.

Table 6 shows the purity results of k-CMM and compared algorithms. Notice that k-CMM outperforms the other algorithms on Credit approval, Heart disease and Horse colic datasets. The k-prototypes outperforms the other algorithms on Cylinder bands and Dermatology datasets, whereas k-prototypes$^{\text{tdm}}$ obtains the highest score on the Postoperative patient dataset. The four algorithms obtain the same purity results on Hepatitis and Sponge datasets. In general, k-CMM works well on datasets that contain many missing values, such as Horse colic. k-CMM yields better purity results than those of other algorithms on datasets that contain missing values on both numerical and categorical attributes, such as Credit approval, Hepatitis, and Horse colic, or datasets that contain missing values only on categorical attributes, such as Heart disease and Sponge datasets. On Dermatology and Postoperative patient datasets, the purity results of k-prototypes are higher than for k-CMM because the missing values appear only on numerical attributes, and these datasets contain fewer missing values. For each dataset, if no missing values appear on categorical attributes, missing values in numerical attributes are imputed by the means of those numerical attributes. Table 7 lists the purity results of k-CMM* and the five compared algorithms. Notice that k-CMM* outperforms the compared algorithms on the Breast cancer, Soybean, and Voting datasets, whereas the k-representatives obtains the highest score on the Mushroom dataset. In general, the imputation method for categorical attributes can improve the performance of the proposed clustering algorithm.

5.3. Normalized mutual information

The mutual information of two random variables expresses their mutual dependence or the amount of information they have in common. In other words, it measures the extent to which one of these variables reduces the uncertainty about the other [37]. Normalized mutual information (NMI) is another external evaluation criterion for measuring the clustering quality. X and Y are used as the random variables described by the set of clusters C and the set of classes P, respectively. Let $I(X,Y)$ denote the mutual information between X and Y and $H(X)$ denote the entropy of X. The NMI was defined in [44].

$$\text{NMI}(X,Y) = \frac{I(X,Y)}{\sqrt{H(X),H(Y)}} \tag{24}$$

this equation can be estimated for C and P as:

Table 8
Comparing the average NMI results between *k*-CMM and *k*-prototypes.

#	Dataset	*k*-prototypes	*k*-prototypes^{tdm}	*k*-CMM	*k*-CMM^{tdm}
1	Credit approval	0.0160	0.0160	**0.1003**	0.0160
2	Cylinder bands	**0.2140**	0.1950	0.0000	0.0130
3	Dermatology	**0.1090**	0.1060	0.0100	0.1070
4	Heart disease	0.0560	0.0380	**0.0593**	0.0400
5	Hepatitis	0.0041	0.0020	**0.0127**	0.0020
6	Horse colic	0.0110	0.0110	**0.0214**	0.0110
7	Postoperative patient	0.0335	0.0294	**0.0369**	0.0110
8	Sponge	**0.0150**	**0.0150**	0.0000	0.0000

Table 9
Comparing the average NMI results between *k*-CMM* and other five algorithms for categorical data.

#	Dataset	*k*-modes	*k*-representatives	Modified 1	Modified 2	Modified 3	*k*-CMM*
1	Breast Cancer	0.0035	0.0018	0.0576	0.0027	0.0027	**0.0606**
2	Mushroom	0.5036	0.5383	0.5310	0.1920	0.2001	**0.5492**
3	Soybean	0.6276	0.7495	0.7413	0.7427	0.7513	**0.7600**
4	Voting	0.4369	0.4954	0.4950	0.5044	0.5044	**0.5080**

$$\mathrm{NMI}(C,P) = \frac{\sum_{j=1}^{k}\sum_{j'=1}^{k'}|C_j \cap P_{j'}|\log\frac{n|C_j \cap P_{j'}|}{|C_j||P_{j'}|}}{\sqrt{\sum_{j=1}^{k}|C_j|\log\frac{|C_j|}{n}\sum_{j'=1}^{k'}|P_{j'}|\log\frac{|P_{j'}|}{n}}} \tag{25}$$

The NMI is always between 0 and 1. It reaches its maximum value of 1 only when the 2 sets of labels have a perfect one-to-one correspondence.

The performances of the algorithms were compared in terms of NMI for the 12 datasets. Tables 8 and 9 present the comparative results. In particular, *k*-CMM outperforms the other algorithms on the Credit approval, Heart disease, Hepatitis, Horse colic, and Postoperative patient datasets. *K*-prototypes outperforms others on the Cylinder bands, Dermatology, and Sponge datasets. *K*-CMM* for pure categorical data outperformed the other five algorithms. Note that *k*-CMM works very well on pure categorical datasets. For mixed datasets in which missing values appear on many categorical attributes, such as Credit approval and Hepatitis, the NMI results of the proposed algorithms are improved significantly.

5.4. Homogeneity and completeness measure

The quality of the clustering results is evaluated further by using homogeneity and completeness measures [39]. A perfect homogeneity case is obtained when a clustering assigns only those data instances that are members of a single class to a single cluster. In this situation, the class distribution within each cluster should be skewed to a single class. Conversely, a perfect completeness case is obtained when a clustering assigns all the data instances that are members of a single class to a single cluster. As a result, each of these distributions is completely skewed to a single cluster. By calculating homogeneity and completeness separately, a more precise evaluation of the performance of the clustering can be obtained [39]. Mathematically, the homogeneity measure can be defined as

$$h = 1 - \frac{H(C|P)}{H(C)} \tag{26}$$

in which $H(C|P)$ is the conditional entropy of the class distribution concerning the proposed clustering, which is defined as follows:

$$H(C|P) = -\sum_{j=1}^{k}\sum_{j'=1}^{k'}\frac{a_{j'j}}{n}log\frac{a_{j'j}}{\sum_{j'=1}^{k'}a_{j'j}} \tag{27}$$

$H(C)$ is the maximum reduction in entropy provided by the clustering information; that is when $H(C|P)$ is maximal:

$$H(C) = -\sum_{j'=1}^{k'}\frac{\sum_{j=1}^{k}a_{j'j}}{n}log\frac{\sum_{j=1}^{k}a_{j'j}}{n} \tag{28}$$

In the above equations, $a_{j'j}$ is the number of data instances that are members of class $P_{j'}$ and elements of cluster C_j.

The completeness is symmetrical to homogeneity and can be defined as

Table 10
Comparing the average (homogeneity, completeness) results between *k*-CMM and *k*-prototypes.

#	Dataset	*k*-prototypes	*k*-prototypes^{tdm}	*k*-CMM	*k*-CMM^{tdm}
1	Credit approval	0.009, 0.137	0.009, 0.137	**0.142**, **0.142**	0.009, 0.137
2	Cylinder bands	**0.164**, **0.309**	0.150, 0.281	0.000, 0.000	0.013, 0.013
3	Dermatology	**0.110**, **0.107**	0.107, 0.105	0.006, 0.027	0.108, 0.106
4	Heart disease	0.047, 0.043	0.036, 0.033	**0.051**, **0.046**	0.039, 0.036
5	Hepatitis	0.002, 0.003	0.002, 0.003	**0.016**, **0.013**	0.001, 0.002
6	Horse colic	0.009, 0.016	0.009, 0.016	**0.026**, **0.025**	0.009, 0.016
7	Postoperative patient	0.030, 0.030	0.030, 0.031	**0.031**, **0.037**	0.010, 0.013
8	Sponge	**0.027**, 0.011	**0.027**, 0.011	0.000, 0.000	0.000, **1.000**

Table 11
Comparing the average (homogeneity, completeness) results on categorical data.

#	Dataset	*k*-modes	*k*-representatives	Modified 1	Modified 2	Modified 3	*k*-CMM*
1	Breast Cancer	0.004, 0.004	0.002, 0.002	0.055, 0.061	0.003, 0.002	0.006, 0.005	**0.093**, **0.104**
2	Mushroom	0.486, 0.509	0.530, 0.547	0.523, 0.540	0.220, 0.226	0.194, 0.195	**0.540**, **0.556**
3	Soybean	0.647, 0.609	0.752, 0.734	0.741, 0.742	0.736, 0.750	0.750, 0.753	**0.764**, **0.756**
4	Voting	0.444, 0.430	0.504, 0.486	0.504, 0.486	0.514, 0.495	0.514, 0.495	**0.518**, **0.498**

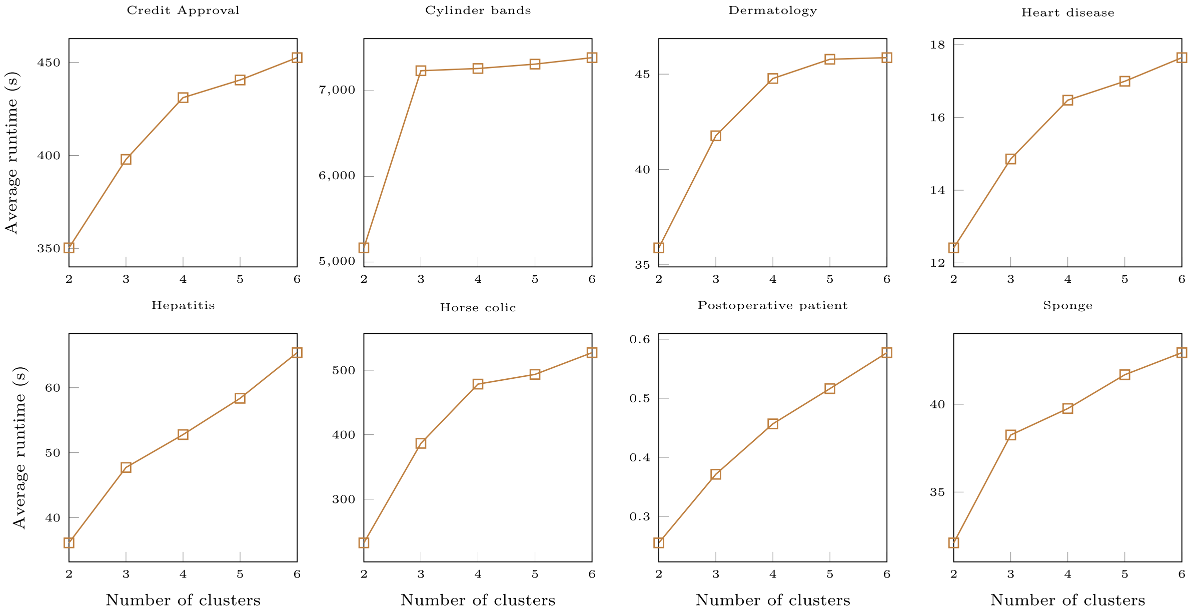


Fig. 8. Average runtimes for various number of clusters.

$$c = 1 - \frac{H(P|C)}{H(P)} \tag{29}$$

in which $H(P|C)$ is the conditional entropy of the proposed cluster distribution, given the class of the component data instances.

$$H(P|C) = -\sum_{j'=1}^{k'}\sum_{j=1}^{k}\frac{a_{j'j}}{n}log\frac{a_{j'j}}{\sum_{j=1}^{k}a_{j'j}} \tag{30}$$

Here, $H(P)$ is defined as

$$H(P) = -\sum_{j=1}^{k}\frac{\sum_{j'=1}^{k'}a_{j'j}}{n}log\frac{\sum_{j'=1}^{k'}a_{j'j}}{n} \tag{31}$$

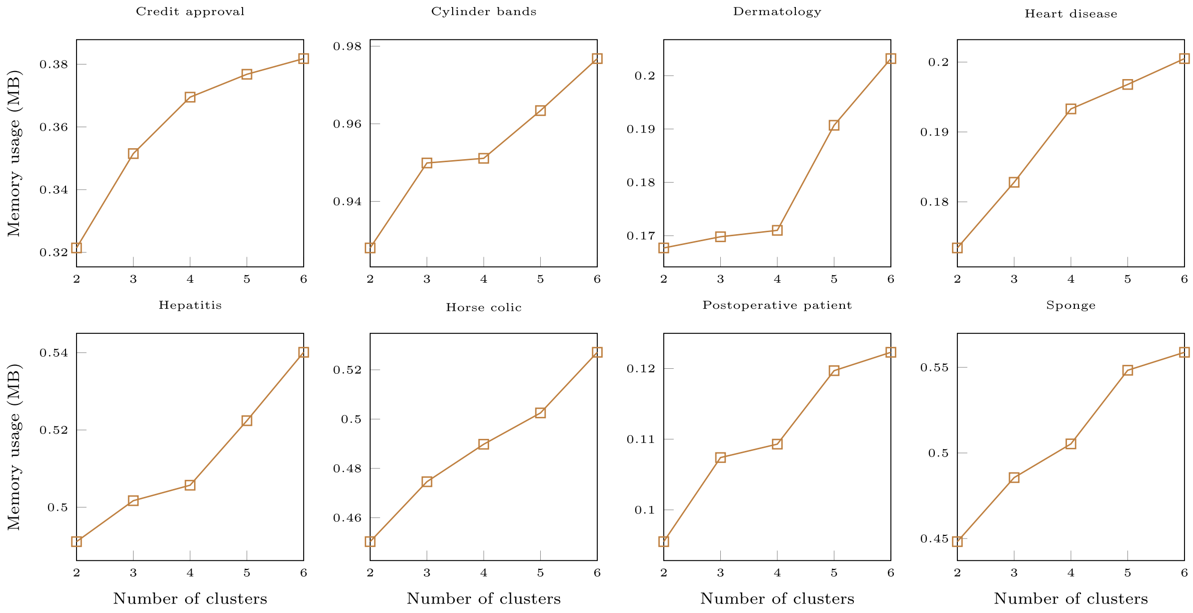


Fig. 9. Average memory usage for various number of clusters.

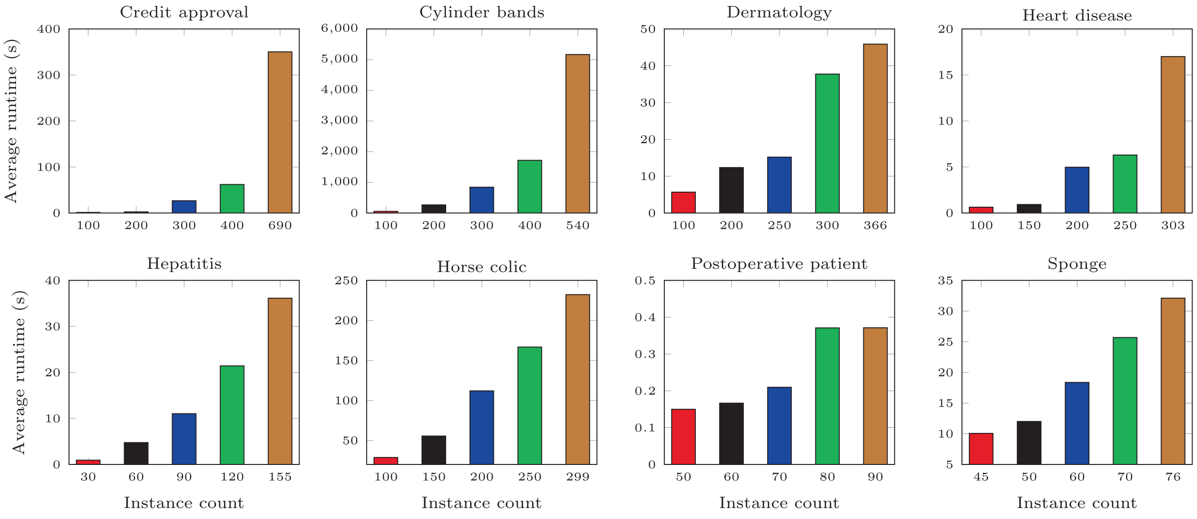


Fig. 10. Average runtimes when varying the database size.

We compared the performances of the k-CMM and other algorithms in terms of homogeneity and completeness for both categorical and mixed datasets. Tables 10 and 11 present the results. Notice that k-CMM outperforms the other algorithms on Credit approval, Heart disease, Hepatitis, Horse colic, and Postoperative patient datasets. The k-prototypes outperforms the others on the cylinder bands and Dermatology datasets. On the Sponge dataset, the k-prototypes and k-prototypes$^{\mathrm{tdm}}$ obtains the same homogeneity score, whereas k-CMM$^{\mathrm{tdm}}$ obtains the highest completeness score among the cases. The k-CMM* algorithm also outperforms the other five algorithms on four categorical datasets. Notice that the homogeneity and completeness results of k-CMM on datasets, in which missing values appear on many categorical attributes such as Credit approval and Hepatitis, are much higher than those of k-prototypes and k-prototypes$^{\mathrm{tdm}}$. Generally, the imputation method in combination with the KDE and the information-theoretic-based dissimilarity measure have facilitated the clustering process and enhanced the quality of the clustering results.

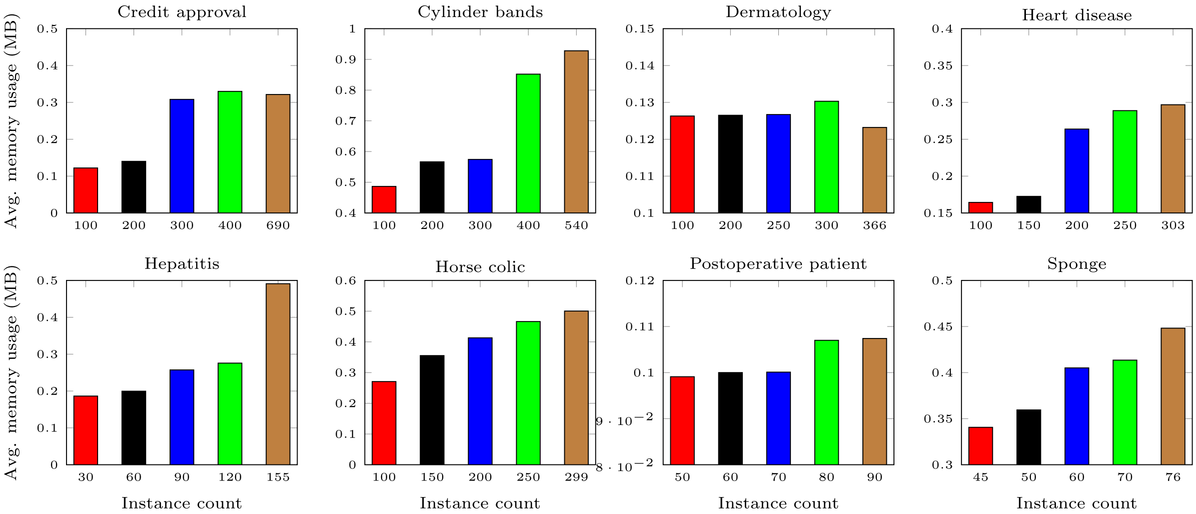


Fig. 11. Average memory usage when varying the database size.

5.5. Execution time and memory usage

Because the *k*-prototypes ran faster than the *k*-CMM, in this section, we measured only the average execution time of *k*-CMM for various numbers of clusters on 10 mixed datasets. Fig. 8 shows the results. In each figure, the vertical and horizontal axes represent the execution time in seconds and the number of clusters, respectively.

For all datasets, when the number of clusters increased, the time required for clustering also increased. For the Credit approval dataset, *k*-CMM took an average of 350.2538 (s), 397.8415 (s), 431.0505 (s), 440.5416 (s), and 452.5545 (s) for 2, 3, 4, 5, and 6 clusters, respectively. On the Cylinder bands dataset, *k*-CMM took an average of 5,165.4505 (s), 7,233.9648 (s), 7,259.905 (s), 7,310.9494 (s), and 7386.4848 (s) for 2, 3, 4, 5, and 6 clusters, respectively. Similar results were observed for other datasets.

The average memory usage of the proposed *k*-CMM algorithm was also evaluated for various numbers of clusters on 10 mixed datasets. Fig. 9 shows the results in terms of memory usage in megabytes (vertical axes) for various numbers of clusters (horizontal axes). In general, memory usage increased when the number of clusters increased in all cases. For the Credit approval dataset, *k*-CMM averaged 0.3214 (MB), 0.3515 (MB), 0.3695 (MB), 0.3768 (MB), and 0.3818 (MB) for 2, 3, 4, 5, and 6 clusters, respectively. On the Cylinder bands dataset, *k*-CMM averaged 0.928 (MB), 0.9499 (MB), 0.9511 (MB), 0.9634 (MB), and 0.9768 (MB) for 2, 3, 4, 5, and 6 clusters, respectively. Similar results were observed for the other datasets.

5.6. Scalability

Another experiment assessed the scalability of the *k*-CMM algorithms. The execution times of the *k*-CMM were measured while varying the number of instances on 10 mixed datasets. The results in Fig. 10 show that the vertical axes indicate the average execution times in seconds, whereas the horizontal axes represent the number of instances used. Generally, the *k*-CMM had linear scalability for all the datasets. With regard to the Credit approval dataset, *k*-CMM averaged 1.2233 (s), 2.4053 (s), 26.6075 (s), 61.9798 (s), and 350.2538 (s) for 100, 200, 300, 400, and 690 instances, respectively. Similar results were observed for the other datasets.

Additionally, the memory usage of the *k*-CMM was also measured on the mixed dataset while varying the number of instances. The results in Fig. 11 show that the vertical axes indicate the average memory usage in megabytes, whereas the horizontal axes represent the number of instances used. Generally, the *k*-CMM also exhibited linear scalability for all datasets. For the Credit approval dataset, *k*-CMM averaged 0.1221 (MB), 0.14 (MB), 0.3081 (MB), 0.3298 (MB), and 0.3214 (MB) for 100, 200, 300, 400, and 690 instances, respectively. Similar results were observed for the other datasets.

6. Conclusion

Missing values are commonly observed in datasets and can significantly affect the efficacy of the research study. Many mixed numerical and categorical datasets in real-life applications can contain missing values. Further, clustering is one of the most popular tasks in data mining, and clustering mixed datasets into meaningful groups is practically useful because frequently encountered objects in real-life datasets are mixed objects. This paper addresses these two issues by proposing an algorithm called *k*-CMM for clustering mixed numerical and categorical datasets with missing values. It integrates the imputation and clustering steps into a common framework to improve the clustering results. In the imputation step, the decision-tree-based method was first used to find the set of correlated objects. It then uses the IS and MCS measures to search for possible imputed values from the correlated set to impute missing values in categorical attributes. The missing values in the numerical attributes are imputed using the mean of the corresponding attributes from the correlated set. In the clustering step, the *k*-CMM uses the kernel density estimation approach and the mean to define the cluster centers for numerical and categorical attributes, respectively. Additionally, to quantify the proximity between data objects, it uses the squared Euclidean and information-theoretic-based dissimilarity measure for numerical and categorical attributes, respectively. Experimental results have shown that the *k*-CMM is more efficient than the *k*-prototypes and tandem versions in terms of clustering quality in most cases. Specifically, using *k*-CMM for clustering pure categorical datasets was also evaluated with the other five state-of-the-art clustering algorithms in terms of clustering quality. The results indicate that the decision-tree-based method and measures used for categorical attributes can enhance the clustering results. Generally, *k*-CMM has a comparative performance in terms of purity, NMI, homogeneity, and completeness. Additionally, we evaluated the runtime, memory consumption, and scalability of the *k*-CMM. The results show that the *k*-CMM is scalable with respect to the number of instances. In future work, we will design a parallel clustering algorithm for large-scale and high-dimensional mixed datasets with missing values and apply the proposed method to other data mining tasks [22,17].

CRediT authorship contribution statement

Duy-Tai Dinh: Data curation, Methodology. **Van-Nam Huynh:** Methodology, Writing - review & editing, Funding acquisition. **Songsak Sriboonchitta:** Writing - review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors are very grateful to the Editor-in-chief and reviewers for their valuable comments and suggestions. This paper is supported in part by the Centre of Excellence in Econometrics, Faculty of Economics, Chiang Mai University, Thailand and by the Air Force Office of Scientific Research/Asian Office of Aerospace Research and Development (AFOSR/AOARD) under award number FA2386-17-1-4046.

References

[1] C.C. Aggarwal, An introduction to cluster analysis, in: Data Clustering: Algorithms and Applications, 2013, pp. 1–28.
[2] J. Aitchison, C.G. Aitken, Multivariate binary discrimination by the kernel method, Biometrika 63 (1976) 413–420.
[3] M.R. Anderberg, Cluster Analysis for Applications. Probability and Mathematical Statistics: A Series of Monographs and Textbooks, 1973.
[4] P. Berkhin, A survey of clustering data mining techniques, Grouping Multidimensional Data (2006) 25–71.
[5] S. Boluki, S.Z. Dadaneh, X. Qian, E.R. Dougherty, Optimal clustering with missing values, BMC Bioinformatics 20 (2019) 321.
[6] L. Bottou, Y. Bengio, Convergence properties of the k-means algorithms, in: Advances in Neural Information Processing Systems, 1995, pp. 585–592.
[7] A. Cena, M. Gagolewski, Genie+ owa: Robustifying hierarchical clustering with owa-based linkages, Information Sciences 520 (2020) 324–336.
[8] L. Chen, S. Wang, Central clustering of categorical data with automated feature weighting, in, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013, pp. 1260–1266.
[9] J.Y. Chen, H.H. He, A fast density-based data stream clustering algorithm with cluster centers self-determined for mixed data, Information Sciences 345 (2016) 271–293.
[10] R. Deb, A.W.C. Liew, Missing value imputation for the analysis of incomplete traffic accident data, Information sciences 339 (2016) 274–289.
[11] S. Laohakiat, V. Sa-ing, An incremental density-based clustering framework using fuzzy local clustering, Information Sciences 547 (2021) 404–426.
[12] Y. Ma, H. Lin, Y. Wang, H. Huang, X. He, A multi-stage hierarchical clustering algorithm based on centroid of tree and cut edge constraint, Information Sciences 557 (2021) 194–219.
[13] W.B. Xie, Y.L. Lee, C. Wang, D.B. Chen, T. Zhou, Hierarchical clustering supported by reciprocal nearest neighbors, Information Sciences 527 (2020) 279–292.
[14] D.T. Dinh, T. Fujinami, V.N. Huynh, Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient, in: International Symposium on Knowledge and Systems Sciences, 2019, pp. 1–17.
[15] D.T. Dinh, V.N. Huynh, k-CCM: A center-based algorithm for clustering categorical data with missing values, in: V. Torra, Y. Narukawa, I. Aguiló, M. González-Hidalgo (Eds.), MDAI 2018: Modeling Decisions for Artificial Intelligence, 2018, pp. 267–279..
[16] D.T. Dinh, V.N. Huynh, k-PbC: an improved cluster center initialization for categorical data clustering, Applied Intelligence 50 (2020) 1–23.
[17] D.T. Dinh, B. Le, P. Fournier-Viger, V.N. Huynh, An efficient algorithm for mining periodic high-utility sequential patterns, Applied Intelligence 48 (2018) 4694–4714.
[18] G. Gan, C. Ma, J. Wu, Data Clustering: Theory, Algorithms, and Applications. ASA-SIAM Series on Statistics and Applied Probability, 2007.
[19] Z. Huang, Clustering large data sets with mixed numeric and categorical values, in: Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, 1997, pp. 21–34..
[20] Z. Huang, Extensions to the k-means algorithm for clustering large data sets with categorical values, Data Mining and Knowledge Discovery 2 (1998) 283–304.
[21] Feng Jiang, G. Liu, J. Du, Y. Sui, Initialization of k-modes clustering using outlier detection techniques, Information Sciences 332 (2016) 167–183.
[22] B. Le, D.T. Dinh, V.N. Huynh, Q.M. Nguyen, P. Fournier-Viger, An efficient algorithm for hiding high utility sequential patterns, International Journal of Approximate Reasoning 95 (2018) 77–92.
[23] J. Liang, X. Zhao, D. Li, F. Cao, C. Dang, Determining the number of clusters using information entropy for mixed data, Pattern Recognition 45 (2012) 2251–2265.

[24] D. Lin, An information-theoretic definition of similarity, in: Proceedings of the Fifteenth International Conference on Machine Learning, 1998, pp. 296–304.
[25] N. Liu, Z. Xu, X.J. Zeng, P. Ren, An agglomerative hierarchical clustering algorithm for linear ordinal rankings, Information Sciences 557 (2021) 170–193.
[26] J.M. Luna-Romera, M. Martínez-Ballesteros, J. García-Gutiérrez, J.C. Riquelme, External clustering validity index based on chi-squared statistical test, Information Sciences 487 (2019) 1–17.
[27] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967, pp. 281–297..
[28] Y. Meng, R. Shang, L. Jiao, W. Zhang, Y. Yuan, S. Yang, Feature selection based dual-graph sparse non-negative matrix factorization for local discriminative clustering, Neurocomputing 290 (2018) 87–99.
[29] T.H.T. Nguyen, D.T. Dinh, S. Sriboonchitta, V.N. Huynh, A method for k-means-like clustering of categorical data, Journal of Ambient Intelligence and Humanized Computing (2019) 1–11.
[30] T.P. Nguyen, D.T. Dinh, V.N. Huynh, A new context-based clustering framework for categorical data, in: X. Geng, B.H. Kang (Eds.), PRICAI 2018: Trends in Artificial Intelligence, 2018, pp. 697–709..
[31] F. Nie, C.L. Wang, X. Li, K-multiple-means: A multiple-means clustering method with specified k clusters, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 959–967.
[32] F. Nie, X. Wang, H. Huang, Clustering and projected clustering with adaptive neighbors, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 977–986.
[33] F. Nie, Z. Zeng, I.W. Tsang, D. Xu, C. Zhang, Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering, IEEE Transactions on Neural Networks 22 (2011) 1796–1808.
[34] M. Pattanodom, N. Iam-On, T. Boongoen, Clustering data with the presence of missing values by ensemble approach, in: 2016 Second Asian Conference on Defence Technology (acdt), IEEE, 2016, pp. 151–156.
[35] W. Pedrycz, Knowledge-based clustering: from data to information granules, 2005..
[36] W. Pedrycz, Knowledge-based clustering in computational intelligence, in: Challenges for Computational Intelligence, 2007, pp. 317–341.
[37] D. Pfitzner, R. Leibbrandt, D. Powers, Characterization and evaluation of similarity measures for pairs of clusterings, Knowledge and Information Systems 19 (2009) 361.
[38] J. Quinlan, C4.5: Programs for Machine Learning. Ebrary online, 2014..
[39] A. Rosenberg, J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), 2007, pp. 410–420..
[40] O.M. San, V.N. Huynh, Y. Nakamori, An alternative extension of the k-means algorithm for clustering categorical data, International Journal of Applied Mathematics and Computer Science 14 (2004) 241–247.
[41] R. Shang, P. Tian, L. Jiao, R. Stolkin, J. Feng, B. Hou, X. Zhang, A spatial fuzzy clustering algorithm with kernel metric based on immune clone for sar image segmentation, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9 (2016) 1640–1652.
[42] R. Shang, Z. Zhang, L. Jiao, C. Liu, Y. Li, Self-representation based dual-graph regularized feature selection clustering, Neurocomputing 171 (2016) 1242–1253.
[43] R. Shang, Z. Zhang, L. Jiao, W. Wang, S. Yang, Global discriminative-based nonnegative spectral clustering, Pattern Recognition 55 (2016) 172–182.
[44] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, Journal of Machine Learning Research 3 (2002) 583–617.
[45] J. Su, H. Zhang, A fast decision tree learning algorithm, in: AAAI, 2006, pp. 500–505.
[46] P.N. Tan, V. Kumar, Interestingness measures for association patterns: A perspective, in: Proc. of Workshop on Postprocessing in Machine Learning and Data Mining, 2000, pp. 00–036..
[47] S. Ubukata, A. Notsu, K. Honda, Objective function-based rough membership c-means clustering, Information Sciences 548 (2021) 479–496.
[48] S.E. Wilson, Methods for Clustering Data with Missing Values Master thesis, Leiden University, 2015.
[49] J. Xu, J. Han, K. Xiong, F. Nie, Robust and sparse fuzzy k-means clustering, in: IJCAI, 2016, pp. 2224–2230.
[50] M. Zaït, H. Messatfa, A comparative study of clustering methods, Future Generation Computer Systems 13 (1997) 149–159.