Doctoral Dissertation

Incorporating the Locality Sensitive Hashing Technique into Clustering Algorithms for Massive Categorical and Mixed Datasets

Nguyen Mau Toan

Supervisor: Professor Huynh Van Nam

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

December 2021

Abstract

Cluster analysis has been an object of research since the 1980s for finding the natural groups of objects in the data so that similar objects stay within the same clusters while different objects stay in different clusters. It is undoubted that cluster analysis is important for a wide range of scientific and industrial processes such as data mining, computer vision, signal processing, and census research. k-means-like algorithms are the most used unsupervised machine learning techniques for handling such problems of cluster analysis. In the context of a k-means-like algorithm, a proper structure for representing the clusters and an appropriate distance measure for measuring the distance between objects and clusters must be accordingly defined. In general, a k-means-like algorithm seeks for the optimal clusters that can minimize the total distance from all objects to their nearest clusters, which makes the cluster representation and distance measure become very important to achieve such clustering target. With different types of data, the formulas of cluster representations must differ accordingly. For instance, *centroid* is specified for numerical data while *mode* and *representative* are specified for categorical data. For the data with both numerical and categorical attributes, the *prototype* structure can be effectively applied by hybridizing the *centroid* and *representative*.

On one hand, a basis k-means algorithm is a local optimization technique that can easily return a locally optimal solution. To achieve a better or the global solution, the clustering algorithm must seek the solutions several times with different initial states. For this reason, a “good” initial state is very important to achieve the global optimum. In this research, we first aim to propose a new scheme to use a dimension reduction technique so-called Locality-Sensitivity Hashing (LSH) to predict the “good” initial state of the cluster so that the global optimal can be potentially obtained. The empirical experiment using real and synthetic datasets showed that our proposed method LSH-k-representatives and LSH-k-prototypes not only can outperform other related works in terms of clustering accuracy but also have the best consistency for clustering categorical and mixed data, respectively. However, the proposed LSH-based cluster prediction requires extra processes in order to create the LSH hash table, which makes the proposed method not ready for handling big data yet.

On the other hand, the complexity of a k-means-like algorithm varies linearly with the volume of data while the volume of data is exploded day by day. Thus, it is essential to reduce the complexity of k-means-like algorithms so that they become capable of processing big and real data. Dimension reduction and data sample are the most used techniques that can approximate the clustering procedures. However, these approaches change the nature of the data instead of changing the algorithm to make it more appropriate. This dissertation also fills such shortcoming by proposing a new heuristic approach for approximately reducing the complexity of a typical k-means-like algorithm. In detail, the proposed method can avoid the potential unnecessary distance computations from objects to cluster representations in each iteration. Consequently, after applying our proposed method into LSH-k-representatives, the incorporated algorithm can process up to 2 to 32 times faster than its own original version with comparable clustering accuracy.

Moreover, we also extend a kernel-based representation of so-called LSH-k-prototypes to make it capable of fuzzy clustering of categorical data. The LSH-based cluster prediction technique is then extended to estimate the fuzzy clusters of categorical data. Eventually, the proposed fuzzy clustering algorithm so-called LSHFk-centers can outrun other state-of-the-art fuzzy clustering approaches.

Keywords: Cluster analysis, fuzzy cluster analysis, categorical data, mixed data, LSH, cluster initialization, big data.

Acknowledgments

With sincere appreciation, I would like to express my deep gratitude to my supervisor, Prof. HUYNH VAN-NAM for helping me when I was having the most difficult time in my life.

I would like to thank Prof. YASUSHI INOGUCHI of Japan Advanced Institute of Science and Technology (JAIST) for guiding and imparting knowledge to me for five years and also giving me the opportunity to work as a teaching assistant.

I would like to acknowledge my second supervisor in my master's course, Prof. MINEO KANEKO, for giving me a lot of valuable suggestions to improve my research. I would also like to express my gratitude to my second supervisor in my Ph.D. course, Prof. KIYOFUMI TANAKA for his helpful advice and suggestions. I also would like to thank Prof. MINEO KANEKO, Prof. KIYOFUMI TANAKA again, Prof. NGUYEN LE MINH, and Prof. KATSUHIRO HONDA (from Osaka Prefecture University) for taking the valuable time to become my examination committees.

I would like to thank the Japan Advanced Institute of Science and Technology (JAIST) for facilitating an excellent learning environment, which encouraged and supported me during my master's course. Special thanks to the Institute for giving me the special opportunity to conduct my minor research at the Technical University of Ostrava.

I would like to thank my spouse, NGO NU DIEU KHUE for always being with me and assisting me with the challenges of study and research.

Finally, I would like to thank my family, lab-mates, and friends for their encouragement and support during the process of writing this thesis.

List of Abbreviations

AMI	Adjusted Mutual Information
ARI	Adjusted Rand Index
DILCA	Distance Learning Dissimilarity for Categorical Data
LSH	Locality-sensitive Hashing
NMI	Normalized Mutual Information

List of Symbols

$\boldsymbol{x}_i$	Data object
D	Number of attributes/dimensions of data
N	Number of objects in the dataset
A_d	The d-th attribute $(1 \leq d \leq D)$
$\mathbb{R}$	The real numeric domain
$\mathbb{A}_d$	The categorical domain for the d-th attribute (only if $A_d \in \mathcal{A}$)
$\mathcal{A}$	The set of categorical attributes
$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$	Data set of N objects
l	Number of hash functions
Dis(,)	Dissimilarity metric
$H = \{h_1, h_2, ..., h_l\}$	Set of l hash functions
$\mathbb{H} = \{B_0, B_1, ..., B_{2^l-1}\}$	Set of 2^l buckets in the hash table $\mathbb{H}$
ω	Weight of categorical attributes
k	The number of clusters
λ_j	Weight of normal distribution of categorical values in the j-th cluster
$C, M, R, \hat{P}, P$	Different notations of sets of cluster representations
$U = [u_{ij}]_{N\times k}$	Membership matrix
O()	Objective function

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Research background

Unsupervised learning plays an important role in knowledge discovery and data mining, especially in the era of big data because labeling data is both costly and time-consuming. Besides that, unsupervised learning has recently received great attention from the research community of machine learning [1]. The popular unsupervised learning problems are data clustering [2], k-Nearest Neighbor (k-nn) search [3], anomaly detection [4], Principal Component Analysis (PCA) [5], Independent Component Analysis (ICA) [6], Singular Value Decomposition (SVD) [7]. Among these problems, data clustering or cluster analysis appears in most of the fields such as computer science, computer vision, medical science, economics, and census studies [8]. Not only being used in those fields, but cluster analysis also have the positive effects on other machine learning techniques such as k-nn classification [9], product quantization [10], outlier detection [11], and collaborative filtering [12]. Technically, the cluster analysis problem is the problem to find the auto-created groups in the data such that the similar objects belong to the same or near clusters while the dissimilar objects belong to different clusters [1, 8, 13]. In the context of cluster analysis, the data clustering problem can be separated into two sub-problems which have two different approaches:

- Flat clustering: Flat clustering aims to divide the data into disjoint groups/clusters. Technically, crisp clustering algorithms optimize an objective function that minimizes the differences of objects within-cluster and maximizes the differences of objects among different clusters [2, 8].
- Hierarchical clustering: Hierarchical clustering creates a partitioning structure to describe the clusters in the data. In this nested structure, a cluster can be a subset of another cluster, which helps in better bonding expression between objects and clusters [8].

Although hierarchical clustering gives better clustering results than flat clustering, it requires a massive computation for calculating the distance between objects and clusters [8], which makes the hierarchical clustering techniques unable to handle datasets with massive volume. Thus, flat clustering becomes much more feasible for big data.

In the context of flat clustering, there are a number of clustering algorithm classes can be effectively utilized such as k-means-like algorithms [2, 14, 15, 16, 17, 18], Expectation–Maximization (EM) algorithms [19, 20], Self-Organizing Map (SOM) approaches [21, 22], Genetic algorithms (GAs) [23, 24, 25]. Among these algorithm classes, k-means-like algorithms stand out above all others because of its simplicity and ease of use. Figure 1.1 presents the hierarchical relations of the cluster analysis problem with other machine learning problems.

In general, a k-means-like algorithm seeks to divide the data into k finite separated clusters and each cluster is represented by a representation, the principle of k-means-like algorithms can be simply described in three steps:

1. *Initialization step*: Select the k clusters randomly or predictably, then assign the objects into k initial clusters.
2. *Iteration step*:
   - Representation updating: The representations are calculated to show the compactness of all clusters.
   - Membership updating: The labels of objects are re-assigned to their nearest cluster representations.
3. *Evaluation step*: Repeat the *Iteration step* until the presentations are converged or maximum number of the maximum number of iterations is reached.

It is obvious that k-means-like algorithms are local optimization algorithms, which do not guarantee to find the global optimum in a single epoch. Observantly, most k-means-like algorithms have multiple epochs and return the most optimal outcome. On the other hand, k-means-like algorithms are the low-complexity algorithms. However, big data clustering remains a significant challenge due to the heterogeneity and massive size of data that principally limit the application of clustering in real-world scenarios.

Another problem of cluster analysis on big data is that the data can have a high number of attributes of mixed numerical and categorical types. For instance, to describe information of a person, the basis attributes such as *Age*, *Weight*, and *Occupation* can be used. Among these attributes, *Age* and *Weight* can be represented by numerical values and it is very easy to state the differences between two persons in terms of *Age* and *Weight*. In contrast, there is no standard measure to state the differences in the context of *Occupation*. For example, we cannot tell the similarity between *Student*,

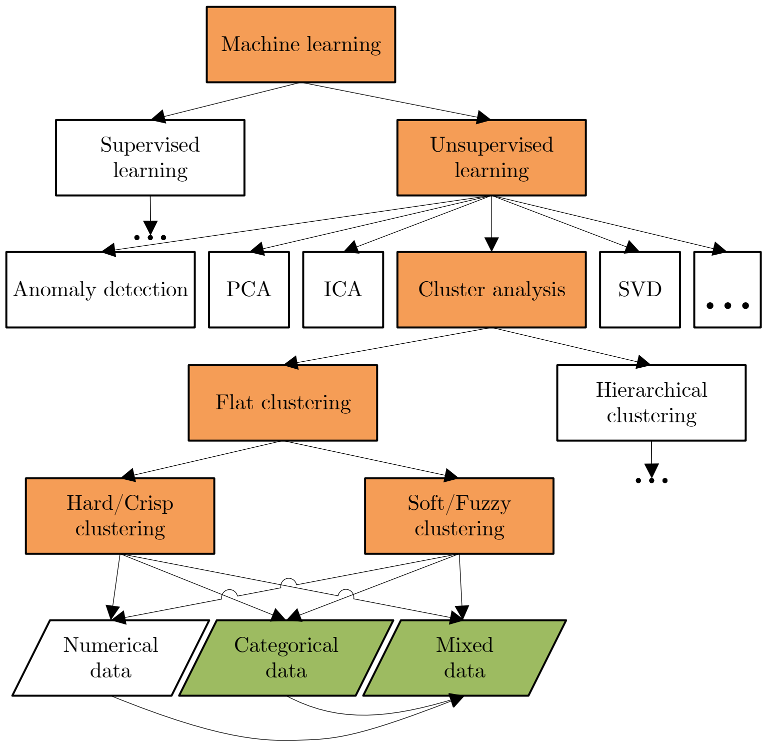

Figure 1.1: Segment of cluster analysis problem

Worker, and *Technician* without any certain contexts. This kind of data is called categorical data. To deal with categorical data, a special dissimilarity measure must be defined to state the degree of difference between two categorical values. In addition, the distance measure is decisive with the outcome of k-means-like algorithms. Thus, to conduct the data clustering of categorical data using k-means-like algorithm, a suitable dissimilarity measure and cluster representation must be defined.

1.2 Motivations

Several attempts have been made to handle the clustering problem of mixed data type and all these approaches define the hybrid representations for clusters. These approaches heavily emphasize clustering accuracy with high complexity, which makes them are not capable to handle big mixed datasets [26, 27, 28, 29, 30]. Besides that, some approximate techniques namely dimension reduction approaches [31, 32] or data sampling approaches [33, 34] can increase the clustering speed for big data. However, these approaches are used for numerical or categorical data only. These facts prove the shortcoming of clustering algorithms for big mixed data. Therefore, in this research, we motivate to propose a new framework for handling the big mixed data which can be applied to various clustering algorithms. In detail, our motivation can be separated into two different aspects:

- To propose a novel technique to predict the potential cluster centers for k-means-like algorithms. A better prediction can lead to a higher chance to catch the global optimal solution. Thus, the number of required epochs can be reduced significantly for getting the global optimum.
- To propose a new approximate method to reduce the computation of clustering iteration. Clustering iterations are the heaviest task in k-means-like algorithms. This task includes the distance computation from objects to cluster representations. Reducing these distance computations can sharply decrease the complexity of the clustering algorithm.

1.3 Research objectives

Based on our motivations, the targets for the research are set as follows:

- Conducting comprehensive surveys as well as analyze the advantages and disadvantages of existing clustering methods for mixed data, some specific topics include:
  - Similarity measurements: Overlap, Eskin, IOF, OF, Lin, Goodall, DILCA.
  - Crisp clustering algorithms: k-means, k-modes, k-representatives, k-centers.
  - Fuzzy clustering algorithms: Fuzzy-k-means, Fuzzy-k-modes, Fuzzy SBC, SGA, MOGA, NSGA-FMC.
- Developing new scalable and effective clustering methods for big data based on the integration of the Locality-sensitive Hashing (LSH) technique and k-means-like clustering paradigm. In detail, three sub-objectives can be listed to complete this framework, specifically:
  - Analyzing the effectiveness of similarity measurements for categorical data for LSH classification. The most effective measure can then be selected for hashing the categorical data.
  - Proposing an applicable hashing function for categorical data in order to group the similar items into semi-groups. For hashing

    the numerical data, the regular locality-sensitive hash function for metric space can be adopted.
  - Introducing a novel approach to approximate the clustering iterations.
  - Concatenating the clustering models for numerical data and for categorical data into one.
- Performing a comprehensive series of experimental studies to validate the proposed clustering methods and illustrate their practical applicability.

1.4 Contributions

The contributions of this dissertation are summarized as follows:

- This dissertation first conducts the comprehensive literature review on categorical and mixed data cluster analysis. Different approaches for k-means-like algorithms are grouped by corresponding categories of improving representation, improving cluster initialization, and improving clustering iteration. The dissertation then states the outstanding problems of categorical and mixed data, especially for big categorical and mixed data. Therefore, the representation for categorical or mixed data are well-defined but the cluster initialization and cluster iteration of the k-means algorithms have not yet been explored much in the problem of big data processing.
- To the best of our knowledge, this dissertation is the first attempt to incorporate the LSH technique into k-means-like algorithms for achieving better cluster initialization states. Our LSH-based cluster prediction method can predict not only the crisp clusters but also the fuzzy clusters of categorical and mixed data.
- Our kernel-based representation for fuzzy cluster analysis of categorical data is the first attempt to combine the uniform distributions and observed distributions of unique categorical values, which can lead to a better representation for clusters with having a large difference in frequencies of unique categorical values.
- Last but not least, we proposed a simple heuristic approach to reduce the complexity of typical k-means-like algorithms. Our approximate method has a relatively high accuracy while not requires many computations like dimension reduction or sampling methods.

1.5 Research map

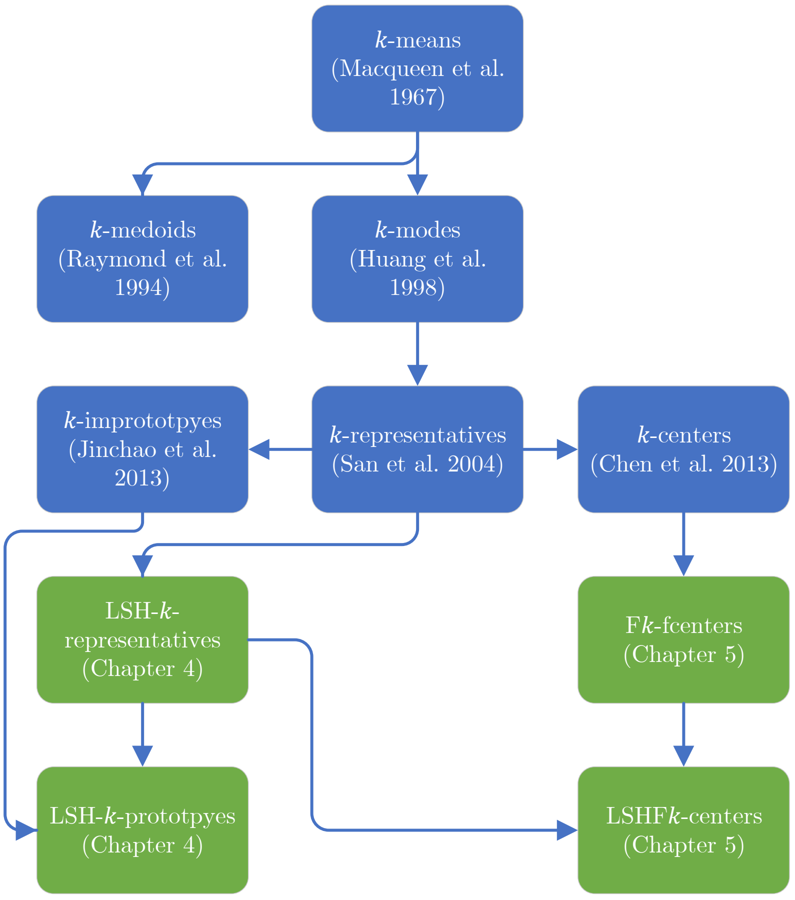
k-means (Macqueen et al. 1967)
k-medoids (Raymond et al. 1994)
k-modes (Huang et al. 1998)
k-imprototpyes (Jinchao et al. 2013)
k-representatives (San et al. 2004)
k-centers (Chen et al. 2013)
LSH-*k*-representatives (Chapter 4)
F*k*-fcenters (Chapter 5)
LSH-*k*-prototpyes (Chapter 4)
LSHF*k*-centers (Chapter 5)

Figure 1.2: Research map of this dissertation

Figure 1.2 shows the relationships between the proposed methods in this dissertation with each other and with the previous approaches.

1.6 Organization of the thesis

The dissertation contains 6 chapters and the contents of chapters can be briefly described as follows:

Chapter 1 first introduces the research background, then states the motivations, research objectives, and contributions of the dissertation.

Chapter 2 describes the details of the cluster analysis problem of big data with categorical and mixed data. The basis k-means-like algorithms and conventional methods are also presented in Chapter 2.

Chapter 3 lists the background techniques that are utilized in our proposed methods. Several related works are also introduced in this chapter.

Chapter 4 describes the principles of LSH-k-representatives and LSH-k-prototypes, which are the main proposals of this dissertation for handling the clustering problem of big categorical and mixed data. The experiments, self-analyzing results, comparison results, and conclusion of those approaches are also shown in Chapter 4.

Chapter 5 first extends the kernel-based representation for fuzzy cluster analysis of categorical data (Fk-centers). The LSH-based cluster prediction approach is then applied to this extension (LSHFk-centers). The experiments, self-analyzing results, comparison results, and conclusion of those approaches are also shown in Chapter 5.

Chapter 6 yields the conclusion, limitations, and future works for the whole dissertation.

Chapter 2

Problem statements and literature review

This chapter presents the mathematics definition of the clustering problem. The achievements and challenges of crisp clustering are also listed.

2.1 Data clustering problem statements

This section aims to state the fundamental definition of general crisp clustering problem which has been used in many prior studies [13, 15, 16, 17, 18, 20, 27, 31, 33, 35, 36, 37, 38, 39, 40, 41].

Let $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ be the set of N objects to be clustered into k disjoint clusters, so that N is the number of objects in the dataset and k is the number of target clusters. Each item $\boldsymbol{x}_i$ $(1 \leq i \leq N)$ is a vector in D-dimensional space called attribute, $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iD}]$. Each dimension at the d-th position can be numerical value $(x_{id} \in \mathbb{R})$ or categorical value $(x_{id} \in \mathbb{A}_d)$, where $\mathbb{A}_d$ $(|\mathbb{A}_d| > 1)$ is the set of all unique categorical values in the d-th attribute. In other words, let A_d be the d-th attribute, if A_d is a categorical attribute then $\mathbb{A}_d$ is the unique domain of attribute A_d. In contrast, if A_d is a numerical attribute then $\mathbb{R}$ is the domain of attribute A_d. Thus, the dataset X can be represented by an $N \times D$ matrix with the columns corresponding to features and the rows corresponding to objects. It is convenient to mark all the categorical attributes for further computation. Then, we denote $\mathcal{A}$ as the set of all categorical attributes. Table 2.1 shows a small portion of UCI Adult dataset, where A_1, A_2, and A_4 are the numerical attributes and the rest attributes are the categorical attributes. Thus, $A_3, A_5, A_6, A_7 \in \mathcal{A}$ and $A_1, A_2, A_4 \notin \mathcal{A}$.

It is also quite important to distinguish each attribute is categorical or numerical because it can make us misunderstand the nature of data. Usually, the values of a categorical domain are stored as text strings such as colors, shapes, and job titles. In this case, we can easily detect the categorical attributes automatically. However, it becomes more complex when the categorical attributes are stored as numeric values (numerous systems store categorical values as numeric values because it can save memory space), in this case, we need to detect the categorical attribute manually.

Table 2.1: A sample of mixed dataset (UCI Adult dataset)

	A_1	A_2	A_3	A_4	A_5	A_6	A_7
$\boldsymbol{x}_1$	39	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family
$\boldsymbol{x}_2$	50	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband
$\boldsymbol{x}_3$	38	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family
$\boldsymbol{x}_4$	53	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband
$\boldsymbol{x}_5$	28	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife
$\boldsymbol{x}_6$	37	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife
$\boldsymbol{x}_7$	49	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family
$\boldsymbol{x}_8$	52	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband
$\boldsymbol{x}_9$	31	45781	Masters	14	Never-married	Prof-specialty	Not-in-family
$\boldsymbol{x}_{10}$	42	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband

2.1.1 Formulation of clusters

Let $\mathcal{G} = \{G_1, \ldots, G_k\}$ be the set of k clusters in dataset X, which must satisfy following conditions:

$$\begin{cases} G_j \neq \emptyset, \quad 1 \leq j \leq k \\ \bigcup_{j=1}^{k} G_j = X \\ G_j \cap G_{j'} = \emptyset, \text{ for } j, j' = 1, \ldots, k \text{ and } j \neq j' \end{cases} \tag{2.1}$$

To make it easy for the calculation, we can use a matrix of membership to store the membership labels of objects to clusters. Let $U = [u_{ij} | 1 \leq i \leq N, 1 \leq j \leq k]$ be such membership matrix, u_{ij} $(1 \leq i \leq N, 1 \leq j \leq k)$ shows the degree of membership of object $\boldsymbol{x}_i$ to the j-th cluster. In the context of crisp clustering, membership degree presents whether an object belongs to a cluster or not:

$$u_{ij} = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \in G_j \\ 0, & \text{otherwise} \end{cases}, \quad 1 \leq i \leq N, 1 \leq j \leq k \tag{2.2}$$

Besides that, an object can be a member of a cluster only, we can infer that:

$$\sum_{j=1}^{k} u_{ij} = 1, \quad 1 \leq i \leq N \tag{2.3}$$

Furthermore, to make sure there is no empty cluster as the first condition in (2.1):

$$\sum_{i=1}^{N} u_{ij} \geq 1, \quad 1 \leq j \leq k \tag{2.4}$$

To generalize, we denote $U = [u_{ij}]_{N\times k}$ as the membership matrix that can present membership statuses of all objects.

2.1.2 Formulation of dissimilarity measures

Similarity/dissimilarity measures play an extremely important role in forming clustering, which defines whether two objects be included in a cluster or not. Therefore, a "good" similarity/dissimilarity measure must be able to reflect the nature of the data. However, several standard measures can work well in most cases. Because similarity measure and dissimilarity measure are symmetrical, we choose to use dissimilarity measure term for this study. A typical dissimilarity measures can be remark as: Given two data objects $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$, $\mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ gives the distance/dissimilarity between them. Because the characteristics of numerical and categorical objects are partially different, we categorize the dissimilarity of numeric, categorical, and mixed data as follows.

2.1.2.1 Dissimilarity measures for numerical objects

There are many basic metrics that can calculate the distances of two numerical multivariate objects such as Euclidean distance, Manhattan distance, Minkowski distance, Cosine distance. In this study, the Euclidean distance is utilized:

$$\mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{i'd})^2} \tag{2.5}$$

2.1.2.2 Dissimilarity measures for categorical objects

Several approaches have been proposed to measure the distance between two categorical objects [42, 43, 44, 45], the most popular one is the overlap function:

$$\mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sum_{d=1}^{D} \delta(x_{id}, x_{i'd}) \tag{2.6}$$

where

$$\delta(x_{id}, x_{i'd}) = \begin{cases} 0 & \text{if } x_{id} = x_{i'd}, \\ 1 & \text{if } x_{id} \neq x_{i'd}. \end{cases}$$

2.1.2.3 Dissimilarity measures for mixed objects

For the objects containing attributes of both numerical and categorical values, the hybrid weighted measure can be applied:

$$\mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \omega \sum_{\substack{d=1 \\ A_d \in \mathcal{A}}}^{D} \delta(x_{id}, x_{i'd}) + (1-\omega) \sqrt{\sum_{\substack{d=1 \\ A_d \notin \mathcal{A}}}^{D} (x_{id} - x_{i'd})^2} \tag{2.7}$$

where $\omega (0 \leq \omega \leq 1)$ is the parameter that weight the importance of categorical attributes. This weighting procedure is necessary because the distance of categorical values are usually smaller than distance of numerical values.

Because the distributions of data are different for different datasets, to achieve the optimal value of ω, it is required to analyze comprehensively the distributions of the data for categorical and numerical attributes on the same datasets. A fast and effective method is to use the value ranges and/or the number of attributes for each type to estimate the value of ω. However, in this research, we do not focus on such optimization; therefore, 0.5 is selected as the default value of ω for all datasets and compared methods in the experiments. We also conduct a experiment to show the contribution of parameter ω to our proposed method (See Figure 4.15).

2.1.3 Objectives of data clustering

As we mentioned, the target of data analysis is to maximize similarities of objects on the same cluster and dissimilarities of objects on different clusters, which means minimizing the Within-Group Sum of Square (WGSS) defined by (2.8) and maximizing the Between-Group Sum of Square (BGSS) defined by (2.9) as follows:

$$\mathrm{WGSS}(G_1, \ldots, G_k) = \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in G_j} \sum_{\boldsymbol{x}_{j'} \in G_j} \mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{x}_{j'}) \tag{2.8}$$

$$\mathrm{BGSS}(G_1, \ldots, G_k) = \sum_{i=1}^{k} \sum_{\substack{j=1 \\ j \neq i}}^{k} \sum_{\substack{\boldsymbol{x}_{i'} \in G_i \\ \boldsymbol{x}_{j'} \in G_j}} \mathrm{Dis}(\boldsymbol{x}_{i'}, \boldsymbol{x}_{j'}) \tag{2.9}$$

In the next section, we will introduce the principle of the k-means-like algorithm, which is the most popular class of algorithms that can optimize the above WGSS and BGSS equations.

2.2 Classes of clustering algorithms

There are numerous methods that can handle the clustering problem, these methods can be categorized into the following headings:

2.2.1 k-means-like algorithms

All k-means-like algorithms are based on the principle of k-means algorithm [46], which are designed to find the optimal centers of all clusters to minimize an objective function, it also partly represents equations (2.8) and (2.9) simultaneity.

Advantages: Relatively simple to implement, can predict the starting positions of centroids and guarantees convergence.

Disadvantages: Being dependent on initial initialization stage, clustering data of varying sizes and density.

Because k-means-like algorithms are the basis of our proposed method. We will introduce the principle of k-means-like algorithms in section 2.3, this section only introduces the advantages and disadvantages of them.

2.2.2 Expectation–Maximization (EM) algorithms

EM algorithms use a statistical model to formulate the clustering problem and find maximum likelihood estimates of parameters to fix the given data. The principle of EM includes two main stages: Expectation step creates a function for the expectation from the EM parameters and maximization step estimates the parameters for maximizing the expected log-likelihood found in the expectation step [47, 48]. The EM algorithm has the similar steps with k-means clustering algorithm:

1. Initialize the statistical parameter $\theta = \{p_1, \mu_1, \sigma_1, \dots, p_k, \mu_k, \sigma_k\}$, where p_j, μ_j, σ_j $(1 \leq j \leq k)$ are the probability, mean, and standard deviation of the j-th cluster, respectively.
2. Expectation: Compute the responsibilities for each object to each cluster:

$$\gamma_{ij} = \frac{p_j \mathrm{PDF}(\boldsymbol{x}_i; \mu_j, \sigma_j)}{\sum_{j'=1}^{k} p_{j'} \mathrm{PDF}(\boldsymbol{x}_i; \mu_{j'}, \sigma_{j'})}, \quad 1 \leq i \leq N, 1 \leq j \leq k \tag{2.10}$$

where PDF(;,) is the probability distribution function.

3. Maximization: Update the estimating parameters θ:

$$p_j = \frac{1}{N}\sum_{i=1}^{N}\gamma_{ij} \quad \mu_j = \frac{\sum_{i=1}^{N}\gamma_{ij}\boldsymbol{x}_i}{\sum_{i=1}^{N}\gamma_{ij}} \quad \sigma_j = \frac{1-\sum_{i=1}^{N}\gamma_{ij}(\boldsymbol{x}_i-\mu_j)^2}{\sum_{i=1}^{N}\gamma_{ij}}, \quad 1 \le j \le k \tag{2.11}$$

4. Repeat steps 2 and 3 until the estimating parameters converge to a local optimum.

The crisp cluster for each object is then calculated by getting the distribution with higher probability for this object.

Several of extensions of EM algorithm can be found in [49, 50, 51], some extensions for clustering categorical data can be found in [52, 53].

Advantages: Good for fitting mixture distributions.

Disadvantages: Bad for high dimensional data because the standard deviations are treated equally for different dimensions.

2.2.3 Genetic Algorithms (GAs)

The clustering problem also can be solved by a genetic algorithm. In details, the clusters are defined by the centers $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k\}$, where $\boldsymbol{c}_j$ $(1 \le j \le k)$ is the center or representation of the j-th cluster. GA algorithms for data clustering problem can have single or multiple objective functions, but cluster minimizing inner-cluster distance (equation (2.12)) and maximizing cluster separation (equation (2.13)) are used quite often:

$$\mathrm{O}(C,X) = \sum_{j=1}^{k}\sum_{i=1}^{N} u_{ij}\mathrm{Dis}(\boldsymbol{c}_j, \boldsymbol{x}_i) \tag{2.12}$$

$$\mathrm{Sep}(C,X) = \sum_{j=1}^{k}\sum_{\substack{j'=1 \\ j'\neq j}}^{k} \mathrm{Dis}(\boldsymbol{c}_j, \boldsymbol{c}_{j'}) \tag{2.13}$$

At this time, the distance functions from object to cluster representation and from cluster representation to cluster representation must be defined. However, we discuss these functions later because the formulas distance functions are depended on the principles of cluster representation. The basic GA for clustering problem use the presentations as the chromosome, the stages of GA can be listed as:

- *Initialization*: Creating multiple chromosomes randomly. A chromosome is a sample of cluster representation set C.

- *Population evaluation*: Evaluate the chromosomes via equation (2.12).
- *Evaluation*: If the number of evolution is reached or the optimum chromosome is obtained then stop the algorithm
- *Selection*: Select the best chromosomes.
- *Crossover*: Create more chromosomes by merging the selected ones.
- *Mutation*: Create more mutated chromosomes by randomly. Then go to *Population evaluation* step.

The advantages and disadvantages of GAs can be seen as:

Advantages: GA supports multi-objective optimization, GA works well on mixed discrete/continuous problem.

Disadvantages: High complexity for high number of clusters k. Difficult to define the suitable crossover and mutation functions.

2.2.4 Self-Organizing Map (SOM) algorithms

SOM is a type of Artificial Neural Network (ANN) that is trained for optimizing clustering problem. SOM is useful for visualization by creating the low-dimensional views for showing the densities of high-dimensional data. Technically, the specific ANN model is designed to map the objects from original space into a low-dimensional space (typically two-dimensional). SOM algorithm can be described as follows:

- Randomly creating a map, a map is usually a two-dimensional mesh.
- For each object $\boldsymbol{x}_i$:
  - Calculating the distance from $\boldsymbol{x}_i$ to all nodes in the map and finding the closest node for the object $\boldsymbol{x}_i$.
  - Updating the nodes and the neighbor nodes in the map by pulling them closer to $\boldsymbol{x}_i$.

Advantages: Easy to visualize, support to quantize the data.
Disadvantages: Difficult to distinguish the clusters.

2.3 k-means-like algorithms

2.3.1 k-means algorithm

k-means algorithm [46] is one of the most popular clustering algorithms for numerical data. Let X be the dataset of N numerical data $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, such that $\boldsymbol{x}_i = \{x_{i1}, \ldots, x_{iD}\} \in \mathbb{R}^D (1 \leq i \leq N)$. Remark the target of k-means algorithm is also to divide the dataset X into k separated groups $\mathcal{G} = \{G_1, \ldots, G_k\}$ which satisfy equation (2.1) and minimize the total distances of object to the representation of the cluster that it belongs to:

$$\text{Minimize:} \quad \mathrm{O}(U, C) = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij} \mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{c}_j) \tag{2.14}$$

In the context of the original k-means algorithm, the representation $\boldsymbol{c}_j$ is the center of gravity of the j-th cluster, so-called centroid of the j-th cluster. Thus, $\boldsymbol{c}_j$ is also a vector of the same space with the dataset, therefore, $\boldsymbol{c}_j = [c_{j1}, \ldots, c_{jD}] \in \mathbb{R}^D$. Therefore, the distance function in equation (2.5) can be utilized to measure the distance from data object to cluster representation. Note that, with more complex representation, the distance function between data object and cluster representation must be redefined accordingly.

To update/calculate the representations of clusters, we use the means of all objects in the clusters:

$$\boldsymbol{c}_j = \Big(\sum_{i=1}^{N} u_{ij} \boldsymbol{x}_i\Big) \Big/ \Big(\sum_{i=1}^{N} u_{ij}\Big), \quad 1 \leq j \leq k \tag{2.15}$$

To assign the degree of membership to an object, the closest cluster that is the nearest to this object is selected:

$$u_{ij} = \begin{cases} 1, & j = \arg\min_{j'} \mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{c}_{j'}) \\ 0, & \text{Others} \end{cases} \tag{2.16}$$

The detail of k-means algorithm is shown in Algorithm 2.1, maximum iteration number $t_{\max}$ and convergence threshold ϵ are the parameters controlling the convergence of the algorithm.

2.3.2 k-means extensions for categorical data clustering

k-means has numerous extensions to handle different challenges [2, 8], many of them are focusing on handling the categorical data. Namely, k-modes is the first k-means-like algorithm for categorical data and k-representatives is the extension that has the categorical cluster representation with highest efficiency [15, 40].

2.3.2.1 k-modes algorithm

Because of the characteristics of categorical data, we cannot compute the mean of all objects in a cluster as in equation (2.15). Huang et al. (1998)

Algorithm 2.1 k-means algorithm

```
Input: Dataset $X$, target number of cluster $k$, maximum iteration number $t_{\max}$, convergence threshold $\epsilon$.
Output: Local optimal membership matrix $U$, cluster representations $C$.

1: Initialization step: Randomly assign the values for the membership matrix $U$, such that the requirements in equations (2.2), (2.3), and (2.4) are satisfied.
2: $\text{tmp}_0 \leftarrow \infty$
3: Iteration steps:
4: for $t \leftarrow 1$; $t \leq t_{\max}$; $t \leftarrow t + 1$ do
5:    Calculate/re-calculate the representations $C$ via equation (2.15).
6:    Re-calculate membership matrix $U$ via equation (2.16).
7:    Compute convergence value $\text{tmp}_t \leftarrow \text{O}(U, C)$ via equation (2.14).
8:    if $|\text{tmp}_t - \text{tmp}_{t-1}| < \epsilon$ then
9:       Break the loop.
10:   end if
11: end for
12: return $U, C$
```

proposed to use a *mode* for a cluster instead of *mean* as the representation for that cluster [40, 54].

$M = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_k]$ denotes the set of k *modes* to represent k corresponding categorical clusters. A *mode* $\boldsymbol{m}_j = [m_{j1}, \ldots, m_{jD}]$ is a vector of categorical values that has the highest frequencies in all attributes:

$$m_{jd} = \arg\max_{v_d \in \mathbb{A}_d} \text{Frequency}_{G_j}(v_d), \quad 1 \leq j \leq k, 1 \leq d \leq D \tag{2.17}$$

with $\text{Frequency}_{G_j}(v_d)$ returns the frequency of categorical value v_d appearing in the j-th cluster.

Observably, *modes* are the categorical objects, thus, we can use the overlap function to measure the distance between *mode* and object. The objective function can then be adjusted accordingly:

$$\text{Minimize:} \quad \text{O}(U, M) = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij} \text{Dis}(\boldsymbol{x}_i, \boldsymbol{m}_j) \tag{2.18}$$

The steps of k-modes are identical to those of k-means algorithm.

2.3.2.2 k-representatives algorithm

Using *mode* is a very simple approach, but it cannot fully describe the compactness of clusters because only the highest categorical value with the highest frequency is stored and the rest are ignored. San et al. (2004) and Kim et al. (2004) introduced to store the frequencies of all categorical values to describe the representations of clusters [15, 55]. This type of representation is called *representative* and it does not miss any useful compactness information. In particular, $R = [\boldsymbol{r}_1, \ldots, \boldsymbol{r}_k]$ denotes the set of k *representatives* for k clusters, and a *representative* can be described as:

$$\boldsymbol{r}_j = [\boldsymbol{r}_{j1}, \ldots, \boldsymbol{r}_{jD}] \tag{2.19}$$

where

$$\boldsymbol{r}_{jd} = [(v_d, \Pr_{G_j}(v_d))|v_d \in A_d], \quad 1 \leq d \leq D \tag{2.20}$$

with $\Pr_{G_j}(v_d)$ being the relative ratio of value v_d taken by all objects in cluster G_j, namely

$$\Pr_{G_j}(v_d) = \frac{\text{Frequency}_{G_j}(v_d)}{|G_j|}, \quad \forall v_d \in \mathbb{A}_d, \text{ and } 1 \leq d \leq D \tag{2.21}$$

It is interesting to note here that a *representative* is a vector of probabilities of all categorical values in all attributes, so $\sum_{v_d \in \mathbb{A}_d} \Pr_{G_j}(v_d) = 1$ for $1 \leq j \leq k$ and $1 \leq d \leq D$. For instance, if a categorical value has high frequency of appearance in a cluster, then the corresponding probability of that categorical value is also high.

In fact, the overlap function cannot be applied for measuring the distance between object and *representative* because *representatives* are not categorical objects. However, if we consider that an object is also a *representative* with only singular active categorical value in each attribute, the similarity distance can then be defined as:

$$\text{Dis}(\boldsymbol{x}_i, \boldsymbol{r}_j) = \sum_{d=1}^{D} \sum_{v_d \in \mathbb{A}_d} \Pr_{G_j}(v_d)\delta(x_{id}, v_d) \tag{2.22}$$

After being combined with the overlap function (see equation(2.6)), we can shorten the equation (2.22) as:

$$\text{Dis}(\boldsymbol{x}_i, \boldsymbol{r}_j) = \sum_{d=1}^{D} \sum_{\substack{v \in A_d \\ v \neq x_{id}}} \Pr_{G_j}(v) = D - \sum_{d=1}^{D} \Pr_{G_j}(x_{id}) \tag{2.23}$$

The objective function can then be adjusted accordingly:

$$\text{Minimize:} \quad \mathrm{O}(U, R) = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij} \mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{r}_j) \tag{2.24}$$

The steps of k-representatives are identical to those of k-means and k-modes algorithms.

2.3.3 k-means extensions for mixed numerical and categorical data

Corresponding to k-modes and k-representatives for clustering categorical data, there are two popular techniques that can handle mixed numerical and categorical data, namely k-prototypes [56] and improved k-prototypes [28].

2.3.3.1 k-prototypes algorithm

Huang et al. (1997) introduced k-prototypes as the extension of k-modes algorithm to handle mixed data clustering [56]. A *prototype* is a mixed structure of categorical values and numerical values following the properties of corresponding attributes. Let $\hat{P} = [\hat{\boldsymbol{p}}_1, \ldots, \hat{\boldsymbol{p}}_k]$ be the set of k *prototypes* for k clusters, where a *prototype* can be formulated as:

$$\hat{\boldsymbol{p}}_j = [\hat{p}_1, \ldots, \hat{p}_D], \quad 1 \leq j \leq k \tag{2.25}$$

and

$$\hat{p}_{jd} \in \begin{cases} \mathbb{R}, & \text{if } A_d \notin \mathcal{A} \\ \mathbb{A}_d, & \text{otherwise} \end{cases}, \quad 1 \leq d \leq D, 1 \leq j \leq k \tag{2.26}$$

Note that $\mathcal{A}$ is the set of all categorical attributes. It is clear that a *prototype* is a mixed vector of numerical and categorical values so that the distance function for two mixed vectors (equation (2.7)) can be utilized to calculate the distance from object to *prototype*.

Similarly, the algorithm of k-prototypes is equivalent to other k-means-like algorithms with the objective function:

$$\text{Minimize:} \quad \mathrm{O}(U, \hat{P}) = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij} \mathrm{Dis}(\boldsymbol{x}_i, \hat{\boldsymbol{p}}_j) \tag{2.27}$$

2.3.3.2 Improved k-prototypes algorithm

If k-representative is the extension of k-modes in the context of categorical data clustering, then the improved k-prototypes is the extension of k-prototype in terms of mixed data clustering [28]. Let $P = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k]$ be the set of k *improved prototypes* for k clusters, an *improved prototype* can be formulated as:

$$\boldsymbol{p}_j = [p_1, \ldots, p_D], \quad 1 \leq j \leq k \tag{2.28}$$

where

$$p_{jd} \in \begin{cases} \mathbb{R}, & \text{if } A_d \notin \mathcal{A} \\ \{[(v_d, \mathrm{Pr}_{G_j}(v_d)) | v_d \in \mathbb{A}_d]\}, & \text{otherwise} \end{cases}, \quad 1 \leq d \leq D, 1 \leq j \leq k \tag{2.29}$$

With such complicated representation, the distance function from object to *improved prototype* (equation (2.30)) must be adjusted accordingly:

$$\mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{p}_j) = \omega \sum_{\substack{d=1 \\ A_d \in \mathcal{A}}}^{D} \left(|\mathcal{A}| - \sum_{d=1}^{D} \mathrm{Pr}_{G_j}(x_{id}) \right) + (1-\omega) \sqrt{\sum_{\substack{d=1 \\ A_d \notin \mathcal{A}}}^{D} (x_{id} - p_{jd})^2} \tag{2.30}$$

Similarly, the algorithm of improved k-prototypes is equivalent to other k-means-like algorithms with the objective function:

$$\text{Minimize:} \quad \mathrm{O}(U, P) = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij} \mathrm{Dis}(\boldsymbol{x}_i, \boldsymbol{p}_j) \tag{2.31}$$

2.4 Optimization techniques for k-means-like algorithm

In the previous sections, we listed several classes of algorithms that can increase the clustering effectiveness for numeric, categorical, and mixed data. Although the complex algorithms can increase the accuracy, they also require a lot of computations. In this section, we will list several approaches to optimize the clustering speed for clustering algorithms.

2.4.1 Clustering initialization optimization approaches

Several attempts have been made to increase the goodness of initialization for k-means-like algorithm for numerical data [57]. Arthur et al. first stated the important of good initialization to a k-means-like algorithm and proposed an initialization optimization method so-call k-means++ [37]. In particular, k-means++ selects the first centroid ($\boldsymbol{c}_1$) randomly from the objects in the dataset. The next centroids ($\boldsymbol{c}_2, \ldots, \boldsymbol{c}_k$) are sequentially selected from the remaining objects with a probability being inverse with the distance from that object to the selected centroids. Therefore, the initial clusters are likely to be spread across all regions of the data and the place with high data density are likely to exist only one initial cluster. However, k-means++ is slowed down when working with dataset of high number of objects with large number of cluster k. This can be explained in a simple way that to examine the capacity of becoming a cluster centroid of an object, it is needed to compute the distance of that object to all the known initial cluster centroids. Furthermore, the randomly selected data objects for centroid prediction in multiple times also take a lot of computations.

Bahmani et al. (2012) introduced an initialization optimization method called k-means|| and claimed that their algorithm can fulfill the drawback of $k$means++, especially in case of clustering large datasets [36]. In fact, k-means|| can run faster than k-means++ by selecting multiple objects in an iteration, and thereby the number of iterations is dramatically reduced. Technically, k-means|| runs $\mathcal{O}(\log\ N)$ iterators and in each iterator select k objects to predict the centroids. From numerical results, k-means|| outperforms k-means++ in terms of clustering speed. However, both k-means++ and k-means|| are proposed for numerical data, so they may not work well for the categorical data clustering.

In terms of initialization optimization techniques for categorical data clustering, Huang et al. introduced k-modes along with two naive initialization methods [40]. The first approach selects k object randomly to set them *modes*. The second approach sorts the categorical values in descending order and sequentially set them to the values of *modes*. However, this approach only considers the separation of categorical values and the correlations among them in different attributes are eliminated. In addition, the second approach of Huang et al. also has the drawback when the number of categorical values is less than k, which makes empty cluster(s). This problem can be treated as selecting other random object(s) to the empty cluster(s).

Cao et al. (2009) proposed to use density of categorical data based on the same number of categorical values appearing on other objects to predict the potential *modes*. The rest objects are then merged to the selected high density *modes* using MaxMin algorithm [58]. Bai et al. (2012) continued the work that can avoid selecting the boundary object as *modes* [59]. Similar to k-means++ algorithm, the distances between high density objects are also used to maximize the distances between potential *modes*.

In 2013, Khan et al. proposed to conduct the multiple clustering on the selected attributes and select the semi-clusters in each attribute. The *modes* are then resembled from these semi-clusters, and they call them *deterministic modes* [60]. In 2016, Jiang et al. continued to propose a technique to eliminate the outliers among all permutation *deterministic modes* [61]. Particularly, the *mode* candidates are justified by measuring the distances and the entropy among them so that the weak *mode* candidates are removed.

Recently, Dinh et al. proposed to find the object sets with maximal frequencies to estimate the cluster initial clusters [29]. After removing the overlapped object sets, the cluster representations are then formed by using a kernel based technique which inherits from *representative*.

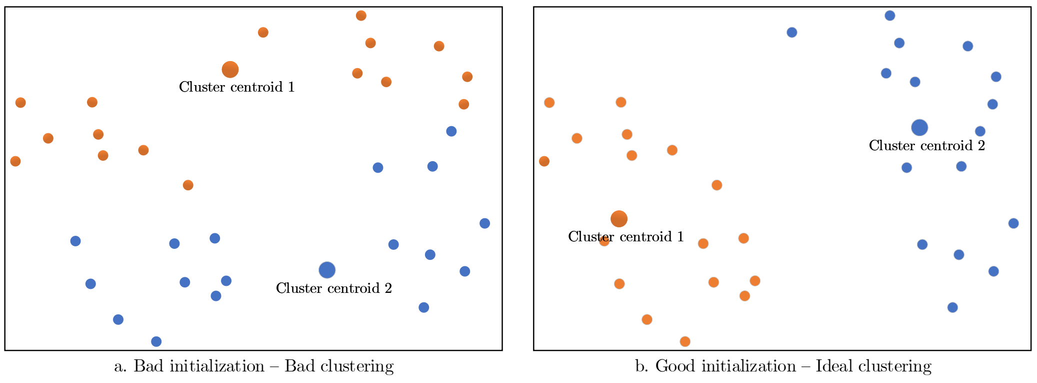
Cluster centroid 1
Cluster centroid 2
a. Bad initialization – Bad clustering
Cluster centroid 2
Cluster centroid 1
b. Good initialization – Ideal clustering

Figure 2.1: Example of bad and good clustering

Figure 2.1 shows the example of how differently the good initialization states and the bad ones affect the clustering results. Figure 2.1.b shows the ideal clustering for the illustrated dataset. To achieve this result, we should predict the location of cluster representations relatively accurately with the ground-truth cluster representations at the beginning of the k-means-like algorithm.

2.4.2 Clustering iteration optimization

2.4.2.1 Sampling methods

The target of sampling is to select the subset of the original data that behaves like the original one so that the outcome of conducting data clustering in the subset can be used for the original dataset [62, 63]. Whereas random sampling is the most naive approach which selects the objects randomly and it expects to capture the same data distributions from the original dataset [62].

Random sampling approach has the critical drawback when working with imbalance data because the small clusters may be abandoned. The density-biased sampling approaches was mainly proposed to handle this problem [64, 65]. Palmer et al. introduced to divide the dataset space into grid with non-overlapping cell, the densities of cells can then be calculated by counting the frequencies of objects on the cells [66]. This approach can overcome the noise data and avoid eliminating small clusters. Nanopoulos et al. used trees instead of mesh to structure the dataset [67].

2.4.2.2 Dimension reduction methods

The target of dimension reduction methods is to reduce the computation cost of distance calculations which are the most often used in clustering [68]. The popular dimension reduction methods for clustering can be listed as:

Principal Component Analysis (PCA): PCA is the most used linear dimension reduction method which is based on the covariance matrix of the dataset. Particularly, PCA uses a few orthogonal linear combinations with the largest variance to reduce the dimensions of the data [69, 70]. The variances in the new coordinate system are in descending order so that the target number of new dimensions can be easily selected. In addition, it is very convenient that the data on two coordinate axes can be easily exchanged based on a transformation matrix.

Factor analysis (FA): FA is also a linear method, which seeks to maximize the second-order data summaries [68]. In details, FA defines a transformation matrix called common *intelligence* factor. The scores of all objects are then calculated using this transformation matrix. Similar to PCA method, the variance of each transformation is justified for selection.

Projection Pursuit (PP): PP is also a linear method but uses higher-order transformation instead of second-order transformation [71]. Shannon entropy or variance scoring can be utilized for examining the effectiveness of the projections. To reduce the complexity of evaluating the second-order transformations, we can apply several heuristic techniques that can guarantee the profitable factor of transformations.

Independent Component Analysis (ICA): ICA is a higher-order transformation method that seeks for the non-orthogonal linear projections [68]. The projections decompose the multivariate vector into independent non-Gaussian distributions. Therefore, ICA can show the advantage when working with data consisting of mixture distributions.

Random Projections (RP): RP is the low-complexity approach to project the data from high-dimensional space to a low-dimensional space using random projection matrix [72]. It is interesting to note here is that the selected projection matrices does not require any evaluation and there are a lot of heuristic techniques to select the beneficial projection function.

2.5 Summary

In this chapter, we formulated the crisp clustering problem for mixed data of numerical and categorical data. The inputs, outputs, objectives has been clearly defined.

Different approaches that can optimize the clustering algorithms were also introduced, namely *clustering representation optimization*, *cluster initialization optimization*, and *cluster iteration optimization*. Their advantages and disadvantages are summarized in Table 2.2.

Table 2.2: Advantages and disadvantages of clustering algorithm optimization approaches

Technique	**Advantages**	**Disadvantages**
Cluster representation optimization	• Can improve the inner-cluster compactness and outer-cluster separation. • Can adapt with different types of data.	• Require more computation for complex representations.
Cluster initialization optimization	• Can help to find the global optimum.	• Require extra computations. • Do not affect the clustering complexity.
Cluster iteration optimization	• Reduce the complexity of the clustering algorithms.	• Accuracy is reduced because of data approximating.

It is clear that *clustering representation optimization* and *cluster initialization optimization* slow down the algorithm because of complex structure and extra computation. Only *clustering iteration optimization* can affect the complexity of the clustering algorithm.

Chapter 3

Research background and related works

3.1 Introduction

In the previous section, we introduced the problem state and different ways that can optimize the crisp clustering algorithm. This section continues with the specific techniques that are applied in our system followed by the related works of this research.

3.2 Locality-sensitive hashing (LSH)

LSH is a dimension reduction method that hashes similar input objects into the same buckets with high probability. Because the number of buckets is much smaller than the universal values of original objects, the dimensions of new space are reduced dramatically when using the buckets to represent the similar objects [73].

Dataset X denotes the set of N objects $\boldsymbol{x}_i$ $(1 \leq i \leq N)$ which are represented as points in a D-dimensional space $\mathbb{R}^D$. LSH uses a family of hash functions to reduce the dimensions of the dataset; each hash value of the new dimensional space forms a bucket that contains every data point that has that hash value. By having the same hash value, the data points in the same bucket have more connections than the data points from separate buckets. The distance of buckets can also be compared using the distance calculation of hash values on metric space. Therefore, LSH is suitable for the Approximate Nearest Neighbor Search (ANNS) problem when the system searches the results in particular selected buckets [73, 74].

We denote l with $l \leq D$ as the number of subsets or the number of binary hash functions in the family of hash functions. Using l hash functions, LSH generates l subsets $I_1, I_2, ..I_l$ for every input point $\boldsymbol{x}_i$ $(1 \leq i \leq N)$. There are l random projection functions that transform input vector $\boldsymbol{x}_i \in \mathbb{R}^D$ to a new vector $\boldsymbol{x}'_i \in \mathbb{B}^l$, where $\mathbb{B}$ is the binary space, $\mathbb{B} = \{0, 1\}$. The value of $\boldsymbol{x}'_i$ will indicate the hash value or the bucket ID for point $\boldsymbol{x}_i$. Let $h_j(\boldsymbol{x})$ $(1 \leq j \leq l)$ be the jth projection function, then $x'_{ij} = h_j(\boldsymbol{x}_i) \in \mathbb{B}$ and $h_j : \mathbb{R}^D \to \mathbb{B}$. Let $H = [h_1, \dots, h_l]$ be the ordered collection of l hash functions so that $H : \mathbb{R}^D \to \mathbb{B}^l$. The new vector $[x'_{i1}, \dots, x'_{il}]$ will be the hash value of input vector $\boldsymbol{x}_i$.

To have quick access to the original objects on new hashed-space, we can create the hash table $\mathcal{H}$ that holds all the hash values and the corresponding data points in the dataset [73, 74]. With hash table $\mathcal{H}$, we can index all the data points to the hash values. For the ANNS problem, we must first calculate the hash value h for query $\boldsymbol{q}$ by using the same set of random projection functions g when building hash table $\mathcal{H}$. The hash value h will index the set of data points B_h in X; these data points will have the same hash value h.

Call p the threshold for evaluating the distance between an ANN candidate and the query point. The ANNS candidates can be evaluated by using the following formula:

$$\text{CheckKNN}(\boldsymbol{x}_i) = \begin{cases} true, & \text{if Distance}(\boldsymbol{q}, \boldsymbol{x}_i) \leq p \\ false, & \text{otherwise} \end{cases} |\boldsymbol{x}_i \in B_v \qquad (3.1)$$

where v is the hash value of the query $\boldsymbol{q}$. The function CheckKNN($\boldsymbol{x}_i$) is used for evaluating the ANNS outputs of the query $\boldsymbol{q}$ for all the data/vectors $\boldsymbol{x}_i$ in the bucket B_h. Depending on the limited number of ANN k, we can stop the comparing steps when the number of returned neighbors reaches k.

For searching multiple buckets, LSH needs to search within several buckets that have similar IDs to v. Call P_2 the threshold for evaluating the similarity between two buckets with $P_2 < p$. The distance between chosen buckets B_h and B'_h must be less than P_2. In this case, we have more chances to obtain the ANN for query $\boldsymbol{q}$ from other buckets [73, 74].

Figure 3.1 shows an example of using LSH when applying using three hyperplanes in two-dimensional space as the three hash function. Each hash function divides the space into two sub-spaces. For example, if $H = [h_0]$ is the family of hash functions then $H(\boldsymbol{x}_1) = H(\boldsymbol{x}_2) = H(\boldsymbol{x}_6) = H(\boldsymbol{x}_7) = H(\boldsymbol{x}_8) = H(\boldsymbol{x}_9) = 0$ and $H(\boldsymbol{x}_3) = H(\boldsymbol{x}_4) = H(\boldsymbol{x}_5) = 1$. In other hand, if $H = [h_0, h_1, h_2]$ then $H(\boldsymbol{x}_1) = H(\boldsymbol{x}_2) = [000]$, $H(\boldsymbol{x}_3) = [100]$, $H(\boldsymbol{x}_4) = H(\boldsymbol{x}_5) = [110]$, $H(\boldsymbol{x}_6) = H(\boldsymbol{x}_7) = H(\boldsymbol{x}_8) = [010]$, and $H(\boldsymbol{x}_9) = [011]$.

The return value $\boldsymbol{x}'_i$ is called the hash value of binary variables $\boldsymbol{x}_i$. This value is converted into a decimal number for indexing the bucket in the hash table $\mathcal{H}$. In the LSH system, that hash value is used for generating the hash table to look up the existing hash table. Continuing the example in Figure

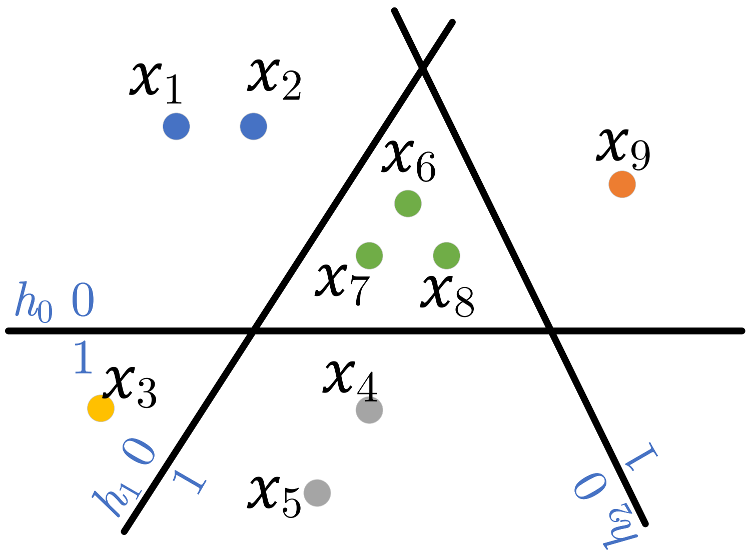

Figure 3.1: Example of applying LSH hash function using random projection hyperplane

3.1, the structure of the hash table is:

$$\begin{aligned}\mathcal{H} =&[\{[000] : \{\boldsymbol{x}_1, \boldsymbol{x}_2\}\}, \{[001] : \emptyset\}, \{[010] : \{\boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8\}\},\\ &\{[011] : \{\boldsymbol{x}_9\}\}, \{[100] : \{\boldsymbol{x}_3\}\}, \{[101] : \emptyset\},\\ &\{[110] : \{\boldsymbol{x}_4, \boldsymbol{x}_5\}\}, \{[111] : \emptyset\}]\end{aligned} \tag{3.2}$$

Note that the sensitive-locality hash value [101] does not exist in this example because h_0 and h_1 lose the value domain of h_2, which makes the order of hash function become important.

In practice, the objects having the same hash values are put in the same set called a bucket. Each bucket uses the key as the hash value of all objects as the bucket key.

Figure 3.2 presents how the LSH technique supports the ANN search problem. The binary "Hash Functions H" is the key component in the stages of hash table building and searching. First of all, the binary hash values of all the points in "Dataset X" need to be calculated. The "Hash Table $\mathcal{H}$" includes multiple buffers that are indexed by different binary/decimal values. The items in "Dataset X" are sorted by their binary hash values in the "Hash Table $\mathcal{H}$". When the query $\boldsymbol{q}$ approaches the system, the hash value for $\boldsymbol{q}$ needs to be calculated using the same "Hash Functions H". In this example, if the hash value of $\boldsymbol{q}$ is [00..00], then the identical bucket of $\boldsymbol{q}$ is the blue buffer. After comparing the distances of $\boldsymbol{q}$ with all the items on that bucket,

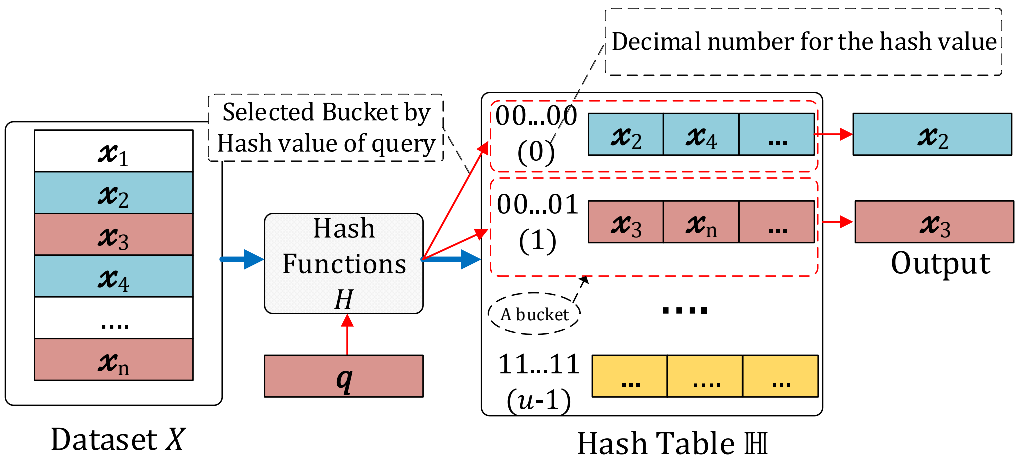

Figure 3.2: An illustration of locality-sensitive hashing

we can return the item $\boldsymbol{x}_2$ as the ANN of $\boldsymbol{q}$. In another case, if the hash value of $\boldsymbol{q}$ is [00..01], $\boldsymbol{x}_3$ will be the ANN of $\boldsymbol{q}$ [73, 74, 75].

The LSH technique can also groups similar objects into the same bucket (or cluster), just like a typical clustering technique does; However, the outcome of the LSH technique cannot fully satisfy the general objectives of the cluster analysis problem because the locality-sensitive factor cannot fully reflect the dissimilarity factor. Besides that, the number of the clusters cannot be determined as the user desire. This makes the optimization techniques like k-means-like algorithms become much more suitable for cluster analysis problems. In this study, LSH is used indirectly for grouping objects at the stage of predicting the initial clusters.

3.3 Dissimilarity measures for categorical data

As we mentioned, the dissimilarity measures for categorical data are very important because they determine how similar and different unique categorical values are. The dissimilarity measures can be divided into two types: context-free measures and context-sensitive measures [45].

3.3.1 Context-free measures

Context-free measures are the ones that consider the attributes to work completely independently. Therefore, the dissimilarity values are decided within a single attribute's domain for a pair of categorical values.

The overlap function is the most basic context-free measure, which uses the number of mismatch values in all attributes as the dissimilarity value (see equation (2.6)). More advanced measures that can be taken into account the frequency of categorical values in the domain, namely Goodall, Smirnov, Anderberg, Lin, Burnaby, Gambryan, OF, IOF, and Eskin [45].

3.3.2 Context-sensitive measures

For the purpose of applying LSH to handle categorical data, context-free measures are not enough because the sensitive factor is not considered in these measures. In contrast, context-sensitive measures can be applied the sensitive factor for different categorical values in different attribute domains.

Some context-sensitive measures that can be listed are Association based similarity measure [44], Distance Learning Dissimilarity for Categorical Data (DILCA) [43], and CBDL [76]. Among them, DILCA is the simplest context-sensitive measure and widely applied [45].

In this research, we select DILCA to generate the dissimilarity matrices for data attributes.

3.4 Maximum cut

In the context of graph theory, the maximum cut or Max-Cut problem is the problem that finds the cut of edges in the graph, such that the total sum of weights on the cut is the highest overall possible cuts that can divide the nodes of the graph into two complementary sets [77]. Stoer-Wagner is the simplest algorithm that can solve the maximum cut problem [77].

The maximum is widely used to find two disjoint complementary sets in the graph [77]. In this study, we use the maximum cuts to estimate the suitable thresholds for the LSH hash functions.

3.5 Related works

3.5.1 Locality-Sensitive Hashing Technique for Categorical Data

Because of the disadvantages of the categorical data, few pieces of research have applying the sensitive hashing technique to the categorical data. Indeed, the locality-sensitive hash value is the consequence of the similarity measure, but the similarity of categorical values is not as clear as the similarity of numerical values [78, 79].

Lee et al. (2016) proposed the use of hierarchical clustering to partition the categorical domains to build the locality-sensitive hash functions. The data-driven measures such as IOF, OF, or Goodall were used to group the clusters in the hierarchical clustering [79]. The proposed method of Lee et al. can be described as:

- Computing the statistical measures of each value on each attribute based on their frequencies.
- Calculating the semantic distance matrices for all pairwise combinations of categorical values on all attributes.
- For each attribute, conducting the hierarchical clustering using their semantic distance matrix.
- Defining the LSH hash functions for all attributes and combine them to form a family of hash functions.
- Generating the hash table with the obtained family of hash functions.

This method was applied to the kNN classification problem with the United States (US) census (1990) dataset, which comprises 68 categorical attributes for 2,458,285 records. As a result, the method can archive a recall of 86.1% with around a third of the number of computations compared with the conventional methods. However, Lee et al.'s method does not consider the information gain of the hash functions. Although each hash function can give different outcomes based on its entropy, the hash functions were randomly selected among all feasible hash functions [80].

In addition, while the data-driven dissimilarity measures gave the differences between categorical values, the cross frequencies of values were ignored. That makes the prediction of clusters inaccurate in the initial process.

3.5.2 MinHash k-Modes (MH-k-Modes)

McConville et al. (2016) proposed the so-called MH-k-Modes as an extension of k-Modes algorithm to accelerate the clustering process for big categorical data [32]. Specifically, MH-k-Modes uses the advantages of the MinHash technique in hashing the document data to build the relevant family of hash functions [32, 81].

A family of MinHash hash functions of l functions in the dataset X is defined as $\mathrm{SIG}_X = \{s_1, \ldots, s_l\}$. For example, if the document with t rows $\{\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t\}$ is given, the MinHash value for the document by the ith hash function h_i can be obtained as $s_i = \min(h_i(\boldsymbol{r}_j) | j = 1, \ldots, t)$. The Cyclic Redundancy Check (CRC) is a common locality-sensitive hash function for word/text data. In detail, the rows of document are segmented into shingles of 3 words, the authors then applied CRC for the shingles to obtain the 32 or 64 bit crisp numbers as the hash values [82].

Similar to other LSH methods, MH-*k*-Modes builds a hash table for the dataset to store the hash values and the neighbors of all document objects. In practice, MH-*k*-Modes uses the neighborhood of an object to find a shortlist of candidate clusters for that object. The overview of the MH-*k*-Modes algorithm can be described as:

- Selecting *k* initial *modes* from the dataset.
- Assigning membership degrees for all objects based on their similarities with the *modes*.
- Applying the MinHash hash function to each object, building the hash table for the dataset, storing the indexes of objects with the same hash values into the same bucket, and creating the cluster reference for each object based on its bucket.
- From the computed shortlist of each object, finding the nearest *mode* in the shortlist instead of all lists of all *modes* and updating the memberships.
- Recomputing the values of all *modes* according to the original *k*-Modes algorithm.
- Repeating the last two steps until no object changes their memberships or the limited number of iterations is reached.

An obvious drawback of Mcconville et al.'s method is the unstable size of the item's shortlist. Because the bucket size cannot be determined [73], several buckets can have long shortlists while others can have very few candidates on their shortlists. For instance, if there exists a bucket that contains only one item, the shortlist of that item has only one cluster. Besides, because the hash value of that item does not change during the clustering process, the item belongs to a cluster only.

In our research, we were inspired by the idea of creating shortlists of items. However, to solve the problem of Mcconville et al.'s method, we use the dynamic shortlists of clusters instead of static shortlists of items. Furthermore, we use the information from LSH hash values to generate the initial clusters.

3.6 Summary

In this chapter, we remark on the research backgrounds that are used in our research. In detail, the principles of LSH is applied to predict the semi-clusters while context-sensitive measures and maximum-cut solution help to achieve the good LSH hash functions. Moreover, the principle, advantages, and disadvantages of the related research with this research are also briefly shown in this chapter.

In the next chapter, we describe in detail the mechanism of applying these research backgrounds into our framework.

Chapter 4

LSH-k-prototypes: a framework for fast clustering of big mixed data

4.1 Introduction

In this chapter, we describe the principle of our proposed method in detail. Particularly, we take advantage of the well-defined cluster representation of k-representatives and improve the k-representatives algorithm in two different directions: *cluster initialization optimization* and *iteration optimization.* The proposed algorithms are called LSH-k-prototypes and LSH-k-representatives in this thesis, the main stages of them are shown in Figure 4.1 [83].

4.2 LSH-k-prototypes initialization

4.2.1 Creating dissimilarity matrices for categorical data

This section only introduces the dissimilarity calculation of categorical data because dissimilarity calculating of numerical data is simple and there are no issues to argue with. Dissimilarity calculating is extremely important for categorical data because it supports to describe the correlations of categorical values. Observatory, the overlap dissimilarity function (see equation (2.6)) is too simple to take advantage of. In addition, the values of dissimilarity matrix of overlap measure are symmetric, which does not support to find the groups of similar categorical values in a categorical domain. Thus, we choose to apply DILCA measurement to explore the dissimilarity between categorical values. In the context of DILCA, the dissimilarity between two categorical values in the d-th dimension can be calculated as:

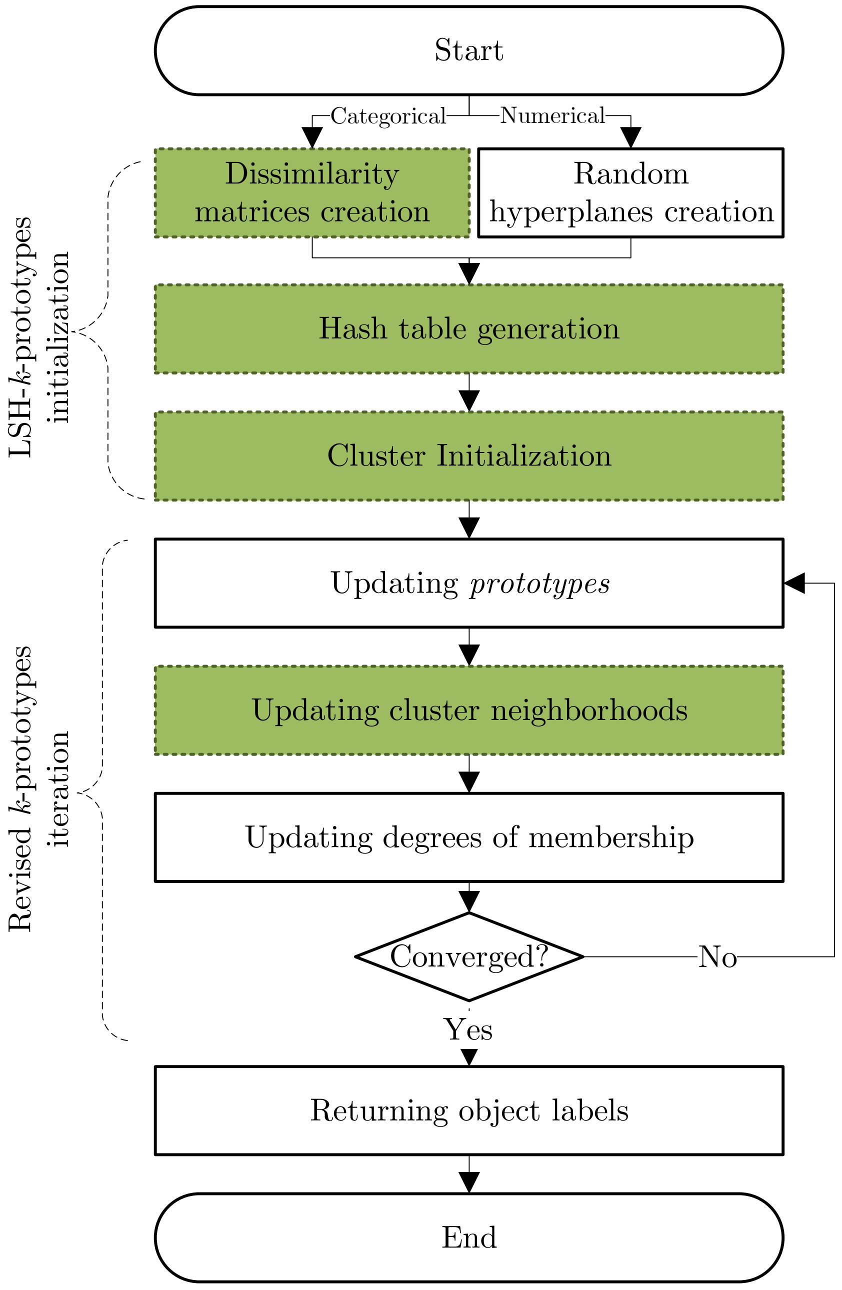
Start
Categorical
Numerical
Dissimilarity matrices creation
Random hyperplanes creation
Hash table generation
Cluster Initialization
LSH-*k*-prototypes initialization
Updating *prototypes*
Updating cluster neighborhoods
Updating degrees of membership
Converged?
No
Yes
Revised *k*-prototypes iteration
Returning object labels
End

Figure 4.1: Overview of the proposed clustering framework

$$\delta(a_{di}, a_{dj}) = \sqrt{\frac{\sum_{\mathbb{A}_{d'} \in \text{context}(\mathbb{A}_d)} \sum_{a_{d'} \in \mathbb{A}_{d'}} (\Pr(a_{di}|a_{d'}) - \Pr(a_{dj}|a_{d'}))^2}{\sum_{\mathbb{A}_{d'} \in \text{context}(\mathbb{A}_d)} |\mathbb{A}_{d'}|}} \tag{4.1}$$

where context(A_d) is the context set of attributes $A_{d'}$ that contains highly correlated attributes to A_d, and Pr(|) is the formula for the conditional probability.

Table 4.1: Semantic dissimilarity matrix for attribute A_d with $N_d = |\mathbb{A}_d|$ categorical values

	a_{d1}	a_{d2}	a_{d3}	...	a_{dN_d}
a_{d1}	0	$\delta(a_{d1}, a_{d2})$	$\delta(a_{d1}, a_{d3})$	...	$\delta(a_{d1}, a_{dN_d})$
a_{d2}	$\delta(a_{d2}, a_{d1})$	0	$\delta(a_{d2}, a_{d3})$	...	$\delta(a_{d2}, a_{dN_d})$
a_{d3}	$\delta(a_{d3}, a_{d1})$	$\delta(a_{d3}, a_{d2})$	0	...	$\delta(a_{d3}, a_{dN_d})$
...	...	...	...	...	...
a_{dN_d}	$\delta(a_{dN_d}, a_{d1})$	$\delta(a_{dN_d}, a_{d2})$	$\delta(a_{dN_d}, a_{d3})$	...	0

It is true that the tasks of measuring the dissimilarity between categorical objects can be repeated for the same pairs of categorical values during the clustering process. In addition, the complexity of DILCA measurement is also high. Therefore, it is better to pre-calculate the dissimilarity values for each pair of categorical values in each domain and store them to reduce the processing time. Table 4.1 shows an example of a dissimilarity matrix for categorical attribute A_d, each cell of the matrix is filled by using equation (4.1).

4.2.2 LSH hash table generation

In this step, we ought to create the LSH hash table for the dataset. However, the dissimilarity measurements of numerical and categorical data are different, we will introduce the LSH hash functions for numerical and categorical data separately.

4.2.2.1 LSH functions for numerical data

Depending on the characteristic of the numerical data, different kinds of LSH functions can show the different effectiveness. The popular LSH functions can be used: E2LSH, random projection function, and random hyperplane based LSH for L1 distance [73, 84, 85].

In this study, the numerical attributes are concatenated and then the random threshold projection functions is then applied to hash the data [85]:

$$\hat{h}_d(x_{id}) = \begin{cases} 0, & \text{if } x_{id} > T_d \\ 1, & \text{otherwise} \end{cases}, \quad 1 \leq d \leq D \text{ and } A_d \notin \mathcal{A} \tag{4.2}$$

where T_d is the random scalar number in the range of all values in attribute A_d.

4.2.2.2 LSH functions for categorical data

We propose to use the binary partitioning space on each categorical domain to formulate the hash function, such that similar categorical values give the same hash value for that domain. There are several techniques that can subdivide the domain into two locality-sensitive subsets. We use Stoer-Wagner algorithm [77] for this task because of its simplicity and low complexity.

First, for creating the dissimilarity graphs for categorical attributes, we use the dissimilarity matrices from the previous step. The nodes of the graphs are created by the unique categorical values and the weights of the edges are created from the dissimilarity values between corresponding unique categorical values. A complete undirected graph is then formed based on a categorical attribute. Note that D_2 is the number of categorical attributes, so we can create D_2 complete undirected graphs in total.

Second, for creating the hash function for each categorical attribute, the Stoer-Wagner algorithm is directly applied to find the maximum cut on each graph. The maximum cut separates the domain into two subsets, such that the nodes (categorical values) on the same subset are tending similar to each other. An example of applying the Stoer-Wagner algorithm to categorical attribute is shown in Figure 4.2.

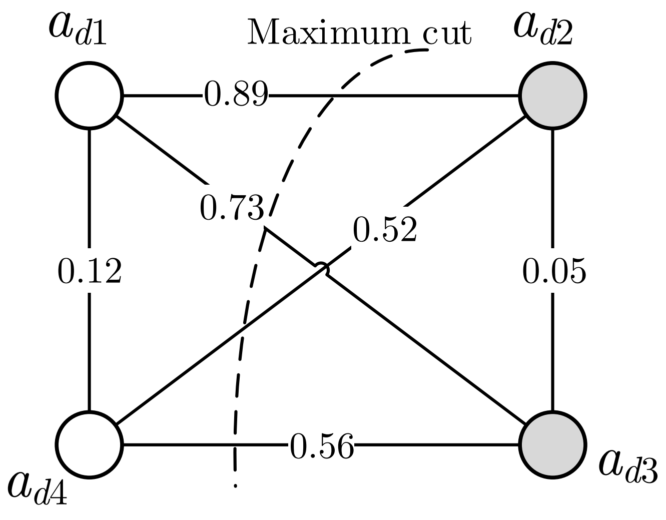

Figure 4.2: Maximum cut on graph of dissimilarity matrix

$\mathbb{A}_{d1}$ and $\mathbb{A}_{d2}$ (the subset order does not affect the accuracy of the algorithm) denote the two subsets in the d-th attribute, the hash function for categorical attribute can be formulated as:

$$\hat{h}_d(x_{id}) = \begin{cases} 0, & \text{if } x_{id} \in \mathbb{A}_{d1} \\ 1, & \text{otherwise} \end{cases}, \quad 1 \leq d \leq D \text{ and } A_d \in \mathcal{A} \tag{4.3}$$

4.2.2.3 Hash table generation for mixed data

So far, we have D LSH hash functions for two different types of data, then select l hash functions to index the dataset and build the hash table $\mathcal{H}$:

$$H = [h_1, \ldots, h_l], \text{ with } h_i \in \{\hat{h}_0, \ldots, \hat{h}_D\}, \quad 1 \leq i \leq l \tag{4.4}$$

Depending on the attribute type, the hash function should be selected accordingly (equation (4.2) or (4.3)).

The objects having the same hash values are stored in the same bucket B_j $(0 \leq j \leq 2^l - 1)$ with the key value of j:

$$\text{Key}(B_j) = j = H(\boldsymbol{x}) | \forall \boldsymbol{x} \in B_j, \quad 0 \leq j \leq 2^l - 1 \tag{4.5}$$

The overall process of constructing the hash table of the dataset is summarized in Algorithm 4.1.

Algorithm 4.1 Proposed hash table generation for categorical dataset

```
Input: $X, A_i (1 \leq i \leq D), l$
Output: Hash table $\mathcal{H}$
1: Use DILCA to create the distance matrix between values for each
   categorical attribute.
2: Generate the hash function for each attribute.
3: Select $l$ attributes randomly. Create the family of hash functions $H =
   [h_1, \ldots, h_l]$.
4: $\mathcal{H} \leftarrow [B_0, B_1, \ldots, B_{2^l-1}] \leftarrow [\emptyset, \emptyset, \ldots, \emptyset]$
5: for Object $\boldsymbol{x}$ in $X$ do
6:     hashValue $\leftarrow H(\boldsymbol{x})$
7:     $B_{\text{hashValue}} \leftarrow B_{\text{hashValue}} \cup \{\boldsymbol{x}\}$
8: end for
9: return $\mathcal{H}$
```

Moreover, different categorical attributes may affect differently to the clustering results (see Figure A.1 for more detail), it is better to select the "good" attributes for the hashing system [99]. We propose to select l hash functions with the highest total of weights in the maximum cuts to build the hash table. MC_d denotes the set of edges in the maximum cut of the d-th attribute. The average weight can be calculated as:

$$\hat{\gamma}_d = \sum_{\{a_{dj}, a_{dj'}\} \in \mathrm{MC}_d} \frac{\mathrm{Weight}(\{a_{dj}, a_{dj'}\})}{|\mathrm{MC}_d|} \tag{4.6}$$

where $\mathrm{Weight}(\{a_{dj}, a_{dj'}\}) = \delta(a_{dj}, a_{dj'})$ is the weight of edge $\{a_{dj}, a_{dj'}\}$ in the graph of the d-th dissimilarity matrix. The l hash functions with the highest average maximum cut weights are then selected for building the hash table:

$$H = [h_1, \ldots, h_l] \text{ where } h_j \in \{\hat{h}_d | 1 \leq d \leq D\}, \quad (1 \leq j \leq l) \tag{4.7}$$

subject to:

$$\forall 1 \leq i, j \leq l \text{ if } j > i \text{ then } \gamma_i \geq \gamma_j \tag{4.8}$$

where γ_i is the corresponding average maximum weight of hash function h_i.

Because our contribution is mainly for creating categorical hash functions, in the current version of our proposed method, the hash functions of categorical attributes have higher priorities than those of number numerical attributes. In this case, if $l \leq |\mathcal{A}|$ we select the hash functions by equation (4.7). In contrast, if $l > |\mathcal{A}|$, we select l hash functions of categorical attributes and select $|\mathcal{A}| - l$ hash functions of numerical attributes randomly.

4.2.3 Cluster prediction

LSH hash tables are commonly used for ANN search problems, but in this research, we use LSH hash tables to predict the potential clusters for the clustering problem. Similar to the principle of the data cluster, objects in the same bucket are more likely to be similar to each other. Thus, we propose to take advantage of these buckets to predict the cluster centers. The detailed procedure of cluster prediction is shown in Algorithm 4.2, which includes three main steps:

First, to make sure the number of buckets is larger than the number of clusters ($k < 2^l \ll N$), we must select the suitable number of hash functions l. Note that, if the number of buckets is too high, the required computation is also high. We recommend using a slightly higher number of the bucket just than the number of clusters k.

Second, the big buckets have a high potential to become clusters. Hence, the k biggest buckets are selected as the semi-cluster in our prediction method.

Then, each smaller bucket is merged into a semi-cluster (core bucket) based on the Hamming distance between their keys. Thus, a predicted cluster is formed from a semi-cluster and neighborhood buckets.

Algorithm 4.2 LSH-based cluster initialization

```
Input: Dataset $X$, $k$, hash table $\mathcal{H}$ containing $2^l$ buckets
Output: $k$ initial clusters $\hat{G}$
1: Select $k$ buckets having the highest number of objects:
       $\hat{\mathcal{H}} = [\hat{B}_1, \dots, \hat{B}_k]$, with $\hat{B}_i \in \mathcal{H}$; $\forall i, \forall j \neq i,\ \hat{B}_i \neq \hat{B}_j$
        subject to $|\hat{B}_i| \geq |\hat{B}_{i+1}|$, $i = 1, \dots, k-1$
        and $|\hat{B}_k| \geq |B_i|$, $\forall B_i \in (\mathcal{H} \setminus \hat{\mathcal{H}})$
2: Get the remaining buckets : $\tilde{\mathcal{H}} = \mathcal{H} \setminus \hat{\mathcal{H}}$
3: for each bucket $\tilde{B}_i$ in $\tilde{\mathcal{H}}$ do
4:     $b \leftarrow \arg\min_{\hat{B}_j \in \hat{\mathcal{H}}} \mathrm{Dis}_{\mathrm{Hamming}}\ (\mathrm{Key}(\tilde{B}_i), \mathrm{Key}(\hat{B}_j))$
       // Note that: Key($\tilde{B}_i$) and Key($\hat{B}_j$) return the keys of the based
       buckets on $\mathcal{H}$.
5:     $b \leftarrow b \cup \tilde{B}_i$
6: end for
7: return $\hat{\mathcal{H}}$
```

Figure 4.3 shows a visual example when using similarity of buckets for cluster prediction.

4.2.4 Optimizing number of hash functions l

To work effectively, it is necessary to have a hash table with the number of non-empty buckets greater than the number cluster k for our method. However, because the sizes of buckets are unpredictable, selecting a suitable number of hash functions l is tricky. Ideally, our method can work effectively when the number of non-empty buckets is the same as the number of cluster k, but in practice, we suggest selecting the number of hash functions l so that the number of maximum buckets is twice the number of clusters k, thereby $l \approx \log_2(k) + 1$. In case the number of non-empty buckets is smaller than k, we can increase the number of hash functions l to 1 or just proceed with empty core bucket(s).

4.3 LSH-k-prototypes iteration

In every iteration of the k-means-like algorithm, it is required to compute the distances from all objects to all cluster representations, which dramatically

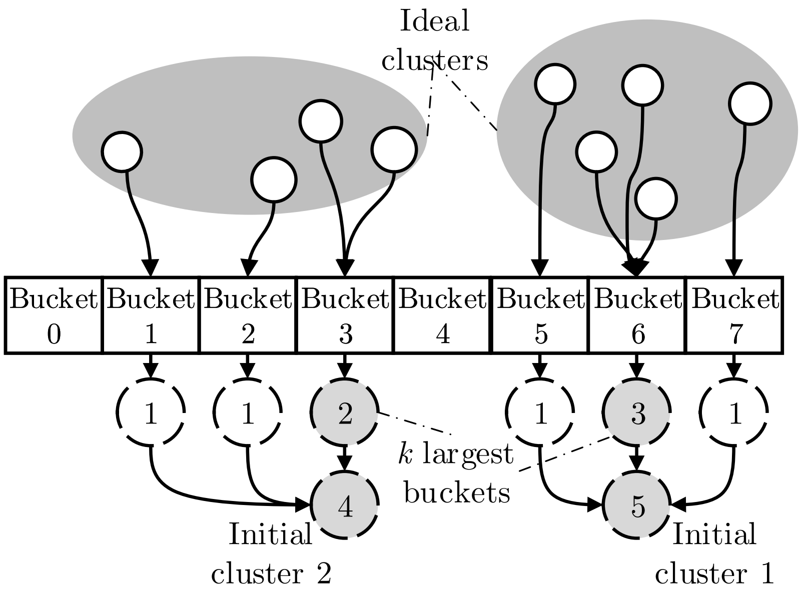

Figure 4.3: Illustration of cluster initialization based on LSH

increases the clustering time of the k-means algorithm for a large dataset or dataset with a high number of clusters k. As the result, in the literature chapter, some studies try to reduce the number of objects or reduce the complexity of distance calculations. However, we focus on a new approach that can also reduce the computation of algorithms by avoiding unnecessary distance computations.

In fact, in each iteration, it needs to find the nearest cluster representation for each data, and an object is more likely to be remained in this own cluster or moved to a nearby cluster. Therefore, we recommend looking for the nearest cluster representation of each object in a portion of all cluster representations, this portion is called a shortlist for each cluster. k' denotes the size of cluster neighborhood, such that $0 < k' \leq k$, and Shortlist(G_j) denotes the set of neighborhood clusters for cluster G_j, such that $|\text{Shortlist}(G_j)| = k'$ for $1 \leq j \leq k$.

Figure 4.4 presents an example of using shortlists to find the nearest clusters in the case of $k = 64$ and $k' = 8$. For instance, if an object is currently belonging to cluster 1, in the next iteration, the object has the possibility to move to either of clusters 0, 9, 26, 29, 32, 37, 43, 50, or 1. As such, the number of distance computations for reassignment is reduced by a factor of $k/k' = 8$ at each iteration.

The initialization shortlists of all clusters can be easily obtained by using the Hamming distance between semi-clusters in the previous step. In

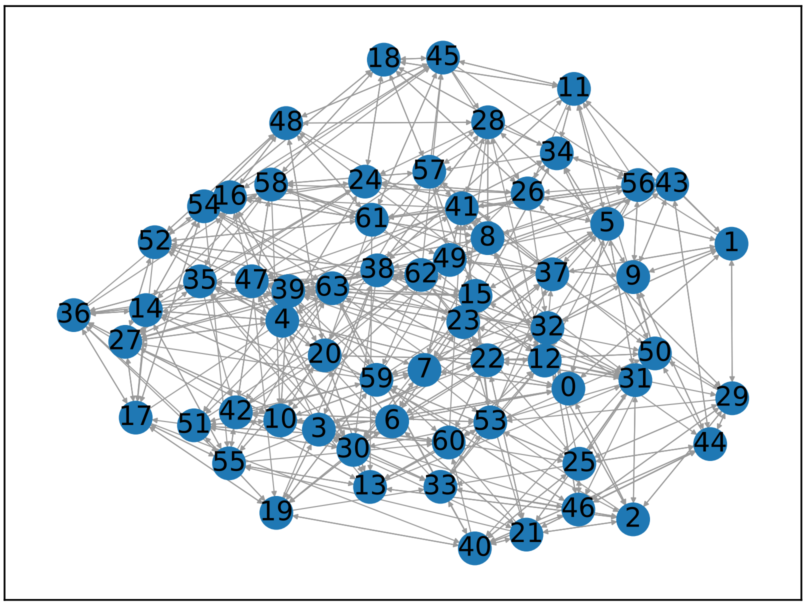
Figure 4.4: Graph visualization of 8-nearest clusters for searching in the case of 64 clusters

each iteration, the shortlists are also updated when the values of cluster representations are changed.

After the clusters are predicted, the membership matrix U can then be established.

4.3.1 Updating ***prototype*s**

Our method utilizes the *improved prototype* to handle the mixed data. To simply, from now on, the *improved prototype* in section 2.3.3.2 is called *prototype*. Remark that, the *prototype* is the set that is notated as $P = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_k]$ and each *prototype* is a mixed structure of numerical values and categorical probabilities. Because our *prototype*s are the mixed structure of numerical values and *representatives* values, the update formula of *prototype* also includes two parts (equations (2.15) and (2.20)):

$$p_{jd} = \begin{cases} \left(\sum_{i=1}^{N} u_{ij} x_{id}\right) \Big/ \left(\sum_{i=1}^{N} u_{ij}\right), & \text{if } A_d \notin \mathcal{A} \\ [(v_d, \Pr_{G_j}(v_d)) | v_d \in \mathbb{A}_d], & \text{otherwise} \end{cases}, \quad 1 \leq d \leq D, 1 \leq j \leq k \tag{4.9}$$

where $\Pr_{G_j}(v_d)$ is the probability of categorical value v_d that appears in cluster G_j.

4.3.2 Updating cluster neighborhoods

After the values of *prototype*s are changed, the neighborhoods of them need to be updated accordingly. Note that, after the first iteration, the semi-cluster may not be the center of cluster anymore. Thus, we need to directly use the *prototype*s to compute the neighborhoods or shortlist. Based on the distance function (see equation (2.30)) from an object to a *prototype*, we can develop the distance function from between *prototype*s as:

$$\begin{aligned}\mathrm{Dis}(\boldsymbol{p}_i, \boldsymbol{p}_j) = \omega \sum_{\substack{d=1 \\ A_d \in \mathcal{A}}}^{D} \left(\sqrt{\sum_{v_d \in \mathbb{A}_d} \left(\mathrm{Pr}_{G_i}(v_d) - \mathrm{Pr}_{G_j}(v_d)\right)^2} \right) \\ + (1-\omega) \sqrt{\sum_{\substack{d=1 \\ A_d \notin \mathcal{A}}}^{D} (p_{id} - p_{jd})^2}\end{aligned} \tag{4.10}$$

In a word, the equation above includes the weighted sum of two Euclidean measurements of *representatives* or *prototype*s.

The neighborhood of each cluster is then recalculated by finding the k' nearest *prototype*s for the *prototype* of every cluster.

4.3.3 A variant of shortlist updating method using shortlist rotation

Because the required number of distance calculations of the original shortlist updating method is $k(k-1)/2$; when the number of clusters is too large, the effectiveness of this method is reduced. For dealing with datasets having massive numbers of clusters k (say, $k \geq 512$), we propose to use the same shortlists created at the initialization stage with a rotation operation defined as follows instead of computing the shortlist of a cluster after each iteration.

$\hat{\mathrm{S}}\mathrm{hortlist}(G_j)$ denotes the initial shortlist of the j-th cluster ($1 \leq j \leq k$) that is achieved from the cluster prediction stage. At the "iter"-th iteration, the shortlist of cluster G_j is determined by:

$$\begin{aligned}\mathrm{Shortlist}^{\mathrm{iter}}(G_j) \leftarrow \hat{\mathrm{S}}\mathrm{hortlist}(G_{j'}), \quad 1 \leq j \leq k \\ \text{where } j' = ((j - 1 + \mathrm{iter}) \mod k) + 1\end{aligned} \tag{4.11}$$

This modification with shortlists rotated for our LSH-k-representatives algorithm is called LSH-k-representatives(RS) in this dissertation. However, this approach may cause biases for the clusters that have high frequencies among all shortlists. To avoid such potential biases, we limit the numbers of presences of all clusters to k' among all shortlists. In addition, each cluster itself must be included in its own current shortlist. Note that, these constraints are also applied when updating the shortlist using equation (4.10).

This rotation approach dramatically drops the required distance computations between cluster representatives. However, the accuracy of LSH-k-representatives(RS) is expected to be slightly reduced compared to its original version of the technique of shortlist updating.

Consequently, for LSH-k-representatives algorithm, we have two different LSH-k-representatives(H) and LSH-k-representatives(RS) with different techniques to update the shortlists during the clustering iterations.

4.3.4 Updating degree of membership

To update the degree of membership for every object, we examine the distance (see equation (2.30)) from objects to cluster *prototype*s that belonging to the shortlist of the current cluster. The objects are join the clusters that are closest to them:

$$u_{ij} = \begin{cases} 1, & \text{if } j = \underset{j' | G_{j'} \in \text{Shortlist}(G_{j[i]})}{\arg\min} \text{Dis}(\boldsymbol{x}_i, \boldsymbol{r}_{j'}) \\ 0, & \text{otherwise} \end{cases}, \quad 1 \leq i \leq N, 1 \leq j \leq k \tag{4.12}$$

4.4 LSH-k-prototypes algorithm

With the previously developed phases of cluster initialization (sections 4.2) and iteration (section 4.3), we are now ready to formulate the proposed clustering method based on LSH and nearest-neighbor search as summarized in **Algorithm 4.3** below.

Algorithm 4.3 Proposed LSH-k-representatives algorithm

```
Input: Categorical dataset $X$, $k$, $l$, max_iter
Output: prototypes $P$, membership matrix $U$ for $k$ clusters that optimize
  the objective function in equation (2.24).
1: Extract the attribute domains from the dataset.
2: Generate the hash table $\mathcal{H}$ with $l$ hash functions by Algorithm 4.1.
3: Create $k$ initial clusters by Algorithm 4.2 using the previously gener-
   ated hash table $\mathcal{H}$.
4: for $i \leftarrow 0$ ; $i <$ max_iter; $i \leftarrow i + 1$ do
5:    Fix $U$, calculate the corresponding $\hat{P}$ via equations (2.19)–(2.20).
6:    $P \leftarrow \hat{P}$
7:    Re-construct Shortlist($G_j$) for $j = 1, \ldots, k$.
8:    for each object $\boldsymbol{x}$ in $X$ do
9:       Call $G_j$ the current cluster that holds $\boldsymbol{x}$.
10:      Retrieve the corresponding Shortlist($G_j$).
11:      Calculate the distances from $\boldsymbol{x}$ to the prototypes in Shortlist($G_j$) by
         equation (2.22).
12:      Update $U$: Assign $\boldsymbol{x}$ to the closest cluster via equation (4.12).
13:   end for
14:   if $U$ is unchanged then
15:      Break the loop.
16:   end if
17: end for
18: return $P, U$
```

4.5 LSH-k-representatives: a modification of LSH-k-prototypes for clustering categorical data

To clarify, LSH-k-prototypes is proposed to handle general mixed data, in the iterations of LSH-k-prototypes, the processes are also divided into two parts: processing numerical data and processing categorical value. Besides that, our proposed LSH-based cluster prediction method is mainly established from categorical attributes, we thus make a modification of LSH-k-prototypes so-called LSH-k-representatives that focus on handling categorical attributes only. Therefore, all the attributes in the categorical datasets are assumed to be the categorical attributes:

$$\forall 1 \leq d \leq D, A_d \in \mathcal{A} \tag{4.13}$$

In this chapter, the experiments are also divided for showing the clustering results of LSH-k-prototypes and LSH-k-representatives.

4.6 An illustrated example on soybean-small dataset

To further clarify how the proposed method works, this section shows an example of applying our LSH-based cluster prediction technique into the soybean-small dataset which is the smallest UCI categorical dataset.

First of all, Figure 4.5 shows the DILCA dissimilarity matrices of 21 categorical attributes in the soybean-small dataset, the black color represent the high dissimilarity between two corresponding pair of categorical values.

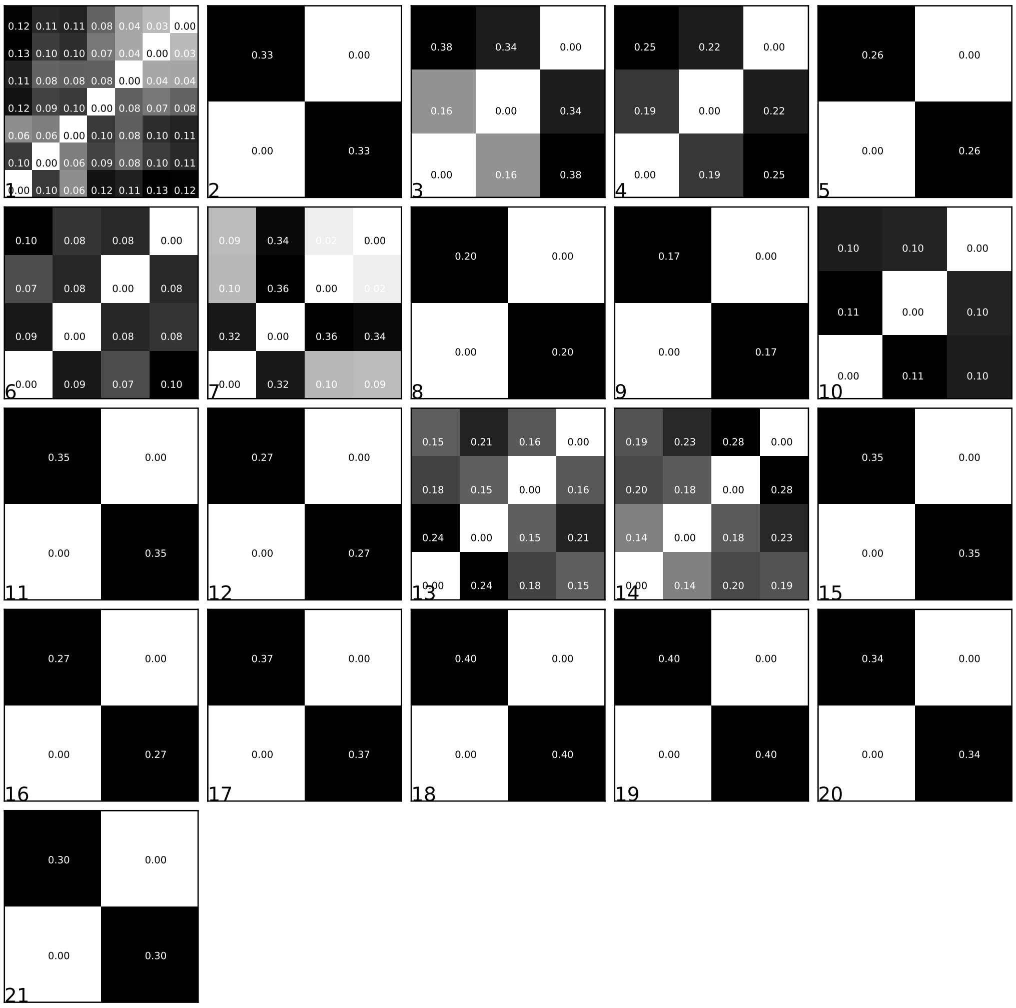

Figure 4.5: Dissimilarity matrices of 21 attributes in the soybean-small dataset

Next, suppose that we want to create the hash table with $l = 5$ hash functions, we then select $\mathbb{A}_{18}, \mathbb{A}_{19}, \mathbb{A}_{17}, \mathbb{A}_{11}$, and $\mathbb{A}_{15}$ to create the 5 hash functions because these attributes have the highest average maximum cut values as 0.398, 0.398, 0.376, 0.356, and 0.348, respectively.

After that, we can create the five corresponding hash functions as:

$$
\begin{aligned}
h_1(\boldsymbol{x}_i) &= \begin{cases} 0 & \text{, if } x_{i18} = 1 \\ 1 & \text{, otherwise} \end{cases} &
h_2(\boldsymbol{x}_i) &= \begin{cases} 0 & \text{, if } x_{i19} = 1 \\ 1 & \text{, otherwise} \end{cases} \\
h_3(\boldsymbol{x}_i) &= \begin{cases} 0 & \text{, if } x_{i17} = 1 \\ 1 & \text{, otherwise} \end{cases} &
h_4(\boldsymbol{x}_i) &= \begin{cases} 0 & \text{, if } x_{i11} = 1 \\ 1 & \text{, otherwise} \end{cases} \\
& & h_5(\boldsymbol{x}_i) &= \begin{cases} 0 & \text{, if } x_{i15} = 1 \\ 1 & \text{, otherwise} \end{cases}
\end{aligned}
\tag{4.14}
$$

From these hash functions, we can calculate the hash values for all objects in the dataset by using the family of hash functions $H = [h_1, h_2, h_3, h_4, h_5]$. All hash values of all categorical objects are shown in Table 4.2, the objects are also colored with different colors depending on their hash values.

Table 4.2: Hash values of 47 objects in the soybean-small dataset

Object	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	$\boldsymbol{x}_6$	$\boldsymbol{x}_7$	$\boldsymbol{x}_8$	$\boldsymbol{x}_9$	$\boldsymbol{x}_{10}$	$\boldsymbol{x}_{11}$	$\boldsymbol{x}_{12}$
$H(\boldsymbol{x})$	28	28	28	28	28	28	28	28	28	28	5	5
Object	$\boldsymbol{x}_{13}$	$\boldsymbol{x}_{14}$	$\boldsymbol{x}_{15}$	$\boldsymbol{x}_{16}$	$\boldsymbol{x}_{17}$	$\boldsymbol{x}_{18}$	$\boldsymbol{x}_{19}$	$\boldsymbol{x}_{20}$	$\boldsymbol{x}_{21}$	$\boldsymbol{x}_{22}$	$\boldsymbol{x}_{23}$	$\boldsymbol{x}_{24}$
$H(\boldsymbol{x})$	5	5	5	5	5	5	5	5	27	31	27	31
Object	$\boldsymbol{x}_{25}$	$\boldsymbol{x}_{26}$	$\boldsymbol{x}_{27}$	$\boldsymbol{x}_{28}$	$\boldsymbol{x}_{29}$	$\boldsymbol{x}_{30}$	$\boldsymbol{x}_{31}$	$\boldsymbol{x}_{32}$	$\boldsymbol{x}_{33}$	$\boldsymbol{x}_{34}$	$\boldsymbol{x}_{35}$	$\boldsymbol{x}_{36}$
$H(\boldsymbol{x})$	31	25	27	31	27	31	29	29	29	29	29	29
Object	$\boldsymbol{x}_{37}$	$\boldsymbol{x}_{38}$	$\boldsymbol{x}_{39}$	$\boldsymbol{x}_{40}$	$\boldsymbol{x}_{41}$	$\boldsymbol{x}_{42}$	$\boldsymbol{x}_{43}$	$\boldsymbol{x}_{44}$	$\boldsymbol{x}_{45}$	$\boldsymbol{x}_{46}$	$\boldsymbol{x}_{47}$	
$H(\boldsymbol{x})$	29	29	29	29	29	29	29	29	29	29	29	

It is interesting that, with $l = 5$, we can have up to $2^5 = 32$ different hash values or bucket IDs but there are only 6 non-empty buckets. The buckets are arranged in descending order of size as follows: $B_{29} = \{\boldsymbol{x}_{31}, \ldots, \boldsymbol{x}_{47}\}$, $B_{28} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{10}\}$, $B_5 = \{\boldsymbol{x}_{11}, \ldots, \boldsymbol{x}_{20}\}$, $B_{31} = \{\boldsymbol{x}_{22}, \boldsymbol{x}_{24}, \boldsymbol{x}_{25}, \boldsymbol{x}_{28}, \boldsymbol{x}_{30}\}$, $B_{27} = \{\boldsymbol{x}_{21}, \boldsymbol{x}_{23}, \boldsymbol{x}_{27}, \boldsymbol{x}_{29}\}$, and $B_{25} = \{\boldsymbol{x}_{26}\}$.

Remind that because we want to have 4 clusters for the soybean-small dataset (k=4), we can select 4 biggest buckets as the core buckets as B_{29}, B_{28}, B_5, and B_{31}. Therefore, objects in buckets B_{27} and B_{25} must be moved to the nearest core bucket.

The Hamming distances between B_{27} and B_{25} to the core buckets can be known as:

$$\begin{aligned} &\text{Hamming}(27,29) = 2 \qquad &&\text{Hamming}(25,29) = 3 \\ &\text{Hamming}(27,28) = 3 &&\text{Hamming}(25,28) = 2 \\ &\text{Hamming}(27,05) = 4 &&\text{Hamming}(25,05) = 5 \\ &\text{Hamming}(27,31) = 1 &&\text{Hamming}(25,31) = 2 \end{aligned} \tag{4.15}$$

Next, because B_{31} is the nearest core bucket for B_{27} and B_{28} is the nearest core bucket for B_{25}, we need to move objects from B_{27} to B_{31} and objects from B_{25} to B_{28}. Thereby, we can illustrate the four predicted clusters with different colors in Table 4.3.

Table 4.3: Prediction labels for soybean-small dataset

🟦 $\boldsymbol{x}_1$	🟦 $\boldsymbol{x}_2$	🟦 $\boldsymbol{x}_3$	🟦 $\boldsymbol{x}_4$	🟦 $\boldsymbol{x}_5$	🟦 $\boldsymbol{x}_6$	🟦 $\boldsymbol{x}_7$	🟦 $\boldsymbol{x}_8$	🟦 $\boldsymbol{x}_9$	🟦 $\boldsymbol{x}_{10}$	🟧 $\boldsymbol{x}_{11}$	🟧 $\boldsymbol{x}_{12}$
🟧 $\boldsymbol{x}_{13}$	🟧 $\boldsymbol{x}_{14}$	🟧 $\boldsymbol{x}_{15}$	🟧 $\boldsymbol{x}_{16}$	🟧 $\boldsymbol{x}_{17}$	🟧 $\boldsymbol{x}_{18}$	🟧 $\boldsymbol{x}_{19}$	🟧 $\boldsymbol{x}_{20}$	🟨 $\boldsymbol{x}_{21}$	🟨 $\boldsymbol{x}_{22}$	🟨 $\boldsymbol{x}_{23}$	🟨 $\boldsymbol{x}_{24}$
🟨 $\boldsymbol{x}_{25}$	🟦 ~~$\boldsymbol{x}_{26}$~~	🟨 $\boldsymbol{x}_{27}$	🟨 $\boldsymbol{x}_{28}$	🟨 $\boldsymbol{x}_{29}$	🟨 $\boldsymbol{x}_{30}$	🟩 $\boldsymbol{x}_{31}$	🟩 $\boldsymbol{x}_{32}$	🟩 $\boldsymbol{x}_{33}$	🟩 $\boldsymbol{x}_{34}$	🟩 $\boldsymbol{x}_{35}$	🟩 $\boldsymbol{x}_{36}$
🟩 $\boldsymbol{x}_{37}$	🟩 $\boldsymbol{x}_{38}$	🟩 $\boldsymbol{x}_{39}$	🟩 $\boldsymbol{x}_{40}$	🟩 $\boldsymbol{x}_{41}$	🟩 $\boldsymbol{x}_{42}$	🟩 $\boldsymbol{x}_{43}$	🟩 $\boldsymbol{x}_{44}$	🟩 $\boldsymbol{x}_{45}$	🟩 $\boldsymbol{x}_{46}$	🟩 $\boldsymbol{x}_{47}$	

Table 4.4: Ground-truth labels for soybean-small dataset

🟦 $\boldsymbol{x}_1$	🟦 $\boldsymbol{x}_2$	🟦 $\boldsymbol{x}_3$	🟦 $\boldsymbol{x}_4$	🟦 $\boldsymbol{x}_5$	🟦 $\boldsymbol{x}_6$	🟦 $\boldsymbol{x}_7$	🟦 $\boldsymbol{x}_8$	🟦 $\boldsymbol{x}_9$	🟦 $\boldsymbol{x}_{10}$	🟧 $\boldsymbol{x}_{11}$	🟧 $\boldsymbol{x}_{12}$
🟧 $\boldsymbol{x}_{13}$	🟧 $\boldsymbol{x}_{14}$	🟧 $\boldsymbol{x}_{15}$	🟧 $\boldsymbol{x}_{16}$	🟧 $\boldsymbol{x}_{17}$	🟧 $\boldsymbol{x}_{18}$	🟧 $\boldsymbol{x}_{19}$	🟧 $\boldsymbol{x}_{20}$	🟨 $\boldsymbol{x}_{21}$	🟨 $\boldsymbol{x}_{22}$	🟨 $\boldsymbol{x}_{23}$	🟨 $\boldsymbol{x}_{24}$
🟨 $\boldsymbol{x}_{25}$	🟨 $\boldsymbol{x}_{26}$	🟨 $\boldsymbol{x}_{27}$	🟨 $\boldsymbol{x}_{28}$	🟨 $\boldsymbol{x}_{29}$	🟨 $\boldsymbol{x}_{30}$	🟩 $\boldsymbol{x}_{31}$	🟩 $\boldsymbol{x}_{32}$	🟩 $\boldsymbol{x}_{33}$	🟩 $\boldsymbol{x}_{34}$	🟩 $\boldsymbol{x}_{35}$	🟩 $\boldsymbol{x}_{36}$
🟩 $\boldsymbol{x}_{37}$	🟩 $\boldsymbol{x}_{38}$	🟩 $\boldsymbol{x}_{39}$	🟩 $\boldsymbol{x}_{40}$	🟩 $\boldsymbol{x}_{41}$	🟩 $\boldsymbol{x}_{42}$	🟩 $\boldsymbol{x}_{43}$	🟩 $\boldsymbol{x}_{44}$	🟩 $\boldsymbol{x}_{45}$	🟩 $\boldsymbol{x}_{46}$	🟩 $\boldsymbol{x}_{47}$	

Compare to the ground-truth labels of soybean-small dataset in Table 4.4, our method assigns the wrong labels for only one object $\boldsymbol{x}_{26}$. Eventually, our LSH-based cluster prediction technique can predict labels correctly for up to 46/47 objects in the soybean-small dataset, which is equivalent to the Purity score of 0.98 (98%) for using the initial step only.

4.7 Experiments and Results

4.7.1 Datasets and testing environments

16 UCI categorical datasets, large synthetic categorical datasets, and a big real-world categorical dataset *US Census* are used in our experiments. The descriptions of these datasets are as follows:

4.7.1.1 16 UCI categorical datasets

The 16 UCI categorical datasets were obtained from UCI repository [86]. These datasets are broadly utilized by numerous studies in classification and clustering with categorical data or mixed data. The characteristics of these datasets are depicted in Table 4.5, the range outline of the likeness of pairwise objects (Visual Assessment of Tendency (VAT) [87]) in these datasets appears in Figure 4.6.

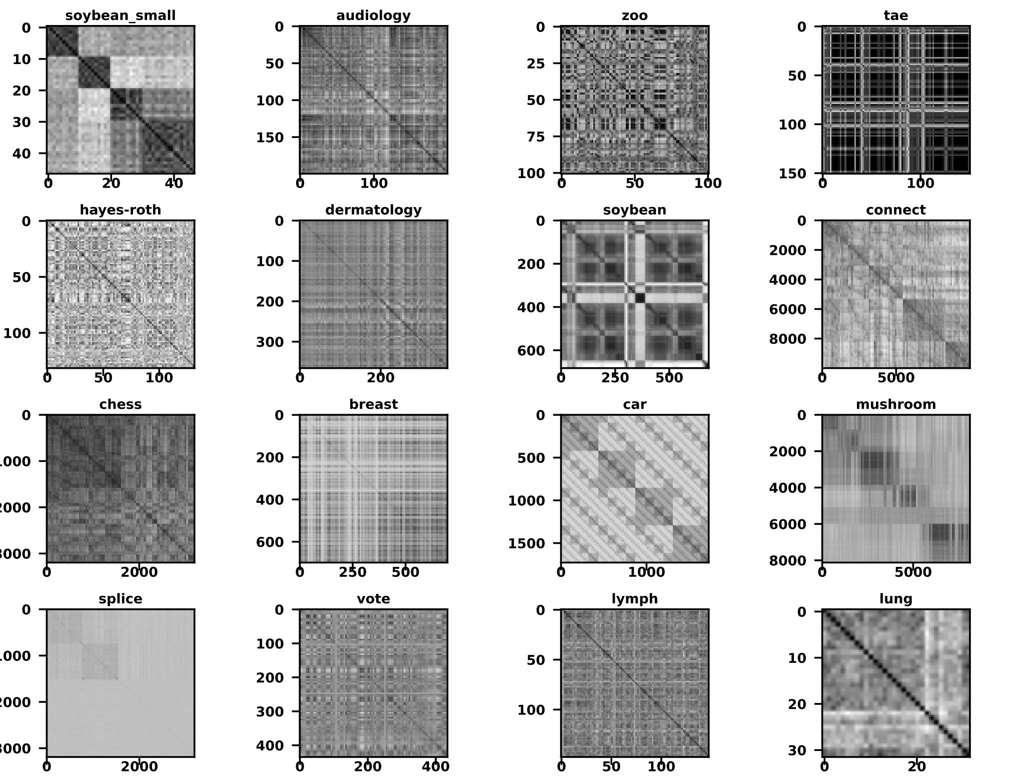

Figure 4.6: Visual Assessment of Tendency (VAT) charts of 16 UCI datasets

Table 4.5: Common categorical datasets

Name	#Items	#Attributes	#Classes	Type
Soybean_small	47	21	4	Categorical
Audiology	226	21	4	Categorical
Zoo	101	17	7	Categorical, Integer
Tae	151	5	3	Categorical, Integer
Hayes-roth	132	5	3	Categorical
Dermatology	366	34	6	Categorical, Integer
Soybean	683	35	19	Categorical
Connect	10,000	42	3	Categorical
Chess	3,196	36	2	Categorical
Breast	699	9	2	Categorical
Car	1,728	6	4	Categorical
Mushroom	8,124	22	2	Categorical
Splice	3,190	60	3	Categorical
Vote	435	16	2	Categorical
Lymph	148	18	4	Categorical
Lung	32	56	3	Categorical

4.7.1.2 Large synthetic categorical datasets

In order to justify the scalability of our proposed algorithm, the Datagen Dataset Generator is utilized to generate numerous synthetic categorical datasets with different variants of distributions for different clusters [88]. Furthermore, the hidden rules for decision-making problems were also applied for all clusters for increasing the context-sensitive factor of generated objects. In particular, the dataset size is in the range of 2^{10} to 2^{28} (1,024 to 268,435,456), the number of clusters is altered from the range of 2^3 to 2^9 (8 to 512), and the number of attributes is ranged from 2^3 to 2^4 (8 to 16).

4.7.1.3 Large real categorical dataset (US Census)

The US Census dataset which includes 1,048,575 categorical objects with 68 attributes was used to justify the capacity of our proposed method in terms of handling real large categorical data [86]. Because the raw US Census dataset has no labels, we conduct the clustering multiple time with different values of the number of clusters k to preset the benchmark labels. In this case, we can use the basic metrics to compare the effectiveness of the compared methods.

4.7.1.4 Testing environments

We select the competitors whose source codes are available to compare the performances including: k-means++ [37], k-means$||$ [36], k-modes [40], Cao [42], k-representatives [15] (k-reps for short), MH-k-modes [32], and mk-centers [89] (mk-cens for short). All source codes, including both competitors and our methods, are written by Python programming language. All experiments were conducted by a high-end computer cluster with Intel Xeon G-6240M 2.6GHz (18 Cores $\times$ 4) CPU.

4.7.2 Evaluation metrics

The evaluation metrics are categorized into two headings: complexity metrics and effectiveness metrics. For complexity analysis, the following concepts are used:

- Convergence: The average number of iterations until converged can show the convergence factor of k-means algorithms.
- Time complexity: Total running time of compared methods together with their preprocessing time is measured as the practical complexity of clustering algorithms. Moreover, the average running time for each iteration can show the relative complexity of k-means-like algorithms.
- Space complexity: The average and the peak memory of each clustering method are utilized to compare the space complexity of compared clustering methods.

For justifying the clustering effectiveness, the accuracy of outcome clusters was evaluated by comparing with the ground-truth labels of datasets. Let $\mathcal{G}^* = \{G_1^*, \ldots, G_k^*\}$ be the ground-truth clusters of all objects and $\mathcal{G} = \{G_1, \ldots, G_k\}$ be the outcome clusters of the algorithms, the groups G can be inferred from the membership matrix U. Technically, the following metrics are utilized:

- Recall, Precision, and Accuracy scores: These metrics are the simplest effectiveness metrics that are commonly used in the supervised learning research field [90]. In particular, these metrics consider the number of matches and discrepancies of the outcome labels and the ground-truth labels of all pairwise objects; and normalize these scores into the scale of zero-match (0) and perfect-match (1).
- Purity score: This metric seeks the highest ratio of correct assignments between $\mathcal{G}^*$ and $\mathcal{G}$:

$$\text{Purity}(\mathcal{G}, \mathcal{G}^*) = \frac{1}{N} \sum_{i}^{k} \max_{j|1 \leq j \leq k} |G_i \cap G_j^*| \tag{4.16}$$

It is clear that with a high number of clusters k, the high value of Purity can be obtained. Therefore, we need other metrics to evaluate the clustering results with a high number of clusters k.

- Normalized Mutual Information (NMI) score: The range of NMI is valid from 0 (no mutual information) to 1 (perfect match): the NMI score between two sets of clusters can be computed as:

$$\text{NMI}(\mathcal{G}, \mathcal{G}^*) = \frac{\text{MI}(\mathcal{G}, \mathcal{G}^*)}{\text{mean}(\text{H}(\mathcal{G}), \text{H}(\mathcal{G}^*))} \tag{4.17}$$

where $\text{H}(\mathcal{G})$ is the entropy value of set of clusters $\mathcal{G}$:

$$\text{H}(\mathcal{G}) = -\sum_{i=1}^{k} \frac{|G_i|}{N} \log\left(\frac{|G_i|}{N}\right) \tag{4.18}$$

and $\text{MI}(\mathcal{G}, \mathcal{G}^*)$ is the cross entropy of $\mathcal{G}$ and $\mathcal{G}^*$:

$$\text{MI}(\mathcal{G}, \mathcal{G}^*) = \sum_{i=1}^{k} \sum_{j=1}^{k} \frac{|G_i \cap G_j^*|}{N} \log\left(\frac{N|G_i \cap G_j^*|}{|G_i||G_j^*|}\right) \tag{4.19}$$

- Adjusted Rand Index (ARI): This metric can be used to determine whether two sets of clusters $\mathcal{G}$ and $\mathcal{G}^*$ are similar to each other or not:

$$\text{ARI}(\mathcal{G}, \mathcal{G}^*) = \frac{\text{RI}(\mathcal{G}, \mathcal{G}^*) - \text{E}[\text{RI}(\mathcal{G}, \mathcal{G}^*)]}{\max(\text{RI}(\mathcal{G}, \mathcal{G}^*)) - \text{E}[\text{RI}(\mathcal{G}, \mathcal{G}^*)]} \tag{4.20}$$

where $\text{RI}(\mathcal{G}, \mathcal{G}^*)$ or Rand Index (RI) is the number of pairs of objects that together belong $\mathcal{G}$ and $\mathcal{G}^*$ at the same time; $\text{E}[\text{RI}(\mathcal{G}, \mathcal{G}^*)]$ is the expectation of $\text{RI}(\mathcal{G}, \mathcal{G}^*)$ and $\max(\text{RI}(\mathcal{G}, \mathcal{G}^*))$ is the largest value of RI over all permutations.

- Adjusted Mutual Information (AMI) score: This metric can normalize the entropy of each clustering (NMI score) with the average and expectation of pairs of all clusters (ARI score):

$$\text{AMI}(\mathcal{G}, \mathcal{G}^*) = \frac{\text{MI}(\mathcal{G}, \mathcal{G}^*) - \text{E}[\text{MI}(\mathcal{G}, \mathcal{G}^*)]}{\text{mean}(\text{H}(\mathcal{G}), \text{H}(\mathcal{G}^*)) - \text{E}[\text{MI}(\mathcal{G}, \mathcal{G}^*)]} \tag{4.21}$$

- Homogeneity score: This metric plays an important role for checking whether each cluster in the outcome clusters contains data objects belonging to a single class in the ground-truth clusters:

$$\text{Homogeneity}(\mathcal{G}, \mathcal{G}^*) = 1 - \frac{\text{H}(\mathcal{G}|\mathcal{G}^*)}{\text{H}(\mathcal{G})} \tag{4.22}$$

with $\text{H}(\mathcal{G}|\mathcal{G}^*)$ being the conditional entropy of G when giving G^*:

$$\text{H}(\mathcal{G}|\mathcal{G}^*) = \sum_{i=1}^{k}\sum_{j=1}^{k} \frac{|G_i \cap G_j^*|}{N} \log\Big(\frac{|G_i \cap G_j^*|}{|G_j^*|}\Big) \tag{4.23}$$

- Silhouette score: This metric is widely used for evaluating the unsupervised learning techniques as this does not use the ground-truth labels. In detail, Silhouette metric blends the average of the inter-cluster distance and the average of outer-cluster distance:

  $$\text{Silhouette}(\mathcal{G}) = \frac{\text{WGSS}^*(\mathcal{G}) - \text{BGSS}^*(\mathcal{G})}{\max(\text{WGSS}^*(\mathcal{G}), \text{BGSS}^*(\mathcal{G}))} \tag{4.24}$$

  where $\text{WGSS}^*(\mathcal{G})$ and $\text{BGSS}^*(\mathcal{G})$ are the average of inter-cluster distance and the average of outer-cluster distance, which can be delivered from WGSS and BGSS functions (see equations (2.8) and (2.9)).

4.7.3 Results on categorical data

For handling categorical datasets, our method can perform the processing for categorical data parts only. In that case, the *prototype*s become the *representatives.* Thus, in this section, our method is called LSH-k-representatives or LSH-k-reps for short. Besides that, for a fair comparison, LSH-k-reps also have two versions LSH-k-reps(Init) and LSH-k-reps(Full); LSH-k-reps(Init) is the proposed method that applies the LSH *cluster prediction optimization* only; while LSH-k-reps(Full) is the complete LSH-k-reps method including *cluster initialization optimization* and *cluster iteration optimization.*

4.7.3.1 Time complexity analysis

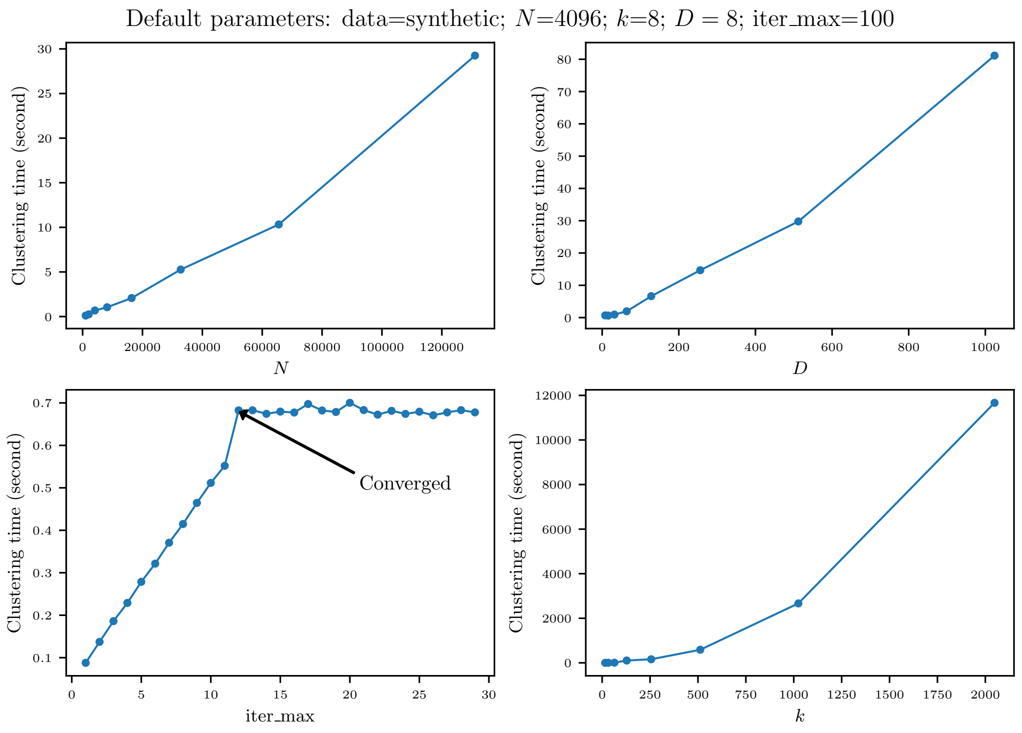

Figure 4.7: Time complexity analysis of LSH-k-prototypes

Figure 4.7 shows the complexity of out LSH-k-prototypes in the context of four main parameters: number of objects N, number of attributes D, maximum iteration number iter_max, and number of clusters k. It is clear that our method has linear complexity for the variables N, D, iter_max, and k. In terms of the maximum number of iterations iter_max, because our method is converged at iteration 13, the algorithm stops at this iteration. Which makes the clustering times are the same when we set the iter_max higher than 13. In terms of the number of clusters k, the pattern is quite similar to the nonlinear complexity pattern. This can be explained by that when clustering data with a higher number of clusters k the method needs more iterations to be converged. Overall, the time complexity of our method is $\mathcal{O}(NDkt)$, where t is the number of required iterations.

4.7.3.2 LSH-based cluster prediction technique analysis

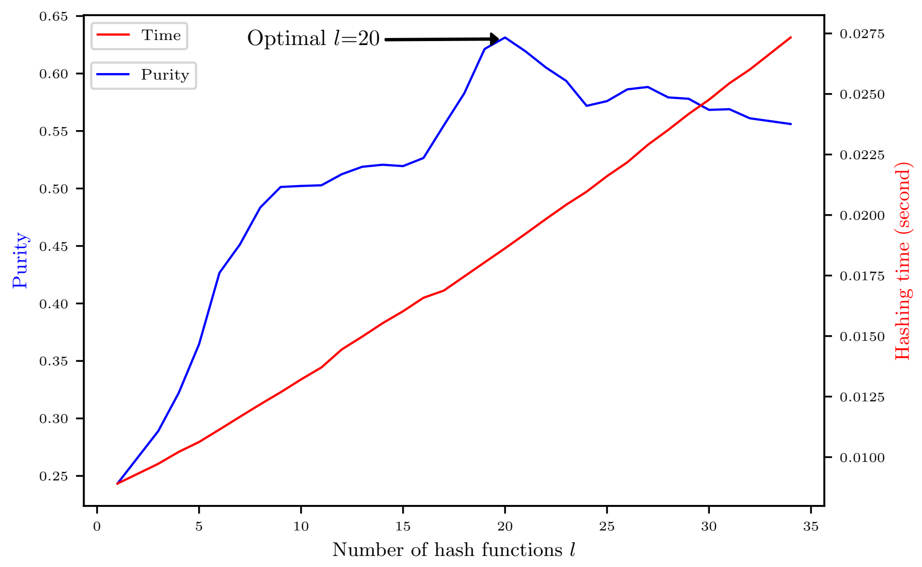

Figure 4.8: Purity scores for different numbers of hash functions l on the Soybean dataset

In Figure 4.8, we experiment with the accuracy of clustering with the different number of hash functions l for the soybean dataset. As a piece of evidence, because the task of hash values with a high number of hash functions requires more computation than it with a lower number of hash functions, the computation time is increasing followed by the increasing of the number of hash functions. On the other hand, when the number of hash functions is too small, the number of core buckets is insufficient, which leads to the poor result of clustering accuracy. Likewise, if the number of hash functions is too high, the number of small buckets is too high and the size of core buckets is inconsiderable, which also leads to the reduction of clustering performance. For soybean-small, the optimal value of the number of the hash functions is 20 and the acceptable value of the number of hash functions is in the range from 9 to 20.

4.7.3.3 Results on 16 UCI categorical datasets

The comprehensive comparisons on the clustering effectiveness between the proposed method and with other competitors are shown in Tables 4.6, 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, and 4.13. Each table shows the average outcome value of an evaluation metric. In this experiment, we take the averages records for 128 times with different random seed for each method in each dataset.

At a glance, LSH-k-reps(Init) outperforms other clustering techniques in most of the evaluation metrics, namely Purity, AMI, ARI, Accuracy, and Precision. Besides that, LSH-k-reps(Init) also gives comparable results for NMI, Homogeneity, and Recall.

Table 4.6: Purity result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.6596	0.6596	**1.0000**	0.7660	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.2200	0.2450	0.2000	0.2200	0.1700	0.2200	0.2400	**0.2900**	0.0650
Zoo	0.6733	0.6733	0.6634	0.6139	0.6634	0.6634	0.7030	**0.9307**	**0.9307**
Tae	0.4106	**0.4172**	0.3907	0.3907	0.4106	0.3907	0.3841	0.3907	0.3907
Hayes-roth	0.3333	**0.3939**	0.3485	0.3333	0.3409	0.3333	0.3333	0.3788	0.3788
Dermatology	**0.7486**	0.6120	0.6230	0.7158	0.6311	0.5355	0.6311	0.6066	0.7240
Soybean	0.4978	0.4963	0.2972	0.3236	0.5476	**0.6223**	0.4656	0.4363	0.4026
Connect	0.2857	0.2813	0.2942	**0.3239**	0.2956	0.3196	0.1575	0.3168	0.3168
Chess	0.4884	0.4884	0.3733	0.4371	0.5116	0.3936	0.3914	**0.5169**	**0.5169**
Breast	0.9585	0.9585	0.9099	0.9113	0.8970	**0.9599**	0.9413	0.8941	0.8941
Car	0.2847	0.2772	0.3027	0.3304	0.3571	**0.3947**	0.2222	0.3524	0.3524
Mushroom	0.7090	0.7090	0.8838	0.8838	**0.8876**	0.7676	0.8776	**0.8876**	**0.8876**
Splice	0.2821	0.2853	0.3724	0.4204	0.6125	0.4646	**0.7028**	0.6191	0.6191
Vote	0.8690	0.8667	0.8644	0.8644	**0.8782**	0.8759	0.8736	0.8736	0.8736
Lymph	0.4122	0.4054	0.5135	0.4527	0.5405	**0.5811**	0.4797	0.4662	0.4730
Lung	0.4375	0.4375	0.4063	0.5625	**0.6563**	0.4063	0.3438	0.5938	0.5938
Average	0.5169	0.5129	0.5277	0.5344	0.5875	0.5580	0.5467	**0.5971**	0.5887

Table 4.7: NMI Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.7225	0.6834	**1.0000**	0.6634	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.4834	0.4672	0.5153	**0.5226**	0.4227	0.4834	0.4773	0.5140	0.3316
Zoo	0.8427	0.6839	0.7747	0.8086	0.8598	0.8525	0.8658	**0.8796**	**0.8796**
Tae	0.0834	0.0251	0.0994	0.0994	0.0834	**0.0994**	0.0740	0.0994	0.0994
Hayes-roth	0.0005	**0.0492**	0.0172	0.0031	0.0014	0.0000	0.0000	0.0083	0.0083
Dermatology	**0.8782**	0.7945	0.4911	0.6346	0.8423	0.7549	0.7674	0.8598	0.8286
Soybean	0.6800	0.6396	0.5901	0.6059	0.7358	0.7411	**0.7511**	0.7289	0.7262
Connect	0.0130	0.0117	0.0261	**0.0278**	0.0161	0.0203	0.0012	0.0161	0.0161
Chess	0.0003	0.0003	**0.0551**	0.0108	0.0008	0.0321	0.0454	0.0015	0.0015
Breast	0.7361	0.7361	0.5749	0.5750	0.5325	**0.7617**	0.6951	0.5244	0.5244
Car	0.0067	0.0046	0.0155	0.0285	0.1054	**0.1204**	0.0000	0.0693	0.0693
Mushroom	0.1963	0.1963	0.5350	0.5350	**0.5383**	0.2289	0.4930	**0.5383**	**0.5383**
Splice	0.0481	0.0466	0.0462	0.0343	0.2765	0.2670	**0.3825**	0.2984	0.2984
Vote	0.4692	0.4693	0.4393	0.4393	**0.5049**	0.4998	0.5010	0.4892	0.4892
Lymph	0.1673	0.1611	0.1587	0.1169	**0.2537**	0.1953	0.1203	0.2343	0.2334
Lung	0.1312	0.1735	0.2281	0.2665	**0.3488**	0.3226	0.1961	0.2471	0.2471
Average	0.3412	0.3214	0.3479	0.3357	**0.4077**	0.3987	0.3981	0.4068	0.3932

Table 4.8: AMI Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.6980	0.6554	**1.0000**	0.6322	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.2800	0.2614	0.3343	0.3385	0.2677	0.2800	0.2805	**0.3923**	0.2349
Zoo	0.8221	0.6427	0.7450	0.7837	0.8413	0.8331	0.8479	**0.8636**	**0.8636**
Tae	0.0699	0.0130	0.0860	0.0860	0.0699	**0.0860**	0.0573	0.0860	0.0860
Hayes-roth	-0.0139	**0.0350**	0.0027	-0.0118	-0.0131	-0.0145	-0.0145	-0.0061	-0.0061
Dermatology	**0.8756**	0.7900	0.4802	0.6268	0.8388	0.7491	0.7624	0.8567	0.8247
Soybean	0.6461	0.6021	0.5470	0.5646	0.7084	0.7141	**0.7256**	0.7004	0.6975
Connect	0.0128	0.0115	0.0259	**0.0276**	0.0159	0.0201	0.0010	0.0159	0.0159
Chess	0.0001	0.0001	**0.0549**	0.0105	0.0006	0.0319	0.0451	0.0013	0.0013
Breast	0.7358	0.7358	0.5744	0.5745	0.5320	**0.7614**	0.6947	0.5238	0.5238
Car	0.0043	0.0022	0.0131	0.0261	0.1033	**0.1183**	0.0000	0.0671	0.0671
Mushroom	0.1962	0.1962	0.5349	0.5349	**0.5382**	0.2288	0.4930	**0.5382**	**0.5382**
Splice	0.0475	0.0460	0.0456	0.0337	0.2761	0.2666	**0.3822**	0.2980	0.2980
Vote	0.4683	0.4684	0.4384	0.4384	**0.5040**	0.4990	0.5001	0.4884	0.4884
Lymph	0.1405	0.1341	0.1317	0.0884	**0.2297**	0.1698	0.0917	0.2097	0.2087
Lung	0.0698	0.1041	0.1712	0.2164	**0.2994**	0.2513	0.1176	0.1912	0.1912
Average	0.3158	0.2936	0.3241	0.3107	0.3883	0.3747	0.3740	**0.3891**	0.3771

Table 4.9: ARI Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.5634	0.5436	**1.0000**	0.6457	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.1103	0.1163	0.1493	0.1476	0.1048	0.1103	0.1130	**0.2727**	0.1654
Zoo	0.8345	0.6539	0.6994	0.6209	0.6973	0.6894	0.8258	**0.9008**	**0.9008**
Tae	0.0477	0.0134	**0.0497**	**0.0497**	0.0477	0.0497	0.0117	**0.0497**	**0.0497**
Hayes-roth	-0.0146	**0.0224**	-0.0048	-0.0132	-0.0135	-0.0149	-0.0149	-0.0043	-0.0043
Dermatology	0.7205	0.5995	0.5253	0.6489	0.7074	0.4933	0.6192	**0.7456**	0.7055
Soybean	0.4403	0.4211	0.3467	0.3148	0.4372	**0.4725**	0.4669	0.4299	0.4036
Connect	0.0191	0.0178	-0.0275	0.0003	0.0172	**0.0334**	-0.0035	0.0207	0.0207
Chess	0.0002	0.0002	**0.0636**	0.0152	0.0001	0.0449	0.0463	0.0007	0.0007
Breast	0.8391	0.8391	0.6647	0.6698	0.6216	**0.8450**	0.7779	0.6121	0.6121
Car	0.0043	-0.0029	0.0153	0.0423	0.0279	**0.0538**	0.0000	0.0500	0.0500
Mushroom	0.1744	0.1744	0.5891	0.5891	**0.6009**	0.2863	0.5704	**0.6009**	**0.6009**
Splice	0.0284	0.0271	0.0137	0.0275	0.1979	0.1840	**0.3668**	0.2081	0.2081
Vote	0.5435	0.5367	0.5298	0.5298	**0.5710**	0.5641	0.5572	0.5572	0.5572
Lymph	0.1157	0.1143	0.1310	0.0676	**0.2331**	0.1672	0.0745	0.1987	0.2022
Lung	0.0166	0.1127	0.1481	0.1860	**0.1933**	0.1479	0.0760	0.1578	0.1578
Average	0.2777	0.2619	0.3058	0.2839	0.3402	0.3204	0.3430	**0.3625**	0.3519

Table 4.10: Accuracy Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.8372	0.8298	**1.0000**	0.8622	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.8445	0.8373	0.8428	**0.8515**	0.7620	0.8445	0.8392	0.7896	0.6186
Zoo	0.9422	0.8822	0.9010	0.8792	0.9044	0.9018	0.9410	**0.9650**	**0.9650**
Tae	0.5078	**0.5639**	0.5073	0.5073	0.5078	0.5073	0.4285	0.5073	0.5073
Hayes-roth	0.5466	**0.5518**	0.5440	0.5371	0.5471	0.5465	0.5465	0.5507	0.5507
Dermatology	0.9108	0.8674	0.8527	0.8902	0.9048	0.8153	0.8802	**0.9158**	0.9027
Soybean	**0.9237**	0.9154	0.9080	0.9062	0.9101	0.9095	0.9036	0.9077	0.9006
Connect	0.4884	0.4885	0.4716	0.4796	0.4871	0.4965	**0.4998**	0.4894	0.4894
Chess	0.5001	0.5001	**0.5320**	0.5078	0.5001	0.5225	0.5234	0.5004	0.5004
Breast	0.9204	0.9204	0.8358	0.8381	0.8149	**0.9230**	0.8894	0.8104	0.8104
Car	0.4810	0.4779	0.4893	0.5042	0.4940	0.5071	**0.5425**	0.5064	0.5064
Mushroom	0.5873	0.5873	0.7946	0.7946	**0.8005**	0.6432	0.7852	**0.8005**	**0.8005**
Splice	0.5362	0.5355	0.5407	0.5511	0.6267	0.6176	**0.7049**	0.6310	0.6310
Vote	0.7717	0.7684	0.7650	0.7650	**0.7855**	0.7820	0.7786	0.7786	0.7786
Lymph	0.5682	0.5709	0.5782	0.5470	**0.6293**	0.5926	0.5472	0.6126	0.6143
Lung	0.5645	0.5685	0.6129	**0.6472**	0.6270	0.5222	0.4798	0.6210	0.6210
Average	0.6832	0.6791	0.6985	0.6918	0.7063	0.6957	0.7056	**0.7116**	0.6998

Table 4.11: Homogeneity Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.7200	0.6810	**1.0000**	0.6773	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.4343	0.4256	0.4739	0.4709	0.4538	0.4343	0.4362	**0.5620**	0.4996
Zoo	0.8385	0.6715	0.7458	0.7683	0.8105	0.8043	0.8422	**0.8768**	**0.8768**
Tae	0.0994	0.0251	**0.1200**	**0.1200**	0.0994	0.1200	0.1138	**0.1200**	**0.1200**
Hayes-roth	0.0005	**0.0500**	0.0172	0.0031	0.0013	0.0000	0.0000	0.0082	0.0082
Dermatology	0.8789	0.8062	0.4853	0.6296	0.8515	0.8023	0.7623	**0.8791**	0.8450
Soybean	0.6557	0.6283	0.5738	0.5860	0.7368	0.7554	**0.7745**	0.7299	0.7333
Connect	0.0110	0.0099	0.0226	**0.0235**	0.0135	0.0171	0.0012	0.0135	0.0135
Chess	0.0003	0.0003	**0.0603**	0.0115	0.0009	0.0327	0.0527	0.0016	0.0016
Breast	0.7399	0.7399	0.6055	0.6033	0.5661	**0.7521**	0.6819	0.5590	0.5590
Car	0.0054	0.0037	0.0126	0.0234	0.0849	0.0967	**1.0000**	0.0562	0.0562
Mushroom	0.2219	0.2219	0.5453	0.5453	**0.5472**	0.2300	0.4996	**0.5472**	**0.5472**
Splice	0.0487	0.0471	0.0452	0.0332	0.2706	0.2636	**0.3747**	0.2926	0.2926
Vote	0.4609	0.4607	0.4324	0.4324	**0.4958**	0.4907	0.4917	0.4804	0.4804
Lymph	0.1419	0.1317	0.1307	0.0965	**0.2052**	0.1705	0.1022	0.1898	0.1891
Lung	0.1330	0.1948	0.2361	0.2655	0.3669	**0.4523**	0.2858	0.2554	0.2554
Average	0.3369	0.3186	0.3442	0.3306	0.4065	0.4014	**0.4637**	0.4107	0.4049

Table 4.12: Precision Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.6792	0.6642	**1.0000**	0.7033	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.3170	0.2979	0.3416	**0.3809**	0.2140	0.3170	0.3007	0.3233	0.2217
Zoo	0.9019	0.7864	0.8716	0.8504	**0.9426**	0.9327	0.9399	0.9394	0.9394
Tae	0.3516	0.3381	0.3524	0.3524	0.3516	**0.3524**	0.3337	0.3524	0.3524
Hayes-roth	0.3353	**0.3593**	0.3421	0.3369	0.3360	0.3351	0.3351	0.3423	0.3423
Dermatology	**0.7755**	0.6515	0.6398	0.7360	0.7477	0.5260	0.7067	0.7621	0.7335
Soybean	**0.5853**	0.5129	0.4581	0.4400	0.4808	0.4806	0.4566	0.4681	0.4335
Connect	0.5815	0.5803	0.5489	0.5671	0.5802	**0.5921**	0.5651	0.5827	0.5827
Chess	0.5009	0.5009	**0.5270**	0.5073	0.5009	0.5222	0.5184	0.5012	0.5012
Breast	0.9223	0.9223	0.8161	0.8202	0.7944	**0.9427**	0.9180	0.7894	0.7894
Car	0.5469	0.5395	0.5567	0.5800	0.5705	**0.5978**	0.5425	0.5906	0.5906
Mushroom	0.5690	0.5690	0.7805	0.7805	**0.7880**	0.6417	0.7756	**0.7880**	**0.7880**
Splice	0.4016	0.4008	0.3939	0.4038	0.5163	0.5032	**0.6269**	0.5224	0.5224
Vote	0.7963	0.7933	0.7879	0.7879	**0.8103**	0.8070	0.8038	0.8034	0.8034
Lymph	0.5540	0.5704	0.5804	0.5238	**0.6927**	0.5877	0.5205	0.6577	0.6607
Lung	0.3314	0.3799	0.4108	**0.4487**	0.4337	0.3825	0.3522	0.4190	0.4190
Average	0.5719	0.5542	0.5880	0.5762	0.6100	0.5950	0.6060	**0.6151**	0.6050

Table 4.13: Recall Result

Dataset	***k*-means++**	***k*-means||**	***k*-modes**	**Cao**	***k*-reps**	***mk*-cens**	**MH-*k*-modes**	**LSH-*k*-reps(Init)**	**LSH-*k*-reps(Full)**
Soybean_small	0.6642	0.6494	**1.0000**	0.7786	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Audiology	0.1242	0.1449	0.1685	0.1471	0.2809	0.1242	0.1375	0.5007	**0.7188**
Zoo	0.8437	0.6788	0.6746	0.5845	0.6279	**0.9327**	0.7978	0.9082	0.9082
Tae	0.5868	0.3394	0.5932	0.5932	0.5868	0.3524	**0.7392**	0.5932	0.5932
Hayes-roth	0.3189	**0.3806**	0.3477	0.3518	0.3196	0.3351	0.3186	0.3273	0.3273
Dermatology	0.7768	0.7179	0.5946	0.6989	0.7877	0.5260	0.6809	**0.8389**	0.8031
Soybean	0.4068	0.4281	0.3479	0.3102	0.4923	0.4806	**0.6019**	0.4936	0.4858
Connect	0.3477	0.3529	0.3811	0.3461	0.3442	**0.5921**	0.5103	0.3498	0.3498
Chess	0.5003	0.5003	0.6383	0.5933	0.5245	0.5222	**0.6823**	0.5280	0.5280
Breast	0.9331	0.9331	0.9036	0.9021	0.8932	**0.9427**	0.8763	0.8916	0.8916
Car	0.2521	0.2560	0.2875	0.3115	0.2722	0.5978	**1.0000**	0.2937	0.2937
Mushroom	0.7240	0.7240	0.8204	0.8204	**0.8227**	0.6417	0.8034	**0.8227**	**0.8227**
Splice	0.4195	0.4191	0.3595	0.3500	0.4697	0.5032	**0.5751**	0.4783	0.4783
Vote	0.7594	0.7555	0.7556	0.7556	0.7720	**0.8070**	0.7648	0.7654	0.7654
Lymph	0.3841	0.3264	0.3476	0.3230	0.3699	**0.5877**	0.3784	0.3545	0.3567
Lung	0.3522	0.5472	0.4780	0.4403	0.5346	0.3825	**0.7421**	0.4717	0.4717
Average	0.5246	0.5096	0.5436	0.5192	0.5686	0.5830	**0.6630**	0.6011	0.6122

4.7.3.4 Cluster initialization comparisons

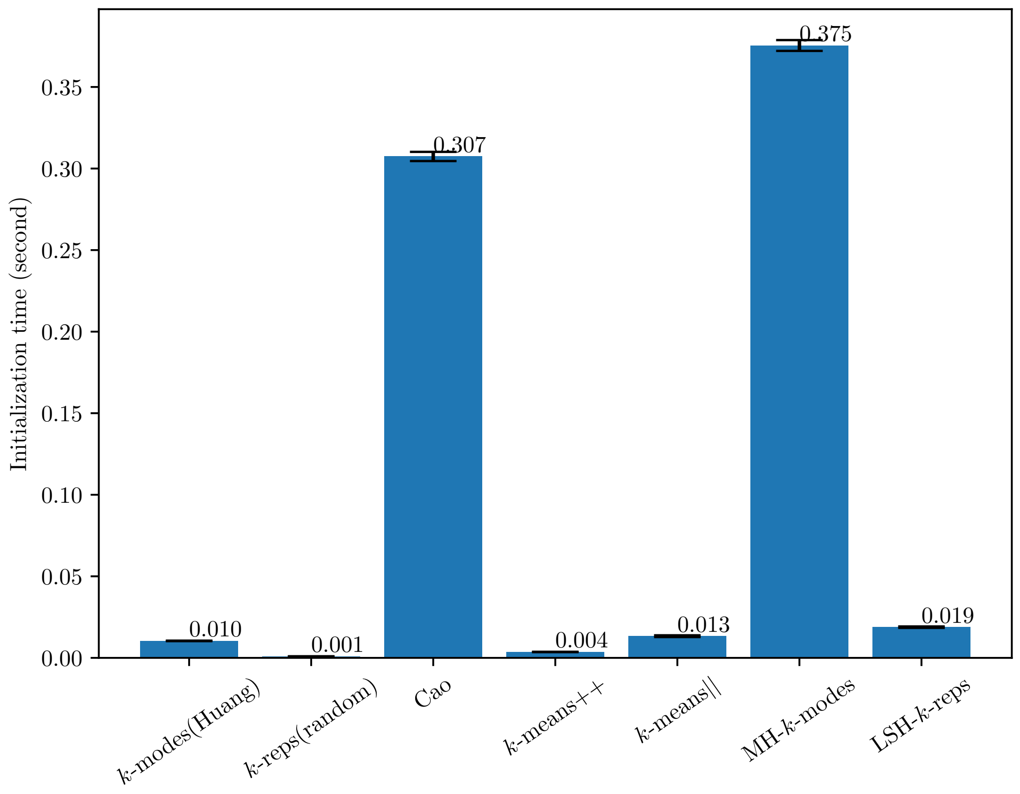

Figure 4.9: Average cluster initialization time of different approaches on Soybean dataset

Figure 4.9 shows the average standard deviation for initialization times of our proposed method together with competitors. Cao's method and MH-k-modes take significant time to initialize because their processes have numerous looking-up procedures. On the other hand, our proposed method is slower than other related initialization methods because the finding of maximum cuts on categorical graphs takes majorities of time. However, the total time of the initialization stage is extremely small compared to the whole clustering time. For instance, the total clustering of LSH-k-representatives(Init) may take around 0.55 seconds for the whole clustering process on the Soybean dataset, while the initialization stage only takes 0.019 seconds (3.45%).

4.7.3.5 Convergence results

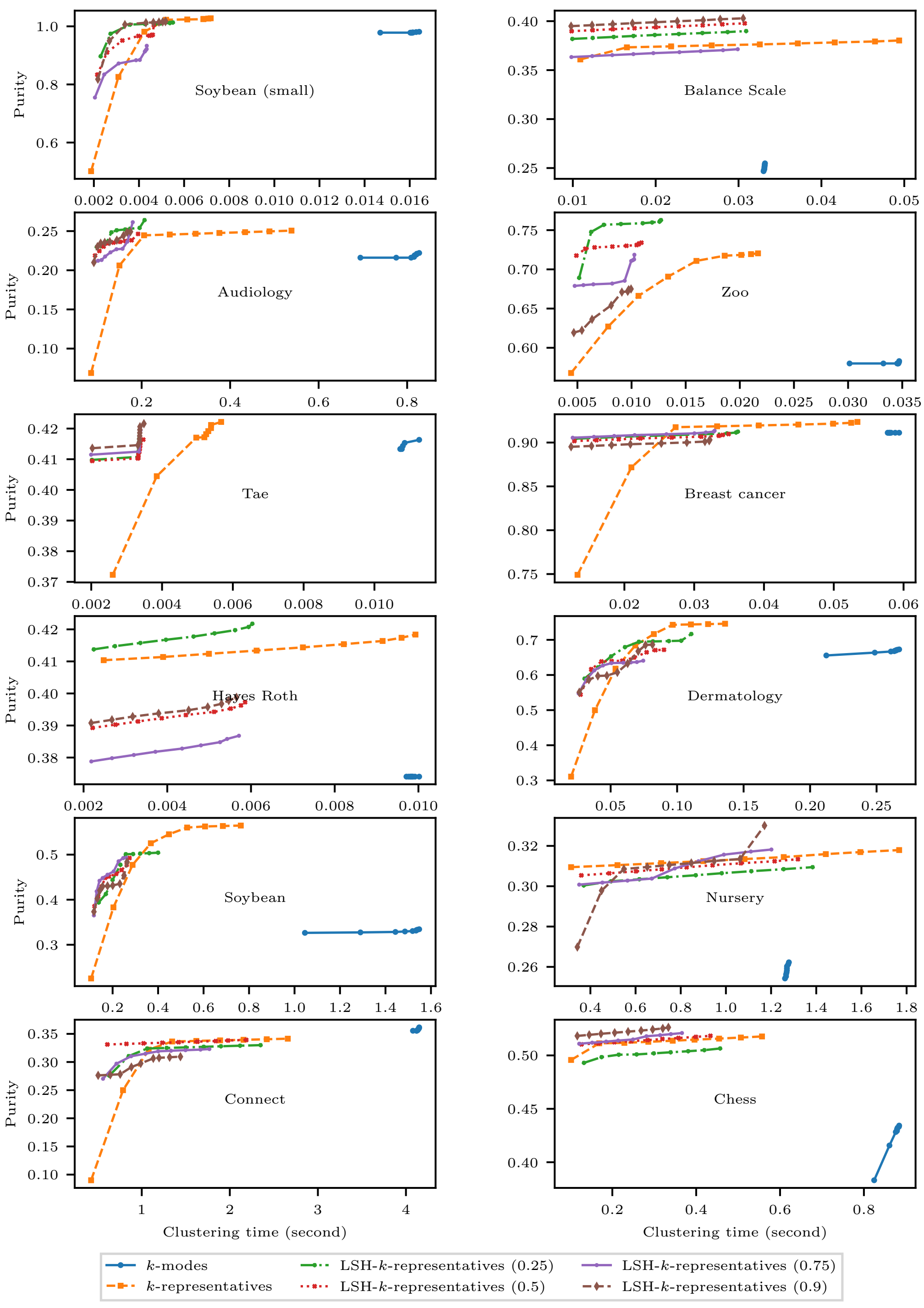

Figure 4.10: Purity versus convergence times on common UCI datasets

Figure 4.10 shows the results on the degree of convergence of LSH-k-prototypes compared to other common clustering algorithms of categorical data. It is worth nothing that our method tends to have higher accuracy than other methods. In addition, k-modes can converge in few iterations but their results are the worst. Besides that, k-representatives can have comparable accuracy with our method but they start at much less favorable points, which makes our method can converge faster than k-representatives.

4.7.3.6 Results on synthetic datasets

The experiments on synthetic datasets is mainly designed to examine the scalability of our method versus other related works.

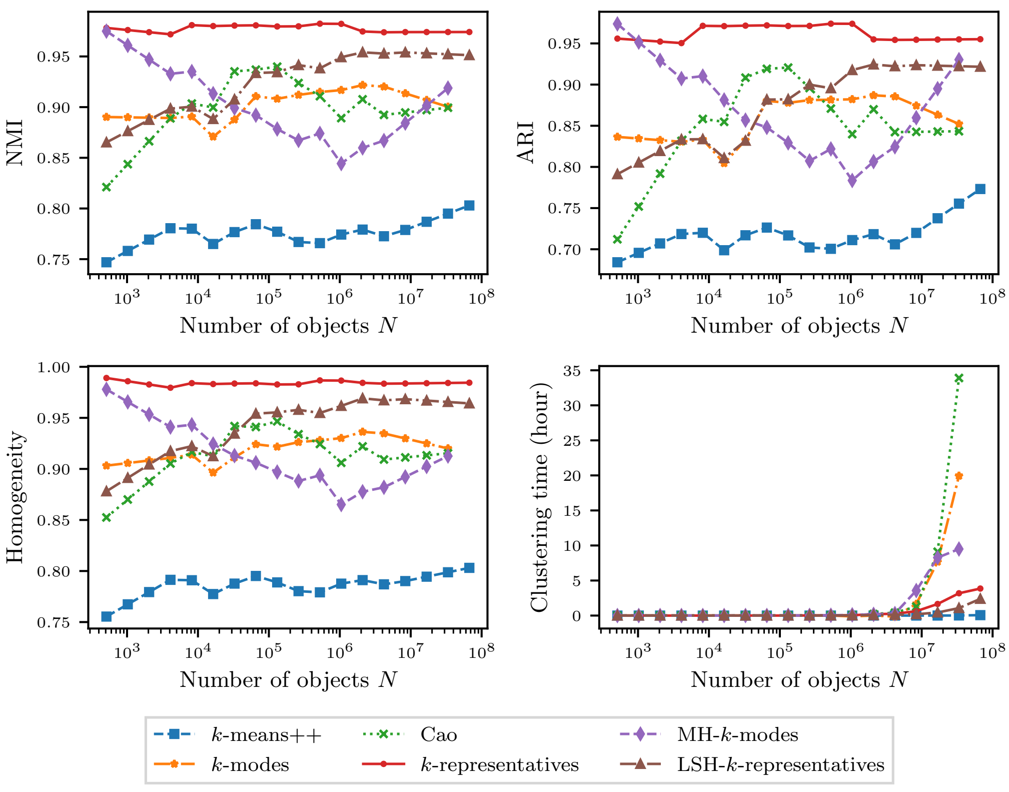

Figure 4.11: Performances of LSH-k-representatives related works on synthetic datasets with different numbers of objects

Figure 4.11 shows the results on the clustering effectiveness of the compared methods when clustering extremely large datasets. k-means can run fastest among all compare methods because it treats the categorical values by floating-point numbers; however, its accuracy is the worst because the encoding processing losses the information of categorical objects. In contrast, Cao et al.'s method is the slowest one while its accuracy is relatively low. k-representatives is the most accurate method, which is slightly higher than our LSH-k-representatives. However, LSH-k-representatives is almost three times faster than k-representatives for the dataset of 268 million categorical objects.

4.7.3.7 Time complexity analysis for datasets of a high number of clusters

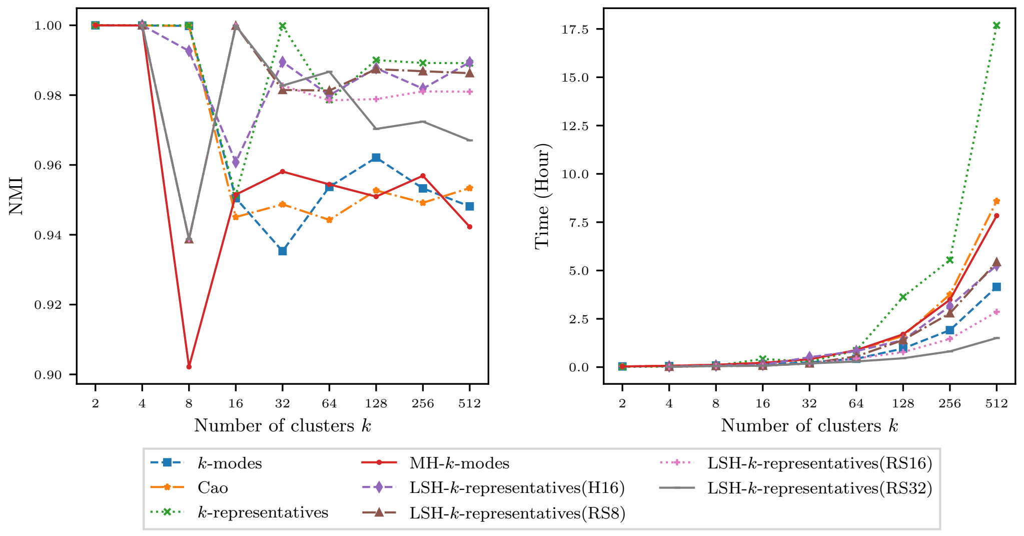

Figure 4.12: NMI score and clustering time for synthetic dataset of 1 million objects, D=16, $|\bar{A}| \approx 16$.

First, Figure 4.12 shows the results of algorithm complexity when clustering datasets of high number of clusters k. Note that, LSH-k-representatives(H16) is the LSH-k-representatives method with $k' = k/16$, which can run the fastest when comparing with other related works in most of the test cases. Moreover, the three modifications of LSH-k-representatives so-called LSH-k-representatives(RS8), LSH-k-representatives(RS16), and LSH-k-representatives(RS32) which use the rotation of shortlists with $k' = k/8, k/16$, and $k/32$, respectively can outperform LSH-k-representatives with the test case of extremely large number of clusters k (512) with slightly lower NMI scores.

Table 4.14: Performance result for USCensus dataset with k=512

Method	**#Iter**	**NMI**	**AMI**	**Hom..**	**Time(s)**	**Time/Iter**	**%NMI**	**%AMI**	**%Hom..**	**Speed-up**
***k*-Modes**	100	0.65	0.64	0.66	81,717	817.17	0.77	0.76	0.78	7.51
Cao	100	0.66	0.65	0.67	147,291	1472.91	0.78	0.77	0.79	4.17
MK-*k*-modes	100	0.65	0.64	0.66	143,711	1,437.11	0.77	0.76	0.78	4.27
***k*-Repsresentatives**	100	0.84	0.84	0.85	613,558	6,135.58	**1.00**	**1.00**	**1.00**	1.00
LSH-*k*-representatives(H16)	100	0.79	0.79	0.89	85,217	852.17	0.94	0.93	**1.00**	7.20
LSH-*k*-representatives(RS32)	100	0.79	0.78	0.84	19,061	**190.61**	0.93	0.93	0.99	**32.19**
LSH-*k*-repss(RS32-CUDA)	100	0.79	0.78	0.84	636	6.36	0.93	0.93	0.99	964.71

Second, Table 4.14 shows the time complexity compassion results when clustering the real-world census dataset. Because this dataset has no labels, we select the best outcome of k-representatives (the most accurate related method) from 50 different clustering times and call it a benchmark. As a result, LSH-k-representatives(H16) can have comparable accuracy with the benchmark but 7 times faster. Surprisingly, LSH-k-representatives(RS32) can run 32 times faster than the benchmark with a little less accuracy. Additionally, we also implement LSH-k-representatives(RS32) using CUDA programming language and conduct the clustering process using an NVidia P100 GPGPU. As the result, the CUDA version of LSH-k-representatives(RS32) can run 964 times faster than the benchmark.

4.7.3.8 Stability analysis and comparison

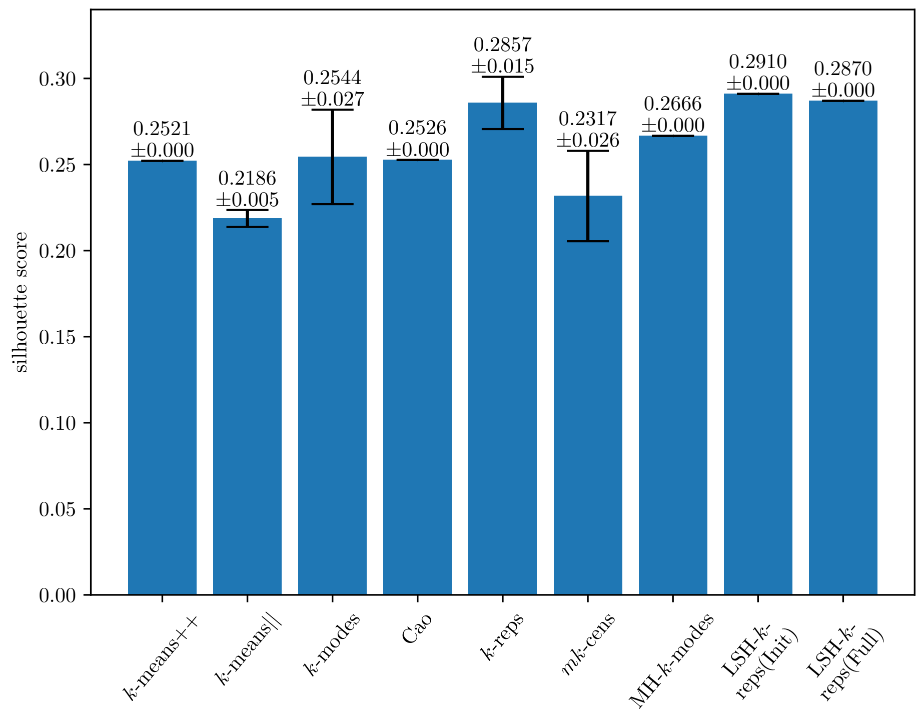

Figure 4.13: The average silhouette scores and standard deviations of compared methods

Figure 4.13 shows the averages and standard deviations of silhouette scores of the compared methods in 16 UCI testing datasets, each method is run 64 times for each dataset. As the results, LSH-k-reps(Init) and LSH-k-reps(Full) are the best and the second-best methods with the average silhouette scores of 0.2910 and 0.2870, respectively. Moreover, k-means++, Cao, MH-k-modes, LSH-k-reps(Init), and LSH-k-reps(Full) are the methods with the highest stability score.

Because of the principle of our cluster prediction technique (See **Algorithm 4.2**); First, the DILCA measure gives a fixed dissimilarity matrix for each domain. Which leads to the consistent dissimilarity graph for that domain. Second, The Stoer-Wagner algorithm also gives the consistent maximum cut for an undirected graph. Third, for each value of the number of hash functions l, the core buckets and the way we merge the small buckets are also fixed. Which makes our method become one of the most stable methods. The reason is our method includes several processes which can produce plenty of solutions but we only select the best outcome for each process. To make it generate different solutions for the same parameters, there are several ways that we can tune the original version as:

- Turning the context set size for DILCA measure, which can give different dissimilarity matrices.
- Instead of selecting the maximum cut, we can select other cuts that also well "divide" the domain.
- Selecting different number of hash function l (The effectiveness turning l is shown in Figure 4.8).
- Randomly merging a small bucket to a core bucket when the Hamming distances of the small bucket to two or more core buckets are the same.

4.7.3.9 Space complexity analysis

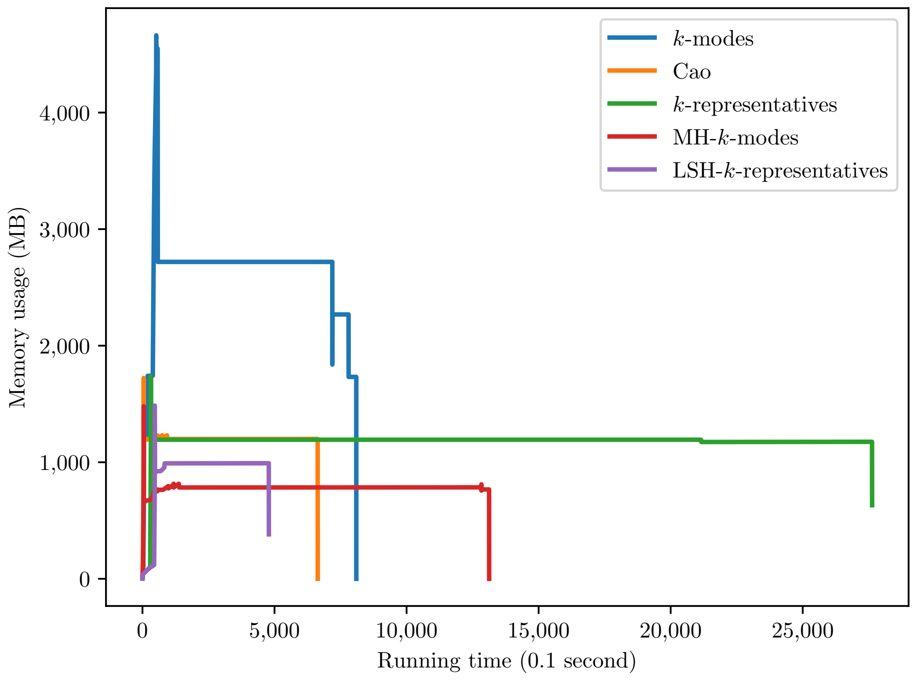

Figure 4.14: Memory usage of clustering methods over time for the USCensus dataset

Last but not least, Figure 4.14 shows the runtime memory usages of clustering methods on the USCensus dataset. Generally, all the methods require a significant amount of memory usages during the initialization process for predicting the initial clusters. It is interesting to note here that our LSH-k-representatives is the fastest method with using a small amount of memory even on the initialization process.

4.7.4 Results on mixed data

In our LSH-k-prototypes method, the hash functions on categorical attributes have higher priorities than the hash functions on numerical categorical attributes. If the number of hash functions on categorical attributes is sufficient, only these hash functions are used. In this section, we analyze the effectiveness of our hash functions for mixed data.

4.7.4.1 Weighting analysis

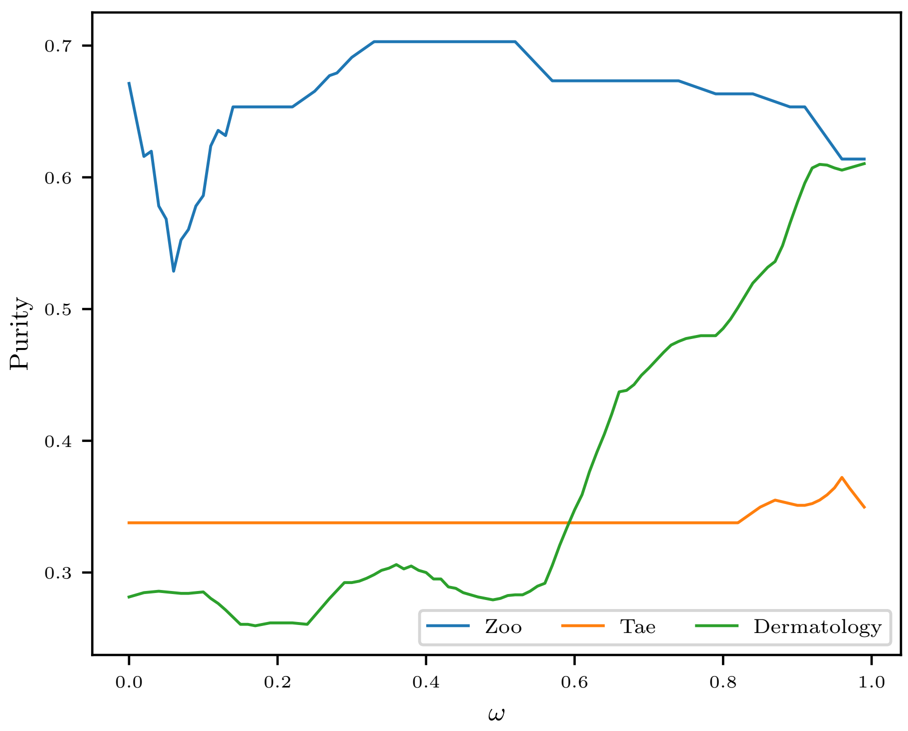

Figure 4.15: Impact of ω on the accuracy of LSH-k-prototypes

Figure 4.15 shows the impact of the smoothing parameter ω in three mixed datasets. It is clear that different datasets have different optimal values for the parameter ω. For instance, we should use ω in the range of 0.35 to 0.51 for clustering Zoo dataset, the optimal value of ω for Tae dataset is 0.96, and the categorical attributes are dominating numerical attributes in the Dermatology dataset.

4.7.4.2 Comparison

Table 4.15: Purity scores on mixed datasets

Dataset	***k*-prototypes**	***k*-imPrototypes**	**LSH*k*-prototypes**
Zoo	0.644±0.00	0.663±0.11	**0.703±0.00**
Tae	0.358±0.00	**0.367±0.03**	0.338±0.00
Dermatology	0.243±0.00	0.266±0.03	**0.281±0.00**
Average	0.415±0.00	0.432±0.06	**0.441±0.00**

Table 4.15 shows the comparisons of our proposed with the mixed data clustering algorithms, each method clusters each dataset 64 times with different random seed numbers. As an evidence, *improved prototype* is superior than *prototype* just same as the results between *representative* and *mode*. Besides that, our proposed method LSH-k-prototypes can enhance the effectiveness of k-imPrototypes by applying the LSH-based cluster prediction technique. Moreover, our method has extremely high stability compared to the original method.

4.8 Summary

In this chapter, we first propose LSH-k-prototypes for clustering mixed datasets with the utilization of LSH to predict the potential natural clusters. The shortlists of neighbor clusters are also used to reduce the required computations in each iteration. Furthermore, LSH-k-representatives is the modification of LSH-k-prototypes, which works for categorical values only. The advantages, disadvantages, and future works of these two methods can be listed as:

4.8.1 Advantages

- LSH-k-prototypes and LSH-k-representatives use the state-of-the-art cluster representations so-called *representative* and *prototype*, respectively, for representing the clusters.

- LSH-k-prototypes and LSH-k-representatives can predict the initial clusters based on the locality-sensitive factors of objects. In detail, the larger LSH buckets are predicted as the core initial clusters. As a result, LSH-k-prototypes and LSH-k-representatives have the highest clustering effectiveness scores among other related works.
- By applying the dynamic shortlists for clusters, LSH-k-representatives can increase the clustering speed up to from 2 to 32 times compared to its original method.

4.8.2 Limitations

- Because our proposed methods come with the process of building the LSH hash table for all attributes, the total time for predicting initial clusters of our methods is higher than others. Especially, when creating the hash function for the attribute with a massive number of unique categorical values, our methods take more time to calculate the dissimilarity matrix and find the maximum cut.
- In the context of our LSH-based cluster prediction technique, the small buckets are merged into the core bucket that is closest to the small buckets. However, in the current version, when two or more core buckets have the same distance to a small bucket, this small bucket is merged to the first core bucket (one with the lowest index). This approach may reduce the flexibility of our proposed methods.

4.8.3 Future works

- Because creating the hash function for a categorical attribute with a massive number of unique categorical values is time-consuming, we first suggest using different optimal dissimilarity measures instead of DILCA to reduce the matrix creating time. In addition, the task of finding the maximum cut is also costly, we recommend using other partitioning techniques to subdivide these categorical domains into multiple two-subsets; for example, a k-means-like algorithm can be utilized for this task.
- In this current version of the LSH-based cluster prediction method, the single domain of an attribute is used to create an LSH hash function. Future research may try a different kind of LSH function; for instance, a hyperplane hash function for categorical values that consider the distribution of multiple attributes at the same time can be analyzed.
- In this chapter, we introduce two different approaches to update the cluster shortlist in real-time. Among them, the shortlist rotation

approach may work with the datasets with a huge number of clusters k but other heuristic techniques may update the shortlist more effectively.

Chapter 5

Extension for fuzzy clustering of categorical data

Continuing the work of LSH-k-prototypes, in this chapter, we apply the proposed method to fuzzy clustering of categorical data.

5.1 Introduction

The fundamental of fuzzy clustering is not much different from that of crisp clustering; however, the membership degrees of objects laying near the cluster boundaries are fuzzed to present their multi-cluster factors [91]. Accordingly, the formulation of membership on fuzzy clustering is also slightly changed compared with the definitions of crisp clustering (section 2.1). The problem of fuzzy clustering can be remarked as follows.

$X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ denotes the dataset of N categorical objects to be fuzzy clustered into k fuzzy clusters. Each object $\boldsymbol{x}_i$ $(1 \leq i \leq N)$ is a vector of D categorical values of D categorical attributes: $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{iD}]$. Each j-th value of an object $\boldsymbol{x}_i$ is a categorical value $(x_{ij} \in \mathbb{A}_j)$, where $\mathbb{A}_j$ $(|\mathbb{A}_j| > 1)$ is the set of all possible categorical values in the j-th attribute. Therefore, $\mathbb{A}_j$ is the unique domain of the attribute A_j.

In the context of fuzzy clustering, all objects in the dataset X belong to all clusters simultaneously but with different membership degrees. Let u_{ij} is the membership degree of the i-th object to the j-th cluster such that the higher value of u_{ij} means the higher interrelationship of the i-th object to the j-th cluster. The value of u_{ij} can be normalized as:

$$0 \leq u_{ij} \leq 1 \text{ and } \sum_{j=1}^{k} u_{ij} = 1, \quad 1 \leq i \leq N \tag{5.1}$$

To generalize, we denote $U = [u_{ij}]_{N \times k}$ as the membership matrix that can present membership statuses of N objects in dataset X to k target clusters.

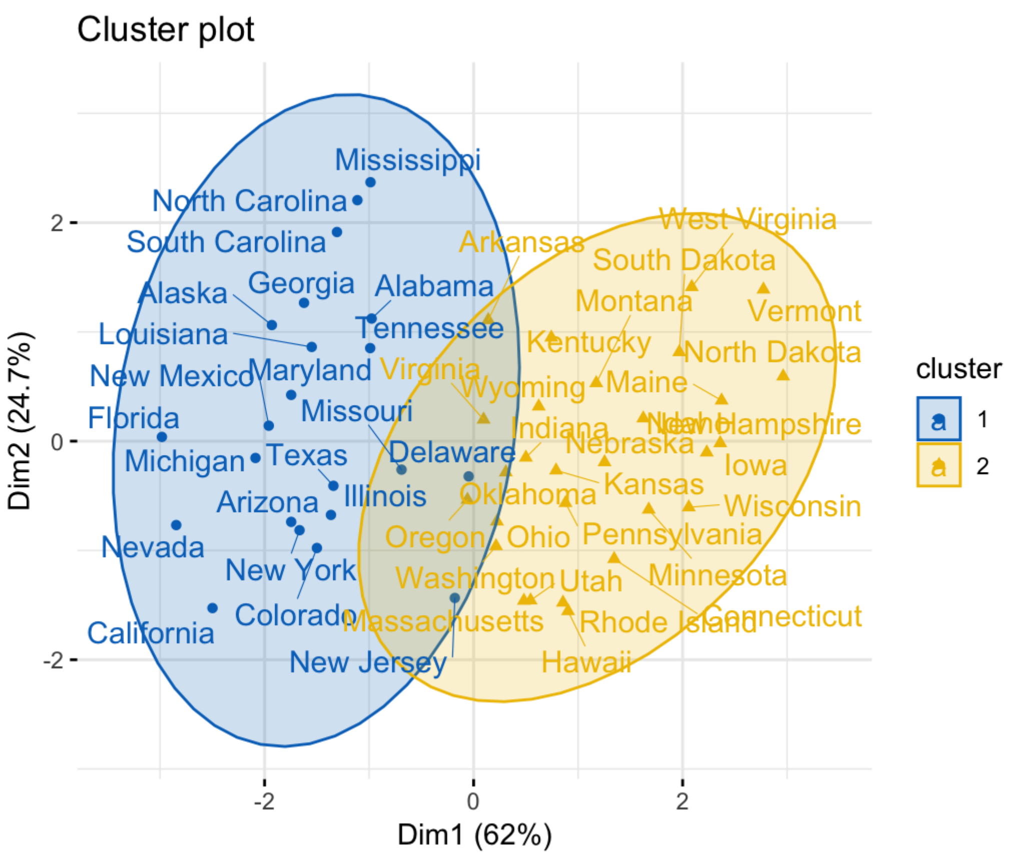

Figure 5.1: Example of fuzzy clustering for USArrests dataset[1]

Figure 5.1 shows an example of fuzzy clustering with 2-dimensional data for the USArrests dataset, which shows the violent crime rates by US states. In this example, the states are divided into two fuzzy clusters in which some states have the same degree of similarity with two fuzzy clusters, namely Virginia, Delaware, Arkansas, Oregon, and New Jersey.

5.2 k-means-like algorithm for fuzzy clustering

Fuzzy k-means (Fk-means for short) is the first fuzzy cluster analysis approach for data of numerical values [92]. We ought to introduce Fk-means because it is the basis for all other k-means-like algorithms for fuzzy clustering.

Same as k-means, the center of gravity (centroid) is used to represent

[1]Image source: https://www.datanovia.com/en/lessons/fuzzy-clustering-essentials/

cluster. $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k\}$ denotes the set of k centroids, each centroid $\boldsymbol{c}_j$ is a vector in the same space with the data objects. The membership degrees of an object for all clusters are calculated by the reciprocal value of the distances of this object to all clusters:

$$u_{ij} = \left(\sum_{j'=1}^{k} \left(\frac{\text{Dis}(\boldsymbol{x}_i, \boldsymbol{c}_j)}{\text{Dis}(\boldsymbol{x}_i, \boldsymbol{c}_{j'})} \right)^{\frac{1}{\alpha-1}} \right)^{-1} , 1 \leq i \leq N, 1 \leq j \leq k \tag{5.2}$$

with α being the parameter that controls the degree of fuzziness of the clustering model, such that $\alpha \in (1, +\infty)$. The higher the value of α, the higher the fuzziness of the clustering model.

To update the centroids after the membership matrix is obtained, the fuzzy-weighted average of all objects is used:

$$\boldsymbol{c}_j = \frac{\sum_{i=1}^{N} u_{ij}^{\alpha} \boldsymbol{x}_i}{\sum_{i=1}^{N} u_{ij}^{\alpha}} \quad , 1 \leq j \leq k \tag{5.3}$$

Note that, the fuzziness parameter α is also used as the weighting factor for every membership value.

Last but not least, the total fuzzy-weighted distance from all objects to all cluster centroids is used as the objective of the algorithm:

$$\text{Minimize: } \text{O}(U, C) = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{ij}^{\alpha} \text{Dis}(\boldsymbol{x}_i, \boldsymbol{c}_j) \tag{5.4}$$

The above three equations are the foundation principles of Fk-means, the algorithm of Fk-means can be briefly described in the following four steps:

- *Step 1:* Randomly selecting k categorical objects and set them the centroids for k clusters.
- *Step 2:* Calculating/recalculating the membership matrix U by equation (5.2).
- *Step 3:* Recalculating the centroids C by the equation (5.3).
- *Step 4:* Repeating *Step 2* and *Step 3* until value $\text{O}(U, C)$ (see equation (5.4)) is converged or the number of the maximum iterations is reached.

5.3 Fuzzy clustering algorithms for categorical data

Some fuzzy clustering algorithms of the class of k-means-like algorithms for categorical data can be seen in [33, 54, 92, 93]. In the following sections, we remark some typical algorithms for fuzzy clustering of categorical data:

5.3.1 Fuzzy k-modes (Fk-modes)

Fk-modes is the first extension of k-modes for fuzzy clustering. To represent a cluster center, Fk-modes also use the categorical vector (*mode*) of categorical values with the highest frequencies on all attributes. Let $\boldsymbol{m}_j = [m_{j1}, \ldots, m_{jD}]$ be the *mode* of the j-th cluster, m_{jd} can be computed as:

$$m_{jd} = \arg\max_{v \in \mathbb{A}_d} \sum_{\substack{i=1 \\ x_{id}=v}}^{N} u_{ij} \quad , 1 \leq j \leq k, 1 \leq d \leq D \tag{5.5}$$

Because *modes* are categorical vectors and dissimilarity measure between categorical vectors is already defined (see equation (2.6)), we can use equation (5.2) to update the membership matrix.

5.3.2 Fuzzy k-representatives

In terms of extension of k-representatives for fuzzy clustering of categorical data, Kim et al. proposed to use the fuzzy centroids which are expansions of *representatives* [55]. It is clear that the *representative* takes the frequencies of all categorical values for all clusters, but in the context of fuzzy membership, an object belongs to all clusters, which makes the probability function (see equation (2.21)) unable to be used in the context of fuzzy membership. The adaptation which takes into account the membership degree can be utilized instead:

$$\mathrm{FPr}_{j,d}(v) = \frac{\sum_{\boldsymbol{x}_i \in X, x_{id}=v} u_{ij}^{\alpha}}{\sum_{i=1}^{N} u_{ij}^{\alpha}} \tag{5.6}$$

5.3.3 k-centers

Continuing the work of k-representatives, Chen et al. introduced to use a kernel combination of uniform distribution and observed distribution to have better representation of cluster [89]. In particular, a concept of $\mathrm{Pr}_{j,d}(v)$ in equation (2.21) is replaced by a new concept of kernel probability $\mathrm{KPr}_{j,d}(v)$, which is defined as follows.

$$\mathrm{KPr}_{j,d}(v) = \lambda_j \frac{1}{|\mathbb{A}_d|} + (1 - \lambda_j)\mathrm{Pr}_{j,d}(v), \forall v \in \mathbb{A}_d \tag{5.7}$$

where λ_j is the smoothing parameter for adjusting the contribution of uniform distribution to the j-th *center*. The optimal value of λ_j can be learned statistically based on the distribution of categorical values on that attribute (Least Squares Cross-Validation (LSCV)) [94]:

$$\lambda_j = \frac{1}{|G_j| - 1} \frac{\sum_{d=1}^{D}(1 - \sum_{v\in\mathbb{A}_d} \text{Frequency}_{j,d}(v)^2)}{\sum_{d=1}^{D}(\sum_{v\in\mathbb{A}_d} \text{Frequency}_{j,d}(v)^2 - \frac{1}{|\mathbb{A}_d|})} \tag{5.8}$$

where $\text{Frequency}_{j,d}(v)$ is the frequency of appearance of categorical value v in the j-th attribute with $v \in \mathbb{A}_d$.

5.4 The proposed method: Fuzzy$k$-centers (Fk-centers)

We extend the *center* into *fcenter* for fuzzy clustering of categorical data. Similar to probability function, the way to calculate the frequency of categorical attribute also needs to be changed to correspond to the concept of fuzzy clustering: $\text{FFrequency}_j(v)$ returns the frequency of the categorical value v in the j-th fuzzy cluster:

$$\text{FFrequency}_{j,d}(v) = \sum_{\boldsymbol{x}_i\in X, x_{id}=v} u_{ij}^{\alpha}, \quad 1 \le d \le D, 1 \le j \le k \tag{5.9}$$

Accordingly, the smoothing parameter in equation (5.8) is modified to match the fuzzy concept:

$$\lambda_j = \frac{1}{(\sum_{i=1}^{N} u_{ij}) - 1} \times \frac{\sum_{d=1}^{D}(1 - \sum_{v\in\mathcal{A}_d} \text{FFrequency}_{j,d}(v)^2)}{\sum_{d=1}^{D}(\sum_{v\in\mathcal{A}_d} \text{FFrequency}_{j,d}(v)^2 - \frac{1}{|\mathbb{A}_d|})} \tag{5.10}$$

It is clear that the components of *fcenters* are the same as *centers*, so the dissimilarity measures in equations (2.7) and (2.22) can be reused.

As a result, $\boldsymbol{c}_j$ $(1 \le j \le k)$ denotes the *fcenter* of the j-th cluster where $\boldsymbol{c}_j = [\boldsymbol{c}_{j1}, \dots, \boldsymbol{c}_{jD}]$, where

$$\boldsymbol{c}_{jd} = [\{v, \text{KPro}_{jd}(v)\} | \forall v \in \mathbb{A}_d], \quad 1 \le d \le D, 1 \le j \le k \tag{5.11}$$

5.5 The proposed method: LSHFk-centers (LSHFk-centers)

In this section, we extend our proposed initial cluster prediction method to estimate the initial fuzzy clusters. In general, the main steps for cluster prediction of LSHFuzzy$k$-centers are equivalent to the steps of LSH-k-prototypes. However, some concepts have to be changed to be compatible with fuzzy clustering. In detail, the following steps are identical with the steps of LSH-k-prototypes: Creating dissimilarity matrices for categorical data (see section 4.2.1) and LSH hash table generation (see section 4.2.2). Because the step of cluster prediction is different, we present the cluster prediction of LSHFuzzy$k$-centers as follows:

5.5.1 LSHFk-centers's cluster prediction

Note that, until this step, the hash table that stores the hash values of all categorical objects in X is known. Remark that, $\mathcal{H} = [B_0, \dots, B_{2^l-1}]$ is the hash table of 2^l buckets, in which $\{B_1^*, \dots, B_k^*\} \subset \{B_0, \dots, B_{2^l-1}\}$ is the set of k largest buckets.

The largest buckets are potentially located in or near the natural clusters in the dataset because they have dense objects with the same locality-sensitive scores. We recommend to set fully the degree of membership of objects in the k largest buckets to the k corresponding clusters:

$$\forall \boldsymbol{x}_i \in \{B_1^*, \dots, B_k^*\},\ u_{ij} = \begin{cases} 1, & \text{if } \boldsymbol{x}_i \in B_j^* \\ 0, & \text{Otherwise} \end{cases} ,1 \le j \le k \tag{5.12}$$

The objects in the remaining buckets share their degrees of membership to k initial clusters based on the distances of them to the k largest buckets, they can be approximated by the Hamming distances between corresponding bucket keys:

$$\text{Dis}(\boldsymbol{x}_i, \boldsymbol{c}_j) \approx \hat{\text{Dis}}(\boldsymbol{x}_i, B_j^*) = \text{Hamming}(\text{Key}(B_{i'}), \text{Key}(B_j^*)) \tag{5.13}$$

where $\text{Key}(B_{i'})$ is the bucket that holds categorical object $\boldsymbol{x}_i$.

When the distances between remaining objects to all cluster centers are approximated, we can simply apply equation (5.2) to calculate the degrees of membership for those objects.

Algorithm 5.1 LSH-based initial cluster prediction algorithm

```
    Input: $X$, $k$, $D$, $\mathbb{A}_d$ $(1 \leq d \leq D)$, $l$
    Output: $k$ initial clusters
 1: Create dissimilarity matrix for each categorical attribute $\mathbb{A}_d (1 \leq d \leq D)$
    using DILCA dissimilarity measure.
 2: Convert dissimilarity matrices into dissimilarity graphs.
 3: Find the maximum cut for each dissimilarity graph using Stoer-Wagner
    algorithm.
 4: Generate the hash function via equation (4.2). The hash values of all
    objects in $X$ then can be obtained. Correspondingly, the hash table
    $\mathcal{H} = [B_1, \ldots, B_{2^l}]$ also can be established.
 5: Select $k$ largest buckets $\{B_1^*, \ldots, B_k^*\}$.
 6: for $\boldsymbol{x}_i \in X$ do
 7:     if $\boldsymbol{x}_i \in \{B_1^*, \ldots, B_k^*\}$ then
 8:         Assign degrees of membership $u_{ij} (1 \leq j \leq k)$ by equation (5.12).
 9:     else
10:         Estimate distance from object $\boldsymbol{x}_i$ to the all core buckets by equation
            (5.13).
11:         Assign degrees of membership $u_{ij} (1 \leq j \leq k)$ by equation (5.2).
12:     end if
13: end for
14: return $U = [u_{ij}]_{N \times k}$
```

Algorithm 5.1 summaries the processes of predicting the fuzzy clusters for LSHFk-centers method. Note that, the algorithm returns the values for the membership matrix U.

5.5.2 LSHFuzzy$k$-centers algorithm

Algorithm 5.2 LSHFk-centers algorithm

```
Input: $X$, $k$, $D$, $\mathbb{A}_d (1 \leq d \leq D)$, $l$ , iter_max, $\epsilon$
Output: $k$ centers of $k$ clusters $R$ and membership matrix $U$ that can
  locality minimize the value $\mathrm{O}(U, C)$ (equation (5.4)).
1: Find the initial membership matrix $U$ by Algorithm 5.1.
2: Set $\lambda_j \leftarrow 0$ for $1 \leq j \leq k$.
3: for iter $\leftarrow 1$ ; iter $\leq$ iter_max; iter++ do
4:     Update the centers $C$ follow equation (5.11) with the fuzzy probability
       in equation (5.7).
5:     Update the membership matrix $U$ follows equation (5.2).
6:     Update the smoothing parameter $\lambda_j$ for $1 \leq j \leq k$ follows equation
       (5.10).
7:     if $\Delta \mathrm{O}(U, C) \leq \epsilon$ then
8:         Break the loop.
9:     end if
10: end for
11: return $C, U$
```

Algorithm 5.2 summaries the mechanisms of our LSHFk-centers algorithm.

5.6 Experiments and results

5.6.1 Datasets and testing environments

16 UCI categorical datasets were obtained from UCI repository [86]. These datasets are widely used by many studies in classification and clustering with categorical data or mixed data. The characteristics of these datasets are described in Table 5.1.

Table 5.1: Common categorical datasets.

Name	**#Items**	**#Attributes**	**#Classes**
Soybean-small	47	21	4
Audiology	226	21	4
Balance-scale	625	4	3
Zoo	101	17	7
Hayes-roth	132	5	3
Dermatology	366	34	6
Soybean	683	35	19
Connect	10,000	42	3
Chess	3,196	36	2
Breast	699	9	2
Car	1,728	6	4
Mushroom	8,124	22	2
Tictactoe	958	9	2
Vote	435	16	2
Flare	1,066	12	3
Lung	32	56	3

The experiment conditions and environments of this chapter are similar to them in Chapter 4.

5.6.2 Evaluation metrics

The performance metrics are divided into two groups: complexity and effectiveness. For complexity analysis, the following space and time measurements were used:

- Time complexity: Including the total running time of compared methods together with their preprocessing time.
- Space complexity: Referring to the computing space required for clustering. Especially, the memory usage over time of compared methods was analyzed.

For effectiveness analysis, we mainly use the Fuzzy-Silhouette (FSilhouette) score to evaluate the fuzzy clustering results:

- Fuzzy-Silhouette (FSilhouette) score [95]: This score is the extension of silhouette score for evaluating the accuracy of fuzzy clustering outcome [95]. The FSilhouette metric uses the pairwise degree of memberships along with their distances:

$$\text{FSilhouette}(U, X) = \frac{\sum_{i=1}^{N} (\mu_{p,i} - \mu_{q,i})^{\alpha} s_i}{\sum_{i=1}^{N} (\mu_{p,i} - \mu_{q,i})^{\alpha}} \tag{5.14}$$

where $\mu_{p,i}$ and $\mu_{q,i}$ are the first and the second largest values on i-th column on the membership matrix U, and s_i is the silhouette score of $\boldsymbol{x}_i$ [95].

5.6.3 Competitors

5.6.3.1 Fk-modes algorithm

It is oblivious that Fk-modes is the first method that can conduct the the fuzzy clustering of categorical data. Fk-modes use modes to present the fuzzy clusters as same as k-modes. The values of mode are set by the categorical values with highest fuzzy frequencies on every categorical domains:

$$\boldsymbol{c}_{jd} = \arg\max_{v \in \mathcal{A}_d} \sum_{i, x_{id} = v} u_{ij}^{\alpha}, \quad 1 \leq d \leq D, 1 \leq j \leq k \tag{5.15}$$

Because of the principle of modes, Fk-modes has the same problem with k-modes [91].

5.6.3.2 Fuzzy centroids clustering algorithm (FCentroids)

FCentroids is the extension of k-representative to handle fuzzy clustering of categorical data [55], the representatives are also called fuzzy centers, in which the tuples of categorical values and their fuzzed probabilities are stored to form cluster center as same as the original k-representatives algorithm:

$$\boldsymbol{c}_{jd} = [\{v, \text{FPr}_{j,d}(v)\} | \forall v \in \mathcal{A}_j], \quad 1 \leq d \leq D, 1 \leq j \leq k \tag{5.16}$$

5.6.3.3 Categorical encoding-based fuzzy clustering methods

Instead of using different structures to form the cluster representations, other methods can transform the original categorical data into numerical data so that the original k-means algorithm can be applied:

- Fk-means uses a unique numerical value to present a unique categorical value [96], in this context, all the categorical values are treated equally and completely independently.

- FEk-means divides the categorical domains into groups of numerical domains, each unique categorical value forms a numerical attribute for itself [96]. This method increases the number of dimensions of the data dramatically.
- Fuzzy Space-Based Clustering (FSBC) directly uses the similarity matrix of pairwise objects to conduct the clustering [97]. This method requires the computation of the dissimilarity matrix of all objects and increase the dimensions same as the number of objects.

5.6.3.4 Genetic algorithms for fuzzy clustering

Recently, genetic algorithms are also widely used to handle fuzzy clustering of categorical data because they have advantages to handle categorical data [24, 98]. The following research on generic algorithms are selected as our competitors:

- SGA-Sep defines a single objective to maximize the sum of distances between clusters.
- SGA-Dist defines a single objective to minimize the sum of distances of all objects to their nearest cluster.
- SGA-SepDist defines a single objective of a fraction of cluster-separation and cluster-competitiveness [25].
- MOGA algorithm defines outer-cluster separation and inner-cluster compactness as the two distinguish objectives [24].
- NSGA-FMC algorithm uses the same these two objectives but it uses the membership matrix as the chromosomes [23].
- MaOFCentroids technique also uses the fuzzy membership for the chromosomes and they include several clustering evaluation metrics to define many objectives of clustering. The many-objectives problem is solved by NSGA-III selection technique [98].

5.6.4 Cluster prediction analysis

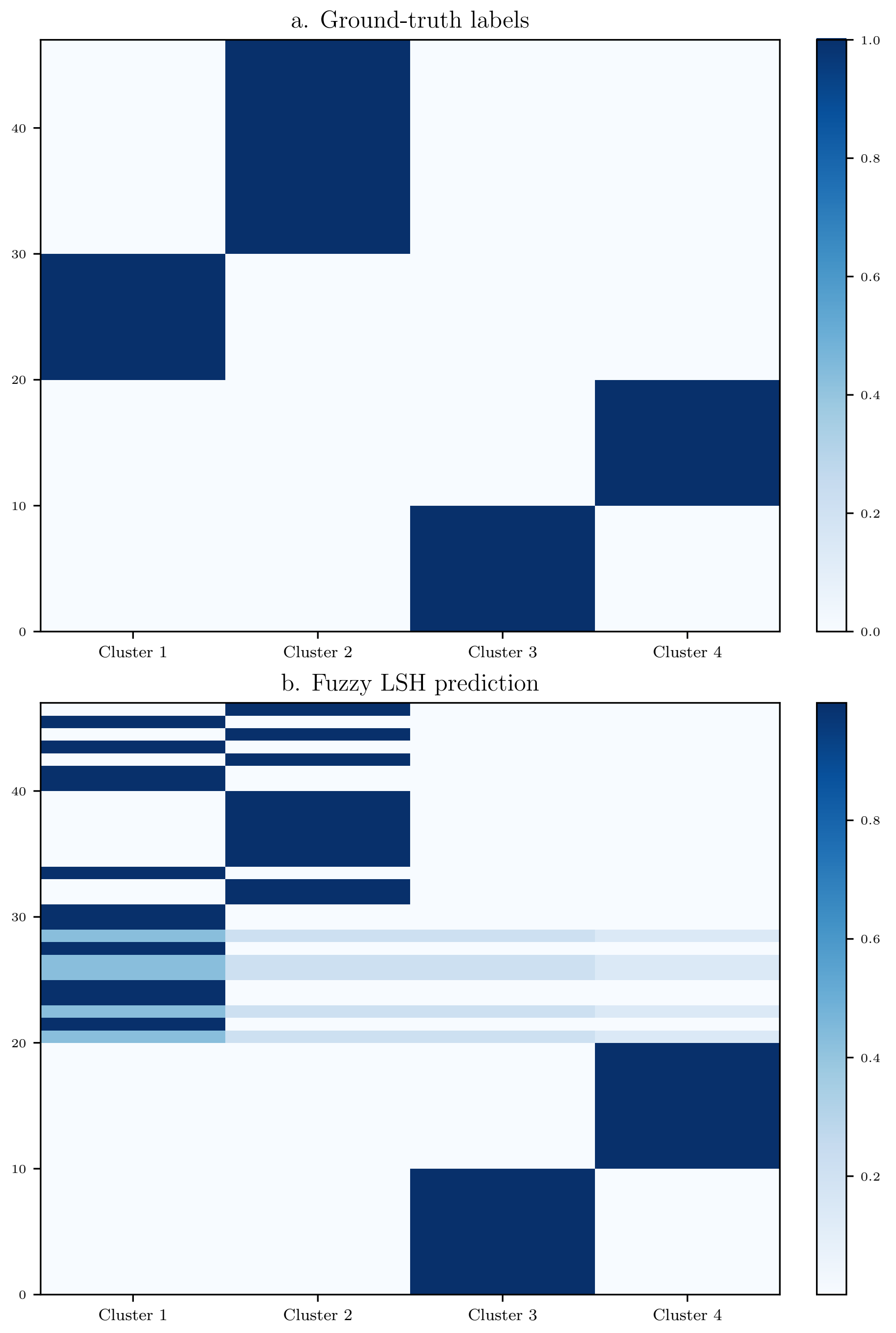

Figure 5.2: Accuracy of LSH-based fuzzy cluster prediction versus ground-truth labels on soybean-small dataset

First of all, the prediction accuracy of proposed method is visualized when applying to UCI soybean-small dataset, this dataset is the most used categorical data for demonstration because of it simplicity. In detail, Figure 5.2.a shows the ground-truth labels[2] as $\hat{G}_1 = \{\boldsymbol{x}_{21}, \dots, \boldsymbol{x}_{30}\}$, $\hat{G}_2 = \{\boldsymbol{x}_{31}, \dots, \boldsymbol{x}_{47}\}$, $\hat{G}_3 = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{10}\}$, and $\hat{G}_4 = \{\boldsymbol{x}_{11}, \dots, \boldsymbol{x}_{20}\}$. In this experiment, our LSH-based prediction model uses $l = 4$ hash functions, with that number of hash functions, we can achieve maximum $2^l = 16$ different hash values. However, after conducting hash table construction, only 5 buckets have objects namely $B_{12} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_{10}\}$, $B_{11} = \{\boldsymbol{x}_{11}, \dots, \boldsymbol{x}_{20}\}$, $B_6 = \{\boldsymbol{x}_{21}, \boldsymbol{x}_{23}, \boldsymbol{x}_{26}, \boldsymbol{x}_{27}, \boldsymbol{x}_{29}\}$, $B_{14} = \{\boldsymbol{x}_{22}, \boldsymbol{x}_{24}, \boldsymbol{x}_{25}, \boldsymbol{x}_{28}, \boldsymbol{x}_{30}, \boldsymbol{x}_{31}, \boldsymbol{x}_{34}, , \boldsymbol{x}_{41}, \boldsymbol{x}_{42}, \boldsymbol{x}_{44}, \boldsymbol{x}_{46}\}$,
and $B_{15} = \{\boldsymbol{x}_{32}, \boldsymbol{x}_{33}, \boldsymbol{x}_{35}, \dots, \boldsymbol{x}_{40}, \boldsymbol{x}_{43}, \boldsymbol{x}_{45}, \boldsymbol{x}_{47}\}$. Therefore, B_{14}, B_{15}, B_{12}, and B_{11} are selected as core buckets while objects in B_6 must share their degrees of membership to the core buckets. As the results, in Figure 5.2.b, because the Hamming distances between bucket B_6 to B_{14} are 1 and get the smallest value of 1, the degrees of membership of objects in B_6 to the core bucket (cluster) B_{14} are the highest. In an intuitive way, our prediction model gives relatively accurate results compared to the ground-truth labels; Especially, it can absolutely accurately predict the true labels for cluster 3 and cluster 4, with an accuracy of 41/47 (87.2%) objects in total.

[2]The ground-truth labels has been swapped to match the predictor's labels.

5.6.5 Crisp clustering performance

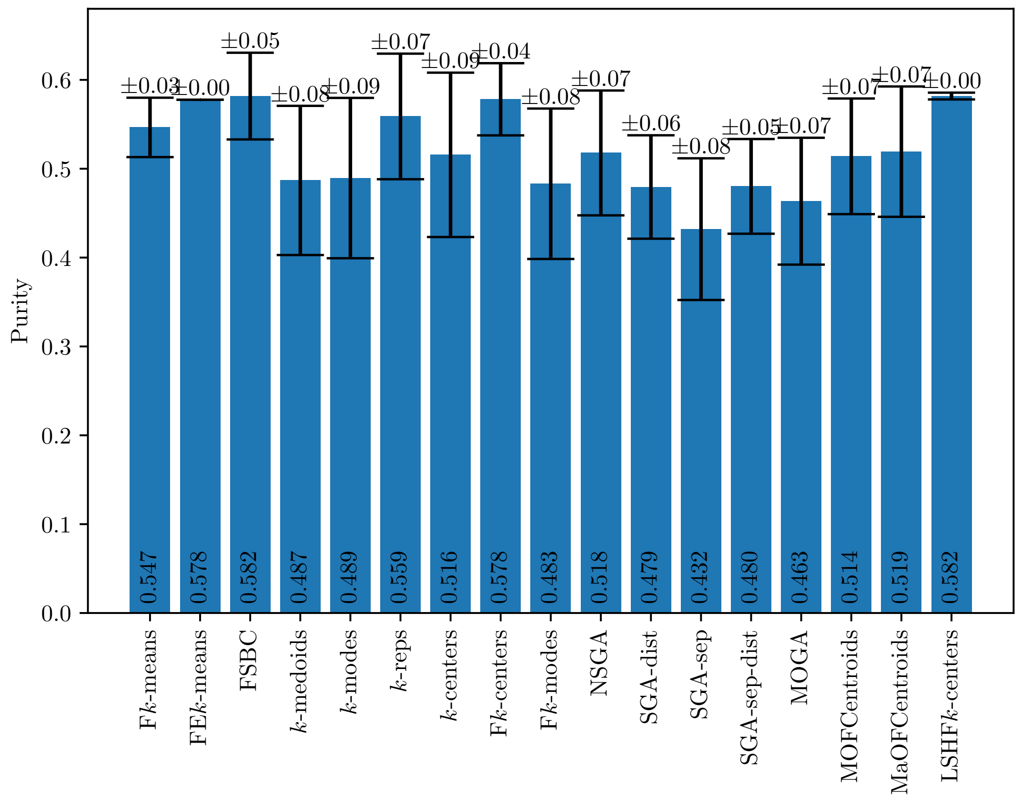

Figure 5.3: Average of purity and standard deviation scores on 16 UCI datasets

Figure 5.3 shows the accuracy and stability of the compared methods. It is obvious that LSHFk-centers and FSBC have the same highest average Purity score of 0.582. However, because of the constancy of the hash table, LSHFk-centers have higher stability with a standard deviation of Purity score of 0.

5.6.6 Fuzzy clustering performance analysis

Table 5.2: Fsilhouette scores of all methods on 16 UCI testing datasets

Dataset	Fk-means	FEk-means	FSBC	k-medoids	k-modes	k-reps	k-centers	Fk-centers	Fk-modes	NSGA	SGA-dist	SGA-sep	SGA-sep-dist	MOGA	MOF Centroids	MaOF Centroids	LSHF k-centers
Balance-scale	0.133 ±0.03	0.180 ±0.00	0.069 ±0.06	0.102 ±0.04	0.103 ±0.05	0.150 ±0.07	0.166 ±0.07	0.086 ±0.08	0.157 ±0.05	0.096 ±0.05	0.104 ±0.05	0.104 ±0.05	0.104 ±0.04	0.152 ±0.05	0.091 ±0.05	0.122 ±0.04	**0.274 ±0.00**
Soybean small	0.393 ±0.08	0.536 ±0.00	0.190 ±0.19	0.411 ±0.22	0.419 ±0.23	0.520 ±0.19	0.378 ±0.21	0.537 ±0.06	0.422 ±0.21	0.417 ±0.20	0.442 ±0.18	0.200 ±0.12	0.370 ±0.16	0.307 ±0.15	0.395 ±0.17	0.399 ±0.21	**0.537 ±0.00**
Zoo	0.572 ±0.11	0.565 ±0.00	0.257 ±0.10	0.551 ±0.15	0.560 ±0.15	0.593 ±0.12	0.527 ±0.11	0.635 ±0.10	0.591 ±0.14	0.520 ±0.11	0.548 ±0.10	0.233 ±0.11	0.252 ±0.08	0.281 ±0.10	0.402 ±0.08	0.583 ±0.09	**0.692 ±0.00**
Hayes-roth	0.218 ±0.07	0.354 ±0.00	0.175 ±0.00	0.242 ±0.03	0.240 ±0.03	0.300 ±0.04	0.299 ±0.08	**0.363 ±0.00**	0.326 ±0.03	0.244 ±0.05	0.253 ±0.01	0.189 ±0.06	0.277 ±0.02	0.331 ±0.03	0.244 ±0.04	0.265 ±0.05	0.363 ±0.00
Derma-tology	0.364 ±0.12	0.247 ±0.00	0.119 ±0.10	0.176 ±0.09	0.157 ±0.10	0.350 ±0.10	0.209 ±0.13	**0.406 ±0.09**	0.206 ±0.08	0.264 ±0.09	-0.200 ±0.06	0.125 ±0.07	-0.272 ±0.04	0.043 ±0.06	0.218 ±0.09	0.331 ±0.09	0.310 ±0.05
Soybean	0.445 ±0.05	0.490 ±0.00	0.194 ±0.06	0.355 ±0.05	0.364 ±0.05	0.577 ±0.06	0.414 ±0.07	**0.623 ±0.08**	0.355 ±0.05	0.341 ±0.05	-0.269 ±0.04	0.014 ±0.04	-0.303 ±0.04	-0.128 ±0.05	0.371 ±0.04	0.545 ±0.05	0.610 ±0.00
Nursery	0.084 ±0.01	**0.162 ±0.00**	N/A	0.104 ±0.04	0.108 ±0.03	0.156 ±0.06	0.135 ±0.05	-0.114 ±0.01	0.149 ±0.04	0.105 ±0.04	0.110 ±0.03	0.106 ±0.02	0.110 ±0.03	0.145 ±0.03	0.107 ±0.04	0.143 ±0.04	-0.110 ±0.01
Connect	**0.158 ±0.01**	0.157 ±0.00	N/A	0.108 ±0.05	0.105 ±0.05	0.148 ±0.03	0.100 ±0.06	0.143 ±0.00	0.143 ±0.04	0.099 ±0.06	-0.122 ±0.00	0.053 ±0.06	-0.134 ±0.00	-0.004 ±0.04	0.105 ±0.05	0.102 ±0.06	0.143 ±0.00
Chess	0.216 ±0.03	0.218 ±0.00	0.038 ±0.00	0.143 ±0.05	0.146 ±0.07	0.195 ±0.04	0.198 ±0.04	**0.246 ±0.00**	0.168 ±0.06	0.152 ±0.05	0.176 ±0.03	0.100 ±0.06	0.023 ±0.00	0.158 ±0.04	0.160 ±0.05	0.147 ±0.05	**0.246 ±0.00**
Breast	0.406 ±0.00	0.412 ±0.00	0.286 ±0.00	0.165 ±0.20	0.156 ±0.20	0.395 ±0.00	0.354 ±0.07	**0.474 ±0.00**	0.268 ±0.23	0.380 ±0.02	0.382 ±0.04	0.190 ±0.09	0.379 ±0.04	0.416 ±0.06	0.394 ±0.01	0.373 ±0.03	**0.474 ±0.00**
Car	0.092 ±0.01	0.190 ±0.00	0.049 ±0.06	0.115 ±0.05	0.116 ±0.06	0.178 ±0.06	0.172 ±0.06	-0.005 ±0.15	0.168 ±0.06	0.112 ±0.03	0.124 ±0.06	0.122 ±0.05	0.124 ±0.05	0.165 ±0.06	0.109 ±0.03	0.163 ±0.04	**0.260 ±0.00**
Mush-room	0.231 ±0.00	0.276 ±0.00	N/A	0.208 ±0.15	0.228 ±0.14	0.250 ±0.15	0.201 ±0.15	**0.283 ±0.00**	0.246 ±0.12	0.212 ±0.12	0.195 ±0.12	0.109 ±0.12	0.147 ±0.11	0.131 ±0.10	0.223 ±0.14	0.208 ±0.11	**0.283 ±0.00**
Tic tactoe	0.124 ±0.00	0.149 ±0.00	0.028 ±0.03	0.104 ±0.07	0.104 ±0.06	0.138 ±0.06	0.135 ±0.09	0.164 ±0.01	0.129 ±0.07	0.094 ±0.05	0.105 ±0.06	0.081 ±0.05	0.105 ±0.06	0.123 ±0.08	0.100 ±0.05	0.097 ±0.04	**0.164 ±0.00**
Vote	0.516 ±0.00	0.519 ±0.00	0.135 ±0.00	0.490 ±0.01	0.483 ±0.05	0.498 ±0.00	0.485 ±0.07	**0.527 ±0.00**	0.519 ±0.01	0.486 ±0.02	0.491 ±0.01	0.183 ±0.16	0.484 ±0.02	0.443 ±0.10	0.501 ±0.01	0.476 ±0.06	**0.527 ±0.00**
Flare	0.347 ±0.00	0.392 ±0.00	0.169 ±0.00	0.303 ±0.06	0.281 ±0.07	0.434 ±0.07	0.252 ±0.15	**0.487 ±0.00**	0.331 ±0.07	0.278 ±0.11	0.366 ±0.10	0.203 ±0.15	0.408 ±0.13	0.347 ±0.13	0.297 ±0.10	0.281 ±0.12	**0.487 ±0.00**
Lung	0.168 ±0.02	0.307 ±0.00	0.119 ±0.07	0.176 ±0.09	0.185 ±0.10	0.273 ±0.09	0.187 ±0.08	0.317 ±0.08	0.207 ±0.10	0.264 ±0.09	0.190 ±0.05	0.148 ±0.07	0.171 ±0.04	0.178 ±0.07	0.234 ±0.09	0.268 ±0.09	**0.318 ±0.00**
Average	0.279 ±0.03	0.322 ±0.00	0.141 ±0.05	0.235 ±0.08	0.235 ±0.09	0.322 ±0.07	0.263 ±0.09	0.323 ±0.04	0.274 ±0.08	0.254 ±0.07	0.181 ±0.06	0.135 ±0.08	0.140 ±0.05	0.193 ±0.07	0.247 ±0.07	0.281 ±0.07	**0.349 ±0.00**

Table 5.2 shows the comprehensive comparisons of the Fk-centers and LSHFk-centers with other related works. Clearly, Fk-centers has higher Fsilhouette scores than all other related works; for this reason, *fcenter* can outperform other representation structures. Moreover, with the support of our LSH-based cluster prediction technique, LSHFk-centers can even achieve higher accuracy scores than its original method, namely Fk-centers. For more detail, Fk-centers has the highest Fsilhouette scores for 8/16 testing datasets while LSHFk-centers can win at 11/16 testing datasets. Moreover, the LSH-based cluster prediction approach also helps to reduce the standard deviation of Fk-centers from ±0.04 to ±0.00.

5.6.7 Complexity analysis

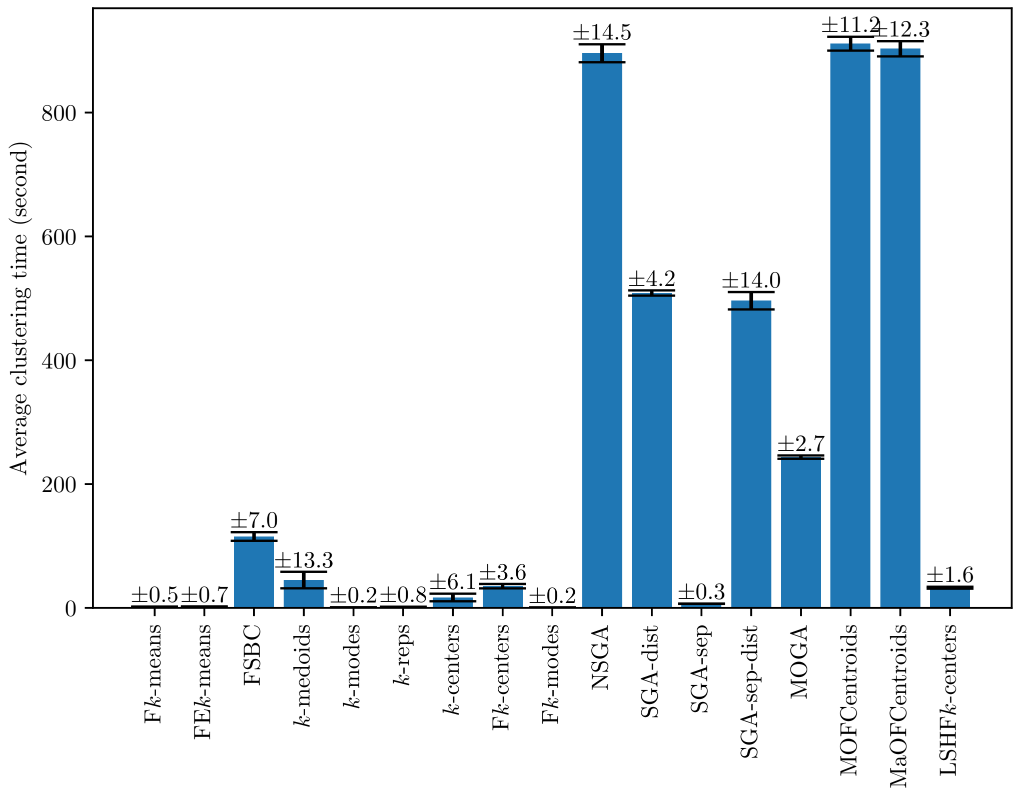

Figure 5.4: Average clustering time on 16 UCI datasets

To analyze the complexity of our methods compared to other related works, we let a clustering algorithm process to each of 16 testing datasets for 128 times. Then, the average and standard deviation values of the clustering time are then calculated for each method on each dataset. Figure 5.4 shows the averages of the average clustering times and their standard deviation values for all methods. From this figure, we can see that the genetic algorithms take significant time to cluster with 100 generations of $4k$ individuals. FSBC is the slowest k-means-like the method because it significantly increases the dimensionality of the data. It is clear that the accuracy-complexity trade-off is always present in all k-means-like algorithms, as an illustration, our methods Fk-centers and LSHFk-centers require more clustering times than most of other k-means-like clustering techniques by the trade-off of achieving higher clustering accuracy. Another key point is that LSHFk-centers requires a slightly higher computation time than its original method Fk-centers.

5.7 Summary

In this chapter, we propose two methods so-called Fk-centers and LSHFk-centers. Fk-centers is the first method to apply the fuzzy kernel-based representation (*fcenter*) into the fuzzy clustering algorithm of categorical data. Moreover, LSHFk-centers is the method that incorporates our LSH-based cluster prediction techniques into Fk-centers algorithm. The advantages and disadvantages of these two methods can be listed as:

5.7.1 Advantages

- For both Fk-centers and LSHFk-centers, we define the frequency of categorical value in a cluster under the context of fuzzy cluster using the corresponding degree of membership.
- Fk-centers has better representation than other related works, which can exploit the observed information of categorical values following the complementarity balance of the importance of each unique categorical value in each fuzzy cluster. As the result, Fk-centers has a higher average fuzzy silhouette score than other related works.
- When applying the modified version of our LSH-based cluster prediction technique into Fk-centers, LSHFk-centers can even have higher average fuzzy silhouette scores with extremely high stability.

5.7.2 Limitations

- The process of Fk-centers calculation requires to compute the optimal values for smoothing parameters, which needs to take into account the fuzzy frequencies of all unique categorical values in all clusters. For this

reason, Fk-centers has a higher clustering time than other k-means-like algorithms.
- Our LSH-based fuzzy cluster prediction approach is highly stable but still not random, which makes the outcome of the predicted cluster the same even though many initializations.

5.7.3 Future works

- In terms of Fk-centers, other research may try different techniques to compute the optimal values for the smoothing parameters λ other than using LSCV. Thereby, it can either reduce the computation time or increase the accuracy of smoothing parameters.
- In terms of LSHFk-centers, further research should analyze the way we assign the degrees of membership of objects on the core buckets, such that the objects in the core buckets can share the degrees of membership to other clusters as well.
- We suggest future research to extend the kernel-based representation (Fk-centers) for handling multi-objective problem with focus on inner-cluster compactness and outer-cluster separation at the same time because separation between fuzzy clusters factor is also very important for fuzzy clustering.

Chapter 6

Conclusion and future work

6.1 Summary

This dissertation states several problems of cluster analysis for categorical and mixed data and proposes four corresponding clustering algorithms of the class of k-means-like algorithms to handle these problems. The clustering effectiveness and clustering performance of these proposed methods have been experimented in the real and synthetic datasets with standard measurements on the same testing conditions. In detail, the abstracts of our proposed methods can be described as follows:

First, to handle the problem of crisp clustering of categorical data, we proposed LSH-k-representatives, which can incorporate the LSH technique to predict the potential *representatives* and use cluster shortlist to reduce the complexity of the algorithm. To emphasize, the LSH-k-representatives(Init) alone outperforms other state-of-the-art clustering algorithms in terms of clustering accuracy. Besides that, LSH-k-representatives(Full) can run faster than LSH-k-representatives(Init) from 2 to 32 times with comparable accuracy.

Second, we introduce the LSH-k-prototypes algorithm to cluster the data of mixed values, which is the general version of LSH-k-representatives. In the context of the LSH-k-prototypes algorithm, the kinds of attributes are separated into two terms: numerical and categorical attributes so that the appropriate techniques are used to handle these attribute types, namely, the Euclidean and overlap metrics are used for calculating the dissimilarity between two mixed objects with the appropriate weights. Moreover, a hybrid cluster representation so-called *prototype* is utilized to represent the clusters of mixed values, which is the combination of *centroid* and *representative*. As can be seen, LSH-k-prototypes has higher clustering accuracy than k-prototypes and k-Imprototypes.

Third, we applied kernel-based representation to fuzzy clustering of categorical data. Our proposed method Fk-centers can increase the influence of unique categorical values that have low occurrence frequencies in a cluster, which helps to reduce the bias of dominant unique categorical values in all domains. Certainly, *fcenter* is more complicated than other representations such as *medoid*, *mode*, and *representative*. Because of that, the Fk-centers algorithm has much better convergence than previous methods.

Fourth, with the new fuzzy clustering algorithm for categorical data, we can incorporate our LSH-based cluster prediction technique into Fk-centers. The incorporated algorithm so-called LSHFk-centers can achieve higher cluster effectiveness than its original method and has extremely high consistency.

6.2 Limitations

There are several inherent limitations of the dissertation as shown below:

- First, the types of hash functions in our LSH-based cluster prediction technique are not diversified yet. A hash function is currently extracted from a single attribute; therefore the inter-attribute properties have not been untapped.
- Second, different dissimilarity measures may give different results for the locality-sensitive factor of the hashing model. In this research, we limited the use of DILCA for exploiting the dissimilarity matrices.
- Third, the number of hash functions is a very important parameter for the prediction model. In this research, we only used a simple calculation to estimate the suitable number of hash functions for each dataset based on the number the target clusters and the number of attributes of this dataset.
- Fourth, this dissertation introduces two different ways to update the cluster shortlists during the clustering process. The first way focuses on accuracy and the second way focuses on speed. There is the shortcoming of techniques for updating the cluster shortlists, which can balance the accuracy and speed.
- Fifth, in terms of limitation our fuzzy clustering algorithms Fk-centers and LSHFk-centers, the process of updating the smoothing parameters is costly, which doubles the clustering time.

6.3 Future works

Based on the mentioned limitations, we list the corresponding recommendations for future research as follows:

- First, the effectiveness of multi-attribute hash functions should be comprehensively analyzed. Expectedly, a multi-attribute hash function should capture better the similarity of objects represented by multiple attributes. Further research should therefore be focused on proposing a "hyperplane" concept for categorical objects in the similar fashion as numerical data.
- Second, there are numerous different dissimilarity metrics for categorical data that have not been analyzed in this study. Future research can propose a new dissimilarity metric that only focuses on the purpose of creating the hash functions for categorical data is also very promising.
- Third, as can be seen in Figure 4.8, our estimated number of hash functions is different from the optimal value. Future research may explore the correlation between distributions of categorical values in categorical domains with the number of hash functions. A new metric to estimate the optimal number of hash functions can then be proposed based on the properties of maximum cuts in categorical domains.
- Fourth, to have the better way to update the cluster shortlist, we can follow a completely different approach by using the states of labels of objects after the label updating process instead of using the distances between cluster representations. In detail, when a high number of objects in a current cluster are moved to a particular cluster, the chances of them being in other shortlists is high. In contrast, in the case just a few or no objects are transferred between a particular pair of clusters, then the distance between them is relatively far away so that they should not include the other cluster into their shortlist.
- Fifth, because the processes of updating the smoothing parameters and the *fcenter* require the process of calculating, the fuzzy frequencies of unique categorical values in each domain, we can reuse these results and avoid the duplicated calculations. Moreover, different techniques other than LSCV can be applied to achieve better performance and complexity.

6.4 Code and reproducibility

For the sake of open source and reproducibility of this study, we extracted the core components of the source code and packed them in packages, which are published into the python repository site (pipy). The descriptions and instructions of these packages are included in the following repositories:

- https://pypi.org/project/lshkrepresentatives/: The python source codes of LSH-k-representatives and LSH-k-prototypes are included.

- https://pypi.org/project/fkcenters/: The python source codes of Fk-centers and LSHFk-centers are included.

The default evaluation metrics are also included in those repositories.

Publications

Journals

[1] T. N. Mau and V.-N. Huynh, "An LSH-based k-Representatives Clustering Method for Large Categorical Data." Neurocomputing, Volume 463, 2021, Pages 29-44, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2021.08.050.

[2] T. N. Mau, Y. Inoguchi and V.-N. Huynh, "A Novel Cluster Prediction Approach based on Locality-Sensitive Hashing for Fuzzy Clustering of Categorical Data." IEEE Access, (Under review).

Conferences

[3] T. N. Mau and V. -N. Huynh, "Kernel-Based k-Representatives Algorithm for Fuzzy Clustering of Categorical Data," 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2021, pp. 1-6, doi: 10.1109/FUZZ45933.2021.9494597.

[4] T. N. Mau and V. -N. Huynh, "Automated Attribute Weighting Fuzzy k-Centers Algorithm for Categorical Data Clustering," Modeling Decisions for Artificial Intelligence. MDAI 2021. Lecture Notes in Computer Science, vol 12898. Springer, Cham. https://doi.org/10.1007/978-3-030-85529-1_17.

References

[1] Pradeep Rai and Shubha Singh. A Survey of Clustering Techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010.

[2] Memoona Khanum, Tahira Mahboob, Warda Imtiaz, Humaraia Abdul Ghafoor, and Rabeea Sehar. A Survey on Unsupervised Machine Learning Algorithms for Automation, Classification and Maintenance. *International Journal of Computer Applications*, 119(13), 2015.

[3] Nathan Wiebe, Ashish Kapoor, and Krysta Svore. Quantum Algorithms for Nearest-Neighbor Methods for Supervised and Unsupervised Learning. *arXiv preprint arXiv:1401.2142*, 2014.

[4] Salima Omar, Asri Ngadi, and Hamid H Jebur. Machine Learning Techniques for Anomaly Detection: an Overview. *International Journal of Computer Applications*, 79(2), 2013.

[5] Sebastian J Wetzel. Unsupervised Learning of Phase Transitions: From Principal Component Analysis to Variational Autoencoders. *Physical Review E*, 96(2):022140, 2017.

[6] Pamela A Sample, Catherine Boden, Zuohua Zhang, John Pascual, Te-Won Lee, Linda M Zangwill, Robert N Weinreb, Jonathan G Crowston, Esther M Hoffmann, Felipe A Medeiros, et al. Unsupervised Machine Learning with Independent Component Analysis to Identify Areas of Progression in Glaucomatous Visual Fields. *Investigative ophthalmology & visual science*, 46(10):3684–3692, 2005.

[7] Thomas Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine learning*, 42(1):177–196, 2001.

[8] Rui Xu and Donald Wunsch. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[9] Marius Muja and David G Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. *VISAPP (1)*, 2(331-340):2, 2009.

[10] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product Quantization for Nearest Neighbor Search. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2010.

[11] Lian Duan, Lida Xu, Ying Liu, and Jun Lee. Cluster-based Outlier Detection. *Annals of Operations Research*, 168(1):151–168, 2009.

[12] Lyle H Ungar and Dean P Foster. Clustering Methods for Collaborative Filtering. In *AAAI Workshop on Recommendation Systems*, volume 1, pages 114–129. Menlo Park, CA, 1998.

[13] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means Clustering Algorithm. *Pattern Recognition*, 36(2):451–461, 2003.

[14] Donald E Gustafson and William C Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. In *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, pages 761–766. IEEE, 1979.

[15] 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*, 14:241–247, 2004.

[16] K Mahesh Kumar and A Rama Mohan Reddy. An Efficient k-means Clustering Filtering Algorithm using Density Based Initial Cluster Centers. *Information Sciences*, 418:286–301, 2017.

[17] Hongfu Liu, Junjie Wu, Tongliang Liu, Dacheng Tao, and Yun Fu. Spectral Ensemble Clustering via Weighted "k"-means: Theoretical and Practical Evidence. *IEEE transactions on knowledge and data engineering*, 29(5):1129–1143, 2017.

[18] Vinod Kumar Dehariya, Shailendra Kumar Shrivastava, and RC Jain. Clustering of Image Data Set using k-means and Fuzzy k-means Algorithms. In *2010 International Conference on Computational Intelligence and Communication Networks*, pages 386–391. IEEE, 2010.

[19] Paul S Bradley, Usama Fayyad, Cory Reina, et al. Scaling EM (Expectation-Maximization) Clustering to Large Databases. *Microsoft Research*, pages 0–25, 1998.

[20] Yong Gyu Jung, Min Soo Kang, and Jun Heo. Clustering Performance Comparison using k-means and Expectation Maximization Algorithms. *Biotechnology & Biotechnological Equipment*, 28(sup1):S44–S48, 2014.

[21] Juha Vesanto and Esa Alhoniemi. Clustering of the Self-Organizing Map. *IEEE Transactions on neural networks*, 11(3):586–600, 2000.

[22] Jouko Lampinen and Erkki Oja. Clustering Properties of Hierarchical Self-Organizing Maps. *Journal of Mathematical Imaging and vision*, 2(2):261–272, 1992.

[23] Chao-Lung Yang, RJ Kuo, Chia-Hsuan Chien, and Nguyen Thi Phuong Quyen. Non-dominated Sorting Genetic Algorithm using Fuzzy Membership Chromosome for Categorical Data Clustering. *Applied Soft Computing*, 30:113–122, 2015.

[24] Anirban Mukhopadhyay, Ujjwal Maulik, and Sanghamitra Bandyopadhyay. Multiobjective Genetic Algorithm-based Fuzzy Clustering of Categorical Attributes. *IEEE transactions on evolutionary computation*, 13(5):991–1005, 2009.

[25] Guojun Gan, J Wu, and Z Yang. A Genetic Fuzzy k-modes Algorithm for Clustering Categorical Data. *Expert Systems with Applications*, 36(2):1615–1620, 2009.

[26] Amir Ahmad and Lipika Dey. A k-mean Clustering Algorithm for Mixed Numeric and Categorical Data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[27] Liyong Zhang, Wanxie Zhong, Chongquan Zhong, Wei Lu, Xiaodong Liu, and Witold Pedrycz. Fuzzy c-means Clustering Based on Dual Expression between Cluster Prototypes and Reconstructed Data. *International Journal of Approximate Reasoning*, 90:389–410, 2017.

[28] 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.

[29] Duy-Tai Dinh and Van-Nam Huynh. k-PbC: an Improved Cluster Center Initialization for Categorical Data Clustering. *Applied Intelligence*, pages 1–23, 2020.

[30] Duy-Tai Dinh, Van-Nam Huynh, and Songsak Sriboonchitta. Clustering Mixed Numerical and Categorical Data with Missing Values. *Information Sciences*, 2021.

[31] Charles Elkan. Using the Triangle Inequality to Accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 147–153, 2003.

[32] Ryan McConville, Xin Cao, Weiru Liu, and Paul Miller. Accelerating Large Scale Centroid-based Clustering with Locality Sensitive Hashing. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 649–660. IEEE, 2016.

[33] Timothy C Havens, James C Bezdek, Christopher Leckie, Lawrence O Hall, and Marimuthu Palaniswami. Fuzzy c-means Algorithms for Very Large Data. *IEEE Transactions on Fuzzy Systems*, 20(6):1130–1146, 2012.

[34] Nicolas Tremblay and Andreas Loukas. Approximating Spectral Clustering via Sampling: a Review. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 129–183. Springer, 2020.

[35] Soumi Ghosh and Sanjay Kumar Dubey. Comparative Analysis of k-means and Fuzzy c-means Algorithms. *International Journal of Advanced Computer Science and Applications*, 4(4), 2013.

[36] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *arXiv preprint arXiv:1203.6402*, 2012.

[37] David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. Technical report, Stanford, 2006.

[38] Mohammad Norouzi and David J Fleet. Cartesian k-means. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3017–3024, 2013.

[39] Brian Kulis and Michael I Jordan. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. *arXiv preprint arXiv:1111.0352*, 2011.

[40] 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.

[41] Paul S Bradley and Usama M Fayyad. Refining Initial Points for k-means Clustering. In *International Conference on Machine Learning*, volume 98, pages 91–99. Citeseer, 1998.

[42] Fuyuan Cao, Jiye Liang, Deyu Li, Liang Bai, and Chuangyin Dang. A Dissimilarity Measure for the k-modes Clustering Algorithm. *Knowledge-Based Systems*, 26:120–127, 2012.

[43] Dino Ienco, Ruggero G Pensa, and Rosa Meo. From Context to Distance: Learning Dissimilarity for Categorical Data Clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(1):1–25, 2012.

[44] Si Quang Le and Tu Bao Ho. An Association-Based Dissimilarity Measure for Categorical Data. *Pattern Recognition Letters*, 26(16):2549–2557, 2005.

[45] Madhavi Alamuri, Bapi Raju Surampudi, and Atul Negi. A Survey of Distance/Similarity Measures for Categorical Data. In *2014 International joint conference on neural networks (IJCNN)*, pages 1907–1914. IEEE, 2014.

[46] James MacQueen et al. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, number 14 in 1, pages 281–297. Oakland, CA, USA, 1967.

[47] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[48] Lei Xu and Michael I Jordan. On Convergence Properties of the EM Algorithm for Gaussian Mixtures. *Neural computation*, 8(1):129–151, 1996.

[49] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-Maximization Attention Networks for Semantic Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9167–9176, 2019.

[50] Shun-ichi Amari and Andrzej Cichocki. Information Geometry of Divergence Functions. *Bulletin of the polish academy of sciences. Technical sciences*, 58(1):183–195, 2010.

[51] Yasuo Matsuyama, Naoto Katsumata, and Ryo Kawamura. Independent Component Analysis Minimizing Convex Divergence. In *Artificial Neural Networks and Neural Information Processing—ICANN/ICONIP 2003*, pages 27–34. Springer, 2003.

[52] F-X Jollois and Mohamed Nadif. Speed-up for the Expectation-Maximization Algorithm for Clustering Categorical Data. *Journal of Global Optimization*, 37(4):513–525, 2007.

[53] Cláudia Silvestre, Margarida GMS Cardoso, and Mário Figueiredo. Feature Selection for Clustering Categorical Data with an Embedded Modelling Approach. *Expert systems*, 32(3):444–453, 2015.

[54] Zhexue Huang and Michael K Ng. A Fuzzy *k*-modes Algorithm for Clustering Categorical Data. *IEEE transactions on Fuzzy Systems*, 7(4):446–452, 1999.

[55] Dae-Won Kim, Kwang H Lee, and Doheon Lee. Fuzzy Clustering of Categorical Data Using Fuzzy Centroids. *Pattern recognition letters*, 25(11):1263–1271, 2004.

[56] Zhexue Huang. Clustering Large Data Sets with Mixed Numeric and Categorical Values. In *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*, pages 21–34. Citeseer, 1997.

[57] 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.

[58] Fuyuan Cao, Jiye Liang, and Liang Bai. A New Initialization Method for Categorical Data Clustering. *Expert Systems with Applications*, 36(7):10223–10228, 2009.

[59] 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.

[60] Shehroz S Khan and Amir Ahmad. Cluster Center Initialization Algorithm for *k*-modes Clustering. *Expert Systems with Applications*, 40(18):7444–7456, 2013.

[61] Feng Jiang, Guozhu Liu, Junwei Du, and Yuefei Sui. Initialization of *k*-modes Clustering Using Outlier Detection Techniques. *Information Sciences*, 332:167–183, 2016.

[62] Steven K Thompson. Adaptive Cluster Sampling. *Journal of the American Statistical Association*, 85(412):1050–1059, 1990.

[63] Liang Wang, Christopher Leckie, Ramamohanarao Kotagiri, and James Bezdek. Approximate Pairwise Clustering for Large Data Sets via Sampling Plus Extension. *Pattern Recognition*, 44(2):222–235, 2011.

[64] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. Optimized Stratified Sampling for Approximate Query Processing. *ACM Transactions on Database Systems (TODS)*, 32(2):9–es, 2007.

[65] Morteza Haghir Chehreghani, Hassan Abolhassani, and Mostafa Haghir Chehreghani. Improving Density-Based Methods for Hierarchical Clustering of Web Pages. *Data & Knowledge Engineering*, 67(1):30–50, 2008.

[66] Christopher R Palmer and Christos Faloutsos. Density Biased Sampling: an Improved Method for Data Mining and Clustering. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 82–92, 2000.

[67] Alexandros Nanopoulos, Yannis Manolopoulos, and Yannis Theodoridis. An Efficient and Effective Algorithm for Density Biased Sampling. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 398–404, 2002.

[68] Imola K Fodor. A Survey of Dimension Reduction Techniques. Technical report, Citeseer, 2002.

[69] J Edward Jackson. *A User's Guide to Principal Components*, volume 587. John Wiley & Sons, 2005.

[70] Katsuhiro Honda and Hidetomo Ichihashi. Regularized Linear Fuzzy Clustering and Probabilistic PCA Mixture Models. *IEEE Transactions on Fuzzy Systems*, 13(4):508–516, 2005.

[71] Miguel A Carreira-Perpinán. A Review of Dimension Reduction Techniques. *Department of Computer Science. University of Sheffield. Tech. Rep. CS-96-09*, 9:1–69, 1997.

[72] Samuel Kaski. Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, pages 413–418. IEEE, 1998.

[73] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[74] Wei Cen and Kehua Miao. An Improved Algorithm for Locality-Sensitive Hashing. In *Computer Science & Education (ICCSE), 2015 10th International Conference on*, pages 61–64. IEEE, 2015.

[75] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. Fast Locality-Sensitive Hashing. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1073–1081. ACM, 2011.

[76] Zeinab Khorshidpour, Sattar Hashemi, and Ali Hamzeh. Distance Learning for Categorical Attribute Based on Context Information. In *2010 2nd International Conference on Software Technology and Engineering*, volume 2, pages V2–296. IEEE, 2010.

[77] Mechthild Stoer and Frank Wagner. A Simple Min-cut Algorithm. *Journal of the ACM (JACM)*, 44(4):585–591, 1997.

[78] Kyung Mi Lee and Keon Myung Lee. A Locality Sensitive Hashing Technique for Categorical Data. In *Applied Mechanics and Materials*, volume 241, pages 3159–3164. Trans Tech Publ, 2013.

[79] Kyung Mi Lee and Keon Myung Lee. Efficient Search for Data with Numerical and Categorical Attributes based on Dual Locality-Sensitive Hashing. *Indian J. Sci. Technol*, 9:24, 2016.

[80] Qiang Wang, Zhiyuan Guo, Gang Liu, and Jun Guo. Entropy Based Locality Sensitive Hashing. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1045–1048. IEEE, 2012.

[81] Andrei Z Broder. On the Resemblance and Containment of Documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[82] Philip Koopman and Tridib Chakravarty. Cyclic Redundancy Code (CRC) Polynomial Selection for Embedded Networks. In *International Conference on Dependable Systems and Networks, 2004*, pages 145–154. IEEE, 2004.

[83] Toan Nguyen Mau and Van-Nam Huynh. An LSH-Based k-Representatives Clustering Method for Large Categorical Data. *Neurocomputing*, 463:29–44, 2021.

[84] Moses S Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[85] Qin Lv, Moses Charikar, and Kai Li. Image Similarity Search with Compact Data Structures. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 208–217, 2004.

[86] Andrew Frank, Arthur Asuncion, et al. UCI Machine Learning Repository, 2010. *URL http://archive. ics. uci. edu/ml*, 15:22, 2011.

[87] James C Bezdek and Richard J Hathaway. VAT: A tool for visual assessment of (cluster) tendency. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pages 2225–2230. IEEE, 2002.

[88] Gabor Melli. The Datgen Dataset Generator, 1999.

[89] Lifei Chen and Shengrui Wang. Central Clustering of Categorical Data with Automated Feature Weighting. In *IJCAI*, pages 1260–1266, 2013.

[90] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. *Introduction to Information Retrieval.* Cambridge university press, 2008.

[91] Toan Nguyen Mau and Van-Nam Huynh. Kernel-Based k-Representatives Algorithm for Fuzzy Clustering of Categorical Data. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2021.

[92] Sadaaki Miyamoto, Hodetomo Ichihashi, Katsuhiro Honda, and Hidetomo Ichihashi. *Algorithms for Fuzzy Clustering.* Springer, 2008.

[93] Chi-Hyon Oh, Katsuhiro Honda, and Hidetomo Ichihashi. Fuzzy Clustering for Categorical Multivariate Data. In *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, volume 4, pages 2154–2159. IEEE, 2001.

[94] Qi Li and Jeffrey Scott Racine. *Nonparametric Econometrics: Theory and Practice.* Princeton University Press, 2007.

[95] Ricardo JGB Campello and Eduardo R Hruschka. A Fuzzy Extension of the Silhouette width Criterion for Cluster Analysis. *Fuzzy Sets and Systems*, 157(21):2858–2875, 2006.

[96] Kedar Potdar, Taher S Pardawala, and Chinmay D Pai. A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International journal of computer applications*, 175(4):7–9, 2017.

[97] Yuhua Qian, Feijiang Li, Jiye Liang, Bing Liu, and Chuangyin Dang. Space Structure and Clustering of Categorical Data. *IEEE transactions on neural networks and learning systems*, 27(10):2047–2059, 2015.

[98] Shuwei Zhu and Lihong Xu. Many-objective fuzzy centroids clustering algorithm for categorical data. *Expert Systems with Applications*, 96:230–248, 2018.

[99] Toan Nguyen Mau and Van-Nam Huynh. Automated Attribute Weighting Fuzzy k-Centers Algorithm for Categorical Data Clustering. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 205–217. Springer, 2021.

Appendix A
Attribute weighting

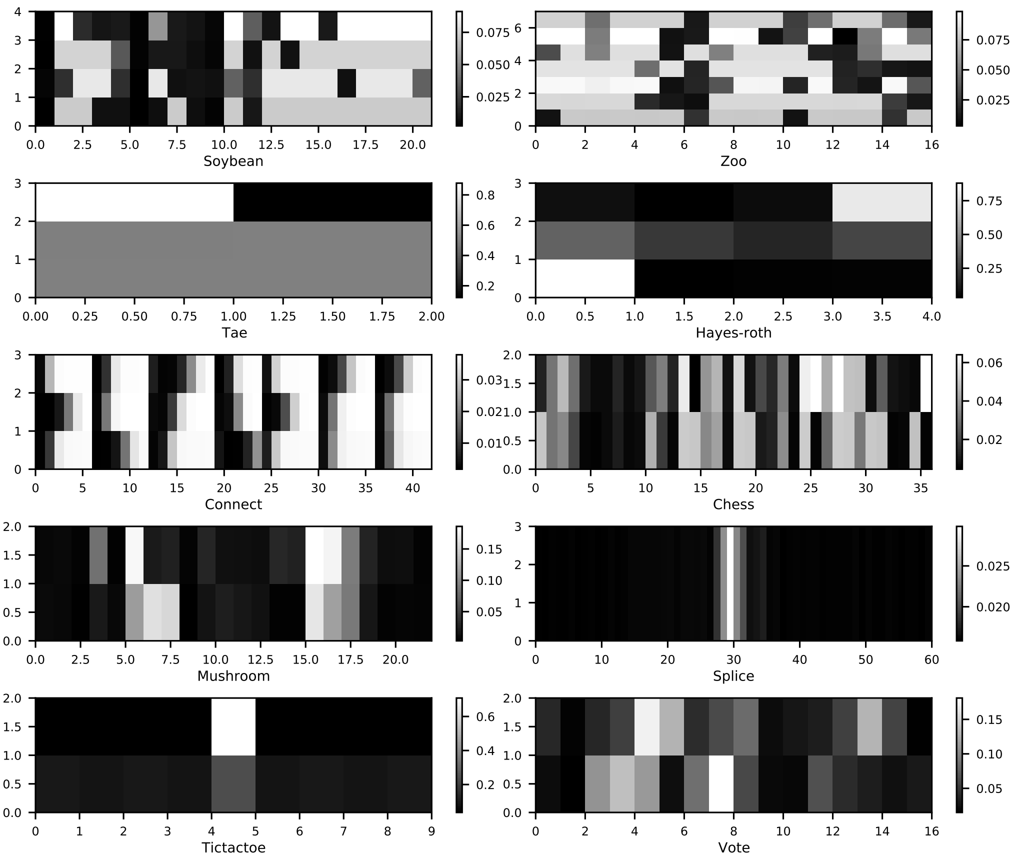

Figure A.1: Contributes of categorical attributes to the clustering outcome

Figure A.1 shows the contribution of each attribute to each cluster on 8 UCI datasets at the final clustering iteration, the vertical axes present the clusters while horizontal axes present the attributes. The values on the charts represent the data dispersion on each dimension on each attribute [99].