Fuzzy Clustering with Similarity Queries

Wasim Huleihel ∗ Arya Mazumdar † Soumyabrata Pal ‡

November 5, 2021

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 $O(\text{poly}(k) \log n)$ 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 algorithm) 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

---

∗W. Huleihel is with the Department of Electrical Engineering-Systems at Tel Aviv university, Tel Aviv 6997801, Israel (e-mail: wasimh@tauex.tau.ac.il).
†A.Mazumdar is with the Halıcıoğlu Data Science Institute at University of California, San Diego, USA (email: arya@ucsd.edu)
‡S. Pal is with the Computer Science Department at the University of Massachusetts Amherst, Amherst, MA 01003, USA (email: spal@cs.umass.edu).
achievable with membership queries, can be also achieved by same cluster queries by asking at most \( k \) times as many queries [4].

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].

1.1 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.

### 1.2 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 \( \beta \) 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 \( \beta \) of the query complexity can be further improved. Despite the strong dependency of the query complexity on \( \beta \), 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 \( \beta \), 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 trade off and effect of different values of \( \beta \) and \( k \) on the estimation error and query complexity.
1.3 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) \([22]\).

1.4 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, followed by several experimental results over both synthetic and real-world datasets. Finally, our conclusion an outlook appear in Section 6. Some technical proofs and discussions are relegated to the appendices.

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 \( \{x_i\}_{i=1}^n \), and assume without loss of generality that \( x_i \in \mathcal{B}(0,R) \), for any \( i \in [n] \), where \( \mathcal{B}(a,R) \) is the \( L_2 \)-Euclidean ball of radius \( R \in \mathbb{R}_+ \) centered around \( a \in \mathbb{R}^d \). Let \( \mathcal{C}_k = \{C_1, C_2, \ldots, C_k\} \) be a clustering (or, partition) of \( \mathcal{X} \), where \( k \in \mathbb{N} \) is the number of clusters. We say that a clustering \( \mathcal{C}_k \) is center-based if there exists a set of centers \( \mu = \{\mu_1, \mu_2, \ldots, \mu_k\} \subset \mathbb{R}^d \) such that the clustering is induced by the Voronoi diagram over those center points. The objective function of hard \( k \)-means is

\[
J_{km}(\mathcal{X}, \mu, U) = \sum_{i=1}^n \sum_{j=1}^k U_{ij} \|x_i - \mu_j\|_2^2,
\]

where \( U \in \{0,1\}^{n \times k} \) is the partition matrix, i.e., the membership weight \( U_{ij} = 1 \) if \( x_i \in C_j \), 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. \((\mu, U)\).

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 $\mu = \{\mu_k\}_{k \in [k]} \subset \mathbb{R}^d$ and a membership matrix $U \in [0,1]^{n \times k}$ minimizing

$$J_{fm}(X, \mu, U) \triangleq \sum_{i=1}^{n} \sum_{j=1}^{k} U_{ij} \|x_i - \mu_j\|_2^2,$$

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

The parameter $\alpha$ is called the fuzzifier, and is not subject to optimization. It can be shown that when $\alpha \rightarrow 1$, 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 $\mu$, it can be shown that the optimal membership weights minimizing the cost in (2) are (see, e.g., [8, Theorem 11.1])

$$U_{ij} = \left[ \frac{1}{\sum_{\ell=1}^{k} \left( \frac{\|x_i - \mu_j\|}{\|x_i - \mu_\ell\|} \right)^{2/(\alpha-1)} } \right]^{-1},$$

for all $i \in [n], j \in [k]$. On the other hand, given a set of membership weights $U$, the optimal centers are given by

$$\mu_j = \frac{\sum_{i=1}^{n} U_{ij}^\alpha x_i}{\sum_{i=1}^{n} U_{ij}^\alpha},$$

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 minima) 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:

$$XB(X, \mu, U) \triangleq \frac{J_{fm}(X, \mu, U)}{nk \cdot \min_{i \neq j} \|\mu_i - \mu_j\|_2^2}.$$  

For the rest of this paper, we denote a clustering by $\mathcal{P} \triangleq (\mu, U)$, and we replace $XB(X, \mu, U)$ by $XB(X, \mathcal{P})$. Accordingly, the optimal solution to the fuzzy $k$-means problem (Definition 1) is denoted by $\mathcal{P}^*$.

Finally, we say that a clustering $\mathcal{P}$ is consistent center-based if:

- Given membership weights $U$, the centers $\mu$ are given by (4).
- The membership weights $U$ are monotone, i.e., if $\|x_i - \mu_j\|_2 \leq \|x_\ell - \mu_j\|_2$ then $U_{ij} \geq U_{\ell j}$, for any $i, \ell \in [n]$ and $j \in [k]$.

\[\text{In (5) we divide by } nk \text{ and not just by } n, \text{ as in [51], where } k \text{ was consider a fixed parameter. If } k \text{ is allowed to grow (e.g., with } n), \text{ then our definition makes more sense, as in general the numerator in (5) can grow with } k.\]
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 12 in the appendix, both conditions are satisfied by the optimal solution $P^*$, 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 5, 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 $P$, a $P$-oracle is a function $O_P$ that answers queries consistent with $P$. 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 $P$ is a consistent center-based clustering, it need not be the optimal solution $P^*$ minimizing (2). The algorithm then tries to recover (or, approximate) $P$ 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 $X$, the clustering $P$ 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 $x_i$ to a cluster $j$, i.e., $O_{fuzzy}(i,j) = U_{ij}$.

Basically we ask the oracle for the membership weight of $x_i$ to cluster $C_j$. Using a certain amount of such queries and the data set $X$ 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 $x_p$ and $x_q$ i.e., $O_{sim}(p,q) = \langle U_p, U_q \rangle$. A fuzzy triplet similarity query asks the similarity of three distinct instances $x_p, x_q, x_r$, i.e., $O_{triplet}(p,q,r) = \sum_{t \in [k]} U_{pt} U_{qt} U_{rt}$.

We show in Appendix D, 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 $A_n$ 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 $T_n + O(mk + k^3)$. 


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 \( O_{\text{fuzzy}}(i, j) = U_{ij} + \zeta_{ij} \), where \( \zeta_{ij} \) 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 C 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 \((X, P)\) be a clustering instance. An algorithm \( A \) is called a \((\epsilon_1, \epsilon_2, Q)\)-solver if it can approximate \( P \) by having access to \( X \) and making at most \( Q \) membership queries to a \( P \)-oracle. Specifically, for any \( \epsilon_1, \epsilon_2 > 0 \), the algorithm outputs \( \hat{P} \) such that \( \max_{j \in [k]} \| \mu_j - \hat{\mu}_j \|_2 \leq \epsilon_1 \) and \( \max_{i \in [n], j \in [k]} | U_{ij} - \hat{U}_{ij} | \leq \epsilon_2 \). Such an algorithm is a polynomial \((\epsilon_1, \epsilon_2, Q)\)-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 that \( Q = O(nk) \) membership queries suffice to recover the \( P \)-oracle clustering. Also, with unbounded computational resources one can find the optimal clustering \( P^* \) 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 \((X, P)\) be a clustering instance, and \( A \) be a \((\epsilon_1, \epsilon_2, Q)\)-solver. Then,

\[
\left| XB(X, P) - XB(X, \hat{P}) \right| \leq \frac{XB(X, P) \cdot O(\epsilon_1) + O(\epsilon_2)}{\min_{i \neq j} \| \mu_i - \mu_j \|_2^2} + o \left( \frac{\epsilon_1^2 + \epsilon_2^2}{\min_{i \neq j} \| \mu_i - \mu_j \|_2^2} \right). \tag{6}
\]

The lemma above shows that if \( \epsilon_1 \) and \( \epsilon_2 \) are small enough, then the \( XB \) measure associated with an \((\epsilon_1, \epsilon_2, Q)\)-solver is "close" to the \( XB \) measure associated with \( P \). Note, however, that the r.h.s. of (6) depends inversely on \( \min_{i \neq j} \| \mu_i - \mu_j \|_2^2 \). Accordingly, if for some clustering instance \((X, P)\) this quantity is too small compared to the estimation error, then the difference \( | XB(X, P) - XB(X, \hat{P}) | \) 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 \( X \) and \( \upsilon \in \mathbb{R}^d \), let \( \pi_\upsilon : [n] \rightarrow [n] \) be the permutation defined on \([n]\) such that

\[
\| \upsilon - x_{\pi_\upsilon(i)} \|_2 \leq \| \upsilon - x_{\pi_\upsilon(j)} \|_2, \tag{7}
\]
for all \( i < j \in [n] \). To wit, \( \pi_\nu \) is the ranking of the elements in \( \mathcal{X} \) when they are sorted in ascending order w.r.t. to their distance from \( \nu \). Then, we let \( \gamma \in \mathbb{R}_+ \) be the maximum positive number such that for all \( j \in [k] \),

\[
\pi_\mu_j = \pi_\mu, \quad \forall \mu \in \mathcal{B}(\mu_j, \gamma).
\]

(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 \( \sum_{i \in [n]} U_{ij} \), for all \( j \in [k] \), and a parameter \( \beta \in \mathbb{R}_+ \) by

\[
\beta \triangleq \min_{j \in [k]} \frac{k}{n} \sum_{i \in [n]} U_{ij}.
\]

(9)

Accordingly, in terms of \( \beta \), the minimum cluster membership size is at least \( \beta n/k \). Notice that \( \beta \) must be less than unity since \( \sum_{i,j} U_{ij} = n \), and there must exist at least one cluster whose membership size is at most \( n/k \). We mention that \( \beta \) was introduced in [18] as well.

## 3 Algorithms and Theoretical Guarantees

In this section we provide our algorithms along with their theoretical guarantees, for the setting described in the previous section. Specifically, we assume that the oracle has a consistent center-based clustering in mind. For this model, we provide two different algorithms; the first is a two-phase algorithm, and the second is a sequential one. Finally, for the special case of \( k = 2 \), we show an algorithm with significantly improved 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:** \( \mathcal{X} \), \( O_{\text{lazy}} \), \( \alpha \), \( m \), and \( \eta \).

**Output:** An estimate \( \hat{\mathcal{P}} \).

1: Let \( \mathcal{S} \) be a set of \( m \) elements sampled at random from \( [n] \) with replacement.
2: Query \( O_{\text{lazy}}(i,j) \), \( \forall i \in \mathcal{S}, j \in [k] \).
3: Compute \( \hat{\mu}_j = \frac{\sum_{i \in \mathcal{S}} U_{ij} \pi_i}{\sum_{i \in \mathcal{S}} U_{ij}}, \forall j \in [k] \).
4: for \( j = 1, 2, \ldots, k \) do
5: \( \{ \hat{U}_{ij} \}^n_{i=1} \leftarrow \text{MEMBERSHIP}(\mathcal{X}, \hat{\mu}_j, \alpha, \eta) \).
6: \( \hat{U}_{ij} \leftarrow \hat{U}_{ij} + \frac{1 - \sum_{k \neq j} \hat{U}_{ik}}{k}, \forall i \in [n] \).
7: end for

**Algorithm 2** MEMBERSHIP(\( \mathcal{X}, \hat{\mu}_j, \alpha, \eta \))

**Input:** \( \mathcal{X} \), and \( O_{\text{lazy}} \).

**Output:** An estimate \( \{ \hat{U}_{ij} \}^n_{i=1} \).

1: \( \hat{\pi}_\mu \) w.r.t. \( \mathcal{X} \).
2: for \( s = 0, 1, \ldots, 1/\eta \) do
3: \( \ell_s = \arg\max_{i \in [n]} U_{\hat{\pi}_\mu(i)j} \geq s\eta \) using BinarySearch(\( \mathcal{X}, \hat{\pi}_\mu, s\eta \)).
4: end for
5: for \( s = 0, 1, 2, \ldots, 1/\eta \) do
6: \( \ell_s = \ell_{s-1} - 1, \ldots, \ell_{s+1} + 1 \) do
7: \( \text{Set } \hat{U}_{\hat{\pi}_\mu(i)j} = s\eta \).
8: end for
9: end for
Next, using the approximated centers, we approximate the membership weights of every element up to a precision of \( \eta \) 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 \( \epsilon \) 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 \( U_{\pi\hat{v}_j(j)} \) are non-increasing in \( i \). Therefore, for any \( j \in [k] \) using the binary search in Algorithm 3, we find the maximum index \( \ell_s \) such that \( U_{\pi\hat{v}_j(t)} \) is larger than \( s\eta \), for \( s \in \{0, 1, \ldots, 1/\eta \} \). This will create an \( \eta \)-spaced grid, and we approximate each membership weight in accordance to the cell it falls into. Specifically, we assign \( \hat{U}_{\pi\hat{v}_j(j)} = s\eta \), for all \( t \in \{\ell_s, \ell_s - 1, \ldots, \ell_{s+1} + 1 \} \). Clearly, the estimation error is at most \( \eta \). 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 \( \hat{U}_{ij} \) to ensure that \( \sum_{j=1}^{n}\hat{U}_{ij} = 1 \), for \( j \in [k] \). We have the following result, the detailed proof of which is delegated to Appendix 4.1.

**Theorem 1.** Let \( (\mathcal{X}, \mathcal{P}) \) be a consistent center-based clustering instance, and recall the definitions of \( \gamma \in \mathbb{R}_+ \) and \( \beta \in (0, 1) \) in (8)-(9). Pick \( \epsilon \leq \gamma \), \( \eta \in (0, 1) \), and \( \delta \in (0, 1) \). Take as input to Algorithm 1 \( m \geq \left(\frac{R\delta}{\epsilon^{3/2}}\right)^4 \log \frac{k}{\delta} \), for some \( c > 0 \). Then, with probability at least \( 1 - \delta \), Algorithm 1 is \((\epsilon, \eta, \mathcal{Q})\)-solver using \( Q = O\left(\frac{Rk^{4\alpha+1}}{(\epsilon^{3/2})^4} \log \frac{k}{\delta} + \frac{k\log n}{\eta}\right) \) membership queries. Furthermore, Algorithm 1 time-complexity is of order \( O\left(\frac{k}{\delta} \log n + \frac{Rk^{4\alpha+1}}{(\epsilon^{3/2})^4} \log \frac{k}{\delta} + k\log n/\eta\right) \).

**Proof Sketch of Theorem 1.** To prove Theorem 1 we first show that with high probability, Steps 1–4 in Algorithm 1 output an \( \epsilon \)-approximation for all centers. To that end, we first show in Lemma 3 that for a given cluster \( j \in [k] \), with probability \( 1 - \delta \), \( \|\mu_j - \hat{\mu}_j\|_2 \leq \frac{4R^2}{\epsilon^2}\sqrt{\frac{\epsilon}{m}} \log \frac{1}{\delta} \), where \( Y \triangleq \min_{j \in [k]} \frac{1}{n} \sum_{i \in [n]} U_{ij}^{\alpha} \). This result follows by noticing that both the numerator and denominator of \( \hat{\mu}_j \) are unbiased estimates of the corresponding numerator and denominator of \( \mu_j \) in (4). Representing \( \hat{\mu}_j \) as a sum of the true underlying center \( \mu_j \) and a certain error term, the generalized Hoeffding’s inequality in Lemma 11 implies the above estimation error guarantee. In light of this result, using Hölder’s inequality and the union bound we show in Lemma 5 that \( \|\hat{\mu}_j - \mu_j\|_2 \leq \epsilon \), 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 4 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 \( \hat{U}_{ij} \), for \( i \in [n] \), such that \( 0 \leq U_{ij} - \hat{U}_{ij} \leq \eta \) and \( \hat{U}_{ij} \in \{0, \eta, 2\eta, \ldots, 1\} \), \( \forall i \in [n] \), using \( O(\log n/\eta) \) 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

---

**Algorithm 3 BinarySearch**

**Input:** \( \mathcal{O}_{\text{lazy}} \).

1. Set low = 1 and high = \( n \).
2. while low \( \neq \) high do
   3. Set mid = \( [(\text{low} + \text{high})/2] \).
   4. Query \( \mathcal{O}_{\text{lazy}}(\pi(mid), j) \).
   5. if \( U_{\pi(mid),j} \geq x \) then
   6. Set low = mid.
   7. else
   8. Set high = mid - 1.
9. end if
10. end while
exponential dependency on the "fuzzifier" $\alpha$. 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 $\alpha$ 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 $\alpha$ is unknown. Indeed, since there is no baseline, one cannot decide which value of $\alpha$ is preferable. Note also that despite the fact that our query complexity naturally depend on $\alpha$, it cannot be used to determine the "best" value of $\alpha$. Assuming the existence of a ground truth, in Appendix 5 we compare the clustering accuracy for different values of $\alpha$ 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 $\beta$. 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 $\sum_{i \in S} U^a_{ij}$, among the $k$ clusters. Corollary 4 in Section 4.2 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 $\eta_1$ and further, the estimate has an additive error of at most $\eta_1$, 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:

\textbf{Theorem 2.} Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, and recall the definitions of $\gamma \in \mathbb{R}_+$ and $\beta \in (0, 1)$ in (8)-(9). Take $m \geq \left( \frac{R^4}{\epsilon} \right)^4 c \log \frac{2k}{\beta} $, $r \geq \frac{R^4 k^{40}}{\epsilon^4 \beta^4} \log \frac{4k}{\eta_2 \beta}$, for some $c > 0$, and any $\delta \in (0, 1)$. Then, with probability at least $1 - \delta$, Algorithm 4 is $(\epsilon, \eta_2, Q)$-solver using $Q = O \left( \frac{R^4 k^{40} + 1}{\eta_1 \beta} \log \frac{4k}{\eta_1 \beta} + \frac{k \log n}{\eta_1} + \frac{k \log n}{\eta_2} \right)$ membership queries, for any chosen $\epsilon \leq \gamma$ and $\eta_2 \leq \eta_1 \leq \frac{1}{\epsilon} \left( 1 - \frac{\beta}{\epsilon} \right)$. Furthermore, Algorithm 4 time-complexity is of order $O \left( \frac{R^4 k^{40} + 1}{\eta_1 \beta} \log \frac{4k}{\eta_1 \beta} + \frac{k \log n}{\eta_1} + \frac{k \log n}{\eta_2} + kn \log n \right)$.

The proof of Theorem 2 is given in Subsection 4.2, and is based on a technically complicated induction arguments. Below, we provide a short sketch.

\textbf{Proof Sketch of Theorem 2.} Suppose that we have approximated the means of $\ell \leq k$ clusters, indexed by $T_{\ell} = \{t_1, t_2, \ldots, t_{\ell}\}$, 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 $T_{\ell}$, up to an additive error $\eta_1$. The leftover clusters, indexed by $[k] \setminus T_{\ell}$, are the unprocessed clusters. The above implies that we can also approximate the sum $\sum_{j \in [\ell]} U_{ij}$, for each $i \in \mathcal{X}$, up to an additive error $\ell \eta_1$. Note that, since $\tilde{U}_{ij}$ is a multiple of $\eta_1$ for all $j \in [\ell]$, $\sum_{j \in [\ell]} \tilde{U}_{ij}$ must also be a multiple of $\eta_1$. Subsequently, we partition $\mathcal{X}$ into bins, where the elements in a particular bin have the same approximated sum of membership weights. Formally, let $\Sigma = \{0, 1, \ldots, 1/\eta_1\}$. For each $s \in \Sigma$, we define $\mathcal{X}_s = \{i \in [n] : \sum_{j \in [\ell]} \tilde{U}_{ij} = sm_1\}$ 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 $r$
While the sequential algorithm proposed in the previous subsection exhibits graceful dependency of the query complexity on $\beta$, it is still unclear to what extent this dependency can be reduced. Noting that the sequential algorithm might be too large if there exists a cluster, this stems from the fact that we estimate the membership weights up to a precision of $\eta_1$, which might be too large if there exists a cluster $j$ such that $\max_{i \in [n]} U_{ij} \leq \eta_1$. 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.

### 3.3 The Special Case of Two Clusters

While the sequential algorithm proposed in the previous subsection exhibits graceful dependency of the query complexity on $\beta$, it is still unclear to what extent this dependency can be reduced. Interestingly, it turns out that for the special case of two clusters $k=2$, we can devise an algorithm whose query complexity is completely independent of $\beta$. The proof of the following result is given in Subsection 4.3.
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 $m = O\left(\frac{\mathbb{E}}{\epsilon}\log \frac{1}{\delta}\right)$ and $r = O\left(\frac{R^4}{\epsilon^4} \log \frac{2}{\eta \delta}\right)$ is a $(\epsilon, \eta, Q)$-solver using membership queries, for any chosen $\epsilon \leq \gamma$, $\delta \in (0, 1)$ and $\eta > 0$. Furthermore, Algorithm 5 time-complexity is of order $O\left(n \log n + \frac{R^4 \log n \log(1/\eta \delta)}{\epsilon^4} + \log^2 n + \frac{\log n}{\eta}\right)$.

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

**Input**: $\mathcal{X}$, $\mathcal{O}_{\text{fuzzy}}$, $\alpha$, $m$, $r$, and $\eta$.

1. Let $\mathcal{S}$ be a set of $m$ elements sampled at random from $[n]$ with replacement.
2. Query $\mathcal{O}_{\text{fuzzy}}(i, j)$ and store $U_{ij}$ for all $j \in \{1, 2\}$.
3. Choose $t_1 = \text{argmax}_{j \in \{1, 2\}} \sum_{i \in \mathcal{S}} U_{ij}^0$.
4. Compute estimator $\hat{\mu}_{t_1} = \frac{\sum_{i \in \mathcal{S}} U_{ij}^n x_i}{\sum_{i \in \mathcal{S}} U_{ij}^n}$.
5. Obtain $P_1, P_2, \mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_q$ and $\hat{U}_{it_2}$ for all $i \in [n]$ using $\text{Membership2}(\mathcal{X}, \hat{\mu}_{t_1}, \alpha)$.
6. for $q = 1, 2, 3, \ldots, 3 \log n$ do
   7. Let $\mathcal{Y}_q$ be a set of $r$ elements sampled at random from $\mathcal{X}_q$ with replacement.
   8. Query $\mathcal{O}_{\text{fuzzy}}(i, t_2)$ to obtain $U_{it_2}$ for all $i \in \mathcal{Y}_q$.
7. end for
9. Let $\mathcal{Q}$ be a set of $r$ elements sampled at random from $P_1$ with replacement.
10. Query $\mathcal{O}_{\text{fuzzy}}(i, t_2)$ to obtain $U_{it_2}$ for all $i \in \mathcal{Q}$.
11. Compute $\hat{\mu}_{t_2}$ in (88).
12. for $j = 1, 2$ do
   13. Run Algorithm $\text{Membership}(\mathcal{X}, \hat{\mu}_j, \alpha, \eta, j)$ to update $\hat{U}_{ij}$ for all $i \in [n]$.
14. end for
15. Return $\hat{\mu}_j$ for all $j \in \{1, 2\}$ and $\hat{U}_{ij}$ for all $i \in [n], j \in \{1, 2\}$.

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., $\sum_{i \in \mathcal{S}} U_{it_1}^n > \sum_{i \in \mathcal{S}} U_{it_2}^n$. The algorithm works as follows. We start by computing an estimate $\hat{\mu}_{t_1}$ for the center of the largest cluster $t_1$. Corollary 4 shows that $O\left((R/\epsilon)^4 \log(4/\delta)\right)$ membership queries are needed to guarantee that $||\hat{\mu}_{t_1} - \mu_{t_1}||_2 \leq \epsilon$. 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 $U_{it_1} + U_{it_2} = 1$. While, in the sequential algorithm, we always approximated the membership weights using bins of fixed size $\eta_1$, 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 $\beta$.

We now describe Algorithm 6 used to estimate the membership weights of cluster $t_2$. We start by sorting $\mathcal{X}$ in ascending order according to their distance from $\hat{\mu}_{t_1}$, and initialize
Algorithm 6 Membership2(\(X, \hat{\mu}_t, \alpha\)) Estimate the memberships \(U_{it_2}\) for all \(i \in [n]\) given an estimated mean \(\hat{\mu}_t\).

**Input:** \(\mathcal{O}_{\text{fuzzy}}\).

1. Sort the elements \(x_1, x_2, \ldots, x_n\) in ascending order according to \(\|x_i - \hat{\mu}_t\|_2\). Denote the resultant permutation of \([n]\) corresponding to the sorted elements by \(\pi \hat{\mu}_t\).

2. Set \(\eta_1 \triangleq 1 - U_{\pi \hat{\mu}_t}(n)t_2\) and \(p_{\eta_1} \triangleq n\).

3. Query \(\mathcal{O}_{\text{fuzzy}}(\pi \hat{\mu}_t(n), t_1)\) to obtain \(U_{\pi \hat{\mu}_t}(n)t_2 = 1 - U_{\pi \hat{\mu}_t}(n)t_1\).

4. Initialize \(P_1, P_2 = \phi\).

5. for \(q = 2, 3, \ldots, 3 \log n\) do

6. Initialize \(X_q = \phi\) and set \(\eta_q = \frac{\eta_{q-1}}{2}\) using BinarySearch2(\(X, \pi \hat{\mu}_t, \eta_q\)).

7. Find \(p_{\eta_q}' = \arg\min_{i \in [n]} U_{\pi \hat{\mu}_t}(i)t_2 \geq \eta_q\).

8. if \(|p_{\eta_q}' - p_{\eta_{q-1}}| \geq \log n\) then

9. Set \(p_{\eta_q} = p_{\eta_q}'\).

10. for \(i = p_{\eta_q}, p_{\eta_q} + 1, \ldots, p_{\eta_{q-1}} - 1\) do

11. Set \(X_q = X_q \cup \pi \hat{\mu}_t(i)\) and set

12. \(U_{\pi \hat{\mu}_t}(i)t_2 = 1 - U_{\pi \hat{\mu}_t}(p_{\eta_q})t_1\)

13. else

14. Query \(\mathcal{O}_{\text{fuzzy}}(\pi \hat{\mu}_t(\min(0, p_{\eta_q} - 1 - \log n - 1)), t_1)\) and obtain the membership \(U_{\pi \hat{\mu}_t}(\min(0, p_{\eta_q} - 1 - \log n - 1)), t_1\).

15. Set \(\eta_q = U_{\pi \hat{\mu}_t}(\min(0, p_{\eta_q} - 1 - \log n - 1)), t_2\) and \(p_{\eta_q} = \min(0, p_{\eta_q} - 1 - \log n)\).

16. for \(i = p_{\eta_q}, p_{\eta_q} + 1, \ldots, p_{\eta_q} - 1 - 1\) do

17. Query \(\mathcal{O}_{\text{fuzzy}}(\pi \hat{\mu}_t(i), t_1)\) to obtain \(U_{\pi \hat{\mu}_t}(i)t_1\).

18. Set \(U_{\pi \hat{\mu}_t}(i)t_2 = 1 - U_{\pi \hat{\mu}_t}(i)t_1\) and set \(X_q' = X_q' \cup \pi \hat{\mu}_t(i)\)

19. end for

20. end if

21. end for

22. for \(i = 0, 1, \ldots, \eta_3 \log n - 1\) do

23. Set \(\hat{\mu}_t = \hat{\mu}_t(\pi \hat{\mu}_t(i), t_2 = 0\) and set \(P_1 = P_1 \cup \pi \hat{\mu}_t(i)\)

24. end for

25. Set \(P_2 = \cup_q X_q'\).

26. Return \(P_1, P_2, X_1, X_2, \ldots, X_q\) and \(\hat{U}_{it_2}\) for all \(i \in [n]\).

\(\eta_1 = U_{\pi \hat{\mu}_t}(n)t_2\) and \(p_{\eta_1} \triangleq n\). To wit, \(\eta_1\) is the membership of the element farthest from \(\hat{\mu}_t\) 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 \(p_{\eta_q}' = \arg\min_{j \in [n]} 1[U_{\pi \hat{\mu}_t}(j)t_2 \geq \eta_q]\). Here, we should think of \(p_{\eta_q}'\) to be the index for which \(x_{\pi \hat{\mu}_t(p_{\eta_q})}\) is the element closest from \(\hat{\mu}_t\), such that its membership to the smaller cluster \(t_2\) is at least \(\eta_q\). Now, if \(|p_{\eta_q}' - p_{\eta_{q-1}}| \geq \log n\), we keep \(\eta_q\) unchanged, set \(p_{\eta_q} = p_{\eta_q}'\), and put all elements which are closer to \(\hat{\mu}_t\) than \(x_{\pi \hat{\mu}_t(p_{\eta_{q-1}})}\), but farther than \(x_{\pi \hat{\mu}_t(p_{\eta_q})}\) into a single bin. Notice that each such bin contains at least \(\log n\) elements by definition. However, if \(p_{\eta_q}'\) is the same as \(p_{\eta_{q-1}}\), then the membership queries in the latter binary search to compute \(p_{\eta_q}'\) is wasteful. In order to fix this issue, we check if there is a sufficient number (at least \(\log n\)) of elements between \(p_{\eta_q}'\) and \(p_{\eta_{q-1}}\). If there are not, then we set \(p_{\eta_q}\) 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 \(\eta_q = U_{\pi \hat{\mu}_t(p_{\eta_q})t_2}\). In other words, we set \(p_{\eta_q}\) such that there are at least \(\log n\) elements between \(p_{\eta_q}\) and \(p_{\eta_{q-1}}\), and accordingly, we set \(\eta_q\) to be the membership of \(\pi \hat{\mu}_t(p_{\eta_q})\) to the cluster \(t_2\). Subsequently, we query \(\hat{U}_{it_2}\) for every element that is closer to \(\hat{\mu}_t\) than \(x_{\pi \hat{\mu}_t(p_{\eta_{q-1}})}\) but farther than \(x_{\pi \hat{\mu}_t(p_{\eta_q})}\). 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 \(x_{\pi \hat{\mu}_t(p_{\eta_{q-1}})}\) and \(x_{\pi \hat{\mu}_t(p_{\eta_q})}\) are assigned a single bin.
Algorithm 7 BinarySearch2($\mathcal{X}, \pi, x$): Search for the minimum index $i$ such that $1 - U_{\pi(i)t_1} \geq x$

Input: $\mathcal{O}_{\text{fuzzy}}$.
1: Set low = 1 and high = $n$.
2: while low $\neq$ high do
3: Set mid = $\lfloor (\text{low} + \text{high})/2 \rfloor$.
4: Query $\mathcal{O}_{\text{fuzzy}}(\pi(\text{mid})t_1)$ to obtain $U_{\pi(\text{mid})t_1}$.
5: if $U_{\pi(\text{mid})t_1} \geq 1 - x$ then
6: Set low = mid + 1
7: else
8: Set high = mid
9: end if
10: end while
11: Return low.

To resolve the edge case, we put all those elements which are closer to $\hat{\mu}_{t_1}$ than $x_{\hat{\mu}_{t_1}(p_{c1\log n})}$ into a separate bin. Let $g$ be the total number of bins formed by this algorithm, and say $X_j$ 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 $X_j$ not containing any special elements, we can show that $\max_{i \in X_j} U_{it_2} \leq \min_{i \in X_j} U_{it_2}$, which should be contrasted with the guarantee we have for the sequential algorithm, i.e., for a particular bin $X_j$, $|\max_{i \in X_j} U_{it_2} - \min_{i \in X_j} U_{it_2}| \leq \eta_1$. Finally, we again sub-sample $r$ elements from each such bin $X_j$ with replacement, and denote the resulting subset by $Y_j$. We approximate the center of the second cluster by

$$\hat{\mu}_{t_2} = \frac{\sum_j |X_j|}{r} \sum_{\gamma_j Y_j} U_{it_2}^\alpha x_i + \sum_{\gamma_j \mathcal{P}} U_{it_2}^\alpha x_i}{\sum_j |X_j|}{\sum_{\gamma_j Y_j} U_{it_2}^\alpha + \sum_{\gamma_j \mathcal{P}} U_{it_2}^\alpha},$$

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

4 Proofs

4.1 Proof of Theorem 1

To prove Theorem 1 we will establish some preliminary results.

4.1.1 Auxiliary Lemmata

We start by analyzing the performance of Algorithm 8, which estimates the center of a given cluster using a set of randomly sampled elements. Note that this algorithm is used as a sub-routine of Algorithm 1.

Lemma 3 (Estimate of mean using uniform sampling). Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, and let $\delta \in (0, 1)$. With probability at least $1 - \delta$, Algorithm 8 outputs an estimate $\hat{\mu}_j$ such that

$$||\mu_j - \hat{\mu}_j||^2_2 \leq \frac{4R^2}{Y^2} \sqrt{\frac{c}{m \log \frac{1}{\delta}}},$$

(12)
Algorithm 8 Algorithm for estimating the mean $\mu_j$ for any $j \in [k]$.

**Input:** $\mathcal{X}, \mathcal{O}_{\text{fuzzy}}, \alpha,$ and $m$.

**Output:** $\hat{\mu}_j$

1: Initialize $S \leftarrow \emptyset$.
2: for $s = 1, 2, \ldots, m$ do
3: Sample $i$ uniformly at random from $[n]$ and update $S \leftarrow S \cup \{i\}$.
4: Query $\mathcal{O}_{\text{fuzzy}}(i, j)$.
5: end for
6: Compute $\hat{\mu}_j = \frac{\sum_{i \in S} U_{ij}^\alpha x_i}{\sum_{i \in S} U_{ij}^\alpha}$.

where $Y \triangleq \min_{j \in [k]} \frac{1}{n} \sum_{i \in [n]} U_{ij}^\alpha$, and $c > 0$ is some absolute constant.

**Proof of Lemma 3.** First, note that

$$\hat{\mu}_j = \frac{\sum_{i \in S} U_{ij}^\alpha x_i}{\sum_{i \in S} U_{ij}^\alpha} = \frac{(1/m) \sum_{i \in S} U_{ij}^\alpha x_i \triangleq \bar{\lambda}_x}{Y}. \quad (13)$$

Recall that the true mean of the $j^{th}$ cluster is

$$\mu_j = \frac{\sum_{i \in [n]} U_{ij}^\alpha x_i}{\sum_{i \in [n]} U_{ij}^\alpha} = \frac{(1/n) \sum_{i \in [n]} U_{ij}^\alpha x_i \triangleq \bar{\lambda}_x}{Y} \quad (14)$$

It is clear that $\bar{\lambda}_x$ and $\bar{Y}$ are an unbiased estimators of $\lambda_x$ and $Y$, respectively. Now, note that we can write $\bar{\lambda}_x$ as an average of $m$ i.i.d random variables $\bar{x}_{i_p} = U_{ij}^\alpha x_{i_p}$, where $i_p$ is sampled uniformly at random from $[n]$, and included to the set $S$ in the third step of Algorithm 8 as the $p^{th}$ sample. Similarly, $\bar{Y}$ can also be written as the average of $m$ i.i.d random variables $\bar{Y}_{i_p} = U_{ij}^\alpha$. Further, notice that $\mathbb{E} \bar{x}_{i_p} = \lambda_x$, $\|\bar{x}_{i_p}\|_2 \leq R$, and similarly, $\mathbb{E} \bar{Y}_{i_p} = Y$, and $|\bar{Y}_{i_p}| \leq 1$, for all $p \in \{1, 2, \ldots, m\}$. Next, note that

$$\hat{\mu}_j = \frac{\bar{\lambda}_x}{\bar{Y}} = \frac{\mu_x}{Y} + \frac{\lambda_x - \bar{\lambda}_x}{Y} + \frac{\bar{\lambda}_x - \lambda_x}{Y} \frac{\bar{Y} - Y}{Y}. \quad (15)$$

Thus, using the triangle inequality we get

$$\|\hat{\mu}_j - \mu_j\|_2 \leq \|\frac{\bar{\lambda}_x - \lambda_x}{Y}\|_2 + \|\frac{\lambda_x - \bar{\lambda}_x}{Y}\|_2 \leq \|\frac{\lambda_x - \bar{\lambda}_x}{Y}\|_2 + R \|\frac{\bar{Y} - Y}{Y}\|_2, \quad (16)$$

where in the last inequality we have used the fact that $\|\hat{\mu}_j\|_2 = \|\bar{\lambda}_x/\bar{Y}\|_2 \leq R$. Then, the generalized Hoeffding’s inequality in Lemma 11, implies that with probability at least $1 - \delta$,

$$\|\lambda_x - \bar{\lambda}_x\|_2^2 \leq R^2 \sqrt{\frac{c}{m} \log \frac{1}{\delta}}, \quad (17)$$

for some $c > 0$, and thus,

$$\|\hat{\mu}_j - \mu_j\|_2^2 \leq \frac{4R^2}{Y^2} \sqrt{\frac{c}{m} \log \frac{1}{\delta}}. \quad (18)$$

which concludes the proof. \qed
The following lemma shows that if for a given cluster \( j \) we have been able to approximate its center well enough, then Algorithm 2 computes good estimates of the corresponding membership weights with high probability.

**Lemma 4** (Estimate of membership given estimated center). Let \((\mathcal{X}, \mathcal{P})\) be a consistent center-based clustering instance, and recall the definition of \( \gamma \in \mathbb{R}_+ \) in (8). Assume that for any \( j \in [k] \), there exists an estimator \( \hat{\mu}_j \) such that \( \|\mu_j - \hat{\mu}_j\|_2 \leq \epsilon \) with \( \epsilon \leq \gamma \). Then, Algorithm 2 outputs \( \hat{U}_{ij} \), for \( i \in [n] \), such that

\[
0 \leq U_{ij} - \hat{U}_{ij} \leq \eta, \quad \hat{U}_{ij} \in \{0, \eta, 2\eta, \ldots, 1\}, \quad \forall i \in [n],
\]

for some \( \eta \in \mathbb{R}_+ \), using \( Q = O(\log n/\eta) \) queries to the membership-oracle.

**Proof of Lemma 4.** First, note that since \( \mathcal{P} \) is a consistent center-based clustering, we have

\[
U_{\sigma_{\mu_j}(i_1)j} \geq U_{\sigma_{\mu_j}(i_2)j} \quad \text{if} \quad i_1 < i_2, i_1, i_2 \in [n].
\]

Indeed, when the elements of \( \mathcal{X} \) are sorted in ascending order according to their distance from \( \mu_j \), if \( x_{u_1} \) is closer to \( \mu_j \) than it is to \( x_{u_2} \), then \( U_{u_1j} > U_{u_2j} \). Also, since \( \|\mu_j - \hat{\mu}_j\|_2 \leq \epsilon \leq \gamma \), using (8), this ordering remains the same. Therefore, sorting the elements in \( \mathcal{X} \) in ascending order from \( \hat{\mu}_j \) as in the first step of Algorithm 2 gives the same ordering with respect to the true mean. Now, given \( \eta \in \mathbb{R}_+ \), for each \( s \in \{0, 1, 2, \ldots, 1/\eta\} \), in the third step of Algorithm 2 we binary search to find an index \( \ell_s \) such that

\[
\ell_s = \arg\max_{i \in [n]} U_{\sigma_{\hat{\mu}_j}(i)j} \geq s\eta.
\]

This is done by using \( O(\log n/\eta) \) membership-oracle queries. Finally, in the last three steps of Algorithm 2, for each \( s \in \{0, 1, 2, \ldots, 1/\eta\} \), and for \( i \in \{\ell_s, \ell_s - 1, \ldots, \ell_{s+1} + 1\} \), we assign \( \hat{U}_{\sigma_{\hat{\mu}_j}(i)j} = s\eta \). It is then clear that the estimated memberships satisfy (19), which concludes the proof. \( \square \)

Using the above lemmas we obtain the following.

**Lemma 5** (Estimate all means). Let \((\mathcal{X}, \mathcal{P})\) be a consistent center-based clustering instance, recall the definition of \( \beta \in (0, 1) \) in (9), and let \( \delta \in (0, 1) \). Then, with probability at least \( 1 - \delta \), Algorithm 8 outputs \( \hat{\mu}_j \) such that \( \|\hat{\mu}_j - \mu_j\|_2 \leq \epsilon \), for all \( j \in [k] \), if \( m \geq \left(\frac{4k^4}{\epsilon^4\delta^4\beta} \right)^{1/4} c \log \frac{k}{\delta} \), for some \( c > 0 \).

**Proof.** Using (9) and Hölder’s inequality, for any \( j \in [k] \) we have,

\[
\left( \sum_{i \in [n]} U_{ij}^\alpha \right)^{1/\alpha} \left( \sum_{i \in [n]} U_{ij}^{\alpha/(\alpha-1)} \right)^{(\alpha-1)/\alpha} \geq \sum_{i \in [n]} U_{ij} \geq \frac{\beta n}{k},
\]

which implies that

\[
\left( \sum_{i \in [n]} U_{ij}^\alpha \right)^{1/\alpha} \geq \frac{\beta n}{kn^{(\alpha-1)/\alpha}} = \frac{\beta n^{1/\alpha}}{k},
\]

(23)
and thus
\[ \sum_{i \in [n]} U_{ij}^\alpha \geq \frac{n \beta^\alpha}{k^\alpha}. \] (24)

Therefore,
\[ Y = \min_{j \in [k]} \frac{1}{n} \sum_{i \in [n]} U_{ij}^\alpha \geq \left( \frac{\beta}{k} \right)^\alpha. \]

Now, using Lemma 3 and the last result, taking a union bound over all \( j \in [k] \), we get
\[ \| \hat{\mu}_j - \mu_j \|_2 \leq \frac{2Rk^\alpha}{\beta^\alpha} \left( \frac{c}{m} \log \frac{1}{\delta'} \right)^{1/4} \leq \epsilon. \] (25)

with probability \( 1 - k\delta' \). Rearranging terms and substituting \( \delta = k\delta' \), we get the proof of the lemma.

4.1.2 Proof of Theorem 1

We are now in a position to prove Theorem 1. Using Lemma 5, we can conclude that by taking \( m \geq \left( \frac{Rk^\alpha}{\epsilon^4} \right)^4 \) \( \frac{1}{\log \frac{k}{\delta'}} \) in Algorithm 1, which would require \( km \) membership-oracle queries, we get \( \| \hat{\mu}_j - \mu_j \|_2 \leq \epsilon \), for all \( j \in [k] \). The time-complexity required to estimate all these means is of order \( O(kdm) \). Furthermore, using Lemma 4, using \( O(\log n/\eta) \) membership-oracle queries, Algorithm 1 outputs membership estimates such that (19) holds. This requires a time-complexity of order \( O(\log n/\eta) \). We note, however, that the membership \( \{ \hat{U}_{ij} \}_{j=1}^k \), for any \( i \in [n] \), may not sum up to unity, which is an invalid solution. To fix that in step 7 of Algorithm 1 we add to each \( \hat{U}_{ij} \) a factor of \( \frac{1 - \sum_{j=1}^k \hat{U}_{ij}}{k} \), and then it is clear that the new estimated membership weights sum up to unity. Furthermore, these updated membership weights satisfy \( |\hat{U}_{ij} - U_{ij}| \leq \eta \), for all \( i \in [n] \) and \( j \in [k] \). Therefore, we have shown that Algorithm 1 is \((\epsilon, \eta, Q)\)-solver with probability at least \( 1 - \delta \), which concludes the proof.

4.2 Proof of Theorem 2

In this subsection, we prove Theorem 2 using induction.

4.2.1 Base Case

As can be seen from Algorithm 4, in the first step of this algorithm we sample \( m \) indices uniformly at random and obtain the multiset \( S \subseteq X \). Subsequently, we query \( U_{ij} \) for all \( i \in S \) and \( j \in [k] \), and then, in the third step of the algorithm we choose the cluster \( t_1 \) with the highest membership value, namely,
\[ t_1 = \arg\max_{j \in [k]} \sum_{i \in S} U_{ij}^\alpha. \] (26)

Then, in the fourth step of this algorithm we estimate the mean of this cluster by
\[ \hat{\mu}_{t_1} \triangleq \frac{\sum_{i \in S} U_{ij}^\alpha x_i}{\sum_{i \in S} U_{ij}^\alpha}. \] (27)

We have the following lemma, which is similar to Lemma 3.
Lemma 6 (Guarantees on the largest cluster). Let \((\mathcal{X}, \mathcal{P})\) be a consistent center-based clustering instance, and let \(\delta \in (0, 1)\). With probability at least \(1 - \delta/k\), the estimator in (27) satisfies
\[
\|\hat{\mu}_t - \mu_t\|_2 \leq 2R\left(\frac{c}{m} \log \frac{2k}{\delta}\right)^{1/4} \frac{1}{k^\alpha} - \sqrt{\frac{c}{2m} \log \frac{2k}{\delta}},
\]
where \(c > 0\) is an absolute constant.

Proof of Lemma 6. Recall that Lemma 3 tells us that with probability at least \(1 - \delta/(2k)\),
\[
\|\hat{\mu}_t - \mu_t\|_2 \leq \frac{2R}{\sqrt{V}} \left(\frac{c}{m} \log \frac{2k}{\delta}\right)^{1/4},
\]
where \(V = (1/n) \sum_{i \in [n]} U_{it}^\alpha\). Now, since \(t_1\) is chosen as the cluster with the maximum membership in the subset \(S\), we will first bound \(\hat{Y} \triangleq (1/m) \sum_{i \in S} U_{it}^\alpha\). Notice that \(\sum_{i \in S} \sum_{j=1}^k U_{ij} = m\), and therefore, using Hölder’s inequality we have that for \(\alpha > 1\),
\[
\left(\sum_{i \in S} \sum_{j=1}^k U_{ij}^\alpha\right)^{1/\alpha} \left(\sum_{i \in S} \sum_{j=1}^k 1^{\alpha/(\alpha-1)}\right)^{(\alpha-1)/\alpha} \geq \sum_{i \in S} \sum_{j=1}^k U_{ij} \geq m,
\]
which implies that
\[
\left(\sum_{i \in S} \sum_{j=1}^k U_{ij}^\alpha\right)^{1/\alpha} \geq \frac{m}{(km)^{(\alpha-1)/\alpha}} = \frac{m^{1/\alpha}}{k^{(\alpha-1)/\alpha}},
\]
and therefore,
\[
\sum_{i \in S} \sum_{j=1}^k U_{ij}^\alpha \geq \frac{m}{k^{\alpha-1}}.
\]
Accordingly, we must have \(\sum_{i \in S} U_{it_1} \geq \frac{m}{k^{\alpha}}\), which in turn implies that \(\hat{Y} \geq \frac{1}{k^\alpha}\). Next, using Hoeffding’s inequality in Lemma 10 we obtain that \(|Y - \hat{Y}| \leq \sqrt{\frac{c}{2m} \log \frac{2k}{\delta}}\), with probability at least \(1 - \delta/(2k)\), and therefore \(Y \geq \hat{Y} - \sqrt{\frac{c}{2m} \log \frac{2k}{\delta}}\), which concludes the proof.

Using the above result and Lemma 4 we obtain the following corollaries.

Corollary 4. Let \((\mathcal{X}, \mathcal{P})\) be a consistent center-based clustering instance, and let \(\delta \in (0, 1)\). Then, with probability at least \(1 - \delta/k\), the estimator in (27) satisfies \(\|\hat{\mu}_t - \mu_t\|_2 \leq \epsilon\), if \(m \geq \frac{R^4k^{4\alpha}}{\epsilon^2} \log \frac{2k}{\delta}\). Also, this estimate requires \(O\left(\frac{R^4k^{4\alpha+1}}{\epsilon^2} \log \frac{2k}{\delta}\right)\) membership-oracle queries, and a time-complexity of \(O\left(\frac{dR^4k^{4\alpha}}{\epsilon^2} \log \frac{2k}{\delta}\right)\).

Proof of Corollary 4. The proof follows from rearranging terms of Lemma 6. The query complexity follows from the fact that we query the membership values \(U_{ij}\) for all the \(i \in S, j \in [k]\) and the time-complexity follows from the fact that we take the mean of \(m\) \(d\)-dimensional vectors in order to return the estimator \(\hat{\mu}_t\).
Corollary 5. Let \((\mathcal{X}, \mathcal{P})\) be a consistent center-based clustering instance, and recall the definition of \(\gamma \in \mathbb{R}_+\) in (8). Assume that there exists an estimator \(\hat{\mu}_t\) such that \(\|\mu_{t_t} - \hat{\mu}_t\|_2 \leq \epsilon\) with \(\epsilon \leq \gamma\). Then, Algorithm 4 outputs \(\hat{U}_{it_t}\), for \(i \in [n]\), such that

\[
0 \leq U_{it_t} - \hat{U}_{it_t} \leq \eta_1, \quad \hat{U}_{it_t} \in \{0, \eta_1, 2\eta_1, \ldots, 1\}, \quad \forall i \in [n],
\]

for some \(\eta \in \mathbb{R}_+\), using \(Q = O(\log n/\eta_1)\) queries to the membership-oracle.

Proof of Corollary 5. The proof of this lemma follows the same steps as in the proof of Lemma 4.

Corollaries 4 and 5 show that the base case of our induction is correct.

### 4.2.2 Induction Hypothesis

We condition on the event that we have been able to estimate \(\mu_{t_1}, \mu_{t_2}, \ldots, \mu_{t_\ell}\) by their corresponding estimators \(\hat{\mu}_{t_1}, \hat{\mu}_{t_2}, \ldots, \hat{\mu}_{t_\ell}\), respectively, such that

\[
\|\hat{\mu}_{t_j} - \mu_{t_j}\|_2 \leq \epsilon, \quad \forall j \in [\ell],
\]

and further, we have been able to recover \(U_{it_j}\), for all \(i \in [n]\) and \(j \in [\ell]\), in the sense that

\[
0 \leq U_{it_j} - \hat{U}_{it_j} \leq \eta_1, \quad \hat{U}_{it_j} \in \{0, \eta_1, 2\eta_1, \ldots, 1\}, \quad \forall i \in [n], j \in [\ell].
\]

The induction hypothesis states that we have been able to estimate the means of \(\ell\) clusters up to an error of \(\epsilon\) and subsequently also estimated the memberships of every element in \(\mathcal{X}\) to those \(\ell\) clusters such that the estimated memberships are an integral multiple of \(\eta_1\) and also have a precision error of at most \(\eta_1\). Given the induction hypothesis, we characterize next the sufficient query complexity and time-complexity required in order to estimate the mean of the \((\ell + 1)\)th cluster and its membership weights.

### 4.2.3 Inductive Step

Let \(Z_\ell \triangleq \sum_{i \in [n]} \sum_{j \in [\ell]} \hat{U}_{it_j}\), and define \(\mathcal{X}_s \triangleq \{i \in [n] : \sum_{j \in [\ell]} \hat{U}_{it_j} = s\eta_1\}\), for \(s \in \{0, 1, \ldots, 1/\eta_1\}\). In step 10 of Algorithm 4, we sub-sample \(r\) indices uniformly at random with replacement from each of the sets \(\mathcal{X}_s\), for \(s \in \{0, 1, 2, \ldots, 1/\eta_1\}\). Let us denote the multi-set of indices chosen from \(\mathcal{X}_s\) by \(\mathcal{Y}_s\). Subsequently, in the step 12 of Algorithm 4, for every \(s \in \{0, 1, 2, \ldots, 1/\eta_1\}\) and for every element in \(\mathcal{Y}_s\), we query the memberships to all the clusters except \(t_1, t_2, \ldots, t_\ell\) from the oracle, and set

\[
t_{\ell+1} = \arg\max_{j \in [\ell]\setminus\{t_1, t_2, \ldots, t_\ell\}} \sum_s \frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} U_{it_j}^s.
\]

Step 13 of Algorithm 4 computes

\[
\hat{\mu}_{t_{\ell+1}} \triangleq \frac{\sum_s |\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \frac{U_{it_{\ell+1}}^s \cdot x_i}{\sum_s |\mathcal{X}_s|}.
\]

The following analysis the performance of the estimator in (37). We relegate the proof of this result to the end of this section.
Lemma 7 (Performance of (37)). Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, and let $\delta \in (0, 1)$. With probability at least $1 - \delta/k$, the estimator in (37) satisfies

$$\|\hat{\mu}_{t+1} - \mu_{t+1}\|_2 \leq \frac{2R \left( \frac{c}{r} \log \frac{4k}{\eta_1} \right)^{1/4} (n - Z_\ell + n t \eta_1)}{\left( n - Z_\ell - n t \eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right)^{1/2}} - \left( n - Z_\ell - n t \eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right)^{1/2},$$

where $c > 0$ is an absolute constant. The query and time-complexity required for evaluating this estimator are of order $O(k \log n/k)$ and $O(r \log k/\eta_1)$, respectively.

Using the above result and Lemma 4 we obtain the following corollaries.

Corollary 6. Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, recall the definition of $\beta \in (0, 1)$ in (9), and let $\delta \in (0, 1)$. Then, with probability at least $1 - \delta/k$, the estimator in (37) satisfies $\|\hat{\mu}_{t+1} - \mu_{t+1}\|_2 \leq \epsilon$, if $r \geq \frac{c R^4 t \log 4k}{\epsilon^2 r^2 \log \frac{4k}{\eta_1 \delta}}$ and $\eta_1 \leq \frac{1}{k} \left( 1 - \frac{\beta}{k} \right)$. Also, this estimate requires $O \left( \frac{R^4 k^4 t \log 4k}{\epsilon^2 r^2 \log \frac{4k}{\eta_1 \delta}} \right)$ membership-oracle queries, and a time-complexity of $O \left( \frac{R^4 k^4 t \log 4k}{\epsilon^2 r^2 \log \frac{4k}{\eta_1 \delta}} \right)$.

Proof of Corollary 6. Using (9) and the fact that $n - Z_\ell = \sum_{i \in [n]} \sum_{j \in [k] \setminus \ell} U_{ij}$, we have the following upper and lower bound

$$\frac{n - Z_\ell}{k - \ell} \geq \frac{\beta n}{k} \quad \text{and} \quad n - Z_\ell \leq n - \frac{\beta n}{k},$$

which follows from the fact that the average membership size of the any $k - \ell$ clusters must be larger than the membership size of the smallest cluster. Thus, if $\eta_1 \leq \frac{1}{k} \left( 1 - \frac{\beta}{k} \right)$, as claimed in the statement of the lemma, we have $n t \eta_1 \leq n - Z_\ell$. With the chosen values of $\eta_1$ and $r$, and the fact that $\ell < k$, we get $n t \eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} = o(n - Z_\ell)$. Therefore,

$$\|\hat{\mu}_{t+1} - \mu_{t+1}\|_2 \leq \frac{4R \left( \frac{\xi}{r} \log \frac{4k}{\eta_1 \delta} \right)^{1/4} (n - Z_\ell)}{\left( \frac{n - Z_\ell}{n^{\alpha-1} (k - \ell)^{\alpha}} - 2 \frac{c}{2r} \log \frac{4k}{\eta_1 \delta} (n - Z_\ell) \right)^{1/4}}$$

$$= \frac{4R \left( \frac{\xi}{r} \log \frac{4k}{\eta_1 \delta} \right)^{1/4}}{\left( \frac{n - Z_\ell}{n^{\alpha-1} (k - \ell)^{\alpha}} - 2 \frac{c}{2r} \log \frac{4k}{\eta_1 \delta} \right)^{1/4}} \leq \frac{\beta^{\alpha-1}}{k^{\alpha} (k - \ell)^{\alpha}} - 2 \frac{c}{2r} \log \frac{4k}{\eta_1 \delta}. \quad (42)$$

Again, for the chosen value of $r$, it is clear that $\sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} = o \left( \frac{\beta^{\alpha-1}}{k^{\alpha} (k - \ell)^{\alpha}} \right)$, and thus we get that $\|\hat{\mu}_{t+1} - \mu_{t+1}\|_2 \leq \epsilon$, with probability at least $1 - \delta/k$. \hfill \box

Corollary 7. Let $(\mathcal{X}, \mathcal{P})$ be a consistent center-based clustering instance, and recall the definition of $\gamma \in \mathbb{R}_+$ in (8). Assume that there exists an estimator $\hat{\mu}_{t+1}$ such that $\|\mu_{t+1} - \hat{\mu}_{t+1}\|_2 \leq \epsilon$ with $\epsilon \leq \gamma$. Then, Algorithm 4 outputs $\hat{U}_{it_{t+1}}$, for $i \in [n]$, such that

$$0 \leq U_{it_{t+1}} - \hat{U}_{it_{t+1}} \leq \eta_1, \quad \hat{U}_{it_{t+1}} \in \{0, \eta_1, 2\eta_1, \ldots, 1\}, \quad \forall i \in [n],$$

for some $\eta \in \mathbb{R}_+$, using $Q = O \left( \log n/\eta_1 \right)$ queries to the membership-oracle.
Proof of Corollary 7. The proof of this lemma follows the same steps as in the proof of Lemma 4. \qed

4.2.4 Proof of Theorem 2

We are now in a position to proof Theorem 2. To that end, we use our induction mechanism. Specifically, Corollaries 4 and 5 prove the base case for the first cluster \( t_1 \). Subsequently, Corollaries 6 and 7 prove the induction step after taking a union bound over all clusters and using \( \eta_1 = \frac{1}{k} \left( 1 - \frac{\beta}{k} \right) \). Finally, we can use Lemma 4 in order to estimate the memberships \( U_{ij} \forall i \in [n], j \in [k] \) up to a precision of \( \eta_2 \) using an addition query and time-complexity of \( O(k \log n/\eta_2) \).

It is left to prove Lemma 7.

Proof of Lemma 7. Let \( \Sigma \triangleq \{ 0, 1, \ldots, 1/\eta_1 \} \). We have

\[
\mu_{\ell+1} = \frac{\sum_{i\in[n]} U_{it_{\ell+1}} x_i}{\sum_{i\in[n]} U_{it_{\ell+1}}} \triangleq \frac{\sum_{s\in\Sigma} \sum_{i\in\lambda_s} U_{it_{\ell+1}} x_i}{\sum_{s\in\Sigma} \sum_{i\in\lambda_s} U_{it_{\ell+1}}} \triangleq \frac{\sum_{s\in\Sigma} \lambda_s}{\sum_{s\in\Sigma} \lambda_s}, \tag{44}
\]

and that

\[
\hat{\mu}_{\ell+1} = \frac{\sum_{s\in\Sigma} \bar{\lambda}_s}{\sum_{s\in\Sigma} \bar{\lambda}_s} \triangleq \frac{\sum_{s\in\Sigma} \lambda_s}{\sum_{s\in\Sigma} \lambda_s} + \frac{\sum_{s\in\Sigma} \lambda_s - \bar{\lambda}_s}{\sum_{s\in\Sigma} \lambda_s - \bar{\lambda}_s} + \frac{\sum_{s\in\Sigma} \bar{\lambda}_s}{\sum_{s\in\Sigma} \lambda_s} \tag{45}
\]

Now, note that we can write \( \bar{\lambda}_s \) as an average of \( r \) i.i.d random variables \( \bar{x}_{s,ip} \triangleq |\lambda_s| U_{ipt_{\ell+1}} x_{ip} \), where \( i_p \) is sampled uniformly at random from \( [n] \), and included to the set \( \mathcal{Y}_s \) in the step 9 of Algorithm 4 as the \( p \)th sample. Similarly, \( \mathcal{Y}_s \) can also be written as the average of \( r \) i.i.d random variables \( \bar{\mathcal{Y}}_{s,ip} = |\lambda_s| U_{ipt_{\ell+1}} x_{ip} \). Therefore, it is evident that \( \mathbb{E} \bar{x}_{s,ip} = \lambda_s \) and \( \mathbb{E} \bar{\mathcal{Y}}_{s,ip} = \mathcal{Y}_s \) for all \( p \in [r] \). This implies that the numerator and denominator of (45) are both unbiased.

Now, note that

\[
\bar{\mu}_{\ell+1} = \frac{\sum_{s\in\Sigma} \bar{\lambda}_s}{\sum_{s\in\Sigma} \bar{\lambda}_s} \triangleq \frac{\sum_{s\in\Sigma} \lambda_s}{\sum_{s\in\Sigma} \lambda_s} + \frac{\sum_{s\in\Sigma} \lambda_s - \bar{\lambda}_s}{\sum_{s\in\Sigma} \lambda_s - \bar{\lambda}_s} + \frac{\sum_{s\in\Sigma} \bar{\lambda}_s}{\sum_{s\in\Sigma} \lambda_s} \tag{46}
\]

Thus, using the triangle inequality we get

\[
\| \hat{\mu}_{\ell+1} - \mu_{\ell+1} \|_2 \leq \| \sum_{s\in\Sigma} \bar{\lambda}_s - \bar{\mu}_{\ell+1} \|_2 + R \sum_{s\in\Sigma} \mathcal{Y}_s - \bar{\mathcal{Y}}_s \|_2, \tag{47}
\]

where the last inequality follows from the fact that \( \| \bar{\mu}_{\ell+1} \|_2 = \| \sum_{s\in\Sigma} \bar{\lambda}_s \|_2 \leq R \).

Next, we note that for \( i \in \mathcal{Y}_s \), we have

\[
U_{it_{\ell+1}}^\alpha \leq (1 - \sum_{j\in[t]} U_{ij})^\alpha \leq (1 - \sum_{j\in[t]} \bar{U}_{ij})^\alpha = (1 - s \eta_1)^\alpha, \tag{48}
\]

where we have used the induction hypothesis in (35). Also, using Lemmas 10 and 11, for all \( s \in \Sigma \), with probability at least \( 1 - \delta/2k \),

\[
\| \bar{\lambda}_s - \lambda_s \|_2 \leq R |\lambda_s| (1 - s \eta_1)^\alpha \left( \frac{c}{r} \log \frac{4k}{\eta_1} \right)^{1/4}
\leq R |\lambda_s| (1 - s \eta_1) \left( \frac{c}{r} \log \frac{4k}{\eta_1} \right)^{1/4}, \tag{49}
\]

21
and
\[ |\bar{Y}_s - Y_s| \leq |X_s|(1 - s\eta_1)^\alpha \sqrt{c \frac{\log 4k}{r \eta_1 \delta}} \]
\[ \leq |X_s|(1 - s\eta_1)^\alpha \sqrt{\frac{c}{2r} \log 4k \eta_1 \delta}. \] (50)

Using the induction hypothesis in (35) once again, we have
\[ Z_\ell = \sum_{s \in \Sigma} \sum_{i \in X_s} \sum_{j \in [\ell]} U_{ij} \leq \sum_{s \in \Sigma} \sum_{i \in X_s} \left( \ell \eta_1 + \sum_{j \in [\ell]} \hat{U}_{ij} \right) \leq \sum_{s \in \Sigma} |X_s| s\eta_1 + n\ell \eta_1, \] (51)
and thus
\[ \sum_{s \in \Sigma} |X_s|(1 - s\eta_1) \leq n - Z_\ell + n\ell \eta_1. \] (52)

Next, we lower bound \( \sum_{s \in \Sigma} Y_s \). To that end, in light of (47), it is suffice to bound \( \sum_{s \in \Sigma} \bar{Y}_s \). Using Hölder’s inequality we have
\[ \left( \sum_{i \in Y_s, j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij} \right)^{1/\alpha} \left( \sum_{i \in Y_s, j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} 1^{\alpha/(\alpha-1)} \right)^{(\alpha-1)/\alpha} \geq \sum_{i \in Y_s, j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}, \] (53)
which implies that
\[ \left( \sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij} \right)^{1/\alpha} \geq \frac{\sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}}{(r(k - \ell))^{(\alpha-1)/\alpha}}, \] (54)
and therefore,
\[ \sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij} \geq \left( \frac{\sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}}{(r(k - \ell))^{(\alpha-1)}} \right)^\alpha. \] (55)

To further lower bound the r.h.s. of (55), we use the power means inequality and ger
\[ \frac{\sum_{s \in \Sigma} |X_s|}{r} \left( \sum_{s \in \Sigma} \frac{\sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}}{r} \right)^\alpha \geq \left( \frac{\sum_{s \in \Sigma} |X_s|}{r} \sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij} \right)^\alpha \] (56)
\[ \sum_{s \in \Sigma} \frac{|X_s|}{r} \sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}^\alpha \geq \left( \frac{\sum_{s \in \Sigma} \frac{|X_s|}{r} \sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}}{(n(k - \ell))^{(\alpha-1)}} \right)^\alpha. \] (57)

Thus, the fact that \( \sum_{s \in \Sigma} \frac{|X_s|}{r} = \frac{n}{r} \), combined with (55) and (56), imply that
\[ \sum_{s \in \Sigma} \frac{|X_s|}{r} \sum_{i \in Y_s} \sum_{j \in [k]\setminus\{t_1, t_2, \ldots, t_\ell\}} U_{ij}^\alpha \geq \left( \frac{n - \sum_{s \in \Sigma} \frac{|X_s|}{r} \sum_{i \in Y_s} \sum_{j \in [t_1, t_2, \ldots, t_\ell]} U_{ij}}{(n(k - \ell))^{(\alpha-1)}} \right)^\alpha. \] (58)
We next upper bound the term inside the brackets at the r.h.s. of (58). To that end, for a given \( s \), we define the random variables

\[
H_{s,i_p} \triangleq |\mathcal{X}_s| \left( \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} - s\eta_1 \right),
\]

where where \( i_p \) is sampled uniformly at random from \([n]\), and included to the set \( \mathcal{Y}_s \) in the step 9 of Algorithm 4 as the \( p \)th sample. With this definition, it is evident that \( \frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \left( \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} - s\eta_1 \right) \) can be written as the average of these \( r \) i.i.d random variables, namely,

\[
\frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \left( \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} - s\eta_1 \right) = \frac{1}{r} \sum_{i \in \mathcal{Y}_s} H_{s,i}.
\]

Note that \( \mathbb{E}H_{s,i_p} = \sum_{i \in \mathcal{X}_s} \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} - |\mathcal{X}_s|\eta_1 \) and \( |H_{s,i_p}| \leq \ell\eta_1 \) for all \( i_p \in [r] \). For simplicity of notation we define \( Z_{s\ell} \triangleq \sum_{i \in \mathcal{X}_s} \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} \). Then, using Hoeffding’s inequality in Lemma 10, we have with probability at least \( 1 - \delta/4k \),

\[
\left| \frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \left( \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} - s\eta_1 \right) - Z_{s\ell} + |\mathcal{X}_s|\eta_1 \right| \leq |\mathcal{X}_s|\ell\eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}},
\]

and thus,

\[
\frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} \leq \max \left( |\mathcal{X}_s|, Z_{s\ell} + |\mathcal{X}_s|\ell\eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right) \leq Z_{s\ell} + \max \left( |\mathcal{X}_s| - Z_{s\ell}, |\mathcal{X}_s|\ell\eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right).
\]

Summing (63) over \( s \in \Sigma \) and using the fact that \( \sum_{s \in \Sigma} Z_{s\ell} = Z_\ell \), we obtain

\[
n - \sum_{s \in \Sigma} \frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \sum_{j \in \{t_1,t_2,\ldots,t_\ell\}} U_{ij} \geq n - Z_\ell - \min \left( n - Z_\ell, n\ell\eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right).
\]

Substituting the last inequality in (58), we finally get

\[
\sum_{s \in \Sigma} \frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} \sum_{j \in [k]\setminus\{t_1,t_2,\ldots,t_\ell\}} U_{ij}^\alpha \geq \frac{n - Z_\ell - n\ell\eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}}}{(n(k - \ell))^{(\alpha - 1)}}. 
\]

Next, recall that the index of the \((\ell + 1)^{th}\) cluster is chosen as

\[
t_{\ell + 1} = \argmax_{j \in [k]\setminus\{t_1,t_2,\ldots,t_\ell\}} \sum_{s \in \Sigma} \frac{|\mathcal{X}_s|}{r} \sum_{i \in \mathcal{Y}_s} U_{ij}^\alpha,
\]

23
and therefore,
\[ \sum_{s \in \Sigma} \bar{Y}_s = \sum_{s \in \Sigma} \frac{|X_s|}{r} \sum_{i \in Y_s} U_{it+1}^\alpha \geq \frac{\left( n - Z_{\ell} - n \ell \eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right) ^\alpha }{n^{\alpha - 1} (k - \ell)^\alpha} . \]  

Combining (50), (52), and (68), we get a lower bound on \( \sum_{s \in \Sigma} Y_s \) as follows
\[ \sum_{s \in \Sigma} Y_s \geq \frac{\left( n - Z_{\ell} - n \ell \eta_1 \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} \right) ^\alpha }{n^{\alpha - 1} (k - \ell)^\alpha} - (n - Z_{\ell} - n \ell \eta_1) \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} . \]  

Finally, combining (47), (49), and (69), we get
\[ \| \hat{\mu}_{t+1} - \mu_{t+1} \|_2 \leq \sum_{s \in \Sigma} \| \bar{Y}_s \|_2 + R \sum_{s \in \Sigma} \frac{Y_s - \bar{Y}_s}{\sum_{s \in \Sigma} Y_s} \|_2 \leq 2R \left( \frac{c}{r} \log \frac{4k}{\eta_1 \delta} \right) ^{1/4} \frac{(n - Z_{\ell} + n \ell \eta_1)}{(n - Z_{\ell} - n \ell \eta_1) \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}}} - (n - Z_{\ell} - n \ell \eta_1) \sqrt{\frac{c}{2r} \log \frac{4k}{\eta_1 \delta}} , \]  

with probability at least \( 1 - \delta/k \), which concludes the proof. \( \square \)

### 4.3 Proof of Theorem 3

To prove Theorem 3 we will establish some preliminary results.

#### 4.3.1 Auxiliary Lemmata

We start with the following result which shows that given a good estimate for the larger cluster among the two, we can approximate the membership weights of the smaller cluster reliably. This is done in Algorithm 6.

**Lemma 8.** Let \( (X, \mathcal{P}) \) be a consistent center-based clustering instance, recall the definition of \( \gamma \in \mathbb{R}_+ \) in (8), and let \( k = 2 \). Assume that there exists an estimator \( \hat{\mu}_{t_1} \) such that \( \| \mu_1 - \hat{\mu}_{t_1} \|_2 \leq \epsilon \) with \( \epsilon \leq \gamma \). Then, Algorithm 6 outputs \( \hat{U}_{it_2} \), for \( i \in [n] \), such that
\[ \hat{U}_{it_2} \in \mathcal{A} \subset \{ U_{it_2} : i \in [n] \} , \]  

with \( |\mathcal{A}| = O(\log_2 n) \), and
\[ \max_{i \in [n]: U_{it_2} = x} U_{it_2} \leq \max \left( \frac{\max_{i \in [n]} U_{it_2}}{n^3}, \min_{i \in [n]: U_{it_2} = x} 2U_{it_2} \right) , \]  

for all \( x \in \mathcal{A} \), and for some \( \eta \in \mathbb{R}_+ \), using \( Q = O(\log_2 n) \) queries to the membership-oracle, and time-complexity \( O(\log_2^2 n) \).
Proof of Lemma 8. First, note that since $\mathcal{P}$ is a consistent center-based clustering, we have

$$U_{\sigma(t_1)_{t_2}} \leq U_{\sigma(t_2)_{t_2}} \quad \text{if} \quad r_1 < r_2, \ r_1, r_2 \in [n]. \quad (74)$$

Indeed, when the elements of $\mathcal{X}$ are sorted in ascending order according to their distance from $\mu_{t_1}$, if $x_i$ is closer to $\mu_{t_1}$ than it is to $x_j$, then $U_{it_1} \geq U_{jt_1}$ and thus $U_{it_2} \leq U_{jt_2}$. Also, since $||\mu_{t_1} - \mu_{t_2}|| \leq \epsilon \leq \gamma$, using (8), this ordering remains the same. Therefore, sorting the elements in $\mathcal{X}$ in ascending order from $\mu_{t_1}$ as in the first step of Algorithm 6 gives the same ordering with respect to the true mean. Now, given $\eta \in \mathbb{R}_+$, we search for the index

$$p_{\eta} \triangleq \arg\min_{j \in [n]} \left[ U_{\pi(j)_{t_2}} \geq \eta \right], \quad (75)$$

which can be done by using the binary search routine in Algorithm 7, which ask for at most $O(\log n)$ membership-oracle queries. We will do this step for $\eta_1, \eta_2, \ldots, \eta_3 \log n$, as described in Algorithm 6. The values of $\{\eta_i\}$ are chosen as follows. We initialize $\eta_1 = U_{\sigma(t_1)_{t_2}}$, and $p_{\eta_1} \triangleq n$, and update the other values of $\eta_i$’s recursively as follows. Let $\mathcal{V} \triangleq \{1, 2, \ldots, 3 \log n\}$. For each $q \in \mathcal{V} \setminus 1$, we first set $\eta_q = \eta_{q-1}/2$ and subsequently, if $|p_{\eta_q} - p_{\eta_{q-1}}| \geq \log n$, then $\eta_q$ remains unchanged and we set $p_{\eta_q} = p_{\eta_{q-1}}$. Otherwise, if $|p_{\eta_q} - p_{\eta_{q-1}}| < \log n$, then we update both $\eta_q$ and $p_{\eta_q}$ as follows:

$$p_{\eta_q} = \min(0, p_{\eta_{q-1}} - 1 - \log n) \quad (76)$$

$$\eta_q = U_{\pi \ni (p_{\eta_q})t_2} \quad (77)$$

For each value of $q \in \mathcal{V}$, we initialize two sets $\mathcal{X}_q, \mathcal{X}'_q = \emptyset$. If $|p_{\eta_q} - p_{\eta_{q-1}}| \geq \log n$, then we update $\mathcal{X}_q = \{\pi(i) : p_{\eta_q} \leq i \leq p_{\eta_{q-1}} - 1, i \in [n]\}$ and if $|p_{\eta_q} - p_{\eta_{q-1}}| < \log n$, we update $\mathcal{X}'_q = \{\pi(i) : p_{\eta_q} \leq i \leq p_{\eta_{q-1}} - 1, i \in [n]\}$. It is clear that $\eta_q \leq \eta_{q-1}/2$ and therefore, we must have $\eta_3 \log n \leq \eta_1/n^3$. We now define the following sets:

$$\mathcal{P}_1 \triangleq \{i \in [n] : U_{it_2} \leq \eta_3 \log n\}, \quad (78)$$

$$\mathcal{P}_2 \triangleq \bigcup_q \mathcal{X}_q', \quad (79)$$

For each $i \in \mathcal{P}_1$, we estimate $\hat{U}_{it_2} = 0$ and since $U_{it_2} \leq U_{\pi(t_1)_{t_2}}/n^3$ and $U_{\pi(t_1)_{t_2}} = \max_{i \in [n]} U_{it_2}$, we must have

$$|\hat{U}_{it_2} - U_{it_2}| \leq \frac{\max_{i \in [n]} U_{it_2}}{n^3} \quad \text{for all} \quad i \in \mathcal{P}_1. \quad (80)$$

For each $i \in \mathcal{P}_2$, we query $U_{it_2}$ and estimate $\hat{U}_{it_2} = U_{it_2}$. Notice that we have

$$[n] \setminus \{\mathcal{P}_1 \cup \mathcal{P}_2\} = \bigcup_q \mathcal{X}_q, \quad (81)$$

and therefore for each $\mathcal{X}_q$ such that $\mathcal{X}_q \neq \emptyset$, we estimate $\hat{U}_{it_2} = U_{\pi(t_1)_{t_2}}$ for all $i \in \mathcal{X}_q$. Now, since

$$U_{\pi(t_k+1)_{t_2}} \leq \eta_k - 1 \quad \text{and} \quad U_{\pi(t_k)_{t_2}} \geq \eta_k = \frac{\eta_k - 1}{2}, \quad (82)$$

25
we must have that for all \( i \in \mathcal{X}_q \) such that \( \mathcal{X}_q \neq \phi \),

\[
\max_{i \in \mathcal{X}_q} \mathbf{U}_{it_2} \leq 2 \min_{i \in \mathcal{X}_q} \mathbf{U}_{it_2},
\]

which proves the lemma. Note that each binary search step in Algorithm 7 requires \( O(\log n) \) queries, and may require an additional \( O(\log n) \) queries if \( \mathcal{X}_q' \neq \phi \). Similarly, the time-complexity of each binary step is \( O(\log n) \) as well. Since we are making at most \( 3 \log n \) binary search steps, we get the desired query and time-complexity results.

Lemma 8 implies the following corollary.

**Corollary 8.** Consider the setting of Lemma 8. Then,

\[
\sum_{i \in [n]} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| \leq \sum_{i \in [n]} \mathbf{U}_{it_2} + \frac{\max_{i \in [n]} \mathbf{U}_{it_2}}{n^2}.
\]

**Proof.** Recall that \( \mathcal{V} \triangleq \{1, 2, \ldots, 3 \log n\} \). Then, note that

\[
\sum_{i \in [n]} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| = \sum_{i \in \mathcal{P}_1} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| + \sum_{i \in \mathcal{P}_2} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| + \sum_{q \in \mathcal{V}} \sum_{i \in \mathcal{X}_q : |\mathcal{X}_q| \neq \phi} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right|.
\]

Next, we bound each of the terms on the right-hand side of the above inequality. We have,

\[
\sum_{i \in \mathcal{P}_2} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| = 0,
\]

\[
\sum_{i \in \mathcal{P}_1} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| \leq \frac{|\mathcal{P}_2| \max_{i \in [n]} \mathbf{U}_{it_2}}{n^3} \leq \frac{\max_{i \in [n]} \mathbf{U}_{it_2}}{n^2}.
\]

Finally, for each \( q \in \mathcal{V} \) such that \( \mathcal{X}_q \neq \phi \), recall that \( \hat{\mathbf{U}}_{it_2} = \min_{j \in \mathcal{X}_q} \mathbf{U}_{it_2}, \) for all \( i \in \mathcal{X}_q \), and therefore,

\[
\max_{i \in \mathcal{X}_q} \mathbf{U}_{it_2} \leq 2 \min_{i \in \mathcal{X}_q} \mathbf{U}_{it_2} \implies \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| \leq \frac{\max_{i \in \mathcal{X}_q} \mathbf{U}_{it_2}}{2} \leq \frac{\sum_{i \in \mathcal{X}_q} \mathbf{U}_{it_2}}{|\mathcal{X}_q|}, \quad \forall i \in \mathcal{X}_q.
\]

Thus,

\[
\sum_{q \in \mathcal{V}} \sum_{i \in \mathcal{X}_q} \left| \hat{\mathbf{U}}_{it_2} - \mathbf{U}_{it_2} \right| \leq \sum_{i \in [n]} \mathbf{U}_{it_2},
\]

which proves the desired result.

It is left to estimate the center of the smaller cluster among the two. This is done in Steps 7-13 of Algorithm 5. Specifically, for each \( q \in \mathcal{V} \), we randomly sample with replacement \( r \) elements from \( \mathcal{X}_q \) where \( \mathcal{X}_q \neq \phi \). We denote this sampled multi-set by \( \mathcal{Y}_q \), and query \( \mathbf{U}_{it_2} \), for each \( i \in \mathcal{Y}_q \). We also sample \( r \) elements from \( \mathcal{P}_1 \), and denote this sampled multi-set by \( \mathcal{Q} \), and query \( \mathbf{U}_{it_2} \) for each \( i \in \mathcal{Q} \). Note that we have already queried \( \mathbf{U}_{it_2} \) for every \( i \in \mathcal{P}_2 \) in Step 19 of Algorithm 6. Subsequently, we propose the following estimate for the center of the smaller cluster,

\[
\hat{\mu}_{it_2} = \frac{\sum_{q \in \mathcal{V}: \mathcal{X}_q \neq \phi} \frac{|\mathcal{X}_q|}{r} \sum_{i \in \mathcal{Y}_q} \sum_{i \in \mathcal{P}_2} \mathbf{U}_{it_2} \mathbf{x}_i + \sum_{i \in \mathcal{P}_2} \mathbf{U}_{it_2} \mathbf{x}_i + \sum_{i \in \mathcal{Q}} \frac{|\mathcal{P}_2|}{r} \mathbf{U}_{it_2} \mathbf{x}_i}{\sum_{q \in \mathcal{V}: \mathcal{X}_q \neq \phi} \frac{|\mathcal{X}_q|}{r} \sum_{i \in \mathcal{Y}_q} \mathbf{U}_{it_2} + \sum_{i \in \mathcal{P}_2} \mathbf{U}_{it_2} + \sum_{i \in \mathcal{Q}} \frac{|\mathcal{P}_2|}{r} \mathbf{U}_{it_2}}.
\]

The following result gives guarantees on the estimation error associated with the smaller cluster among the two.
Lemma 9. Let \((X, \mathcal{P})\) be a consistent center-based clustering instance, and let \(\delta \in (0, 1)\). Then, with probability at least \(1 - \delta/2\), the estimator in (88) satisfies \(\|\hat{\mu}_{t_2} - \mu_{t_2}\|_2 \leq \epsilon\), if \(r \geq c' \frac{n \log \frac{2\sqrt{d}}{\eta_0}}{\epsilon^2}\), where \(c' > 0\) is an absolute constant. Also, this estimate requires \(O(r \log n)\) membership-oracle queries, and a time-complexity of \(O(dr \log n)\).

Proof of Lemma 9. First, note that \([n] = \bigcup_{q: x_q \neq \phi} \mathcal{X}_q \cup \mathcal{P}_1 \cup \mathcal{P}_2\). Therefore,

\[
\mu_{t_2} = \frac{\sum_{q: x_q \neq \phi} \lambda_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 x_i + \rho}{\sum_{q: x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}},
\]

where \(\lambda_q \triangleq \sum_{i \in \mathcal{X}_q} U_{it_2}^0 x_i\), \(\rho \triangleq \sum_{i \in \mathcal{P}_1} U_{it_2}^0 x_i\), \(\bar{Y}_q \triangleq \sum_{i \in \mathcal{X}_q} U_{it_2}^0\), and \(\bar{B} \triangleq \sum_{i \in \mathcal{P}_1} U_{it_2}^0\). Similarly, using (88), we have

\[
\hat{\mu}_{t_2} = \frac{\sum_{q \in \mathcal{V} : x_q \neq \phi} \tilde{\lambda}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 x_i + \bar{\rho}}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}},
\]

where \(\tilde{\lambda}_q \triangleq \frac{|\mathcal{X}_q|}{r} \sum_{i \in \mathcal{X}_q} U_{it_2}^0 x_i\), \(\bar{\rho} \triangleq \sum_{i \in \mathcal{P}_1} \frac{\big|\mathcal{P}_1\big|}{r} U_{ip}^0 x_i\), \(\bar{Y}_q \triangleq \sum_{i \in \mathcal{X}_q} U_{it_2}^0\), and \(\bar{B} \triangleq \sum_{i \in \mathcal{P}_1} \frac{\big|\mathcal{P}_1\big|}{r} U_{it_2}^0\). Notice that for each \(q \in \mathcal{V}\), the random variable \(\tilde{\lambda}_q\) can be written as a sum of random variables \(\tilde{\lambda}_{q, i_p} \triangleq |\mathcal{X}_q| U_{it_2}^0 x_{i_p}\), where \(i_p\) is sampled uniformly over \([n]\). Similarly, \(\bar{Y}_q\) written as a sum of i.i.d random variables \(\bar{Y}_{q, i_p} \triangleq |\mathcal{X}_q| U_{it_2}^0\), where again \(i_p\) is sampled uniformly over \([n]\). Finally, both \(\bar{\rho}\) and \(\bar{B}\) can also be written as a sum of i.i.d random variables \(\bar{\rho}_{i_p} \triangleq |\mathcal{P}_1| U_{ip}^0 x_{i_p}\) and \(\bar{B}_{i_p} \triangleq |\mathcal{P}_1| U_{ip}^0\), respectively, where \(i_p\) is sampled uniformly over \([n]\). Thus, it is evident that \(\mathbb{E} \tilde{\lambda}_{q, i_p} = \sum_{i \in \mathcal{X}_q} U_{it_2}^0 x_i\) and \(\mathbb{E} \bar{Y}_{q, i_p} = \sum_{i \in \mathcal{X}_q} U_{it_2}^0\), for all \(p \in [r]\). Similarly, \(\mathbb{E} \bar{\rho}_{i_p} = \sum_{i \in \mathcal{P}_1} U_{ip}^0 x_i\) and \(\mathbb{E} \bar{B}_{i_p} = \sum_{i \in \mathcal{P}_1} U_{ip}^0\), for all \(p \in [r]\). Next, we note that

\[
\hat{\mu}_{t_2} = \mu_{t_2} + \sum_{q \in \mathcal{V}} \frac{\tilde{\lambda}_q - \lambda_q}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}} + \sum_{q \in \mathcal{V} : x_q \neq \phi} \frac{\bar{Y}_q - \tilde{Y}_q}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}} + \hat{\mu}_{t_2} \cdot \sum_{q \in \mathcal{V} : x_q \neq \phi} \frac{\bar{Y}_q - \tilde{Y}_q}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}}.
\]

Therefore, using the triangle inequality and the fact that \(\|\hat{\mu}_{t_2}\|_2 \leq R\), we get

\[
\|\hat{\mu}_{t_2} - \mu_{t_2}\|_2 \leq \sum_{q \in \mathcal{V}} \frac{\|\tilde{\lambda}_q - \lambda_q\|_2}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}} + \sum_{q \in \mathcal{V} : x_q \neq \phi} \frac{\|\bar{Y}_q - \tilde{Y}_q\|_2}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}} + R \sum_{q \in \mathcal{V} : x_q \neq \phi} \frac{\|\bar{Y}_q - \tilde{Y}_q\|_2}{\sum_{q \in \mathcal{V} : x_q \neq \phi} \bar{Y}_q + \sum_{i \in \mathcal{P}_2} U_{it_2}^0 + \bar{B}}.
\]

Now, for any \(i \in \mathcal{X}_q\), we have by definition,

\[
U_{it_2}^0 \leq \left(2U_{\tilde{\rho}_{\bar{P}_{it_2}}}^{\|P_{Y_{it_2}}\|}\right)^\alpha,
\]

and for any \(i \in \mathcal{P}_1\),

\[
U_{it_2}^0 \leq \frac{\max_{j \in [n]} U_{jt_2}^0}{n^\alpha}.
\]
Next, using Lemmas 10 and 11, we have for all $q \in \mathcal{V}$, with probability at least $1 - \delta$,

\[
\|\tilde{\lambda} - \lambda_q\|_2 \leq R|\mathcal{X}_q|(2U_{\pi_{\tilde{\mu}_t}}(p_{\tilde{\mu}_q})t_2)^\alpha \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4},
\]

\[
|Y_q - Y_q| \leq |\mathcal{X}_q|(2U_{\pi_{\tilde{\mu}_t}}(p_{\tilde{\mu}_q})t_2)^\alpha \sqrt{\frac{c}{2r} \log \frac{t_2}{\eta \delta}},
\]

\[
\|\bar{\rho} - \rho\|_2 \leq \frac{R|\mathcal{P}_1| \max_j \{\bar{U}_{ij}\}}{n^3} \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4},
\]

\[
|\bar{B} - B| \leq \frac{|\mathcal{P}_1| \max_j \{\bar{U}_{ij}\}}{n^3} \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4},
\]

for some $c > 0$. Substituting the above results in (92), we get

\[
\|\tilde{\mu}_t - \mu_t\|_2 \leq 2R \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4} \cdot \sum_{q \in \mathcal{V} : \mathcal{X}_q \neq \emptyset} |\mathcal{X}_q|(2U_{\pi_{\tilde{\mu}_t}}(p_{\tilde{\mu}_q})t_2)^\alpha \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4} \cdot \sum_{q \in \mathcal{V} : \mathcal{X}_q \neq \emptyset} |\mathcal{X}_q|(2U_{\pi_{\tilde{\mu}_t}}(p_{\tilde{\mu}_q})t_2)^\alpha \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4} \cdot \sum_{i \in \mathcal{P}_2} \bar{U}_{ij} + B.
\]

Noting to the facts that

\[
\sum_{q \in \mathcal{V} : \mathcal{X}_q \neq \emptyset} Y_q = \sum_{i \in \mathcal{X}_q : \mathcal{X}_q \neq \emptyset} U_{ij} \geq |\mathcal{X}_q|(U_{\pi_{\tilde{\mu}_t}}(p_{\tilde{\mu}_q})t_2)^\alpha
\]

and

\[
\sum_{q \in \mathcal{V} : \mathcal{X}_q \neq \emptyset} Y_q + \sum_{i \in \mathcal{P}_2} \bar{U}_{ij} + B = \sum_{i \in \mathcal{P}_2} \sum_{j \in \mathcal{X}_q} U_{ij} \geq \max_{i \in \mathcal{P}_2} \sum_{j \in \mathcal{X}_q} U_{ij},
\]

and finally that $|\mathcal{P}_1| \leq n$, we obtain

\[
\|\tilde{\mu}_t - \mu_t\|_2 \leq 2^{\alpha + 1} R \left(\frac{c}{r} \log \frac{t_2}{\eta \delta}\right)^{1/4}.
\]

Therefore, for any $\epsilon > 0$, with $r \geq \frac{R^4}{c^4} \log \frac{2}{\eta \delta}$, we obtain $\|\tilde{\mu}_t - \mu_t\|_2 \leq \epsilon$, which proves the lemma.

\[\square\]

### 4.3.2 Proof of Theorem 3

First, Corollary 4 implies that a query complexity of $O \left(\frac{R^4}{c^4} \log \frac{1}{\delta}\right)$, and time-complexity of $O \left(\frac{R^4}{c^4} \log \frac{1}{\delta}\right)$, suffice to approximate the center of the first cluster $t_1$ with probability at least $1 - \delta/2$. Then, Lemma 8 allows us to estimate $U_{it}$ for all $i \in [n]$ using a query complexity of $O(\log^2 n)$ and time-complexity of $O(\log^2 n)$. Also, Lemma 9 shows that a query complexity of $O \left(\frac{R^4}{c^4} \log \frac{n}{\eta}\right)$, and time-complexity of $O \left(\frac{R^4}{c^4} \log \frac{n}{\eta}\right)$, suffice to approximate $\tilde{\mu}_t$ up to an error of $\epsilon$. Finally, we can use Lemma 4 to approximate $U_{ij}$ up to an error of $\eta$ using query complexity of $O(\log n/\eta)$, and a time-complexity of $O(n \log n + \log n/\eta)$, for all $i \in [n]$ and $j \in \{1, 2\}$. 

\[\square\]
(a) Comparison of two-phase algorithm and the sequential algorithm (see Algorithm 1 and 4. The error in recovery of means is plotted with varying $\beta$.

(b) The error in recovery of means using the two-phase algorithm and sequential algorithm (see Algorithm 1 and 4 with increasing queries keeping $\beta$ fixed.

(c) Comparison of the two-phase algorithm and the sequential algorithm (see Algorithm 1 and 4. The error in recovery of means is plotted with varying number of clusters ($k$).

Figure 1: Testing algorithms over synthetic datasets.

(a) Iris  
(b) Wine  
(c) Breast Cancer

Figure 2: Classification accuracy of algorithms for the Iris, Wine and Breast Cancer datasets.

5 Experiments

5.1 Synthetic Datasets

We conduct in-depth simulations of the proposed techniques over the following synthetic dataset. Specifically, we generate the dataset $\mathcal{X}$ by choosing $k = 4$ centers with dimension $d = 10$, such that $\mu_1$ is significantly separated from the other centers; the distance from each coordinate of $\mu_1$ to the coordinates of the other means is at least 1000. Subsequently, for each $i \in \{1, 2, 3, 4\}$ we randomly generate $L_i$ vectors from a spherical Gaussian distribution, with mean $\mu_i$, and a standard deviation of 20 per coordinate. We then run the Fuzzy C-means algorithm\(^2\), and obtain a target solution $P$ to be used by the oracle for responses. In order to understand the effect of $\beta$, we fix $L_1 = 5000$, and vary $L_2, L_3, L_4 \in 5000 \cdot \zeta$, where $\zeta \in \{1, 2, \ldots, 24\}$. It can be checked that $\beta = 4/(1 + 3\zeta)$. We run Algorithms 1 and 4. For the two-phase algorithm we take $\alpha = 2$, $m = \nu$, and $\eta_1 = 0.1$, and $\alpha = 2$, $m = \nu/2.5$, $\eta_1 = 0.1$, and $\eta_2 = 0.1$, for the sequential algorithm, where $\nu \in \{2000, 6000\}$. Setting the parameters in this way keeps the same query complexity for both algorithms, so as to keep a fair comparison. We run each algorithm 20 times. For each algorithm, we evaluate the maximal error in

\(^2\)https://github.com/omadson/fuzzy-c-means
estimating the centers. The results are shown in Fig. 1. Specifically, Fig. 1a presents the estimation error as a function of $\beta$. It can be seen that for small $\beta$'s, the sequential algorithm is significantly better compared to the two-phase algorithm, whereas for larger $\beta$'s, they are comparable. Then, for $\beta = 0.25, 0.1$, Fig. 1b shows the estimation error as a function of the number of queries. Finally, to understand the effect of the number of clusters, we generate $k$ clusters using a similar method as above. We take $L_1 = 1000$, and $L_i = 12000$, for all $2 \leq i \leq k$. We vary $k \in \{2,3,\ldots,11\}$. For the two-phase algorithm, we take $\alpha = 2$, $m = \nu$, and $\eta = 0.1$, and $\alpha = 2$, $m = \nu/2.5$, $\eta_1 = 0.1$, and $\eta_2 = 0.1$, for the sequential algorithm, where $\nu = \{2000,6000\}$. Fig. 1c shows the estimation error as a function of $k$. We can clearly observe that the two-phase algorithm performs significantly better as $k$ increases but the sequential algorithm works better for small $k$.

\section{5.2 Real-World Datasets}

In our experiments, we use three well-known real-world datasets available in scikit-learn [41]: the Iris dataset (150 elements, 4 features, and 3 classes), the Wine dataset (178 elements, 13 features, and 3 classes), and the Breast Cancer dataset (569 elements, 30 features, and 2 classes). For the Iris and Wine datasets, we run the two-phase and sequential algorithms. We take $\alpha = 2$, $m = \nu$, and $\eta = 0.1$, for the two-phase algorithm, and $\alpha = 2$, $m = 2\nu/3$, $r = m/\eta_1$, $\eta_1 = 0.1$, and $\eta_2 = 0.1$, for the sequential algorithm, where $\nu \in \{10,20,\ldots,410\}$, keeping the same query complexity for both algorithms. These values do not necessarily satisfy what is needed by our theoretical results. We run both algorithms with each set of parameters 500 times to account for the randomness. In our experiments, we use a hard cluster assignment as ground truth (or rather the target clustering $P$ to be used by the oracle for responses), and use our algorithms to return a fuzzy assignment. We must point out over here that our fuzzy algorithms can be used to solve hard clustering problems as well and therefore, it is not unreasonable to have hard clusters as the target solution.

Subsequently, we estimate the membership weights for all elements, and for each element, we predict the class the element belongs to as the one to which the element has the highest membership weight (i.e., $\arg\max_j \hat{U}_{ij}$, for element $i$)\footnote{This is similar to rounding in Linear Programming}. Once we have classified all the data-points using our algorithms, we can check the classification accuracy since we possess the ground-truth labels. Note that the ground truth labels can be inconsistent with the best clustering or $P^*$, the solution that minimizes the objective in the Fuzzy $k$-means problems (Definition 1) but we assume that the “ground truth” labels that are given by humans are a good proxy for the best clustering.

We then plot the classification accuracy as a function of the number of queries. Fig. 2 shows the average classification accuracy for the above three data-sets by comparing the predicted classes and the ground truth. For the Breast Cancer dataset, since the number of clusters is two, we additionally compare the two-phase and sequential algorithms to Algorithm 5 with $\alpha = 2$, $m = 2\nu/3$, $r = m/\eta_1$, and $\eta = 0.1$. It turns out that for these real-world datasets, the performance of all algorithms are comparable. It can be seen that the accuracy increases as a function of the number of queries, as expected. Further, by using the well-known Lloyd’s style iterative Fuzzy C-means algorithm with random initialization [21], we get an average classification accuracy (over 20 trials) of only 31.33\%, 35.96\% and 14.58\% on the Iris, Wine, and Breast Cancer datasets, respectively. This experiment shows that using a few membership queries increases the accuracy of a poly-time algorithm drastically, corroborating the results of our paper.
5.3 Accuracy as a Function of $\alpha$

As discussed in right after Theorem 2, the "fuzzifier" $\alpha$ is not subject to an optimization. Nonetheless, if we assume the existence of a ground truth, we can compare the clustering accuracy for different values of $\alpha$. Accordingly, in Fig. 3 we test the performance of our algorithms on the Iris dataset for a few values of $\alpha$. We calculate the average accuracy over 500 trials for each set of parameters. We conclude this section by discussing the issue of comparing our semi-supervised fuzzy approach to the semi-supervised hard objective [4]. Generally speaking, in the absence of a ground truth, comparing both approaches is meaningless. When the ground truth represents a disjoint clustering, then it is reasonable that following a hard approach (essentially $\alpha = 1$) will capture this ground truth better. However, the whole point of using fuzzy clustering in the first place is when the clusters, in some sense, overlap. Indeed, the initial main motivation for studying fuzzy clustering is that it is applicable to datasets where datapoints show affinity to multiple labels, the clustering criteria are vague and data features are unavailable. Nonetheless, in Fig. 3 we compare the performance of both the fuzzy and hard approaches (essentially $\alpha = 1$) on the Iris dataset.

6 Conclusion and Outlook

In this paper, we studied the fuzzy $k$-means problem, and proposed a semi-supervised active clustering framework, where the learner is allowed to interact with a membership-oracle, asking for the memberships of a certain set of chosen items. We studied both the query and computational complexities of clustering in this framework. In particular, we provided two probabilistic algorithms (two-phase and sequential) for fuzzy clustering that ask $O(\text{poly}(k) \log n)$ membership queries and run with polynomial-time-complexity. The main difference between these two algorithms is the dependency of their query complexities on the size of the smallest cluster $\beta$. The sequential algorithm exhibits more graceful dependency on $\beta$. Finally, for $k = 2$ we were able to remove completely the dependency on $\beta$ (see, Appendix 4.3). We hope our work has opened more doors than it closes. Apart from tightening the obtained query...
complexities, there are several exciting directions for future work:

- It is important to understand completely the dependency of the query complexity on $\beta$. Indeed, we showed that for $k = 2$ there exists an algorithm whose query complexity is independent of $\beta$, but what happens for $k > 2$?

- It would be interesting to understand to what extent the algorithms and analysis in this paper, can be applied to other clustering problems which depend on different metrics other than the Euclidean one.

- Our paper presents upper bounds (sufficient conditions) on the query complexity. It is interesting and challenging to derive algorithm-independent information-theoretic lower bounds on the query complexity.

- As mentioned in the introduction it is not known yet whether the fuzzy $k$-means problem lies in NP like the hard $k$-means problem. Answering this question will give a solid motivation to the semi-supervised setting considered in this paper. Furthermore, just as the information-theoretic lower bounds, it would be interesting to derive computational lower bounds as well.

- In this paper we focused on the simplest form of oracle responses. However, there are many other interesting and important settings, e.g., the noisy setting (Appendix C). Another interesting problem would be to consider adversarial oracles who intentionally provide corrupted responses.

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 non-negative 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. Blasiok, 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.
Appendices

A  Proof of Lemma 2

To prove Lemma 2 we start by proving a few inequalities. Since $A$ is an $(\epsilon_1, \epsilon_2, Q)$-solver, using Definition 4 and Taylor's expansion, we get for any $i \in [n]$ and $j \in [k]$,

$$
\hat{U}_{ij}^\alpha \leq (U_{ij} + \epsilon_2)^\alpha \tag{103}
$$

$$
= U_{ij}^\alpha + \alpha U_{ij}^{\alpha-1} \epsilon_2 + o(\epsilon_2^2) \tag{104}
$$

$$
\leq U_{ij}^\alpha + \alpha \epsilon_2 + o(\epsilon_2^2), \tag{105}
$$

where the last inequality follows from the fact that $U_{ij} \leq 1$. Also, for any $i \in [n]$ and $j \in [k]$, we have

$$
\|x_i - \hat{\mu}_j\|^2 = \|x_i - \mu_j\|^2 + 2 \cdot (x_i - \mu_j)^T (\mu_j - \hat{\mu}_j) + \|\mu_j - \hat{\mu}_j\|^2 \tag{106}
$$

$$
\leq \|x_i - \mu_j\|^2 + 4 \cdot R \cdot \epsilon_1 + \epsilon_1^2, \tag{107}
$$

where the inequality follows from the fact that $A$ is an $(\epsilon_1, \epsilon_2, Q)$-solver, the Cauchy-Schwarz inequality, and the fact that since $x_i \in B(0, R)$ so as $\mu_j \in B(0, R)$, and thus $\|x_i - \mu_j\|^2 \leq 4 \cdot R^2$. Finally, we have for any $i \neq j \in [k]$,

$$
\|\hat{\mu}_i - \hat{\mu}_j\|^2 \geq \|\mu_i - \mu_j\|^2 + \|\hat{\mu}_i - \mu_j\|^2 + \|\hat{\mu}_j - \mu_j\|^2 + 2(\mu_i - \mu_j)^T (\hat{\mu}_i - \mu_i) \tag{108}
$$

$$
+ 2(\mu_i - \mu_j)^T (\hat{\mu}_j - \mu_j) + 2(\hat{\mu}_i - \mu_i)^T (\hat{\mu}_j - \mu_j) \tag{109}
$$

$$
\geq \|\mu_i - \mu_j\|^2 - 2\epsilon_1^2 - 8R\epsilon_1 - 2\epsilon_1^2 \tag{110}
$$

Now, with the above results, we note that

$$
J_{fm}(\mathcal{X}, \hat{\mathcal{P}}) = \sum_{i=1}^n \sum_{j=1}^k \hat{U}_{ij}^\alpha \|x_i - \hat{\mu}_j\|^2 \tag{111}
$$

$$
\leq \sum_{i=1}^n \sum_{j=1}^k U_{ij}^\alpha \|x_i - \hat{\mu}_j\|^2 + \alpha \epsilon_2 \sum_{i=1}^n \sum_{j=1}^k \|x_i - \mu_j\|^2 + o(\epsilon_2^2). \tag{112}
$$

Now since $\mu_j \in B(0, R)$, we can say that $\hat{\mu}_j \in B(0, R + \epsilon_1)$. Therefore, $\|x_i - \hat{\mu}_j\|^2 \leq 2[R^2 + (R + \epsilon_1)^2]$. Hence,

$$
J_{fm}(\mathcal{X}, \hat{\mathcal{P}}) \leq \sum_{i=1}^n \sum_{j=1}^k U_{ij}^\alpha \|x_i - \hat{\mu}_j\|^2 + 2nk\alpha \epsilon_2 [R^2 + (R + \epsilon_1)^2] + o(\epsilon_2^2). \tag{113}
$$

Next, using (107), we get

$$
\sum_{i=1}^n \sum_{j=1}^k U_{ij}^\alpha \|x_i - \hat{\mu}_j\|^2 \leq \sum_{i=1}^n \sum_{j=1}^k U_{ij}^\alpha \|x_i - \mu_j\|^2 + [4 \cdot R \cdot \epsilon_1 + \epsilon_1^2] \sum_{i=1}^n \sum_{j=1}^k U_{ij}^\alpha \tag{114}
$$

$$
\leq \sum_{i=1}^n \sum_{j=1}^k U_{ij}^\alpha \|x_i - \mu_j\|^2 + n [4 \cdot R \cdot \epsilon_1 + \epsilon_1^2] \tag{115}
$$

$$
= J_{fm}(\mathcal{X}, \mathcal{P}) + n [4 \cdot R \cdot \epsilon_1 + \epsilon_1^2], \tag{116}
$$

36
where the second inequality follows from the fact that $U_{ij} \in [0,1]$, and thus $\sum_{i=1}^{n} \sum_{j=1}^{k} U_{ij}^2 \leq \sum_{i=1}^{n} \sum_{j=1}^{k} U_{ij} = n$. Therefore, we obtain

$$J_{fm}(\mathcal{X}, \mathcal{P}) \leq J_{fm}(\mathcal{X}, \mathcal{P}) + n \left[4 \cdot R \cdot \epsilon_1 + \epsilon_1^2\right] + 2nk\alpha\epsilon_2[R^2 + (R + \epsilon_1)^2] + o(\epsilon_2^2)$$

$$\leq J_{fm}(\mathcal{X}, \mathcal{P}) + n \cdot O(\epsilon_1) + nk \cdot O(\epsilon_2) + n \cdot o(\epsilon_1^2) + nk \cdot o(\epsilon_2^2).$$  \hspace{1cm} (117) \hspace{1cm} (118)

We are now in a position to bound $\text{XB}(\mathcal{X}, \mathcal{P})$. Using (110) and (118), we have

$$\text{XB}(\mathcal{X}, \mathcal{P}) = \frac{J_{fm}(\mathcal{X}, \mathcal{P})}{nk \cdot \min_{i \neq j} \|\mu_i - \mu_j\|^2_2}$$

$$\leq \frac{J_{fm}(\mathcal{X}, \mathcal{P}) + n \cdot O(\epsilon_1) + nk \cdot O(\epsilon_2) + n \cdot o(\epsilon_1^2) + nk \cdot o(\epsilon_2^2)}{nk \cdot \left[\min_{i \neq j} \|\mu_i - \mu_j\|^2_2 - 8R\epsilon_1 - 4\epsilon_1^2\right]}$$

$$= \frac{J_{fm}(\mathcal{X}, \mathcal{P})}{nk \cdot \min_{i \neq j} \|\mu_i - \mu_j\|^2_2} + \frac{n \cdot O(\epsilon_1) + nk \cdot O(\epsilon_2) + n \cdot o(\epsilon_1^2) + nk \cdot o(\epsilon_2^2)}{nk \cdot \min_{i \neq j} \|\mu_i - \mu_j\|^2_2}$$

$$= \text{XB}(\mathcal{X}, \mathcal{P}) + \frac{O(\epsilon_1)}{\min_{i \neq j} \|\mu_i - \mu_j\|^2_2} + \frac{O(\epsilon_2)}{\min_{i \neq j} \|\mu_i - \mu_j\|^2_2}$$

$$+ o\left(\frac{\epsilon_1^2 + \epsilon_2^2}{\min_{i \neq j} \|\mu_i - \mu_j\|^2_2}\right).$$ \hspace{1cm} (122)

Using the same steps a similar lower bound can be obtained, which concludes the proof.

**B Auxiliary Lemmata**

In this section we present and prove a few auxiliary results which will be used in the proofs of our main results. We start with the following standard concentration inequalities.

**Lemma 10** (Hoeffding’s inequality). Let $X_1, X_2, \ldots, X_n$ be i.i.d random variables, such that $|X_i| \leq R$ a.s., and $\mathbb{E}X_i = \mu$, for all $i \in [n]$. Then, with probability at least $1 - \delta$,

$$\left|\frac{1}{n} \sum_{i=1}^{n} X_i - \mu\right| \leq R\epsilon,$$  \hspace{1cm} (123)

if $n \geq \frac{c\log(1/\delta)}{2\epsilon^2}$, where $c > 0$ is some absolute constant.

**Lemma 11** (Generalized Hoeffding’s inequality). Let $X_1, X_2, \ldots, X_n$ be i.i.d random vectors, such that $\|X_i\|_2 \leq R$ a.s., and $\mathbb{E}X_i = \mu_*$, for all $i \in [n]$. Then, with probability at least $1 - \delta$,

$$\left\|\frac{1}{n} \sum_{i=1}^{n} X_i - \mu_*\right\|_2^2 \leq R^2\epsilon,$$  \hspace{1cm} (124)

if $n \geq \frac{c\log(1/\delta)}{\epsilon^2}$, where $c > 0$ is some absolute constant.

The following locality lemma states that the fuzzy $k$-means function is strictly increasing.
Lemma 12. Let $(\mathcal{X}, \mathcal{P}^*)$ be a clustering instance, where $\mathcal{P}^*$ refers to the optimal solution for the fuzzy $k$-means problem (namely, minimizes the objective in (2)). Then, for any $i, j \in [n]$ and $\ell \in [k]$ with $\|x_i - \mu_\ell^i\|^2_2 \leq \|x_j - \mu_\ell^j\|^2_2$, we have $U_{i\ell}^\alpha \geq U_{j\ell}^\alpha$.

Proof of Lemma 12. Consider some $i, j \in [n]$ and $\ell \in [k]$ with $\|x_i - \mu_\ell^i\|^2_2 \leq \|x_j - \mu_\ell^j\|^2_2$. By definition $\{U_{i\ell}^\alpha\}_{i=1}^n$ minimizes the cost $\sum_{i=1}^n U_{i\ell}^\alpha \|x_i - \mu_\ell\|^2_2$. This implies that,

$$U_{i\ell}^\alpha \|x_i - \mu_\ell\|^2_2 + U_{j\ell}^\alpha \|x_j - \mu_\ell\|^2_2 \leq U_{j\ell}^\alpha \|x_i - \mu_\ell\|^2_2 + U_{i\ell}^\alpha \|x_j - \mu_\ell\|^2_2,$$

which is equivalent to,

$$U_{i\ell}^\alpha \|x_i - \mu_\ell\|^2_2 + U_{j\ell}^\alpha \|x_j - \mu_\ell\|^2_2 \leq U_{j\ell}^\alpha \|x_i - \mu_\ell\|^2_2 + U_{i\ell}^\alpha \|x_j - \mu_\ell\|^2_2.$$

Since $\|x_i - \mu_\ell\|^2_2 - \|x_j - \mu_\ell\|^2_2 \leq 0$, we get $U_{i\ell}^\alpha \geq U_{j\ell}^\alpha$, which concludes the proof.

C Noisy Oracle Responses

In this section, we discuss briefly the effect of a noisy membership-oracle, defined as follows.

Definition 5 (Noisy Membership-Oracle). A fuzzy query asks the membership weight of an instance $x_i$ to a cluster $j$ and obtains in response a noisy answer $O_{\text{noisy}}(i, j) = U_{ij} + \xi_{ij}$ where $\xi_{ij}$ is a zero mean random variable with variance $\sigma^2$.

To present our main result, let $\rho \in \mathbb{R}_+$ be defined as

$$\min_{j \in [k]} \sum_{i \in [n]} U_{ij}^\alpha = \rho n.$$  

(127)

The result below handles the situation where the oracle responses are noisy.

Theorem 9. Let $\kappa > 0$, and assume that there exists an $(\epsilon_1, \epsilon_2, Q)$-solver for a clustering instance $(\mathcal{X}, \mathcal{P})$ using the membership-oracle responses $O_{\text{fuzzy}}$. Then, there exist a

$$\left(\frac{2\rho \alpha (\epsilon_2 + \kappa)}{\rho}, \epsilon_2 + \kappa, \frac{8Q\sigma^2 \log n}{\kappa^2}\right) - \text{solver},$$

for $(\mathcal{X}, \mathcal{P})$ using the noisy oracle $O_{\text{noisy}}$.

Proof. Assume that algorithm $A_{\text{noiseless}}$ is an $(\epsilon_1, \epsilon_2, Q)$-solver for a clustering instance $(\mathcal{X}, \mathcal{P})$ using queries to a noiseless oracle $O_{\text{fuzzy}}$. In order to handle noisy responses we propose the following algorithm $A_{\text{noisy}}$: apply algorithm $A_{\text{noiseless}}$ for $T$ steps using noisy queries to $O_{\text{noisy}}$. We will show that this algorithm obtains the guarantees in Theorem. To that end, in each such step, we obtain noisy estimates for the memberships and the centers. Then, we use these local estimates to obtain a clean final estimate for the memberships and the centers. Specifically, consider $T$ independent and noisy clustering instances $\{\mathcal{P}^t \equiv (\mu^t, \mathcal{V}^t)\}_{t=1}^T$, such that

$$V_{ij}^t = U_{ij} + \xi_{ij} \quad \text{and} \quad \mu_j^t = \frac{\sum_{i=1}^n V_{ij}^t x_i}{\sum_{i=1}^n V_{ij}^t},$$

(128)

for $t \in [T]$. Note that the randomness in the definition of the aforementioned clustering instances lies in the realization of the independent random variables $\xi_{ij}^1, \xi_{ij}^2, \ldots, \xi_{ij}^T$. For each
such instance we apply one of the algorithms we developed for the noiseless oracle. Accordingly, for all $t \in [T]$, suppose we have a $(\epsilon_1, \epsilon_2, Q)$-solver that makes $Q$ queries to the $P_t$-oracle to compute $\hat{V}_{ij}^t$. Then, we know that,

$$ \left| \hat{V}_{ij}^t - V_{ij}^t \right| \leq \epsilon_2, \quad (129) $$

for all $t < [T]$. Now, all we have to do is to use these local estimates to calculate our final estimates for the underlying memberships and centers. Specifically, for $T' < T$, we must have

$$ \left| \frac{1}{T'} \sum_{j=1}^{T'} \hat{V}_{ij}^t - U_{ij} \right| \leq \epsilon_2 + \left| \frac{1}{T'} \sum_{t=1}^{T'} \zeta_{ij}^t \right|. \quad (130) $$

By Chebychev’s inequality, for any $\kappa > 0$, we get

$$ \Pr\left[ \left| \frac{1}{T'} \sum_{t=1}^{T'} \zeta_{ij}^t \right| \geq \kappa \right] \leq \frac{\sigma^2}{\kappa^2 T'}. \quad (131) $$

Next, we partition the $T$ responses from the oracle into $B$ batches of size $T'$ each. For batch $b \in [B]$, define the random variable $Y^b \triangleq 1 \left[ \left| \frac{1}{T'} \sum_{t \in \text{Batch } b} \zeta_{ij}^t \right| \geq \kappa \right]$. Clearly, $\Pr(Y^b = 1) \leq \frac{\sigma^2}{\kappa^2 T'}$ and further, $Y^1, Y^2, \ldots, Y^B$ are independent random variables. Therefore, Chernoff bound implies that

$$ \Pr\left( \sum_{b=1}^{B} Y^b \geq B/2 \right) \leq \exp \left[ -2B \left( \frac{1}{2} - \frac{\sigma^2}{\kappa^2 T'} \right)^2 \right]. \quad (132) $$

Our final membership estimate is evaluated as follows:

$$ \hat{U}_{ij} \triangleq \text{median} \left( \frac{1}{T'} \sum_{t \in \text{Batch } 1} V_{ij}^t, \frac{1}{T'} \sum_{t \in \text{Batch } 2} V_{ij}^t, \ldots, \frac{1}{T'} \sum_{t \in \text{Batch } B} V_{ij}^t \right), \quad (133) $$

namely, $\hat{U}_{ij}$ is the median of the mean of $\hat{V}_{ij}^t$ in each batch. Therefore, for $B = 6 \log n$ and $T' = 4\sigma^2/\kappa^2$ (hence $T = 8\sigma^2 \log n/\kappa^2$), we must have that

$$ \Pr\left( \left| \hat{U}_{ij} - U_{ij} \right| \geq \epsilon_2 + \kappa \right) \leq \frac{2}{n^3}. \quad (134) $$

Therefore, by taking a union bound over all $i \in [n], j \in [k]$, we can compute $\hat{U}_{ij}$, an estimate of $U_{ij}$, such that

$$ \left| \hat{U}_{ij} - U_{ij} \right| \leq \epsilon_2 + \kappa, \quad (135) $$

for all $i \in [n], j \in [k]$ with probability at least $1 - 1/n$. Finally, we estimate the means $\mu_j$’s using the already computed $\hat{U}_{ij}$’s as follows. Note that,

$$ \hat{x} = \frac{\sum_i \hat{U}_{ij}^\alpha x_i}{\sum_i \hat{U}_{ij}^\alpha} \triangleq \frac{\hat{x}_\alpha}{\hat{Y}}. \quad (136) $$
\[ \mu_j = \frac{\sum_{i \in [n]} U_{ij} x_i}{\sum_{i \in [n]} U_{ij}} = \frac{\lambda_x Y}{Y}. \]  

(137)

Therefore, we get

\[ \|\hat{\mu}_j - \mu_j\|_2 \leq \|\hat{\lambda}_x - \lambda_x\|_2 + \|\frac{\hat{\lambda}_x Y - \hat{Y}}{Y}\|_2 \leq \|\frac{\hat{\lambda}_x - \lambda_x}{Y}\|_2 + R \left|\frac{\hat{Y} - Y}{Y}\right|. \]  

(138)

Using (135) it is evident that

\[ |\hat{Y} - Y| \leq \alpha n(\epsilon_2 + \kappa)(1 + o(1)), \]  

(139)

\[ \|\hat{\lambda}_x - \lambda_x\|_2 \leq R\alpha n(\epsilon_2 + \kappa)(1 + o(1)). \]  

(140)

Combining (138)–(140) together with (127), we finally obtain that

\[ \|\hat{\mu}_j - \mu_j\|_2 \leq \frac{2R\alpha(\epsilon_2 + \kappa)}{\rho}, \]  

(141)

for all \( j \in [k] \), which concludes the proof. \( \square \)

D Membership Queries From Similarity Queries

Recall that \( \mathcal{X} \subset \mathbb{R}^d, |\mathcal{X}| = n \) is the set of points provided as input along with their corresponding \( d \)-dimensional vector assignments denoted by \( \{x_i\}_{i=1}^n \). Recall that the membership-oracle \( O_{\text{fuzzy}}(i, j) = U_{ij} \) returns the membership weight of the instance \( x_i \) to a cluster \( j \). However, such oracle queries are often impractical in real-world settings since it requires knowledge of the relevant clusters. Instead, a popular query model that takes a few elements (two or three) as input and is easy to implement in practice is the following similarity query "How similar are these elements?" \cite{4} showed that for the hard clustering setting, a membership query can be simulated by \( k \) pairwise similarity queries since a pairwise similarity query reveals whether two items belong to the same cluster or not in the hard clustering setting. In the fuzzy problem we model the oracle response to the similarity query by the inner product of their membership weight vectors. More formally, we have

**Definition 6** (Restatement of Definition 3). A fuzzy pairwise similarity query asks the similarity of two distinct instances \( x_i \) and \( x_j \) i.e., \( O_{\text{sim}}(i, j) = \langle U_i, U_j \rangle \). A fuzzy triplet similarity query asks the similarity of three distinct instances \( x_p, x_q, x_r \) i.e. \( O_{\text{triplet}}(p, q, r) = \sum_{t \in [k]} U_{pt} U_{qt} U_{rt} \).

Now, we show that fuzzy pairwise similarity queries can often be used to simulate \( O_{\text{fuzzy}}(i, j) \). Note that if we possess the membership weight vectors of \( k \) elements that are linearly independent, then, for a new element, responses to fuzzy pairwise similarity queries with the aforementioned \( k \) elements reveals all the membership weights of the new element. Now, the question becomes "How can we obtain the membership weights of the \( k \) elements in the first place?". Suppose we sub-sample a set of elements \( \mathcal{Y} \subseteq \mathcal{X} \) such that \( |\mathcal{Y}| = m > k \) and we make all fuzzy pairwise similarity queries among the elements present in \( \mathcal{Y} \). Let us denote by \( V \) the membership weight matrix \( U \) constrained to the rows corresponding to the elements in
Clearly, the fuzzy pairwise similarity queries between all pairs of elements in \( Y \) reveals \( \mathbf{V}\mathbf{V}^T \), the gram matrix of \( \mathbf{V} \). If we can recover \( \mathbf{V} \) uniquely from \( \mathbf{V}\mathbf{V}^T \), and \( \mathbf{V} \) is full rank, then we are done. If we assume almost any continuous distribution according to which the membership weight vectors are generated, then with probability 1, the matrix \( \mathbf{V} \) is full rank. On the other hand, the question of uniquely recovering \( \mathbf{V} \) from \( \mathbf{V}\mathbf{V}^T \) is trickier. In general it is not possible to recover \( \mathbf{V} \) uniquely from \( \mathbf{V}\mathbf{V}^T \) since \( \mathbf{VR} \), for any orthonormal matrix \( \mathbf{R} \), also has the gram matrix \( \mathbf{V}\mathbf{V}^T \). However, recall that in our case, the entries of \( \mathbf{V} \) are non-negative and furthermore, the rows of \( \mathbf{V} \) add up-to 1 leading to additional constraints. This leads to the problem

\[
\text{Find } \mathbf{M} \text{ such that } \mathbf{MM}^T = \mathbf{V}\mathbf{V}^T \text{ subject to } \mathbf{M} \in \mathbb{R}^{m \times k}, \sum_{j \in [k]} M_{ij} = 1 \forall i \in [m].
\]

As a matter of fact, this is a relatively well-studied problem known as the Symmetric Non-Negative matrix factorization or SNMF. We will say that the solution to the SNMF problem is unique if \( \mathbf{VP} \) is the only solution to the problem for any permutation matrix \( \mathbf{P} \).

Below, we state the following sufficient condition that guarantees the uniqueness of the solution to the SNMF problem.

**Lemma 13** (Lemma 4 in [26]). If \( \text{rank}(\mathbf{V}) = k \), then the solution to the SNMF problem is unique if and only if the non-negative orthant is the only self-dual simplicial cone \( \mathcal{A} \) with \( k \) extreme rays that satisfies \( \text{cone}(\mathbf{V}^T) \subseteq \mathcal{A} = \mathcal{A}^\star \), where \( \mathcal{A}^\star \) is the dual cone of \( \mathcal{A} \), defined as \( \mathcal{A}^\star = \{ \mathbf{y} | \mathbf{x}^T \mathbf{y} \geq 0 \forall \mathbf{x} \in \mathcal{A} \} \).

More recently, Lemma 13 was used in [34] to show the following result that is directly applicable to our setting:

**Lemma 14.** If \( \mathbf{V} \) contains any permutation matrix of dimensions \( k \times k \), then the solution of the SNMF problem is unique.

Suppose we have the guarantee that for each cluster \( j \in [k] \), there exists a set \( \mathcal{Z}_j \) of at least \( \rho n \) elements belonging purely to the \( j \)-th cluster i.e. \( U_{ij} = 1 \) for all \( i \in \mathcal{Z}_j \). Then, for \( m \geq \rho^{-1} \log(nk) \), the matrix \( \mathbf{V} \) will contain a permutation matrix with probability at least \( 1 - n^{-1} \). As a result, this will lead to an overhead of \( O(m^2) = O(\rho^{-2} \log^2(nk)) \) queries.

If it is possible to make more complex similarity queries such as the fuzzy triplet similarity query, we can significantly generalize and improve the previous guarantees. Before proceeding further, let us provide some background on tensors beginning with the following definition:

**Definition 7** (Kruskal rank). The Kruskal rank of a matrix \( \mathbf{A} \) is defined as the maximum number \( r \) such that any \( r \) columns of \( \mathbf{A} \) are linearly independent.

Consider a tensor \( \mathcal{A} \) of order \( w \in \mathbb{N} \) for \( w > 2 \) on \( \mathbb{R}^n \), denoted by \( \mathcal{A} \in \mathbb{R}^n \otimes \mathbb{R}^n \otimes \cdots \otimes \mathbb{R}^n \) (\( w \) times). Let \( \mathcal{A}_{i_1,i_2,\ldots,i_w} \) where \( i_1, i_2, \ldots, i_w \in \{0, 1, \ldots, n - 1\} \), denote the element in \( \mathcal{A} \) whose location along the \( j \)-th dimension is \( i_j + 1 \), i.e., there are \( i_j \) elements along the \( j \)-th dimension before \( \mathcal{A}_{i_1,i_2,\ldots,i_w} \). Notice that this indexing protocol uniquely determines the element within the tensor. For a detailed review of tensors, we defer the reader to [29]. In this work, we are interested in low-rank decomposition of tensors. A tensor \( \mathcal{A} \) can be described as
a rank-1 tensor if it can be expressed as

\[
A = z \otimes z \otimes \cdots \otimes z
\]

for some \( z \in \mathbb{R}^n \), i.e., \( A_{i_1, i_2, \ldots, i_w} = \prod_{j=1}^w z_{i_j} \). For a given tensor \( A \), we are concerned with the problem of uniquely decomposing \( A \) into a sum of \( R \) rank-1 tensors. A tensor \( A \) that can be expressed in this form is denoted as a rank-\( R \) tensor, and such a decomposition is also known as the Canonical Polyadic (CP) decomposition. Below, we state a result due to [45] describing the sufficient conditions for the unique CP decomposition of a rank-\( R \) tensor \( A \).

**Lemma 15** (Unique CP decomposition [45]). Suppose \( A \) is the sum of \( R \) rank-1 tensors, i.e.,

\[
A = \sum_{r=1}^R z^r \otimes z^r \otimes \cdots \otimes z^r,
\]

and further, the Kruskal rank of the \( n \times R \) matrix whose columns are formed by \( z^1, z^2, \ldots, z^R \) is \( J \). Then, if \( wJ \geq 2R + (w - 1) \), then the CP decomposition is unique and we can recover the vectors \( z^1, z^2, \ldots, z^R \) up to permutations.

**Algorithm 9** Jennrich’s Algorithm(\( A \))

**Input:** A symmetric rank-\( R \) tensor \( A \in \mathbb{R}^n \otimes \mathbb{R}^n \otimes \mathbb{R}^n \) of order 3.

1. Choose \( a, b \in \mathbb{R}^n \) uniformly at random such that it satisfies \( \|a\|_2 = \|b\|_2 = 1 \).
2. Compute \( T^{(1)} = \sum_{i \in [n]} a_i A_{:, :, i}, \ T^{(2)} = \sum_{i \in [n]} b_i A_{:, :, i} \).
3. if \( \text{rank}(T^{(1)}) < R \) then
   4. Return Error
5. end if
6. Solve the general eigenvalue problem \( T^{(1)} v = \lambda v, T^{(2)} v = \lambda v \).
7. Return the eigen-vectors \( v \) corresponding to the non-zero eigen-values.

Notice that for the special case of \( w = 3 \), the underlying vectors \( z^1, z^2, \ldots, z^R \) can be recovered uniquely if they are linearly independent. Now, we are ready to show that \( k \) fuzzy triplet similarity queries can be used to recover the memberships weights of \( k \) elements uniquely. As before, we can sub-sample a set of elements \( Y \subseteq X \) such that \( |Y| = k \) and we make all possible \( \binom{k}{3} \) fuzzy triplet similarity queries among the elements present in \( Y \). Again, let us denote by \( V \) the membership weight matrix \( \bar{U} \) constrained to the rows corresponding to the elements in \( Y \). Let us denote by \( v^1, v^2, \ldots, v^k \) the \( k \) columns of the matrix \( V \). Notice that the responses to all the fuzzy triplet similarity queries reveals the following symmetric tensor

\[
\sum_{r=1}^k v^r \otimes v^r \otimes v^r.
\]

Suppose the matrix \( V \) is full rank. This will happen with probability 1 if the membership weights are assumed to be generated according to any continuous distributions. Algorithmically, Jennrich’s algorithm (see Section 3.3, [39]) can be used to efficiently recover the unique CP decomposition of a third order low rank tensor whose underlying vectors are full rank. We have provided the algorithm (see Algorithm 9) for the sake of completeness.

---

4In this work, we focus on the special case of rank-1 tensors where every component is identical. In general, rank-1 tensors can be described as \( A = z^1 \otimes z^2 \otimes \cdots \otimes z^w \) for \( z^1, \ldots, z^w \in \mathbb{R}^n \).