Effective Clustering Algorithms for Categorical and Mixed Data

DINH DUY TAI

Japan Advanced Institute of Science and Technology

Effective Clustering Algorithms for Categorical and Mixed Data

DINH DUY TAI

S1720019

Supervisor: Prof. HUYNH VAN NAM

Second Supervisor: Prof. TAKASHI HASHIMOTO

A dissertation presented for the degree of

Doctor of Philosophy

Graduate School of Advanced Science and Technology

Japan Advanced Institute of Science and Technology

Knowledge Science

March, 2020

Examining Committee

Professor. **VAN-NAM HUYNH**

Japan Advanced Institute of Science and Technology

Professor. **TSUTOMU FUJINAMI**

Japan Advanced Institute of Science and Technology

Associate Professor. **HIEU-CHI DAM**

Japan Advanced Institute of Science and Technology

Associate Professor. **TAKAYA YUIZONO**

Japan Advanced Institute of Science and Technology

Professor. **HONG-BIN YAN**

East China University of Science and Technology

Abstract

This dissertation focuses on several topics for categorical and mixed data clustering. It provides a thorough background, theoretical models and empirical studies of the proposed frameworks. Key concepts and terminologies are also introduced. First, we design a novel clustering algorithm for categorical data. The algorithm uses a kernel-based method for the formation of cluster centers. This approach provides an interpretation of cluster centers being consistent with the statistical interpretation of the cluster means in numeric data clustering. In addition, taking the underlying distribution of categorical attributes into consideration, we define an information-theoretic based measure of dissimilarity for categorical data. This dissimilarity measure is used for computing the distance between categorical objects and cluster centers. The kernel-based method and information-theoretic based measure will be further used for clustering steps of all proposed frameworks in the dissertation. Second, we design an integrated framework for clustering categorical data with missing values. The proposed model can impute missing values occurring in data objects and assign them into appropriate clusters. For the imputation, we use a decision tree-based method to fill in missing values within data. This method has shown to be suitable for categorical data since it can find the set of complete objects that are highly correlated with the data object having missing values. From that, appropriate values are selected for missing positions. The kernel-based method and information-theoretic based measure are used for clustering steps. Third, we extend the second model to solve the problem of clustering mixed numeric and categorical data with missing values. For the imputation, the model splits an input data set into two sub-datasets based on their data types. The decision-tree based method is also used for imputing missing values inside objects constituted by categorical attributes. The missing values in numeric attributes are imputed by using the *mean* of corresponding attributes from the correlated set. For the clustering, we use the mean and the kernel-based method to define cluster centers for numeric and categorical attributes, respectively. The squared Euclidean and information-theoretic based dissimilarity measure is used to calculate distances for numeric and categorical attributes, respectively. Fourth, we design a framework to address the limitation of random initialization in categorical data clustering. Specifically, a maximal frequent itemset mining approach is used to find the sets of correlated itemsets (patterns). Each pattern describes the largest set of categories occurring in the corresponding categorical object. The group of data objects containing each pattern is considered as an initial cluster. The kernel-based method and information-theoretic based measure are used for clustering steps. Fifth, we design a framework to estimate the optimal number of clusters (k) in categorical data clustering. The silhouette analysis-based approach is used to evaluate different clustering results so as to choose the best k for each data set. The kernel-based method and information-theoretic based measure are used for clustering steps. All proposed frameworks are tested on real benchmark data sets from open access data repositories. We compare them with previous clustering algorithms in terms of clustering quality and computational complexity by using several internal and external validation metrics. In general, the proposed frameworks can enhance clustering results and can be used to perform clustering tasks for any real categorical and mixed data sets as long as their formats match the input requirement of algorithms.

Keywords: clustering, partitional clustering, categorical data, mixed data, missing values, kernel-based method, information-theoretic based dissimilarity, cluster center initialization, optimal number of clusters

Acknowledgments

First of all, I would like to express my sincerest gratitude to my advisor Professor HUYNH VAN-NAM from Japan Advanced Institute of Science and Technology (JAIST) for his guidance, encouragement, patience and support to me during my study at JAIST. He has gently inspired me in both research and lifestyle. I have learnt from him the ways to determine problems, analyse and find the solutions to solve them. I have also positively changed my attitude to any problems and difficulties in my life. Without his kind support, I could not finish my Ph.D program. In addition, I am very grateful to all Professors in the examining committee for their valuable comments and suggestions.

Next, I would like to express my special thanks to Professor TSUTOMU FUJINAMI from JAIST for his fruitful discussions for my minor research. In addition, I would like to thank Professor HO TU-BAO and Professor DAM HIEU-CHI from JAIST for teaching me several fundamental and advanced data analytics courses, which help me a lot for my research. Moreover, I would like to thank Professor LE HOAI-BAC from University of Science, VNU-HCMC for his suggestion and recommendation for me to study at JAIST.

Furthermore, I am indebted to the JAIST Doctoral Research Fellow (DRF) for granting me a scholarship during three years of my research. I would also like to express my gratitude to the Air Force Office of Scientific Research/Asian Office of Aerospace Research and Development (AFOSR/AOARD) for the grant under award number FA2386-17-1-4046. In addition, I would like to thank Professor NGUYEN CANH-HAO from Kyoto University for his advises when we worked in the same project. I would like to thank all JAIST staff for their great enthusiasm during the time I lived and studied at JAIST. I would love to send my sincere appreciation to all members of HUYNH's lab. Being a member of this lab is a wonderful time of my life. Especially, I want to thank all Vietnamese friends in HUYNH's lab. We joined many events and had great times together.

Finally, I would like to express my sincere gratitude to my parents, parents-in-law and my younger sister for supporting me with great love and support. I would also like to express my sincere gratitude to my wife. I would never complete my study without her support and sympathy. Last but not least, I want to express my gratitude to my little son. He is my motivation to do anything.

DINH DUY TAI

Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

ANN	**A**rtificial **N**eural **N**etwork
DM	**D**ata **M**ining
DT	**D**ecision **T**ree
FI	**F**requent **I**temset
FIM	**F**requent **I**temset **M**ining
KDD	**K**nowledge **D**iscovery in **D**atabases
KDE	**K**ernel **D**ensity **E**stimation
MFI	**M**aximal **F**requent **I**temset
ITBD	**I**nformation-**T**heoretic **B**ased **D**issimilarity

Chapter 1

Introduction

1.1 Research Background

The quick growth of information technologies and data acquisition technologies allows people to collect large amounts of data every day from different sources such as sensors, internet, satellites, applications, devices and other automatic equipment. The data reflect the behavior of analyzed systems, thus they may contain useful information and knowledge. To successfully obtain knowledge from large databases, a framework using many tools and results of other science fields called *knowledge discovery in databases* (KDD) has been used. KDD is a process of using data mining methods to find useful information and patterns in data. This task also has a variety of names such as knowledge extraction, information discovery, and data pattern recognition. However, the term *data mining* (DM) has been mostly used by statisticians, data analysts, and information systems managers. Data mining is the application of specific algorithms for extracting patterns from data. Figure 1.1 shows the general process of KDD [2]. It is a whole process to discover knowledge from data, while data mining is a step of this process. To ensure that useful knowledge is derived from the data, other additional steps such as data selection, preprocessing, transformation, and results interpretation are conducted in the KDD process.

Clustering is one of the important data mining methods for discovering knowledge in multivariate and multidimensional data. The goal of clustering is to identify patterns or groups of similar objects within a data set. The methodology consists of various algo-

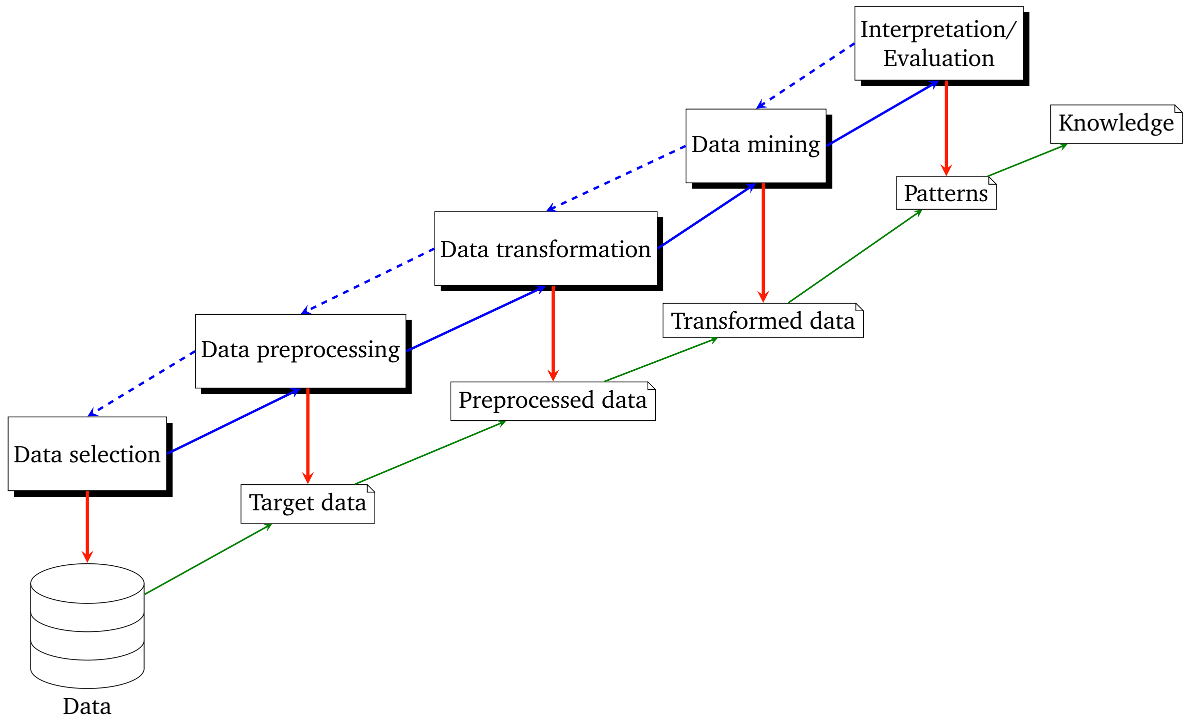
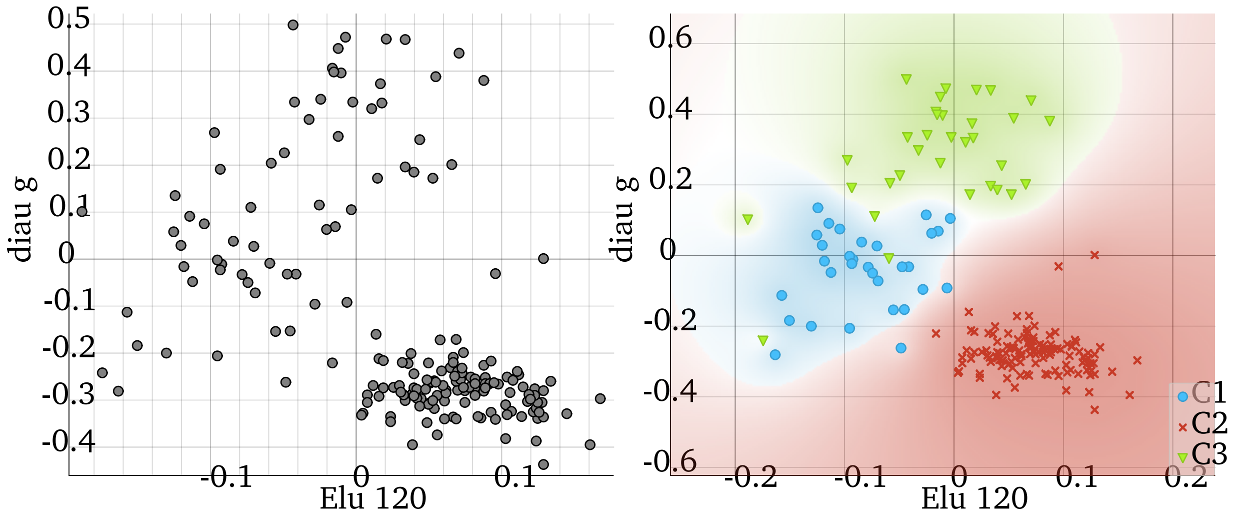


Figure 1.1: A general structure of Knowledge discovery in databases

rithms each of which seeks to organize a given data set into homogeneous subgroups, or clusters [63]. A cluster is generally considered as a group of objects in which each object is close to a central point of the cluster and that members of different clusters are far away from each other. In other words, those objects within each cluster are more closely related to one another than objects assigned to other different clusters. The objects may be words, text, images, database records, nodes in a graph, or any collection in which individuals are described by a set of features or distinguishing relationships. In the literature, clustering algorithms fall into the group of unsupervised machine learning, “unsupervised” because they are not guided by a priori ideas of which variables or samples belong in which clusters, and “learning” because the machine algorithms learn how to cluster [69].

The problem of clustering has been widely studied in data mining and machine learning literature since it can be applied to intermediate steps for other fundamental data mining problems and numerous application domains such as scientific data exploration, information retrieval and text mining, web analysis, marketing, collaborative filtering, customer segmentation, data summarization, dynamic trend detection, multimedia data analysis, medical diagnostics, biological data analysis and social network



(a) Scatter plot of the original data (b) Scatter plot of clustering results

Figure 1.2: k-means clustering for the Brown data set

analysis [15]. For example, clustering can be used in marketing to segment the market into homogeneous groups by identifying subgroups of customers with similar profiles. A market researcher may group consumers who seek similar benefits from a product so he can communicate with them better or provide appropriate advertising. A market analyst may be interested in grouping financial characteristics of companies so he is able to relate them to their stock market performance. In health-care research, clustering can be used to classify patients into subgroups according to their gene expression profile and identify the molecular profile of patients with good or bad prognostic, as well as for understanding the disease. In bioinformatics and genetics research, clustering can be used to identify groups of genes with similar patterns of expression, and this can help provide answers to questions of how gene expression is affected by various diseases and which genes are responsible for specific hereditary diseases. For instance, Figure 1.2a shows the scatter plots on two features (*diau g* and *Elu 120*) of the Brown data set, which contains 186 gene expressions of baker's yeast. Figure 1.2b shows three groups of instances by using k-means algorithm on the two features.

Clustering methods may belong to several broad categories as shown in Fig 1.3. In technique-based methods, distance-based algorithms are often desirable due to their simplicity and ease of implementation to a wide variety of scenarios. In addition, they are popularly used in the literature since they can be applied with almost any data type, as long as an appropriate distance function is created for that data type. Therefore, the problem of clustering can be reduced to the problem of finding a distance function for that data type [4]. Distance-based methods can be generally divided into two categories: hierarchical methods and partitional (flat) methods. Hierarchical methods aim to construct a tree (dendrogram) depicting specified relationships among the objects in the data set. Hierarchical methods can be divided into two paradigms: agglomerative (bottom-up) and divisive (top-down) clustering. Agglomerative clustering begins with each object being its own cluster. These clusters are then successively merged until only a single cluster remains. Divisive clustering begins with all objects as members of a single cluster. That cluster is then split into two separate clusters. Every successive cluster is performed in the same manner until each object is its own cluster. Partitional methods (also called as non-hierarchical or partitioning methods) split the data into a predetermined number k of groups or clusters, which together satisfy the requirements of a partition: each group must contain at least one object and each object must belong to exactly one group [70]. The advantage of partitional algorithms is that they are linear in the number of data points, scales well to large data sets and can be adapted to parallelization frameworks [7].

K-means [84] is the most widely used partitional clustering algorithm. The procedure aims to partition objects into a predetermined number of clusters. It is often used for large-scale clustering projects due to its simplicity and efficiency [63]. However, one of the major limitations of k-means is its ability to deal with nonnumerical attributes. Specifically, it can not be applied directly to categorical data, which is common in many real applications. This is because real data sets containing attributes such as age group, blood type, race, sex and zip code are inherently discrete and do not take on a natural ordering. To tackle this problem, a data transformation based method can be used to first transform categorical data into a new feature space, and then apply k-means to the newly transformed space to obtain the final results. However, this method has proven to be very ineffective and does not produce good clusters [97]. Thus, categorical data and mixed numeric and categorical data lead to challenges for clustering algorithms. First, a new similarity measure needs to be defined for categorical data since the standard similarity or distance functions for numeric data can no longer be used. Second, the representatives such as the means or medians for numeric data need to be appro-

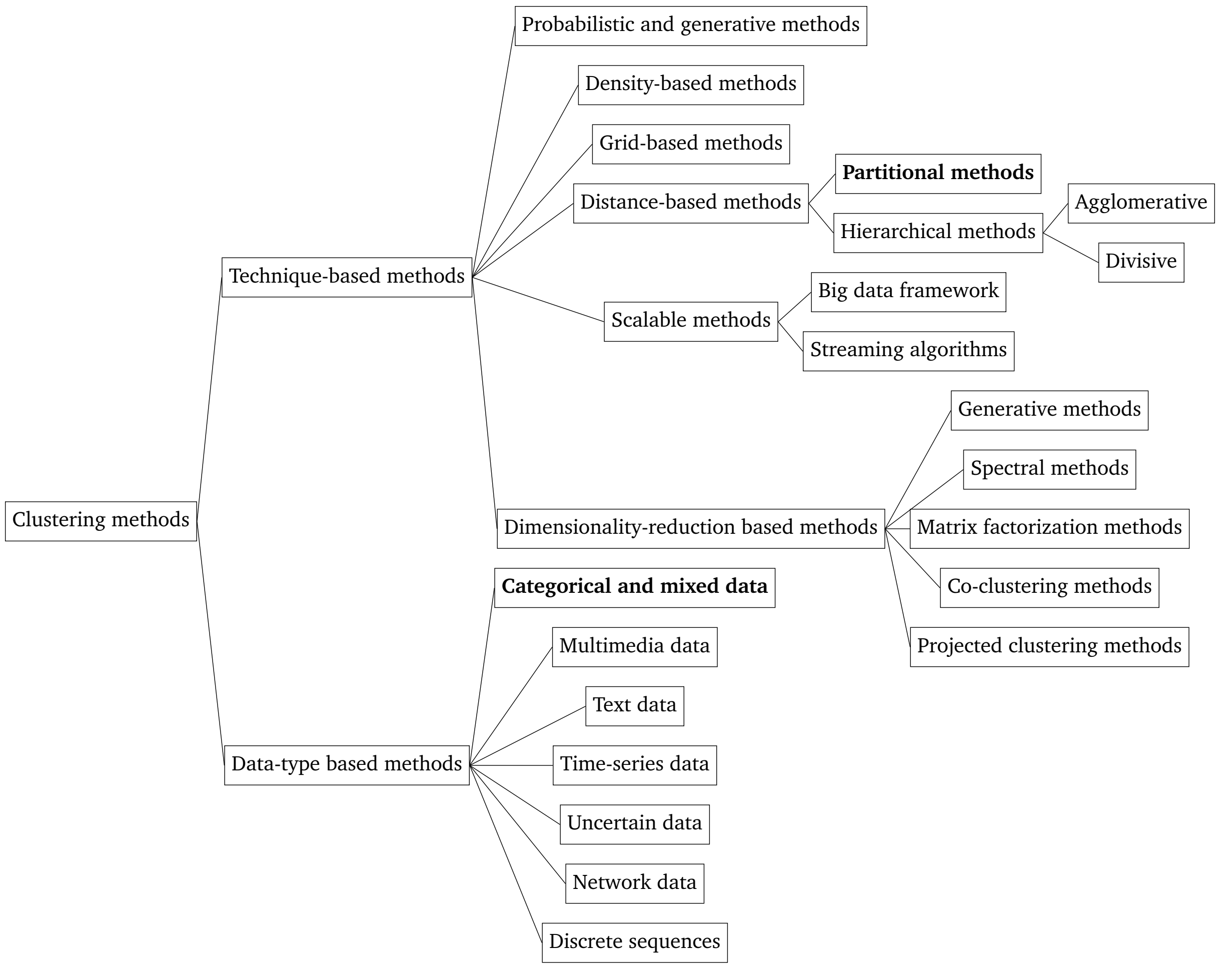


Figure 1.3: A classification of clustering algorithms

priately modified for discrete data [4].

During the last two decades or so, several attempts have been made to remove the numeric-only limitation of k-means algorithm and make it applicable to clustering for categorical data (so-called k-means-like methods) [18, 20, 25, 26, 59, 60, 68, 71, 72, 89–92, 99]. While these k-means-like algorithms use a similar clustering procedure to the k-means algorithm, they are different in the way of defining cluster centers (or cluster representatives) and distance measures for categorical data. The general process of these algorithms is as follows:

➢ Step 1: Start with the fixed number of clusters and select an initial partition of the objects.

➢ Step 2: After determining the cluster centers, assign each object to the object's nearest cluster center.

- Step 3: Determine the new cluster centers, or centroids of the clusters, based on the new partition created by the completion of step 2.
- Step 4: Repeat steps 2 and 3 until an optimum value of the objective function is achieved.

It is worth noting that both k-means and k-means-like algorithms find a local rather than a global optimum. The targets of these algorithms are to define (1) a cluster center that can represent categorical or mixed features (2) a distance measure that can deal with categorical features or combination of numeric and categorical features, and (3) a cost function, which is minimized iteratively, that can handle categorical or mixed data. In general, most of the partitional clustering algorithms optimize the following objective function:

$$\sum_{i=1}^{n} \mathbf{d}(x_i, C_i) \tag{1.1}$$

where n is the number of data instances in the data set, C_i is the cluster center nearest to data point x_i and $\mathbf{d}(,)$ is a distance measure between x_i and C_i.

1.2 Research Motivations

The motivations of this dissertation are based on the following observations:

- Clustering is one of the most popular research topics in data mining and knowledge discovery in databases. As mentioned before, 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, there is no perfect model for all clustering tasks. In addition, from both research and application viewpoints, the interest in clustering seems to be unwaning. Thus, we also focused on this topic for the dissertation.
- Categorical data and mixed numeric and categorical data are very common in real applications such as marketing, finance, medical and health care. However, it is challenging to directly use operations such as summation or averaging to compute the distance or dissimilarity measures for these kinds of data since their feature

values are not inherently ordered. Several methods have transformed categorical features into binary features. However, data transformation may lead to loss of information and result in misleading outcomes. Therefore, in the dissertation, we focused on categorical and mixed data clustering and find the other solution for measuring the similarity between categorical or mixed features.

- ➢ Partitional clustering algorithms are popularly used in clustering since they are linear to the size of the data sets and surely converge at a local optimum. They are also easy to implement since all they need are a distance function and a formation of cluster representatives. Inspired by their advantages, we used partitional methods for the design of proposed algorithms in the dissertation.

- ➢ In the era of information technology, missing values may occur frequently in the databases due to different mechanisms. Unfortunately, they may hide the true answers underlying in the data and reduce the performance of algorithms. Before clustering, an easy way is to pre-process the data sets so that they contain no missing values inside. However, pre-processing may be inconvenient for users and lead to a potentially biased data set when using inappropriate methods. Thus, we focused on the problem of clustering categorical and mixed data having missing values. The objective is to design a framework that can impute missing values and perform the clustering in a common process.

- ➢ Cluster center initialization and estimating the number of clusters are existing challenges of partitional clustering algorithms. Most of algorithms use a random selection method for cluster center initialization. However, this method may lead to different clustering results on different runs of the algorithms and bad results may be obtained in some cases. Thus, it is difficult to rely on such clustering results. In addition, most of algorithms work under the assumption that the number of clusters is known in advance. However, it is hard to guarantee that the chosen number of clusters corresponds to the natural number of clusters in the data and an unsuitable choice may influence the interpretation of the results. Thus, we focused on solving the two well-known problems of the partitional clustering algorithms for categorical data.

1.3 Research Topics

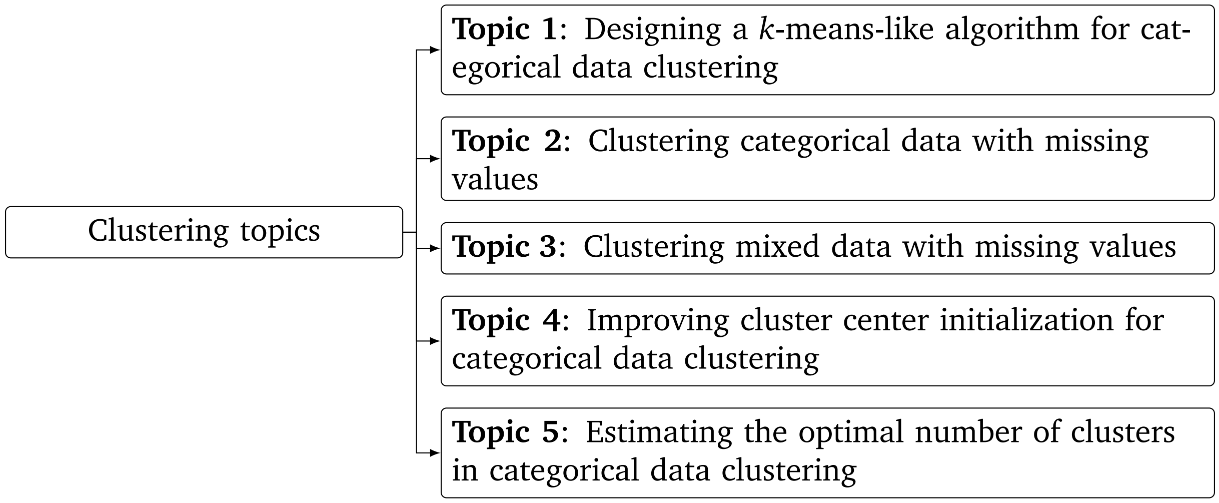

Clustering topics
Topic 1: Designing a k-means-like algorithm for categorical data clustering
Topic 2: Clustering categorical data with missing values
Topic 3: Clustering mixed data with missing values
Topic 4: Improving cluster center initialization for categorical data clustering
Topic 5: Estimating the optimal number of clusters in categorical data clustering

Figure 1.4: Clustering tasks in the dissertation

In this dissertation, we focused on five topics regarding clustering for categorical and mixed data as shown in Figure 1.4. The main tasks of these topics are highlighted as follows:

- ***Topic #1***: For this topic, we proposed a novel k-means-like approach for clustering categorical data, making use of an information theoretic-based dissimilarity measure and a kernel-based method for representation of cluster means for categorical objects. Such an approach allows us to formulate the problem of clustering categorical data in the fashion similar to k-means clustering, while a kernel-based definition of centers also provides an interpretation of cluster means being consistent with the statistical interpretation of the cluster means for numeric data. In order to demonstrate the performance of the new clustering method, a series of experiments on real data sets from UCI Machine Learning Repository was conducted and the obtained results were compared with several previously developed algorithms for clustering categorical data.

- ***Topic #2***: For this topic, we proposed a framework for clustering categorical data with missing values. The proposed method can impute missing values occurring in data objects and then assign them into appropriate clusters. For the imputation step, we used a decision tree-based method to fill in missing values. For the clustering step, we used a kernel-based method to define cluster centers and an information theoretic-based dissimilarity measure to quantify the differences between data objects and cluster centers. We then proposed an algorithm named clustering categorical data with missing values (k-CCM) for this task. An experimental evaluation was performed on real data sets with missing values to compare the performance of the proposed algorithm with other popular clustering algorithms in terms of clustering quality.

➢ ***Topic #3***: For this topic, we proposed a framework for clustering mixed numeric 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 numeric and categorical data with missing values (k-CMM). To impute missing values in mixed data, it employs the decision tree-based method to find the set of correlated data objects. To form centers of clusters, it uses the mean for numeric attributes and a kernel-based method for categorical attributes. To quantify the distances between data objects and cluster centers, it uses the squared Euclidean and an information-theoretic based dissimilarity measure for numeric and categorical attributes, respectively. To reduce the complexity of k-CMM, a dissimilarity measure was developed to select highly relevant values for the imputation. An extensive experimental evaluation was conducted on both synthetic and real data sets 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 were also evaluated for the various number of clusters or data sizes.

➢ ***Topic #4***: The performance of a partitional clustering algorithm is sensitive to the choice of initial cluster centers. An improper choice may lead to poor clustering results. This topic addresses the problem of the random initialization in categorical data clustering from the view of pattern mining. Specifically, a maximal frequent itemset mining approach was utilized to find a set of initial clusters. For the clustering step, we used a kernel-based method to define cluster centers and an information-theoretic based dissimilarity measure to determine the distance between data objects and cluster centers. We then proposed an algorithm named pattern based clustering for categorical data (k-PbC) that takes advantage of non-random initialization to improve clustering quality. A comparative experiment was performed on real categorical data sets to compare the performance of the proposed algorithm with previous clustering algorithms in terms of clustering quality.

➢ ***Topic #5***: Determining the number of clusters (say k) is a major challenge in partitional clustering. For this topic, we proposed a silhouette analysis based framework to estimate the optimal k, namely k-SCC. For the clustering, the framework also uses the kernel based method and an information-theoretic based distance measure for the assignment and update steps. The framework then calculates the average silhouette values (asv) for different clustering results obtained from the clustering step and select the best k that yields the highest asv. The experiments were performed on both real-life and synthetic data sets to evaluate the effectiveness of k-SCC. In addition, the proposed framework was applied to classify unlabeled Sake wine data set as a case study.

1.4 Research Contributions

The contributions of this dissertation are summarized as follows:

➢ The dissertation deals with several challenges of categorical and mixed data clustering. The first challenge is to find innovative ways to define a novel measure of similarity between categorical features, and to form cluster centers (representatives). To address this challenge, the dissertation introduces a new information-theoretic based dissimilarity measure and a kernel density estimation based method to improve the performance of the clustering algorithm. The second and third challenges are to solve the problem of clustering categorical and mixed data having missing values, which are common in many real applications. To address these challenges, the dissertation introduces an integrated framework that combines the imputation and clustering steps into a common process. It means that users do not need to pre-process or impute missing values in data sets in advance before using clustering. The fourth challenge is to enhance the initialization of cluster centers because the random initialization method leads to different results for different runs of the algorithm and bad results may be obtained in some cases. To address this challenge, the dissertation finds initial groups of correlated objects by using a pattern mining approach. The fifth challenge is to suggest the best suitable (optimal) number of clusters (k) for each data set. To address this challenge, the dissertation uses the silhouette coefficient to compute the average silhouette values for a range of k and choose the best one that yields the highest value.

➢ The dissertation provides a thorough background and theoretical frameworks for the corresponding topics. It then proposes algorithms and conducts extensive experiments to evaluate the performance of each algorithm on synthesis or real data sets. It also compares the proposed algorithms with previous clustering algorithms in terms of clustering quality by using several internal or external validation metrics. The proposed algorithms can be used to perform clustering tasks for any real categorical and mixed data sets as long as their formats match the input requirement of algorithms.

➢ The dissertation also contributes to knowledge science. It is obvious that the proposed frameworks belong to a branch of data mining, which is a step of the KDD process. Particularly, the proposed frameworks aim to reveal groups or clusters of similar entities in data. In other words, they help to find non-trivial or hidden patterns in data collected in databases and discover knowledge from them. Moreover, the proposed methods can be utilized in any steps of other research topics for the clustering task.

1.5 Dissertation Outline

The outline of this dissertation is shown in Figure 1.5. It contains eight chapters and the contents of chapters are briefly described as follows:

➢ Chapter 1 introduces the background, motivation and contribution of this research, as well as the topics conducted in the dissertation.

➢ Chapter 2 states the preliminaries of the categorical and mixed data clustering problem, in which the terminologies, definitions and problem statements are given

Organization of dissertation

- **Chapter 1**: Introduction
- **Chapter 2**: Background and Literature Review
- **Chapter 3**: A k-means-like method for clustering categorical data
- **Chapter 4**: Clustering categorical data with missing values
- **Chapter 5**: Clustering mixed numeric and categorical data with missing values
- **Chapter 6**: Improving cluster center initialization
- **Chapter 7**: Estimating the optimal number of clusters
- **Chapter 8**: Conclusion

Figure 1.5: Structure of the dissertation

in detail.

➢ Chapter 3 proposes a k-means-like framework for categorical data clustering, which uses a kernel-based method and an information-theoretic based dissimilarity measure to perform clustering. The theoretical framework and comparative experiment are presented and discussed in detail.

➢ Chapter 4 proposes a framework for clustering and dealing with the missing values in categorical data. The theoretical framework and comparative experiment are presented and discussed in detail.

➢ Chapter 5 proposes a framework for clustering and dealing with the missing values in mixed numeric and categorical data. The theoretical framework and comparative experiment are presented and discussed in detail.

➢ Chapter 6 proposes a framework for improving the cluster center initialization in categorical data clustering. The theoretical framework and comparative experiment are presented and discussed in detail.

➢ Chapter 7 proposes a framework to determine the optimal number of clusters in categorical data clustering. The theoretical framework and comparative experiment are presented and discussed in detail.

- ➢ Chapter 8 draws a summary of this dissertation and outlines research directions for future works.

Chapter 2

Background and Literature Review

2.1 Terminologies

The following terminologies used in this dissertation have been defined in several previous works [2, 63, 87].

➢ **Object** (or instance, record, observation, entity): is the item for clustering which corresponds to a data table row.

➢ **Feature** (or variable, attribute): corresponds to a data table column where the feature values can be compared to each other. For example, in categorical features, feature values are coincident or not, while in quantitative features (integer or continuous features), their values can be averaged or summed over any subset of objects.

➢ **Binary feature**: is the character-string feature that has only two possible values such as "male" or "female", "yes" or "no". It can be coded to "0" or "1" and called a dummy variable.

➢ **Nominal feature**: has a fixed number of values and cannot be usefully ordered. It is a general version of the binary feature which typically coded alphanumerically.

➢ **Ordinal feature**: is the character-string feature whose values are linearly ordered. For example, a feature named *performance* may contains values such as "excellent", "good", "fair", "poor", "bad", or a feature named *opinion* may contain values in the range of “strongly disagree” to “strongly agree”.

➢ **Integer feature**: contains non-negative numbers and can be used as a count.

➢ **Continuous feature**: can be specified as numeric or decimal depending upon the precision required.

➢ **Data**: relates to a database, data structure, similarity or dissimilarity matrix.

➢ **Data table** (or data matrix): is a two-dimensional array whose rows correspond to objects, columns to features and entries to feature values at objects.

2.2 Preliminaries

Table 2.1: List of notations

k	$\triangleq$	a predefined number of clusters
$\mathcal{D}_{cat}$	$\triangleq$	a categorical data set
$\mathcal{D}_{num}$	$\triangleq$	a numerical data set
x_i	$\triangleq$	a categorical object in $\mathcal{D}_{cat}$
$\mathcal{A}_j$	$\triangleq$	j^{th} attribute of $\mathcal{D}_{cat}$
$\mathcal{C}_l$	$\triangleq$	l^{th} cluster
n_l	$\triangleq$	the number of objects in $\mathcal{C}_l$
$\mathcal{Z}_l$	$\triangleq$	the center of cluster $\mathcal{C}_l$
$\mathcal{X}_j^l$	$\triangleq$	the random variable associated with observations in $\mathcal{C}_l$
$\mathcal{O}_j$	$\triangleq$	the set of categories at j^{th} attribute of $\mathcal{D}_{cat}$
$\mathcal{O}_j^l$	$\triangleq$	the set of categories at j^{th} attribute of cluster $\mathcal{C}_l$
o_{ij}	$\triangleq$	a category at i^{th} element and j^{th} attribute of $\mathcal{D}_{cat}$
o_{ij}^l	$\triangleq$	a category at i^{th} element and j^{th} attribute of $\mathcal{C}_l$
z_j^l	$\triangleq$	the value at j^{th} attribute of the center $\mathcal{Z}_l$

This section presents the fundamental definitions for the problem of clustering categorical data which has been the subject of several prior studies [18, 20, 25, 26, 59, 68, 71, 72, 89–92, 99]. Let $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\}$ be a set of m distinct categorical features, each of which is associated with a finite set $\mathcal{O}_j$ $(1 \leq j \leq m)$ as its domain such that $DOM(\mathcal{A}_j) = \mathcal{O}_j$ ($|\mathcal{O}_j| > 1$ discrete values). A categorical data set is a set of n categorical objects (instances) $\mathcal{D}_{cat} = \{x_1, x_2, \ldots, x_n\}$, where each categorical object $x_i \in \mathcal{D}_{cat}$ $(1 \leq i \leq n)$ is a tuple $x_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \in \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_m$. In other words,

$\mathcal{D}_{cat}$ can be represented by an $n \times m$ matrix ($n \gg m$), where n and m are the number of instances and features in data set $\mathcal{D}_{cat}$, respectively. Rows will always correspond to objects, columns to features. The element at position (i, j) $(1 \leq i \leq n,\ 1 \leq j \leq m)$ of the matrix indicates the value of the object x_i at the attribute j^{th}, such that $x_{ij} \in \mathcal{O}_j$. For example, Table 2.2 shows a categorical data set that contains ten objects with four categorical attributes.

Regarding the clustering problem discussed in this dissertation, we considered two types of data: *numeric* and *categorical*. The domain of numeric attributes consists of continuous real values. Thus, a distance measure such as Euclid distance or Manhattan distance can be used. A domain is defined as categorical if it is finite and unordered (nominal feature) so that only a comparison operation is allowed in this domain. It means, for any a, b of this domain, we have either $a = b$ or $a \neq b$.

Table 2.2: A categorical data set

Object	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
x_1	yellow	small	stretch	adult
x_2	yellow	small	stretch	child
x_3	purple	small	dip	adult
x_4	purple	small	dip	child
x_5	yellow	small	stretch	adult
x_6	yellow	small	stretch	child
x_7	purple	small	dip	adult
x_8	yellow	small	dip	child
x_9	yellow	large	stretch	adult
x_{10}	yellow	large	stretch	child

Definition 1 (Clusters) Given a categorical data set $\mathcal{D}_{cat} = \{x_1, \ldots, x_n\}$. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ be a set of k disjoint subsets that contain the indices of objects in $\mathcal{D}_{cat}$. These subsets become clusters of $\mathcal{D}_{cat}$ if they satisfy the two following conditions:

$$\begin{cases} \mathcal{C}_l \cap \mathcal{C}_{l'} = \emptyset \quad \forall\, l \neq l' \\ \bigcup_{l=1}^{k} \mathcal{C}_l = \mathcal{D}_{cat} \end{cases}$$

Example 1 Assume that objects in Table 2.2 are assigned into two sets. Then, $\mathcal{C}_1 = \{x_1, x_2, x_5, x_6, x_8\}$ and $\mathcal{C}_2 = \{x_3, x_4, x_7, x_9, x_{10}\}$ are two clusters of $\mathcal{D}_{cat}$, while $\mathcal{C}_3 = \{x_1,$

$x_2, x_5, x_6, x_8, x_9, x_{10}\}$, $\mathcal{C}_4 = \{x_3, x_4, x_7, x_9, x_{10}\}$ are not.

Definition 2 (Relative frequency) Given a categorical data set $\mathcal{D}_{cat}$, the relative frequency of the category o_{ij} at the j^{th} attribute of $\mathcal{D}_{cat}$ is denoted and defined as:

$$\mathfrak{f}(o_{ij}) = \frac{\#(o_{ij})}{n} \tag{2.1}$$

where $\#(o_{ij})$ is the number of objects that contain o_{ij} at the j^{th} attribute of $\mathcal{D}_{cat}$.

Given cluster a $\mathcal{C}_l$ and a category o^l_{ij} $(1 \leq i \leq n_l, 1 \leq j \leq m)$ at the j^{th} attribute of $\mathcal{C}_l$, the relative frequency of o^l_{ij} in $\mathcal{C}_l$ is denoted and defined as:

$$\mathfrak{f}_l(o^l_{ij}) = \frac{\#_l(o^l_{ij})}{n_l} \tag{2.2}$$

where $\#_l(o^l_{ij})$ is the number of objects that contain o^l_{ij} in $\mathcal{C}_l$ at the j^{th} attribute.

Example 2 In table 2.2, the relative frequency of category "stretch" in the attribute $\mathcal{A}_3$ is $\frac{6}{10}$, while its relative frequency in $\mathcal{C}_1 = \{x_1, x_2, x_5, x_6, x_8\}$ is $\frac{4}{5}$.

The key points when designing a partitional clustering algorithm are (1) using an appropriate method for representing cluster centers and (2) choosing a suitable distance measure for a specific data type. In this dissertation, we used a kernel density estimation based (kernel-based) approach and an information-theoretic based dissimilarity measure for the formation of cluster centers and distance function, respectively. They will be the core functions for the clustering algorithms proposed in the remaining chapters of the dissertation. From the statistical point of view, the center of a numeric cluster is the expectation of a continuous random variable associated with the data, based on the assumption that the variable follows a Gaussian distribution. Following this perspective, the center of a categorical cluster can be estimated by using the kernel-based method, called the probabilistic center. This method is a variation on Aitchison & Aitken's kernel function [8] to estimate the probability density function of each attribute in the center.

Definition 3 (Kernel-based method) Let $\mathcal{X}^l_j$ and $p(\mathcal{X}^l_j)$ denote a random variable associated with observations x_{ij} $(1 \leq i \leq n_l)$ occurring in cluster $\mathcal{C}_l$ at the j^{th} attribute and its probability of density, respectively. Let $\mathcal{O}^l_j$ and $\lambda_l \in [0, 1]$ denote the set of categories occurring at the j^{th} attribute of $\mathcal{C}_l$ such that $\mathcal{O}_j^l = \bigcup_{i=1}^{n_l} x_{ij}$ and the unique smoothing bandwidth for cluster $\mathcal{C}_l$, respectively. Using a kernel-density estimation method (KDE), $p(\mathcal{X}_j^l)$ is defined on the kernel function, given by $\mathcal{K}(\mathcal{X}_j^l, o_{ij}^l)$. For each value o_{ij}^l in $\mathcal{O}_j^l$ $(1 \leq i \leq n_l)$, the variation on Aitchison & Aitken's kernel function is denoted and defined as:

$$\mathcal{K}(\mathcal{X}_j^l, o_{ij}^l, \lambda_l) = \begin{cases} 1 - \frac{|\mathcal{O}_j^l|-1}{|\mathcal{O}_j^l|}\lambda_l & \text{if } \mathcal{X}_j^l = o_{ij}^l \\ \frac{1}{|\mathcal{O}_j^l|}\lambda_l & \text{otherwise} \end{cases} \tag{2.3}$$

The kernel density estimator of $p(\mathcal{X}_j^l)$ is denoted and calculated as in the following form (see [112]):

$$\hat{p}(\mathcal{X}_j^l, \lambda_l, \mathcal{C}_l) = \sum_{o_{ij}^l \in \mathcal{O}_j^l} \mathfrak{f}_l(o_{ij}^l)\mathcal{K}(\mathcal{X}_j^l, o_{ij}^l, \lambda_l) \tag{2.4}$$

Definition 4 (Smoothing bandwidth parameter) Let there be a cluster $\mathcal{C}_l$, a smoothing bandwidth parameter using the least-squares cross-validation is used to minimize the total error of the resulting estimation over data objects in this cluster [20, 92]. The optimal smoothing parameter for $\mathcal{C}_l$ is denoted and defined as:

$$\lambda_l = \frac{1}{(n_l - 1)} \frac{\sum_{j=1}^m (1 - \sum_{o_{ij}^l \in \mathcal{O}_j^l} [\mathfrak{f}_l(o_{ij}^l)]^2)}{\sum_{j=1}^m (\sum_{o_{ij}^l \in \mathcal{O}_j^l} [\mathfrak{f}_l(o_{ij}^l)]^2 - \frac{1}{|\mathcal{O}_j^l|})} \tag{2.5}$$

Definition 5 (Cluster Center) Let $\mathcal{O}_j^l$ denote a set of categories occurring at the j^{th} attribute of a given cluster $\mathcal{C}_l = \{x_1, x_2, \ldots, x_{n_l}\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ $(1 \leq i \leq n_l)$. The center of $\mathcal{C}_l$ is denoted and defined as:

$$\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\} \tag{2.6}$$

where the value at j^{th} element of $\mathcal{Z}_l$ is a probability distribution on $\mathcal{O}_j^l$ estimated by a kernel-based method using Eq. (2.4) and is defined as follows:

$$z_j^l = [\mathcal{P}_j^l(o_{1j}^l), \mathcal{P}_j^l(o_{2j}^l), \ldots, \mathcal{P}_j^l(o_{|\mathcal{O}_j^l|j}^l)] \tag{2.7}$$

The value of each categorical value o_{ij}^l $(1 \leq i \leq |\mathcal{O}_j^l|)$ is measured by using Eqs.

(2.2), (2.3) and (2.4) as follows:

$$\mathcal{P}_j^l(o_{ij}^l) = \begin{cases} \lambda_l \frac{1}{|\mathcal{O}_j^l|} + (1-\lambda_l)\mathfrak{f}_l(o_{ij}^l) & \text{if } o_{ij}^l \in \mathcal{O}_j^l \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

Example 3 Assume that objects in Table 2.2 are assigned into two clusters $\mathcal{C}_1 = \{x_1, x_2, x_5, x_6, x_8\}$ and $\mathcal{C}_2 = \{x_3, x_4, x_7, x_9, x_{10}\}$. Then, their cluster centers are defined as $\mathcal{Z}_1 = \{\{\text{“yellow”} : 1.0\}, \{\text{“small”} : 1.0\}, \{\text{“stretch”} : 0.8, \text{“dip”} : 0.2\}, \{\text{“adult”} : 0.4, \text{“child”} : 0.6\}\}$, and $\mathcal{Z}_2 = \{\{\text{“purple”} : 0.6, \text{“yellow”} : 0.4\}, \{\text{“small”} : 0.6, \text{“large”} : 0.4\}, \{\text{“dip”} : 0.6, \text{“stretch”} : 0.4\}, \{\text{“adult”} : 0.6, \text{“child”} : 0.4\}\}$, respectively.

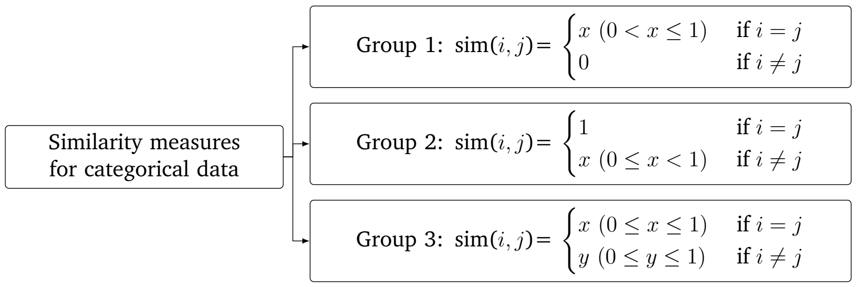

Similarity measures for categorical data
Group 1: sim(i, j)= $\begin{cases} x\ (0 < x \le 1) & \text{if } i = j \\ 0 & \text{if } i \ne j \end{cases}$
Group 2: sim(i, j)= $\begin{cases} 1 & \text{if } i = j \\ x\ (0 \le x < 1) & \text{if } i \ne j \end{cases}$
Group 3: sim(i, j)= $\begin{cases} x\ (0 \le x \le 1) & \text{if } i = j \\ y\ (0 \le y \le 1) & \text{if } i \ne j \end{cases}$

Figure 2.1: A taxonomy of similarity measures for categorical data

In recent years, many researchers have introduced similarity measures for categorical data [16, 20, 33, 58–60, 81, 92, 99]. Figure 2.1 shows a classification of similarity measures for categorical data. Generally, these distance metrics can be classified into three groups. In the first group, if two attribute values are similar, then possible values other than zero is assigned for the similarity, otherwise, zero is assigned for the similarity. In the second group, if two attribute values are similar, then one is assigned for the similarity, otherwise, possible values other than one is assigned for the similarity. In the third group, different values are assigned when matching and mismatching occur [33]. In 1998, Dekang Lin proposed a similarity measure that mimics the idea from information theory. This similarity belongs into the third group. It defines the similarity as the ratio between the common and different information such that less frequent item has a higher information gain. Several works then extended Lin's similarity for categorical data [16, 92]. This dissertation also used an information-theoretic based dissimilarity measure to estimate distances between objects and cluster centers or between objects.

Theorem 1 (Similarity theorem for the probabilistic model) In [81], Lin developed an information-theoretic framework for similarity within which a formal definition of similarity can be derived from a set of underlying assumptions. Basically, Lin's definition of similarity is stated in information theoretic terms, as quoted "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." Formally, the similarity between A and B is generally defined as:

$$\text{sim}(A, B) = \frac{\log P(\text{common}(A, B))}{\log P(\text{description}(A, B))} \tag{2.9}$$

where $P(s)$ is the probability of a statement s, and the information contained in s is measured by $[-\log P(s)]$. To show the universality of the information theoretic definition of similarity, Lin [81] also discussed it in different settings, including ordinal domain, string similarity, word similarity and semantic similarity.

In 2008, Boriah et al. [16] applied Lin's framework to the categorical setting and proposed a similarity measure for categorical data as follows. Let D_{cat} be a data set consisting of objects defined over a set of m categorical attributes with finite domains denoted by $\mathcal{O}_1$,...,$\mathcal{O}_m$, respectively. For each $j = 1, \ldots, m$, the similarity between two categorical values o_{ij}, $o_{i'j} \in \mathcal{O}_j$ is defined by:

$$\text{sim}_j(o_{ij}, o_{i'j}) = \begin{cases} 2 \log \mathfrak{f}_j(o_{ij}) & \text{if } o_{ij} = o_{i'j} \\ 2 \log(\mathfrak{f}_j(o_{ij})) + \mathfrak{f}_j(o_{i'j}) & \text{otherwise} \end{cases} \tag{2.10}$$

where $\mathfrak{f}_j(x) = \frac{\#(x)}{|\mathcal{D}_{cat}|}$, and $\#(x)$ being the number of objects in $\mathcal{D}_{cat}$ having the category x at j^{th} attribute. In fact, Boriah et al. [16] also proposed another similarity measure derived from Lin's framework and conducted an experimental evaluation of many different similarity measures for categorical data in the context of outlier detection.

It should be emphasized here that the similarity measure $\text{sim}_j(\cdot, \cdot)$ does not satisfy the Assumption 4 assumed in Lin's framework [81], which states that the similarity between a pair of identical object is 1. Particularly, the range of $\text{sim}_j(o_{ij}, o_{i'j})$ for $o_{ij} = o_{i'j}$ is $[-2 \log |\mathcal{D}_{cat}|, 0]$, with the minimum being attained when o_{ij} occurs only once and the maximum being attained when $\mathcal{O}_j = \{o_{ij}\}$. Similarly, the range of $\text{sim}_j(o_{ij}, o_{i'j})$ for

$o_{ij} \neq o_{i'j}$ is $[-2\log\frac{|\mathcal{D}_{cat}|}{2}, 0]$, with the minimum being attained when o_{ij} and $o_{i'j}$ each occur only once, and the maximum value is attained when o_{ij} and $o_{i'j}$ each occur $\frac{|\mathcal{D}_{cat}|}{2}$ times, as pointed out in [16]. Based on the general definition of similarity given in 2.9 and its application to similarity between ordinal values briefly discussed in [81], we introduced another similarity measure for categorical values as follows.

Definition 6 (Dissimilarity between two categories) Let there be two categories o_{ij} and $o_{i'j}$ occurring in two objects x_i and $x_{i'}$ at the j^{th} attribute. The similarity between them is denoted and defined as:

$$\text{sim}_j(o_{ij}, o_{i'j}) = \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{2.11}$$

where $\mathfrak{f}(o_{ij}, o_{i'j}) = \frac{\#(o_{ij}, o_{i'j})}{|\mathcal{D}_{cat}|}$, $\#(o_{ij}, o_{i'j})$ is the number of objects in data set $\mathcal{D}_{cat}$ that receive the value belonging to $\{o_{ij}, o_{i'j}\}$ at the j^{th} attribute.

The dissimilarity between o_{ij} and $o_{i'j}$ at the j^{th} attribute is measured as:

$$\text{dsim}_j(o_{ij}, o_{i'j}) = 1 - \text{sim}_j(o_{ij}, o_{i'j}) = 1 - \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{2.12}$$

Example 4 The dissimilarity of categories “yellow” and “yellow” at the attribute $\mathcal{A}_1$ of objects x_1 and x_2 in $\mathcal{D}_{cat}$ is dsim$_j$(“yellow”, “yellow”) = 1- sim$_j$(“yellow”, “yellow”) = 1 - $\frac{2\log \mathfrak{f}(\text{“yellow”, “yellow”})}{\log \mathfrak{f}(\text{“yellow”})+\log \mathfrak{f}(\text{“yellow”})}$ = 1 - $\frac{2\log\frac{7}{10}}{\log\frac{7}{10}+\log\frac{7}{10}}$ = 0, while the dissimilarity of “yellow” and “purple” in objects x_1 and x_3 is dsim$_j$(“yellow”, “purple”) = 1 - sim$_j$(“yellow”, “purple”) = 1 - $\frac{2\log \mathfrak{f}(\text{“yellow”, “purple”})}{\log \mathfrak{f}(\text{“yellow”})+\log \mathfrak{f}(\text{“purple”})}$ = 1 - $\frac{2\log\frac{10}{10}}{\log\frac{7}{10}+\log\frac{3}{10}}$ = 1.

Definition 7 (Dissimilarity between objects and cluster centers) Given a categorical object $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and a cluster $\mathcal{C}_l$ with its center is $\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\}$. Let $\mathcal{O}_j^l$ denote a set of categories occurring at the j^{th} attribute of $\mathcal{Z}_l$ (i.e. z_j^l). The dissimilarity between x_i and $\mathcal{Z}_l$ at the j^{th} attribute is calculated by accumulating the probability distribution on $\mathcal{O}_j^l$ and the dissimilarity between j^{th} component x_{ij} of the object x_i and the j^{th} component z_j^l of the center $\mathcal{Z}_l$. Mathematically, the definition can be formulated as follows:

$$\text{dis}_j(x_i, \mathcal{Z}_l) = \sum_{o_{ij}^l \in \mathcal{O}_j^l} \mathcal{P}_j^l(o_{ij}^l)\text{dsim}_j(x_{ij}, o_{ij}^l) \tag{2.13}$$

The dissimilarity between x_i and cluster center $\mathcal{Z}_l$ can be formulated as:

$$\text{dis}(x_i, \mathcal{Z}_l) = \sum_{j=1}^{m} \text{dis}_j(x_i, \mathcal{Z}_l) \tag{2.14}$$

From Eq. 2.14, if x_i and $\mathcal{Z}_l$ contain identical categories at each attribute or $\mathcal{Z}_l$ contains only x_i, then the dissimilarity between them is zero. If categories at each attribute of x_i and $\mathcal{Z}_l$ are totally different, then the upper bound dissimilarity between them equals to the number of features.

Example 5 Given two clusters $\mathcal{C}_1 = \{x_1, x_2, x_5, x_6, x_8\}$ and $\mathcal{C}_2 = \{x_3, x_4, x_7, x_9, x_{10}\}$ and their cluster centers $\mathcal{Z}_1 = \big\{$ {“yellow” : 1.0}, {“small” : 1.0}, {“stretch” : 0.8, “dip” : 0.2}, {“adult” : 0.4, “child” : 0.6} $\big\}$ and $\mathcal{Z}_2 = \big\{$ {“purple” : 0.6, “yellow” : 0.4}, {“small” : 0.6, “large” : 0.4}, {“dip” : 0.6, “stretch” : 0.4}, {“adult” : 0.6, “child” : 0.4} $\big\}$, respectively. The dissimilarity between object x_1 and $\mathcal{Z}_1$ is dis(x_1,$\mathcal{Z}_1$) $= 0 + 0 + 0.2 + 0.6 = 0.8$, while the dissimilarity between object x_1 and $\mathcal{Z}_2$ is dis(x_1,$\mathcal{Z}_2$) $= 0.6 + 0.4 + 0.6 + 0.4 = 2.0$.

The algorithm for categorical data clustering can be formulated in terms of an optimization problem as follows:

$$\mathcal{F}(\mathcal{U}, \mathcal{Z}) = \sum_{l=1}^{k} \sum_{i=1}^{n} u_{i,l} \times \text{dis}(x_i, \mathcal{Z}_l) \tag{2.15}$$

subject to

$$\begin{cases} \sum_{l=1}^{k} u_{i,l} = 1 & 1 \leq i \leq n \\ u_{i,l} \in \{0, 1\} & 1 \leq l \leq k,\ 1 \leq i \leq n \end{cases} \tag{2.16}$$

where $U = [u_{i,l}]_{n \times k}$ is the partition matrix in which $u_{i,l}$ takes value 1 if object x_i is in cluster $\mathcal{C}_l$ and 0 otherwise.

2.3 Validation Metrics

The cluster validation metrics measure the goodness of clustering results. Generally, these metrics can be categorized into 3 groups: internal validation metrics, external validation metrics and relative validation metrics. The first validation group uses the internal information of the clustering process to evaluate the goodness of a clustering structure without reference to external information. The second validation group compares the results of a cluster analysis to an externally provided class labels (ground-truth or gold standard classes). It measures how well cluster labels produced by clustering algorithms match to ground-truth labels. This approach is mainly used for selecting the right clustering algorithm for a specific data set. The third validation group evaluates the clustering structure by varying different parameter values for the same algorithm [69]. In the scope of this dissertation, the internal and external validation metrics are used for the assessment.

2.3.1 Internal validation metrics

An internal validation metric called silhouette coefficient [25, 98] was used to evaluate how well data objects are clustered by determining how close each object in one cluster is to objects in the neighboring clusters. The average silhouette value is mainly used in Chapter 7. However, we also used it to evaluate clustering results in Chapter 6. For the sake of brevity, in this section, we defined the average silhouette value for categorical data clustering as follows.

Definition 8 (Dissimilarity of two categorical objects) Given two objects $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and $x_{i'} = (x_{i'1}, x_{i'2}, \ldots, x_{i'm})$, the distance between x_i and $x_{i'}$ is denoted and defined as:

$$\text{dis\_objs}(x_i, x_{i'}) = \sum_{j=1}^{m} \text{dsim}_j(x_{ij}, x_{i'j}) \tag{2.17}$$

Definition 9 (Silhouette value for categorical objects) Given a categorical object x_i belonging to cluster $\mathcal{C}_l$ $(i \leq n_l)$. Let intra_dis(x_i) denote the average distance of the x_i to all other members of the same cluster $\mathcal{C}_l$. Let $\mathcal{C}_{l'}$ and dis$(x_i, C_{l'})$ denote some cluster other than $\mathcal{C}_l$ and the average distance of the x_i to all members of $\mathcal{C}_{l'}$, respectively. Let compute the average distances dis$(x_i, C_{l'})$ for all clusters $\mathcal{C}_{l'}$ other than $\mathcal{C}_l$ and choose inter_dis$(x_i) = \min_{C_{l'} \neq C_l}$ dis$(x_i, C_{l'})$. If cluster $\mathcal{C}_{l''}$ $(1 \leq l'' \leq k)$ satisfies the condition that inter_dis(x_i) = dis$(x_i, C_{l''})$ then $\mathcal{C}_{l''}$ is selected as the second-best cluster and the neighbor of x_i. The silhouette value of x_i is formulated by:

$$\text{sil}(x_i) = \frac{\text{inter\_dis}(x_i) - \text{intra\_dis}(x_i)}{\max\{\text{intra\_dis}(x_i), \text{inter\_dis}(x_i)\}} \tag{2.18}$$

Generally, sil(x_i) measures how well object x_i is classified into cluster $\mathcal{C}_l$. Its value is in the range $[-1, 1]$. High positive values of sil(x_i) mean that the *within* dissimilarity intra_dis(x_i) is much smaller than the smallest *between* dissimilarity inter_dis(x_i) and thus x_i is well-clustered. Conversely, high negative values of sil(x_i) means that x_i is poor-clustered. If sil(x_i) is about zero, it means that x_i lies between two clusters.

Definition 10 (Average silhouette value) Given a set of k clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, C_k\}$, the average silhouette value is calculated by averaging the silhouette values of all objects in $\mathcal{C}$:

$$\text{avg\_sil} = \frac{\sum_{i=1}^{n} \text{sil}(x_i)}{n} \tag{2.19}$$

2.3.2 External validation metrics

We first used three external validation metrics including accuracy, precision, recall to measure the performance of proposed algorithms. The accuracy, precision, recall are defined as in [68, 71, 88, 89]. These metrics are derived from information retrieval. Given a set of k classes yielded by a clustering method, objects in the cluster $\mathcal{C}_l$ $(1 \leq l \leq k)$ are assumed to be classified either correctly or incorrectly with respect to a given class of objects. Let a_l, b_l and c_l denote the number of correctly classified objects, the number of incorrectly classified objects and the number of objects in a given class but not in the cluster $\mathcal{C}_l$, respectively. The clustering accuracy (AC), precision (PR) and recall (RE) are defined as follows:

$$\text{AC} = \frac{\sum_{l=1}^{k} a_l}{n} \tag{2.20}$$

$$\text{PR} = \frac{\sum_{l=1}^{k} (\frac{a_l}{a_l + b_l})}{k} \tag{2.21}$$

$$\text{RE} = \frac{\sum_{l=1}^{k}\left(\frac{a_l}{a_l+c_l}\right)}{k} \tag{2.22}$$

To compute the AC, PR and RE values, we had to compute a confusion matrix to match the predicted cluster labels to ground-truth labels. In the dissertation, we used the same way as in [89] to find the best confusion matrix. Specifically, we evaluated k! mapping between the set of predicted labels and ground-truth. We then used the Hungarian algorithm [75] to find the best mapping and corresponding confusion matrix. The following example shows how to compute the AC, PR and RE.

Assume that data objects in Tables 2.2 are partitioned into two clusters, where their ground-truth and predicted labels are shown in Table 2.3. The best confusion matrix

Table 2.3: Ground-truth and predicted labels

Object ID	1	2	3	4	5	6	7	8	9	10
Ground-truth labels	t	t	f	f	t	t	f	t	f	f
Predicted labels	2	2	1	1	1	2	1	2	2	2

is selected by evaluating mappings from the set of predicted labels $\{1, 2\}$ to the set of ground-truth labels $\{t, f\}$. For example, the mapping $\{1 \rightarrow t, 2 \rightarrow f\}$ produces the confusion matrix as shown in Table 2.4. The value in each cell of this table shows

Table 2.4: The confusion matrix corresponding to the mapping $\{1 \rightarrow t, 2 \rightarrow f\}$

	t	f
1	**1**	3
2	4	**2**

the number of pairs (predicted, ground-truth) appearing in Table 2.3. The sum of each column is the number of objects in the corresponding class label (column), and the sum of each row is the numbers of objects assigned to corresponding predicted label (row). The values of AC, PR and RC are calculated as follows:

$$\text{AC} = \frac{1+2}{10} = 0.3000$$

$$\text{PR} = \frac{\left(\frac{1}{1+3} + \frac{2}{4+2}\right)}{2} = 0.2917$$

$$\text{RE} = \frac{\left(\frac{1}{1+4} + \frac{2}{3+2}\right)}{2} = 0.3000$$

Similarly, the mapping $\{1 \rightarrow f, 2 \rightarrow t\}$ produces the confusion matrix as shown in Table 2.5. Then the values of AC, PR and RC are calculated as follows:

Table 2.5: The confusion matrix corresponding to the mapping $\{1 \rightarrow f, 2 \rightarrow t\}$

	t	f
1	1	**3**
2	**4**	2

$$\text{AC} = \frac{3+4}{10} = 0.7000$$

$$\text{PR} = \frac{\left(\frac{3}{1+3} + \frac{4}{4+2}\right)}{2} = 0.7083$$

$$\text{RE} = \frac{\left(\frac{4}{1+4} + \frac{3}{3+2}\right)}{2} = 0.7000$$

which is the best confusion matrix and the AC, PR and RE of this case are selected as the final results.

Other three external validation metrics including Purity, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) were used to evaluate how well the assignment of objects to clusters matches their original class information. The definitions of these criteria are given in [85] as below.

Let there be a data set $\mathcal{D}_{cat}$ of n instances. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ denote the set of clusters created by a clustering algorithm and $\mathcal{Q} = \{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_{k'}\}$ denote the set of ground-truth classes in $\mathcal{D}_{cat}$. For the Purity metric, each cluster $\mathcal{C}_l$ $(1 \leq l \leq k)$ is assigned to the majority class $\mathcal{Q}_{l'}$ $(1 \leq l' \leq k')$ that occurs most frequently in $\mathcal{C}_l$. The Purity is then calculated by counting numbers of members of the majority class for all clusters and dividing them by n. This metric's values are in the range of $[0, 1]$ in which zero indicates a bad clustering and one indicates a perfect clustering.

$$\text{Purity}(\mathcal{C}, \mathcal{Q}) = \frac{1}{n} \sum_{l} \max_{l'} |\mathcal{C}_l \cap \mathcal{Q}_{l'}| \tag{2.23}$$

In some cases, the large number of clusters (say k) may lead to high Purity results. For this reason, it can not be used to trade off clustering quality against k. The NMI metric gives an insight into clustering quality, which is independent from k. It determines the average mutual information between any pairs of $\mathcal{C}$ and $\mathcal{Q}$. This metric's values are in the range of $[0, 1]$ in which the maximum value is obtained if generated clusters match the original partitions completely.

$$\mathrm{NMI}(\mathcal{C}, \mathcal{Q}) = \frac{\sum_{l=1}^{k} \sum_{l'=1}^{k'} |\mathcal{C}_l \cap \mathcal{Q}_{l'}| \log \frac{n|\mathcal{C}_l \cap \mathcal{Q}_{l'}|}{|\mathcal{C}_l||\mathcal{Q}_{l'}|}}{\sqrt{\sum_{l=1}^{k} |\mathcal{C}_l| \log \frac{|\mathcal{C}_l|}{n} \sum_{l'=1}^{k'} |\mathcal{Q}_{l'}| \log \frac{|\mathcal{Q}_{l'}|}{n}}} \tag{2.24}$$

Let $\mathfrak{p}$ denote the number of pairs of objects that belong to the same cluster in $\mathcal{C}$ and same class in $\mathcal{Q}$. Its expected value is measured by $E[\mathfrak{p}] = \frac{n_{\mathcal{C}} * n_{\mathcal{Q}}}{n(n-1)/2}$, where $n_{\mathcal{C}}$ and $n_{\mathcal{Q}}$ denote the number of pairs of objects from the same clusters in $\mathcal{C}$ and the same classes in $\mathcal{Q}$, respectively. The maximum value of $\mathfrak{p}$ is measured by $max(\mathfrak{p}) = \frac{1}{2}(n_{\mathcal{C}} + n_{\mathcal{Q}})$. The ARI [61] estimates the agreement of $\mathcal{C}$ and $\mathcal{Q}$ in regard to the hypothetical value of $\mathfrak{p}$ obtained when $\mathcal{C}$ and $\mathcal{Q}$ are two random, independent partitions [92]. It measures the deviation of $\mathfrak{p}$ from its expected value as below:

$$\mathrm{ARI}(\mathcal{C}, \mathcal{Q}) = \frac{\mathfrak{p} - E[\mathfrak{p}]}{max(\mathfrak{p}) - E[\mathfrak{p}]} \tag{2.25}$$

This metric's values are in range of $[-1, 1]$. If ARI is one, identical partitions are obtained.

2.4 Literature Review

2.4.1 *K*-means algorithm

K-means [84] is one of the top ten algorithms in data mining [117]. It partitions a given data set into k clusters such that objects within the same cluster are as similar as possible (high intra-class similarity), whereas objects from different clusters are as dissimilar as possible (low inter-class similarity). It starts either by assigning objects to one of k clusters and then computing k cluster centers or by choosing k representative objects as the initial centers, which correspond to the mean of points assigned to the clusters. Iteratively, k-means computes the distance of each object to its current cluster center. It then reassigns each object to its nearest cluster center so that the error sum of squares (ESS) (also called as the sum of squared error (SSE)) is reduced in magnitude. The cluster centers are updated after each reassignment. The procedure stops when no further reassignment reduces the value of ESS [63, 97]. Given a dataset $\mathcal{D}_{num} = \{x_1, \ldots, x_n\}$ of n instances and m features, assume that instances in $\mathcal{D}_{num}$ are assigned into k clusters, then the ESS in k-means is formulated by:

$$\mathcal{F}(\mathcal{U}, \mathcal{Z}) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j=1}^{m} u_{il} \times \mathbf{d}(x_{ij}, z_{lj}) \tag{2.26}$$

where $\mathcal{U} = [u_{il}]_{n \times k}$ be a partition matrix that satisfies:

$$\begin{cases} u_{il} \in \{0, 1\} \\ \sum_{l=1}^{k} u_{il} = 1 (1 \leq i \leq n) \end{cases}$$

and $\mathcal{Z} = \{\mathcal{Z}_l, l = 1, \ldots, k\}$ be a set of cluster centers where each element is composed by a set of m values, each being the mean of an attribute in $\mathcal{Z}_l$:

$$z_j^l = \frac{\sum_{x_i \in \mathcal{Z}_l} x_{ij}}{|\mathcal{Z}_l|}$$

and $\mathbf{d}(\cdot, \cdot)$ is the squared Euclidean metric of two attribute values. K-means aims at minimizing the ESS score and it often terminates at a local minimum of ESS.

The problem $\mathcal{F}$ can be solved by iteratively solving two problems:

- Fix $\mathcal{Z} = \hat{\mathcal{Z}}$ then solve the reduced problem $\mathcal{F}(\mathcal{U}, \hat{\mathcal{Z}})$ to find $\hat{\mathcal{U}}$.
- Fix $\mathcal{U} = \hat{\mathcal{U}}$ then solve the reduced problem $\mathcal{F}(\hat{\mathcal{U}}, \mathcal{Z})$.

Basically, the k-means algorithm iterates through a three-step process until $\mathcal{F}(\mathcal{U}, \mathcal{Z})$ converges to some local minimum:

1. Select an initial $\mathcal{Z}^{(0)} = \mathcal{Z}_1^{(0)}, \mathcal{Z}_2^{(0)}, \ldots, \mathcal{Z}_k^{(0)}$ and set $t = 0$.
2. Keep $\mathcal{Z}^{(t)}$ fixed and solve $\mathcal{F}(\mathcal{U}, \mathcal{Z}^{(t)})$ to obtain $\mathcal{U}^{(t)}$. That is, having the cluster centers, we then assign each object to the cluster of its nearest cluster center.
3. Keep $\mathcal{U}^{(t)}$ fixed and generate $\mathcal{Z}^{(t+1)}$ such that $\mathcal{F}(\mathcal{U}^{(t)}, \mathcal{Z}^{(t+1)})$ is minimized. That is, construct new cluster centers according to the current partition.

4. In the case of convergence or if a given stopping criterion is fulfilled, output the result and stop. Otherwise, set $t = t + 1$ and go to step 2.

In the numerical clustering problem, the Euclidean norm is often chosen as a natural distance measure in the k-means algorithm. With this distance measure, we calculated the partition matrix in step 2 as below, and the cluster center is computed by the mean of cluster's objects.

$$\text{if dis}(x_i, \mathcal{Z}_l) \leq \text{dis}(x_i, Z_{l'}) \text{ then}$$
$$u_{i,l} = 1, \text{ and } u_{i,l'} = 0, \text{ for } 1 \leq l' \leq k,\ l \neq l'$$

The main advantages of k-means are easy to implement and extremely efficient for dealing with large data sets. However, as an inherent downside, k-means can not directly handle categorical data with nonnumerical attributes. Such data is very common in many real applications such as health care, online shopping and retail.

2.4.2 Extensions of *k*-means for Categorical Data

K-modes Algorithm

To deal with the challenge of k-means for categorical data, many algorithms have been proposed in the ways that keep the same scheme and characteristics of k-means, while removing the numerical-only limitation and making them applicable for categorical data. K-modes is probably pioneer algorithm for this topic [60]. It uses the simple matching measure (also called as the Hamming distance) to count the number of mismatches between two objects. Given two objects x_1 and x_2 consisting of m categorical attributes, then the distance between them is measured by:

$$\text{dis}(x_1, x_2) = \sum_{j=1}^{m} \delta(x_{1j}, x_{2j}) \tag{2.27}$$

where

$$\delta(x_{1j}, x_{2j}) = \begin{cases} 0 & \text{if } x_{1j} = x_{2j} \\ 1 & \text{otherwise} \end{cases} \tag{2.28}$$

It uses a frequency-based method which employs the *mode* in each attribute to define cluster centers. K-modes can also be considered as an optimization problem as the form of (2.26) where $\mathcal{Z}$ be a set of *mode* vectors and each element $\mathcal{Z}_l$ is composed by m values $(z_1^l, z_2^l, \ldots, z_m^l)$, each being the mode of an attribute. K-modes keeps the same advantage as well as weak features of k-means. Its performance is also affected by object order and the way to initiate cluster modes [17, 60]. In other words, the locally optimal solutions produced in k-modes are sensitive to initialization methods.

K-representatives Algorithm

Instead of using the modes to form cluster centers, k-representatives [99] uses the notion of *representatives* for clusters. Particularly, each attribute of a representative is a distribution of categories occurring in that attribute. The dissimilarity between each object and representative is measured by the multiplication of relative frequencies of categories within the cluster and the simple matching measure between categories. The formulation of representatives and distance function are defined as in the following.

Again, let $\mathcal{C}_l = \{x_1, \ldots, x_p\}$ be a cluster of categorical objects and $x_i = (x_{i1}, \ldots, x_{im})$, $(1 \leq i \leq p)$, $p = n_l$. For each $j = 1, \ldots, m$, let $\mathcal{O}_j^l$ denote the set forming from categorical values $x_{1j}, \ldots, x_{pj}$. Then, the representative of $\mathcal{C}_l$ is defined by $\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\}$, where:

$$z_j^l = \left\{ \left(o_{ij}^l, \mathfrak{f}_l(o_{ij}^l)\right) | o_{ij}^l \in \mathcal{O}_j^l \right\} \tag{2.29}$$

where $\mathfrak{f}_l(o_{ij}^l)$ is the relative frequency of category o_{ij}^l within $\mathcal{C}_l$ and is defined as in Eq. 2.2. More formally, each z_j^l is a distribution on $\mathcal{O}_j^l$ defined by relative frequencies of categorical values appearing within the cluster.

Then, the dissimilarity between object $x_i = (x_{i1}, \ldots, x_{im})$ $(1 \leq i \leq p)$ and representative $\mathcal{Z}_l$ is defined based on the simple matching measure δ (Eq. 2.28) by:

$$\text{dis}(x_i, \mathcal{Z}_l) = \sum_{j=1}^{m} \sum_{o_{ij}^l \in \mathcal{O}_j^l} \mathfrak{f}_l(o_{ij}^l) \times \delta(x_{ij}, o_{ij}^l) \tag{2.30}$$

As such, the dissimilarity dis$(x_i, \mathcal{Z}_l)$ is mainly dependent on the relative frequencies of categorical values within the cluster and simple matching between categorical values.

K-centers Algorithm

More generally, Chen and Wang [20] proposed the k-centers algorithm by using the kernel-based method to define cluster centers, called the probabilistic centers. It incorporates a built-in feature weighting in which each attribute is automatically assigned with a weight to measure its contribution to the clusters. To estimate the distance between data objects and cluster centers, it uses the simple matching as an indicator function to represent each data object by a set of vectors and the Euclidean norm to quantify the dissimilarity.

Specifically, the center of a cluster $\mathcal{C}_l$ is defined as:

$$\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\} \tag{2.31}$$

in which the j^{th} element z_j^l is a probability distribution on $\mathcal{O}_j$ estimated by a kernel density estimation method [8]. More particularly, let denote $\mathcal{X}_j^l$ a random variable associated with observations x_{ij}, for $i = 1, \ldots, n_l$, appearing in $\mathcal{C}_l$ at the j^{th} attribute, and $p(\mathcal{X}_j^l)$ is its probability density. Using the kernel density estimation method (KDE), $p(\mathcal{X}_j^l)$ is defined on the kernel function, given by $\mathcal{L}(\mathcal{X}_j^l, o_{ij}, \lambda_l)$. Let $\mathcal{O}_j$ be the set forming from categorical values o_{ij} $(1 \leq i \leq n)$ occurring at the j^{th} attribute of the data set $\mathcal{D}_{cat}$. The variation on Aitchison and Aitken's kernel function is defined as:

$$\mathcal{L}(\mathcal{X}_j^l, o_{ij}, \lambda_l) = \begin{cases} 1 - \frac{|\mathcal{O}_j|-1}{|\mathcal{O}_j|}\lambda_l & \text{if } \mathcal{X}_j^l = o_{ij} \\ \frac{1}{|\mathcal{O}_j^l|}\lambda_l & \text{otherwise} \end{cases} \tag{2.32}$$

where $\lambda_l \in [0, 1]$ is the unique bandwidth for cluster $\mathcal{C}_l$ and is defined as in Eq. 2.5.

It is worth noting here that the kernel function $\mathcal{L}(\mathcal{X}_j^l, o_{ij}, \lambda_l)$ is defined in terms of the cardinality of the whole domain $\mathcal{O}_j$ but not in terms of the cardinality of the sub-domain $\mathcal{O}_j^l$ of the given cluster $\mathcal{C}_l$.

The kernel estimator of $p(\mathcal{X}_j^l)$ is calculated as:

$$\begin{aligned} \hat{p}(\mathcal{X}_j^l, \lambda_l, \mathcal{C}_l) &= \frac{1}{n_l}\sum_{i=1}^{n_l} \mathcal{L}(\mathcal{X}_j^l, x_{ij}, \lambda_l) \\ &= \mathfrak{f}_l(\mathcal{X}_j^l) + \left(\frac{1}{|\mathcal{O}_j|} - \mathfrak{f}_l(\mathcal{X}_j^l)\right)\lambda_l \end{aligned} \tag{2.33}$$

The probabilistic center of a categorical cluster in Eq. 2.31 can be estimated based on Eq. 2.33 as:

$$z_j^l = \left[\mathcal{P}_j^l(o_{1j}), \mathcal{P}_j^l(o_{2j}), \ldots, \mathcal{P}_j^l(o_{|\mathcal{O}_j|j})\right]$$

where:

$$\begin{aligned} \mathcal{P}_j^l(o_{ij}) &= \hat{p}(o_{ij}, \lambda_l, \mathcal{C}_l) \\ &= \lambda_l \frac{1}{|\mathcal{O}_j|} + (1-\lambda_l)\mathfrak{f}_l(o_{ij}) \end{aligned} \tag{2.34}$$

As mentioned in [20], the probabilistic center at an attribute of the cluster can be viewed as a Bayes-type probability estimator. Specifically, the first term of Eq. 2.34, which is a weighted average of a uniform probability, roles as a prior, while the second term, which is a frequency estimator, roles as the posterior. When $\lambda_l = 0$, the center degenerates to the pure frequency estimator, which is originally used in the k-representatives algorithm to define the center of a categorical cluster.

To measure the distance between a data object and its center, each data x_i is represented by a set of vectors $\{y_{ij}\}_{j=1}^m$ as defined below:

$$y_{ij} = \left[I(x_{ij} = o_{1j}), \ldots, I(x_{ij} = o_{lj}), \ldots, I(x_{ij} = o_{|\mathcal{O}_j|j})\right] \tag{2.35}$$

where $I(\cdot)$ is an indicator function whose value is either 1 or 0, indicating whether x_{ij} is the same as $o_{lj} \in \mathcal{O}_j$ or not. The dissimilarity on the j^{th} attribute is measured by:

$$\text{dis}_j(x_i, \mathcal{Z}_l) = \left\|y_{ij} - z_j^l\right\|_2 \tag{2.36}$$

where the Euclidean norm $\|a\|_2$ of a vector $a = \langle a_1, \ldots, a_l, \ldots \rangle$ is calculated by $\|a\|_2$ = $\sqrt{\sum_l a_l{}^2}$

It can be seen that k-centers uses a different way to calculate the dissimilarities between objects and cluster centers, but the idea of comparing two categorical values is still based on the simple matching method (represented by indicator function $I(\cdot)$). The remains of k-centers mimic the idea of k-means algorithm.

Chapter 3

A *k*-means-like method for clustering categorical data

3.1 Introduction

During the last decades, data mining has emerged as a rapidly growing interdisciplinary field, which merges together databases, statistics, machine learning and other related areas in order to extract useful knowledge from data [53]. Clustering is one of the fundamental tasks in data mining and machine learning that aims at partitioning a set of data objects into multiple clusters, so that objects within a cluster are similar to one another but dissimilar to objects in other clusters. Dissimilarities and similarities between objects are assessed based on those attribute values describing the objects and often involve distance measures. Clustering has been widely applied in a variety of fields, ranging from medical sciences, economics, computer sciences, engineering to social sciences and earth sciences. There is a vast body of knowledge in the area of clustering and there have been attempts to analyze and categorize them for a large number of applications [7, 15, 37, 118, 119]. However, clustering of massive data sets with categorical and mixed-type data is still a challenge in many applications of big data mining.

Additionally, according to [78], unsupervised learning will become far more important in the longer term, because labeling data is both costly and time-consuming, and sometimes impossible especially in the context of big data. As an important branch in unsupervised learning, clustering has recently reemerged as an active research topic in big data mining. Particularly, it can be considered as an important tool for analyzing the massive volume of data generated by modern applications: having big data clustered/grouped in a compact format that is still an informative version of the entire data [37]. There are difficulties in applying clustering techniques to big data due to new challenges that are raised with big data [102]. Especially, big data mining adds to clustering the complications of very large data sets with many attributes of different types [15].

Typically, in clustering, objects can be considered as vectors in n-dimensional space, where n is the number of features. When objects are described by numerical features, the similarity between objects can be defined based on distance measures such as Euclid distance or Manhattan distance. However, such distance measures are not applicable for categorical data which contains values, for instance, from gender, locations, etc. Clustering data with categorical attributes has increasingly gained considerable attention in the research community [44, 48, 51, 52, 59, 60, 99]. As for categorical data, the simple matching measure is most naturally used to define the similarity between objects [59]. However, this metric does not distinguish between the different values taken by the attribute, since we only measure the equality between pairs of values, as argued in [62].

The classical k-means algorithm [84] is probably the most popular and widely used clustering technique. Numerous k-means-like algorithms have been also proposed in the literature; each of which typically uses different similarity measures [74]. The k-means-like algorithms are easy to implement, linear time complexity in size of the data, and almost surely convergence to local optima [101]. Consequently, k-means-like algorithms are very efficient for handling large data sets. However, working only on numerical data prohibits them from being used for clustering categorical data. Despite recent efforts, the challenge in clustering categorical and mixed data in the context of big data still remains due to the lack of an inherently meaningful measure of similarity between categorical objects and the high computational complexity of existing clustering techniques.

So far, numerous attempts have been made in order to overcome the numerical-only limitation of the k-means algorithm, making it applicable for categorical data clustering, such as k-modes algorithm [60] and k-representatives algorithm [99], k-centers algorithm [20]. Particularly, in the k-modes algorithm [59], the simple matching similarity measure is used to compute the distance between categorical objects, and "modes" are used instead of "means" for cluster centers. The mode of a cluster is a data point, in which the value of each attribute is assigned the most frequent value of the attribute's domain set appearing in the cluster. Furthermore, Huang [60] also combined the k-modes algorithm with k-means algorithm in order to deal with mixed numerical and categorical databases. It is worth, however, noting that in k-modes algorithm a cluster can have more than one mode and, therefore, the performance of k-modes algorithm depends strongly on the selection of modes during the clustering process. This observation had motivated the authors in [99] to introduce a new notion of "cluster centers" called representatives for categorical objects and develop a so-called k-representatives algorithm for clustering categorical data. In particular, the representative of a cluster is defined by making use of the distributions of categorical values appearing in clusters. Then, the dissimilarity between a categorical object and the representative of a cluster is easily defined based on relative frequencies of categorical values within the cluster and the simple matching measure between categorical values. In such a way, the k-representatives algorithm is also formulated in a similar fashion to the k-means algorithm. In fact, it has been shown that the k-representatives algorithm is very effective for clustering categorical data [88]. More recently, Chen and Wang [20] have proposed a new kernel density-based method for defining cluster centers in central clustering of categorical data. Then the so-called k-centers algorithm that incorporates the new formulation of cluster centers and the weight attributes calculation scheme has been also developed. The experimental results have shown that the k-centers algorithm has good performance especially for the task of recognizing biological concepts in DNA sequences.

While the above-mentioned k-means based algorithms use a similar clustering procedure to the k-means algorithm, they are different in the way of defining "cluster center" or "similarity measure" for categorical data. In this chapter, we proposed a new extension of the k-means algorithm for clustering categorical data. In particular, as for measuring dissimilarity between categorical objects, we made use of the information-theoretic definition of similarity proposed in [81], which is intuitively defined based on the amount of information contained in the statement of commonality between values in the domain set of a categorical attribute. On the other hand, the definition of cluster centers is generalized using the kernel-based density estimates for categorical clusters as similarly considered in [20], instead of using the frequency estimates as originally in [99]. We then developed a new clustering method by incorporating a feature weighting scheme that automatically measures the contribution of individual attributes to the formation of the clusters.

3.2 The Proposed Clustering Method

Basically, the clustering method proposed in this chapter follows a general procedure for clustering categorical data outlined as follows.

1. Based on kernel smoothing techniques [8, 112], we developed a kernel-based method for representation of cluster centers for categorical objects. Such a kernel-based approach could provide an interpretation of cluster centers being consistent with the statistical interpretation of the cluster means for numerical data.

2. Taking the underlying distribution of categorical attributes into consideration, we defined an information-theoretic based measure of dissimilarity for categorical data. This dissimilarity measure will be further extended for computing the distance between categorical objects and cluster centers.

3. Under the above considerations, we can formulate the problem of clustering categorical data in the fashion similar to partitioning-based clustering. Eventually, we can also investigate incorporating weighting schemes introduced in [58] into partitioning-based clustering for categorical and mixed data.

In particular, we now introduce a new extension of the k-means clustering for categorical data by combining a kernel-based estimation method of cluster centers and an information-theoretic based dissimilarity measure for categorical objects as defined in Definitions 2.6 and 2.14, respectively.

Again, For each cluster $\mathcal{C}_l$, let us define the center of $\mathcal{C}_l$ as:

$$\mathcal{Z}_l = \{z_1^l, z_2^l, \dots, z_m^l\} \tag{3.1}$$

where z_j^l is a probability distribution on $\mathcal{O}_j^l$ estimated by a kernel density estimation method that can be obtained as:

$$z_j^l = [\mathcal{P}_j^l(o_{1j}^l), \mathcal{P}_j^l(o_{2j}^l), \dots, \mathcal{P}_j^l(o_{|\mathcal{O}_j^l|j}^l)] \tag{3.2}$$

where

$$\mathcal{P}_j^l(o_{ij}^l) = \begin{cases} \lambda_l \frac{1}{|\mathcal{O}_j^l|} + (1-\lambda_l)\mathfrak{f}_l(o_{ij}^l) & \text{if } o_{ij}^l \in \mathcal{O}_j^l \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

and

$$\lambda_l = \frac{1}{(n_l - 1)} \frac{\sum_{j=1}^m (1 - \sum_{o_{ij}^l \in \mathcal{O}_j^l} [\mathfrak{f}_l(o_{ij}^l)]^2)}{\sum_{j=1}^m (\sum_{o_{ij}^l \in \mathcal{O}_j^l} [\mathfrak{f}_l(o_{ij}^l)]^2 - \frac{1}{|\mathcal{O}_j^l|})} \tag{3.4}$$

Adapted from Huang's W-k-means algorithm [58], we also used a weighting vector $\mathcal{W} = [w_1, w_2, \dots, w_m]$ for m attributes and β being a parameter for attribute weight, where $0 \le w_j \le 1$ and $\sum_j w_j = 1$. The principal for attribute weighting is to assign a larger weight to an attribute that has a smaller sum of the *within cluster* distances and a smaller one to an attribute that has a larger sum of the *within cluster* distances. More details of this weighting scheme can be found in [58]. Then, the weighted dissimilarity between data object x_i and a cluster Z_l, is denoted and defined by:

$$\begin{aligned} \text{dis}^*(x_i, \mathrm{Z}_l) &= \sum_{j=1}^m w_j^\beta \text{dis}_j(x_i, \mathrm{Z}_l) \\ &= \sum_{j=1}^m w_j^\beta \sum_{o_{ij}^l \in \mathcal{O}_j^l} P_j^l(o_{ij}^l) \text{dsim}_j(x_{ij}, o_{ij}^l) \end{aligned} \tag{3.5}$$

where

$$\text{dsim}_j(o_{ij}, o_{i'j}) = 1 - \text{sim}_j(o_{ij}, o_{i'j}) = 1 - \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{3.6}$$

The clustering algorithm now aims to minimize the following objective function:

$$\mathcal{F}(\mathcal{U},\mathcal{Z},\mathcal{W}) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m} u_{i,l} \times w_j^{\beta} \times \text{dis}_j^*(x_i,\mathcal{Z}_l) \tag{3.7}$$

subject to:

$$\begin{aligned}
&\textstyle\sum_{l=1}^{k} u_{i,l} = 1, \quad 1 \le i \le n\\
&u_{i,l} \in 0,1, \qquad 1 \le i \le n, 1 \le l \le k\\
&\textstyle\sum_{j=1}^{m} w_j = 1, \quad 0 \le w_j \le 1\\
&\mathcal{U}{=}[u_{i,l}]_{n\times k} \text{ is a partition matrix.}
\end{aligned}$$

The proposed clustering algorithm is then formulated as below.

Algorithm 1: THE PROPOSED CLUSTERING ALGORITHM

```
input  : $\mathcal{D}_{cat}$: a categorical data set, $k$: a user-specified number of clusters
output: $k$ clusters of $\mathcal{D}_{cat}$
1 Select an initial $\mathcal{Z}^{(0)} = \{\mathcal{Z}_1^{(0)},\ldots,\mathcal{Z}_k^{(0)}\}$, set $t = 0$, $\lambda_j$= 0 for $j = 0,\ldots,k$, set
    $\mathcal{W}^{(0)} = [\frac{1}{m},\ldots,\frac{1}{m}]$
2 repeat
3     Keep $\mathcal{Z}^{(t)}$ and $\mathcal{W}^{(t)}$ fixed, generate $\mathcal{U}^{(t)}$ to minimize the distances between
        objects and cluster centers using Eq. (3.5)
4     Keep $\mathcal{U}^{(t)}$ fixed, update $\mathcal{Z}^{(t+1)}$ using Eq. (3.1)
5     Generate $\mathcal{W}^{(t+1)}$ as done in [58]
6     $t = t + 1$
7 until The partitions does not change.
```

3.3 Experimental Studies

In this section, we provided experiments conducted on ten data sets obtained from the UCI Machine Learning Repository [34] to compare the clustering performances of k-modes, k-representatives, and the proposed clustering method (denoted by New). In addition, in order to see the effectiveness and efficiency of the new dissimilarity measure and our modification of the kernel-based concept of cluster centers, we also conducted experiments for two modified versions of k-representatives and the proposed clustering method, respectively. In the modified version of k-representatives (denoted by M-k-representatives), we simply replaced the simple matching dissimilarity measure used in k-representatives with the information-theoretic based dissimilarity measure defined by Eq. 2.14. For the modified version of the proposed clustering method (denoted by M-k-centers), we used the information-theoretic based dissimilarity measure defined by Eq. 2.14 in combination with the concept of cluster centers defined by Chen and Wang [20]. That is, M-k-centers is as Algorithm 1 formulated above but with the cluster centers being updated by Eq. 2.34 instead of Eq. 2.8.

3.3.1 Data sets

The main characteristics of the used data sets are summarized in Table 3.1. These data sets are chosen to test our algorithm because of their public availability and since all attributes can be treated as categorical ones.

Table 3.1: Categorical data sets for the experiment

#	Data set	# instances	# attributes	# classes
1	Balance scale	625	4	3
2	Breast cancer	286	9	2
3	Car evaluation	1,728	6	4
4	Lenses	24	4	3
5	Mushroom	8,124	22	2
6	Nursery	12,960	8	5
7	Soybean (small)	47	35	4
8	Soybean (large)	683	35	19
9	Tictactoe	958	9	2
10	Tae	151	6	3

1. Balance scale data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced.

2. Breast cancer data set was obtained from the University of Wisconsin hospitals. Each instance has one of two possible classes: benign or malignant.

3. Car evaluation data set was derived from a simple hierarchical decision model originally developed for the demonstration of DEX (M. Bohanec, 1990). Each instance belongs to one of four classes: unacc, acc, good and vgood.

4. Lenses data set is used for fitting contact lenses. All instances in this data set is classified into three classes: contact lenses, soft contact lenses, no contact lenses.

5. Mushroom data set records drawn from the Audubon Society Field Guide to North American Mushrooms in 1981. It includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as edible or poisonous.

6. Nursery data set was derived from a hierarchical decision model originally developed to rank applications for nursery schools. All instances in this data set belong to five classes: not_recom, recommend, very_recom, priority, spec_prior.

7. Soybean (large) is the research of R.S. Michalski and R.L. Chilausky in 1980. It contains 307 instances with 35 attributes. All instances are classified into 19 classes.

8. Soybean (small) is a small subset of the original soybean database. It contains 47 instances and classified into 4 classes.

9. Tictactoe data set encodes the complete set of possible board configurations at the end of tic-tac-toe games, where “x” is assumed to have played first. It contains 958 instances that belong to two classes: positive and negative.

10. Tae data set consists of the evaluation of teaching performance over three regular semesters and two summer semesters of 151 teaching assistant assignments at the Statistics Department of the University of Wisconsin-Madison. All instances belong to three classes: low, medium and high.

3.3.2 Experimental Results

Evaluating the clustering quality is often a hard and subjective task [62]. Generally, objective functions in clustering are purposely designed so as to achieve high intra-cluster similarity and low inter-cluster similarity. This can be viewed as an internal criterion for the quality of a clustering [85]. However, as observed in the literature, good scores on an internal criterion do not necessarily translate into good effectiveness in an application. Here, by the same way as in [62], we used three external criteria to evaluate the results: Purity (Eq. 2.23), Normalized Mutual Information (NMI) (Eq. 2.24) and Adjusted Rand Index (ARI) (Eq. 2.25). These methods make use of the original class information of each object and the cluster to which the same objects have been assigned to evaluate how well the clustering result matches the original classes.

Table 3.2: Purity results of compared algorithms

#	Data set	k-modes	k-representatives	M-k-representatives	M-k-centers	New
1	Balance scale	0.5446	0.5708	0.5731	0.5752	**0.5777**
2	Breast cancer	0.7028	0.7028	**0.7098**	0.7028	0.7028
3	Car evaluation	0.7005	0.7047	0.7052	0.7050	**0.7058**
4	Lenses	0.6429	0.6949	0.7032	0.7051	**0.7061**
5	Mushroom	0.8818	0.7312	0.8861	0.7235	**0.8876**
6	Nursery	0.4377	0.4425	0.4403	0.4311	**0.4686**
7	Soybean (small)	0.9854	**1.0000**	**1.0000**	0.9873	0.9877
8	Soybean (large)	0.6107	0.7139	0.7113	0.7159	**0.7181**
9	Tictactoe	0.6534	0.6534	0.6534	0.6534	**0.6556**
10	Tae	0.4268	0.4583	0.4692	0.4696	**0.4702**

The experiments were run on a VPC cluster [38]. Each node is equipped with an Intel Xeon E5-2680v2@2.80GHz×20, 64 GB of RAM, running Red Hat Enterprise Linux 6.4. The proposed algorithm was implemented in Python using PyCharm. For each categorical data set, we ran 300 times per algorithm. We provided the parameter k equals to the number of classes in each data set. The performance of three evaluation metrics is calculated by the average after 300 times of running. The weighting exponent β was set to 8 as experimentally recommended in [58].

Table 3.3: NMI results of compared algorithms

#	Data set	k-modes	k-representatives	M-k-representatives	M-k-centers	New
1	Balance scale	0.0277	0.0434	0.0460	0.0477	**0.0485**
2	Breast cancer	0.0040	0.0018	**0.0576**	0.0042	0.0052
3	Car evaluation	0.0509	0.1269	0.1240	0.1256	**0.1273**
4	Lenses	0.1780	0.3312	0.3504	0.3558	**0.3597**
5	Mushroom	0.1937	0.2077	0.5310	0.5446	**0.5383**
6	Nursery	0.0593	0.0948	0.0941	0.0798	**0.0993**
7	Soybean (small)	0.9690	**1.0000**	**1.0000**	0.9802	0.9824
8	Soybean (large)	0.6085	0.7491	0.7509	0.7495	**0.7552**
9	Tictactoe	0.0179	0.0087	0.0373	0.0375	**0.0391**
10	Tae	0.0429	0.0730	0.0834	0.0858	**0.0859**

As we can see from Tables 3.2, 3.3 and 3.4, the proposed clustering method produces the best results in eight out of ten data sets in Purity and NMI indexes, and in seven out of ten data sets in Adjusted Rand Index. In these tables, boldfaced numbers indicate the best performances among compared algorithms for each data set. Comparing the performance of k-representatives and M-k-representatives, we can see that the new dissimilarity measure has yielded better results, especially in NMI and Adjusted Rand Index. Similarly, by comparing the performance of M-k-centers and the proposed algorithm, it is shown that our kernel-based concept of cluster centers has significantly improved the clustering performance in comparison to the kernel-based concept of cluster centers previously defined in [7]. In conclusion, the new approach has been proved to enhance the performance of previously developed k-means-like algorithms for clustering categorical data.

Table 3.4: Adjusted Rand Index results of compared algorithms

#	Data set	k-modes	k-representatives	M-k-representatives	M-k-centers	New
1	Balance scale	0.0300	0.0459	0.0486	0.0490	**0.0506**
2	Breast cancer	0.0019	-0.0030	**0.1351**	0.0021	0.0055
3	Car evaluation	0.0280	0.0562	0.0529	0.0562	**0.0644**
4	Lenses	0.0508	0.1596	0.1760	**0.2259**	0.2164
5	Mushroom	0.2357	0.2357	0.5963	0.5924	**0.6009**
6	Nursery	0.0496	0.0576	0.0591	0.0491	**0.0610**
7	Soybean (small)	0.9585	**1.0000**	**1.0000**	0.9733	0.9763
8	Soybean (large)	0.3759	0.4675	0.4679	0.4686	**0.4767**
9	Tictactoe	0.0217	0.0218	0.0357	0.0359	**0.0377**
10	Tae	0.0192	0.0412	**0.0477**	0.0475	**0.0477**

3.3.3 Significant Analysis

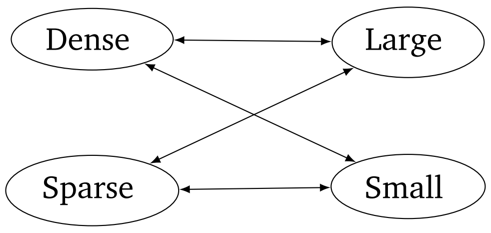


Figure 3.1: Types of data set in the experiment

In this section, we analysed the significance of the clustering results obtained in the previous section. Figure 3.1 shows the types of data sets used in the experiment. Particularly, we evaluated the efficiency of the New algorithm on large-dense, small-dense, large-sparse, small-sparse data sets. Table 3.5 shows the summary of Purity, NMI and ARI results of New algorithm proposed in this chapter. It can be observed that New algorithm has the highest values for all Purity, NMI and ARI metrics on eight over ten data sets: Balance scale, Car evaluation, Lenses, Mushroom, Nursery, Soybean (large), Tae and Tictactoe. From the practical viewpoint, the proposed algorithm can work well on both dense data sets such as Mushroom, Soybean (large) and sparse data sets such as Balance scale, Car evaluation, Lenses and Tae. In addition, it is also efficient on large data sets such as Mushroom, Nursery or small data sets such as Balance scale, Car evaluation, Soybean (large), Tictactoe, Lenses and Tae. Moreover, it performs well on large-dense data set such as Mushroom, large-sparse data set such as Nursery, small-dense data set such as Soybean (large) and small-sparse data set such as Balance scale, Lense, Tae. For Breast cancer and Soybean (small), each attribute domain contains less categorical values. In addition, the same values in each attribute tend to concentrate on the objects of the same class label. Also, objects are quite similar in several attributes. They are not so different in each selection of random initialization. Therefore, the frequency-based method and simple matching measure seem to be more efficient than other algorithms on those kinds of data sets.

Table 3.5: Significant analysis of New algorithm

#	Data set	Purity		NMI		ARI	
		1st rank	2nd rank	1st rank	2nd rank	1st rank	2nd rank
1	Balance scale	✓		✓		✓	
2	Breast cancer		✓		✓		✓
3	Car evaluation	✓		✓		✓	
4	Lenses	✓		✓			✓
5	Mushroom	✓		✓		✓	
6	Nursery	✓		✓		✓	
7	Soybean (small)		✓		✓		✓
8	Soybean (large)	✓		✓		✓	
9	Tictactoe	✓		✓		✓	
10	Tae	✓		✓		✓	

3.4 Conclusion

In this chapter, we have introduced a new k-means-like method for clustering categorical data based on an information-theoretic based dissimilarity measure and a kernel density estimate-based concept of cluster centers for categorical objects. Several variations of the proposed clustering method have been also examined. The experimental results on real data sets from UCI Machine Learning Repository have shown that the proposed clustering method outperformed the k-means-like algorithms previously developed for clustering categorical data.

Chapter 4

Clustering Categorical Data with Missing Values

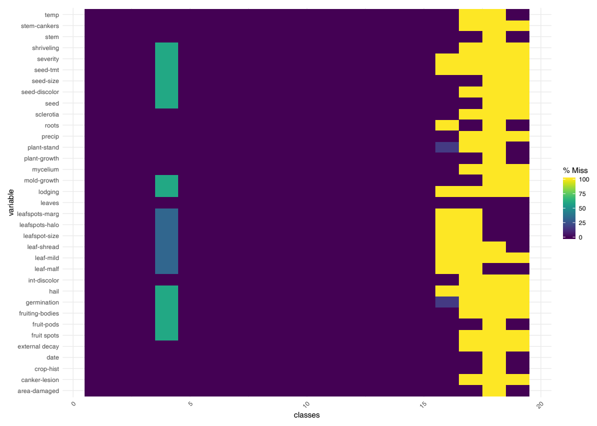
4.1 Introduction

Data mining techniques are designed to discover novel, useful, and unexpected patterns in databases, which can then be used for taking strategic decisions. Clustering, also called cluster analysis, segmentation analysis, taxonomy analysis, or unsupervised classification, is a method of creating groups of objects, or clusters, in such a way that objects in one cluster are very similar and objects in different clusters are quite distinct. It is one of the most important topics in data mining. The goal of clustering is to assign data points with similar properties to the same groups and dissimilar data points to different groups [43]. From a machine learning perspective, clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. Therefore, clustering is an unsupervised learning of a hidden data concept [15].

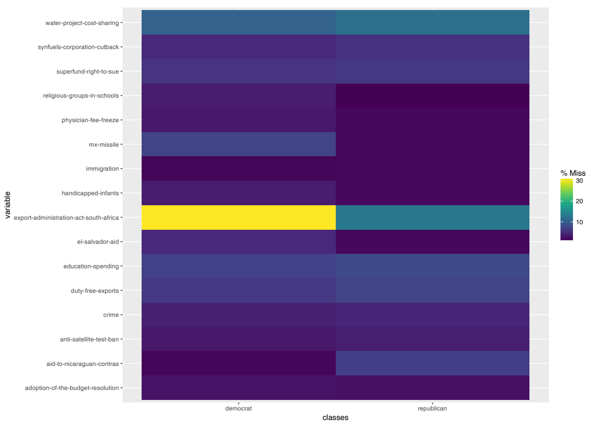
In general, clustering algorithms can be classified into two categories: hierarchical clustering and partitional clustering. A hierarchical clustering is a set of nested clusters where lower-level clusters are sub-clusters of higher-level clusters [109]. There are two types of hierarchical algorithms: divisive (top-down) hierarchical clustering and agglomerative (bottom-up) hierarchical clustering. In a divisive hierarchical algorithm, the algorithm starts with one large cluster containing all the data points in the data set and continues splitting clusters. In an agglomerative hierarchical algorithm, the algorithm starts with clusters each containing one data point and continues merging clusters. Unlike hierarchical algorithms, partitional algorithms create a one-level non-overlapping partitioning of the data points. For large data sets, hierarchical methods become impractical because the complexity of hierarchical algorithms is $\mathcal{O}(n^3)$ for CPU time and $\mathcal{O}(n^2)$ for memory space, while non-hierarchical methods generally have a time and space complexity of order n, where n is the number of data points in the data set [120]. Moreover, partitional clustering algorithms have shown their efficiency because their computational complexities are linearly proportional to the size of the data sets, they often terminate at a local optimum. Clustering algorithms are very highly associated with data types. Categorical data, which is also referred to as nominal data, appears popularly in many real-life applications. Categorical attributes are simply used as name, gender, age group, and educational level, etc. Designing partitional clustering algorithms for categorical data has attracted the attention of many researchers over the last two decades.

Nowadays, with the rapid development of information technology, data can be collected from various sources such as sensors, digital devices, machines and humans. They have made the amount of data quickly increase in a short time. However, collecting data is not always an easy task and may lead to missing values due to different mechanisms. There are many reasons that can lead to the problem of missing data. Missing data can be caused by human error, equipment malfunctions, system generated errors, the inconsistency between other recorded data and thus deleted, data not entered due to misunderstanding, certain data may not be considered important at the time of entry, changes of the data. Missing values also occur due to observing conditions, instrument sensitivity limitations, and other real-life considerations and so on. Unfortunately, missing values may hide a true answer underlying the data. Furthermore, it may reduce the performance of algorithms.

It can be easily seen that many categorical data from the UCI Machine Learning Repository [34] and the CMU data sets archive [115] contain missing values. Moreover, some existing frameworks for clustering categorical data such as [23] strongly suggest that users should consider filling in the missing data themselves in a way that makes



(a) Soybean data set



(b) Voting data set

Figure 4.1: Categorical data sets with missing values

sense for the problem at hand. This is especially important in the case of many missing values. This observation motivates a design of an algorithm for clustering categorical data with missing values. Generally, there are two ways to facilitate a clustering algorithm to run over categorical data sets with missing values. The first way is to pre-process data sets so that they only consist of complete values and then run the clustering algorithm. The second way is to develop a clustering algorithm that can deal with incomplete data sets. In this research, we focused on the latter way. More specifically, we developed a center-based algorithm that can run over incomplete categorical data sets without a pre-processing procedure. The key contributions of this chapter are summarized as follows:

- ➢ Based on the imputation method proposed in [24], we designed a new measure to quantify the similarity between an object with missing values and an object with no missing values. By using our proposed measure and the IS measure [107], we can find the most similar object with no missing values for an object with missing values. From that, the appropriate values can be chosen for imputation.
- ➢ We designed an integrated framework that combines imputation and clustering steps into a common process.
- ➢ We proposed a new categorical clustering algorithm named k-CCM that takes into account the advantages of missing values imputation to improve the performance of clustering algorithm.

- ➢ We carried out an extensive experimental evaluation on benchmark data sets from the UCI Machine Learning Repository [34] and the CMU data sets archive [115] to evaluate the performance of the proposed algorithm in terms of clustering quality.

The rest of this chapter is organized as follows. In the second section, related work is reviewed. In the third section, preliminaries and the problem statement are introduced. In the fourth section, a new clustering algorithm for categorical data with missing values is proposed. Next, the fifth section describes an experimental evaluation. Finally, the last section draws a conclusion.

4.2 Related work

The details of clustering methods for categorical data clustering are provided in section 2.4. Data now can be easily collected thanks to the development of information technologies and data acquisition technologies. However, they may contain missing values due to different mechanisms. Generally, there are three basic types of missing data [82]: missing completely at random (MCAR), missing at random (MAR) and not missing at random (NMAR). Methods to handle missing data also have vast variations and these choices are often made based on the data missing mechanism. The methods to deal with missing values can be classified into two groups: pre-replacing methods that replace missing values before the data mining process, and embedded methods that deal with missing values during the data mining process [42]. Several methods have been proposed to deal with missing data such as case deletion methods, imputation methods and non-imputation methods [1].

Imputation of missing values is an important task for improving the quality of the data mining results. Some of these methods are: expectation maximization imputation (EMI), decision tree-based methods, similarity-based imputation, k-decision tree-based imputation, k-nearest neighbor-based imputation, genetic algorithm and correlation-based imputation [24]. For imputation categorical data with missing values, Fujikawa et al. proposed two algorithms named Natural Cluster Based Mean-and-Mode (NCBMM) and attribute Rank Cluster Based Mean-and-Mode (RCBMM) [42]. The NCBMM can be applied to supervised data where missing value attributes can be either categorical or numeric. The RCBMM can be applied to both supervised and unsupervised data by filling up missing values for categorical attributes independently with the class attribute.

In 2013, Rahman proposed DMI and SiMI algorithms [96]. The DMI uses the decision tree and majority class voting method in the decision tree leaves to impute for categorical missing values. The SiMI uses the decision forest algorithm and the most frequent values method to impute for categorical missing values.

In 2016, Deb et al. proposed an imputation method named DSMI [24] that exploits the within-record and between-record correlations to impute missing data of numerical or categorical values. The DSMI algorithm first utilizes the decision tree to find the set of correlated records. Then, it uses the IS measure and the weighted similarity measure to exploit the correlation between missing and non-missing attributes within a record. The missing values are imputed by random sampling from a list of potential imputed values based on their degree of affinity. By modifying this imputation method, we integrated the imputation step into the clustering step to make it applicable to clustering categorical data with missing values. The next section introduces preliminary definitions and problem statement.

4.3 Preliminaries and problem statement

The clustering step of the proposed algorithm in this chapter uses a kernel-based method and an information-theoretic based dissimilarity measure as defined in section 2.2 of Chapter 2. Given a categorical data set $\mathcal{D}_{cat} = \{x_1, x_2, \ldots, x_n\}$ of n categorical objects where $x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ is a set of m categorical values at the i^{th} element of $\mathcal{D}_{cat}$. In other words, $\mathcal{D}_{cat}$ is an $n \times m$ matrix ($n \gg m$), where n and m are the numbers of objects and attributes in $\mathcal{D}_{cat}$, respectively. The element at position (i, j) $(1 \leq i \leq n$, $1 \leq j \leq m)$ of the matrix stores the value of the i^{th} object at the j^{th} attribute, such that $x_{ij} \in O_j$ $(|O_j| > 1))$ discrete values as the domain of the j^{th} attribute. Note that if a categorical value in $\mathcal{D}_{cat}$ is a missing value, then it is represented as "?" or " " (empty). For the sake of brevity, we denoted a categorical object/data set without missing values as a complete object/data set, while a categorical object/data set with missing values is denoted as an incomplete object/data set.

In 2016, Deb et al. used two measures, namely IS measure (ISM) and weighted similarity measure (WSM), to exploit the correlations between categorical values within an object and between objects in a data set. The ISM was first introduced by Tan et al. [107]. It computes the correlations between attributes and values of different attributes in an object. The WSM [49] computes the similarity between two values of an attribute in two different objects. It first builds graphs from related objects to calculate the direct relationship (called the first-level similarity) and transitive relationship (called the second-level similarity). Then, the WSM between two categorical values of an attribute is the weighted sum of the first level and second level similarity. Following this way, the WSM between two objects is calculated by averaging the weighted similarity measure of each attribute of the two objects. However, the computational cost of constructing graphs and calculating WSM is very high when dealing with high-dimensional data sets or data sets containing multiple missing values. To address this issue, we extended the information-theoretic based similarity measure Eq. (2.11) to make it applicable to measuring the similarity between complete and incomplete objects. We proposed a missing and complete similarity measure, namely MCS. The details of IS and MCS measures are as follows.

ISM contains the product of two quantities: interest factor and support count that compute the correlations between values of different attributes in an object. In other words, the ISM takes into account both the interestingness and significance of a pattern. Another interpretation of the ISM is as the geometric mean of confidence of rules that can be generated from two items, that is:

$$\text{IS}(A, B) = \sqrt{\text{Confidence}(A \Rightarrow B) \times \text{Confidence}(B \Rightarrow A)} \tag{4.1}$$

where the $\text{Confidence}(A \Rightarrow B) = P(A, B)/P(A)$, $P(A) = \#A/n$, $\#A$ and n be the number of data objects that contain A and the number of data objects in the data set, respectively.

Definition 11 (IS measure [24]) Given a categorical data set $\mathcal{D}_{cat} = \{x_1, x_2, \ldots, x_n\}$. Let there be a set T that contains both complete and incomplete categorical objects in $\mathcal{D}_{cat}$. Let $\mathcal{A}' = \{\mathcal{A}'_1, \mathcal{A}'_2, \ldots, \mathcal{A}'_{m'}\}$ and $\mathcal{A}'' = \{\mathcal{A}''_1, \mathcal{A}''_2, \ldots, \mathcal{A}''_{m''}\}$ $(\mathcal{A}', \mathcal{A}'' \subset \mathcal{A}; m', m'' < m)$ be two sets of categorical attributes that contain missing values and non-missing values in T, respectively. For all $a^{'} = \{a_1^{'}, a_2^{'}, \ldots, a_n^{'}\} \in \mathcal{A}_1^{'} \times \mathcal{A}_2^{'} \times \cdots \times \mathcal{A}_{m'}^{'}$ and $a^{''} = \{a_1^{''}, a_2^{''}, \ldots, a_n^{''}\} \in \mathcal{A}_1^{''} \times \mathcal{A}_2^{''} \times \cdots \times \mathcal{A}_{m''}^{''}$, the IS measure between $a^{'}$ and $a^{''}$ is defined as:

$$\text{IS}(a^{'}, a^{''}) = \frac{\text{Support}(a^{'}, a^{''})}{\sqrt{\text{Support}(a^{'}) \times \text{Support}(a^{''})}} \tag{4.2}$$

Where $\text{Support}(a^{'}, a^{''}) = \frac{\#(a^{'}, a^{''})}{|T|}$, $\#(a^{'}, a^{''})$ is the number of 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 Eq. (2.11) to make it applicable for measuring the proximity of complete and incomplete objects.

Definition 12 (Missing-complete similarity measure (MCS measure)) Let there be a set T that contains both complete and incomplete objects. Let there be two categorical values o_{ij} and $o_{i'j}$ appearing in T at the j^{th} attribute. The similarity between them is defined as:

$$\text{sim}_j^{mis}(o_{ij}, o_{i'j}) = \begin{cases} \frac{2 \log \mathfrak{f}_T(o_{ij}, o_{i'j})}{\log \mathfrak{f}_T(o_{ij}) + \log \mathfrak{f}_T(o_{i'j})} & \text{if } o_{ij} \neq ? \text{ and } o_{i'j} \neq ? \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where $\mathfrak{f}_T(o_{ij}) = \frac{\#_T(o_{ij})}{n_T}$, $\#_T(o_{ij})$ denotes the number of o_{ij} 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 and incomplete categorical object, respectively. The MCS between x_i and $x_{i'}$ is then defined as follows :

$$\text{MCS}(x_i, x_{i'}) = \sum_{j=1}^{m} \text{sim}_j^{mis}(x_{ij}, x_{i'j}) \tag{4.4}$$

Based on these definitions, the clustering algorithm for categorical data sets with missing values now aims to minimize the following objective function:

$$\mathcal{F}(\mathcal{U}, \mathcal{Z}) = \sum_{l=1}^{k} \sum_{i=1}^{n} u_{i,l} \times dis(x_i, \mathcal{Z}_l) \tag{4.5}$$

subject to

$$\begin{cases} \sum_{l=1}^{k} u_{i,l} = 1 & 1 \leq i \leq n \\ u_{i,l} \in \{0,1\} & 1 \leq l \leq k,\ 1 \leq i \leq n \end{cases} \tag{4.6}$$

where $\mathcal{U} = [u_{i,l}]_{n \times k}$ is the partition matrix ($u_{i,l}$ takes value 1 if object x_i is in cluster $\mathcal{C}_l$ and 0 otherwise).

4.4 The proposed *k*-CCM algorithm

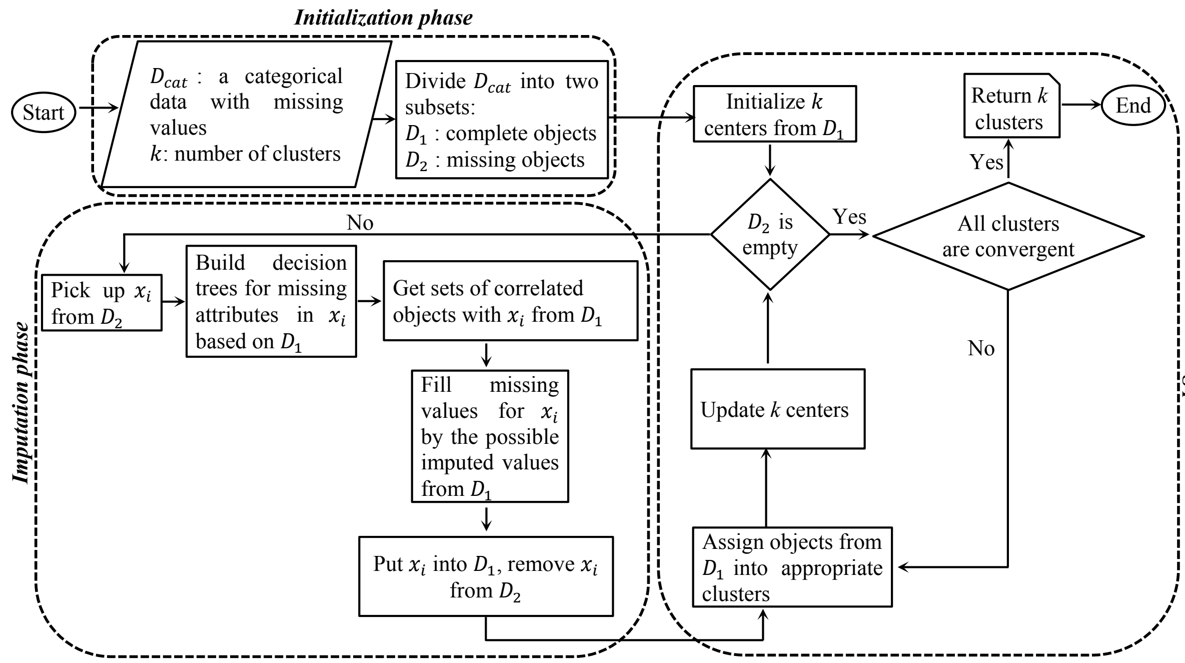

Initialization phase
Start
D_{cat} : a categorical data with missing values
k: number of clusters
Divide D_{cat} into two subsets:
D_1 : complete objects
D_2 : missing objects
Initialize k centers from D_1
Return k clusters
End
Yes
D_2 is empty
Yes
All clusters are convergent
No
No
Pick up x_i from D_2
Build decision trees for missing attributes in x_i based on D_1
Get sets of correlated objects with x_i from D_1
Fill missing values for x_i by the possible imputed values from D_1
Put x_i into D_1, remove x_i from D_2
Update k centers
Assign objects from D_1 into appropriate clusters
Imputation phase
Clustering phase

Figure 4.2: The flowchart of the k-CCM algorithm for categorical data with missing values

The proposed k-CCM algorithm is based on the general framework depicted in Figure 4.2. According to this model, a full categorical data set is divided into two sub-datasets: complete data set and missing data set. The first data set is used to perform the clustering process, while the second data set is used for the imputation process. For every incomplete object in the second data set, the algorithm uses a decision tree-based imputation approach to build decision trees for each missing attributes of this object. The incomplete object is then assigned to the corresponding tree's leaf. Once a missing object is assigned to a leaf node, the missing values in the incomplete object are imputed using objects that are found in the leaf node. Then, the imputed object is put into the first data set and removed from the second data set. Next, the algorithm performs clustering tasks by assigning objects in the first data set into appropriate clusters and update centers of clusters. That process is repeated until the second data set is empty. If the termination condition is not met, the algorithm performs the clustering process until all clusters are convergent. Finally, it returns k clusters.

Algorithm 2: THE k-CCM ALGORITHM

```
input  : $\mathcal{D}_{cat}$: a categorical database, $k$: a user-specified number of clusters
output: $k$ clusters of $\mathcal{D}_{cat}$
1  Scan $\mathcal{D}_{cat}$ once to split it into two sub-datasets, $\mathcal{D}_{cat} = \mathcal{D}_1$ (complete data set) +
     $\mathcal{D}_2$ (incomplete data set)
2  Randomly initiate $k$ cluster centers from $\mathcal{D}_1$: $\mathcal{Z}^{(0)} = \{\mathcal{Z}_1^{(0)}, \ldots, \mathcal{Z}_k^{(0)}\}$
3  $\mathcal{U} \leftarrow \emptyset$, $t = 0$
4  foreach object $x_i$ in $\mathcal{D}_2$ do
5      Categorical_Imputation($x_i$, $\mathcal{D}_1$)
6      $\mathcal{D}_1 = \mathcal{D}_1 \cup \{x_i\}$, $\mathcal{D}_2 = \mathcal{D}_2 \setminus \{x_i\}$
7      Keep $\mathcal{Z}^{(t)}$ fixed, generate $\mathcal{U}^{(t)}$ to minimize the distance between objects and
         cluster modes using Eq. (2.14)
8      Keep $\mathcal{U}^{(t)}$ fixed, update $\mathcal{Z}^{(t)}$ using Eq. (2.6)
9      $t = t + 1$
10 end
11 while Clusters are not convergent do
12     Keep $\mathcal{Z}^{(t)}$ fixed, generate $\mathcal{U}^{(t)}$ to minimize the distance between objects and
         cluster centers using Eq. (2.14)
13     Keep $\mathcal{U}^{(t)}$ fixed, update $\mathcal{Z}^{(t)}$ using Eq. (2.6)
14     $t = t + 1$
15 end
16 return $k$ clusters;
```

The pseudo code of the k-CCM algorithm is shown in Algorithm 2. It initially scans the categorical database $\mathcal{D}_{cat}$ once to divide this data set into two sub-datasets, namely $\mathcal{D}_1$ and $\mathcal{D}_2$, which are the complete and incomplete data sets, respectively (line 1). First, k-CCM randomly initiates k cluster centers from $\mathcal{D}_1$ (line 2). Each cluster center is formed by the Eq. (2.6). The set $\mathcal{U}$ is initiated to store the partition matrix (line 3). In the next step, the algorithm scans all objects in $\mathcal{D}_2$ to impute missing values and assign objects to clusters (lines 4-10). For each object x_i in $\mathcal{D}_2$, it calls the Categorical_Imputation procedure to fill in missing values inside x_i (line 5). The algorithm then puts x_i into $\mathcal{D}_1$ and removes it from $\mathcal{D}_2$ (line 6). Next, it assigns objects in $\mathcal{D}_1$ into appropriate clusters and updates cluster centers (lines 7-8). A similar process is performed for

Algorithm 3: CATEGORICAL_IMPUTATION PROCEDURE

```
input : $x_i$: a missing categorical object, $\mathcal{D}_1$: the set of complete categorical objects
output: the imputed categorical object $x_i$
1  $\mathcal{A}^c \leftarrow \emptyset$, $DTSet \leftarrow \emptyset$, $NodeSet \leftarrow \emptyset$, $\Omega_i \leftarrow \emptyset$
2  Put attributes having missing values in $x_i$ to $\mathcal{A}^c$
3  foreach attribute $\mathcal{A}_i^c$ in $\mathcal{A}^c$ do
4      if $DTSet$ does not contain $DT$ for $\mathcal{A}_i^c$ then
5          Build a decision tree $DT$ that uses $\mathcal{A}_i^c$ as class attribute from $\mathcal{D}_1$
6          $DTSet \leftarrow DT$
7      end
8  end
9  foreach decision tree $DT$ in $DTSet$ do
10     Assign $x_i$ 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 non-missing attribute(s) in $x_i$,
     and let $N$ be the number of such objects
15 for i = 1 to N do
16     $\Omega_i \leftarrow$ possible imputed value(s) from the $i^{th}$ matched object
17     Calculate the IS measure by Eq. (4.1) for $\Omega_i$
18     Calculate the MCS measure by Eq. (4.3) between the $i^{th}$ matched object and $x_i$
19     Calculate the affinity degree $\delta_i$ for $\Omega_i$ based on IS and MCS
20 end
21 Impute missing value(s) for $x_i$ by using random sampling from the set of possible
     imputed values $\{\Omega_1, \ldots, \Omega_N\}$ based on the sampling probabilities specified by the set of
     affinity degrees $\{\delta_1, \ldots, \delta_N\}$
22 return $k$ clusters;
```

all incomplete objects in the $\mathcal{D}_2$. Then, the k-CCM performs the clustering step until all clusters are convergent (lines 11-15). Finally, it returns k clusters as the desired output (line 16).

Algorithm 3 shows the pseudo code of Categorical_Imputation procedure. The input of this procedure is a missing categorical object x_i and the complete data set $\mathcal{D}_1$. The procedure first finds attributes containing missing values (called as missing attributes) and puts them into the set $\mathcal{A}^c$ (line 1). For each missing attribute $\mathcal{A}_i^c$ in $\mathcal{A}^c$, the procedure checks if there exists any decision tree (DT) that uses $\mathcal{A}_i^c$ as the class attribute. If there is no such DT, then a DT that uses $\mathcal{A}_i^c$ as the class attribute is constructed from the complete data set $\mathcal{D}_1$. This process is repeated until all missing attributes in $\mathcal{A}_i^c$ have their corresponding DTs (lines 3-8). Next, object x_i is assigned to the leaf nodes of the trees with the same class attribute as the missing attribute. Once x_i is assigned into appropriate leaves, each leaf node consists of objects from $\mathcal{D}_1$ and x_i that are correlated together (lines 9-12). Each leaf node in DT is represented as a list of objects. We used the same manner as the DSMI algorithm [24]. That is, if an object has more than one missing value fallen into multiple leaves, the algorithm will merge these leaves and group objects into the same collection (line 13). To impute the missing values in x_i, the algorithm searches for objects in the node which have the maximum number of non-missing attributes in common to the complete object (line 14). Then, the attribute values in these objects corresponding to the missing attributes in the missing object are taken to be the possible imputed values. For each such object, the procedure finds possible imputed values and calculate the IS and MCS measures for these values. The affinity degree of possible imputed values is given by the average of the IS and MCS measures computed for each possible imputed values (lines 15-20). Once affinity degrees of possible imputed values are obtained, the procedure assigns actual imputed values by random sampling from the list of possible imputed values based on their affinity degrees (line 21). According to [24], random sampling ensures that uncertainty and randomness in attribute values are accounted for and helps to reduce systematic bias in the imputed data set. Finally, the procedure return the imputed objected x_i without having missing values (line 22).

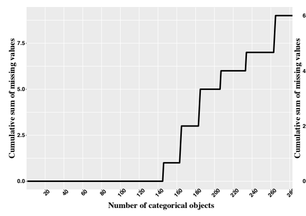
4.5 Comparative Experiment

Experiments were performed to evaluate the performance of k-CCM on a VPC cluster [38]. Each node is equipped with an Intel Xeon E5-2680v2@2.80GHz$\times$20, 64 GB of RAM, running Red Hat Enterprise Linux 6.4. The proposed algorithm was implemented in Python using PyCharm. The performance of the proposed k-CCM algorithm was compared with five partitional clustering algorithms: k-modes [59], k-representatives [99], M-k-representatives, M-k-centers and New [92]. Standard benchmark data sets were used for the experiment. Breast cancer, mushroom, soybean, sponge and congressional voting records (voting records) were obtained from the UCI Machine Learning Repository [34], while gsssex survey, runshoes and negotiation were obtained from the CMU data sets archive [115]. The characteristics of these data sets are shown in Table 4.1.

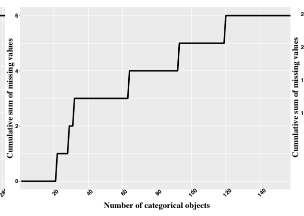
They have varied characteristics. By using these data sets, the performance of the proposed algorithm and the compared algorithms are evaluated for the main types of data encountered in the real-life world. For each algorithm, we ran 300 times per data set.

Table 4.1: Characteristics of the experimental data sets

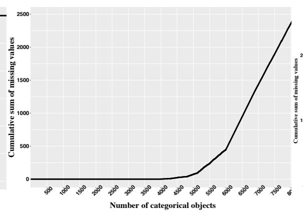
#	Data set	#objs	#attrs	#missing attrs	#missing objs	#missing values	#classes
1	Breast cancer	286	9	2	9	9	2
2	Gsssex survey	159	9	1	6	6	5
3	Mushroom	8,124	22	1	2,480	2,480	2
4	Negotiation	92	6	4	17	26	6
5	Runshoes	60	10	1	14	14	7
6	Soybean	307	35	34	41	712	19
7	Sponge	76	45	1	22	22	12
8	Voting records	435	16	16	203	392	2



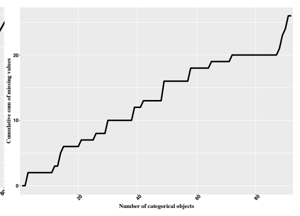
(a) Breast cancer



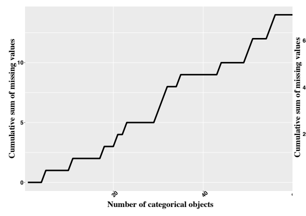
(b) Gss Sex Survey



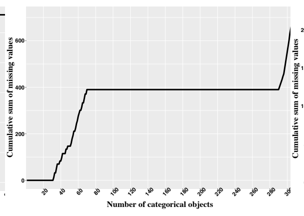
(c) Mushroom



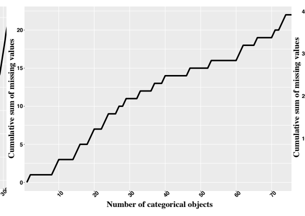
(d) Negotiation



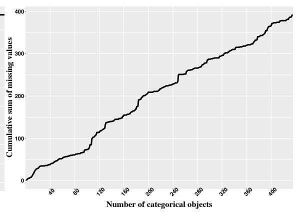
(e) Run shoes



(f) Soybean



(g) Sponge



(h) Voting

Figure 4.3: Missing values in experimental data sets

For the experiment, we used three metrics: Purity, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) as defined in section 2.3.2 to evaluate the performance of the proposed algorithm. The Purity, NMI and ARI results of the k-CCM and compared algorithms are shown in Tables 4.2, 4.3 and 4.4, respectively. In these tables, boldfaced numbers indicate the best performances among compared algorithms for each data set. It can be observed that k-CCM has better results than compared algorithms in most cases. For Purity, it outperforms other algorithms on Breast cancer, Gsssex survey, Negotiation, Runshoes, Sponge and Voting records data sets. On Mushroom, k-representatives has a higher result when compared to k-CCM. However, the

NMI and ARI results of k-CCM are better than those of k-representatives. The New algorithm outperforms other algorithms on the Soybean data set for all metrics. Similar results can be observed for the NMI and ARI metrics. In general, the proposed algorithm takes advantage of the imputation method and thus improves the performance of clustering process.

Table 4.2: Purity results of clustering algorithms

#	Data set	k-modes	k-representatives	M-k-representatives	M-k-centers	New	k-CCM
1	Breast cancer	0.7028	0.7028	0.7030	0.7028	0.7028	**0.7098**
2	Gsssex survey	0.7799	0.7814	0.7802	0.7803	0.7803	**0.7816**
3	Mushroom	0.8818	**0.8876**	0.8858	0.7235	0.7312	0.8861
4	Negotiation	0.4692	0.4829	0.4469	0.4517	0.4645	**0.5002**
5	Runshoes	0.4613	0.4591	0.4725	0.4725	0.4798	**0.4851**
6	Soybean	0.6107	0.7139	0.6955	0.6924	**0.7181**	0.6957
7	Sponge	0.9211	0.9211	0.9211	0.7159	0.9211	**0.9214**
8	Voting records	0.8581	0.8764	0.8713	0.8760	0.8775	**0.8805**

Table 4.3: NMI results of clustering algorithms

#	Data set	k-modes	k-representatives	M-k-representatives	M-k-centers	New	k-CCM
1	Breast cancer	0.0040	0.0018	0.0047	0.0042	0.0040	**0.0057**
2	Gsssex survey	0.0536	0.0606	0.0430	0.0428	0.0630	**0.0634**
3	Mushroom	0.5446	0.5383	0.5310	0.1937	0.2077	**0.5492**
4	Negotiation	0.0939	0.1342	0.1193	0.1039	0.1193	**0.1353**
5	Runshoes	0.2158	0.2192	0.2224	0.2246	0.2267	**0.2289**
6	Soybean	0.6085	0.7552	0.7314	0.7243	0.7509	**0.7555**
7	Sponge	0.0668	0.0638	0.0765	0.0748	**0.0887**	0.0770
8	Voting records	0.4359	0.4990	0.4961	0.4950	0.4947	**0.5002**

Table 4.4: Adjusted Rand Index results of clustering algorithms

#	Data set	k-modes	k-representatives	M-k-representatives	M-k-centers	New	k-CCM
1	Breast cancer	0.0019	-0.0030	**0.1351**	0.0021	0.0055	**0.1351**
2	Gsssex survey	0.0066	0.0135	0.0168	0.0140	**0.0171**	**0.0171**
3	Mushroom	0.5924	0.5952	0.5963	0.2357	0.2527	**0.6009**
4	Negotiation	0.0143	**0.0267**	0.0105	0.0111	0.0193	0.0155
5	Runshoes	0.0381	0.0320	0.0385	0.0335	0.0385	**0.0392**
6	Soybean	0.3759	**0.4767**	0.4163	0.4167	0.4686	0.4167
7	Sponge	-0.0173	-0.0176	-0.0024	-0.0030	**0.0190**	-0.0007
8	Voting records	0.5119	0.5658	0.5504	0.5540	0.5644	**0.5779**

4.5.1 Significant Analysis

In this section, we analysed the significance of the clustering results of k-CCM obtained in the previous section. We considered the same types of data sets as shown in Figure

3.1. Particularly, we evaluated the efficiency of k-CCM algorithm on large-dense, small-dense, large-sparse, small-sparse data sets. Table 4.5 shows the summary of Purity, NMI and ARI results of k-CCM proposed in this chapter. It can be observed that k-CCM has the highest values for at least two of three metrics on five over eight data sets: Breast cancer, Gsssex survey, Mushroom, Runshoes, Voting records; and highest values for two metrics: Negotiation data set. On Sponge, k-CCM has the highest value for Purity value, while achieving the second highest values for NMI and ARI. On Soybean, k-CCM has the highest value for NMI value, while not achieving good results for Purity and ARI. Generally, from the practical results, we can see that k-CCM can work well on small-sparse data sets such as Breast cancer, Gsssex survey, Runshoes, Negotiation and Voting records. It also can work well with large-dense data set such as Mushroom. However, for small-dense data sets such as Soybean and Sponge, k-CCM does not seem to be working very well on those kinds of data sets. Algorithms other than k-CCM consider missing values as new categories when forming cluster centers. Also, the simple matching measure considers two missing values as identical. Regarding the degree of missing values, when missing values spread in most attributes such as in the case of Soybean or when the number of missing values is highly proportional to the size of data set such as in the case of Sponge, other algorithms such as k-representatives outperform k-CCM.

Table 4.5: Significant analysis of the k-CCM algorithm

#	Data set	Purity			NMI			ARI		
		1st rank	2nd rank	3rd rank	1st rank	2nd rank	3rd rank	1st rank	2nd rank	3rd rank
1	Breast cancer	✓			✓			✓		
2	Gsssex survey	✓			✓			✓		
3	Mushroom		✓		✓			✓		
4	Negotiation	✓			✓					✓
5	Runshoes	✓			✓			✓		
6	Soybean			✓	✓					✓
7	Sponge	✓				✓			✓	
8	Voting records	✓			✓			✓		

4.6 Conclusion

In this chapter, we have proposed an algorithm named k-CCM for clustering categorical data sets with missing values. The proposed algorithm integrates the imputation step and clustering step into a common process. In this way, all incomplete objects are first imputed and then assigned into appropriate clusters. In particular, we extended a decision tree-based imputation method [24] to fill in missing values. For clustering, we used a kernel density estimation approach to define cluster centers and an information-theoretic based dissimilarity measure to quantify the difference between objects and cluster centers. An extensive experimental evaluation was conducted on benchmark categorical data sets to evaluate the performance of the proposed algorithm. According to the experimental results, the designed algorithm has a comparative result in terms of clustering quality when compared to the other five algorithms. Thus, the imputation step can improve the quality of the clustering.

Chapter 5

Clustering Mixed Numeric and Categorical Data with Missing Values

5.1 Introduction

The basic problem of clustering may be stated as "Given a set of data points, partition them into a set of groups which are as similar as possible" [3]. In other words, clustering aims to divide the set of objects into homogeneous groups in which two arbitrary objects belonging to the same group are more similar to each other than two arbitrary objects belonging to different groups [116]. It can be applied in many areas of engineering and sciences such as life sciences, behavioral and social sciences, medical sciences, earth sciences, engineering sciences, information, policy and decision sciences [9]. Fig. 5.1 proposes a classification of distance-based clustering algorithms. It consists of two categories: hard clustering and fuzzy clustering (or soft clustering). In the first category, a data point belongs to one and only one cluster, while in the second category, a data point may belong to two or more clusters with some probabilities.

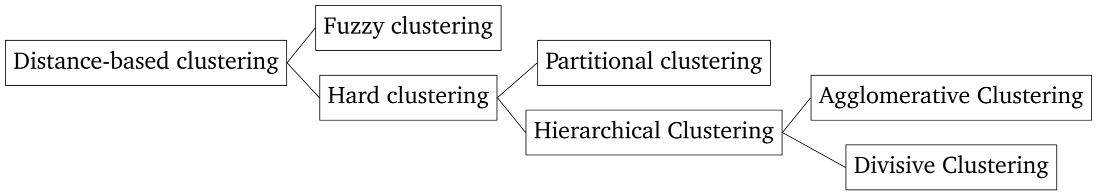


Figure 5.1: A classification of distance-based clustering algorithms

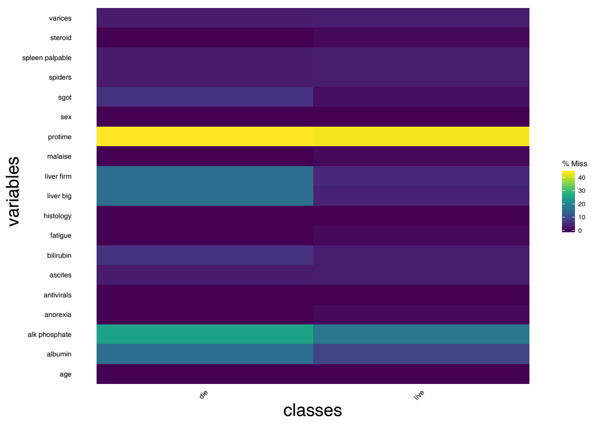
There are two types of hard clustering methods: hierarchical clustering and partitional clustering. Hierarchical clustering methods produce a nested series of partitions based on a criterion for merging or splitting clusters based on similarity. It does not require to predetermine the number of clusters to be produced. There are two types of hierarchical clustering: agglomerative and divisive clustering. The former initially considers each data object as a cluster of its own. It then successively merges the most similar clusters until there is just one single big cluster. In contrast to the former, the later initially considers all objects as one cluster. It then successively divides the most heterogeneous clusters until all data objects are in their own cluster. The result of hierarchical clustering is a tree-based representation of the objects, which is also known as a dendrogram. Partitional clustering methods group a given data into a user-specified k clusters by optimizing a certain objective function (usually locally) that captures a local and global structure of grouping. They start with an initial assignment and then use an iterative relocation procedure to move data objects from one cluster to another to optimize the objective function. They have been widely used in several real applications because of their simplicity and their competitive computational complexity.

Clustering algorithms are associated with certain types of data. The numeric data are represented by continuous values, whereas the categorical data, which are a special case of the discrete type, can have only a finite number of values. The categorical data appear frequently in many real-life applications. For example, they can be used as name, gender, age group, educational level, and blood type. The mixed data sets contain both numeric and categorical values. Real-life data are often of mixed types. For example, medical data include categorical values such as nationality, gender, job, education, marital status, and smoking or non-smoking in addition to numeric values such as age, height, weight, and salary. Retail purchase transactions consist of categorical values such as categories of items, types and customers' locations and numeric values such as quantity, unit profit, and price. Many researchers have focused on designing algorithms for clustering mixed data during the last two decades.

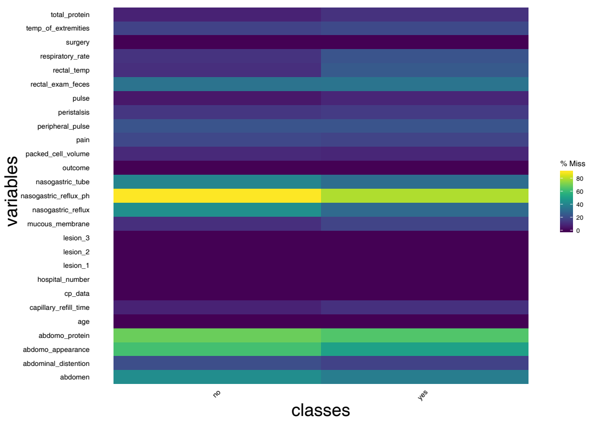
Most clustering algorithms handle either only categorical data or numeric data. Before using such type of algorithms, data preprocess such as discretization or one-hot encoding is performed to convert the numeric data to categorical data, and vice versa. However, the discretization process leads to loss of information because the membership degree of a value to discretized values is not taken into account, while it is challenging to give correct numeric values to categorical values 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 one, two and three will not be appropriate since these values have the same order. Moreover, the inappropriate transformation may impair the information inherent in the values, resulting in misleading outcomes [6, 57, 80].

In the era of information technology, data can be collected from various sources such as sensors, digital devices, machines, and humans. These sources have generated large amounts of data in a short time. However, collecting data is not always an easy task and may lead to missing values in data (simply called missing data or missing values) due to different mechanisms. Missing data can be caused by human error, equipment malfunctions, system generated errors, inconsistency between other recorded data thus deleted, data not entered due to misunderstanding, certain data not be considered significant at the time of entry and changes of the data. Missing data also occurs due to observing conditions, instrument sensitivity limitations, and other real-life considerations. Unfortunately, these may hide the correct answer underlying the data. Furthermore, they may reduce the performance of algorithms. This forces researchers who want to use a statistical analysis that requires complete data to choose between imputing data or discarding missing values. However, simply discarding missing data is not a reasonable practice, as valuable information may be lost and inferential power compromised. It can even cause selection bias in some cases. In addition, deleting observations with missing values may result in very few observations remaining in the data when there is a large number of predictive variables that contain missing values. As a result, it is advisable to impute the data before any analysis is performed [111].

Many data sets from the UCI Machine Learning Repository [34] contain missing values. Fig. 5.2 shows the Hepatitis and Horse colic data sets with the missing rates are approximately 6% and 20%, respectively. In addition, existing frameworks and packages for clustering categorical and mixed data strongly recommend that data sets should be filled in a way that makes sense for the problem at hand, especially in the case of many missing values [23]. The above observations motivated the design of an



(a) Hepatitis data set



(b) Horse colic data set

Figure 5.2: data sets with missing values

algorithm that can cluster mixed numeric and categorical data having missing values. In general, existing algorithms use two approaches for clustering missing data. For the first approach, data sets are pre-processed so that these do not contain any missing values and then inputted to the clustering algorithms. For the second approach, clustering algorithms are designed to work directly with incomplete data sets. In this research, we used the second approach for the proposed algorithm. Specifically, we embedded the imputation process into the clustering process so that users do not need to preprocess data sets such as filling missing values and discretization the data beforehand. The contribution of this chapter is to design a new framework for solving the problem of clustering missing mixed data. It then proposes an algorithm that takes advantage of missing values imputation to improve the quality of clustering results.

This chapter is structured as follows. Section 2 reviews previous works related to clustering and missing values imputation. Section 3 gives the preliminaries and definitions. Section 4 describes the proposed k-CMM algorithm. Section 5 discusses the experimental results. Finally, section 6 draws a conclusion and outlines directions for future work.

5.2 Literature review and related work

5.2.1 Partitional clustering algorithms for categorical and mixed data

This section introduces a brief review of methods for clustering categorical and mixed data. The details are provided in section 2.4. K-means algorithm [84] splits a numerical data set into a predetermined number k of clusters. This algorithm consists of two phases: initialization phase and iteration phase. In the initialization phase, the algorithm randomly assigns objects into k clusters. In the iteration phase, it uses the Euclidean to compute the distance between objects and cluster centers and then assigns them into the nearest clusters. The main steps of k-means algorithm are as follows. It first randomly selects k of the objects in a data set D_{num}, 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 the within-cluster variation. For each cluster, it computes the new mean using the objects assigned to the cluster in the previous iteration. All the objects are then reassigned using the updated means as the new cluster centers. The iterations continue until the assignment is stable, i.e., the clusters formed in the current round are the same as those formed in the previous round [53]. K-means algorithm has some remarkable properties: it is efficient in clustering large data sets, since its computational complexity is linearly proportional to the size of the data sets; it often terminates at a local optimum; the clusters have convex shapes, such as a ball in three-dimensional space, the performance is dependent on the initialization of the centers [43]. However, working only on numerical data prohibits some applications of k-means algorithm where categorical and mixed data are involved. The traditional approach to converting categorical data into numeric values does not produce meaningful results in the case where categorical domains are not ordered [60].

Several k-means-like algorithms have been proposed to address the limitation of k-means algorithm [20, 26, 59, 68, 90, 92, 99]. K-modes algorithm [59] can be considered as a pioneering work for clustering categorical data. The main idea of this algorithm is to specify the number of clusters and then to select k initial modes, followed by allocating every object to the nearest mode. It uses the modes to represent clusters and a frequency-based method to update 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's domain set appearing in the cluster. Since k-modes comes from k-means algorithm, it can also be treated as an optimization problem. It also has the same properties as those of k-means algorithm, except that it works only on categorical data.

In 2004, San et al. proposed k-representatives algorithm [99]. The Cartesian product and union operations are utilized to form centers of clusters. It uses the relative frequencies of categorical values within the cluster and the simple matching measure between categorical values to quantify the distance between objects. The main procedure of k-representatives is performed in the same manner as k-modes algorithm. K-populations algorithm [72] extends k-modes by using the notion of population that describes the information distributed in the cluster to represent the centroid of each cluster. The population is a set of pairs that contain categorical values and their confidence degrees for each attribute. The soft decision minimizes the uncertainty and imprecision in the representation of cluster centroids.

In 2013, Chen et al. proposed k-centers algorithm [20]. It uses the kernel density estimation approach to form centers of clusters. Moreover, to measure the individual contribution of each attribute for clusters, it uses a built-in feature weighting in which categorical attributes are assigned with weights. The dissimilarity is measured by using the simple matching as an indicator function to represent each data object by a set of vectors and the Euclidean norm. In 2016, Nguyen et al. proposed a new extension of the k-means clustering algorithm 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 [91, 92]. More specifically, it extends the kernel function for cluster centers (proposed in k-centers algorithm) regarding the cardinality of an attribute's sub-domain of a given cluster instead of using the whole domain of attribute.

For clustering mixed numeric and categorical data, Huang also proposed k-prototypes algorithm [60]. The clustering process of k-prototypes algorithm is similar to k-means algorithm except that it uses k-modes approach to updating the categorical attribute values of cluster prototypes. To quantify the distance between objects, it uses the squared Euclidean distance and the simple matching for numeric attributes and categorical attributes, respectively. Several works [6, 66, 80] was then proposed by improving k-prototypes algorithm. Given a set of n mixed objects, which are described by attributes $\mathcal{A}_1^r, \mathcal{A}_2^r, \ldots, \mathcal{A}_p^r, \mathcal{A}_{p+1}^c, \ldots, \mathcal{A}_m^c$. The objective function for k-prototypes is as follows:

$$\mathcal{F}(U, Z) = \sum_{l=1}^{k} (\mathcal{F}_l^r + \mathcal{F}_l^c) \tag{5.1}$$

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

5.2.2 Imputation methods for categorical and mixed data with missing values

Missing data are common in real-life applications. It is worth to remind that there are three basic types of missing data [82]: missing completely at random (MCAR), missing at random (MAR) and not missing at random (NMAR). Methods to handle missing data also have a large variation, and these choices are often made based on the missing data mechanism. The methods to deal with missing values can be classified into two groups: pre-replacing methods that replace missing values before the data mining process, and embedded methods that deal with missing values during the data mining process [42]. Several methods have been proposed to deal with missing data such as case deletion methods, imputation methods and non-imputation methods [1].

In general, missing data imputation is defined as a procedure of completing the missing values with plausible values that are estimated based on the observed data (called complete data) in the given data set [121]. Missing data imputation has attracted more attention of researchers in the past few years. Many methods have been proposed to replace missing values. Table 5.1 shows a brief literature review of imputation methods for numeric and categorical data.

Table 5.1: The literature review of imputation methods

Method	Type of data	Cost	Related sources
Mean/mode-based imputation	numeric, categorical, mixed	low	[42]
Regression-based imputation	numeric, categorical	low	[47, 104, 123]
Decision tree-based imputation	categorical, mixed	middle	[24, 96]
Nearest neighbor-based imputation	numeric, categorical, mixed	high	[22, 46, 67, 121]
ANN-based imputation	numeric	high	[45, 103, 104]
Clustering-based imputation	numeric, categorical, mixed	high	[12, 42, 67, 122]
Hybrid model-based imputation	numeric	high	[12, 100]

The simplest imputation method is to replace missing values with the mean and mode of the known values in the complete instances for missing numeric and categorical attribute, respectively. One of the most popular techniques in the statistics-based imputation method is regression-based imputation. Each missing attribute is approximated with the regression function. Several methods have been used to impute missing values such as multiple linear regression and logistic regression for numeric data, multinomial logistic regression for categorical data with more than two categorical values [104].

Decision tree (DT) based imputation is a method to replace missing values in categorical and mixed data. For numeric attributes in mixed data, this method quantizes them by the square root of their domain size or using discretization. The DT based method schemes to first partition a full data set into two sub-datasets: incomplete data set that contains data objects with any missing values and complete data set that contains data objects without any missing values. Several DTs are then built in the complete data set using the attributes having missing values in the incomplete data set as class attributes. Afterward, each data object in the incomplete data set is assigned to the leaf nodes of the corresponding DTs. Finally, missing values are imputed using the information from the correlated data objects in the leaf nodes of DTs.

The nearest neighbor based imputation (NNI) fills in missing values in attributes of an object by using the information of corresponding attributes of its k nearest neighbors. The mean value and the most frequent value among all neighbors are usually used to impute for missing values in numeric and categorical attributes, respectively. This method can be used for both numeric and categorical data by using an appropriate distance function. The cost of the NNI method can be expensive for large data set since it must perform an exhaustive search in the data set for each missing data object [100].

The artificial neural network is another method for handling missing values. Especially, the multilayer perceptron neural network (MLP) has been widely used for this method. The most common way is to build the training process on the whole portion of the input data set. The generated neural network then imputes missing values using the trained network. There are several ways to train the model. The first way is to build the training process for each attribute on the complete data. Particularly, for each attribute, networks are trained on data objects for which the target value is not missing and then the generated networks are used to fill in missing values. The second way is to build the training process of the network with the entire data set, using both complete and missing data. In this way, the numbers of neurons in the input and output layers of MLP are the same as the number of attributes in data set. MLP based imputation is mainly used for imputing numeric data in which numeric attributes are normalized to ensure that all attributes have the same importance in the model. The MLP based imputation can be used for categorical data. However, categorical attributes must be codified by vectors of dummy variables zero and one [104].

Clustering can be used for missing value imputation. The main idea of the clustering-based imputation is to assign both missing and complete data objects into clusters. From that, each missing data object is imputed by using the information of similar objects to that object in the same cluster. The other way is to combine clustering and nearest neighbors to cluster data and then select the closest neighbor with the missing object from centroids of resulted clusters. Missing values are imputed with the values from corresponding attributes in a selected neighbor [67]. The fuzzy clustering also can be used for imputation purposes. Specifically, fuzzy clustering such as the c-means is applied to input data to separate data objects into clusters. Note that, in fuzzy clustering, each data object has a degree of membership between zero and one indicating which clusters it belongs to. Missing values of a data object are then imputed with plausible values that are generated by applying regression models to the data objects belong to a set of fuzzy clusters [100].

Hybrid-based imputation is a technique that combines several methods in a model to improve the quality of imputation. Aydilek et al. combined the fuzzy c-means, support vector regression and genetic algorithm to estimate missing values [12]. Sefidian et al. combined the fuzzy c-means and the local mutual information based feature selection to select only highly relevant features in each cluster. The regression model is then applied to the selected features to estimate the missing values [100]. Moreover, other imputation methods such as expectation maximization based imputation, genetic algorithm based imputation, fuzzy-rough sets based imputation and correlation-based imputation can be used for imputing missing values. The cost of imputation methods in Table 5.1 is referenced from [42, 100].

Recently, Deb et al. proposed the DSMI algorithm [24]. This algorithm is based on the procedure of the decision tree-based imputation that has been mentioned above to handle missing values. After all objects in the incomplete data set are assigned into corresponding DTs, correlated data objects are identified from leaf nodes of DTs. For each data object in the incomplete data set, a table is populated using this object and correlated data objects from the complete data set. Candidates for missing values to be imputed are extracted from the complete data objects of each table. If there are multiple candidates, DSMI uses the IS measure and the Weighted Similarity measure (WSM) to compute the correlation between each candidate and the missing object. Finally, it uses random sampling from the list of candidates based on their degree of affinity to choose actual imputed values for imputation. The DSMI is efficient for imputing missing values in traffic accident data. However, two limitations can be observed from this algorithm. The first problem is that the DSMI focuses mainly on categorical attributes and it is not clear how to handle numeric attributes. The second problem is that the computational cost of constructing graphs and calculating WSM is very high when dealing with high-dimensional data sets or data sets containing multiple missing values. Our imputation method is based upon the DSMI algorithm. However, we modified it to address the above limitations and then integrated the imputation step into the clustering step to result in an algorithm that can cluster categorical or mixed data with missing values. The following sections define the problem of clustering mixed data with missing values and then propose an algorithm named k-CMM.

5.3 Preliminaries and problem statement

The problem of clustering mixed data has been the subject of several prior studies [60, 66, 80]. Let $\mathcal{A}^r = \{\mathcal{A}^r_1, \mathcal{A}^r_2, \ldots, \mathcal{A}^r_p\}$ be the set of p numeric attributes where the domain of $\mathcal{A}^r_\alpha$ $(1 \leq \alpha \leq p)$, denoted by DOM($\mathcal{A}^r_\alpha$), is represented by continuous values. Let $\mathcal{A}^c = \{\mathcal{A}^c_1, \mathcal{A}^c_2, \ldots, \mathcal{A}^c_q\}$ be the set of q categorical attributes where the domain of $\mathcal{A}^c_\beta$ $(1 \leq \beta \leq q)$, denoted by DOM($\mathcal{A}^c_\beta$), is represented by a finite and unordered set that contains only singletons. Let $\mathcal{A}$ =$\{\mathcal{A}_1, \mathcal{A}_2, \ldots, A_m\}$ such that $\mathcal{A} = \mathcal{A}^r \cup \mathcal{A}^c$ and $m = p+q$ be the set of m distinct attributes where the j^{th} attribute $\mathcal{A}_j$ $(1 \leq j \leq m)$ is either a numeric attribute or a categorical attribute. A mixed numeric and categorical object (mixed object or mixed record) is a tuple of the form $\langle id, x \rangle$ where id is its unique identifier and x is represented by a tuple $t \in$ DOM($\mathcal{A}_1$) $\times$ DOM($\mathcal{A}_2$) $\times \cdots \times$ DOM(A_m). For the simplicity, a mixed object x having $id = i$ is denoted as x_i. A mixed data set $\mathcal{D}_{mix} = \{x_1, x_2, \ldots, x_n\}$ is a set of n mixed objects where $x_i = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ is a set of m mixed numeric and categorical values at the i^{th} element of $\mathcal{D}_{mix}$. In other words, $\mathcal{D}_{mix}$ is an $n \times m$ matrix $(n \gg m)$, where n and m are the number of objects and attributes in data set $\mathcal{D}_{mix}$, respectively. The element at position (i, j) $(1 \leq i \leq n$, $1 \leq j \leq m)$ of the matrix stores the value of the object i^{th} at the attribute j^{th}, such that $x_{ij} \in$ DOM($\mathcal{A}$) = DOM($\mathcal{A}^r$) $\cup$ DOM($\mathcal{A}^c$). Note that if a categorical value in $\mathcal{D}_{mix}$ is a missing value, then it is represented as "?" or "" (empty). For example, Table 5.3 shows a mixed data set with six attributes: the first four attributes are categorical, while the last two attributes are numeric. It contains fifteen objects with eight missing values and used for the running example. For the sake of brevity, a mixed object with and with no missing value is denoted as the incomplete and complete object, respectively; while a mixed data set with and with no missing values is denoted as the incomplete and complete data set, respectively.

Definition 13 (Clusters) Let there be a mixed data set $\mathcal{D}_{mix}$ and a set $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots,$ $\mathcal{C}_k\}$ that contains k disjoint subsets. $\mathcal{C}_l$ $(1 \leq l \leq k)$ is called a cluster of $\mathcal{D}_{mix}$ iff for every $\mathcal{C}_i \in \mathcal{C}$ $(1 \leq i \leq k \wedge i \neq l)$, $\mathcal{C}_l \cap \mathcal{C}_i = \emptyset$ and $\mathcal{D}_{mix} = \bigcup_{l=1}^{k} \mathcal{C}_l$. The number of data objects in the cluster $\mathcal{C}_l$ is denoted by n_l.

Example 6 In the data set shown in Table 5.3, assume that there are three subsets of

Table 5.2: Table of notation

Symbol	Description
k	number of clusters
x_i	mixed object with index i
x_i^r	numeric object with index i
x_i^c	categorical object with index i
$\mathcal{X}_j$	random variable
$\mathcal{D}_{mix}$	mixed data set
$\mathcal{A}_j$	j^{th} attribute of $\mathcal{D}_{mix}$
$\mathcal{A}_j^r$	j^{th} numeric attribute of $\mathcal{D}_{mix}$
$\mathcal{A}_j^c$	j^{th} categorical attribute of $\mathcal{D}_{mix}$
$\mathcal{C}_l$	l^{th} cluster
$\mathcal{Z}_l$	center of cluster $\mathcal{C}_l$
x_{ij}	value appears at the i^{th} element and j^{th} attribute of $\mathcal{D}_{mix}$
$\bar{o}_{ij}^l$	numeric value appears at the i^{th} element and j^{th} attribute of cluster $\mathcal{C}_l$
o_{ij}^l	categorical value appears at the i^{th} element and j^{th} attribute of cluster $\mathcal{C}_l$
$\mathcal{O}_j$	set of categorical values appears at the j^{th} attribute of $\mathcal{D}_{mix}$
$\mathcal{O}_j^l$	set of categorical values appears at the j^{th} attribute of cluster $\mathcal{C}_l$
z_j^{rl}	value of the j^{th} numeric attribute in the center $\mathcal{Z}_l$
z_j^{cl}	value of the j^{th} categorical attribute in the center $\mathcal{Z}_l$

Table 5.3: A mixed numeric and categorical data set with missing values

Attribute / Object	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_6$
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

$\mathcal{D}_{mix}$: $\mathcal{C}_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$, $\mathcal{C}_2 = \{ x_4, x_6, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14} \}$, $\mathcal{C}_3 = \{ x_1, x_2, x_3 \}$. Then, $\{\mathcal{C}_1, \mathcal{C}_2\}$ are clusters of $\mathcal{D}_{mix}$, while $\{\mathcal{C}_1, \mathcal{C}_3\}$, $\{\mathcal{C}_2, \mathcal{C}_3\}$ and $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ are not.

Definition 14 (Relative frequency of a categorical value) Let there be a cluster $\mathcal{C}_l$ and a categorical value o^l_{ij} appearing in $\mathcal{C}_l$ at the j^{th} categorical attribute, the relative frequency of o^l_{ij} in $\mathcal{C}_l$ is denoted and defined as:

$$\mathfrak{f}_{\mathcal{C}_l}(o^l_{ij}) = \frac{\#_l(o^l_{ij})}{n_l} \tag{5.2}$$

where $\#_l(o^l_{ij})$ is the number of categorical values o^l_{ij} appearing in the cluster $\mathcal{C}_l$ at the j^{th} attribute.

The relative frequency of o^l_{ij} in the data set $\mathcal{D}_{mix}$ at the j^{th} categorical attribute is denoted and defined as:

$$\mathfrak{f}(o^l_{ij}) = \frac{\#(o^l_{ij})}{|\mathcal{D}_{mix}|} \tag{5.3}$$

where $\#(o^l_{ij})$ is the number of categorical values o^l_{ij} appearing in data set $\mathcal{D}_{mix}$ at the j^{th} attribute.

Example 7 In the data set shown in Table 5.3, assume that cluster $\mathcal{C}_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$, then the relative frequency of the categorical value "b" in the attribute $\mathcal{A}_1$ is $\mathfrak{f}_{\mathcal{C}_1}(b) = \frac{3}{6} = 0.5$.

In this chapter, to represent the center of a cluster, we used the *mean* for numeric attributes and the variation on Aitchison & Aitken's kernel function [8] to estimate the probability density function of each categorical attribute in the center.

Definition 15 (The mean of numeric attributes in a cluster) Let there be a cluster $\mathcal{C}_l$ that contains p numeric attributes $\mathcal{A}^r = \{\mathcal{A}^r_1, \mathcal{A}^r_2, \ldots, \mathcal{A}^r_p\}$ and $\mathcal{Z}_l$ is the center of $\mathcal{C}_l$. The mean of each attribute $\mathcal{A}^r_j$ $(1 \leq j \leq p, p < m)$ in the cluster $\mathcal{C}_l$ is defined and denoted as:

$$z^{rl}_j = \frac{1}{n_l}\sum_{i=1}^{n_l} \bar{o}^l_{ij} \tag{5.4}$$

Example 8 In the data set shown in Table 5.3, assume that cluster $\mathcal{C}_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$, then mean value of the attribute $\mathcal{A}_6$ is $z^{r1}_6 = \frac{(14+13+13+14+14+14)}{6} = 13.67$.

Recall that a density estimator is an algorithm which takes a d-dimensional data set and produces an estimate of the d-dimensional probability distribution which that data is drawn from [114]. Kernel density estimation (KDE) is a method of estimating the probability distribution of a random variable based on a random sample.

Definition 16 (Center of cluster) Let there be a cluster $\mathcal{C}_l = \{x_1, x_2, \ldots, x_{n_l}\}$ where $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, $m = |\mathcal{A}|$. The center of $\mathcal{C}_l$ is then defined as:

$$\mathcal{Z}_l = \{z^l_1, z^l_2, \ldots, z^l_m\} \tag{5.5}$$

where the j^{th} attribute z^l_j $(1 \leq j \leq m)$ is either z^{rl}_j or z^{cl}_j. Specifically, if the j^{th} attribute of $\mathcal{Z}_l$ is a numeric attribute, then its representative is calculated by using the Definition 15 (Eq. 5.4). Otherwise, the representative of a categorical attribute of $\mathcal{Z}_l$ is the probability distribution on $\mathcal{O}^l_j$ estimated by the kernel density estimation method using Eq. (2.4), which is defined as:

$$z^{cl}_j = [\mathcal{P}^l_j(o^l_{1j}), \mathcal{P}^l_j(o^l_{2j}), \ldots, \mathcal{P}^l_j(o^l_{|\mathcal{O}^l_j|j})] \tag{5.6}$$

where the probabilistic value of a categorical value o^l_{ij} $(1 \leq l \leq n_l)$ can be estimated as:

$$\mathcal{P}^l_j(o^l_{ij}) = \begin{cases} \lambda_j \frac{1}{|\mathcal{O}^l_j|} + (1-\lambda_j)\mathfrak{f}_{\mathcal{C}_l}(o^l_{ij}) & \text{if } o^l_{ij} \in \mathcal{O}^l_j \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

Example 9 In the data set shown in Table 5.3, assume that cluster $\mathcal{C}_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$. Then the center $\mathcal{Z}_1$ of $\mathcal{C}_1$ at the categorical attributes $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$ and numeric attributes $\mathcal{A}_5$, $\mathcal{A}_6$ 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, 14.1111, respectively.

Previous studies have used several methods to quantify the dissimilarity between a mixed object and its center [60, 66, 80]. Particularly, distance measures such as the Euclidean, Manhattan, Minkowski, and Mahalanobis [43] can be applied for numeric attributes, while the simple matching dissimilarity measure [58–60, 99], the Euclidean norm [20] and the information-theoretic based dissimilarity measure [92] can be applied for categorical attributes. In this chapter, we used the squared Euclidean and the information-theoretic based dissimilarity measure to quantify the dissimilarity between numeric and categorical attributes in mixed objects, respectively. The information-theoretic definition of similarity [81] is applicable for domains that have probabilistic models.

Definition 17 (Dissimilarity measure for categorical attributes) Let there be two categorical values o^l_{ij} and $o^l_{i'j}$ at the j^{th} attribute. The similarity between them is defined as:

$$\text{sim}_j(o^l_{ij}, o^l_{i'j}) = \frac{2\log \mathfrak{f}(o^l_{ij}, o^l_{i'j})}{\log \mathfrak{f}(o^l_{ij}) + \log \mathfrak{f}(o^l_{i'j})} \tag{5.8}$$

where $\mathfrak{f}(o^l_{ij}, o^l_{i'j}) = \frac{\#(o^l_{ij}, o^l_{i'j})}{|\mathcal{D}_{mix}|}$ with $\#(o^l_{ij}, o^l_{i'j})$ is the number of mixed objects in data set $\mathcal{D}_{mix}$ that receives the categorical values belonging to $\{o^l_{ij}, o^l_{i'j}\}$ at the j^{th} attribute, while $\mathfrak{f}(o^l_{ij})$ is measured by the Eq. 5.3.

The dissimilarity measure between two categorical values o^l_{ij} and $o^l_{i'j}$ at the j^{th} attribute can be defined as:

$$\text{dsim}_j(o^l_{ij}, o^l_{i'j)}) = 1 - \text{sim}_j(o^l_{ij}, o^l_{i'j}) = 1 - \frac{2\log \mathfrak{f}(o^l_{ij}, o^l_{i'j})}{\log \mathfrak{f}(o^l_{ij}) + \log \mathfrak{f}(o^l_{i'j})} \tag{5.9}$$

Example 10 In the data set shown in Table 5.3, we omit four incomplete objects from the data set. The dissimilarity of categorical values “d” and “b” in the attribute $\mathcal{A}_1$ is $\text{dsim}_1(\text{“d”,“b”}) = 1 - \frac{2\times\log(\frac{6}{11})}{\log(\frac{3}{11})+\log(\frac{3}{11})} = 0.5335$.

Definition 18 (Dissimilarity between an object and a cluster) Let there be a cluster $\mathcal{C}_l$ and a mixed object $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$. The dissimilarity between x_i and the center $\mathcal{Z}_l = \{z^l_1, z^l_2, \ldots, z^l_m\}$ at the j^{th} attribute is defined as:

$$\text{dis}_j(x_i, \mathcal{Z}_l) = \begin{cases} (x_{ij} - z^{rl}_j)^2 & \text{if } \mathcal{A}_j \text{ is a numeric attribute} \\ \sum_{o^l_{ij} \in \mathcal{O}^l_j} \mathcal{P}^l_j(o^l_{ij})\text{dsim}_j(x_{ij}, o^l_{ij}) & \text{if } \mathcal{A}_j \text{ is a categorical attribute} \end{cases} \tag{5.10}$$

Specifically, the dissimilarity between x_i and $\mathcal{Z}_l$ at the j^{th} attribute is measured based on the type of this attribute. For numeric attributes, the squared Euclidean distance is used to quantify the distance between the mean of clusters and the numeric value of mixed objects. For categorical attributes, the proximity is measured by accumulating the probability distribution on $\mathcal{O}^l_j$ and the dissimilarity between j^{th} component x_{ij} of the object x_i and the j^{th} component z^{cl}_j of the center $\mathcal{Z}_l$. Finally, the dissimilarity between mixed object x_i and cluster center $\mathcal{Z}_l$ is defined as follows:

$$\text{dis}(x_i, \mathcal{Z}_l) = \sum_{j=1}^{m} \text{dis}_j(x_i, \mathcal{Z}_l) \tag{5.11}$$

Example 11 In the data set shown in Table 5.3, assume that cluster $\mathcal{C}_1 = \{x_1, x_2, x_3, x_5, x_7, x_{15}\}$ and its cluster $\mathcal{Z}_1$ is $\Big\{${“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, 14.1111$\Big\}$. The dissimilarity between x_1 = {d, b, f, e, 7, 14} and $\mathcal{Z}_1$ is dis$(x_1, \mathcal{Z}_1) = 0.3818 + 0.5 + 0.3155 + 0.2489 + 11.8642 + 0.0123 = 13.3227$.

In the following, two measures for the imputation step are presented. The first measure is the IS measure (ISM). It is used to evaluate the degree of associations between two sets of categorical values in a data object. The ISM was first introduced by Tan et al. [107]. It contains the product of two quantities: interest factor and support count that compute the correlations between values of different attributes in an object as defined in Eq. 4.2.

Definition 19 (IS measure [24]) Let there be a set T that contains both complete and incomplete categorical objects. Let $\mathcal{A}' = \{\mathcal{A}'_1, \mathcal{A}'_2, \ldots, \mathcal{A}'_{m'}\}$ and $\mathcal{A}'' = \{\mathcal{A}''_1, \mathcal{A}''_2, \ldots, \mathcal{A}''_{m''}\}$ $(\mathcal{A}', \mathcal{A}'' \subset A; m', m'' < m)$ be two sets of categorical attributes that contain missing values and non-missing values in T, respectively. For all $a' = \{a'_1, a'_2, \ldots, a'_n\} \in \mathcal{A}'_1 \times \mathcal{A}'_2 \times \cdots \times \mathcal{A}'_{m'}$ and $a'' = \{a''_1, a''_2, \ldots, a''_n\} \in \mathcal{A}''_1 \times \mathcal{A}''_2 \times \cdots \times \mathcal{A}''_{m''}$, the IS measure between a' and a'' is defined as:

$$\text{IS}(a', a'') = \frac{\text{Support}(a', a'')}{\sqrt{\text{Support}(a') \times \text{Support}(a'')}} \tag{5.12}$$

Where 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 extend the information-theoretic based similarity measure Eq. (5.8) to make it applicable for measuring the proximity of complete and incomplete objects.

Definition 20 (Missing-complete similarity measure (MCS measure)) Let there be a set T that contains both complete and incomplete objects. Let there be two categorical values o^l_{ij} and $o^l_{i'j}$ appearing in T at the j^{th} attribute. The similarity between them is defined as:

$$\text{sim}^{mis}_j(o^l_{ij}, o^l_{i'j}) = \begin{cases} \frac{2\log \mathfrak{f}_T(o^l_{ij}, o^l_{i'j})}{\log \mathfrak{f}_T(o^l_{ij}) + \log \mathfrak{f}_T(o^l_{i'j})} & \text{if } o^l_{ij} \neq ? \textit{ and } o^l_{i'j} \neq ? \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

where $\mathfrak{f}_T(o^l_{ij}) = \frac{\#_T(o^l_{ij})}{n_T}$, $\#_T(o^l_{ij})$ denotes the number of o^l_{ij} 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 follows :

$$\text{MCS}(x_i, x_{i'}) = \sum_{j=1}^{m} \text{sim}^{mis}_j(x_{id}, x_{i'd}) \tag{5.14}$$

where the j^{th} attributes of x_i and $x_{i'}$ are categorical attributes.

Example 12 This example illustrates how to impute missing values in categorical attributes using the IS and MCS measures. The subset that contains complete categorical objects extracting from Table 5.3 is shown in Table 5.4. Assume that the incomplete

Table 5.4: The complete categorical data set extracting from Table 5.3

Object \ Attribute	$\mathcal{A}^c_1$	$\mathcal{A}^c_2$	$\mathcal{A}^c_3$	$\mathcal{A}^c_4$
x^c_1	d	b	f	e
x^c_2	b	b	c	e
x^c_3	b	b	c	e
x^c_4	a	b	a	b
x^c_5	c	a	f	d
x^c_7	a	a	c	e
x^c_9	c	b	a	c
x^c_{11}	d	a	d	c
x^c_{12}	d	b	d	d
x^c_{14}	a	a	d	d
x^c_{15}	b	a	f	e

categorical object $x^c_6 = \langle d, b, f, ? \rangle$ in Table 5.3 is chosen for imputation. The DT that uses the attribute $\mathcal{A}_4^c$ as the class attribute is built for x_6^c based on the data set in Table 5.4 (Fig. 5.3). The object x_6^c is then assigned to leaf 7. The set of complete categorical

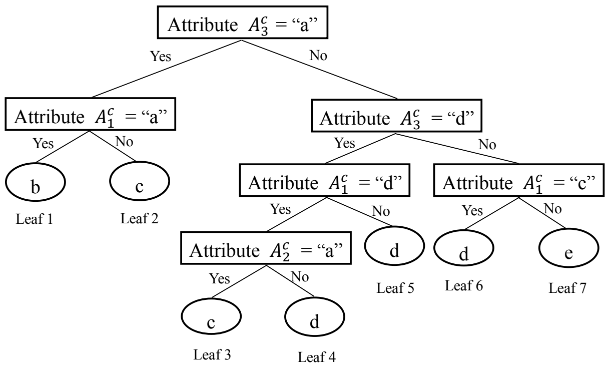
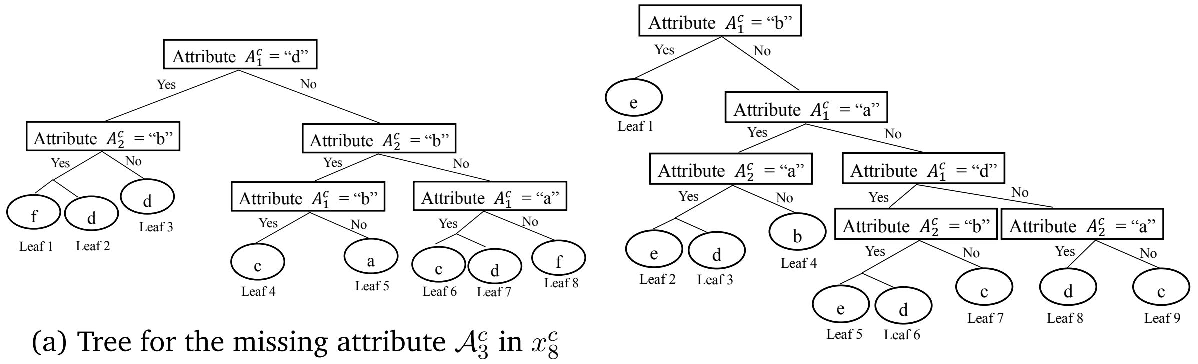


Figure 5.3: Tree for the missing attribute $\mathcal{A}_4^c$ in x_6^c

objects that are correlated with x_6^c are $\{x_1^c, x_2^c, x_3^c, x_7^c, x_{15}^c\}$. The set of categorical values in the complete attributes $\mathcal{A}_1^c$, $\mathcal{A}_2^c$, $\mathcal{A}_3^c$ contains $\{d, b, f\}$, $\{b, b, c\}$, $\{a, a, c\}$ and $\{b, a, f\}$. The set of categorical values in the incomplete attribute $\mathcal{A}_4^c$ contains $\{e\}$. The possible imputed value is only e. Thus it is chosen for imputing the missing value in x_6^c, i.e., $x_6^c = \langle d, b, f, e \rangle$.

Next, assume that the incomplete categorical object $x_8^c = \langle b, a, ?, ? \rangle$ in Table 5.3 is chosen for imputation. There are two missing values in x_8^c at the attributes $\mathcal{A}_3^c$ and $\mathcal{A}_4^c$. Therefore, two DTs that use the attribute $\mathcal{A}_3^c$ and $\mathcal{A}_4^c$ as the class attributes are built for x_8^c based on the data set in Table 5.4 (Fig. 5.4), respectively. For the missing value in



(a) Tree for the missing attribute $\mathcal{A}_3^c$ in x_8^c

(b) Tree for the missing attribute $\mathcal{A}_4^c$ in x_8^c

Figure 5.4: Trees for the incomplete categorical object x_8^c

the attribute $\mathcal{A}_3^c$, the object x_8^c is assigned to leaf 8 of the tree 5.4a. The set of complete categorical objects that are correlated with x_8^c contains $\{x_5^c, x_{15}^c\}$. For the missing value in the attribute $\mathcal{A}_4^c$, the object x_8^c is assigned to leaf 1 of the tree 5.4b. The set of complete categorical objects that are correlated with x_8^c contains $\{x_2^c, x_3^c, x_{15}^c\}$. Because x_8^c falls into multiple leaves, the objects from all these leaves are grouped in one collection. Thus, the set of correlated objects are $\{x_2^c, x_3^c, x_5^c, x_{15}^c\}$. The set of categorical values in the complete attributes $\mathcal{A}_1^c$, $\mathcal{A}_2^c$ contains $\{b, b\}$, $\{c, a\}$, $\{b, a\}$, while the set of categorical values in the incomplete attribute $\mathcal{A}_3^c$ and $\mathcal{A}_4^c$ contains $\{c, e\}$, $\{f, d\}$ and $\{f, e\}$. The IS and MCS measures for each pair of categorical values in the complete attributes and incomplete attributes are: IS($\{b, b\}$,$\{c, e\}$) = $\frac{0.5}{\sqrt{0.5\times0.5}}$ = 1, IS($\{c, a\}$,$\{f, d\}$) = $\frac{0.25}{\sqrt{0.25\times0.25}}$ = 1, IS($\{b, a\}$,$\{f, e\}$) = $\frac{0.25}{\sqrt{0.25\times0.25}}$ = 1, MCS(x_2^c, x_8^c) = MCS(x_3^c, x_8^c) = $\frac{2\log 0.8}{\log 0.8+\log 0.8}$ + $\frac{2\log 1}{\log 0.4+\log 0.6}$ + 0 + 0 = 1, MCS(x_5^c, x_8^c) = $\frac{2\log 1}{\log 0.2+\log 0.8}$ + $\frac{2\log 0.6}{\log 0.6+\log 0.6}$ + 0 + 0 = 1, MCS(x_{15}^c, x_8^c) = $\frac{2\log 0.8}{\log 0.8+\log 0.8}$ + $\frac{2\log 0.6}{\log 0.6+\log 0.6}$ + 0 + 0 = 2. The affinity degree of possible imputed values is calculated by the average of the IS and MCS measures for each pair of categorical values in the complete attributes and incomplete attributes. Thus, $\delta(\{c, e\})$ = (1+1)/2 = 1, $\delta(\{f, d\})$ = (1+1)/2 = 1, $\delta(\{f, e\})$ = (1+2)/2 = 1.5. The actual imputed values is chosen by random sampling according to the affinity degrees. Specifically, the sampling probabilities of $\{c, e\}$, $\{f, d\}$, $\{f, e\}$ are 0.2857, 0.2857 and 0.4286, respectively. Thus, the $\{f, e\}$ has its probability of been chosen as the actual imputed values for x_8^c.

Based on the above definitions, the problem of clustering for mixed numeric and categorical data sets with missing values aims to minimize the following objective function:

$$\mathcal{F}(\mathcal{U}, \mathcal{Z}) = \sum_{l=1}^{k}\sum_{i=1}^{n} u_{i,l} \times \text{dis}(x_i, \mathcal{Z}_l) \tag{5.15}$$

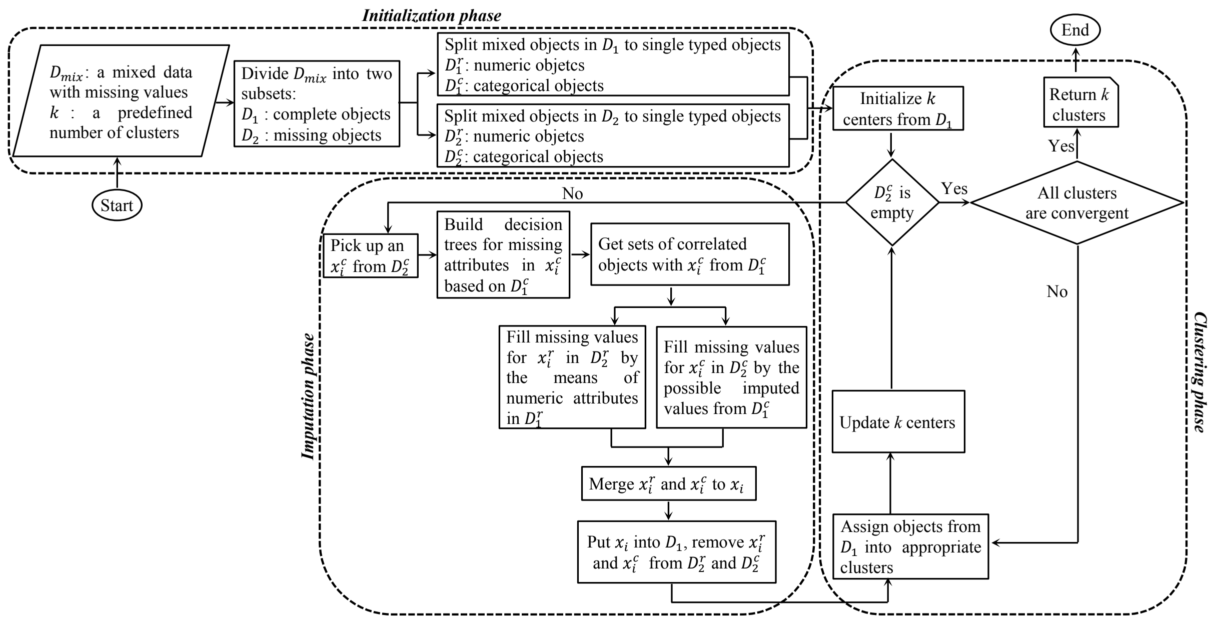
subject to

$$\begin{cases} \sum_{l=1}^{k} u_{i,l} = 1 & 1 \le i \le n \\ u_{i,l} \in \{0, 1\} & 1 \le l \le k,\ 1 \le i \le n \end{cases} \tag{5.16}$$

where $\mathcal{U} = [u_{i,l}]_{n\times k}$ is the partition matrix, $u_{i,l}$ takes value 1 if object x_i is in cluster $\mathcal{C}_l$ and 0 otherwise.

5.4 The proposed *k*-CMM algorithm

The general framework of the proposed k-CMM algorithm is represented in Fig. 5.5. According to this flowchart, k-CMM algorithm performs three phases: *initialization phase*,



Initialization phase
D_{mix}: a mixed data with missing values
k : a predefined number of clusters
Start
Divide D_{mix} into two subsets:
D_1 : complete objects
D_2 : missing objects
Split mixed objects in D_1 to single typed objects
D_1^r: numeric objetcs
D_1^c: categorical objects
Split mixed objects in D_2 to single typed objects
D_2^r: numeric objetcs
D_2^c: categorical objects
Initialize k centers from D_1
D_2^c is empty
Yes
No
All clusters are convergent
Yes
No
Return k clusters
End
Imputation phase
Pick up an x_i^c from D_2^c
Build decision trees for missing attributes in x_i^c based on D_1^c
Get sets of correlated objects with x_i^c from D_1^c
Fill missing values for x_i^r in D_2^r by the means of numeric attributes in D_1^r
Fill missing values for x_i^c in D_2^c by the possible imputed values from D_1^c
Merge x_i^r and x_i^c to x_i
Put x_i into D_1, remove x_i^r and x_i^c from D_2^r and D_2^c
Clustering phase
Update k centers
Assign objects from D_1 into appropriate clusters

Figure 5.5: The flowchart of *k*-CMM algorithm for mixed data with missing values

imputation phase and *clustering phase*. In the initialization phase, a full mixed data set with missing values is divided into two sub-datasets: complete data set and missing data set, denoted as $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Each complete object in $\mathcal{D}_1$ is then split into two sub-objects based on the type of attributes. Specifically, the algorithm separately extracts numeric values and categorical values from mixed objects and puts them into two sub-sets $\mathcal{D}_1^r$ and $\mathcal{D}_1^c$, respectively. The same step is applied for missing objects in $\mathcal{D}_2$. The numeric values and categorical values are put into two sub-sets $\mathcal{D}_2^r$ and $\mathcal{D}_2^c$, respectively. Next, the algorithm randomly initializes k clusters from the complete set $\mathcal{D}_1$. For every incomplete categorical object in the set $\mathcal{D}_2^c$, 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. Particularly, each missing attribute is considered as a class attribute to construct a DT from the complete categorical data set $\mathcal{D}_1^c$. The incomplete categorical object x_i^c is then assigned to the corresponding tree's leaf that contains a set of correlated objects with x_i^c. Once 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 that are found in the leaf node. In addition, the missing numeric object that has the same *id* as x_i^c; i.e., x_i^r in $\mathcal{D}_2^r$, is imputed by using the *means* of values in the numeric attributes of the complete numeric objects in $\mathcal{D}_1^r$ which have the corresponding *ids* with those of correlated objects obtained in the previous step. After that, the imputed objects x_i^r and x_i^c are merged to form the complete mixed numeric and categorical object, denoted as x_i. It is worth to note that the order of attributes of x_i after merging is the same as that of mixed objects in the original data set $\mathcal{D}_{mix}$. Object x_i is then put into the $\mathcal{D}_1$ data set, x_i^r and x_i^c are eliminated from the $\mathcal{D}_2^r$ and $\mathcal{D}_2^c$, respectively. Next, the algorithm performs the clustering phase by assigning objects in the $\mathcal{D}_1$ data set into appropriate clusters and updating centers of clusters. The imputation phase is repeated until the $\mathcal{D}_2^c$ is empty. If the termination condition is not met, the algorithm performs the clustering phase until all clusters are stable. Finally, it returns k clusters.

Algorithm 4: k-CMM ALGORITHM

```
input : $\mathcal{D}_{mix}$: a mixed database, $k$: a user-specified integer number to specify the
        number of clusters
output: $k$ clusters of $\mathcal{D}_{mix}$

1  Scan $\mathcal{D}_{mix}$ once to split it into two sub-datasets, $\mathcal{D}_{mix} = \mathcal{D}_1$ (complete data set) +
     $\mathcal{D}_2$ (incomplete data set)
2  Extract mixed attributes of complete objects in $\mathcal{D}_1$ to single typed objects such that
     $A = \mathcal{A}^r \cup \mathcal{A}^c$, $\mathcal{D}_1^r$ contains numeric objects and $\mathcal{D}_1^c$ contains categorical objects
3  Extract mixed attributes of missing objects in $\mathcal{D}_2$ to single typed objects such that
     $A = \mathcal{A}^r \cup \mathcal{A}^c$, $\mathcal{D}_2^r$ contains numeric objects and $\mathcal{D}_2^c$ contains categorical objects
4  Randomly initiate $k$ cluster centers from $\mathcal{D}_1$ $\mathcal{Z}^{(0)} = \{\mathcal{Z}_1^{(0)}, \ldots, \mathcal{Z}_k^{(0)}\}$
5  $U \leftarrow \emptyset$, $t = 0$
6  foreach object $x_i^c$ in $\mathcal{D}_2^c$ do
7      $IDList$, $x_i^c$ = Categorical_Imputation($x_i^c$, $\mathcal{D}_1^c$)
8      $x_i^r$ = Numeric_Imputation($x_i^r$, $\mathcal{D}_1^r$, $IDList$)
9      $x_i$ = merge $x_i^r$ and $x_i^c$
10     $\mathcal{D}_1 = \mathcal{D}_1 \cup \{x_i\}$, $\mathcal{D}_2^r = \mathcal{D}_2^r \setminus \{x_i^r\}$, $\mathcal{D}_2^c = \mathcal{D}_2^c \setminus \{x_i^c\}$
11     Keep $\mathcal{Z}^{(t)}$ fixed, generate $\mathcal{U}^{(t)}$ to minimize the distance between objects and
         cluster centers using Eq. (5.11)
12     Keep $\mathcal{U}^{(t)}$ fixed, update $\mathcal{Z}^{(t)}$ using Eq. (5.5)
13     $t = t + 1$
14 end
15 while Clusters are not convergent do
16     Keep $\mathcal{Z}^{(t)}$ fixed, generate $\mathcal{U}^{(t)}$ to minimize the distance between objects and
         cluster mode using Eq. (5.11)
17     Keep $\mathcal{U}^{(t)}$ fixed, update $\mathcal{Z}^{(t)}$ using Eq. (5.5)
18     $t = t + 1$
19 end
20 return $k$ clusters;
```

The pseudo code of k-CMM algorithm is shown in Algorithm 4. k-CMM first scans the mixed data set $\mathcal{D}_{mix}$ to divide it into two sub-datasets, namely $\mathcal{D}_1$ and $\mathcal{D}_2$, which are the complete and incomplete data sets, respectively (line 1). It then extracts separately mixed objects in $\mathcal{D}_1$ and $\mathcal{D}_2$ to single typed objects. More specifically, complete numeric objects, complete categorical objects, missing numeric objects, missing categorical objects are stored in $\mathcal{D}_1^r$, $\mathcal{D}_1^c$, $\mathcal{D}_2^r$, and $\mathcal{D}_2^c$, respectively (lines 2 and 3). Next, the algorithm randomly initiates k cluster centers from $\mathcal{D}_1$ (line 4). Each center is represented by the Eq. (5.5). Two variables U and t are created to store clusters and count for the number of iterations of the clustering process (line 5). In the next step, k-CMM scans all objects in $\mathcal{D}_2^c$ to impute missing values inside these objects and assign them into clusters (lines 6-14). For each object x_i^c in $\mathcal{D}_2^c$, k-CMM calls the Categorical_Imputation and Numeric_Imputation procedures to replace missing values in missing categorical object and missing numeric object, respectively.

The pseudo code of the Categorical_Imputation procedure is shown in Algorithm 5. The input of this procedure is a missing categorical object x_i^c and the set of complete categorical objects $\mathcal{D}_1^c$. It first creates four variables $\mathcal{A}^c$, $DTSet$, $NodeSet$, and Ω_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 the set $\mathcal{A}^c$ (line 2). For every missing attribute $\mathcal{A}_i^c$ in $\mathcal{A}^c$, the procedure checks if there exists a DT that uses the $\mathcal{A}_i^c$ as the class attribute of this DT. If there is no such DT, the procedure builds a DT by using the missing attribute $\mathcal{A}_i^c$ as the class attribute and complete categorical objects from $\mathcal{D}_2^c$ to generate decision rules. In this research, we used $C4.5$ [64] to generate the DTs. This process is performed until all missing attributes in $\mathcal{A}_i^c$ have their corresponding DTs (lines 3-8). The resulting DTs are then put into the $DTSet$. After constructing the DTs, object x_i^c is assigned to the leaf node of the tree that has 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 $\mathcal{D}_1^c$ that are correlated with x_i^c. The resulting nodes are put into the set $NodeSet$ (lines 9-12). Each leaf node in the $NodeSet$ is represented as a list of objects. If x_i^c falls into multiple leaves, the procedure will merge these leaves and group objects into one collection T (line 13). In the next step, the procedure chooses complete objects in T that have the maximum number of complete attributes in common to the x_i^c. In

Algorithm 5: CATEGORICAL_IMPUTATION PROCEDURE

```
input  : $x_i^c$: a missing categorical object, $\mathcal{D}_1^c$: 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  $\mathcal{A}^c \leftarrow \emptyset$, $DTSet \leftarrow \emptyset$, $NodeSet \leftarrow \emptyset$, $\Omega_i \leftarrow \emptyset$
2  Put attributes having missing values in $x_i^c$ to $\mathcal{A}^c$
3  foreach attribute $\mathcal{A}_i^c$ in $\mathcal{A}^c$ do
4      if $DTSet$ does not contain decision tree $DT$ for $\mathcal{A}_i^c$ then
5          Build a decision tree $DT$ that uses $\mathcal{A}_i^c$ as class attribute from $\mathcal{D}_1^c$
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     $\Omega_i \leftarrow$ possible imputed value(s) from the $i^{th}$ matched object
17     Calculate the IS measure by Eq. (5.12) for $\Omega_i$
18     Calculate the MCS measure by Eq. (5.13) between the $i^{th}$ matched object and
         $x_i^c$
19     Calculate the affinity degree $\delta_i$ for $\Omega_i$ 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 $\{\Omega_1, \ldots, \Omega_N\}$ based on the sampling probabilities specified by
     the set of affinity degrees $\{\delta_1, \ldots, \delta_N\}$
22 return $IDList$, $x_i^c$;
```

addition, the id of the most closely correlated objects with x_i^c are put into the $IDList$. This list is later used in the Numeric_Imputation procedure (line 14). The categorical values in these selected objects corresponding to the missing attributes in the x_i^c are then taken to be the possible imputed values. The procedure then calculates the IS and MCS measures for possible imputed values in each complete object, using Eqs. 5.12 and 5.13, respectively. Next, each list of possible imputed values is associated with an affinity degree given by the average of the IS and MCS values (lines 15-20). When 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 the $IDList$ and the complete x_i^c in which all missing values are imputed (line 22).

Algorithm 6: NUMERIC_IMPUTATION PROCEDURE

```
input  : $x_i^r$: a missing numeric object, $\mathcal{D}_1^r$: the set of complete numeric objects,
         $IDList$: the list of id numbers of correlated objects with $x_i^r$
output: the imputed numeric object $x_i^r$
1 $\mathcal{A}^r \leftarrow \emptyset$
2 Put attributes having missing values in $x_i^r$ to $\mathcal{A}^r$
3 Get the set of correlated numeric objects in $\mathcal{D}_1^r$ that have the id in $IDList$,
    denoted as $CorrSet$
4 foreach attribute $\mathcal{A}_i^r$ in $\mathcal{A}^r$ do
5     Replace the missing value by mean of numeric values in $\mathcal{A}_i^r$ of $CorrSet$, using
        Eq. (5.4)
6 end
7 return $x_i^r$;
```

The pseudo code of the Numeric_Imputation procedure is shown in Algorithm 6. The input of this procedure is a missing numeric object x_i^r, the set of complete numeric objects $\mathcal{D}_1^r$ and the list of *ids* of correlated objects with x_i^r that is obtained in Algorithm 5. The procedure first finds attributes that contain missing values and puts them into the set $\mathcal{A}^r$ (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 $\mathcal{A}_i^r$ in $\mathcal{A}^r$, the procedure replaces the missing values in this attribute by using the mean of numeric values appearing at the same attribute of complete objects in $CorrSet$, using Eq. 5.4 (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 such that the order of attributes of x_i is the same as that of objects in the original data set (line 9). It then adds the x_i into $\mathcal{D}_1$ and removes x_i^r and x_i^c from $\mathcal{D}_2^r$ and $\mathcal{D}_2^c$, respectively (line 10). In the next step, k-CMM assigns objects in $\mathcal{D}_1$ into appropriate clusters and updates cluster centers (lines 11-13). The algorithm works in the same manner for all incomplete objects in the $\mathcal{D}_2^c$ and $\mathcal{D}_2^r$. If the termination condition is not met, 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).

In general, the main flow of clustering missing categorical data shown in Chapter 4 resembles clustering missing mixed data, except that we change methods to represent the centers of clusters, quantify the distances and impute only for missing categorical values. It is worth to note that clustering for missing mixed data is nontrivial and more challenging than clustering for missing categorical data. It is because different attributes in mixed data need to be treated in a heterogeneous way. 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 numeric data imputation, missing categorical data imputation, numeric data clustering and categorical data clustering, while the framework for categorical data handles two tasks of clustering and imputation for categorical type.

5.5 Comparative Experiment

Experiments were performed to evaluate the performance of the proposed k-CMM on a high-performance computing servers VPC cluster [38]. Each node is equipped with an Intel Xeon Gold 6130 2.1GHz (16 Cores$\times$2), 64 GB of RAM, running CentOS 7.2. The proposed algorithm was implemented in Python using PyCharm. The source code and data sets are provided at https://goo.gl/twPGZX. The performance of k-CMM algorithm is compared with k-prototypes algorithm [60]. Each algorithm was run ten trials for each data set. The overall performances were then calculated by averaging the results of all trials.

5.5.1 Data sets

The performances of the algorithms have been compared on both synthetic and real data sets. The synthetic data set SD2K is the first 2,000 rows of the SD10K data set generated by the Dataset Generator [47]. It contains 2,000 instances consisting of four categorical attributes and two numeric attributes. The instances are classified into two classes: $c1$ and $c2$ with the distributions are 51% and 49%, respectively. Moreover, eight benchmark mixed data sets with missing values were used for the experiment. The characteristics of these data sets are shown in Tables 5.5 and 5.6. They are real-life data sets having various characteristics that were obtained from the UCI Machine Learning

Table 5.5: Characteristics of the datasets

#	Dataset	#instances	#num attrs	#cat attrs	#classes
1	Credit approval	690	6	9	2
2	Cylinder bands	540	20	19	2
3	Dermatology	366	33	1	6
4	Heart disease	303	5	8	5
5	Hepatitis	155	6	13	2
6	Horse colic	299	11	16	2
7	Postoperative patient	90	1	7	3
8	Sponge	76	3	42	2
9	SD2K	2,000	2	4	2

Table 5.6: Missing values information inside data sets

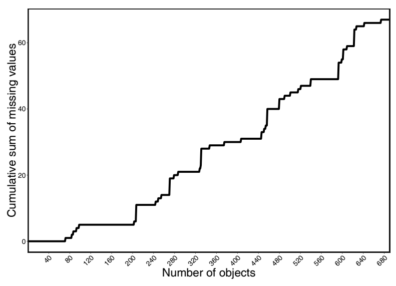
#	Data set	#missing attrs	#missing objs	#missing values
1	Credit approval	7	37	67
2	Cylinder bands	28	263	999
3	Dermatology	1	8	8
4	Heart disease	2	6	6
5	Hepatitis	15	75	167
6	Horse colic	19	293	1,602
7	Postoperative patient	1	3	3
8	Sponge	1	22	22
9	SD2K	6	911	1,136

Repository [34].

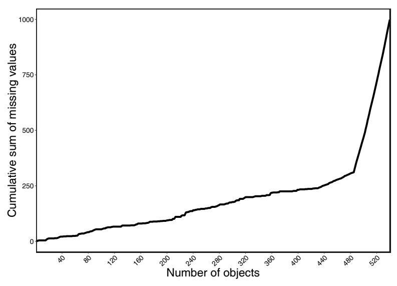
1. Credit approval is a mixed data set that concerns credit card applications where attribute names and values have been changed to meaningless symbols to protect the confidentiality of the data. The instances are classified into two classes: "+" (approval) and "-" (disapproval). Missing values appear in both numeric and categorical attributes.

2. Cylinder bands is a mixed data set for the classification task to classify a process delay known as cylinder banding in rotogravure printing is banded or not. Missing values appear in 19 numeric attributes and nine categorical attributes.

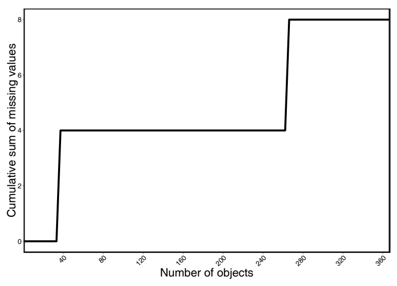
3. Dermatology is a mixed data set that contains instances of dermatology cancer occurrences defined by 34 attributes. The family history attribute is a categorical feature that has the value 1 if the disease has been observed in the family, and 0 otherwise. Other attributes are given a degree in the range of 0 to 3 where



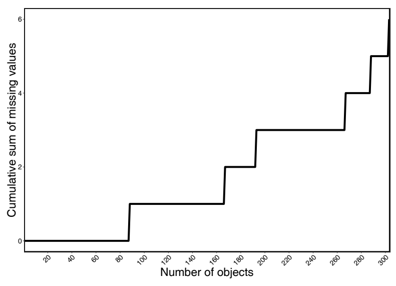
(a) Credit approval



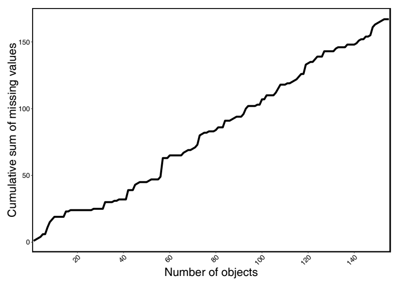
(b) Cylinder bands



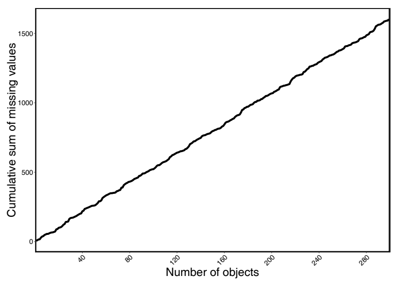
(c) Dermatology



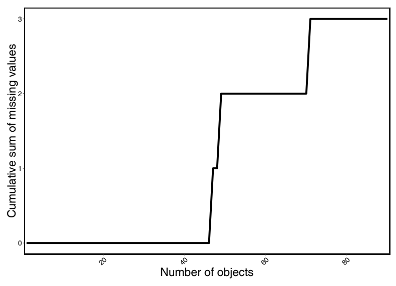
(d) Heart disease



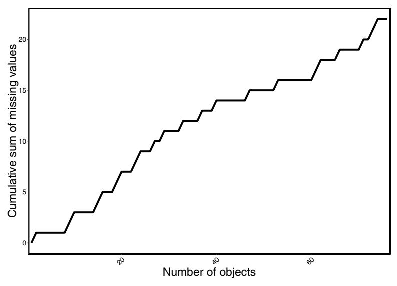
(e) Hepatitis



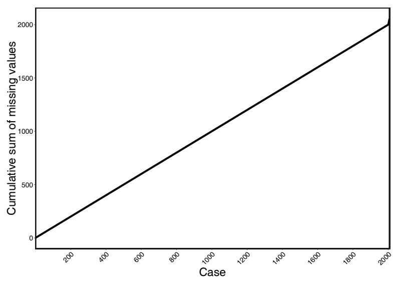
(f) Horse colic



(g) Postoperative patient



(h) Sponge



(i) Synthetic SD2K

Figure 5.6: Cumulative sum of missing values in data sets

0 indicates that the feature was not present, “3” indicates the largest amount possible, and “1”, “2” indicate the relative intermediate values. The instances are classified into six classes representing six kinds of dermatology diseases. Missing values appear only in one numeric attribute.

4. Heart disease is a mixed data set for the classification task to classify the presence of heart disease in the patients. The class attribute is integer valued from 0 (no presence) to 4. It contains 76 attributes, but a subset of 14 of them was mostly used. In this work, we used the Heart disease data set generated at the Cleveland clinic for the experiment. This data set contains 303 instances defined by 13 attributes. The instances are classified into two classes: 0 (normal) and 1 to 4

(people with different degrees of heart disease). Missing values appear in two categorical attributes.

5. Hepatitis is a mixed data set for the classification tasks to classify the status of patients who have hepatitis based on their hepatitis information such as bilirubin, albumin, sgot and prothrombin time. The instances are classified into two classes: died or not. Missing values appear in five numeric attributes and ten categorical attributes.

6. Horse colic is a mixed data set for the classification task to predict if a horse can survive based upon past medical conditions. The training set that contains 299 instances is used for the experiment. The instances are classified into two classes: 1 (yes) and 2 (no) indicating if the lesion is surgical or not. Missing values appear in seven numeric attributes and twelve categorical attributes.

7. Postoperative Patient is a mixed data set for the classification task to determine where patients in a postoperative recovery area should be sent to next. The instances are classified into three classes: "I", "S" and "A" that correspond to patient sent to intensive care unit, patient prepared to go home and patient sent to general hospital floor, respectively. Missing values appear only in the numeric attribute.

8. Sponge is a mixed data set for clustering task to assign Atlantic Mediterranean marine sponges into several groups. The instances are classified into twelve groups induced by the donor's conceptual clustering algorithm. In our work, we used the last attribute as the class attribute and relabeled the majority values [1] as "P" (positive) and "N" (all others as negative). All missing values appear in a categorical attribute.

Note that, the number of attributes of each data set shown in Table 5.5 is the sum of the number of numeric attributes and the number of categorical attributes (excluding the class attribute). By using these data sets, the performance of k-CMM algorithm is evaluated for the different types of data encountered in real-life applications.

[1]https://www.openml.org/d/1001

5.5.2 Experimental results

We used three metrics: Purity, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) to evaluate the quality of clustering results. These metrics use class information in original data sets, which can be called as the gold standard [85], and clustering results generated by an algorithm to measure how well the output clusters match the gold standard classes. Particularly, we omitted the class attributes in data sets during the clustering process and used them later for evaluating the clustering results. The definitions of these metrics are defined in Section 2.3.2.

Table 5.7: Comparison of the Purity results between k-CMM and k-prototypes

#	Data set	k-prototypes	k-CMM
1	Credit approval	0.562	**0.6581**
2	Cylinder bands	0.6625	**0.6678**
3	Dermatology	**0.354**	0.306
4	Heart disease	0.541	**0.548**
5	Hepatitis	**0.794**	**0.794**
6	Horse colic	0.635	**0.685**
7	Postoperative patient	**0.7176**	0.711
8	Sponge	**0.921**	**0.921**
9	SD2K	0.679	**0.6904**

The purity results of k-CMM are compared with k-prototypes. Results (Table 5.7) show that k-CMM outperforms k-prototypes for five data sets: Credit approval, Cylinder bands, Heart disease, Horse colic, and SD2K, while k-prototypes outperforms k-CMM on the Dermatology, Postoperative patient data sets. They have the same results on the Hepatitis and Sponge data sets. In general, k-CMM works well on data sets having more missing values such as the Cylinder bands, Horse colic, and SD2K data sets. Regarding the degree of missing values, it can be observed that the purity results of k-CMM are higher in case of data sets having missing values on both numeric and categorical attributes such as Credit approval, Cylinder bands, Hepatitis, and Horse colic, or in case of data sets having missing values only on categorical attributes such as Heart disease and Sponge data sets. In the case of the Dermatology and Postoperative patient data sets, the purity results of k-prototypes are higher than k-CMM because the missing values appear only on numeric attributes. Particularly, if no missing values appear on categorical attributes, then missing numeric values are imputed using the means of corresponding numeric attributes for each data set.

k-CMM was also compared with k-prototypes in terms of NMI for nine data sets. NMI results of the compared algorithms are shown in Tables 5.8. Particularly, k-CMM outperforms k-prototypes in most cases except for the Cylinder bands and SD2K data sets. Regarding the degree of missing values, it can be observed that the NMI results of k-CMM on data sets in which missing values appear on many categorical attributes such as Credit approval and Hepatitis are much higher than the NMI results of k-prototypes.

Table 5.8: Comparison of the average NMI results between k-CMM and k-prototypes

#	Data set	k-prototypes	k-CMM
1	Credit approval	0.016	**0.1003**
2	Cylinder bands	**0.1576**	0.1376
3	Dermatology	0.0952	**0.101**
4	Heart disease	0.056	**0.0593**
5	Hepatitis	0.0041	**0.0127**
6	Horse colic	0.011	**0.0214**
7	Postoperative patient	0.0335	**0.0369**
8	Sponge	0.015	**0.0156**
9	SD2K	**0.251**	0.2406

Table 5.9: Comparison of the average ARI results between k-CMM and k-prototypes

#	Data set	k-prototypes	k-CMM
1	Credit approval	0.004	**0.1144**
2	Cylinder bands	**0.1056**	0.0707
3	Dermatology	0.0306	**0.066**
4	Heart disease	0.076	**0.0874**
5	Hepatitis	0.0326	**0.046**
6	Horse colic	-0.025	**0.006**
7	Postoperative patient	**0.0197**	-0.014
8	Sponge	0.009	**0.013**
9	SD2K	0.129	**0.134**

Two algorithms were also compared in terms of ARI for the nine data sets. The results are shown in Tables 5.9. Particularly, k-CMM outperforms k-prototypes in most cases except for the Cylinder bands and Postoperative patient data sets. Regarding the degree of missing values, k-CMM works well on data sets having missing values only on categorical attributes or where missing values occur in categorical attributes more than numerical attributes.

5.5.3 Significant Analysis

Table 5.10: Significant analysis of the k-CMM algorithm

#	Data set	Purity		NMI		ARI	
		1st rank	2nd rank	1st rank	2nd rank	1st rank	2nd rank
1	Credit approval	✓		✓		✓	
2	Cylinder bands	✓			✓		✓
3	Dermatology		✓	✓		✓	
4	Heart disease	✓		✓		✓	
5	Hepatitis	✓		✓		✓	
6	Horse colic	✓		✓		✓	
7	Postoperative patient		✓	✓			✓
8	Sponge	✓		✓		✓	
9	SD2K	✓			✓	✓	

In this section, we analysed the significance of the clustering results of k-CMM obtained in the previous section. We considered the proportion of categorical and numerical attributes inside data sets. We denoted data sets having more missing values in categorical attributes than numerical attributes as cat-mixed data sets, while those having more missing values in numerical attributes than categorical attributes as num-mixed data sets. Table 5.10 shows the summary of Purity, NMI and ARI results of k-CMM proposed in this chapter. It can be observed that k-CMM has the highest values for all Purity, NMI and ARI metrics on five over nine data sets: Credit approval, Heart disease, Hepatitis, Horse colic, Sponge; and highest values for two metrics on two over nine data sets: Dermatology and SD2K. On Cylinder bands, k-CMM has the highest value for Purity value. On Postoperative patient, it has the highest value for NMI value. It can be observed that the ARI results of k-CMM are low on data sets in which missing values appear only in numerical attributes or spread on many numeric and categorical data such as Postoperative patient and Cylinder bands data sets. Generally, from the practical results, it can be observed that k-CMM can work well on cat-mixed rather than num-mixed data sets. k-CMM also can work well for both real or synthetic data sets. It means that the imputation method used in k-CMM is suitable for imputing missing categorical attributes and thus enhances the performance of the clustering algorithm.

5.5.4 Execution time

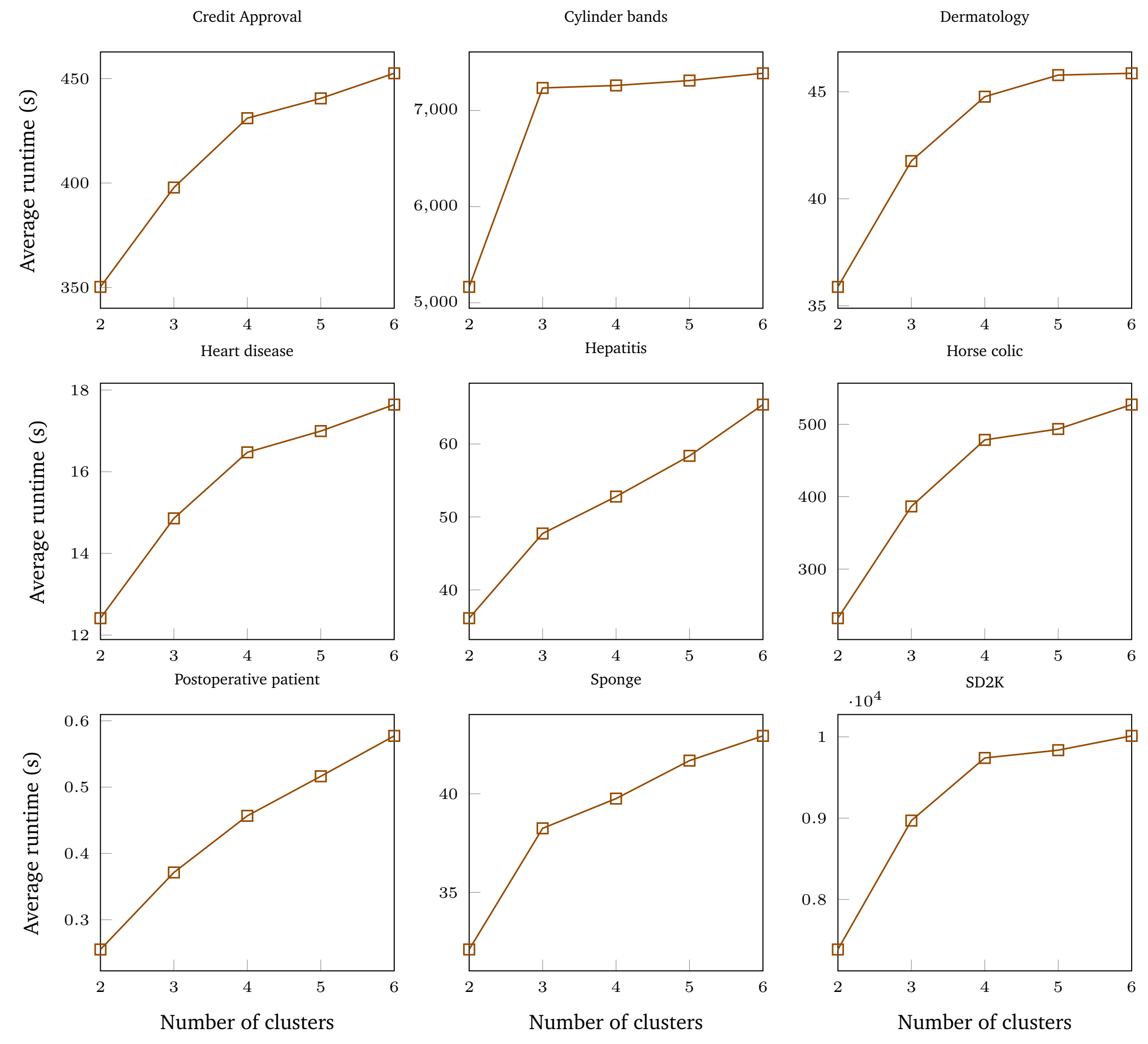


Figure 5.7: Average runtimes for various number of clusters

In this section, the average execution time of k-CMM was evaluated for the various number of clusters for nine mixed data sets. The results are shown in Figure 5.7. In each of these figures, the vertical axis and horizontal axis represent the execution time in seconds and the number of clusters, respectively. For all data sets, when the number of clusters is increased, the time required for clustering them also increases. On Credit approval, k-CMM takes on average 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 SD2K, k-CMM takes on average 7384.7626 (s), 8970.0156 (s), 9740.4216 (s), 9836.2261 (s) and 10011.9882 (s) for 2, 3, 4, 5 and 6 clusters, respectively. Similar results are observed for the other data sets.

5.5.5 Memory usage

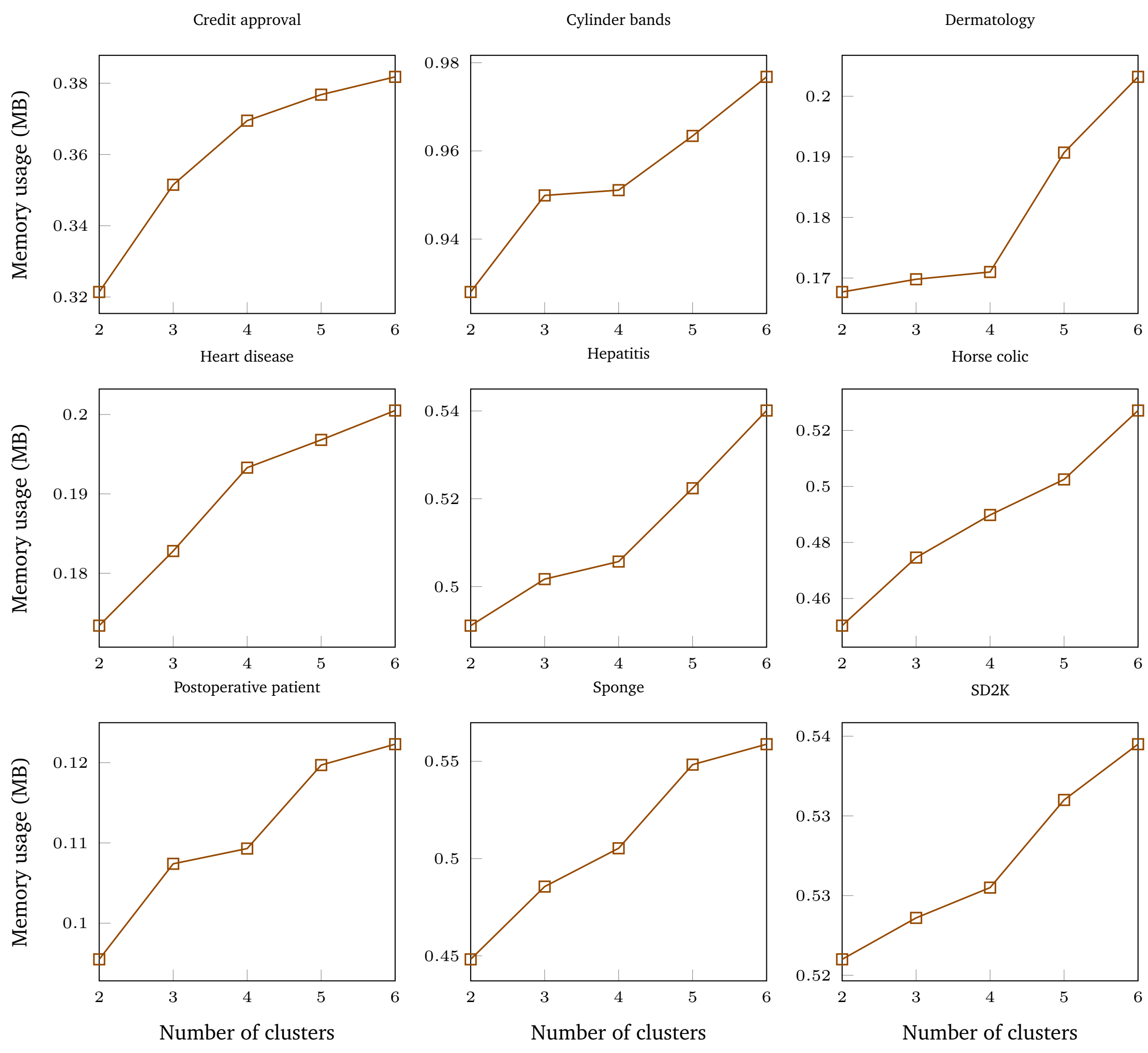


Figure 5.8: Average memory usage for various number of clusters

The average memory usage of the proposed k-CMM algorithm was also evaluated for the various number of clusters for nine mixed data sets. The results are shown in Figure 5.8 in terms of memory usage in megabytes (vertical axes) for the various number of clusters (horizontal axes). In general, the memory usage increases when the number of clusters is increased in most cases and k-CMM also has linear scalability for all data sets. On Credit approval, k-CMM takes on average 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 Cylinder bands, k-CMM takes on average 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 are observed for other data sets.

5.5.6 Scalability

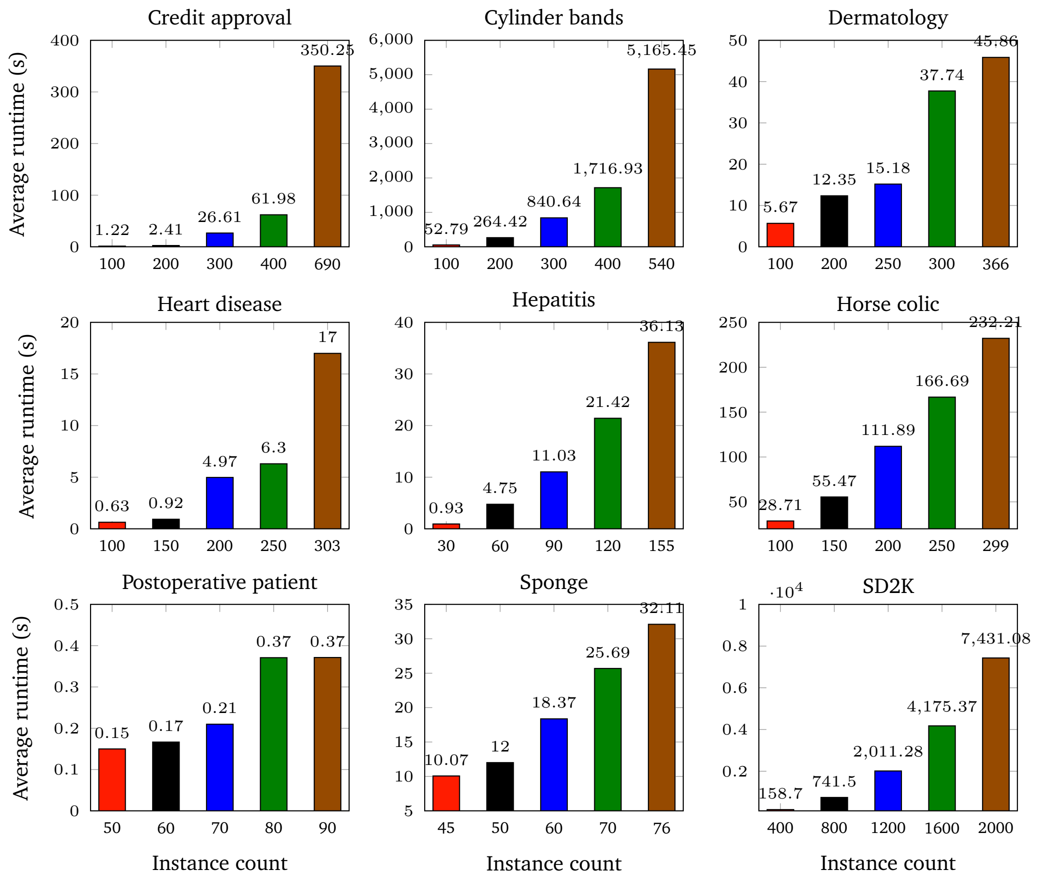


Figure 5.9: Average runtimes when varying the database size

Another experiment was performed to assess the scalability of k-CMM algorithms. The execution times of k-CMM were measured while varying the number of instances for the nine mixed data sets. The results are shown in Figure 5.9. In this figure, vertical axes indicate the average execution times in seconds, while horizontal axes represent the number of instances used. In general, k-CMM has linear scalability for all data sets. On Credit approval, k-CMM takes on average 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 are observed for other data sets.

Moreover, the memory usage of k-CMM was also measured on the mixed data set while varying the number of instances. The results are shown in Figure 5.10. In this

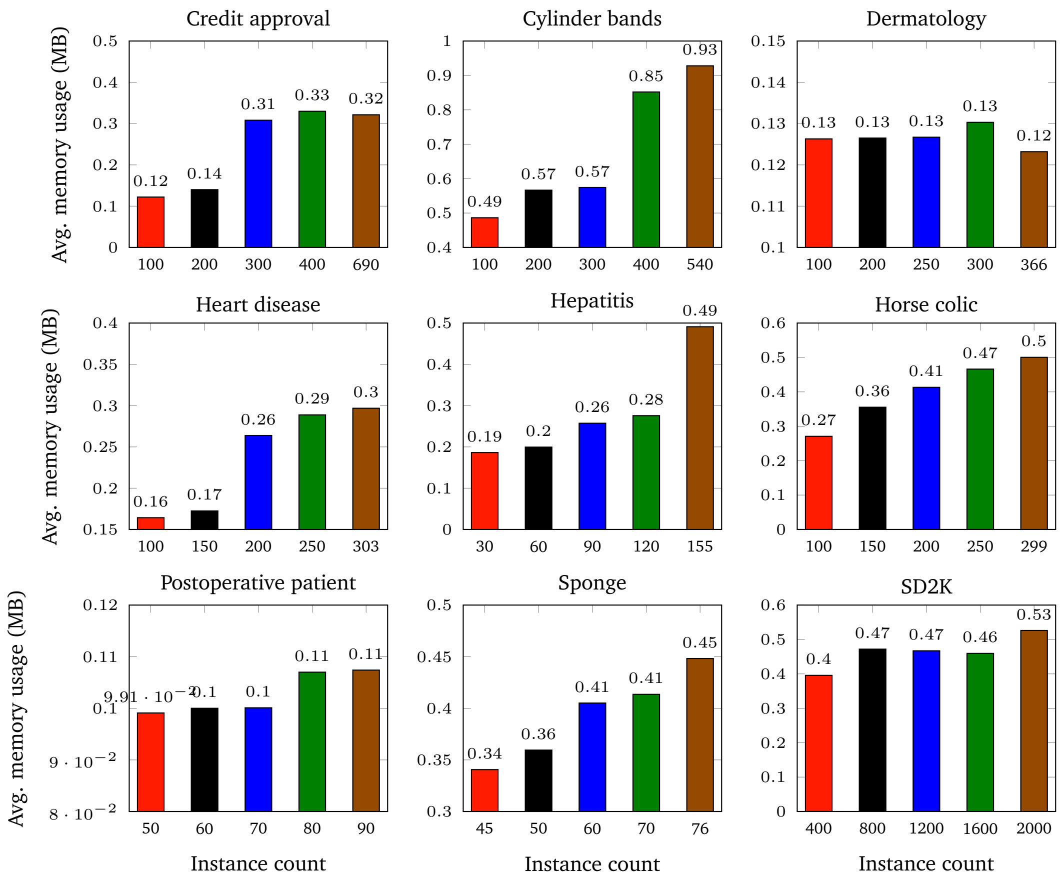


Figure 5.10: Average memory usage when varying the database size

figure, vertical axes indicate the average memory usage in megabytes, while horizontal axes represent the number of instances used. In general, k-CMM also has linear scalability for all data sets. On Credit approval, k-CMM takes on average 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 are observed for other data sets.

5.6 Conclusion

Missing values appear commonly in data sets and can significantly affect the efficacy of the research study. It can be seen that many mixed numeric and categorical data sets in real-life applications contain missing values. It is also worth to remark that clustering is one of the most popular tasks in data mining, and clustering mixed data sets into meaningful groups is practically useful because frequently encountered objects in real-life data sets are mixed objects. This chapter has addressed the above two issues by proposing an algorithm named k-CMM for clustering mixed numeric and categorical data sets with missing values. It integrates the imputation and clustering steps into a common framework to improve clustering results. In the imputation step, it first uses the decision-tree based method 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 for missing values in categorical attributes. The missing values in numeric attributes are imputed using the mean of corresponding attributes from the correlated set. In the clustering step, k-CMM uses the kernel density estimation approach and the mean to define cluster centers for numeric and categorical attributes, respectively. In addition, to quantify the proximity between data objects, it uses the squared Euclidean and the information-theoretic based dissimilarity measure for numeric and categorical attributes, respectively. Experimental results have shown that k-CMM is more efficient than k-prototypes in terms of clustering quality in most cases. It means that the imputation method used in the paper can enhance clustering results. Generally, k-CMM has a comparative performance in terms of Purity, NMI and ARI. Moreover, we also evaluated the runtime, memory consumption, and scalability of k-CMM. Results show that k-CMM is scalable with respect to the number of instances.

Chapter 6

An Improved Cluster Center Initialization for Categorical Data Clustering

6.1 Introduction

Clustering has a long history in data mining and knowledge discovery. It is a common data mining tool employed in many fields ranging from computer science, engineering, medical science to social science and economics. Generally, clustering aims at partitioning a set of data objects into several groups such that objects inside a group are very similar, but are very dissimilar to objects in other groups [53]. In the literature, distance-based clustering is preferable thanks to its ease of understanding and implementation. The algorithms of this branch can be generally categorized into two groups: hierarchical and partitional (partitioning) clustering. Partitional clustering is different from hierarchical clustering in pre-determining the number of clusters to be grouped. Thus, its performances are depended on and influenced by the choice of initial cluster centers and the number of clusters [97]. This chapter focuses on the former factor due to the following reasons. First, initialization methods have a strong impact on clustering results. Most algorithms use random initialization because of simplicity. Nevertheless, a limitation of this method is in producing different results for different runs of the algorithms. Among these results, low clustering quality may occur. Second, it is difficult to comprehend non-repeatable results because inappropriate choices may lead to highly undesirable cluster structures. The randomly initial choices are proper in the case that selected initial clusters are similar to the real clusters existing in data sets [18, 68, 71]. Third, it is recommended that clustering algorithms should be run multiple times from different random configurations to find the lowest minimum of within-cluster variation. However, such approaches would have been time-consuming to meet the convergence criteria, and final results may still be bad. These reasons motivate the design of a non-random initialization framework in this work for categorical data clustering.

Frequent Itemset Mining (FIM) aims to discover association rules between items in transaction data sets [5]. It has become a popular research topic in data mining. Although it was initially designed to find useful patterns and analyze sales data, it can be more generally considered as the task of finding groups of attribute values frequently co-occur in data sets [40]. However, finding a large number of itemsets that may contain much redundancy in some cases is an inherent limitation of FIM algorithms. Thus, many extensions to the task of FIM have been proposed to address this limitation such as discovering closed itemsets, maximal itemsets, and generator itemsets. Maximal frequent itemsets (MFIs) are the set of frequent itemsets that do not have supersets that are frequent [40, 50]. Obviously, MFIs are the largest FIs. Mining MFIs is often faster than mining the full set of FIs. This chapter applies an MFI-based approach to form initial clusters. The reasons for using this technique are based on the following observations:

Observation 1 A categorical data set can be considered as a data set of customer transactions in which each transaction is described by items (categorical values) bought by customers. In other words, the pattern mining techniques can be adapted to categorical data clustering problem and vice versa.

Observation 2 A frequent pattern appears in a transaction data set such that its support is larger than or equal to a given a minimum support threshold ($minsup$). Particularly, if $minsup$ is set to Δ, then at least Δ transactions contain that frequent pattern. It means that categorical objects corresponding to these transactions contain the same sets of categorical values (categories) and thus make them more similar (correlated) than other data objects. These instances can be merged to form an initial cluster for the clustering algorithm.

Observation 3 Two categorical objects are more similar if they have more categories in common. The MFI can be used in this case since it represents the largest set of categories in each object. In addition, mining MFIs is often faster and more concise than mining the full set of frequent itemsets. Thus, MFI mining (MFIM) does not require much time consuming when applied to the clustering problem.

The following points are the main contributions of this chapter:

- To our best knowledge, this is the first study that considers the combination of frequent pattern mining and partitional clustering. The target is to design a non-randomized method for clustering algorithm by exploiting the pattern mining approach instead of picking objects randomly to form initial clusters.

- We propose an algorithm named Pattern Based Clustering for categorical data (k-PbC) that takes the advantages of non-randomized initialization to improve clustering results. k-PbC uses an MFIM algorithm to find a list of MFIs for the initial clusters. It is worth mentioning that there is always only a correct answer to an MFIM task for a given transaction data set and a threshold value. Hence, any MFIM algorithm can be adapted to the k-PbC algorithm. In this chapter, the FPMAX algorithm [50] is extended to find patterns that contain the largest sets of categories frequently co-occurring in a categorical data set. Objects containing the same patterns are merged to form an initial cluster. The way of forming initial clusters and strategy to avoid the overlaps between them are given in detail in section 6.3. For clustering, k-PbC uses a kernel density estimation method for the formation of cluster centers and an information-theoretic based distance method to estimate the dissimilarities between cluster centers and data objects.

- An extensive experiment was conducted on real data sets to evaluate the performance of k-PbC for both internal and external validation metrics. Experimental results show that the proposed initialization step can enhance clustering results and the proposed algorithm outperforms state-of-the-art categorical data clustering algorithms.

The remainder of this chapter is structured into following sections. Section 6.2 reviews related work. Section 6.3 gives preliminary definitions. Section 6.4 proposes the k-PbC algorithm. Section 6.5 describes experimental results. Finally, Section 6.6 draws a conclusion.

6.2 Related work

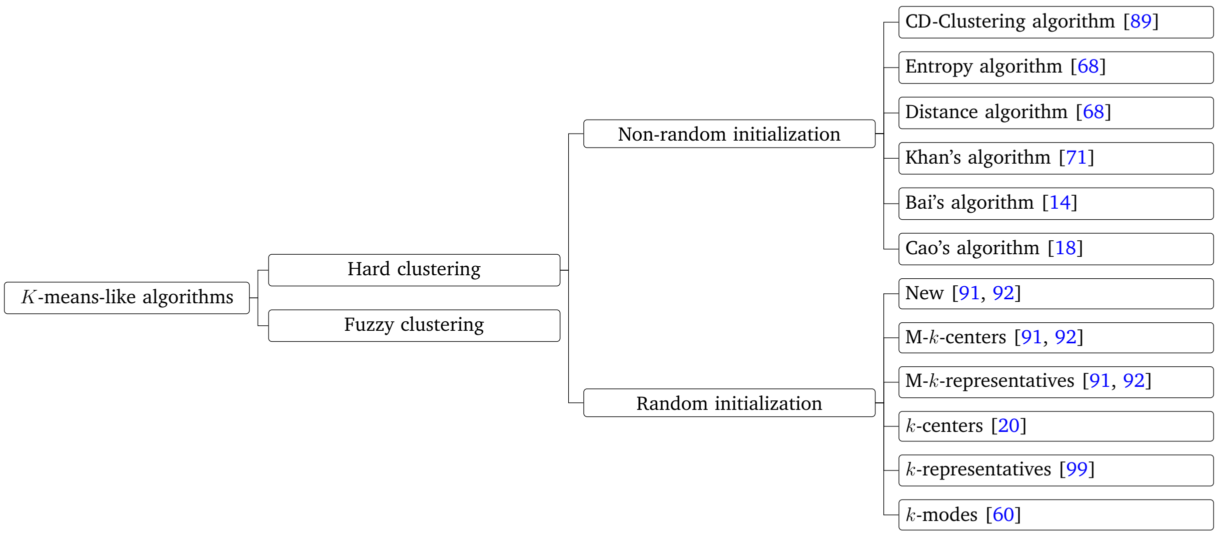


Figure 6.1: A taxonomy of k-means-like algorithms

The details of related clustering methods for categorical data clustering are provided in section 2.4. In the numerical setting, a variety of initialization methods were proposed to improve the performance of k-means [19]. K-means++ [11] addresses the problem of random initialization of k-means. It guarantees to yield an $\mathcal{O}(logk)$ approximation to the optimal k-means solution. Its workflow can be expressed as follows. It first chooses an initial center $\mathcal{C}_1$ uniformly at random from the data set $\mathcal{D}_{num}$. Then, each remaining cluster center $\mathcal{C}_l$ $(2 \leq l \leq k)$ is selected from the remaining objects $x' \in \mathcal{D}_{num}$ with a probability $\frac{d(x')^2}{\sum_{i=1}^{n} d(x_i)^2}$, where $d(x)$ be the shortest distance from x to the previously selected centers. In other words, k-means++ randomly chooses an object as a new center with probability proportional to its squared distance $d(x)$. It repeats the previous step until k centers are chosen and then performs the clustering using the standard k-means. It was proven to improve both the accuracy and runtime of k-means. However, this method was designed for numerical data and may not work well for the categorical data setting.

Several initialization methods were proposed to improve clustering results. In k-modes [60], Huang introduced two initialization methods. The first one selects the first k instances from data set for forming k initial *modes*. However, it only works in case the selected instances are picked from k disjoint clusters. The second one first calculates the frequencies of categories in attributes and uses an array to keep this information in descending order of frequency. The most frequent categories in attributes are selected to form k *modes*. To avoid empty clusters, for each initial mode, it selects the data object which is most similar to that mode and replaces it with the object as an initial mode. The second initialization method can yield better clustering results when compared to the first method since the former diversifies initial *modes*. However, it lacks a uniform criterion in the selection of initial clusters [18].

In 2009, Cao et al. introduced a method that determines the average density of objects and the distance between objects to initialize cluster centers [18]. Specifically, it uses the frequency of categories to define the average density of an object. The first cluster center is formed by selecting the object which has the maximum density. It then extends the MaxMin algorithm in the combination of objects' density and the distance between objects for remaining clusters. The clustering algorithms are then used the initialization approach in combination with k-modes and fuzzy k-modes. In 2012, Bai et al. proposed an extension of Cao's initialization method [14]. The first cluster center is selected by using distances between objects and the center of the whole data set. Thus, it can avoid choosing the boundary objects among clusters. The selected object is called a *cluster exemplar*, which is integrated with the neighbor objects around it to generate candidates. Finally, the algorithm uses several criteria to select initial cluster centers from the list of candidates.

In 2013, Khan et al. proposed an algorithm [71] that uses categories present in different attributes to perform multiple clustering on a given data set. It first selects the most relevant attributes, called *prominent attributes*, and uses an unsupervised learning approach to compute the rank of *significant attributes*. It then creates initial cluster centers called *deterministic modes* by running multiple clustering on selected attributes. In 2016, Jiang et al. improved k-modes by two initialization methods [68]. Both methods aim to ensure the initial cluster centers are not outliers by determining the outlierness degrees of data objects. They use the distance-based and entropy-based outlier detection techniques to achieve this purpose.

Recently, Nguyen improved k-modes by using the community detection technique for the initialization step [89]. Specifically, the algorithm named CD-Clustering first employs the Louvain community detection technique to create an unweighted graph from a given data set. It then uses the Hamming distance and a threshold to discover highly homogeneous groups (cohesive groups of nodes) from data. The threshold R is determined by the distance distribution and the number of clusters k. The top k communities in descending order of size are used for k initial modes. CD-Clustering was proven to outperform the random k-modes [60], Cao's method [18] and Khan's method [71]. However, this algorithm faces difficulties in finding top k communities when the *intra/inter* cluster distance gap and the number of clusters are both small [89].

6.3 Preliminaries

Table 6.1: List of notations

k	$\triangleq$	a pre-defined number of clusters
$\mathcal{D}_{cat}$	$\triangleq$	a categorical data set
$\mathcal{S}$	$\triangleq$	a transaction data set
x_i	$\triangleq$	a categorical object in $\mathcal{D}_{cat}$
$\mathcal{T}_q$	$\triangleq$	a transaction in $\mathcal{S}$
$\mathcal{A}_j$	$\triangleq$	the j^{th} attribute of $\mathcal{D}_{cat}$
$\mathcal{C}_l$	$\triangleq$	the l^{th} cluster
n_l	$\triangleq$	the number of objects in $\mathcal{C}_l$
$\mathcal{Z}_l$	$\triangleq$	the center of cluster $\mathcal{C}_l$
$\mathcal{X}_j^l$	$\triangleq$	the random variable associated with observations in $\mathcal{C}_l$
$\mathcal{O}_j$	$\triangleq$	the categories set appeared in $\mathcal{D}_{cat}$ at the j^{th} attribute
$\mathcal{O}_j^l$	$\triangleq$	the categories set appeared in cluster $\mathcal{C}_l$ at the j^{th} attribute
o_{ij}	$\triangleq$	a category at the i^{th} element and j^{th} attribute of $\mathcal{D}_{cat}$
o_{ij}^l	$\triangleq$	a category at the i^{th} element and j^{th} attribute of $\mathcal{C}_l$
z_j^l	$\triangleq$	a category appeared in the center $\mathcal{Z}_l$ at the j^{th} attribute

Let $\mathcal{A}$ $=\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\}$ be a set of m distinct categorical attributes (features), each of which is associated with a finite set $\mathcal{O}_j$ $(1 \leq j \leq m)$ as its domain such that $DOM(\mathcal{A}_j) = \mathcal{O}_j$ (> 1 discrete values). A categorical data set $\mathcal{D}_{cat} = \{x_1, x_2, \dots, x_n\}$ is a set of n categorical objects (instances), where each object $x_i \in \mathcal{D}_{cat}$ $(1 \leq i \leq n)$ is a tuple $x_i = (x_{i1}, x_{i2}, \dots, x_{im}) \in \mathcal{O}_1 \times \mathcal{O}_2 \times \cdots \times \mathcal{O}_m$. In other words, $\mathcal{D}_{cat}$ can be represented by a data matrix with n rows and m column $(n \gg m)$, and each element at position i $(1 \leq i \leq n)$ and j $(1 \leq j \leq m)$ of the matrix contains the value of the object x_i at the attribute j^{th} such that $x_{ij} \in \mathcal{O}_j$.

Table 6.2: A categorical data set

Object	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
x_1	yellow	small	stretch	adult
x_2	yellow	small	stretch	child
x_3	purple	small	dip	adult
x_4	purple	small	dip	child
x_5	yellow	small	stretch	adult
x_6	yellow	small	stretch	child
x_7	purple	small	dip	adult
x_8	yellow	small	dip	child
x_9	yellow	large	stretch	adult
x_{10}	yellow	large	stretch	child

Table 6.3: The equivalent transaction data set

TID	Transaction
$\mathcal{T}_1$	{yellow,small,stretch,adult}
$\mathcal{T}_2$	{yellow,small,stretch,child}
$\mathcal{T}_3$	{purple,small,dip,adult}
$\mathcal{T}_4$	{purple,small,dip,child}
$\mathcal{T}_5$	{yellow,small,stretch,adult}
$\mathcal{T}_6$	{yellow,small,stretch,child}
$\mathcal{T}_7$	{purple,small,dip,adult}
$\mathcal{T}_8$	{yellow,small,dip,child}
$\mathcal{T}_9$	{yellow,large,stretch,adult}
$\mathcal{T}_{10}$	{yellow,large,stretch,child}

A categorical data set can be considered as a transaction data set where each instance is a transaction and each category is an item bought by a customer. The problem of FIM is defined as in [40]. Given a set of items $I = \{i_1, i_2, \dots, i_\omega\}$. X is called an itemset if it contains a subset of items in I. Moreover, if X contains k items, i.e. $|X| = k$, then it is called a k-itemset. A transaction $\mathcal{T}_q$ is also a set of items in I such that $\mathcal{T}_q \subseteq I$, where q $(1 \leq q \leq n)$ is the unique identifier of $\mathcal{T}$ as its transaction identifier (TID). A transaction data set $S = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n\}$ consists of n transactions. Table 6.2 presents a data set having ten categorical objects described by four attributes. Table 6.3 shows the equivalent transaction data set converted from Table 6.2. It contains ten transactions, each of which is a set of items which correspond to categories in the categorical data set.

Definition 21 (Frequent itemset) Given an itemset X and a transaction data set $\mathcal{S}$.

The support of X is the number of transactions that contain it, denoted by:

$$sup(X) = |\{\mathcal{T}|X \subseteq \mathcal{T} \wedge \mathcal{T} \in \mathcal{S}\}| \tag{6.1}$$

If the support of X is no less than a user-specified minimum support threshold $minsup$, then X is called a frequent itemset (FI). The list of FIs is defined as:

$$FIs = \{X|sup(X) \geq minsup\} \tag{6.2}$$

Example 13 Assum that $minsup = 4$, the set of 14 FIs discovered from Table 6.3 is $\{$ ⟨dip⟩ $: 4$, ⟨dip, small⟩ $: 4$, ⟨child⟩ $: 5$, ⟨child, yellow⟩ $: 4$, ⟨child, small⟩ $: 4$, ⟨adult⟩ $: 5$, ⟨adult, small⟩ $: 4$, ⟨stretch⟩ $: 6$, ⟨stretch, small⟩ $: 4$, ⟨stretch, yellow, small⟩ $: 4$, ⟨stretch, yellow⟩ $: 6$, ⟨yellow⟩ $: 7$, ⟨yellow,small⟩ $: 5$, ⟨small⟩ $: 8\}$, where the number besides each MFI is its support.

Definition 22 (Maximal frequent itemset) A maximal frequent itemset is a frequent itemset that has no frequent supersets. The list of MFIs is defined as:

$$MFIs = \{X|X \in FIs \wedge \nexists Y \in FIs \text{ such that } X \subset Y\} \tag{6.3}$$

MFIs are the largest FIs. The set of MFIs is a subset of the set of FI ($MFIs \subseteq FIs$). The goal of MFIM is to find all MFIs in a given transaction data set.

Example 14 Table 6.4 shows all MFIs discovered from Table 6.3 for $minsup$ from one to ten. The number besides each MFI is its support.

The key points when designing a partitional clustering algorithm are using an appropriate method for cluster centers and choosing a suitable distance measure for a specific data type. For the first point, recent methods for categorical data consider the cluster centers as the expectation of a random variable associated with the data, in the assumption that this variable follows a Gaussian distribution from the statistical point of view. The goal is to find a method that can guarantee the consistency in the statistical interpretation of the cluster centers for categorical data as the *means* for numerical data.

Table 6.4: The sets of MFIs for various minimum support threshold values

minsup	MFIs	minsup	MFIs
1	$\langle$child,large,stretch,yellow$\rangle$: 1 $\langle$adult,large,stretch,yellow$\rangle$: 1 $\langle$child,dip,purple,small$\rangle$: 1 $\langle$child,dip,small,yellow$\rangle$: 1 $\langle$adult,dip,purple,small$\rangle$: 2 $\langle$child,small,stretch,yellow$\rangle$: 2 $\langle$adult,small,stretch,yellow$\rangle$: 2	6	$\langle$small$\rangle$: 8 $\langle$stretch,yellow$\rangle$: 6
2	$\langle$large,stretch,yellow$\rangle$: 2 $\langle$child,dip,small$\rangle$: 2 $\langle$dip,purple,small$\rangle$: 2 $\langle$child,small,stretch,yellow$\rangle$: 2 $\langle$adult,small,stretch,yellow$\rangle$: 2	7	$\langle$yellow$\rangle$: 7 $\langle$small$\rangle$: 8
3	$\langle$dip,purple,small$\rangle$: 3 $\langle$child,small,yellow$\rangle$: 3 $\langle$child,stretch,yellow$\rangle$: 3 $\langle$adult,stretch,yellow$\rangle$: 3 $\langle$adult,small$\rangle$: 4 $\langle$small,stretch,yellow$\rangle$: 4	8	$\langle$small$\rangle$: 8
4	$\langle$dip,small$\rangle$: 4 $\langle$child,yellow$\rangle$: 4 $\langle$chid,small$\rangle$: 4 $\langle$adult,small$\rangle$: 4 $\langle$small,stretch,yellow$\rangle$: 4	9	$\emptyset$
5	$\langle$child$\rangle$: 5 $\langle$adult$\rangle$: 5 $\langle$small,yellow$\rangle$: 5 $\langle$stretch,yellow$\rangle$: 6	10	$\emptyset$

This chapter also uses a kernel-based method to define cluster centers, called probabilistic cluster centers. This method estimates the probability density function of each attribute in cluster centers, which is derived from Aitchison & Aitken's kernel function [8]. The detail of the probabilistic center used the kernel density estimation approach is defined in Section 2.2. For the sake of brevity, we restated these definitions as the following.

Definition 23 (Cluster Center) Let $\mathcal{O}_j^l$ denote the categories set appeared in $\mathcal{C}_l = \{x_1, x_2, \ldots, x_{n_l}\}$ at the j^{th} attribute, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ $(1 \le i \le n_l)$. The center of $\mathcal{C}_l$ is denoted and defined as:

$$\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\} \tag{6.4}$$

where the value at j^{th} element of $\mathcal{Z}_l$ is measured by a kernel-based method using Eq. (2.4) as a probability distribution on $\mathcal{O}_j^l$:

$$z_j^l = [\mathcal{P}_j^l(o_{1j}^l), \mathcal{P}_j^l(o_{2j}^l), \ldots, \mathcal{P}_j^l(o_{|\mathcal{O}_j^l|j}^l)] \tag{6.5}$$

and the value of each categorical value o_{ij}^l $(1 \leq i \leq |\mathcal{O}_j^l|)$ is formulated by the combination of Eqs. (2.2), (2.3), (2.4) as the following:

$$\mathcal{P}_j^l(o_{ij}^l) = \begin{cases} \lambda_l \frac{1}{|\mathcal{O}_j^l|} + (1-\lambda_l)\mathfrak{f}_l(o_{ij}^l) & \text{if } o_{ij}^l \in \mathcal{O}_j^l \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

Designing dissimilarity measures for categorical data has been considered in many previous works [16, 33]. In 1998, Dekang Lin proposed a similarity measure inspired by the idea of information theory in which less frequent words have a higher information gain [81]. This measure describes the relationship between the common and different information components. Particularly, the information-theoretic based similarity of x_i and $x_{i'}$ is estimated by the ratio between the amount of information needed to state the commonality of x_i and $x_{i'}$ and the information needed to fully describe what x_i and $x_{i'}$ are. Several studies proposed information-theoretic based dissimilarity (ITBD) measures based on [81] for categorical data [16, 91, 92]. This measure is also used in this chapter for distance calculation.

Definition 24 (Categories dissimilarity) Let there be categories o_{ij} and $o_{i'j}$ appeared in x_i and $x_{i'}$ at the j^{th} attribute, respectively. The information-theoretic based similarity of the two categories is measured as:

$$\text{sim}_j(o_{ij}, o_{i'j}) = \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{6.7}$$

where $\mathfrak{f}(o_{ij}, o_{i'j}) = \frac{\#(o_{ij}, o_{i'j})}{|D|}$, $\#(o_{ij}, o_{i'j})$ indicates the number of objects in $\mathcal{D}_{cat}$ at the

j^{th} attribute having the value in $\{o_{ij}, o_{i'j}\}$.

The dissimilarity of o_{ij} and $o_{i'j}$ is formulated by:

$$\text{dsim}_j(o_{ij}, o_{i'j}) = 1 - \text{sim}_j(o_{ij}, o_{i'j}) = 1 - \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{6.8}$$

Definition 25 (Dissimilarity between objects and cluster centers) Let there be an object $x_i = (x_{i1}, x_{i2}, \dots, x_{im})$ and a cluster $\mathcal{C}_l$ with its center is $\mathcal{Z}_l = \{z_1^l, z_2^l, \dots, z_m^l\}$. Let $\mathcal{O}_j^l$ be the categories set appeared in z_j^l. The dissimilarity at the j^{th} attribute of x_i and $\mathcal{Z}_l$ is calculated by accumulating probability values of categories in $\mathcal{O}_j^l$ and dissimilarities between category x_{ij} of x_i and categories at the j^{th} component $\ddagger_j^l$ of $\mathcal{Z}_l$. Mathematically, the definition can be formulated by:

$$\text{dis}_j(x_i, \mathcal{Z}_l) = \sum_{o_{ij}^l \in \mathcal{O}_j^l} \mathcal{P}_j^l(o_{ij}^l)\text{dsim}_j(x_{ij}, o_{ij}^l) \tag{6.9}$$

The dissimilarity between x_i and cluster center $\mathcal{Z}_l$ can be formulated by:

$$\text{dis}(x_i, \mathcal{Z}_l) = \sum_{j=1}^{m} \text{dis}_j(x_i, \mathcal{Z}_l) \tag{6.10}$$

From Eq. 6.10, if x_i and $\mathcal{Z}_l$ contain identical categories at each attribute or $\mathcal{Z}_l$ contains only x_i, then the dissimilarity between them is zero. If categories at each attribute of x_i and $\mathcal{Z}_l$ are totally different, then the dissimilarity between them equals to the number of features.

The algorithm for categorical data clustering can be considered as the following optimization function:

$$\mathcal{F}(\mathcal{U}, \mathcal{Z}) = \sum_{l=1}^{k}\sum_{i=1}^{n} u_{i,l} \times \text{dis}(x_i, \mathcal{Z}_l) \tag{6.11}$$

subject to

$$\begin{cases} u_{i,l} \in \{0, 1\} & 1 \le l \le k,\ 1 \le i \le n \\ \sum_{l=1}^{k} u_{i,l} = 1 & 1 \le i \le n \end{cases} \tag{6.12}$$

where $\mathcal{Z}$ is the set of cluster centers formed by Eq. 6.4, $\mathcal{U} = [u_{i,l}]_{n\times k}$ is the partition matrix where $u_{i,l} = 1$ if x_i belongs to $\mathcal{C}_l$ and 0 otherwise.

6.4 The *k*-PbC algorithm

Figure 6.2 shows the general process of k-PbC algorithm. According to the model, k-PbC consists of two phases. For the initialization phase, it uses an MFIM algorithm to mine MFIs in the converted transaction data set. The top k MFIs with high support values and long sizes are selected from the obtained set. The reason for this choice is to find high correlated objects to form initial groups of objects. For each MFI in the top-k set, k-PbC tracks the set of transactions where that itemset occurs. Next, an overlap filter and an initial assignment strategy are used to remove the overlapping between groups and assign remaining objects in the data set into k initial groups, respectively. These steps guarantee that the initial groups will become clusters as the definition 1. For the clustering phase, the assignment and update steps are performed in turn to assign objects in $\mathcal{D}_{cat}$ into k clusters until convergence.

```
Algorithm 7: THE k-PbC ALGORITHM
    input  : $k$: the number of clusters, $\mathcal{D}_{cat}$: a categorical data set
    output: $k$ clusters
1 Perform the Initiate_Centers($\mathcal{D}_{cat}, k$) procedure (Algorithm 8) to form $k$ initial
    cluster centers such that $\mathcal{Z}^{(0)} = \{\mathcal{Z}_1^{(0)}, \ldots, \mathcal{Z}_k^{(0)}\}$
2 $\mathrm{i} = 0, \mathcal{U} \leftarrow \emptyset$
3 while clusters are not stable do
4     Fix $\mathcal{Z}^{(\mathrm{i})}$ and find $\mathcal{U}^{(\mathrm{i})}$ to minimize $\mathcal{F}(\mathcal{U}^{(\mathrm{i})}, \mathcal{Z}^{(\mathrm{i})})$
5     Fix $\mathcal{U}^{(\mathrm{i})}$ and update $\mathcal{Z}^{(\mathrm{i})}$ to minimize $\mathcal{F}(\mathcal{U}^{(\mathrm{i})}, \mathcal{Z}^{(\mathrm{i})})$
6     $\mathrm{i} = \mathrm{i} + 1$
7 end
8 return $k$ clusters;
```

The pseudo-code of k-PbC algorithm is presented in Algorithm 7. It takes a categorical data set $\mathcal{D}_{cat}$ and a pre-defined parameter k as the input. k-PbC first performs the Initiate_Centers procedure to form k initial cluster centers (line 1). It is worth noting again that most of the previous works simply used the random method for their initialization. However, such a way yields different results for different runs on the same data set and poor results may occur in some cases. Thus, Initiate_Centers is an important step for the whole model since it takes advantage of non-random initialization to improve clustering results. After forming k initial clusters, k-PbC defines centers for these

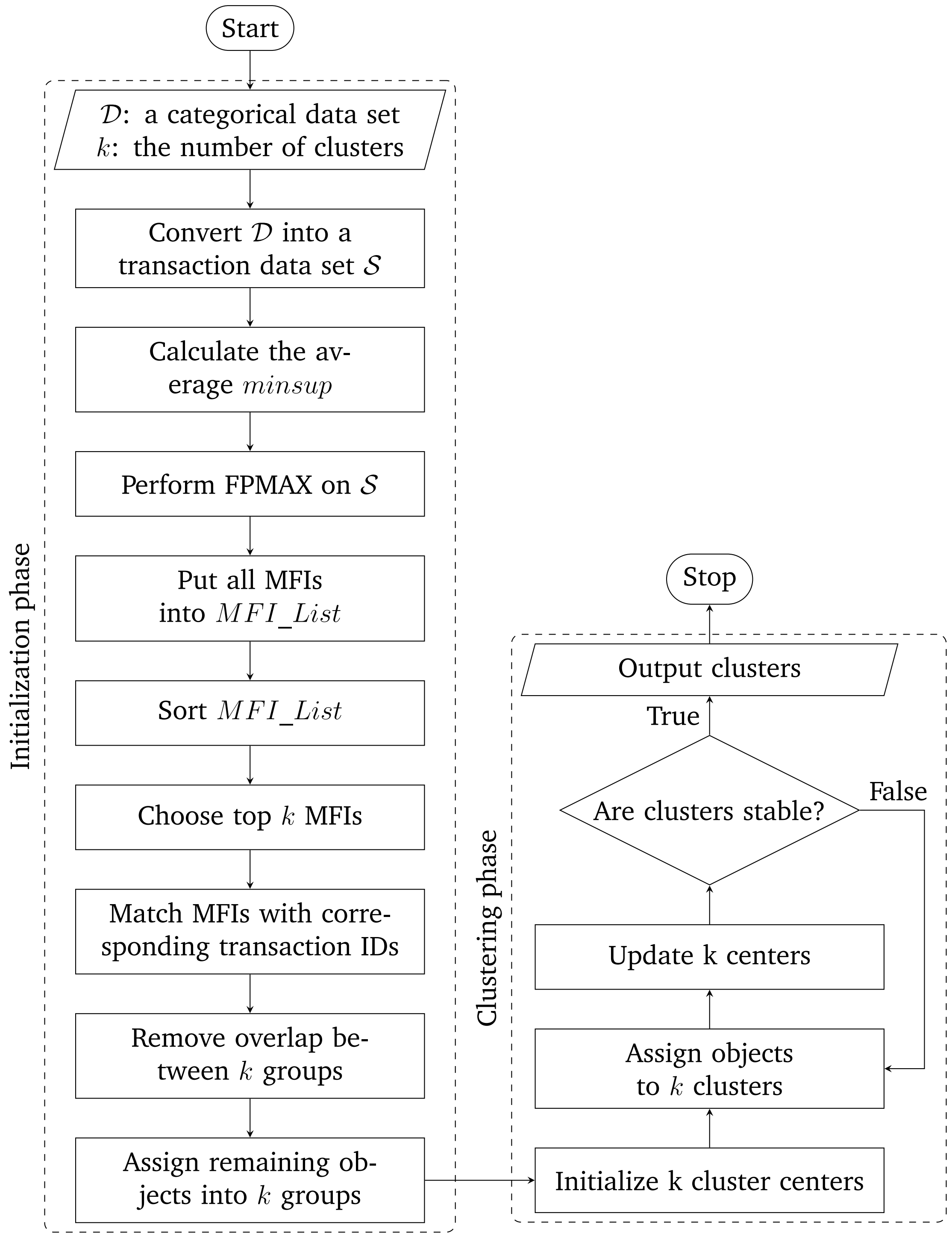

Start
$\mathcal{D}$: a categorical data set
k: the number of clusters
Convert $\mathcal{D}$ into a transaction data set $\mathcal{S}$
Calculate the average $minsup$
Perform FPMAX on $\mathcal{S}$
Put all MFIs into MFI_List
Sort MFI_List
Choose top k MFIs
Match MFIs with corresponding transaction IDs
Remove overlap between k groups
Assign remaining objects into k groups
Initialization phase
Initialize k cluster centers
Assign objects to k clusters
Update k centers
Are clusters stable?
False
True
Output clusters
Stop
Clustering phase

Figure 6.2: The flowchart of k-PbC algorithm

clusters by using Eq. (6.4). For each data object in $\mathcal{D}_{cat}$, k-PbC calculates the distance between that object and k clusters by using the Eq. (6.10) and then assigns it to the nearest cluster (line 4). In the next step, cluster centers are updated by using Eq. (6.4) (lines 5). It repeatedly operates the assignment and update steps in the same manner until all clusters are stable (lines 3-6). Finally, it outputs k clusters (line 7).

Algorithm 8: THE INITIATE_CENTERS PROCEDURE

```
input  : $\mathcal{D}_{cat}$: a categorical data set, $k$: the number of clusters, $minsup$: an optional minimum
         support threshold
output: $k$ initial clusters
1  $Items \leftarrow \emptyset$, $map\_Items\_TIDs \leftarrow \emptyset$
2  Convert categorical data set $\mathcal{D}_{cat}$ into the form of the transaction data set $\mathcal{S}$
3  foreach transaction $\mathcal{T}_i$ in S do
4      foreach item $i_q$ in $\mathcal{T}_i$ do
5          /* Collect all distinct items and calculate their supports by using the
              dictionary $Items\ \langle key : i_q, value$ : its support$\rangle$                    */
6          $new\_support \leftarrow 0$
7          if $i_q \notin Items$ then
8              $Items \leftarrow (i_q, 1)$
9          end
10         else
11             $new\_support \leftarrow$ support of $i_q$ + 1
12             $Items \leftarrow (i_q, new\_support)$
13         end
14         /* Map each item in data set $\mathcal{S}$ to all transaction IDs that contain it by
              using the dictionary
              $map\_Items\_TIDs\ \langle key : i_q, value$ : the set of transaction IDs that contain $i_q\rangle$   */
15         $TID\_Set \leftarrow \emptyset$
16         if $i_q \notin map\_Items\_TIDs$ then
17             $map\_Items\_TIDs \leftarrow (i_q, \{\emptyset\})$
18         end
19         else
20             $TID\_Set \leftarrow$ value of $i_q$ in $map\_Items\_TIDs$
21             Put ID of $\mathcal{T}_i$ into $TID\_Set$
22             $map\_Items\_TIDs \leftarrow (i_q, TID\_Set)$
23         end
24     end
25 end
26 if the input parameter $minsup$ is omitted then
27     Calculate average minimum support $minsup = \frac{1}{|Items|} \sum_{\forall i_q \in S} sup(i_q)$
28 end
29 Initiate the FP-tree $T$ and its corresponding header table
30 Perform the FPMAX(T) algorithm (Algorithm 9) and put all MFIs into $MFI\_List$
31 Sort $MFI\_List$ in descending order of length of itemsets and their supports
32 Choose top $k$ MFIs in $MFI\_List$
33 foreach MFI in top $k$ MFIs do
34     foreach item $i_q$ in MFI do
35         Get the corresponding $value$ of $i_q$ in $map\_Items\_TIDs$
36     end
37     Intersect $value$ sets of all items in MFI to get a common set that contains this MFI pattern
38 end
39 Remove overlaps between $k$ initial groups by calling the OVERLAP_FILTER procedure (Algorithm
   10)
40 Assign remaining objects in $\mathcal{S}$ into $k$ initial groups by calling the INITIAL_ASSIGNMENT procedure
   (Algorithm 11)
41 return $k$ initial clusters;
```

Algorithm 8 shows the pseudo-code of the INITIATE_CENTERS procedure. It takes a categorical data set $\mathcal{D}_{cat}$, a pre-defined number of cluster k and an optional minimum support threshold ($minsup$) as the input. The procedure first creates two dictionaries named $Items$ and map_Items_TIDs. The former keeps the information of all distinct items and their supports, while the latter keeps the information of transaction IDs that contain each item. In the pre-processing step, the procedure converts the categorical data set $\mathcal{D}_{cat}$ into a transaction data set $\mathcal{S}$ (line 2). If categorical attributes contain similar categories, the index of each attribute is concatenated to the last position of categories to ensure that the mining process performs properly. Next, the procedure scans the transaction database to collect all distinct items and calculate their support (lines 5-13). This information is then used to perform the maximal frequent itemset mining algorithm. In addition, the procedure also tracks all transaction IDs containing each item (lines 14-25). This information is latter used for the matching step. If the $minsup$ is omitted, then the algorithm will automatically determine this value for the mining process. For mining frequent itemsets without support threshold, [21] pointed out that algorithms should start from the frequent item at the middle position of the header table, then from this point move up to the top of the table, and from the next point of the middle item move down to the bottom of the table. This choice is better than starting from the top (top-down) or the bottom (bottom-up) of the header table. Inspired by this idea, this chapter estimates the $minsup$ by averaging the supports of all 1-itemset in $Items$ (lines 26-28). The $minsup$ is then used for the mining process. In the next step, the procedure sorts frequent items in descending order of their supports and creates the header table of frequent items as well as initiates the FP-tree(line 29). Such information is then used to perform the FPMAX algorithm [50] to find the set of MFIs.

FPMAX algorithm was proposed by Grahne and Zhu [50] based on the famous algorithm FP-growth. FP-growth [54] falls into the group of pattern-growth approach, which aims to reduce the cost of data set scanning. It utilizes a tree structure named FP-tree to maintain FIs in a data set, where the root node is the topmost node of the tree and its children are *item prefix subtrees* associated with a *frequent-item header table*. Each branch in the FP-tree keeps an FI, while nodes along the branch contain items in the FI and they are sorted in descending order of their frequencies, where least frequent items are kept at leaves. The *item prefix subtree* contains nodes with three fields including *item-name*, *count*, *node-link*. The *frequent-item header table* (header table) contains rows with two fields including *item-name* and *head of node-link* pointing to the first node of the same *item-name* in the FP-tree. FP-growth uses two database scans to mine all FIs. The initial FP-tree associated with its header table of frequent items is created in the first scan. The first FP-tree containing all frequency information of the original data set is constructed in the second scan. FP-growth uses *conditional pattern base* (or projected database [40]) to restrict the original data set to those transactions containing a given itemset. Mining FIs in the original data set then becomes mining the FP-tree of the projected database. Specifically, for each frequent item in the header table, a new FP-tree is constructed based on the frequency information in the subset of transactions containing that item. This process recursively performs on the current FP-tree until the resulting smaller tree has only a single path. Such single paths then generate FIs. FPMAX extends FP-growth to mine all MFIs. It also uses the pattern-growth approach to perform the mining process. Algorithm 9 shows the pseudo-code of this algorithm. In the first data set scan, FPMAX constructs the FP-tree as the first step. To form the *conditional pattern base* for every recursive call of the algorithm, it uses a linked list named Head to keep all items necessary for this process. In the initial call of FPMAX, both Head and FP-tree does not contain items which form a subset of existing MFI. If the FP-tree contains a single path, the union of items in this path and items in Head is an MFI. FPMAX uses the *maximal frequent itemset tree* (MFI-tree) to keep MFIs (lines 1-2). This tree resembles the FP-tree structure. Its header table contains items with the same order as that of FP-tree. A newly discovered FI is inserted into the MFI-tree if it has no super-set in the tree. In the next step, each item in the header table of FP-tree is put into the Head of MFI-tree if the former tree does not contain a single-path (lines 4-5). The algorithm then constructs the *head-conditional pattern base* for items in Head (line 6). FPMAX constructs the FP-tree for the new Head and calls recursively to find new MFIs on that FP-tree if the union of a new Head and all frequent items in the *head-conditional pattern base* has no super-sets from existing MFIs (lines 7-15). The details of how to construct the FP-tree, MFI-tree and FPMax can be referred to [50]. The output of FPMAX is put into a list named MFI_List that contains MFIs and their corresponding supports (line 30). Next, the MFI_List is sorted in descending order of length of itemsets. If there

Algorithm 9: THE FPMAX ALGORITHM

```
Input  : an FP-tree FPT
Output: List of MFIs
1  if FPT has a single path SP then
2      Put Head ∪ SP into MFI-tree
3  else
4      foreach frequent item $fi$ in header-table of FPT do
5          Insert $fi$ into Head
6          Build the head-conditional pattern base of the new Head
7          Tail = frequent items in head-conditional pattern base
8          Call subset_checking(Head ∪ Tail)
9          if Head ∪ Tail is not in MFI-tree then
10             Construct the FP-tree $FPT_{Head}$
11             Call FPMAX($FPT_{Head}$)
12         end
13         Eliminate $fi$ from $Head$
14     end
15 end
```

are multiple MFIs of the same length, the procedure sorts them according to the descending order of their supports. The target is to find a large combination of categories that occur in multiple categorical objects (line 31). The procedure selects top k MFIs from MFI_List (line 32). For each MFI, the procedure performs the matching step to find the set of transaction IDs that contain this pattern (lines 33-38). Specifically, for each item i_q of a selected MFI, the procedure gets the corresponding $value$ that is the set of transaction IDs containing i_q in map_Items_TIDs. Then it takes the intersection of these $value$ sets to get a common set that contains the whole MFI. After this step, k initial groups may contain the same transaction IDs and thus the overlaps between them need to be removed. To solve this problem, the algorithm calls the OVERLAP_FILTER procedure (Algorithm 10) (line 39).

The input of the OVERLAP_FILTER procedure is k initial groups of transaction IDs collected from the previous step. The procedure performs two loops to remove the overlap between pairs of initial groups. The outer loop considers each group G_l $(1 \leq l \leq k-1)$, while the inner loop traverses remaining groups G_j $(l+1 \leq j \leq k)$. At each iteration of the second loop, the procedure finds the intersection of groups G_l and G_j called $Overlap\_Set$, then removes $Overlap\_Set$ from G_j and temporarily puts it into the set named $Union\_Set$. The procedure performs other iterations of the inner

Algorithm 10: THE OVERLAP_FILTER PROCEDURE

```
input  : $k$ initial groups of transaction IDs
output: $k$ groups without overlaps
1  foreach group $G_l$ in range $l \in [1, k-1]$ do
2      $Union\_Set \leftarrow \emptyset$
3      foreach group $G_j$ in range $j \in [l+1, k]$ do
4          $Overlap\_Set \leftarrow$ Intersection of $G_l$ and $G_j$
5          Remove $Overlap\_Set$ from $G_j$
6          Put $Overlap\_Set$ into $Union\_Set$
7      end
8      Remove $Union\_Set$ from $G_l$
9  end
10 return $k$ initial groups without overlap;
```

loop to remove the overlap of G_l and G_j (lines 3-7). At the end of the inner loop, the $Union\_Set$ contains the overlaps between group G_l and the remaining groups G_j. The procedure then removes it from G_l (line 8). The procedure handles other groups in the outer loop in the same manner until all groups are not overlapping. Finally, it returns k initial groups without overlap between them. However, one problem may occur after this step. That is the union of k groups maybe not equal to the whole data set which leads to a conflict of cluster definition (1). To address this issue, the INITITE_CENTERS procedure calls the INITIAL_ASSIGNMENT procedure to initially assign remaining objects to k groups (Algorithm 11) (line 40).

Algorithm 11: THE INITIAL_ASSIGNMENT PROCEDURE

```
input  : $k$ initial groups without overlap, $\mathcal{D}_{cat}$: a categorical data set
output: $k$ initial clusters
1  Generate $k$ representatives from $k$ initial groups
2  foreach object $x_i$ in $\mathcal{D}_{cat}$ do
3      if $i \notin k$ initial groups then
4          Calculate distance from $x_i$ and $k$ representatives
5          Assign $i$ into the nearest group
6      end
7  end
8  return $k$ initial clusters;
```

The input of this algorithm is k initial non-overlapping groups and the original categorical data set $\mathcal{D}_{cat}$. The procedure uses the same ways as in k-representatives algorithm [99] for the formation of cluster centers and distance function. At first, the procedure generates k representatives of k initial groups (line 1). Specifically, the representative of a group G_l is defined by $Q_l = (q_1, q_2, \ldots, q_m)$ where:

$$q_j = \left\{ \left(o_{ij}, \mathfrak{f}_{o_{ij}} \right) | o_{ij} \in D_j \right\} \tag{6.13}$$

where $\mathfrak{f}_{o_{ij}}$ and D_j are the relative frequency of category o_{ij} (Eq. 2.2) and the set formed from categories appearing at the j^{th} attribute of G_l, respectively. For each data object $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ $(1 \leq i \leq n_l)$ that does not belong to any group, the procedure calculates the distance from x_i to k representatives (line 4) by the following dissimilarity measure:

$$d(x_i, Q_l) = \sum_{j=1}^{m} \sum_{o_{ij} \in D_j} \mathfrak{f}_{o_{ij}} \times \delta(x_{ij}, o_{ij}) \tag{6.14}$$

where $\delta(x_{ij}, o_{ij})$ is the simple matching distance measure as defined in Eq.2.28. Based on the distances from x_i to representatives of k initial groups, the procedure will assign this object to the nearest group (line 5). Other remaining objects in $\mathcal{D}_{cat}$ are performed in the same manner until they belong to one and exactly one group (lines 2-7). From now on, the k initial groups are called k initial clusters (line 41).

6.5 Comparative Experiment

6.5.1 Data sets

This section presents a performance comparison of k-PbC and other ten clustering algorithms: k-means++ [11], Cao's algorithm (Cao) [18], Khan's algorithm (Khan) [71], Distance and Entropy algorithms [68], k-modes [60], k-representatives (k-reps) [99], M-k-Centers (Mod-2) and New (Mod-3) [91, 92] and CD-Clustering [89]. The k-PbC, k-means++, Cao's algorithm, k-modes, k-representatives, Mod-2 and Mod-3 were written in Python, Khan's algorithm was written in Java and obtained from [18], CD-Clustering was written in C++ and obtained from [89]. For the experiment, we used a VPC cluster equipped with an Intel Xeon Gold 6130 2.1GHz (16Cores×2), 64 GB of RAM, running CentOS 7.2 for each CPU node. The source code and data sets are provided at https://github.com/ClarkDinh/k-PbC. The experimental data sets in Table 6.5 were collected from the UCI Machine Learning Repository [34]. Since k-modes, k-reps, Mod-

2 and Mod-3 use the random initialization to form cluster centers, each of them was run 100 trials for each data set, and the overall result was calculated by averaging the results of all trials. The k-means++, Cao's algorithm, Khan's algorithm, CD-Clustering and k-PbC have repeatable clustering results, so each of them was run one time for each data set. The experimental data sets were converted to numerical data sets by using one-hot encoding for k-means++ because it only deals with numerical data. For the evaluation, we used both internal and external validation metrics in the comparative experiment.

Table 6.5: Characteristics of the experimental data sets

#	Data set	#instances	#attributes	#Classes
1	Balance scale	625	4	3
2	Breast cancer	699	9	2
3	Car evaluation	1,728	6	4
4	Chess	3,196	36	2
5	Dermatology	366	34	6
6	Lung cancer	32	56	3
7	Lymphography	148	18	4
8	Mushroom	8,124	22	2
9	Nursery	12,960	8	5
10	Solar flare	1,066	12	3
11	Soybean	683	35	19
12	Soybean-small	47	21	4
13	Splice	3,190	60	3
14	Tic-tac-toe	958	9	2
15	Vote	435	16	2
16	Zoo	101	16	7

1. Balance scale (balance) contains 625 instances with four attributes. It was generated to model psychological experimental results. Objects are classified into three groups as having balance scale tip to: left (46 %), right (46 %), or be balanced (8 %). No missing values occur in this data set.

2. Breast cancer (breast) contains 699 instances with nine attributes. It was collected from the University of Wisconsin Hospitals from 1989 to 1991. Objects are classified into two groups: benign (65.5%) and malignant (34.5%). Missing values occur in 16 instances of this data set.

3. Car evaluation (car) contains 1,728 instances with six attributes: buying, doors, lug_boot, maint, persons, safety. It was derived from a simple hierarchical decision model to evaluate cars according to two concept structures including price and technical characteristics. Objects are classified into four groups: v-good (3.762%), good (3.993 %), acc (22.222 %), unacc (70.023 %). No missing values occur in this data set.

4. Chess (King-Rook vs. King-Pawn) (chess) contains 3,196 instances with 36 attributes. It was originally generated and described by Alen Shapiro. Objects are classified into two groups: White can win (52%) or White cannot win (48%). No missing values occur in this data set.

5. Dermatology (derma) contains 366 instances with 33 features. It was collected to diagnose the erythema-squamous diseases in dermatology. Objects are classified into six groups: pityriasis rubra pilaris (5.46%), cronic dermatitis (14.21%, pityriasis rosea (13.39%), lichen planus (19.67%), seboreic dermatitis (16.67%), psoriasis (30.6%). Missing values occur in eight attributes.

6. Lung contains 32 instances with 56 attributes. It describes three types of pathological lung cancers. Objects are classified into three groups corresponding to three types of disease: type 1 (9 instances), type 2 (13 instances), type 3 (10 instances). Missing values occur in attributes #5 and #39.

7. Lymphography domain (lymph) contains 148 instances with 18 attributes. It was collected from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. This is one of three domains (Breast cancer and primary-tumor data sets) provided by the Oncology Institute. Objects are classified into four groups: normal find (1.4 %), fibrosis (2.7 %), malign lymph (41.2 %) and metastases (54.7 %). No missing values occur in this data set.

8. Mushroom (mush) contains 8,124 instances with 22 attributes. It describes the hypothetical samples of 23 species of Agaricus and Lepiota mushrooms. Objects are classified into two groups: edible (51.8 %), poisonous (48.2 %). Missing values occur only in attribute #11 (2,480 values).

9. Nursery contains 12,960 instances with eight attributes. It comes from a hierarchical decision model originally created to rank nursery school applications in Ljubljana, Slovenia during the 1980s. Objects are classified into five groups: not_recom (33.333 %), spec_prior (31.204 %), priority (32.917 %), very_recom (2.531 %), recommend (0.015 %). No missing values occur in this data set.

10. Solar flare (solar) contains 1,066 instances (flare.data2) with 10 attributes. Each instance describes captured features for one active region on the sun. From all these predictors three classes of flares are predicted, which are represented in the last three columns. In this chapter, we used the last column as the ground-truth. Thus, the data set contains 12 attributes in total. No missing values occur in this data set.

11. Large soybean (soy-1) contains 307 and 376 instances for the training and test sets with 35 attributes. It describes the information on soybean disease diagnosis. Objects are classified into 19 groups. Missing values occur in most attributes, except the attribute #12.

12. Small soybean (soy-2) contains 47 instances with 35 attributes, which comes from the large soybean data set. In the experiment, we used 21 non-singleton attributes as in [89]. Objects are classified into four classes: D1 (10 instances), D2 (10 instances), D3 (10 instances), D4 (17 instances). No missing values occur in this data set.

13. Molecular Biology (splice) contains 3,190 instances with 60 attributes. Originally, it was designed for the purpose of evaluating hybrid learning algorithms that use examples to inductively refine pre-existing knowledge. Objects are classified into three groups: EI (25%), IE (25%), Neither (50%). No missing values occur in this data set.

14. Tic-tac-toe Endgame (tic) contains 958 instances with nine attributes. It includes the complete set of possible board configurations at the end of Tic-tac-toe games. Objects are classified into two groups: positive (65.3%) and negative (34.7%). No missing values occur in this data set.

15. Congressional voting records (vote) contains 435 instances with 16 attributes. Objects are classified into two groups: democrat (45.2%) and republican (54.8%). Missing values occur in all attributes of this data set.

16. Zoo contains seven classes of animals. It contains 101 instances with 17 instances. In this chapter, we remove the first attribute since it contains only distinct animal names. Objects are classified into seven groups. No missing values occur in this data set.

6.5.2 Validation metrics

Internal validation metrics

We used an internal validation metric called the silhouette coefficient [25, 98] to evaluate how well data objects are clustered by determining how close each object in one cluster is to objects in the neighboring clusters. For each data set, we set the number of clusters as the number of classes in the ground-truth. To calculate the average silhouette value for each algorithm, we first compute a pairwise distance matrix for objects in $\mathcal{D}_{cat}$. This is a symmetric matrix that is later used to calculate the *intra* and *inter* distances between objects. It is worth noting that the avg_sil results rely on the distance metrics applied to perform clustering and calculate the pairwise distance matrix. To compute the pairwise distance matrix in the experiment, the Euclidean distance was used for k-means++; the simple matching distance was used for k-modes, k-representatives, Cao's method and Khan's methods; the ITBD measure (Eq. 2.17) was used for Mod-2, Mod-3 and k-PbC. Table 6.6 shows the average silhouette results of algorithms. It can be observed that k-PbC has higher results than those of other algorithms in most cases. It means that the kernel-based method and the ITBD measure used in k-PbC are more efficient than the frequency-based method and the simple matching in k-representatives and k-modes. In addition, k-PbC yields better clustering results when compared to the methods of Cao and Khan. It also outperforms Mod-2 and Mod-3 although they use similar ways to define cluster centers and determine distance. It means that the initialization phase improves the performance of k-PbC and makes them more efficient than other algorithms.

Table 6.6: Average silhouette values

Data sets	#Clusters	k-means++	k-modes	k-reps	Cao's method	Khan's method	Mod-2	Mod-3	k-PbC
Balance scale	3	**0.1743**	0.0764	0.1122	0.0759	0.0805	0.1127	0.1128	0.1308
Breast cancer	2	**0.5962**	0.1577	0.3928	0.3876	0.1183	0.5428	0.5427	0.5422
Car evaluation	4	0.1568	0.0717	0.1609	0.0757	0.0644	0.1768	0.1763	**0.2005**
Chess	2	0.1097	0.1426	0.1918	0.1754	0.1457	0.1956	0.1933	**0.2081**
Dermatology	6	0.2405	0.0653	0.1837	0.1457	0.0921	0.2625	0.2662	**0.2924**
Lung cancer	3	0.0819	0.0687	0.1001	0.068	0.0841	**0.1353**	0.1330	0.1072
Lymphography	4	0.1704	0.1023	0.1583	0.0702	0.1389	0.1816	0.1829	**0.1991**
Mushroom	2	0.2725	0.2316	0.2558	0.2517	0.2355	0.2541	0.2481	**0.2969**
Nursery	5	0.1301	**0.2491**	0.1271	0.0571	0.0555	0.1483	0.1468	0.1757
Solar flare	3	**0.4426**	0.0566	0.3310	0.2415	0.1512	0.3088	0.3086	0.3172
Soybean	19	0.2189	0.1933	0.3228	0.291	0.2185	0.3498	0.3501	**0.3782**
Soybean-small	4	0.3204	0.3335	0.4292	0.4752	0.4484	0.4498	0.4498	**0.5169**
Splice	3	**0.0700**	0.0169	0.0374	0.0204	0.0218	0.0349	0.0360	**0.0700**
Tic-tac-toe	2	0.1009	0.1029	0.1357	0.1013	0.1009	0.1477	0.1462	**0.1595**
Vote	2	0.2650	0.4877	0.4956	0.4893	0.4856	0.5107	**0.5146**	**0.5146**
Zoo	7	0.4157	0.4024	0.4598	0.4826	0.5528	0.4773	0.4709	**0.5614**

External validation metrics

We first used three external validation metrics including accuracy, precision, recall to measure the performance of k-PbC and compared algorithms. The definitions of these metrics are given in section 2.3.1. Tables 6.7, 6.8 and 6.9 show the AC, PR and RE of algorithms, respectively. In these tables, the bold-faced numbers indicate the best performances among the compared algorithms for each data set. It can be observed that k-PbC outperforms other compared algorithms in most cases. For clustering accuracy, it has better results than those of compared algorithms except for Car evaluation, Chess, Solar flare and Tic-tac-toe data sets. On Chess, Solar flare and Tic-tac-toe data sets, similar categories spread in different attributes of the data sets, Khan's algorithm is thus more efficient than other algorithms since it can choose significant attributes before performing multiple clustering on the selected attributes. Similar results are observed for clustering precision and recall. In general, with the initialization step, the proposed k-PbC algorithm can enhance clustering results and is more efficient than using the random initialization method. It outperforms other initialization methods in most cases.

We also compared the AC, PR and RC of k-PbC with the Distance and Entropy algorithms [68]. To avoid incorrect implementation and make a fair assessment, we used several experimental results provided in [68] for these two algorithms. The results are shown in Table 6.10. It can be observed that k-PbC has better results than those of Distance and Entropy algorithms on Breast cancer, Lung cancer and Vote data sets. They have the same results on Soybean-small data set. Moreover, Distance and Entropy

Table 6.7: Clustering accuracy of compared algorithms

Data sets	k-means++	k-modes	k-reps	Cao	Khan	Mod-2	Mod-3	CD-Clustering	k-PbC
Balance scale	0.5168	0.4497	0.4342	0.3760	0.4192	0.4310	0.4323	0.4129	**0.5680**
Breast cancer	0.9585	0.7043	0.8949	0.9113	0.6323	0.9576	0.9570	0.9514	**0.9614**
Car evaluation	0.2789	0.3592	0.3811	**0.4936**	0.3576	0.3725	0.3831	0.3125	0.3640
Lung cancer	0.5313	0.5188	0.5400	0.5313	0.4375	0.5022	0.4922	0.5938	**0.6875**
Chess	0.5116	0.5465	0.5367	0.5663	**0.7040**	0.5279	0.5385	0.5156	0.5047
Dermatology	0.7404	0.5375	0.7161	0.5874	0.6175	0.7280	0.7404	0.8552	**0.9645**
Lymphography	0.3176	0.4379	0.5301	0.3514	0.5068	0.5433	0.5341	0.5000	**0.5946**
Mushroom	0.7093	0.7291	0.7897	0.8754	0.8288	0.7474	0.7682	0.7244	**0.8861**
Nursery	0.2409	0.3324	0.3161	0.3673	0.2804	0.3165	0.3128	0.4156	**0.4492**
Solar flare	0.4540	0.4524	0.5087	0.5432	**0.6463**	0.5526	0.5579	0.4343	0.5901
Soybean	0.5183	0.4439	0.6025	0.5754	0.4612	0.6242	0.6225	0.6852	**0.6881**
Soybean-small	0.7234	0.7657	0.8834	**1.0000**	0.9787	0.9070	0.8666	**1.0000**	**1.0000**
Splice	0.3937	0.4054	0.5430	0.4009	0.4279	0.6496	0.6741	0.7260	**0.7746**
Tic-tac-toe	0.5585	0.5675	0.5605	0.6106	**0.6347**	0.5625	0.5598	0.6044	0.6326
Vote	0.8690	0.8622	0.8751	0.8644	0.8506	0.8764	0.8764	0.8713	**0.8805**
Zoo	0.7723	0.6868	0.6986	0.6733	0.8614	0.7601	0.7524	0.8218	**0.8911**

Table 6.8: Clustering precision of compared algorithms

Data sets	k-means++	k-modes	k-reps	Cao	Khan	Mod-2	Mod-3	CD-Clustering	k-PbC
Balance scale	**0.4896**	0.3993	0.4276	0.3282	0.3609	0.4177	0.4238	0.3751	0.4825
Breast cancer	**0.9571**	0.7163	0.9182	0.9292	0.5535	0.9466	0.9459	0.9470	0.9517
Car evaluation	0.2789	0.2868	0.3488	**0.3826**	0.2415	0.3407	0.3440	0.3125	0.3704
Lung cancer	0.5333	0.5349	0.5937	0.5468	0.4468	0.5726	0.5515	0.5980	**0.7421**
Chess	0.5110	**0.7018**	0.5416	0.5796	0.5312	0.5326	0.5425	0.5162	0.5095
Dermatology	0.7583	0.5077	0.6603	0.5604	0.6841	0.6696	0.6589	0.7543	**0.9612**
Lymphography	0.3147	0.3906	0.4659	0.2698	0.4226	0.4750	0.4599	0.4724	**0.4752**
Mushroom	0.7740	0.7496	0.8018	0.9019	0.8688	0.7586	0.7781	0.7990	**0.9000**
Nursery	0.2323	0.2905	0.2995	0.2978	0.2304	0.2954	0.2930	**0.6494**	0.4352
Solar flare	0.3381	0.3355	0.3400	0.3381	0.3361	0.3415	0.3403	0.3377	**0.3425**
Soybean	0.5659	0.3894	0.6216	0.6133	0.4342	0.6401	0.6432	**0.7574**	0.6956
Soybean-small	0.7569	0.7576	0.8769	**1.0000**	0.9773	0.9060	0.8522	**1.0000**	**1.0000**
Splice	0.3377	0.4095	0.5960	0.4518	0.4260	0.6655	0.6898	0.7198	**0.7815**
Tic-tac-toe	0.5509	0.5540	0.5521	0.5859	0.6071	0.5604	0.5597	0.5746	**0.6396**
Vote	0.8636	0.8573	0.8705	0.8568	0.8484	0.8724	0.8724	0.8670	**0.8762**
Zoo	0.6948	0.5523	0.5788	0.5996	0.7390	0.6518	0.6193	0.5688	**0.7503**

outperform k-PbC on Mushroom and Zoo data sets, respectively. They can detect and remove the outliers appeared inside data sets and thus improve clustering results.

We also used three other external validation metrics called Purity, Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) to measure how well the assignment of objects to clusters matches their original class information. The definitions of these criteria are given in Definitions 2.3.2

The Purity, NMI and ARI results are shown in figures 6.3, 6.4, 6.5. Generally, k-PbC produces better results in most data sets. For purity, it outperforms other algorithms on Balance scale, Breast, Car evaluation, Dermatology, Lung cancer, Lymphography, Mushroom, Nursery, Large soybean, Soybean small, Splice, Vote and Zoo data sets. It

Table 6.9: Clustering recall of compared algorithms

Data sets	k-means++	k-modes	k-reps	Cao	Khan	Mod-2	Mod-3	CD-Clustering	k-PbC
Balance scale	0.4585	0.3852	0.4039	0.3228	0.3541	0.3957	0.4009	0.3540	**0.4673**
Breast cancer	0.9506	0.6951	0.8535	0.8773	0.5336	0.9637	0.9632	0.9550	**0.9656**
Car evaluation	0.3554	0.3034	0.3794	**0.4875**	0.2499	0.3650	0.3639	0.4114	0.4465
Lung cancer	0.5581	0.5350	0.5380	0.5507	0.4470	0.4595	0.4512	0.6208	**0.6900**
Chess	0.5110	**0.6962**	0.5389	0.5537	0.5540	0.5296	0.5394	0.5162	0.5091
Dermatology	0.7649	0.4735	0.6811	0.5260	0.6165	0.6960	0.6943	0.8189	**0.9679**
Lymphography	0.3425	0.4569	0.5746	0.2955	0.4451	0.5438	0.5328	**0.6255**	0.5579
Mushroom	0.7002	0.7256	0.7858	0.8709	0.8228	0.7472	0.7686	0.7151	**0.8829**
Nursery	0.2061	0.2581	0.2551	0.2273	0.2044	0.2516	0.2426	0.2569	**0.3370**
Solar flare	0.4841	0.3924	0.4884	0.4310	0.4379	**0.4923**	0.4807	0.4775	0.4467
Soybean	0.5539	0.3945	0.6141	0.5372	0.4593	0.6190	0.6153	**0.7462**	0.6875
Soybean-small	0.7574	0.7691	0.8786	**1.0000**	0.9853	0.9006	0.8567	**1.0000**	**1.0000**
Splice	0.3366	0.4205	0.6163	0.4379	0.4472	0.7015	0.7291	0.7722	**0.8421**
Tic-tac-toe	0.5560	0.5579	0.5559	0.5917	0.6129	0.5652	0.5644	0.5785	**0.6538**
Vote	0.8822	0.8751	0.8898	0.8730	0.8672	0.8921	0.8920	0.8863	**0.8960**
Zoo	0.7080	0.5936	0.6129	0.6233	0.7648	0.6503	0.6494	0.6860	**0.7860**

Table 6.10: The comparison of AC, PR, RC of k-PbC and algorithms in [68]

Data set	AC			PR			RC		
	Distance	Entropy	k-PbC	Distance	Entropy	k-PbC	Distance	Entropy	k-PbC
Breast cancer	0.9242	0.9328	**0.9614**	0.9309	0.9424	**0.9517**	0.9009	0.9094	**0.9656**
Lung cancer	0.5313	0.6250	**0.6875**	0.6569	0.6833	**0.7421**	0.5274	0.5932	**0.6900**
Mushroom	**0.8941**	0.8876	0.8861	**0.9138**	0.9095	0.9000	**0.8903**	0.8835	0.8829
Soybean-small	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Vote	0.8690	0.8690	**0.8805**	0.8630	0.8630	**0.8762**	0.8811	0.8811	**0.8960**
Zoo	0.8911	**0.9010**	0.8911	0.7695	**0.8906**	0.7503	0.8146	**0.8432**	0.7860

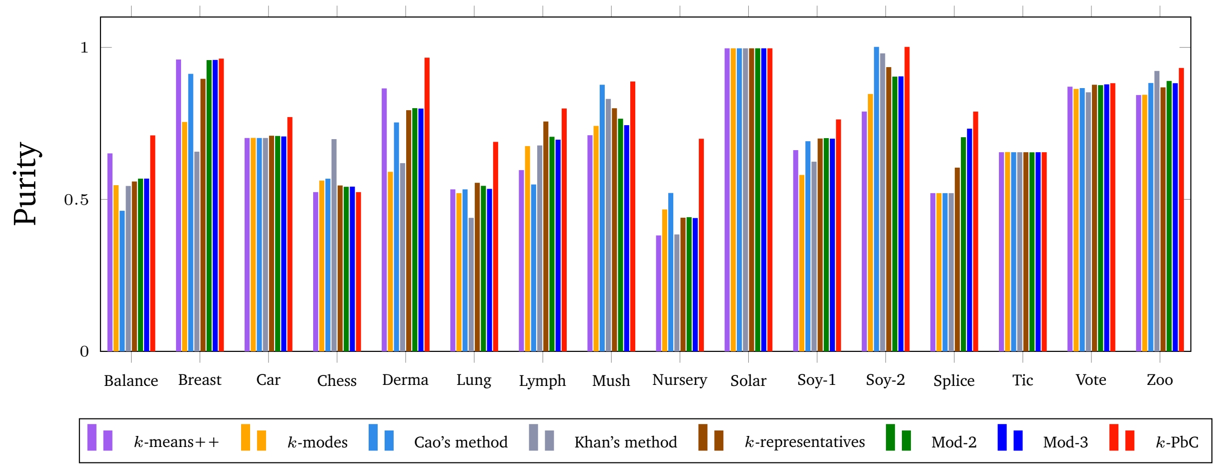


Figure 6.3: Purity results

has similar results on Solar flare and Tic-tac-toe data sets. On Chess, Khan's algorithm outperforms other algorithms. Similar results can be observed for the NMI and ARI metrics. Generally, the proposed approach has been proven to enhance clustering results in terms of both internal and external metrics.

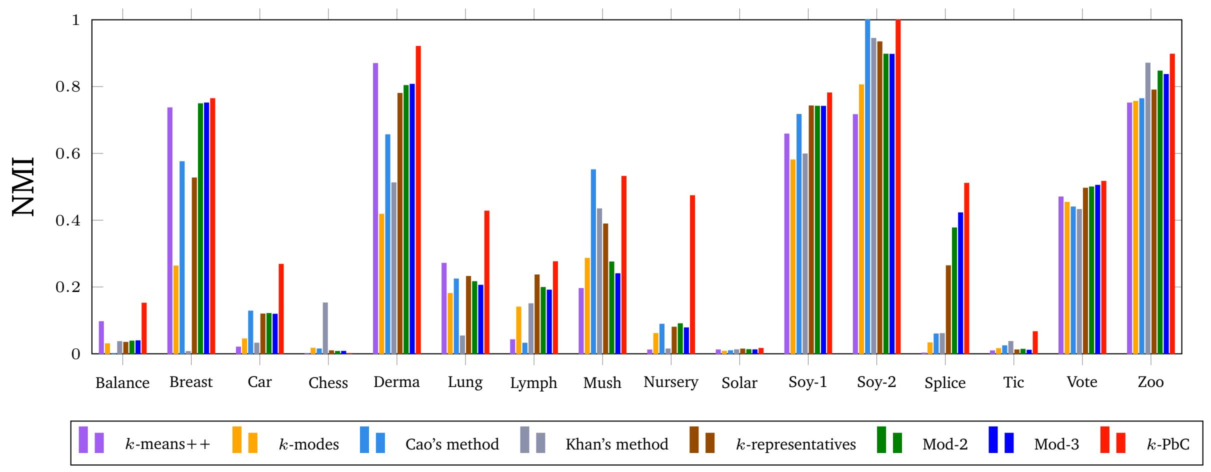


Figure 6.4: NMI results

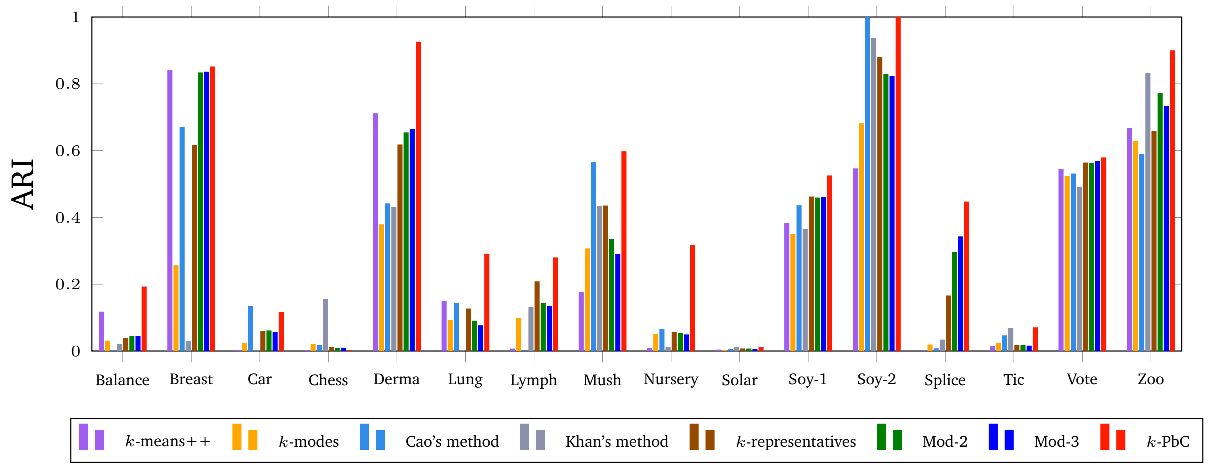


Figure 6.5: ARI results

6.5.3 Significant Analysis

In this section, we analysed the significance of the clustering results of k-PbC obtained in the previous section. We considered the significance of the results on both internal and external validation metrics. Table 6.11 shows the summary of accuracy, precision, recall and average silhouette values results of k-PbC proposed in this chapter. It can be observed that k-PbC has the highest values for four metrics on six over sixteen data sets: Dermatology, Mushroom, Soybean-small, Splice, Vote, Zoo. It has the highest values for three metrics on three over sixteen data sets: Lung cancer, Lymphography and Tic-tac-toe. For Balance scale, Breast cancer, Nursery and Soybean, it has the highest values on two metrics. Thus, it can be seen that k-PbC can work well on a variety of data sets. On Car evaluation, Chess and Solar flare, k-PbC does not seem to be working

Table 6.11: Significant analysis of k-PbC on four validation metrics

#	Data set	Accuracy			Precision			Recall			Average silhouette values		
		1st rank	2nd rank	≤3rd rank	1st rank	2nd rank	≤3rd rank	1st rank	2nd rank	≤3rd rank	1st rank	2nd rank	≤3rd rank
1	Balance scale	✓				✓		✓				✓	
2	Breast cancer	✓				✓		✓					✓
3	Car evaluation			✓		✓			✓		✓		
4	Lung cancer	✓			✓			✓					✓
5	Chess			✓			✓			✓	✓		
6	Dermatology	✓			✓			✓			✓		
7	Lymphography	✓			✓					✓	✓		
8	Mushroom	✓			✓			✓			✓		
9	Nursery	✓				✓		✓				✓	
10	Solar flare		✓		✓					✓			✓
11	Soybean	✓				✓			✓		✓		
12	Soybean-smal	✓			✓			✓			✓		
13	Splice	✓			✓			✓			✓		
14	Tic-tac-toe		✓		✓			✓			✓		
15	Vote	✓			✓			✓			✓		
16	Zoo	✓			✓			✓			✓		

well when compared to Cao's and Khan's methods. Table 6.12 shows the summary of Purity, NMI and ARI results of k-PbC. It can be observed that it has the highest values for three metrics on almost data sets, except for the Chess data set. As discussed above, when similar categories spread in different attributes of the data sets such as in the case of Chess, Khan's algorithm is more efficient than other algorithms since it can choose significant attributes before performing multiple clustering on the selected attributes. Generally, the practical results show that the initialization method used in k-PbC can enhance clustering results.

Table 6.12: Significant analysis of k-PbC on three other external validation metrics

#	Data set	Purity			NMI			ARI		
		1st rank	2nd rank	3rd rank	1st rank	2nd rank	3rd rank	1st rank	2nd rank	3rd rank
1	Balance scale	✓			✓			✓		
2	Breast cancer	✓			✓			✓		
3	Car evaluation	✓			✓			✓		
4	Lung cancer	✓			✓			✓		
5	Chess			✓			✓			✓
6	Dermatology	✓			✓			✓		
7	Lymphography	✓			✓			✓		
8	Mushroom	✓				✓		✓		
9	Nursery	✓			✓			✓		
10	Solar flare	✓			✓			✓		
11	Soybean	✓			✓			✓		
12	Soybean-smal	✓			✓			✓		
13	Splice	✓			✓			✓		
14	Tic-tac-toe		✓		✓			✓		
15	Vote	✓			✓			✓		
16	Zoo	✓			✓			✓		

6.6 Conclusion

This chapter has solved the problem of random initialization in categorical data clustering by proposing an algorithm named k-PbC. It uses a maximal frequent pattern mining based approach to generate initial cluster centers. Particularly, it extends the FPMAX algorithm [50] in the way that the minimum support threshold does not need to be specified in advance. The set of maximal frequent itemsets found in the equivalent transaction data set represents the largest set of categories occurring in categorical objects. The group of object IDs containing each maximal frequent pattern is then taken for an initial cluster after removing overlaps between obtained groups. For clustering, k-PbC uses the kernel-based method for the formation of cluster centers and the ITBD measure for distance calculation. The comparative results have revealed that the proposed algorithm has improved clustering results in terms of both internal and external validation metrics.

Chapter 7

Estimating the Optimal Number of Clusters in Categorical Data Clustering

7.1 Introduction

Clustering is an important task in data mining and knowledge discovery. It has been widely used in a diversity of fields ranging from data exploration, information retrieval, text mining, web analysis to computational biology, medical diagnostics, marketing, social science and many others [15]. It also can be used to discover hidden patterns and joined as a step in other fundamental research topics. Basically, clustering aims at identifying groups of homogeneous objects from a data set. A group of objects or a cluster contains several objects such that they are similar to other objects in the same cluster and dissimilar to objects in other clusters.

In the literature, clustering algorithms can be grouped by several criteria as shown in Figure 1.3. In general, they may be classified by clustering types: hard vs. soft clustering, flat vs. hierarchical clustering, model-based vs. cost-based clustering; by data types: interval scaled, nominal, ordinal, image; or by regime: parametric vs. non-parametric clustering [56]. In the branch of flat clustering, partitional clustering algorithms use a specific distance function to assign objects in a given data set into homogeneous clusters by iteratively reducing an objective function and gradually improving the quality of clusters [97].

In distance-based clustering, k-means [84] is popularly used thanks to its simplicity and efficiency. It uses the *mean* to represent cluster centers and a distance metric such as Euclidean to determine distances between objects and cluster centers. It has a linear scale to the size of data sets and can be applied to any steps of other research topics. However, as its drawback, k-means can not be used to cluster directly for categorical and mixed data sets that appear common in many real-life applications. To overcome the limitation of k-means while keeping its benefits, several methods have been proposed to make applicable for categorical data, so-called k-means-like algorithms [20, 25, 26, 59, 60, 90–92, 99]. These algorithms keep the same scheme of k-means, but difference in the use of distance metrics and cluster center representations.

Two major factors that may affect the performance of partitional clustering algorithms are forming initial cluster centers and determining the number of clusters (k) [97]. The former factor has been discussed in the previous chapter. For the latter factor, existing algorithms specify k in advance. Nevertheless, a fixed and inaccurate k is hard to predict the actual k for a given data and may reduce the interpretation of clustering results. In addition, the under-estimation and over-estimation of k are considerably influencing on clustering quality [80]. Hence, determining the optimal k is an important and non-trivial task in partitional clustering. This chapter focuses on this problem by proposing a framework to estimate k in categorical data clustering. The following points are the main contributions of this chapter:

- ➢ We proposed a $\underline{\text{s}}$ilhouette based framework for estimating the optimal number of clusters in $\underline{\text{c}}$ategorical data $\underline{\text{c}}$lustering, namely k-SCC. The proposed framework performs the clustering process in a range of user-specified minimum and maximum of k to select the best k that yields the highest average silhouette value. It uses the kernel-based method and an information-theoretic based distance measure to perform the clustering process.

- ➢ The proposed framework was tested on both synthetic and real-life data sets from the UCI Machine Learning repository. In addition, it was used to classify objects in a real-life Sake wine data set as a case study.

The remainder of this chapter is structured into following sections. Section 7.2 reviews related work. Section 7.3 gives preliminary definitions. Section 7.4 proposes the k-SCC algorithm. Section 7.5 describes experimental results. Finally, Section 7.6 draws a conclusion.

7.2 Related work

The details of related clustering methods for categorical data clustering are provided in section 2.4. Here we remarked several important points of partitional algorithms for this field. In the numerical setting, k-means [84] is a popularly used algorithm in the branch of partitional clustering. The workflow of k-means is as follows. It first selects k objects from a given data set as the initial clusters. It then performs the assignment step to assign all objects to the nearest clusters by using a distance metric such as Euclidean, Cosine and Manhattan. In the next step, k-means updates the cluster means based on the new partition matrix. It performs the assignment and update steps in turn until all clusters are convergent. Generally, it reduces the objective function:

$$\mathcal{F}(\mathcal{U},\mathcal{Z}) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m} u_{il} \times d(x_{ij}, z_{lj}) \tag{7.1}$$

where n and m be the number of objects and attributes in the data set, $\mathcal{U} = [u_{il}]_{n\times k}$ is a partition matrix such that $\sum_{l=1}^{k} u_{il} = 1$ and $u_{il} \in \{0, 1\}$; $\mathcal{Z}$ be a set of k cluster centers, each contains the *means* of values inside the cluster at m attributes, $d(\cdot, \cdot)$ is a distance metric. K-means is widely used since it is easy for implementation and interpretation, as well as efficient for large-scale data sets. However, the limitation of k-means is incapable of dealing with categorical data that is common in many applications.

To address the limitation of k-means, several algorithms were proposed for categorical data, so-called k-means-like algorithms. These algorithms keep the advantages of k-means and make applicable for categorical data. One of the pioneer algorithms in this field is the k-modes algorithm [59, 60]. It uses *mode* instead of *mean* to represent cluster centers. Moreover, it uses the simple matching measure as the distance function.

Let two categories a_i and a_j. The simple matching for a_i and a_j is defined as:

$$\delta(a_i, a_j) = \begin{cases} 0 & \text{if } a_i = a_j \\ 1 & \text{if } a_i \neq a_j \end{cases} \tag{7.2}$$

K-representatives [99] considers categories inside an attribute as a distribution and defines the representative of a cluster by using the relative frequency associated with categories in the cluster. It measures the distance between objects and representatives as the multiplication of the simple matching by relative frequencies of categories in the representative for all attributes. K-centers [20] considers the cluster center as a probabilistic center by using the kernel-based method. For distance measure, it uses the simple matching as an indicator function to represent each data object by a set of vectors and then uses the Euclidean norm to estimate the distance between object and cluster center. Recently, [25, 26, 90–92] have improved previous categorical data clustering algorithms by using the kernel-based method and an information-theoretic based distance (ITBD) measure. In [91, 92], we introduced three algorithms. The M-k-representatives (Modified-1) improves k-representatives by using the ITBD instead of the simple matching measure. The M-k-centers (Modified-2) uses the ITBD instead of the Euclidean norm and keeps the same cluster centers definition as in [20]. The New algorithm (Modified-3) uses the ITDB in the combination with the modified version of [20] for cluster centers using the kernel-based method. The New algorithm outperforms the other two improved versions.

Rousseeuw proposed the silhouette coefficient [98] as the graphical aid to the interpretation and validation of clustering analysis. It measures the goodness of a clustering structure by considering both the *inter-cluster* and *intra-cluster* distances of all objects within clusters. The way of calculating this coefficient is as follows. Given a clustering result containing k clusters, for each object x_i, it first calculates the intra_dis(x_i) by averaging the distances from x_i to all objects inside the same cluster. In addition, it calculates the inter_dis(x_i) by averaging the distances from x_i to all objects in the neighbour cluster of x_i. The intra_dis(x_i) and intra_dis(x_i) are then used for estimating the silhouette value of x_i. The average silhouette value is determined by averaging the silhouette values of all objects in the data set. In clustering, the average silhouette value can be used as an internal validation metric as in Section 2.3.1. In the numeric data setting, [13, 113] estimated the number of clusters by using the silhouette coefficient. However, in the categorical data setting, to our best knowledge, no prior work focused on this method. In this chapter, we used the silhouette coefficient to determine the optimal number of clusters for a given categorical data set by selecting the k that yields the highest average silhouette value.

7.3 Preliminaries

The clustering framework proposed in this chapter used the kernel-based method for the representation of cluster centers and an information-theoretic based dissimilarity measure for calculating distances between objects and cluster centers. The detail of these definitions is shown in Section 2.2 of Chapter 2. Again, given a categorical data set $\mathcal{D}_{cat}$ described by n instances and m attributes. Each object $x_i \in \mathcal{D}_{cat}$ $(1 \leq i \leq n)$ is a tuple of m values $x_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \in \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_m$, where $\mathcal{A}_j$ is an attribute characterized by $\mathcal{O}_j$ as its domain such that DOM($\mathcal{A}_j$) = $\mathcal{O}_j$ (> 1 values). Moreover, let o_{ij} $(1 \leq i \leq |\mathcal{O}_j|)$ denote a category in $\mathcal{O}_j$. The following table presents a categorical data set containing ten objects described by six features.

Table 7.1: A categorical data set

Attr / Obj	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$	$\mathcal{A}_6$
x_1	a_1	d_2	b_3	e_4	a_5	c_6
x_2	d_1	a_2	a_3	b_4	c_5	a_6
x_3	d_1	d_2	d_3	c_4	c_5	a_6
x_4	b_1	e_2	c_3	e_4	a_5	c_6
x_5	a_1	d_2	a_3	a_4	a_5	e_6
x_6	a_1	a_2	c_3	e_4	a_5	c_6
x_7	b_1	e_2	e_3	e_4	a_5	c_6
x_8	d_1	c_2	d_3	e_4	a_5	c_6
x_9	d_1	c_2	d_3	e_4	a_5	c_6
x_{10}	d_1	d_2	b_3	b_4	c_5	a_6

In this chapter, we also used the kernel-based method to represent cluster centers, called the probabilistic center [20, 26, 90, 92]. Particularly, given a cluster $\mathcal{C}_l$, then its center is defined as $\mathcal{Z}_l = \left\{z_j^l\right\}_{j=1}^{m}$, each element is a vector in the probability space, $\mathcal{O}_j^l$

is the sample space, $\mathcal{P}_j^l$ is the probability measure defined on $\mathcal{C}_l$. For easy tracking, we restated these definitions like the following.

Given a cluster $\mathcal{C}_l = \{x_1, x_2, \ldots, x_{n_l}\}$ where $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ $(1 \le i \le n_l)$. The center of $\mathcal{C}_l$ is defined as:

$$\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\} \tag{7.3}$$

where z_j^l is defined as:

$$z_j^l = [\mathcal{P}_j^l(o_{1j}^l), \mathcal{P}_j^l(o_{2j}^l), \ldots, \mathcal{P}_j^l(o_{|\mathcal{O}_j^l|j}^l)] \tag{7.4}$$

and the probability of each category o_{ij}^l $(1 \le i \le |\mathcal{O}_j^l|)$ is measured as:

$$\mathcal{P}_j^l(o_{ij}^l) = \begin{cases} \lambda_l \frac{1}{|\mathcal{O}_j^l|} + (1-\lambda_l)\mathfrak{f}_l(o_{ij}^l) & \text{if } o_{ij}^l \in \mathcal{O}_j^l \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

In this chapter, we also used the information-theoretic based dissimilarity measure for determining the distance between objects and cluster centers. First, the similarity of two categorical values o_{ij} and $o_{i'j}$ is measured as:

$$\text{sim}_j(o_{ij}, o_{i'j}) = \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{7.6}$$

where $\mathfrak{f}(o_{ij}, o_{i'j}) = \frac{\#(o_{ij}, o_{i'j})}{|D|}$ denotes the relative frequency of the set $\{o_{ij}, o_{i'j}\}$ at the j^{th} attribute of $\mathcal{D}_{cat}$. The dissimilarity between is then quantified as:

$$\text{dsim}_j(o_{ij}, o_{i'j}) = 1 - \text{sim}_j(o_{ij}, o_{i'j}) = 1 - \frac{2\log \mathfrak{f}(o_{ij}, o_{i'j})}{\log \mathfrak{f}(o_{ij}) + \log \mathfrak{f}(o_{i'j})} \tag{7.7}$$

Again, we defined the distance between two categorical objects $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and $x_{i'} = (x_{i'1}, x_{i'2}, \ldots, x_{i'm})$ as follows:

$$\text{dis\_objs}(x_i, x_{i'}) = \sum_{j=1}^{m} \text{dsim}_j(x_{ij}, x_{i'j}) \tag{7.8}$$

In this chapter, for the purpose of measuring the average silhouette values for each clustering structure, we first computed the pairwise distance matrix using the information-theoretic based distance measure (Eq. 7.8). The distance matrix for data objects in data set 7.1 is shown in Table 7.2, which is a symmetric matrix where dis_objs(x_i,$x_{i'}$) = dis_objs($x_{i'}$,x_i) and dis_objs(x_i,x_i) $= 0$ $(1 \leq i, i' \leq n)$. This matrix is later used to calculate the *intra* and *inter* distances between objects.

Table 7.2: The pairwise distance matrix of objects in Table 7.1

	1	2	3	4	5	6	7	8	9	10
1	**0.0000**	4.3518	3.7898	1.4107	2.308	1.4107	1.3645	1.7617	1.7617	3.3256
2	4.3518	**0.0000**	1.4872	4.2182	3.1476	3.7875	4.172	3.6046	3.6046	1.0262
3	3.7898	1.4872	**0.0000**	4.4165	2.9759	4.4165	4.3866	3.2191	3.2191	0.8917
4	1.4107	4.2182	4.4165	**0.0000**	2.9497	0.8614	0.3845	1.6281	1.6281	4.383
5	2.308	3.1476	2.9759	2.9497	**0.0000**	2.5191	2.9035	3.2858	3.2858	2.9827
6	1.4107	3.7875	4.4165	0.8614	2.5191	**0.0000**	1.2458	1.6281	1.6281	4.383
7	1.3645	4.172	4.3866	0.3845	2.9035	1.2458	**0.0000**	1.5983	1.5983	4.3368
8	1.7617	3.6046	3.2191	1.6281	3.2858	1.6281	1.5983	**0.0000**	0.0000	3.7694
9	1.7617	3.6046	3.2191	1.6281	3.2858	1.6281	1.5983	0.0000	**0.0000**	3.7694
10	3.3256	1.0262	0.8917	4.383	2.9827	4.383	4.3368	3.7694	3.7694	**0.0000**

The distance between an object $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$ and the center $\mathcal{Z}_l = \{z_1^l, z_2^l, \ldots, z_m^l\}$ at the j^{th} attribute is measured as:

$$\text{dis}_j(x_i, \mathcal{Z}_l) = \sum_{o_{ij}^l \in \mathcal{O}_j^l} \mathcal{P}_j^l(o_{ij}^l)\text{dsim}_j(x_{ij}, o_{ij}^l) \tag{7.9}$$

The distance between x_i and $\mathcal{Z}_l$ for all attributes is formulated as:

$$\text{dis}(x_i, \mathcal{Z}_l) = \sum_{j=1}^{m} \text{dis}_j(x_i, \mathcal{Z}_l) \tag{7.10}$$

The clustering algorithm proposed in this chapter tries to solve the cost function:

$$\mathcal{F}(\mathcal{U}, \mathcal{Z}) = \sum_{l=1}^{k}\sum_{i=1}^{n} u_{i,l} \times \text{dis}(x_i, \mathcal{Z}_l) \tag{7.11}$$

subject to

$$\begin{cases} u_{i,l} \in \{0, 1\} \\ \sum_{l=1}^{k} u_{i,l} = 1 \end{cases} \tag{7.12}$$

To measure the goodness of the structure produced by a clustering algorithm, we used the average silhouette value which is defined in detail in Eqs (2.18) and (2.19).

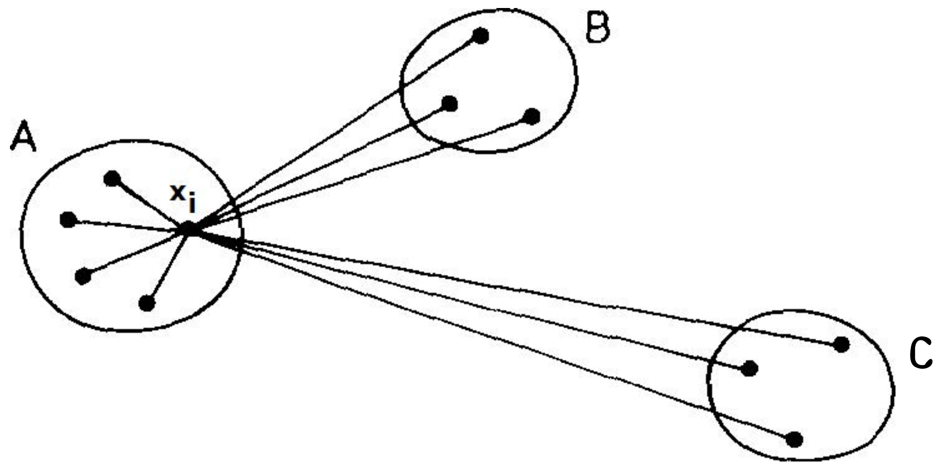


Figure 7.1: The silhouette value of object x_i

Figure 7.1 shows an example of how to calculate the silhouette value for data object x_i. Assume that data objects in a given categorical data set are assigned into three clusters A, B, C, where x_i belongs to cluster A. To calculate its silhouette, the algorithm needs to determine the average distance from x_i to all objects in the same cluster A, so-called intra-cluster distance of x_i. In addition, it finds the neighbouring cluster other than cluster A that has the shortest average distance to x_i, so-called inter-cluster distance of x_i. Again, the silhouette value of x_i is denoted and defined as:

$$\text{sil}(x_i) = \frac{\text{inter\_dis}(x_i) - \text{intra\_dis}(x_i)}{\max\{\text{intra\_dis}(x_i), \text{inter\_dis}(x_i)\}} \tag{7.13}$$

sil(x_i) measures how closely x_i is matched to data within its cluster (intra-cluster) and how loosely x_i is matched to data of the neighbouring cluster (inter-cluster).

The average silhouette values for all objects assigned to k clusters is formulated as follows:

$$\text{avg\_sil} = \frac{\sum_{i=1}^{n} \text{sil}(x_i)}{n} \tag{7.14}$$

The goal of determining the optimal k in categorical data clustering for a given data set is to find the k whose clustering structure yields the maximum average silhouette value defined in Eq. (7.14).

7.4 The proposed algorithm

Fig. 7.2 shows the general framework for the proposed k-SCC algorithm. The framework first assigns objects in a given data set into k clusters. It then computes the avg_sil for the clustering results obtained from the current k. The avg_sil of each k is put into the set $SilSet$ that maintains the average silhouette values for the whole process. The framework finally selects the k that produces the corresponding maximum avg_sil value in $SilSet$ as the optimal number of clusters for the input data set.

```
Algorithm 12: THE k-SSC ALGORITHM
input : $\mathcal{D}_{cat}$: a categorical data set, $k_{\min}$, $k_{\max}$: minimum and maximum number
        of cluster
output: the optimal $k$ and corresponding $k$ clusters of $\mathcal{D}_{cat}$
1  $k = k_{\min}$, SilSet $\leftarrow \emptyset$
2  Compute the pairwise distance matrix for objects in $\mathcal{D}_{cat}$
3  while $k \leq k_{\max}$ do
4      Generate $k$ initial cluster centers $\mathcal{Z}^{(0)} = \{\mathcal{Z}_1^{(0)}, \dots, \mathcal{Z}_k^{(0)}\}$
5      $\mathrm{i} = 0, \mathcal{U} \leftarrow \emptyset$
6      while clusters are not stable do
7          Fix $\mathcal{Z}^{(\mathrm{i})}$ and find $\mathcal{U}^{(\mathrm{i})}$ to minimize $\mathcal{F}(\mathcal{U}^{(\mathrm{i})}, \mathcal{Z}^{(\mathrm{i})})$
8          Fix $\mathcal{U}^{(\mathrm{i})}$ and update $\mathcal{Z}^{(\mathrm{i})}$ to minimize $\mathcal{F}(\mathcal{U}^{(\mathrm{i})}, \mathcal{Z}^{(\mathrm{i})})$
9          $\mathrm{i} = \mathrm{i} + 1$
10     end
11     Calculate the avg_sil for the current clustering result
12     $SilSet \leftarrow$ avg_sil, $k = k + 1$
13 end
14 return $k_{opt}$ clusters such that $s_{k_{opt}} = \max(SilSet)$;
```

Algorithm 14 shows the pseudo code of k-SCC algorithm. It takes a data set $\mathcal{D}_{cat}$ and two user-specified parameters $k_{\min}$ and $k_{\max}$ as the inputs. k-SCC will find the optimal number of clusters in the range $[k_{\min}, k_{\max}]$. By default, $k_{\min}$ is set to 2, while $k_{\max}$ is set to $n-1$. The empty set $SilSet$ is generated to keep the avg_sil produced by clustering results at each k (line 1). k-SCC computes the pairwise distance matrix for all object pairs in $\mathcal{D}_{cat}$ (line 2). The matrix is later used to compute the intra_dis and inter_dis of clusters. For each k in the range of $k_{\min}$ and $k_{\max}$, k-SCC performs the assignment and update steps to assign objects into k clusters (lines 4-9). It first randomly selects k objects from the data set to form k clusters. It then computes the

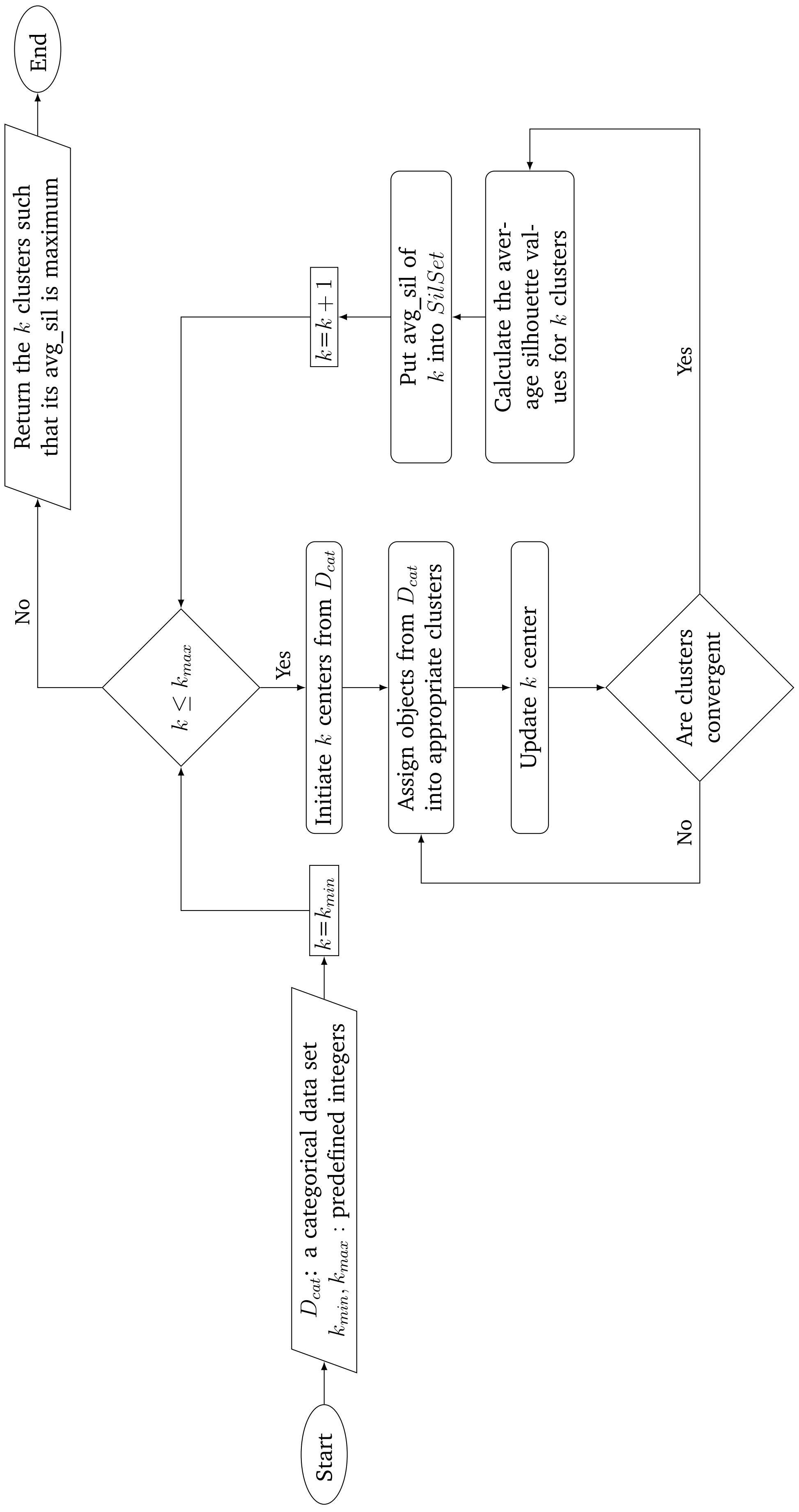

Start
D_{cat}: a categorical data set
k_{min}, k_{max} : predefined integers
$k = k_{min}$
$k \leq k_{max}$
Yes
No
Initiate k centers from D_{cat}
Assign objects from D_{cat} into appropriate clusters
Update k center
Are clusters convergent
No
Yes
Calculate the average silhouette values for k clusters
Put avg_sil of k into $SilSet$
$k = k + 1$
Return the k clusters such that its avg_sil is maximum
End

Figure 7.2: The flowchart of k-SCC algorithm

distance between objects and clusters to assign objects into the nearest clusters. In other words, the algorithm finds a new partition matrix which indicates the membership of each object to k clusters. In the next step, it updates k cluster centers from the obtained partition matrix. k-SCC iteratively performs the two previous steps until all clusters are stable, which means that objects in clusters do not further change their membership values. The clustering results are then used to calculate the average silhouette values. For every object in each cluster, its silhouette value is determined by using the Eq. 7.13. k-SCC calculates the average silhouette values for all objects in the data set by using Eq. 7.14 and keeps it in the $SilSet$ (lines 11-12). The algorithm performs in the same manner for other k. Finally, it selects the k that yields the maximum average silhouette value in the $SilSet$. This k can be considered as the most appropriate number of clusters for the data set $\mathcal{D}_{cat}$ (line 14).

7.5 Comparative Experiment

7.5.1 Data sets

Table 7.3: Characteristics of the experimental datasets

#	dataset	#instances	#attributes	#classes	type
1	Car evaluation	1,728	6	4	real-life
2	Chess	3,196	36	2	real-life
3	Connect-4	10,000	42	3	real-life
4	Nursery	12,960	8	5	real-life
5	Soybean (small)	47	35	4	real-life
6	Spect heart	267	22	2	real-life
7	Tic-tac-toe	958	9	2	real-life
8	SD5k	5,000	6	4	synthetic
9	SD10k	10,000	6	2	synthetic

The experiment was conducted to evaluate the performance of k-SSC on both real-life and synthetic data sets. Table 7.3 shows the characteristics of these data sets. The first seven data sets are real-life data collected from the UCI machine learning repository [34], while the last two data sets were generated by [86]. We compared k-SSC with k-modes [59], Modified-3 [92] and another version of k-SCC, namely k-SCC+, using the simple matching distance instead of information-theoretic based distance to compute the distance matrix. We used Python for implementing algorithms, source code and data sets can be found at http://bit.ly/2HiwgKl. Experiments were run in a VPCC cluster [38]. We ran 100 times for the initialization step and selected the best one for each data set.

7.5.2 Experimental results

For each data sets, we measured the average silhouette values by varying the number of clusters. Particularly, we set $k_{min} = 2$ and $k_{max} = 10$, the optimal k was selected from this range. Figure 7.3 shows the comparative results, in which horizontal and vertical axes indicate the number of clusters and corresponding average silhouette values, respectively. It can be observed that k-SCC achieves higher average silhouette values than those of other compared algorithms for all data sets. Specifically, k-SCC outperforms k-modes for all data sets. Hence, the kernel-based method and the information-theoretic based distance used in k-SCC are more efficient than the *mode* and simple matching distance used in k-modes. k-SCC is also better than Modified-3 in most cases. Moreover, in terms of computation complexity, the runtime of k-SCC is lower than Modified-3 since it does not use the feature weighting scheme as in Modified-3. For the comparison of k-SCC and k-SCC+, the former outperforms the latter for most data sets. Thus, the information-theoretic based distance is more efficient than the simple matching measure. On Chess, the peak average silhouette values are at k = 2 and k = 3. It suggests that we may assign objects in this data set into two or three clusters. On Connect-4, the peak average silhouette value is at k = 3, thus we may assign objects in this data set into 3 groups. On SD5K, the highest average silhouette value is at k = 4, thus we may assign the data set into four groups. The remaining data sets have similar observations.

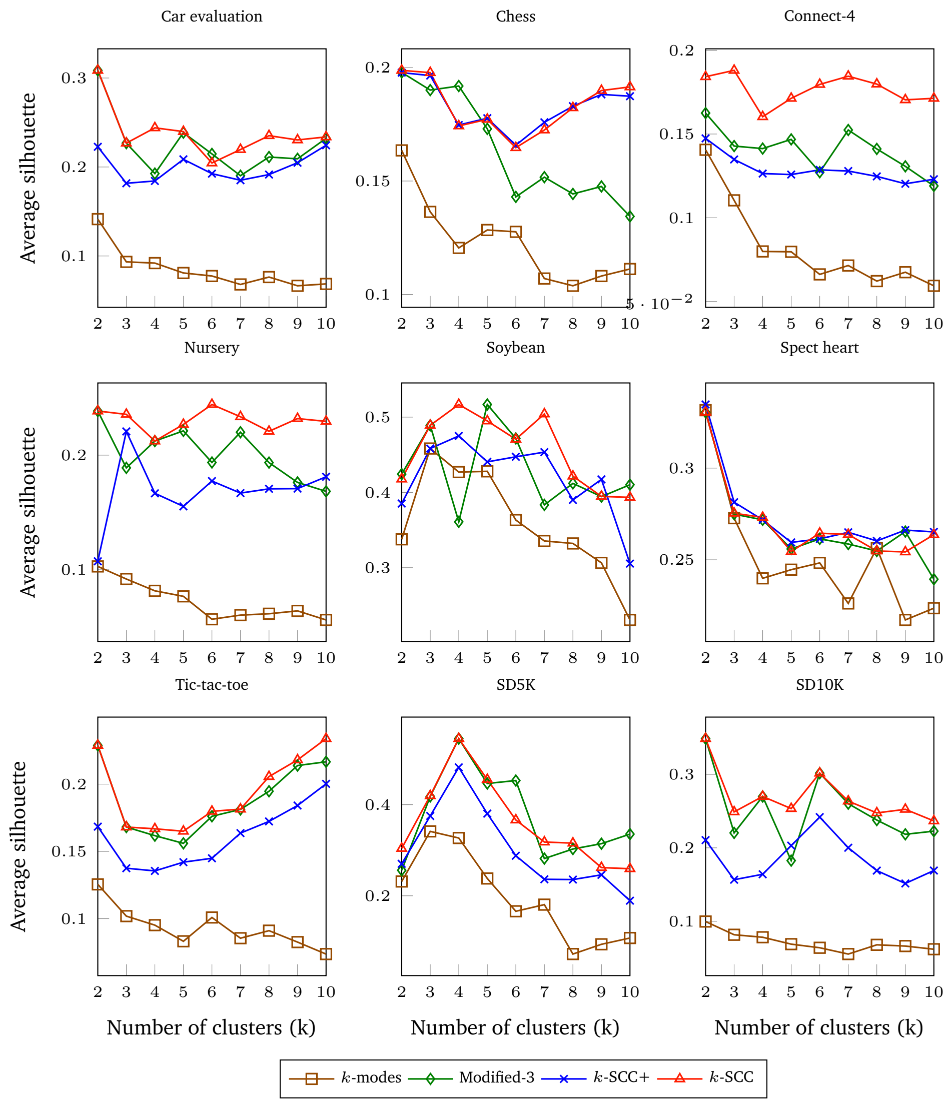


Figure 7.3: Average silhouette for various number of clusters

7.5.3 A case study: Sake wine data set

For the experiment, we also used a real-life data set collected from Fujinami lab [1]. It is a data set of 68 Sake wine instances described by eight attributes including initial taste and aftertaste. The taste was measured by the TS-5000Z sensing system. This sensor uses a similar way as the human tongue to convert the taste into numerical data. It measures two kinds of tastes: initial taste and aftertaste. The former is the taste perceived when wine first enters the mouth including *astringency*, *bitterness*, *saltiness*, *sourness*, *sweetness* and *umami*, while the latter is the persistent taste remaining in the

[1]http://www.jaist.ac.jp/ fuji/index.html

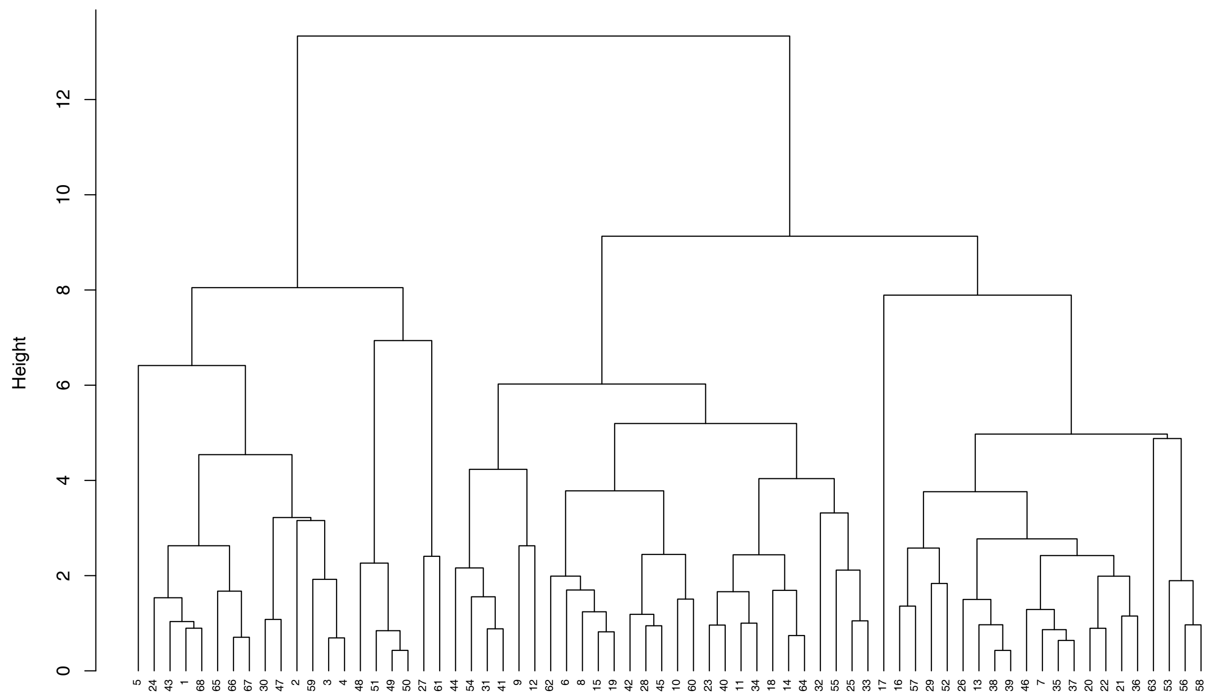
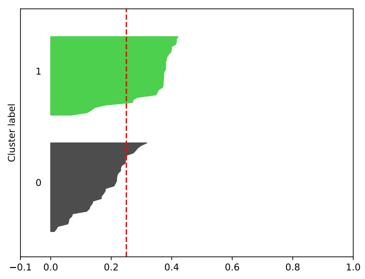
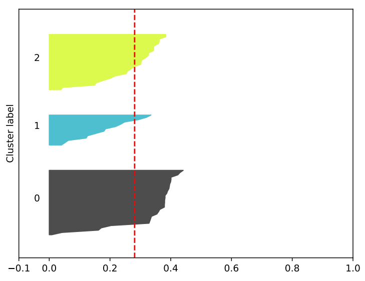


Figure 7.4: Hierarchical clustering on Sake wine data using Complete-lingkage

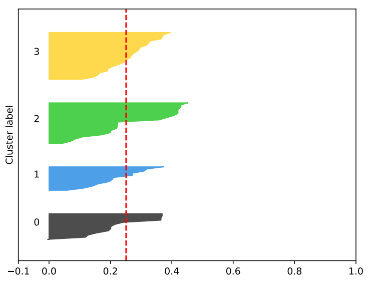
mouth after the wine has been drunk including *aftertaste from astringency* and *richness*. To validate the results, we used the complete-linkage hierarchical clustering to classify the Sake wine data set. The clustering result is shown in Figure 7.4. If we cut the dendrogram at the height of ten, nine or eight, then we obtained two, three and four clusters, respectively. To apply the k-SCC on the Sake data set, we first converted the original numeric data into categorical form by using the Kansei words with five linguistic grades: *very low*, *low*, *neither*, *high*, *very high*. Specifically, each attribute was first discretized into five scales and each scale in the attribute was then mapped with the corresponding grade. Table 7.4 and Figure 7.5 show the average silhouette values and silhouette plots on the Sake data set with the k in the range of $[2, 10]$, respectively. It can be seen that k-SCC outperforms other algorithms in most cases, with the peak average silhouette value is at k = 3 or 4. Thus, we may partition the Sake data into three or four clusters, which are matched to the results obtained in Figure 7.4.



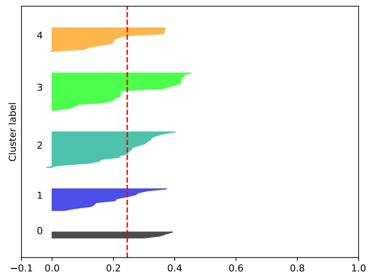
(a) #clusters=2



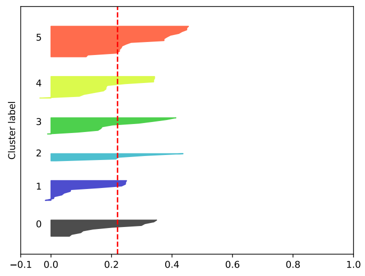
(b) #clusters=3



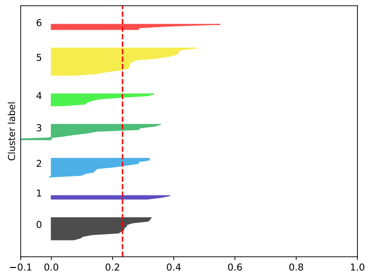
(c) #clusters=4



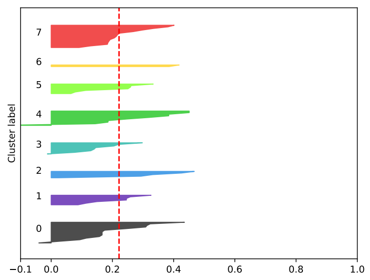
(d) #clusters=5



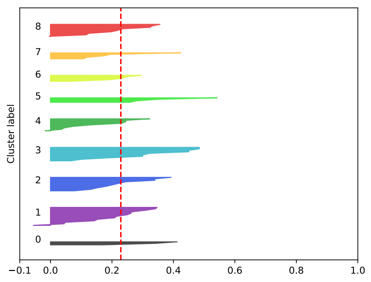
(e) #clusters=6



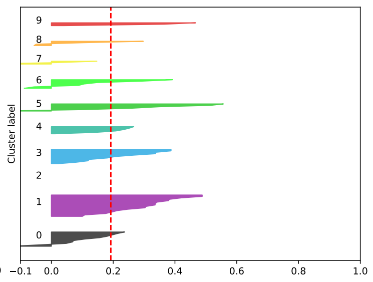
(f) #clusters=7



(g) #clusters=8



(h) #clusters=9



(i) #clusters=10

Figure 7.5: Silhouette plots of Sake wine data set for k in range of [2,10]

Table 7.4: Average silhouette values on the Sake wine data set

Algos / k	k-modes	Modified-3	k-SCC+	k-SCC
2	0.1461	**0.2501**	0.2014	**0.2501**
3	0.1405	0.2744	0.2096	**0.2811**
4	0.1562	0.2252	0.1584	**0.2585**
5	0.1351	**0.2511**	0.1923	0.2451
6	0.1736	0.2135	0.1879	**0.2200**
7	0.1778	**0.2556**	0.2128	0.2328
8	0.1512	0.2028	0.2204	**0.2217**
9	0.1168	0.2168	0.1968	**0.2288**
10	0.1412	0.1469	**0.2148**	0.1924

7.6 Conclusion

This chapter proposed a silhouette coefficient based approach to determine the number of clusters in categorical data clustering. An algorithm named k-SCC has been introduced for this task. k-SCC used the kernel-based method and an information-theoretic based distance for clustering step. The information-theoretic based dissimilarity is also used to compute the pairwise distance matrix for calculating average silhouette values. We conducted the experiment on both synthetic and real-life data sets. Results show that k-SCC outperforms compared algorithms in estimating k for a given data set. The limitation of the proposed framework is that it strongly depends on the distance measure used in computing the intra_dis and inter_dis of average silhouette values.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Categorical data are common in many types of real data sets such as health care, market basket, marketing, retail, and social fields. The attributes of these data sets are inherently discrete and do not take on a natural ordering. In many cases, the data sets may be mixed, in which some attributes are numerical, whereas other attributes are categorical. Categorical and mixed data sets lead to numerous challenges for partitional clustering algorithms. First, new distance measures need to be defined for categorical and mixed data. Second, statistics such as the mean or median are naturally defined for numerical data but need to be appropriately modified for discrete data. The problem becomes more difficult when dealing with mixed data since the different attributes now need to be treated in a heterogeneous way, as well as the similarity functions need to explicitly account for the underlying heterogeneity.

Generally, the dissertation has focused on several problems of categorical and mixed data clustering. First, we designed a k-means-like method for categorical data clustering. Second, we designed a framework for dealing with both clustering and missing values in categorical data. Third, we extended the second framework and designed a framework for clustering mixed numeric and categorical data with missing values. Fourth, we designed a framework to address the limitation of the random initialization in categorical data clustering. Finally, we designed a framework to estimate the optimal number of clusters in categorical data clustering.

Particularly, **Chapter 3** introduces a novel algorithm for categorical data clustering. The proposed algorithm uses a kernel density estimation based method to represent cluster centers. Such a method is applicable for categorical data setting while maintaining consistency with the statistical interpretation of the cluster means for numerical data. In addition, it uses an information-theoretic based dissimilarity to compute the distance between categorical objects and cluster centers. This distance metric is based upon the concept of information theory to measure the similarity of two information components, making use of the logarithm function to calculate the real value in such a way that less frequent words have a higher information gain. The proposed method was tested on several data sets from the UCI Machine Learning Repository and compared with previous algorithms for clustering categorical data. The results have revealed that the proposed method outperforms previous methods and can efficiently cluster for pure categorical data sets. The kernel-based technique and the information-theoretic based dissimilarity proposed in this chapter were later used to perform clustering steps for the next chapters.

It can be seen that data now can be obtained from different sources by humans or devices thanks to the rapid development of information technologies and data acquisition technologies. However, collecting data is not always an easy task and may lead to missing values in data due to different mechanisms. Unfortunately, the missing values can cause bad effects on the analysis results. Thus, in **Chapter 4**, we focused on the problem of clustering categorical data having missing values. We designed an integrated framework that can do both imputation and clustering a categorical data set with missing values. The proposed framework uses a decision tree-based method to fill in missing values in advance before clustering. This method has shown to be suitable for categorical data since it can find the set of data objects that are highly correlated with each data objects having missing values. From that, the appropriate values can be chosen for the imputation. It then uses the kernel-based method and the information-theoretic based dissimilarity for the formation of clustering. The proposed k-CCM method was tested on several data sets from the UCI Machine Learning Repository and compared with previous algorithms in terms of clustering quality. The results have revealed that k-CCM can improve the clustering results by taking advantage of imputation steps for missing values.

It is worth to note again that clustering for missing mixed data is nontrivial and more challenging than clustering for missing categorical data. It is because different attributes in mixed data need to be treated in a heterogeneous way. 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 numeric data imputation, missing categorical data imputation, numeric data clustering and categorical data clustering, while the framework for categorical data handles two tasks of clustering and imputation for categorical type. Thus, in **Chapter 5**, we extended the method proposed in Chapter 4 and made it applicable for clustering mixed numeric and categorical data having missing values. For the imputation step, the proposed k-CMM algorithm also uses the decision-tree based method to find the set of correlated objects and chooses possible imputed values from the correlated set to impute for missing values in categorical attributes. The missing values in numeric attributes are imputed using the mean of corresponding attributes from the correlated set. For the clustering step, it uses the mean and the kernel-based method to define cluster centers at numeric and categorical attributes, respectively. In addition, to quantify the proximity between data objects, it uses the squared Euclidean and the information-theoretic based dissimilarity measure for numeric and categorical attributes, respectively. Experimental results have shown that k-CMM is more efficient than k-prototypes in terms of clustering quality in most cases in terms of clustering quality. Moreover, we also evaluated the runtime, memory consumption, and scalability of k-CMM. Results show that k-CMM is scalable with respect to the number of instances.

The performance of a partitional clustering algorithm is sensitive to the choice of initial cluster centers. An improper choice may lead to poor clustering results. Thus, in **Chapter 6**, we addressed the problem of random initialization in categorical data clustering from the view of pattern mining. Specifically, we used a maximal frequent itemset mining approach name FPMax to find the sets of correlated itemsets. These sets of maximal frequent itemsets found in the equivalent transaction data set represent the largest sets of categories occurring in categorical objects. The group of object IDs containing each maximal frequent pattern is then taken for an initial cluster. For the clustering step, we used the kernel-based method and the information-theoretic based dissimilarity measure. The proposed k-PbC algorithm was tested on benchmark data sets from the UCI Machine Learning Repository. The comparative results have revealed that k-PbC has improved clustering results in terms of both internal and external validation metrics.

The problem of estimating the number of clusters (say k) is one of the major challenges for the partitional clustering. Thus, in **Chapter 7**, we proposed an algorithm named k-SCC to estimate the optimal k in categorical data clustering. For the clustering step, we used the kernel-based method to define cluster centers and the information-theoretic based dissimilarity to measure the distance between cluster centers and data objects. The silhouette analysis-based approach is then used to evaluate the quality of different clustering obtained in the former step to choose the best k. The experiments were conducted on both synthetic and real data sets to evaluate the performance of k-SCC algorithm. Experimental results show that k-SCC outperforms the compared algorithms in determining the number of clusters for each data set.

8.2 Limitations

There are several inherent limitations of the dissertation as shown in the following discussions.

- ➢ First, despite the fact that partitional clustering algorithms have proved to be efficient and had the competitive computational complexity compared to other clustering methods, they fail to perform clustering on very large-scale data sets and do not scale with a huge volume of data. The clustering frameworks proposed in the dissertation also face this challenge and take time-consuming on large-scale datasets. To deal with this challenge, they need to be designed in ways that can take advantages of acceleration techniques such as parallel methods and data reduction-based methods.

- ➢ Second, although the kernel-based method and the information-theoretic based dissimilarity can enhance clustering results, they are more complex than other traditional methods such as "mode" and the simple matching measure. Thus, finding simple ways to reduce complexity while maintaining the efficiency of the clustering algorithms is not a trivial task.

➢ Third, the data sets used in the dissertation were mainly collected from the UCI Machine Learning repository [34]. Although the performances of the proposed algorithms were evaluated for the main types of data encountered in real-life, more real data sets in other fields such as retail, finance and health care should be used as specific domains and case studies for proposed algorithms.

8.3 Future Work

The directions for future research are planned as follows:

➢ **Big data clustering**: We plan to extend the proposed methods for clustering large-scale data sets and big data. Specifically, accompany with the advantages of multi-core processors and graphics processing units (GPU), data mining and analysis applications on large-scale databases have more abilities to improve their performance by adapting the distributed or parallel computing techniques. Thus, we aim to propose parallel methods to improve the existing methods proposed in the dissertation in terms of computational complexity and scalability.

➢ **Dynamic and distributed data clustering**: The proposed methods in the dissertation can only handle the static and centralized data sets. Thus, we plan to propose approaches for the problems raised in the dissertation on the dynamic and distributed environment.

➢ **Uncertainty in clustering and Fuzzy clustering**:

- In principle, uncertainty may concern all aspects of the learning process. The data that can not be precisely classified or can not be presented as precise numbers are called fuzzy or non-precise. In future work, we may concern about designing clustering algorithms for uncertain data and observations.
- In fuzzy (soft) clustering, each data object may belong to two or more clusters with different membership values. In many situations, fuzzy clustering is

more natural than hard clustering since data objects may contribute to more than one cluster. Thus, we intend to extend clustering algorithms in the dissertation to the topics of fuzzy clustering.

➢ **Hierarchical clustering**: In some cases, especially for small and medium-scale data sets, hierarchical clustering is still useful and efficient. Thus, we will design hierarchical clustering algorithms for categorical and mixed data, or combine hierarchical and partitional clustering together.

➢ **Developing more efficient algorithms**: We plan to improve existing proposed algorithms in terms of clustering quality, computational complexity and scalability.

➢ **Missing data imputation methods**: The imputation methods for categorical and mixed data also need to be improved to reduce computational complexity. As used in Chapter 6, the association rule mining can be used to find the set of correlated objects inside the data. In that sense, we intend to use it for imputing missing values in categorical data by finding the set of complete objects correlated with each incomplete object. From that, possible values can be selected for the imputation. The imputation method can then be applied for the problem of clustering categorical or mixed data with missing values.

➢ **Specific domain**: The clustering methods proposed in the dissertation can be applied in any steps of other research topics or used for clustering specific domains such as market segmentation and health care. We plan to propose a method for patient similarity for treatment regimen discovery.

➢ **Application point of view**: We intend to develop several packages in R for clustering topics proposed in the dissertation.

Main Publications

[1] Duy-Tai Dinh, Van-Nam Huynh, and Songsak Sriboonchitta, “Clustering mixed numerical and categorical data with missing values”, *Information Sciences*, 2020. (Major revision)

[2] Duy-Tai Dinh, and Van-Nam Huynh, “*k*-PbC: An improved cluster center initialization for categorical data clustering”, *Applied Intelligence*, pages 1-23, 2020.

[3] Thu-Hien Thi Nguyen, Duy-Tai Dinh, Songsak Sriboonchitta, and Van-Nam Huynh, “A method for k-means-like clustering of categorical data”, *Journal of Ambient Intelligence and Humanized Computing*, pages 1-11, 2019.

[4] Duy-Tai Dinh, Bac Le, Philippe Fournier-Viger, and Van-Nam Huynh, “An efficient algorithm for mining periodic high-utility sequential patterns”, *Applied Intelligence*, volumn 48, pages 4694-4714, 2018.

[5] Duy-Tai Dinh, and Van-Nam Huynh, “*k*-CCM: A center-based algorithm for clustering categorical data with missing values”, *Fifteenth International Conference on Modeling Decisions for Artificial Intelligence (MDAI 2018)*, pages 267-279, Springer, 2018.

[6] Duy-Tai Dinh, Tsutomu Fujinami, and Van-Nam Huynh, “Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient”, *Twentieth International Symposium on Knowledge and Systems Sciences (KSS 2019)*, pages 1-17, Springer, 2019.

[7] Thanh-Phu Nguyen, Duy-Tai Dinh, and Van-Nam Huynh, “A new context-based clustering framework for categorical data”, *Fifteenth Pacific Rim International Conference on Artificial Intelligence (PRICAI 2018)*, pages 697-709, Springer, 2018.

Other Publications

[1] Bac Le, Duy-Tai Dinh, Van-Nam Huynh, Quang-Minh Nguyen, and Philippe Fournier-Viger , “An efficient algorithm for hiding high utility sequential patterns”, *International Journal of Approximate Reasoning*, volumn 95, pages 77-92, 2018.

[2] Bac Le, Ut Huynh, and Duy-Tai Dinh, “A pure array structure and parallel strategy for high-utility sequential pattern mining”, *Expert Systems with Applications*, volumn 104, pages 107-120, 2018.

[3] Duy-Tai Dinh, Van-Nam Huynh, and Bac Le, “Mining periodic high utility sequential patterns”, *Ninth Asian Conference on Intelligent Information and Database Systems,* pages 545-555, Springer, 2017.

[4] Duy-Tai Dinh, Van-Nam Huynh, Bac Le, Philippe Fournier-Viger, Ut Huynh, Quang-Minh Nguyen, “A survey of privacy preserving utility mining”, *High-Utility Pattern Mining*, pages 207-232, Springer, 2019.

[5] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Yimin Zhang, Duy-Tai Dinh, Hoai Bac Le, “Mining correlated high-utility itemsets using various measures”, *Logic Journal of the IGPL*, volumn 28, pages 19-32, Oxford University Press, 2020.

[6] Nhat-Vinh Lu, Trong-Nhan Vuong, Duy-Tai Dinh, “Combining Correlation-Based Feature and Machine Learning for Sensory Evaluation of Saigon Beer”, *International Journal of Knowledge and Systems Science (IJKSS)*, volume 11, pages 71-85, IGI Global, 2020.

Bibliography

[1] Loai AbdAllah and Ilan Shimshoni. k-means over incomplete datasets using mean euclidean distance. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 113–127. Springer, 2016. doi:10.1007/978-3-319-41920-6_9.

[2] Janos Abonyi and Balazs Feil. *Cluster analysis for data mining and system identification*. Springer Science & Business Media, 2007.

[3] C C Aggarwal and C K Reddy. *Data Clustering: Algorithms and Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis, 2013. URL: https://books.google.co.jp/books?id=edl7AAAAQBAJ.

[4] Charu C Aggarwal. An introduction to cluster analysis. In *Data Clustering: Algorithms and Applications*, pages 1–28. Chapman and Hall/CRC, 2013.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms For Mining Association Rules In Data mining. In *International Journal of Scientific & Technology Research*, volume 2 of *VLDB '94*, pages 13–24. Morgan Kaufmann Publishers Inc., 2013.

[6] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63(2):503–527, 2007. doi:10.1016/j.datak.2007.03.016.

[7] Amir Ahmad and Shehroz S Khan. Survey of state-of-the-art mixed data clustering algorithms. *IEEE Access*, 7:31883–31902, 2019. doi:10.1109/ACCESS.2019.2903568.

[8] J. Aitchison and C. G.G. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976. doi:10.1093/biomet/63.3.413.

[9] Michael R Anderberg. *Cluster Analysis for Applications*. Elsevier Science, 1973. doi:10.1016/c2013-0-06161-0.

[10] Bill Andreopoulos, Aijun An, and Xiaogang Wang. Hierarchical density-based clustering of categorical data and a simplification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 11–22. Springer, 2007. doi:10.1007/978-3-540-71701-0_5.

[11] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, volume 07-09-January-2007, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

[12] Ibrahim Berkan Aydilek and Ahmet Arslan. A hybrid method for imputation of missing values using optimized fuzzy c-means with support vector regression and a genetic algorithm. *Information Sciences*, 233:25–35, 2013. doi:10.1016/j.ins.2013.01.021.

[13] Rasool Azimi, Mohadeseh Ghayekhloo, Mahmoud Ghofrani, and Hedieh Sajedi. A novel clustering algorithm based on data transformation approaches. *Expert Systems with Applications*, 76:59–70, 2017. doi:10.1016/j.eswa.2017.01.024.

[14] Liang Bai, Jiye Liang, Chuangyin Dang, and Fuyuan Cao. A cluster centers initialization method for clustering categorical data. *Expert Systems with Applications*, 39(9):8022–8029, 2012. doi:10.1016/j.eswa.2012.01.131.

[15] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[16] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM international conference on data mining*, pages 243–254. SIAM, 2008.

[17] Mohamed Bouguessa. Clustering categorical data in projected spaces. In *Data Mining and Knowledge Discovery*, volume 29, pages 3–38. Chapman and Hall/CRC, 2013. doi:10.1007/s10618-013-0336-8.

[18] Fuyuan Cao, Jiye Liang, and Liang Bai. A new initialization method for categorical data clustering. *Expert Systems with Applications*, 36(7):10223–10228, 2009. doi:10.1016/j.eswa.2009.01.060.

[19] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert systems with applications*, 40(1):200–210, 2013. doi:10.1016/j.eswa.2012.07.021.

[20] Lifei Chen and Shengrui Wang. Central clustering of categorical data with automated feature weighting. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1260–1266, 2013.

[21] Yin Ling Cheung and Ada Wai Chee Fu. Mining frequent itemsets without support threshold: With and without item constraints. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1052–1069, 2004. doi:10.1109/TKDE.2004.44.

[22] Shounak Datta, Debaleena Misra, and Swagatam Das. A feature weighted penalty based dissimilarity measure for k-nearest neighbor classification with missing features. *Pattern Recognition Letters*, 80:231–237, 2016. doi:10.1016/j.patrec.2016.06.023.

[23] Nico De Vos. Python implementations of the k-modes and k-prototypes clustering algorithms, 2018. URL: https://github.com/nicodv/kmodes.

[24] Rupam Deb and Alan Wee Chung Liew. Missing value imputation for the analysis of incomplete traffic accident data. *Information Sciences*, 339:274–289, 2016. doi:10.1016/j.ins.2016.01.018.

[25] Duy-Tai Dinh, Tsutomu Fujinami, and Van-Nam Huynh. Estimating the optimal number of clusters in categorical data clustering by silhouette coefficient. In *International Symposium on Knowledge and Systems Sciences*, pages 1–17. Springer, 2019. doi:10.1007/978-981-15-1209-4_1.

[26] Duy-Tai Dinh and Van-Nam Huynh. k-ccm: a center-based algorithm for clustering categorical data with missing values. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 267–279. Springer, 2018. doi:10.1007/978-3-030-00202-2_22.

[27] Duy-Tai Dinh and Van-Nam Huynh. k-pbc: an improved cluster center initialization for categorical data clustering. *Applied Intelligence*, pages 1–23, 2020. doi:10.1007/s10489-020-01677-5.

[28] Duy-Tai Dinh, Van-Nam Huynh, Bac Le, Philippe Fournier-Viger, Ut Huynh, and Quang-Minh Nguyen. A survey of privacy preserving utility mining. In *High-Utility Pattern Mining*, pages 207–232. Springer, 2019. doi:10.1007/978-3-030-04921-8_8.

[29] Duy-Tai Dinh, Van-Nam Huynh, and Sriboonchitta Songsak. Clustering mixed numerical and categorical data with missing values. *Information Sciences*, 2020.

[30] Duy-Tai Dinh, Bac Le, Philippe Fournier-Viger, and Van-Nam Huynh. An efficient algorithm for mining periodic high-utility sequential patterns. *Applied Intelligence*, 48(12):4694–4714, 2018. doi:10.1007/s10489-018-1227-x.

[31] Tai Dinh, Van-Nam Huynh, and Bac Le. Mining periodic high utility sequential patterns. In *Asian Conference on Intelligent Information and Database Systems*, pages 545–555. Springer, 2017. doi:10.1007/978-3-319-54472-4_51.

[32] Tai Dinh, Minh Nguyen Quang, and Bac Le. A Novel Approach for Hiding High Utility Sequential Patterns. In *Proceedings of the Sixth International Symposium on Information and Communication Technology - SoICT 2015*, volume 03-04-Dece, pages 1–8. ACM Press, 2015. doi:10.1145/2833258.2833271.

[33] Tiago R L dos Santos and Luis E Zárate. Categorical data clustering: What similarity measure to recommend? *Expert Systems with Applications*, 42(3):1247–1260, 2015. URL: https://doi.org/10.1016/j.eswa.2014.09.012.

[34] Dheeru Dua and Casey Graff. UCI machine learning repository, 2019. URL: http://archive.ics.uci.edu/ml.

[35] Craig K Enders. *Applied missing data analysis*. Guilford press, 2010.

[36] B.S. Everitt, S. Landau, M. Leese, and D. Stahl. *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2011. URL: https://books.google.co.jp/books?id=w3bE1kqd-48C.

[37] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y Zomaya, Sebti Foufou, and Abdelaziz Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing*, 2(3):267–279, 2014. doi:10.1109/TETC.2014.2330519.

[38] Research Center for Advanced Computing Infrastructure. High performance computing servers, 2018. URL: https://www.jaist.ac.jp/iscenter/en/mpc/.

[39] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Tai Dinh, and Hoai Bac Le. Mining correlated high-utility itemsets using the bond measure. In *International Conference on Hybrid Artificial Intelligence Systems*, pages 53–65. Springer, 2016. doi:10.1007/978-3-319-32034-2_5.

[40] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Bay Vo, Tin Truong Chi, Ji Zhang, and Hoai Bac Le. A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(4):e1207, 2017. doi:10.1002/widm.1207.

[41] Philippe Fournier-Viger, Yimin Zhang, Jerry Chun-Wei Lin, Duy-Tai Dinh, and Hoai Bac Le. Mining correlated high-utility itemsets using various measures. *Logic Journal of the IGPL*, 28(1):19–32, 01 2020. doi:10.1093/jigpal/jzz068.

[42] Yoshikazu Fujikawa and TuBao Ho. Cluster-Based Algorithms for Dealing with Missing Values. *Advances in Knowledge Discovery and Data Mining*, pages 549–554, 2002. doi:10.1007/3-540-47887-6_54.

[43] Guojun Gan, Chaoqun Ma, and Jianhong Wu. *Data clustering: theory, algorithms, and applications*, volume 20. SIAM, 2007. doi:10.1137/1.9780898718348.

[44] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Cactus-clustering categorical data using summaries. In *KDD*, volume 99, pages 73–83, 1999. doi:10.1145/312129.312201.

[45] Pedro J GarcíA-Laencina, José-Luis Sancho-GóMez, and AníBal R Figueiras-Vidal. Classifying patterns with missing values using multi-task learning perceptrons. *Expert Systems with Applications*, 40(4):1333–1341, 2013. doi:10.1016/j.eswa.2012.08.057.

[46] Pedro J García-Laencina, José-Luis Sancho-Gómez, Aníbal R Figueiras-Vidal, and Michel Verleysen. K nearest neighbours with mutual information for simultaneous classification and missing data imputation. *Neurocomputing*, 72(7-9):1483–1493, 2009. doi:10.1016/j.neucom.2008.11.026.

[47] Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge university press, 2006. doi:10.1017/CBO9780511790942.

[48] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Clustering categorical data: An approach based on dynamical systems. *Databases*, 1:75, 1998. doi:10.1007/s007780050005.

[49] Helen Giggins and Ljiljana Brankovic. VICUS - A noise addition technique for categorical data. In *Conferences in Research and Practice in Information Technology Series*, volume 134, pages 139–148. Australian Computer Society, Inc., 2012. doi:10.1016/j.chemosphere.2014.04.074.

[50] Gösta Grahne and Jianfei Zhu. High performance mining of maximal frequent itemsets. In *6th International Workshop on High Performance Data Mining*, volume 16, page 34, 2003.

[51] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: an efficient clustering algorithm for large databases. In *ACM Sigmod Record*, volume 27, pages 73–84. ACM, 1998. doi:10.1145/276305.276312.

[52] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000.

[53] Jiawei Han, Jian Pei, and Micheline Kamber. *Data Mining: Concepts and Techniques*. A volume in The Morgan Kaufmann Series in Data Management Systems. Elsevier, 2011. doi:10.1016/C2009-0-61819-5.

[54] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000. doi:10.1145/342009.335372.

[55] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Unsupervised learning. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, pages 485–585. Springer, 2009. doi:10.1007/978-0-387-84858-7_14.

[56] Christian Hennig, Marina Meila, Fionn Murtagh, and Roberto Rocci. *Handbook of cluster analysis*. CRC Press, 2015.

[57] Chung-Chian Hsu and Shu-Han Lin. Visualized analysis of mixed numeric and categorical data via extended self-organizing map. *IEEE transactions on neural networks and learning systems*, 23(1):72–86, 2011. doi:10.1109/TNNLS.2011.2178323.

[58] Joshua Zhexue Huang, Michael K Ng, Hongqiang Rong, and Zichen Li. Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 657–668, 2005. doi:10.1109/TPAMI.2005.95.

[59] Zhexue Huang. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference*, pages 21–34. Singapore: World Scientific, 1997.

[60] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998. doi:10.1023/A:1009769707641.

[61] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985. doi:10.1007/BF01908075.

[62] Dino Ienco, Ruggero G Pensa, and Rosa Meo. Context-based distance learning for categorical data clustering. In *International Symposium on Intelligent Data Analysis*, pages 83–94. Springer, 2009.

[63] Alan Julian Izenman. Cluster Analysis. In *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*, pages 407–462. Springer, New York, NY, 2008. doi:10.1007/978-0-387-78189-1_12.

[64] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Elsevier Science, 1993. URL: https://books.google.co.jp/books?id=b3ujBQAAQBAJ.

[65] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. Unsupervised Learning. In *An Introduction to Statistical Learning: with Applications in R*, pages 373–418. Springer, 2013.

[66] Jinchao Ji, Tian Bai, Chunguang Zhou, Chao Ma, and Zhe Wang. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. *Neurocomputing*, 120:590–596, 2013. doi:10.1016/j.neucom.2013.04.011.

[67] Chao Jiang and Zijiang Yang. Cknni: an improved knn-based missing value handling technique. In *International Conference on Intelligent Computing*, pages 441–452. Springer, 2015. doi:10.1007/978-3-319-22053-6_47.

[68] Feng Jiang, Guozhu Liu, Junwei Du, and Yuefei Sui. Initialization of K-modes clustering using outlier detection techniques. *Information Sciences*, 332:167–183, 2016. doi:10.1016/j.ins.2015.11.005.

[69] Alboukadel Kassambara. *Practical guide to cluster analysis in R: Unsupervised machine learning*, volume 1. STHDA, 2017.

[70] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[71] Shehroz S Khan and Amir Ahmad. Cluster center initialization algorithm for K-modes clustering. *Expert Systems with Applications*, 40(18):7444–7456, 2013. doi:10.1016/j.eswa.2013.07.002.

[72] Dae-Won Kim, KiYoung Lee, Doheon Lee, and Kwang H Lee. A k-populations algorithm for clustering categorical data. *Pattern Recognition*, 38(7):1131–1134, 2005. doi:10.1016/j.patcog.2004.11.017.

[73] Ronald S King. *Cluster analysis and data mining: An introduction*. Stylus Publishing, LLC, 2015.

[74] Jacob Kogan, Marc Teboulle, and Charles Nicholas. Data driven similarity measures for k-means like clustering algorithms. *Information Retrieval*, 8(2):331–349, 2005.

[75] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[76] Bac Le, Duy-Tai Dinh, Van-Nam Huynh, Quang-Minh Nguyen, and Philippe Fournier-Viger. An efficient algorithm for hiding high utility sequential patterns. *International Journal of Approximate Reasoning*, 95:77–92, 2018. doi:10.1016/j.ijar.2018.01.005.

[77] Bac Le, Ut Huynh, and Duy-Tai Dinh. A pure array structure and parallel strategy for high-utility sequential pattern mining. *Expert Systems with Applications*, 104:107–120, 2018. doi:10.1016/j.eswa.2018.03.019.

[78] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[79] M Li, S Deng, L Wang, S Feng, and J Fan. Hierarchical clustering algorithm for categorical data using a probabilistic rough set model. *Knowledge-Based Systems*, 65:60–71, 2014. doi:https://doi.org/10.1016/j.knosys.2014.04.008.

[80] Jiye Liang, Xingwang Zhao, Deyu Li, Fuyuan Cao, and Chuangyin Dang. Determining the number of clusters using information entropy for mixed data. *Pattern Recognition*, 45(6):2251–2265, 2012. doi:10.1016/j.patcog.2011.12.017.

[81] Dekang Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, 1998. URL: http://dl.acm.org/citation.cfm?id=645527.657297.

[82] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

[83] Nhat-Vinh Lu, Trong-Nhan Vuong, and Duy-Tai Dinh. Combining correlation-based feature and machine learning for sensory evaluation of saigon beer. *International Journal of Knowledge and Systems Science (IJKSS)*, 11(2):71–85, 2020. doi:10.4018/IJKSS.2020040104.

[84] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, University of California Press*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[85] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. doi:10.1017/CBO9780511809071.

[86] Gabor Meli. The datgen Dataset Generator, 1999. URL: http://www.datasetgenerator.com.

[87] Boris Mirkin. *Clustering: a data recovery approach*. Chapman and Hall/CRC, 2016.

[88] Michael K Ng, Mark Junjie Li, Joshua Zhexue Huang, and Zengyou He. On the impact of dissimilarity measure in k-modes clustering algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):503–507, 2007. doi:10.1109/TPAMI.2007.53.

[89] Huu Hiep Nguyen. Clustering Categorical Data Using Community Detection Techniques. *Computational intelligence and neuroscience*, 2017. doi:10.1155/2017/8986360.

[90] Thanh-Phu Nguyen, Duy-Tai Dinh, and Van-Nam Huynh. A new context-based clustering framework for categorical data. In *Pacific Rim International Conference on Artificial Intelligence*, pages 697–709. Springer, 2018. doi:10.1007/978-3-319-97304-3_53.

[91] Thu-Hien Thi Nguyen, Duy-Tai Dinh, Songsak Sriboonchitta, and Van-Nam Huynh. A method for k-means-like clustering of categorical data. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–11, 2019. doi:10.1007/s12652-019-01445-5.

[92] Thu-Hien Thi Nguyen and Van-Nam Huynh. A k-means-like algorithm for clustering categorical data using an information theoretic-based dissimilarity measure. In *FoIKS*, pages 115–130. Springer, 2016. doi:10.1007/978-3-319-30024-5_7.

[93] Darius Pfitzner, Richard Leibbrandt, and David Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(3):361–394, 2009. doi:10.1007/s10115-008-0150-6.

[94] Minh Nguyen Quang, Tai Dinh, Ut Huynh, and Bac Le. Mhhusp: An integrated algorithm for mining and hiding high utility sequential patterns. In *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*, pages 13–18. IEEE, 2016. doi:10.1109/KSE.2016.7758022.

[95] Minh Nguyen Quang, Ut Huynh, Tai Dinh, Nghia Hoai Le, and Bac Le. An approach to decrease execution time and difference for hiding high utility sequential patterns. In *International Symposium on Integrated Uncertainty in Knowledge Modelling and Decision Making*, pages 435–446. Springer, 2016. doi:10.1007/978-3-319-49046-5_37.

[96] Md Geaur Rahman and Md Zahidul Islam. Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. *Knowledge-Based Systems*, 53:51–65, 2013. doi:10.1016/j.knosys.2013.08.023.

[97] Chandan K Reddy and Bhanukiran Vinzamuri. A survey of partitional and hierarchical clustering algorithms. In *Data Clustering: Algorithms and Applications*, pages 87–110. Chapman and Hall/CRC, 2013.

[98] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[99] Ohn Mar San, Van-Nam Huynh, and Yoshiteru Nakamori. An alternative extension of the k-means algorithm for clustering categorical data. *International Journal of Applied Mathematics and Computer Science*, 14(2):241–247, 2004.

[100] Amir Masoud Sefidian and Negin Daneshpour. Missing value imputation using a novel grey based fuzzy c-means, mutual information based feature selection, and regression model. *Expert Systems with Applications*, 115:68–94, 2019. doi:10.1016/j.eswa.2018.07.057.

[101] Shokri Z Selim and Mohamed A Ismail. K-means-type algorithms: A generalized convergence theorem and characterization of local optimality. *IEEE Transactions on pattern analysis and machine intelligence*, pages 81–87, 1984. doi:10.1109/TPAMI.1984.4767478.

[102] Ali Seyed Shirkhorshidi, Saeed Aghabozorgi, Teh Ying Wah, and Tutut Herawan. Big data clustering: a review. In *International conference on computational science and its applications*, pages 707–720. Springer, 2014. doi:10.1007/978-3-319-09156-3_49.

[103] Esther-Lydia Silva-Ramírez, Rafael Pino-Mejías, and Manuel López-Coello. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, 29:65–74, 2015. doi:10.1016/j.asoc.2014.09.052.

[104] Esther-Lydia Silva-Ramírez, Rafael Pino-Mejías, Manuel López-Coello, and María-Dolores Cubiles-de-la Vega. Missing value imputation on missing completely at random data using multilayer perceptrons. *Neural Networks*, 24(1):121–129, 2011. doi:10.1016/j.neunet.2010.09.008.

[105] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3(3):583–617, 2003. doi:10.1162/153244303321897735.

[106] Haojun Sun, Rongbo Chen, Shulin Jin, and Yong Qin. A hierarchical clustering for categorical data based on holo-entropy. In *2015 12th Web Information System*

and Application Conference (WISA), pages 269–274. IEEE, 2015. doi:10.1109/WISA.2015.18.

[107] Pang-ning Tan and Vipin Kumar. Interestingness Measures for Association Patters: A Perspective. In *KDD Workshop on Postprocessing in Machine Learning and Data Mining*, pages 1–9, 2000.

[108] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right objective measure for association analysis. *Information Systems*, 29(4):293–313, 2004. doi:10.1016/S0306-4379(03)00072-3.

[109] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. URL: https://books.google.co.jp/books?id=KZQ0jgEACAAJ.

[110] Pang-Ning Tan, Michael Steinbach, Vipin Kumar, et al. Cluster analysis: basic concepts and algorithms. *Introduction to data mining*, 8:487–568, 2006.

[111] Fei Tang and Hemant Ishwaran. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(6):363–377, 2017. doi:10.1002/sam.11348.

[112] DM Titterington. A comparative study of kernel-based density estimates for categorical data. *Technometrics*, 22(2):259–268, 1980.

[113] Ramazan Ünlü and Petros Xanthopoulos. Estimating the number of clusters in a dataset via consensus clustering. *Expert Systems with Applications*, 125:33–39, 2019. doi:https://doi.org/10.1016/j.eswa.2019.01.074.

[114] Jake VanderPlas. *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, 2016. URL: https://books.google.co.jp/books?id=xYmNDQAAQBAJ.

[115] Pantelis Vlachos. StatLib—Datasets Archive, 2005. URL: http://lib.stat.cmu.edu/datasets.

[116] Sławomir T Wierzchoń and Mieczysław Kłopotek. *Modern algorithms of cluster analysis*. Springer, 2018.

[117] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008. doi:10.1007/s10115-007-0114-2.

[118] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015. doi:10.1007/s40745-015-0040-1.

[119] Rui Xu and Donald C Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16:645–678, 2005. doi:10.1109/TNN.2005.845141.

[120] Mohamed Zaït and Hammou Messatfa. A comparative study of clustering methods. *Future Generation Computer Systems*, 13(2-3):149–159, 1997. doi:10.1016/S0167-739X(97)00018-6.

[121] Shichao Zhang. Shell-neighbor method and its application in missing data imputation. *Applied Intelligence*, 35(1):123–133, 2011. doi:10.1007/s10489-009-0207-6.

[122] Shichao Zhang, Jilian Zhang, Xiaofeng Zhu, Yongsong Qin, and Chengqi Zhang. Missing Value Imputation Based on Data Clustering. In *Transactions on Computational Science I*, pages 128–138. Springer, 2008. doi:10.1007/978-3-540-79299-4_7.

[123] Z. Zhang. Missing data imputation: Focusing on single imputation. *Annals of Translational Medicine*, 4(1), 2016. doi:10.3978/j.issn.2305-5839.2015.12.38.