Fuzzy Clustering with Similarity Queries

Wasim Huleihel
Department of Electrical Engineering
Tel Aviv University
Tel Aviv 6997801, Israel
wasimh@tauex.tau.ac.il

Arya Mazumdar
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, CA 92093
arya@ucsd.edu

Soumyabrata Pal
College of Information & Computer Sciences
University of Massachusetts Amherst
Amherst, MA 01003
soumyabratap@umass.edu

Abstract

The fuzzy or soft k-means objective is a popular generalization of the well-known k-means problem, extending the clustering capability of the k-means to datasets that are uncertain, vague and otherwise hard to cluster. In this paper, we propose a semi-supervised active clustering framework, where the learner is allowed to interact with an oracle (domain expert), asking for the similarity between a certain set of chosen items. We study the query and computational complexities of clustering in this framework. We prove that having a few of such similarity queries enables one to get a polynomial-time approximation algorithm to an otherwise conjecturally NP-hard problem. In particular, we provide algorithms for fuzzy clustering in this setting that ask $O(\mathsf{poly}(k)\log n)$ similarity queries and run with polynomial-time-complexity, where n is the number of items. The fuzzy k-means objective is nonconvex, with k-means as a special case, and is equivalent to some other generic nonconvex problem such as non-negative matrix factorization. The ubiquitous Lloyd-type algorithms (or alternating-minimization algorithms) can get stuck at a local minima. Our results show that by making few similarity queries, the problem becomes easier to solve. Finally, we test our algorithms over real-world datasets, showing their effectiveness in real-world applications.

1 Introduction

The k-means objective for clustering is perhaps the most ubiquitous of all unsupervised learning paradigms. It is extremely well studied, with Lloyd's algorithm being the most popular solution method, while the problem in general being NP Hard [11, 7]. A variety of approximate solutions for k-means exists (such as, [20, 3]).

In recent times, there have been efforts to bring in a flavor of active learning in k-means clustering; by allowing the learner to make a limited number of carefully chosen label/membership queries [4, 2, 18]. In this setting, the main objective is to show that a polynomial-time solution exists provided that small number of such queries to an oracle is permitted. Note that, an alternative to membership query (i.e., query of the form "does the i^{th} element belong to the j^{th} cluster") is the same-cluster/similarity query (i.e., "do elements i and j belong to the same cluster"). Given representatives from each cluster is available, one can simulate any membership query with at most k same cluster queries. Hence whatever is achievable with membership queries, can be also achieved by same cluster queries by asking at most k times as many queries [4].

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

How realistic is to allow the learner to make such label queries, and how realistic is the oracle assumption? It turns out that many clustering tasks, primarily entity-resolution, are being delegated to crowdsourcing models. One can easily assume that the crowd is playing the role of an oracle here. It is natural in these models to assign crowd workers with sub-tasks, which can either be answering the membership query or the same cluster query [5, 49, 50, 38, 23].

Fuzzy k-means or soft k-means is a generalized model of the k-means objective that covers lot more grounds being applicable to dataset where datapoints show affinity to multiple labels, the clustering criteria are vague and data features are unavailable [8]. In fuzzy clustering, instead of an element being part of one fixed cluster, it is assumed that it has a membership value between 0 and 1 to each of the clusters. The objective of the clustering is to recover all of the membership values for each element or datapoint. This relaxation in membership values from $\{0,1\}$ to $[0,1]$ makes the fuzzy k-means an attractive model for overlapping clusters [52], multilabel classification [30] and numerous other applications such as, pattern recognition [8], computational biology [6], bioinformatics [47], image segmentation [1, 19] and so on.

What is more interesting is that solving fuzzy clustering is equivalent to approximate solution of nonnegative symmetric matrix factorization problems [22]. Given such applications, it is somewhat surprising that the theoretical results known about fuzzy clustering is quite limited. As is even the computational tractability of fuzzy k-means is only conjecturally known. In fact algorithmic study of fuzzy k-means algorithm seems to have been only recently started [9, 10, 31].

Membership vs. similarity queries. As mentioned above, in the hard-clustering case, one can simulate any membership query with at most k same-cluster (or, similarity) queries [4]. In fuzzy clustering, we are given a set of n vectors (items) in the d-dimensional Euclidean space, and we are tasked the problem of determining the k-dimensional *membership vector* for each of the items. The clustering algorithm has access to the set of vectors, and can also make queries about the underlying soft clustering. As in hard-clustering, these queries can take at least two forms. Specifically, membership queries refer to the existence of a *membership-oracle* which whenever queried, replies with the membership value of the i^{th} item to the j^{th} cluster, i.e., the j^{th} entry of the i^{th} membership vector. One could argue that querying the membership values, and specifically, the existence of such a membership-oracle, is not realistic, and often impractical in real-world settings since it requires knowledge of the relevant clusters. Instead, the similarity query model that takes two or three elements as input and ask *"How similar are all these elements?"* is much easier to implement in practice. Accordingly, in a similar vein to hard-clustering, we show that membership queries can be simulated by similarity queries, measured by the inner-product between pairs (and triplets) of membership vectors. Finally, as will be explained in more detail, the responses of the oracle or the target solution do not need to correspond to the optimum solution of the fuzzy k-means objective, but instead need to satisfy a few mild conditions. Accordingly, the target solution can represent noisy/quantized version of the optimum solution of the fuzzy k-means objective function making our set-up highly practical in crowdsourcing settings.

A natural question that one can ask about fuzzy clustering following the lead of [4] is how many membership/similarity queries one may ask so that the clustering can be efficiently solved with a good approximation of the target solution? In this paper, our main motivation is to answer this rather basic question. It has to be noted that revealing a fractions of labels during clustering algorithm in fuzzy k-means has been studied with some heuristics in [42], but not only the model was different from ours, a rigorous theoretical study was also lacking in the past literature.

Our contributions and techniques. We design several algorithms and analyze their query and computational complexities. Specifically, the first algorithm we propose works in two phases: in the first phase, we estimate the center of each cluster using *non-adaptive* queries, and in the second phase we recover the membership of each element to each cluster using the estimated centers obtained in the first phase. We show with this algorithm it is possible to get an approximation of the membership vectors by making only $O(\text{poly}(k/\beta)\log n)$ queries, where β is ratio of the volume of the smallest cluster to the average volume, informally speaking. Then, we show that by allowing one to ask *adaptive/sequential* queries in the first phase, the polynomial dependence on β of the query complexity can be further improved. Despite the strong dependency of the query complexity on β, for a large regime of parameters, our query complexity is non-trivial and sublinear in n. Interestingly, for the special case of $k = 2$, we can completely get rid of the dependency of the query complexity on β, and it suffices to ask $O(\log^2 n)$ queries only.

For our theoretical results to hold, we make an assumption on the dataset that quantifies the clusterability of the data. We would like to emphasize that our assumption is only needed for deriving the theoretical guarantees; our algorithms are demonstrated to perform well on real-world datasets irrespective of whether our assumption holds or not. The computational complexity of all the algorithms scales as d times the number of queries.

It is to be noted that the analysis of the aforementioned algorithms are technically much more demanding than solving the analogous query complexity problem for hard k-means. For the case of hard k-means, a general principle would be to sample a small set of items and estimate the centre of the largest cluster from the samples (see, [4]). Subsequently all the items closest to the estimated centre can be assigned to it and the cluster can be removed from consideration. One can iterate over this algorithm to find all the clusters. However the size of a cluster is not well-defined in the fuzzy setting; and also it is not possible to assign items to a cluster and remove them from consideration, since they can very well have nonzero memberships to other clusters.

To tackle these difficulties, we have to use more challenging technical tools and analysis methods than the hard k-means literature. In a nutshell, since at any step of the algorithm we cannot remove items from consideration, we propose sophisticated random sub-sampling algorithms that explore the space of items efficiently. Specifically, having reliable approximations for the centers and memberships of some processed clusters, we partition the space of items into bins that have the same approximated sum of membership weights to unprocessed clusters. Sampling equally from every bin ensures that we obtain enough samples from items which have high membership weight to those clusters which have not yet been approximated. Also, contrary to hard k-means where a simple (unbiased) empirical mean estimator is used to approximate the means, in the fuzzy setting, we end up with a form of self-normalized importance sampling estimators. While these estimators are biased, we managed to bound their deviations in an elegant way using concentration inequalities. We tested our algorithms over both synthetic and real-world datasets, and illustrate the tradeoff and effect of different values of β and k on the estimation error and query complexity.

Related work. Clustering with limited label-queries have mostly followed two different lines of works. In the first line of work, queries were used to make the objective based clustering, such as k-means, polynomial time solvable [4, 2, 18, 12, 13, 14]. The problem being considered here is most closely related to this line of work, albeit, instead of hard k-means, we focus on a soft objective.

In the second line of work, instead of an objective based clustering, it is assumed that there exists a ground-truth clustering, and by label-querying, one gets a (noisy) answer from an oracle about this ground-truth. The aim of this line of work has been to give statistical-computational guarantees on recovery of the ground truth [36, 37, 35, 2, 15, 46, 43]. The work in this canon that is closest to our problem is [28], where an overlapping clustering groundtruth was considered. At a high level, we also consider overlapping clusters, however, focus on objective-based clustering therein.

It is worth noting that clustering with same cluster, or other label queries, has found connections with and application to correlation clustering [15, 43], community detection [37, 17], heavy-hitters [44] and possibly other areas, that we might have missed. Our problem can be seen as approximately equivalent to non-negative symmetric matrix factorization [40] with limited access to the entries of the factors (via queries) [22].

Organization. The rest of the paper is organized as follows. In Section 2 we present our model and the learning problem. Then, in Section 3 we present our algorithms and their theoretical guarantees. The main theoretical results can be found in Theorems 1,2,3. **Due to space limitation, proofs, some pseudocodes, experimental results over both synthetic and real-world datasets, and conclusions are relegated to the supplementary material.**

2 Model formulation and background

2.1 The optimization problem

We start by describing the hard k-means problem, and then move forward to the fuzzy k-means problem (soft clustering). Let $\mathcal{X} \subset \mathbb{R}^d$ be a set of d-dimensional points, with $|\mathcal{X}| = n$. We denote this set of vectors by $\{\boldsymbol{x}_i\}_{i=1}^n$, and assume without loss of generality that $\boldsymbol{x}_i \in \mathcal{B}(\mathbf{0}, \mathsf{R})$, for any $i \in [n]$, where $\mathcal{B}(\boldsymbol{a}, \mathsf{R})$ is the L_2-Euclidean ball of radius $\mathsf{R} \in \mathbb{R}_+$ centered around $\boldsymbol{a} \in \mathbb{R}^d$. Let

$\mathcal{C}_{\mathcal{X}} \triangleq \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ be a clustering (or, partition) of $\mathcal{X}$, where $k \in \mathbb{N}$ is the number of clusters. We say that a clustering $\mathcal{C}_{\mathcal{X}}$ is center-based if there exists a set of centers $\boldsymbol{\mu} \triangleq \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k\} \subset \mathbb{R}^d$ such that the clustering is induced by the Voronoi diagram over those center points. The objective function of hard k-means is

$$\mathsf{J}_{\mathsf{km}}(\mathcal{X}, \boldsymbol{\mu}, \mathsf{U}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \mathsf{U}_{ij} \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|_2^2, \tag{1}$$

where $\mathsf{U} \in \{0,1\}^{n \times k}$ is the partition matrix, i.e., the membership weight U_{ij} is 1 if $\boldsymbol{x}_i \in \mathcal{C}_j$, and 0 otherwise. In hard clustering each data point is assigned exactly to one cluster and every cluster must contain at least one data point. The goal of k-means is to minimize (1) w.r.t. $(\boldsymbol{\mu}, \mathsf{U})$.

In soft-clustering there is no partitioning of $\mathcal{X}$. Instead, we describe the k^{th} cluster of a fuzzy clustering as a vector of the fractions of points assigned to it by the membership function. In particular, the memberships weights take on real-valued numbers in $[0,1]$ rather than $\{0,1\}$, as for hard clustering. The fuzzy k-means problem is defined as follows.

Definition 1 (Fuzzy k-means)**.** *Let $\alpha \geq 1$ and $k \in \mathbb{N}$. The fuzzy k-means problem is to find a set of means $\boldsymbol{\mu} = \{\boldsymbol{\mu}_\ell\}_{\ell \in [k]} \subset \mathbb{R}^d$ and a membership matrix $\mathsf{U} \in [0,1]^{n \times k}$ minimizing*

$$\mathsf{J}_{\mathsf{fm}}(\mathcal{X}, \boldsymbol{\mu}, \mathsf{U}) \triangleq \sum_{i=1}^{n} \sum_{j=1}^{k} \mathsf{U}_{ij}^{\alpha} \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|_2^2, \tag{2}$$

subject to $\sum_{j=1}^{k} \mathsf{U}_{ij} = 1$ and $0 < \sum_{i=1}^{n} \mathsf{U}_{ij} < n$, for all $i \in [n]$ and $j \in [k]$.

The parameter α is called the *fuzzifier*, and is not subject to optimization. It can be shown that when $\alpha \to 1$, the fuzzy k-means solution coincides with that of the hard k-means problem. Similarly to the classical k-means problem, it is easy to optimize the means or memberships of fuzzy k-means, assuming the other part of the solution is fixed. Specifically, given a set of means $\boldsymbol{\mu}$, it can be shown that U in (3) below is the optimal membership weights minimizing the cost in (2) (see, e.g., [8][Theorem 11.1]). On the other hand, given a set of membership weights U, the weighted centers in (4) below are the optimal minima.

$$\mathsf{U}_{ij} = \left[\sum_{\ell=1}^{k} \left(\frac{\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|}{\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_\ell\right\|}\right)^{2/(\alpha-1)}\right]^{-1} \tag{3}$$

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{n} \mathsf{U}_{ij}^{\alpha} \boldsymbol{x}_i}{\sum_{i=1}^{n} \mathsf{U}_{ij}^{\alpha}}, \tag{4}$$

for all $i \in [n],\ j \in [k]$. Iterating these solutions is known as the Lloyd's algorithm [32], which provably lands on a local optima. It is well-known, however, that finding the optimal solution (i.e., global minimia) to the k-means problem is is NP-hard in general [33, 48]. While the same result is unknown for the fuzzy formulation, it is strongly believed to be the case (e.g., [9, 16]).

Finally, although several validity indexes for the performance of fuzzy k-means solutions have been proposed in the literature (see, e.g., [24, 25]), the Xie–Beni [51] is the most widely used in the literature. This measure is defined as follows[1]

$$\mathsf{XB}(\mathcal{X}, \boldsymbol{\mu}, \mathsf{U}) \triangleq \frac{\mathsf{J}_{\mathsf{fm}}(\mathcal{X}, \boldsymbol{\mu}, \mathsf{U})}{nk \cdot \min_{i \neq j} \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2^2}. \tag{5}$$

For the rest of this paper, we denote a clustering by $\mathcal{P} \triangleq (\boldsymbol{\mu}, \mathsf{U})$, and we replace $\mathsf{XB}(\mathcal{X}, \boldsymbol{\mu}, \mathsf{U})$ by $\mathsf{XB}(\mathcal{X}, \mathcal{P})$. Accordingly, the optimal solution to the fuzzy k-means problem (Defintion 1) is denoted by $\mathcal{P}^{\star}$. Finally, we say that a clustering $\mathcal{P}$ is *consistent center-based* if:

- Given membership weights U, the centers $\boldsymbol{\mu}$ are given by (4).
- The membership weights U are monotone, i.e., if $\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|_2 \leq \left\|\boldsymbol{x}_\ell - \boldsymbol{\mu}_j\right\|_2$ then $\mathsf{U}_{ij} \geq \mathsf{U}_{\ell j}$, for any $i, \ell \in [n]$ and $j \in [k]$.

The first condition appears also in [4]. The second condition excludes inconsistent membership weights U. In particular, as is shown in [16], and in Lemma 5 in the appendix, both conditions

[1] In (5) we divide by nk and not just by n, as in [51], where k was consider a fixed parameter. If k is allowed to grow (e.g., with n), then our definition makes more sense, as in general the numerator in (5) can grow with k.

are satisfied by the optimal solution $\mathcal{P}^\star$, as well as by (4). We would like to clarify that the above assumptions are *only* required for the theoretical analysis; Our proposed algorithms can be implemented regardless of these assumptions. As is shown in Appendix F, in real-world applications, our algorithm works well even if these assumptions do not hold.

2.2 Domain expert/oracle

Our goal is to construct *efficient* algorithms with the aid of a domain expert who can give an approximate solution to the fuzzy k-means problem. Specifically, given a clustering $\mathcal{P}$, a $\mathcal{P}$-oracle is a function $\mathcal{O}_{\mathcal{P}}$ that answers queries consistent with $\mathcal{P}$. One can think of such an oracle as a user that has some idea about its desired solution, enough to answer the algorithm's queries. Importantly, note that as long as $\mathcal{P}$ is a consistent center-based clustering, it need not be the optimal solution $\mathcal{P}^\star$ minimizing (2). The algorithm then tries to recover (or, approximate) $\mathcal{P}$ by querying the oracle a small number of times. For hard clustering, the same-cluster query oracle model (i.e., "do elements i and j belong to the same cluster") was considered in [4]. Using a certain number of such same-cluster queries and the data points $\mathcal{X}$, the clustering $\mathcal{P}$ can be found efficiently [4], while finding the optimal solution without the aid of an oracle is NP-hard. For the fuzzy formulation, one can think of analogous oracle models, and in this paper, we focus on the following.

Definition 2 (Membership-Oracle)**.** *A membership query asks the membership weight of an instance* $\boldsymbol{x}_i$ *to a cluster* j*, i.e.,* $\mathcal{O}_{\text{fuzzy}}(i,j) = \mathsf{U}_{ij}$.

Basically we ask the oracle for the membership weight of $\boldsymbol{x}_i$ to cluster $\mathcal{C}_j$. Using a certain amount of such queries and the data set $\mathcal{X}$ we would like to approximate the solution for fuzzy k-means problem. As discussed in the Introduction, one could argue that querying the membership values, and specifically, the existence of such a membership-oracle, is not realistic, and often impractical in real-world settings since it requires knowledge of the relevant clusters. Instead, the following similarity query models are easy to implement in practice.

Definition 3 (Similarity-Oracle)**.** *A fuzzy pairwise similarity query asks the similarity of two distinct instances* $\boldsymbol{x}_p$ *and* $\boldsymbol{x}_q$ *i.e.,* $\mathcal{O}_{\text{sim}}(p,q) = \langle \mathsf{U}_p, \mathsf{U}_q \rangle$*. A fuzzy triplet similarity query asks the similarity of three distinct instances* $\boldsymbol{x}_p, \boldsymbol{x}_q, \boldsymbol{x}_r$*, i.e.,* $\mathcal{O}_{\text{triplet}}(p,q,r) = \sum_{t\in[k]} \mathsf{U}_{pt}\mathsf{U}_{qt}\mathsf{U}_{rt}$.

We show in Appendix H, however, that each membership query can be simulated with a few similarity queries. From a theoretical perspective, membership queries can be simulated with only fuzzy pairwise similarity queries if there exists many *pure* instances (instances that have a membership weight of 1 to a particular cluster in the dataset), see Lemma 14 in appendix. Unfortunately this strong condition is necessary as well. But it turns out that by using a few triplet similarity queries at the beginning, we can resolve this issue. Specifically, we prove the following result, informally stated, that is interesting in its own right.

Lemma 1 (*Membership to Similarity Queries Reduction, Informal*)**.** *Suppose there exists an algorithm* $\mathcal{A}_n$ *with time complexity* T_n *that uses* m *membership queries to provide some guaranteed solution of the fuzzy* k*-means problem. In that case, under mild conditions on the dataset, the same guarantee can be obtained with* $O(k^3)$ *fuzzy triplet similarity queries and* $O(km)$ *fuzzy pairwise similarity queries, with a time complexity of* $\mathsf{T}_n + O(mk + k^3)$.

Therefore, we can express everything in terms of the more natural notion of similarity queries and hence, we use membership queries for the rest of the paper to make the representation of our algorithms simpler.

Noisy/Inaccurate Oracle. Our algorithms work even with inaccuracies in the oracle answer, provided the deviation is small. Under the assumption that such inaccuracies are random, in particular, a repetition algorithm can remedy such errors. Specifically, consider the case where $\mathcal{O}_{\text{fuzzy}}(i,j) = \mathsf{U}_{ij} + \zeta_{ij}$, where ζ_{ij} is a zero-mean random variable, and assume the errors to be independent, i.e., the answers are collected from independent crowd-workers. Then, in Appendix G we show how one can convert any "successful" solver for the noiseless problem into a solver for the noisy problem; generally speaking, by repeating the noiseless solver for T consecutive steps using the noisy responses, and then averaging the results to obtain the final estimate. Note that, the technique used to handle the subsequent steps are also applicable to *any bounded error*. For simplicity of exposition and clarity, we opted to focus on the noiseless setting given in Definition 2.

2.3 Objective and learning problem

The goal of the algorithm is to find an approximated clustering that is consistent with the answers given to its queries. We have the following definition.

Definition 4 (Oracle-Based Solver)**.** *Let $(\mathcal{X}, \mathcal{P})$ be a clustering instance. An algorithm $\mathcal{A}$ is called a $(\epsilon_1, \epsilon_2, \mathsf{Q})$-solver if it can* approximate *$\mathcal{P}$ by having access to $\mathcal{X}$ and making at most Q membership queries to a $\mathcal{P}$-oracle. Specifically, for any $\epsilon_1, \epsilon_2 > 0$, the algorithm outputs $\widehat{\mathcal{P}}$ such that $\max_{j\in[k]} \left\|\boldsymbol{\mu}_j - \widehat{\boldsymbol{\mu}}_j\right\|_2 \leq \epsilon_1$ and $\max_{i\in[n],j\in[k]} |\mathsf{U}_{ij} - \widehat{\mathsf{U}}_{ij}| \leq \epsilon_2$. Such an algorithm is a polynomial $(\epsilon_1, \epsilon_2, \mathsf{Q})$-solver if its time-complexity is polynomial in $(n, k, \epsilon_1^{-1}, \epsilon_2^{-1})$.*

We would like to emphasize that we seek an approximate solution rather than exact one. It is important to note $\mathsf{Q} = O(nk)$ membership queries suffice to recover the $\mathcal{P}$-oracle clustering. Also, with unbounded computational resources one can find the optimal clustering $\mathcal{P}^\star$ without any queries. Nonetheless, in this paper, we would like to have a *sub-linear* dependency of the query complexity on n and a polynomial dependency of the time complexity on n, k. Having an algorithm such as the one in Definition 4, implies the following guarantee, proved in Appendix A.

Lemma 2. *Let $(\mathcal{X}, \mathcal{P})$ be a clustering instance, and $\mathcal{A}$ be a $(\epsilon_1, \epsilon_2, \mathsf{Q})$-solver. Then,*

$$\left|\mathsf{XB}(\mathcal{X}, \mathcal{P}) - \mathsf{XB}(\mathcal{X}, \widehat{\mathcal{P}})\right| \leq \frac{\mathsf{XB}(\mathcal{X}, \mathcal{P}) \cdot O(\epsilon_1) + O(\epsilon_2)}{\min_{i\neq j} \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2^2} + o\left(\frac{\epsilon_1^2 + \epsilon_2^2}{\min_{i\neq j} \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2^2}\right). \tag{6}$$

The lemma above shows that if ϵ_1 and ϵ_2 are small enough, then the XB measure associated with an $(\epsilon_1, \epsilon_2, \mathsf{Q})$-solver is “close" to the XB measure associated with $\mathcal{P}$. Note, however, that the r.h.s. of (6) depends inversely on $\min_{i\neq j} \left\|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\right\|_2^2$. Accordingly, if for some clustering instance $(\mathcal{X}, \mathcal{P})$ this quantity is too small compared to the estimation error, then the difference $|\mathsf{XB}(\mathcal{X}, \mathcal{P}) - \mathsf{XB}(\mathcal{X}, \widehat{\mathcal{P}})|$ is uncontrolled. This is reasonable since when clusters are “close", the estimation task gets harder.

Finally, we introduce certain notations that will be needed for the presentation of our main results. Given a set of points $\mathcal{X}$ and $\boldsymbol{v} \in \mathbb{R}^d$, let $\pi_{\boldsymbol{v}} : [n] \to [n]$ be the permutation defined on $[n]$ such that

$$\left|\left|\boldsymbol{v} - \boldsymbol{x}_{\pi_{\boldsymbol{v}}(i)}\right|\right|_2 \leq \left|\left|\boldsymbol{v} - \boldsymbol{x}_{\pi_{\boldsymbol{v}}(j)}\right|\right|_2, \tag{7}$$

for all $i < j \in [n]$. To wit, $\pi_{\boldsymbol{v}}$ is the ranking of the elements in $\mathcal{X}$ when they are sorted in ascending order w.r.t. to their distance from $\boldsymbol{v}$. Then, we let $\gamma \in \mathbb{R}_+$ be the maximum positive number such that for all $j \in [k]$,

$$\pi_{\widehat{\boldsymbol{\mu}}} = \pi_{\boldsymbol{\mu}_j}, \quad \forall \widehat{\boldsymbol{\mu}} \in \mathcal{B}(\boldsymbol{\mu}_j, \gamma). \tag{8}$$

Intuitively, if one has reliable estimates for the centers, then the ordering of $\mathcal{X}$ w.r.t. these estimated centers is the same as the ordering w.r.t. the target centers. Next, we define the *cluster membership size* associated with $\mathcal{P}$ by $\sum_{i\in[n]} \mathsf{U}_{ij}$, for all $j \in [k]$, and a parameter $\beta \in \mathbb{R}_+$ by $\beta \triangleq \min_{j\in[k]}\ kn^{-1} \cdot \sum_{i\in[n]} \mathsf{U}_{ij}$. Accordingly, in terms of β, the minimum cluster membership size is at least $\beta n/k$. Notice that β must be less than unity since $\sum_{i,j} \mathsf{U}_{ij} = n$, and there must exist at least one cluster whose membership size is at most n/k. We mention that β was introduced in [18] as well.

3 Algorithms and theoretical guarantees

3.1 Two-Phase algorithm

We start by providing a non-formal description of our algorithm, whose pseudocode is given in Algorithms 1–3. Specifically, in the first three steps of Algorithm 1 we approximate the centers of all clusters. To that end, we start by sampling at random m elements from $[n]$ with replacement, and denote the obtained samples by $\mathcal{S}$. The value of m is specified in Theorem 1 below. Subsequently, we query the membership weights U_{ij}, for all $i \in \mathcal{S}$ and $j \in [k]$. Using these queries, we approximate the center of every cluster using the estimator defined in Step 3 of Algorithm 1. Note that this estimator is the most natural candidate when one knows the membership weights only for a set of sub-sampled elements. Although this algorithm is quite intuitive, it suffers from the natural drawback that if the size of a particular cluster is small, many samples are needed for a reliable estimate of its center.

Algorithm 1 Parallel algorithm for approximating $\mathcal{P}$.

```
Input: $\mathcal{X}$, $\mathcal{O}_{\mathsf{fuzzy}}$, $\alpha$, $m$, and $\eta$.
Output: An estimate $\widehat{\mathcal{P}}$.
1: Let $\mathcal{S}$ be a set of $m$ elements sampled at random from $[n]$ with replacement.
2: Query $\mathcal{O}_{\mathsf{fuzzy}}(i,j), \forall i \in \mathcal{S}, j \in [k]$.
3: Compute $\widehat{\boldsymbol{\mu}}_j = \frac{\sum_{i\in\mathcal{S}} \mathsf{U}_{ij}^{\alpha}\boldsymbol{x}_i}{\sum_{i\in\mathcal{S}} \mathsf{U}_{ij}^{\alpha}}, \ \forall j \in [k]$.
4: for $j = 1, 2, \ldots, k$ do
5:     $\{\widehat{\mathsf{U}}_{ij}\}_{i=1}^{n} \leftarrow$MEMBERSHIP$(\mathcal{X}, \widehat{\boldsymbol{\mu}}_j, \alpha, \eta)$.
6:     $\widehat{\mathsf{U}}_{ij} \leftarrow \widehat{\mathsf{U}}_{ij} + \frac{1-\sum_{j=1}^{k}\widehat{\mathsf{U}}_{ij}}{k}, \ \forall i \in [n]$.
7: end for
```

Algorithm 2 MEMBERSHIP$(\mathcal{X}, \widehat{\boldsymbol{\mu}}_j, \alpha, \eta)$

```
Input: $\mathcal{X}$, and $\mathcal{O}_{\mathsf{fuzzy}}$.
Output: An estimate $\{\widehat{\mathsf{U}}_{ij}\}_{i=1}^{n}$.
Find $\pi_{\widehat{\boldsymbol{\mu}}_j}$ w.r.t. $\mathcal{X}$.
for $s = 0, 1, \ldots, 1/\eta$ do
    Find $\ell_s = \mathsf{argmax}_{i\in[n]} \mathsf{U}_{\pi_{\widehat{\boldsymbol{\mu}}_j}(i)j} \geq s\eta$ using BINARYSEARCH$(\mathcal{X}, \pi_{\widehat{\boldsymbol{\mu}}_j}, s\eta)$.
end for
for $s = 0, 1, 2, \ldots, 1/\eta$ do
    for $i = \ell_s, \ell_s - 1, \ldots, , \ell_{(s+1)} + 1$ do
        Set $\widehat{\mathsf{U}}_{\pi_{\widehat{\boldsymbol{\mu}}^j}(i)j} = s\eta$.
    end for
end for
```

Next, using the approximated centers, we approximate the membership weights of every element up to a precision of η by using Algorithm 2. To that end, in the first step of this algorithm, we sort the elements of $\mathcal{X}$ in an ascending order according to their distance from the estimated center. Accordingly, if the estimation error of the centers ϵ satisfies $\epsilon \leq \gamma$, then (8) implies that the obtained ordering of elements after sorting from the estimated centers is same as the true ordering of the elements obtained by sorting according to distance (in ascending order) from the exact centers (instead of approximated centers). Furthermore, the fact that $\mathcal{P}$ is a consistent center-based clustering implies that for any $j \in [k]$ the membership weights $\mathsf{U}_{\pi_{\widehat{\boldsymbol{\mu}}_j(i)}j}$ are non-increasing in i. Therefore, for any $j \in [k]$ using the binary search in Algorithm 3, we find the maximum index ℓ_s such that $\mathsf{U}_{\pi_{\widehat{\boldsymbol{\mu}}_j(\ell_s)}j}$ is larger than $s\eta$, for $s \in \{0, 1, \ldots, 1/\eta\}$. This will create an η-spaced grid, and we approximate each membership weight in accordance to the cell it falls into. Specifically, we assign $\widehat{\mathsf{U}}_{\pi_{\widehat{\boldsymbol{\mu}}_j(t)}j} = s\eta$, for all $t \in [\ell_s, \ell_s - 1, \ldots, \ell_{s+1} + 1]$. Clearly, the estimation error is at most η. Overall, each binary search step takes $O(\log n)$ queries, and therefore the total query complexity of this step is $O(\log n/\eta)$. Finally, Step 6 transforms $\widehat{\mathsf{U}}_{ij}$ to ensure that $\sum_{i=1}^{n} \widehat{\mathsf{U}}_{ij} = 1$, for $j \in [k]$. We have the following result, the detailed proof of which is delegated to Appendix C.

Algorithm 3 BINARYSEARCH$(\mathcal{X}, \pi, x)$

```
Input: $\mathcal{O}_{\mathsf{fuzzy}}$.
Set $\mathsf{low} = 1$ and $\mathsf{high} = n$.
while $\mathsf{low} \neq \mathsf{high}$ do
    Set $\mathsf{mid} = \lceil(\mathsf{low} + \mathsf{high})/2\rceil$.
    Query $\mathcal{O}_{\mathsf{fuzzy}}(\pi(\mathsf{mid}), j)$.
    if $\mathsf{U}_{\pi(\mathsf{mid})j} \geq x$ then
        Set $\mathsf{low} = \mathsf{mid}$.
    else
        Set $\mathsf{high} = \mathsf{mid} - 1$.
    end if
end while
```

Theorem 1. *Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, and recall the definitions of $\gamma \in \mathbb{R}_+$ and $\beta \in (0,1)$. Pick $\epsilon \leq \gamma$, $\eta \in (0,1)$, and $\delta \in (0,1)$. Take as input to Algorithm 1 $m = c\left(\frac{\mathsf{R}k^{\alpha}}{\epsilon\beta^{\alpha}}\right)^4 \log\frac{k}{\delta}$, for some constant $c > 0$. Then, with probability at least $1-\delta$, Algorithm 1 is $(\epsilon, \eta, \mathsf{Q})$-solver using $\mathsf{Q} = O\left(\frac{\mathsf{R}^4 k^{4\alpha+1}}{(\epsilon\beta^{\alpha})^4}\log\frac{k}{\delta} + \frac{k\log n}{\eta}\right)$ membership queries. Furthermore, Algorithm 1 time-complexity is of order $O\left(kn\log n + \frac{d\mathsf{R}^4 k^{4\alpha+1}}{(\epsilon\beta^{\alpha})^4}\log\frac{k}{\delta} + \frac{k\log n}{\eta}\right)$.*

Proof Sketch of Theorem 1. To prove Theorem 1 we first show that with high probability, Steps 1–4 in Algorithm 1 output an ϵ-approximation for all centers. To that end, we first show in Lemma 6 that for a given cluster $j \in [k]$, with probability $1-\delta$, $\left\|\boldsymbol{\mu}_j - \widehat{\boldsymbol{\mu}}_j\right\|_2^2 \leq \frac{4\mathsf{R}^2}{\mathsf{Y}^2}\sqrt{\frac{c}{m}\log\frac{1}{\delta}}$, where $\mathsf{Y} \triangleq \min_{j\in[k]} \frac{1}{n}\sum_{i\in[n]} \mathsf{U}_{ij}^{\alpha}$. This result follows by noticing that both the numerator and denominator of $\widehat{\boldsymbol{\mu}}_j$ are unbiased estimates of the corresponding numerator and denominator of $\boldsymbol{\mu}_j$ in (4). Representing $\widehat{\boldsymbol{\mu}}_j$ as a sum of the true underlying center $\boldsymbol{\mu}_j$ and a certain error term, the generalized Hoeffding's inequality in Lemma 4 implies the above estimation error guarantee. In light of this result, using Hölder's inequality and the union bound we show in Lemma 8 that $\left\|\widehat{\boldsymbol{\mu}}_j - \boldsymbol{\mu}_j\right\|_2 \leq \epsilon$, *for all* $j \in [k]$, if m is as given in Theorem 1. Note that this centers estimation algorithm requires a total of km membership-oracle queries. Next, conditioned on the above proxy of any given center, we show in

Lemma 7 that Algorithm 2 estimates the corresponding membership weights within a certain range of error with high probability. Specifically, for a given cluster j, we prove that Algorithm 2 outputs $\widehat{\mathsf{U}}_{ij}$, for $i \in [n]$, such that $0 \leq \mathsf{U}_{ij} - \widehat{\mathsf{U}}_{ij} \leq \eta$ and $\widehat{\mathsf{U}}_{ij} \in \{0, \eta, 2\eta, \ldots, 1\}$, $\forall i \in [n]$, using $O\left(\log n/\eta\right)$ queries to the membership-oracle. Roughly speaking, this result follows from the arguments presented in the paragraph preceding the statement of Theorem 1. Finally, it is evident from the above that the total number of membership-oracle queries is $O(km + k\log n/\eta)$. □

Note that both the query and computational complexities in Theorem 1 exhibit an exponential dependency on the "fuzzifier" α. As mentioned in Subsection 2.1, this parameter is not subject to optimization and, usually, practitioners pick $\alpha = 2$. Nonetheless, there has been a lot of work on this topic; one notable example is [27] which presents an initial theoretical approach towards the investigation of this question. It seems that the value of α is influenced by the structure of the dataset, i.e., the number of clusters and the distribution of each cluster. Currently, a definite answer on how to pick α is unknown. Indeed, since there is no baseline, one cannot decide which value of α is preferable. Note also that despite the fact that our query complexity naturally depend on α, it cannot be used to determine the "best" value of α. Assuming the existence of a ground truth, in Appendix F we compare the clustering accuracy for different values of α on a real-world dataset.

3.2 Sequential algorithm

As mentioned in the previous subsection, the main drawback faced by Algorithm 1 is its undesirable strong dependency on β. The algorithm proposed in this section remedies this drawback to some extent. The algorithm works as follows. In the first four steps of Algorithm 1, we randomly draw m samples from $[n]$, and query the membership weights that correspond to all these samples. Using these samples we estimate the center of *largest* cluster t_1, having the maximal total membership weight $\sum_{i\in\mathcal{S}} \mathsf{U}_{ij}^{\alpha}$, among the k clusters. Corollary 4 in Section D specifies the number of such elements needed to have a reliable estimate. Then, using this approximated center, we estimate the membership of every element in cluster t_1 using Algorithm 2 such that the estimate is a multiple of η_1 and further, the estimate has an additive error of at most η_1, just as we did in the two-phase algorithm. For the rest of the clusters, we follow the sequential algorithm in Steps 5–13 of Algorithm 4, the guarantees of which are characterized in the following result:

Theorem 2. *Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, and recall the definitions of $\gamma \in \mathbb{R}_+$ and $\beta \in (0,1)$. Take $m = c\left(\frac{\mathsf{R}k^{\alpha}}{\epsilon}\right)^4 \log\frac{2k}{\delta}$, $r = \frac{c\mathsf{R}^4k^{4\alpha}}{\epsilon^4\beta^{4\alpha-4}}\log\frac{4k}{\eta_1\delta}$, for some constant $c > 0$, and any $\delta \in (0,1)$. Then, with probability at least $1-\delta$, Algorithm 4 is $(\epsilon, \eta_2, \mathsf{Q})$-solver using $\mathsf{Q} = O\left(\frac{\mathsf{R}^4k^{4\alpha+1}}{\eta_1\epsilon^4\beta^{4\alpha-4}}\log\frac{4k}{\eta_1\delta} + \frac{k\log n}{\eta_1} + \frac{k\log n}{\eta_2}\right)$ membership queries, for any chosen $\epsilon \leq \gamma$ and $\eta_2 \leq \eta_1 \leq \frac{1}{k}\left(1 - \frac{\beta}{k}\right)$. Furthermore, Algorithm 4 time-complexity is of order $O\left(\frac{d\mathsf{R}^4k^{4\alpha+1}}{\eta_1\epsilon^4\beta^{4\alpha-4}}\log\frac{4k}{\eta_1\delta} + \frac{k\log n}{\eta_1} + \frac{k\log n}{\eta_2} + kn\log n\right)$.*

The proof of Theorem 2 follows from a technically complicated induction. Due to space limitations, the details have been delegated to Appendix D and below, we provide only a short sketch.

Proof Sketch of Theorem 2. Suppose that we have approximated the means of $\ell \leq k$ clusters, indexed by $\mathcal{T}_\ell \triangleq \{t_1, t_2, \ldots, t_\ell\}$, with an estimation error at most $\epsilon \leq \gamma$, and further approximated the membership weights of every element in $\mathcal{X}$ for all clusters indexed by $\mathcal{T}_\ell$, up to an additive error η_1. The leftover clusters, indexed by $[k] \setminus \mathcal{T}_\ell$, are the *unprocessed clusters*. The above implies that we can also approximate the sum $\sum_{j\in[\ell]} \mathsf{U}_{it_j}$, for each $i \in \mathcal{X}$, up to an additive error $\ell\eta_1$. Note that, since $\widehat{\mathsf{U}}_{it_j}$ is a multiple of η_1 for all $j \in [\ell]$, $\sum_{j\in[\ell]} \widehat{\mathsf{U}}_{it_j}$ must also be a multiple of η_1. Subsequently, we partition $\mathcal{X}$ into bins, where the elements in a particular bin have the same approximated sum of membership weights. Formally, let $\Sigma \triangleq \{0, 1, \ldots, 1/\eta_1\}$. For each $s \in \Sigma$, we define $\mathcal{X}_s \triangleq \{i \in [n] : \sum_{j\in[\ell]} \widehat{\mathsf{U}}_{it_j} = s\eta_1\}$ to be the s^{th} bin. Then, in Steps 9-10 of Algorithm 4, we randomly sample $\min(r, |\mathcal{X}_s|)$ elements from each bin with replacement, and denote the resulted sub-sampled set by $\mathcal{Y}_s$. The value of r is specified in Theorem 2. Sampling equally from every bin ensures that we obtain enough samples from elements which have high membership weight to the unprocessed clusters, i.e., $\sum_{j\in[k]\setminus[\ell]} \mathsf{U}_{it_j}$ is large. Given these sub-sampled sets, we query the membership weights U_{ij}, for all $i \in \cup_{s\in\Sigma}\mathcal{Y}_s$ and $j \in [k] \setminus \mathcal{T}_\ell$. Using these queries we would like to approximate the center of some unprocessed cluster. We choose the unprocessed cluster having the largest total membership size, weighted by the size of each bin. Mathematically, the index of the $(\ell+1)^{\mathsf{th}}$ cluster, and its approximated center are given as follows,

$$t_{\ell+1} \triangleq \underset{j\in[k]\setminus\mathcal{T}_\ell}{\mathsf{argmax}} \sum_{s\in\Sigma}\frac{|\mathcal{X}_s|}{r}\sum_{i\in\mathcal{Y}_s}\mathsf{U}_{ij}^{\alpha} \quad \text{and} \quad \widehat{\boldsymbol{\mu}}_{t_{\ell+1}} \triangleq \frac{\sum_{s\in\Sigma}\frac{|\mathcal{X}_s|}{r}\sum_{i\in\mathcal{Y}_s}\mathsf{U}^{\alpha}_{it_{\ell+1}}\boldsymbol{x}_i}{\sum_{s\in\Sigma}\frac{|\mathcal{X}_s|}{r}\sum_{i\in\mathcal{Y}_s}\mathsf{U}^{\alpha}_{it_{\ell+1}}}. \tag{9}$$

We choose the number of samples r large enough so that the estimation error is less than ϵ. The above steps are repeated until all unprocessed clusters are exhausted. Finally, once we have approximated all the centers, we apply Algorithm 2 to approximate the membership weight of every cluster up to an additive error of $\eta_2 \leq \eta_1$. □

Algorithm 4 Sequential algorithm for approximating $\mathcal{P}$.

```
Input: $\mathcal{X}$, $\mathcal{O}_{\mathsf{fuzzy}}$, $\alpha$, $m$, $r$, $\eta_1$, and $\eta_2$.
Output: An estimate $\widehat{\mathcal{P}}$.
Let $\mathcal{S}$ be a set of $m$ elements sampled at random from $[n]$ with replacement.
Query $\mathcal{O}_{\mathsf{fuzzy}}(i,j),\ \forall i \in \mathcal{S}, j \in [k]$.
$t_1 = \mathsf{argmax}_{j\in[k]} \sum_{i\in\mathcal{S}} \mathsf{U}_{ij}^{\alpha}$.
Compute $\widehat{\boldsymbol{\mu}}_{t_1} = \frac{\sum_{i\in\mathcal{S}} \mathsf{U}_{it_1}^{\alpha}\boldsymbol{x}_i}{\sum_{i\in\mathcal{S}} \mathsf{U}_{it_1}^{\alpha}}$.
for $\ell = 1, 2, \ldots, k-1$ do
    $\{\widehat{\mathsf{U}}_{i,\ell}\}_{i=1}^{n} \leftarrow$ MEMBERSHIP($\mathcal{X}, \widehat{\boldsymbol{\mu}}_{t_\ell}, \alpha, \eta_1$).
    Set $\mathcal{X}_s \triangleq \{i \in [n] : \sum_{j\in[\ell]} \widehat{\mathsf{U}}_{it_j} = s\eta_1\}$ for $s \in \{0, 1, \ldots, 1/\eta_1\}$.
    for $s = 0, 1, \ldots, 1/\eta_1$ do
        Let $\mathcal{Y}_s$ be a set of $\min(r, |\mathcal{X}_s|)$ elements sampled at random from $\mathcal{X}_s$ with replacement.
    end for
    Compute $t_{\ell+1}$ using (9).
    Compute $\widehat{\boldsymbol{\mu}}_{t_{\ell+1}}$ using (9).
end for
for $j = 1, 2, \ldots, k$ do
    $\{\widehat{\mathsf{U}}_{ij}\}_{i=1}^{n} \leftarrow$ MEMBERSHIP($\mathcal{X}, \widehat{\boldsymbol{\mu}}_j, \alpha, \eta_2$).
    $\widehat{\mathsf{U}}_{ij} \leftarrow \widehat{\mathsf{U}}_{ij} + \frac{1-\sum_{j=1}^{k}\widehat{\mathsf{U}}_{ij}}{k}$, $\forall i \in [n]$.
end for
```

Comparing Theorems 1 and 2 we see that the sequential algorithm has a more graceful dependency on β, the size of the smallest cluster. On the other hand, the query complexity of the two-phase algorithm is better in terms of k (recall that $\eta_1 \leq 1/k$ for the sequential algorithm). Note that the query complexity of the sequential algorithm still has some dependency on β. This stems from the fact that we estimate the membership weights up to a precision of η_1, which might be too large if there exists a cluster j such that $\max_{i\in[n]} \mathsf{U}_{ij} \leq \eta_1$. Accordingly, in this case, all elements will fall into a single bin in the partition, leading to the same drawback the two-phase algorithm suffers from. Finally, we mention that for $k = 2$, we devise an algorithm whose query complexity is completely independent of β. Due to space limitation, the details are given in Appendix E, but we state here our main conclusion and provide a short sketch of the proof.

Theorem 3. *Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, recall the definition of $\gamma \in \mathbb{R}_+$ in* (8)*, and let $k = 2$. Then, with probability at least $1-\delta$, Algorithm 5 with $m = O\left(\left(\frac{\mathsf{R}}{\epsilon}\right)^4 \log\frac{4}{\delta}\right)$ and $r = O\left(\frac{\mathsf{R}^4}{\epsilon^4}\log\frac{2}{\eta\delta}\right)$ is a $(\epsilon, \eta, \mathsf{Q})$-solver using $\mathsf{Q} = O\left(\frac{\mathsf{R}^4 \log n \log(1/\eta\delta)}{\epsilon^4} + \log^2 n + \frac{\log n}{\eta}\right)$ membership queries, for any chosen $\epsilon \leq \gamma$, $\delta \in (0,1)$ and $\eta > 0$. Furthermore, Algorithm 5 time-complexity is of order $O\left(n\log n + \frac{d\mathsf{R}^4 \log n \log(1/\eta\delta)}{\epsilon^4} + \log^2 n + \frac{\log n}{\eta}\right)$.*

We now explain the main ideas behind Algorithm 5. First, let $\mathcal{S}$ be a set of m elements sampled from $[n]$ with replacement. Denote by $t_1 \in \{1,2\}$ the index of the larger cluster and $t_2 \in \{1,2\}$ the index of the smaller cluster, i.e., $\sum_{i\in\mathcal{S}} \mathsf{U}_{it_1}^{\alpha} > \sum_{i\in\mathcal{S}} \mathsf{U}_{it_2}^{\alpha}$. The algorithm works as follows. We start by computing an estimate $\widehat{\boldsymbol{\mu}}_{t_1}$ for the center of the largest cluster t_1. Corollary 4 shows that $O\left((\mathsf{R}/\epsilon)^4 \log(4/\delta)\right)$ membership queries are needed to guarantee that $\left\|\widehat{\boldsymbol{\mu}}_{t_1} - \boldsymbol{\mu}_{t_1}\right\|_2 \leq \epsilon$. Now, the main novelty in Algorithm 5 is a non-trivial querying scheme used to estimate the memberships of the second cluster, U_{it_2}, for all $i \in [n]$, using the center estimated for the first cluster t_1. Note that for the special case of two clusters, querying U_{it_1} reveals U_{it_2} since $\mathsf{U}_{it_1} + \mathsf{U}_{it_2} = 1$. While, in the sequential algorithm, we always approximated the membership weights using bins of fixed size η_1, here, we will approximate U_{it_2} with bins whose size is *data-dependent*. The reason for using variable bin sizes rather than fixed size bins is that with the latter we might ask for many redundant membership queries simply because many bins might be empty. Accordingly, by adaptively choosing the bin size for approximating the membership weights of the smaller cluster t_2, while querying membership weights to the larger cluster t_1, allows us to get around this issue completely; this results in a theoretical guarantee that is completely independent of β.

Algorithm 5 Sequential algorithm for approximating $\mathcal{P}$ with two clusters

```
Input: $\mathcal{X}$, $\mathcal{O}_{\mathsf{fuzzy}}$, $\alpha$, $m$, $r$, and $\eta$.
Let $\mathcal{S}$ be a set of $m$ elements sampled at random from $[n]$ with replacement.
Query $\mathcal{O}_{\mathsf{fuzzy}}(i,j)$ and store $\mathsf{U}_{ij}$ for all $j \in \{1,2\}$.
Choose $t_1 = \mathsf{argmax}_{j\in\{1,2\}} \sum_{i\in\mathcal{S}} \mathsf{U}_{ij}^{\alpha}$.
Compute estimator $\widehat{\boldsymbol{\mu}}_{t_1} = \frac{\sum_{i\in\mathcal{S}} \mathsf{U}_{it_1}^{\alpha}\boldsymbol{x}_i}{\sum_{i\in\mathcal{S}} \mathsf{U}_{it_1}^{\alpha}}$.
Obtain $\mathcal{P}_1, \mathcal{P}_2, \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_q$ and $\widehat{\mathsf{U}}_{it_2}$ for all $i \in [n]$ using MEMBERSHIP2($\mathcal{X}, \widehat{\boldsymbol{\mu}}_{t_1}, \alpha$)
for $q = 1, 2, 3, \ldots, 3\log n$ do
    Let $\mathcal{Y}_q$ be a set of $r$ elements sampled at random from $\mathcal{X}_q$ with replacement.
    Query $\mathcal{O}_{\mathsf{fuzzy}}(i,t_2)$ to obtain $\mathsf{U}_{it_2}$ for all $i \in \mathcal{Y}_q$
end for
Let $\mathcal{Q}$ be a set of $r$ elements sampled at random from $\mathcal{P}_1$ with replacement.
Query $\mathcal{O}_{\mathsf{fuzzy}}(i,t_2)$ to obtain $\mathsf{U}_{it_2}$ for all $i \in \mathcal{Q}$.
Compute $\widehat{\boldsymbol{\mu}}_{t_2}$ in (111).
for $j = 1, 2$ do
    Run Alg MEMBERSHIP($\mathcal{X}, \widehat{\boldsymbol{\mu}}_j, \alpha, \eta, j$) to update $\widehat{\mathsf{U}}_{ij}$ for all $i \in [n]$.
end for
Return $\widehat{\boldsymbol{\mu}}_1, \widehat{\boldsymbol{\mu}}_2$ and $\widehat{\mathsf{U}}_{ij}$ $\forall$ $i \in [n], j \in \{1,2\}$.
```

We now describe Algorithm 7 used to estimate the membership weights of cluster t_2. We start by sorting $\mathcal{X}$ in ascending order according to their distance from $\widehat{\boldsymbol{\mu}}_{t_1}$, and initialize $\eta_1 = \mathsf{U}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(n)t_2}$ and $p_{\eta_1} \triangleq n$. To wit, η_1 is the membership of the element farthest from $\widehat{\boldsymbol{\mu}}_{t_1}$ to the smaller cluster t_2. For each $q \in= \{2, 3 \ldots, 3\log n\}$, we set $\eta_q = \eta_{q-1}/2$, and then perform binary search to find $p'_{\eta_q} = \mathsf{argmin}_{j\in[n]} \mathbb{1}[\mathsf{U}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(j)t_2} \geq \eta_q]$. Here, we should think of p'_{η_q} to be the index for which $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p'_{\eta_q})}$ is the element closest from $\widehat{\boldsymbol{\mu}}_{t_1}$, such that its membership to the smaller cluster t_2 is at least η_q. Now, if $|p'_{\eta_q} - p_{\eta_{q-1}}| \geq \log n$, we keep η_q unchanged, set $p_{\eta_q} = p'_{\eta_q}$, and put all elements which are closer to $\widehat{\boldsymbol{\mu}}_{t_1}$ than $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_{q-1}})}$, but farther than $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_q})}$ into a single bin. Notice that each such bin contains at least $\log n$ elements by definition. However, if p'_{η_q} is the same as $p_{\eta_{q-1}}$, then the membership queries in the latter binary search to compute p'_{η_q} is wasteful. In order to fix this issue, we check if there are a sufficient number (at least $\log n$) of elements between p'_{η_q} and $p_{\eta_{q-1}}$. If there are not, then we set p_{η_q} such that the above condition is satisfied. More formally, if $|p'_{\eta_q} - p_{\eta_{q-1}}| \leq \log n$ we update $p_{\eta_q} = \min(0, p_{\eta_{q-1}} - 1 - \log n)$ and $\eta_q = \mathsf{U}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_q})t_2}$. In other words, we set p_{η_q} such that there at least $\log n$ elements between p_{η_q} and $p_{\eta_{q-1}}$, and accordingly, we set η_q to be the membership of $\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_q})$ to the cluster t_2. Subsequently, we query U_{it_2} for every element that is closer to $\widehat{\boldsymbol{\mu}}_{t_1}$ than $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_{q-1}})}$ but farther than $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_q})}$. In this case, we call these elements *special elements*, since we query every one of them. We will assume that all these special elements between $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_{q-1}})}$ and $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_q})}$ are assigned a single bin.

To resolve the edge case, we put all those elements which are closer to $\widehat{\boldsymbol{\mu}}_{t_1}$ than $\boldsymbol{x}_{\pi_{\widehat{\boldsymbol{\mu}}_{t_1}}(p_{\eta_{3\log n}})}$ into a separate bin. Let g be the total number of bins formed by this algorithm, and say $\mathcal{X}_j$ is the j^{th} bin formed by this method that does not contain any special elements. Let $\mathcal{P}$ be the set of all special elements. Notice that the number of bins g is at most $1 + 3\log n$, and the number of special elements is at most $3\log^2 n$ (since each bin can contain at most $\log n$ special elements). Moreover, for any bin $\mathcal{X}_j$ not containing any special elements, we can show that $\max_{i\in\mathcal{X}_j} \mathsf{U}_{it_2} \leq 2\min_{i\in\mathcal{X}_j} \mathsf{U}_{it_2}$, which should be contrasted with the guarantee we have for the sequential algorithm, i.e., for a particular bin $\mathcal{X}_j$, $|\max_{i\in\mathcal{X}_j} \mathsf{U}_{it_2} - \min_{i\in\mathcal{X}_j} \mathsf{U}_{it_2}| \leq \eta_1$. Finally, we again sub-sample r elements from each such bin $\mathcal{X}_j$ with replacement, and denote the resulting subset by $\mathcal{Y}_j$. We approximate the center of the second cluster by

$$\widehat{\boldsymbol{\mu}}_{t_2} = \frac{\sum_j \frac{|\mathcal{X}_j|}{r} \sum_{i\in\mathcal{Y}_j} \mathsf{U}_{it_2}^{\alpha}\boldsymbol{x}_i + \sum_{i\in\mathcal{P}} \mathsf{U}_{it_2}^{\alpha}\boldsymbol{x}_i}{\sum_j \frac{|\mathcal{X}_j|}{r} \sum_{i\in\mathcal{Y}_j} \mathsf{U}_{it_2}^{\alpha} + \sum_{i\in\mathcal{P}} \mathsf{U}_{it_2}^{\alpha}}, \tag{10}$$

and show that the required r is independent of the cluster size.

Acknowledgements: This work is supported in part by NSF awards 2133484, 2127929, and 1934846.

References

[1] M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. Farag, and T. Moriarty. A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data. *IEEE transactions on medical imaging*, 21(3):193–199, 2002.

[2] N. Ailon, A. Bhattacharya, R. Jaiswal, and A. Kumar. Approximate clustering with same-cluster queries. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[3] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In N. Bansal, K. Pruhs, and C. Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035. SIAM, 2007.

[4] H. Ashtiani, S. Kushagra, and S. Ben-David. Clustering with same-cluster queries. In *Advances in Neural Information Processing Systems 29*, pages 3216–3224. 2016.

[5] M.-F. Balcan and A. Blum. Clustering with interactive feedback. In *International Conference on Algorithmic Learning Theory*, pages 316–328. Springer, 2008.

[6] A. Ben-Dor, R. Shamir, and Z. Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.

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

[8] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[9] J. Blömer, S. Brauer, and K. Bujna. A theoretical analysis of the fuzzy k-means problem. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 805–810. IEEE, 2016.

[10] J. Blömer, S. Brauer, and K. Bujna. Coresets for Fuzzy K-Means with Applications. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:12, 2018.

[11] A. Blum, J. Hopcroft, and R. Kannan. *Foundations of data science*. Cambridge University Press, 2020.

[12] M. Bressan, N. Cesa-Bianchi, S. Lattanzi, and A. Paudice. Exact recovery of mangled clusters with same-cluster queries. *arXiv preprint arXiv:2006.04675*, 2020.

[13] M. Bressan, N. Cesa-Bianchi, S. Lattanzi, and A. Paudice. Exact recovery of clusters in finite metric spaces using oracle queries. *arXiv preprint arXiv:2102.00504*, 2021.

[14] M. Bressan, N. Cesa-Bianchi, S. Lattanzi, and A. Paudice. On margin-based cluster recovery with oracle queries. *arXiv preprint arXiv:2106.04913*, 2021.

[15] M. Bressan, N. Cesa-Bianchi, A. Paudice, and F. Vitale. Correlation clustering with adaptive similarity queries. In *Advances in Neural Information Processing Systems*, pages 12510–12519, 2019.

[16] K. Bujna. *Soft Clustering Algorithms Theoretical and Practical Improvements*. PhD thesis, Paderborn University, 2017.

[17] I. Chien, C.-Y. Lin, and I.-H. Wang. Community detection in hypergraphs: Optimal statistical limit and efficient algorithms. In *International Conference on Artificial Intelligence and Statistics*, pages 871–879, 2018.

[18] I. Chien, C. Pan, and O. Milenkovic. Query k-means clustering and the double dixie cup problem. In *Advances in Neural Information Processing Systems*, pages 6649–6658, 2018.

[19] K.-S. Chuang, H.-L. Tzeng, S. Chen, J. Wu, and T.-J. Chen. Fuzzy c-means clustering with spatial information for image segmentation. *Computerized Medical Imaging and Graphics*, 30(1):9–15, 2006.

[20] W. F. De La Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 50–58, 2003.

[21] M. L. D. Dias. fuzzy-c-means: An implementation of fuzzy c-means clustering algorithm., May 2019.

[22] C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 606–610. SIAM, 2005.

[23] D. Firmani, S. Galhotra, B. Saha, and D. Srivastava. Robust entity resolution using a crowdoracle. *IEEE Data Eng. Bull.*, 41(2):91–103, 2018.

[24] Y. Fukuyama. A new method of choosing the number of clusters for the fuzzy c-mean method. 1989.

[25] I. Gath and A. B. Gev. Unsupervised optimal fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):773–780, July 1989.

[26] K. Huang, N. D. Sidiropoulos, and A. Swami. Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition. *IEEE Transactions on Signal Processing*, 62(1):211–224, 2013.

[27] M. Huang, Z. Xia, H. Wang, Q. Zeng, and Q. Wang. The range of the value for the fuzzifier of the fuzzy c-means algorithm. *Pattern Recognition Letters*, 33(16):2280 – 2284, 2012.

[28] W. Huleihel, A. Mazumdar, M. Médard, and S. Pal. Same-cluster querying for overlapping clusters. In *Advances in Neural Information Processing Systems*, pages 10485–10495, 2019.

[29] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[30] X. Lin and X.-w. Chen. Mr. knn: soft relevance for multi-label classification. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 349–358, 2010.

[31] Q. Liu, J. Liu, M. Li, and Y. Zhou. A novel initialization algorithm for fuzzy c-means problem. In J. Chen, Q. Feng, and J. Xu, editors, *Theory and Applications of Models of Computation*, pages 215–225. Springer International Publishing, 2020.

[32] S. P. Lloyd. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982.

[33] M. Mahajan, P. Nimbhorkar, and R. Kasturi. The planar k-means problem is np-hard. volume 442, pages 274–285, 02 2009.

[34] X. Mao, P. Sarkar, and D. Chakrabarti. On mixed memberships and symmetric nonnegative matrix factorizations. In *International Conference on Machine Learning*, pages 2324–2333. PMLR, 2017.

[35] A. Mazumdar and S. Pal. Semisupervised clustering, and-queries and locally encodable source coding. In *Advances in Neural Information Processing Systems*, pages 6489–6499, 2017.

[36] A. Mazumdar and B. Saha. Clustering with noisy queries. In *Advances in Neural Information Processing Systems*, pages 5788–5799, 2017.

[37] A. Mazumdar and B. Saha. Query complexity of clustering with side information. In *Advances in Neural Information Processing Systems*, pages 4682–4693, 2017.

[38] A. Mazumdar and B. Saha. A theoretical analysis of first heuristics of crowdsourced entity resolution. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[39] A. Moitra. Algorithmic aspects of machine learning. *Lecture notes*, 2014.

[40] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[42] W. Pedrycz and J. Waletzky. Fuzzy clustering with partial supervision. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(5):787–795, 1997.

[43] B. Saha and S. Subramanian. Correlation clustering with same-cluster queries bounded by optimal cost. In *27th Annual European Symposium on Algorithms (ESA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[44] S. Sarmasarkar, K. S. Reddy, and N. Karamchandani. Query complexity of heavy hitter estimation. *arXiv preprint arXiv:2005.14425*, 2020.

[45] N. D. Sidiropoulos and R. Bro. On the uniqueness of multilinear decomposition of n-way arrays. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 14(3):229–239, 2000.

[46] C. E. Tsourakakis, M. Mitzenmacher, K. G. Larsen, J. Błasiok, B. Lawson, P. Nakkiran, and V. Nakos. Predicting positive and negative links with noisy queries: Theory & practice. *arXiv preprint arXiv:1709.07308*, 2017.

[47] F. VALAFAR. Pattern recognition techniques in microarray data analysis: A survey. *Annals of the New York Academy of Sciences*, 980:41–64, 2002.

[48] A. Vattani. The hardness of k-means clustering in the plane.

[49] N. Vesdapunt, K. Bellare, and N. Dalvi. Crowdsourcing algorithms for entity resolution. *Proceedings of the VLDB Endowment*, 7(12):1071–1082, 2014.

[50] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11), 2012.

[51] X. L. Xie and G. Beni. A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(8):841–847, Aug. 1991.

[52] S. Zhang, R.-S. Wang, and X.-S. Zhang. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications*, 374(1):483–490, 2007.