

Data-Driven Clustering via Parameterized Lloyd’s Families*

Maria-Florina Balcan

Travis Dick

Colin White

Abstract

Algorithms for clustering points in metric spaces is a long-studied area of research. Clustering has seen a multitude of work both theoretically, in understanding the approximation guarantees possible for many objective functions such as k -median and k -means clustering, and experimentally, in finding the fastest algorithms and seeding procedures for Lloyd’s algorithm. The performance of a given clustering algorithm depends on the specific application at hand, and this may not be known up front. For example, a “typical instance” may vary depending on the application, and different clustering heuristics perform differently depending on the instance.

In this paper, we define an infinite family of algorithms generalizing Lloyd’s algorithm, with one parameter controlling the initialization procedure, and another parameter controlling the local search procedure. This family of algorithms includes the celebrated k -means++ algorithm, as well as the classic farthest-first traversal algorithm. We design efficient learning algorithms which receive samples from an application-specific distribution over clustering instances and learn a near-optimal clustering algorithm from the class with respect to that distribution with provable sample complexity guarantees. We show the best parameters vary significantly across application domains such as MNIST, CIFAR, and mixtures of Gaussians. Our learned algorithms never perform worse than k -means++, and in some application domains we see significant improvements.

1 Introduction

Clustering is a fundamental problem in machine learning with applications in many areas including text analysis, transportation networks, social networks, and so on. The high-level goal of clustering is to divide a dataset into natural subgroups. For example, in text analysis we may want to divide documents based on topic, and in social networks we might want to find communities. A common approach to clustering is to set up an objective function and then approximately find the optimal solution according to the objective. There has been a wealth of both theoretical and empirical research in clustering using this approach [19, 12, 7, 6, 24, 35, 11, 2].

The most popular method in practice for clustering is local search, where we start with k centers and iteratively make incremental improvements until a local optimum is reached. For example, Lloyd’s method (sometimes called k -means) [30] and k -medoids [17, 13] are two popular local search algorithms. There are multiple decisions an algorithm designer must make when using a local search algorithm. First, the algorithm designer must decide how to seed local search, e.g., how the algorithm chooses the k initial centers. There is a large body of work on seeding algorithms, since the initial choice of centers can have a large effect on both the quality of the outputted clustering and the time it takes for the algorithm to converge [21, 37, 3]. The best seeding method often depends on the specific application at hand. For example, a “typical problem instance” in one setting may have significantly different properties from that in another, causing some seeding

*Authors’ addresses: {ninamf,tdick,crwhite}@cs.cmu.edu.

methods to perform better than others. Second, the algorithm designer must decide on an objective function for the local search phase (k -means, k -median, etc.) For some applications, there is an obvious choice. For instance, if the application is Wi-Fi hotspot location, then the explicit goal is to minimize the k -center objective function. For many other applications such as clustering communities in a social network, the goal is to find clusters which are close to an unknown target clustering, and we may use an objective function for local search in the hopes that approximately minimizing the chosen objective will produce clusterings which are close to matching the target clustering (in terms of the number of misclassified points). As before, the best objective function for local search may depend on the specific application.

In this paper, we show positive theoretical and empirical results for learning the best initialization and local search procedures over a large family of algorithms. We take a transfer learning approach where we assume there is an unknown distribution \mathcal{D} over problem instances corresponding to our application, and the goal is to use experience from the early instances to perform well on the later instances. For example, if our application is clustering facilities in a city, we would look at a sample of cities with existing optimally-placed facilities, and use this information to find the empirically best seeding/local search pair from an infinite family, and we use this pair to cluster facilities in new cities.

(α, β) -Lloyds++. We define an infinite family of algorithms generalizing Lloyd’s method, with two parameters α and β . Our algorithms have two phases, a seeding phase to find k initial centers (parameterized by α), and a local search phase which uses Lloyd’s method to converge to a local optimum (parameterized by β). In the seeding phase, each point v is sampled with probability proportional to $d_{\min}(v, C)^\alpha$, where C is the set of centers chosen so far and $d_{\min}(v, C) = \min_{c \in C} d(v, c)$. Then Lloyd’s method is used to converge to a local minima for the ℓ_β objective. By ranging $\alpha \in [0, \infty) \cup \{\infty\}$ and $\beta \in [1, \infty) \cup \{\infty\}$, we define our infinite family of algorithms which we call (α, β) -Lloyds++. Setting $\alpha = \beta = 2$ corresponds to the k -means++ algorithm [6]. The seeding phase is a spectrum between random seeding ($\alpha = 0$), and farthest-first traversal [19, 14] ($\alpha = \infty$), and the Lloyd’s step is able to optimize over common objectives including k -median ($\beta = 1$), k -means ($\beta = 2$), and k -center ($\beta = \infty$). We design efficient learning algorithms which receive samples from an application-specific distribution over clustering instances and learn a near-optimal clustering algorithm from our family.

Theoretical analysis. In Section 4 we study both the sample and computational complexity of learning the parameters for (α, β) -Lloyds++ that have the lowest expected cost on the application-specific distribution \mathcal{D} . The expected cost is over two sources of randomness: the distribution \mathcal{D} and the algorithmic randomness during the seeding phase of (α, β) -Lloyds++. To aid in our analysis, we define an associated deterministic version of (α, β) -Lloyds++ that takes as input a clustering instance \mathcal{V} and a vector $\vec{Z} = (z_1, \dots, z_k) \in [0, 1]^k$, where the value z_i is used to deterministically choose the i^{th} center during the seeding phase. When \vec{Z} is sampled uniformly from $[0, 1]^k$, the distribution over outputs of the deterministic algorithm run with \vec{Z} is identical to the randomized version of (α, β) -Lloyds++. Our learning procedure receives a sample $(\mathcal{V}_1, \vec{Z}_1), \dots, (\mathcal{V}_m, \vec{Z}_m)$ drawn i.i.d. from $\mathcal{D} \times \text{Uniform}([0, 1]^k)$ and returns the parameters $\hat{\alpha}$ and $\hat{\beta}$ so that the deterministic algorithm has the lowest average cost on the sample. First, we show that when the sample size m is sufficiently large, these parameters have approximately optimal cost in expectation over both \mathcal{D} and the internal randomness of (α, β) -Lloyds++. We also give efficient algorithms for finding the empirically optimal parameters.

We prove that when the sample size is $m = \tilde{O}(k/\epsilon^2)$, where k is the number of clusters and $\tilde{O}(\cdot)$ suppresses logarithmic terms, the empirically optimal parameters $(\hat{\alpha}, \hat{\beta})$ have expected cost at most

ϵ higher than the optimal parameters (α^*, β^*) over the distribution, with high probability over the random sample. The key challenge is that for any clustering instance \mathcal{V} and vector $\vec{Z} \in [0, 1]^k$, the cost of the outputted clustering is not even a continuous function of α or β since a slight tweak in the parameters may lead to a completely different run of the algorithm. In fact, we show that for any clustering instance \mathcal{V} and vector \vec{Z} , the cost is a piecewise constant function of the parameters α and β . The key step in our sample complexity guarantees is to bound the number of discontinuities of the cost function. This requires a delicate reasoning about the structure of “decision points”, which are parameter values where the algorithm output changes, each introducing a discontinuity in the cost function. Our key technical contribution is to leverage the randomness over $\vec{Z} \sim \text{Uniform}([0, 1]^k)$ to prove polynomial bounds on the expected number of decision points for the α parameter. By contrast, if we ignored the distribution of \vec{Z} and applied the techniques exploited by prior work, we would only get exponential bounds on the number of decision points.

Next, we complement our sample complexity result with a computational efficiency result. Specifically, we give a novel meta-algorithm which efficiently finds a near-optimal value $\hat{\alpha}$ with high probability. The high-level idea of our algorithm is to run depth-first-search over the “execution tree” of the algorithm, where a node in the tree represents a state of the algorithm, and edges represent decision points. A key step in our meta-algorithm is to iteratively solve for the decision points of the algorithm, which itself is nontrivial since the equations governing the decision points do not have closed-form solutions. We show the equations have a certain structure which allows us to binary search through the range of parameters to find the decision points.

Experiments. We give a thorough experimental analysis of our family of algorithms by evaluating their performance on a number of different real-world and synthetic application domains including MNIST, Cifar10, CNAE-9, and mixtures of Gaussians. In each case, we create clustering instances by choosing subsets of the labels. For example, we look at an instance of MNIST with digits $\{0, 1, 2, 3, 4\}$, and also an instance with digits $\{5, 6, 7, 8, 9\}$. We show the optimal parameters transfer from one instance to the other. Among domains, there is no single parameter setting that is nearly optimal, and for some domains, the best algorithm from the (α, β) -Lloyds++ family performs significantly better than known algorithms such as k -means++ and farthest-first traversal.

2 Related Work

Clustering. The iterative local search method for clustering, known as Lloyd’s algorithm or sometimes called k -means, is one of the most popular algorithms for k -means clustering [30], and improvements are still being found [34, 32, 15, 36, 23, 24]. The worst-case runtime of Lloyd’s method is exponential [5] even in \mathbb{R}^2 [44], however, it converges very quickly in practice [20], and the smoothed complexity is polynomial [4]. Many different initialization approaches have been proposed [21, 37, 3]. When using d^2 -sampling to find the initial k centers, the algorithm is known as k -means++, and the approximation guarantee is provably $O(\log k)$ [6]. If the data satisfies a natural stability condition, k -means++ returns a near-optimal clustering [35]. The farthest-first traversal algorithm is an iterative method to find k centers, and it was shown to give a 2-approximation algorithm for k -center [19], and an 8-approximation for hierarchical k -center [14].

Transfer learning for unsupervised settings. Balcan et al. shows provable guarantees for learning over a different family of algorithms, linkage-based clustering with dynamic pruning, in the same distribution as the current work, however, they provide no experimental guarantees [9]. There are several related models for learning the best representation and transfer learning for

clustering. For example, Ashtiani and Ben-David analyze the problem of learning a near-optimal data embedding function from a given family of embeddings for the k -means objective [8].

There are a few models for the question of finding the best clustering algorithm to use on a single instance, given a small amount of expert advice. Ackerman et al. (building off of the celebrated clustering impossibility result of [25]) study the problem of taxonomizing clustering algorithmic paradigms, by using a list of abstract properties of clustering functions [1]. In their work, the goal is for a user to choose a clustering algorithm based on the specific properties which are important for her application.

Another related area is the problem of unsupervised domain adaption. In this problem, the machine learning algorithm has access to a labeled training dataset, and an unlabeled target dataset over a different distribution. The goal is to find an accurate classifier over the target dataset, while only training on the training distribution [40, 18, 43].

There has been more research on related questions for transfer learning on unlabeled data and unsupervised tasks. Raina et al. study transfer learning using unlabeled data, to a supervised learning task [39]. Jiang and Chung, and Yang et al. study transfer learning for clustering, in which a clustering algorithm has access to unlabeled data, and uses it to better cluster a related problem instance [45, 22]. This setting is a bit different from ours, since we assume we have access to the target clustering for each training instance, but we tackle the harder question of finding the best clustering objective.

3 Preliminaries

Clustering. A clustering instance \mathcal{V} consists of a point set V of size n , a distance metric d (such as Euclidean distance in \mathbb{R}^d), and a desired number of clusters $1 \leq k \leq n$. A clustering $\mathcal{C} = \{C_1, \dots, C_k\}$ is a k -partitioning of V . Often in practice, clustering is carried out by approximately minimizing an objective function (which maps each clustering to a nonzero value). Common objective functions such as k -median and k -means come from the ℓ_p family, where each cluster C_i is assigned a center c_i and $\text{cost}(\mathcal{C}) = \left(\sum_i \sum_{v \in C_i} d(v, c_i)^p\right)^{\frac{1}{p}}$ (k -median and k -means correspond to $p = 1$ and $p = 2$, respectively). There are two distinct goals for clustering depending on the application. For some applications such as computing facility locations, the algorithm designer’s only goal is to find the best centers, and the actual partition $\{C_1, \dots, C_k\}$ is not needed. For many other applications such as clustering documents by subject, clustering proteins by function, or discovering underlying communities in a social network, there exists an unknown “target” clustering $\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$, and the goal is to output a clustering \mathcal{C} which is close to \mathcal{C}^* . Formally, we define \mathcal{C} and \mathcal{C}' to be ϵ -close if there exists a permutation σ such that $\sum_{i=1}^k |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$. For these applications, the algorithm designer chooses an objective function while hoping that minimizing the objective function will lead to a clustering that is close to the target clustering. In this paper, we will focus on the cost function set to the distance to the target clustering, however, our analysis holds for an abstract cost function `cost` which can be set to an objective function or any other well-defined measure of cost.

Algorithm Configuration. In this work, we assume that there exists an unknown, application-specific distribution \mathcal{D} over a set of clustering instances such that for each instance \mathcal{V} , $|V| \leq n$. We suppose there is a cost function that measures the quality of a clustering of each instance. As discussed in the previous paragraph, we can set the cost function to be the expected Hamming distance of the returned clustering to the target clustering, the cost of an ℓ_p objective, or any other function. The learner’s goal is to find the parameters α and β that approximately minimize the

expected cost with respect to the distribution \mathcal{D} . Our main technical results bound the intrinsic complexity of the class of (α, β) -Lloyds++ clustering algorithms, which leads to generalization guarantees through standard Rademacher complexity [10, 27]. This implies that the empirically optimal parameters are also nearly optimal in expectation.

4 (α, β) -Lloyds++

In this section, we define an infinite family of algorithms generalizing Lloyd’s algorithm, with one parameter controlling the initialization procedure, and another parameter controlling the local search procedure. Our main results bound the intrinsic complexity of this family of algorithms (Theorems 4 and 5) and lead to sample complexity results guaranteeing the empirically optimal parameters over a sample are close to the optimal parameters over the unknown distribution. We measure optimality in terms of agreement with the target clustering. We also show theoretically that no parameters are optimal over all clustering applications (Theorem 2). Finally, we give an efficient algorithm for learning the best initialization parameter (Theorem 8).

Our family of algorithms is parameterized by choices of $\alpha \in [0, \infty) \cup \{\infty\}$ and $\beta \in [1, \infty) \cup \{\infty\}$. Each choice of (α, β) corresponds to one local search algorithm. A summary of the algorithm is as follows. The algorithm has two phases. The goal of the first phase is to output k initial centers. Each center is iteratively chosen by picking a point with probability proportional to the minimum distance to all centers picked so far, raised to the power of α . The second phase is an iterative two step procedure similar to Lloyd’s method, where the first step is to create a Voronoi partitioning of the points induced by the initial set of centers, and then a new set of centers is chosen by computing the ℓ_β mean of the points in each Voronoi tile.

Our goal is to find parameters that return clusterings close to the ground-truth in expectation. Setting $\alpha = \beta = 2$ corresponds to the k -means++ algorithm. The seeding phase is a spectrum between random seeding ($\alpha = 0$), and farthest-first traversal ($\alpha = \infty$), and the Lloyd’s algorithm can optimize for common clustering objectives including k -median ($\beta = 1$), k -means ($\beta = 2$), and k -center ($\beta = \infty$).

On the way to proving our main results, we analyze a deterministic version of (α, β) -Lloyds++ that takes as input both a clustering instance \mathcal{V} and a vector $\vec{Z} = (z_1, \dots, z_k) \in [0, 1]^k$. The deterministic algorithm uses the value z_t when choosing the t^{th} center in the first phase of the algorithm. More specifically, the algorithm chooses the t^{th} center as follows: for each point $v_i \in V$ we determine the d^α -sampling probability of choosing v_i as the next center. Then, we construct a partition of $[0, 1]$ into $|V|$ intervals, where each point v_i is associated with exactly one interval, and the width of the interval is equal to the d^α -sampling probability of choosing v_i . Finally, the algorithm chooses the next center to be the point v_i whose interval contains the value z_t . When z_t is drawn uniformly at random from $[0, 1]$, the probability of choosing the point v_i to be the next center is the width of its corresponding interval, which is the d^α -sampling probability. Therefore, for any fixed clustering instance \mathcal{V} , sampling the vector \vec{Z} uniformly at random from the cube $[0, 1]^k$ and running this algorithm on \mathcal{V} and \vec{Z} has the same output distribution as (α, β) -Lloyds++. Pseudocode for the deterministic version of (α, β) -Lloyds++ is given in Algorithm 1.

Notation. Before presenting our results, we introduce some convenient notation. We define cost functions for both the randomized and deterministic versions of (α, β) -Lloyds++. Given a clustering instance \mathcal{V} and a vector $\vec{Z} \in [0, 1]^k$, we let $\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})$ be the cost of the clustering output by Algorithm 1 (i.e., the distance to the ground-truth clustering for \mathcal{V}) when run on (\mathcal{V}, \vec{Z}) with parameters α and β . Since the algorithm is deterministic, this is a well-defined function. Next, we also define $\text{clus}_{\alpha, \beta}(\mathcal{V}) = \mathbb{E}_{\vec{Z}}[\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})]$ to denote the expected cost of (randomized)

(α, β) -Lloyds++ run on instance \mathcal{V} , where the expectation is taken over the randomness of the algorithm (i.e., over the draw of the vector $\vec{Z} \sim \text{Uniform}([0, 1]^k)$). To facilitate our analysis of phase 2 of Algorithm 1, we let $\text{lloyds}_\beta(\mathcal{V}, C, T)$ denote the cost of the clustering obtained by running Lloyd’s algorithm with parameter β starting from initial centers C for at most T iterations on the instance \mathcal{V} .

Algorithm 1 Deterministic (α, β) -Lloyds++ Clustering

Input: Instance $\mathcal{V} = (V, d, k)$, vector $\vec{Z} = (z_1, \dots, z_k) \in [0, 1]^k$, parameters α and β .

Phase 1: Choosing initial centers with d^α -sampling

1. Initialize $C = \emptyset$.
2. For each $t = 1, \dots, k$:
 - (a) Partition $[0, 1]$ into n intervals, where there is an interval I_{v_i} for each v_i with size equal to the probability of choosing v_i during d^α -sampling in round t (see Figure 2).
 - (b) Denote c_t as the point such that $z_t \in I_{c_t}$, and add c_t to C .

Phase 2: Lloyd’s algorithm

5. Set $C' = \emptyset$. Let $\{C_1, \dots, C_k\}$ denote the Voronoi tiling of V induced by centers C .
6. Compute $\text{argmin}_{x \in V} \sum_{v \in C_i} d(x, v)^\beta$ for all $1 \leq i \leq k$, and add it to C' .
7. If $C' \neq C$, set $C = C'$ and goto 5.

Output: Centers C and clustering induced by C .

When analyzing phase 1 of Algorithm 1, we let $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ denote the vector of centers output by phase 1 when run on a clustering instance \mathcal{V} with vector \vec{Z} . For a given set of centers C , we let d_i denote the distance from point v_i to the set C ; that is, $d_i = \min_{c \in C} d(v_i, c)$. For each point index i , we define $D_i(\alpha) = \sum_{j=1}^i d_j^\alpha$, so that the probability of choosing point i as the next center under d^α -sampling is equal to $d_i^\alpha / D_n(\alpha)$, and the probability that the chosen index belongs to $\{1, \dots, i\}$ is $D_i(\alpha) / D_n(\alpha)$. When we use this notation, the set of centers C will always be clear from context. Finally, for a point set V , we let $R = \max\{d(x, x') / d(y, y') \mid x, x', y, y' \in V, d(y, y') \neq 0\}$ denote the maximum ratio between any pair of non-zero distances in the point set. The notation used throughout the paper is summarized in Appendix A.

We start with two structural results about the family of (α, β) -Lloyds++ clustering algorithms. The first shows that for sufficiently large α , phase 1 of Algorithm 1 is equivalent to farthest-first traversal. This means that it is sufficient to consider α parameters in a bounded range.

Farthest-first traversal [19] starts by choosing a random center, and then iteratively choosing the point farthest to all centers chosen so far, until there are k centers. We assume that ties are broken uniformly at random. Farthest-first traversal is equivalent to the first phase of Algorithm 1 when run with $\alpha = \infty$. The following result guarantees that when α is sufficiently large, Algorithm 1 chooses the same initial centers as farthest-first traversal with high probability.

Lemma 1. *For any clustering instance $\mathcal{V} = (V, d, k)$ and $\delta > 0$, if $\alpha > \log(\frac{nk}{\delta}) / \log s$, where s denotes the minimum ratio d_1 / d_2 between two distances $d_1 > d_2$ in the point set, then $P_{\vec{Z}}(\text{seed}_\alpha(\mathcal{V}, \vec{Z}) = \text{seed}_\infty(\mathcal{V}, \vec{Z})) \geq 1 - \delta$.*

For some datasets, $1 / \log s$ might be very large. In Section 5, we empirically observe that for

all datasets we tried, (α, β) -Lloyds++ behaves the same as farthest-first traversal for $\alpha > 20$.¹

Next, to motivate learning the best parameters, we show that for *any* pair of parameters (α^*, β^*) , there exists a clustering instance such that (α^*, β^*) -Lloyds++ outperforms all other values of α, β . This implies that d^β -sampling is not always the best choice of seeding for the ℓ_β objective.

Theorem 2. For $\alpha^* \in [.01, \infty) \cup \{\infty\}$ and $\beta^* \in [1, \infty) \cup \{\infty\}$, there exists a clustering instance \mathcal{V} whose target clustering is the optimal ℓ_{β^*} clustering, such that $\text{clus}_{\alpha^*, \beta^*}(\mathcal{V}) < \text{clus}_{\alpha, \beta}(\mathcal{V})$ for all $(\alpha, \beta) \neq (\alpha^*, \beta^*)$.

Proof sketch. Consider $\alpha^*, \beta^* \in [0, \infty) \cup \{\infty\}$. The clustering instance consists of 6 clusters, C_1, \dots, C_6 . These 6 clusters are both the target clustering for the instance, and they are optimal for the ℓ_{β^*} objective. The proof consists of three sections. First, we construct C_1, \dots, C_4 so that d^{α^*} sampling has the best chance of putting exactly one point into each optimal cluster. Then we add “local minima traps” to each cluster, so that if any cluster received two centers in the sampling phase, Lloyd’s method will not be able to move the centers to a different cluster. Finally, we construct C_5 and C_6 so that if seeding put one point in each cluster, then β^* -Lloyd’s method will outperform any other $\beta \neq \beta^*$.

We construct the instance in an abstract metric space where we can define pairwise distances to take any values, provided that they still satisfy the triangle inequality. We refer to a collection of points as a clique if all pairwise distances within the collection are equal.

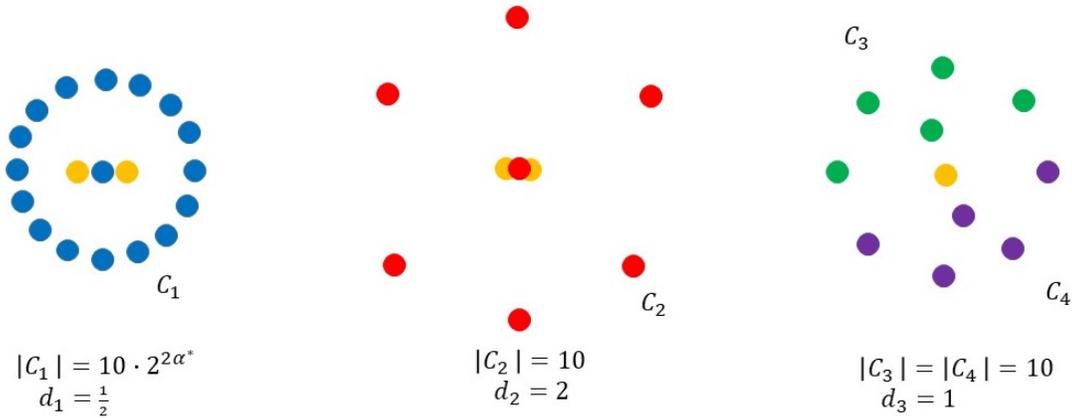


Figure 1: Optimal instance for d^{α^*} -sampling

Step 1: we define C_1 and C_2 to be two different cliques, and we define a third clique whose points are equal to $C_3 \cup C_4$. See Figure 1. We space these cliques arbitrarily far apart so that with high probability, the first three sampled centers will each be in a different clique. Now the idea is to define the distances and sizes of the cliques so that α^* is the value of α with the greatest chance of putting the last center into the third clique. If we set the distances in cliques 1,2,3 to $d_1 = 2$, $d_2 = 1/2$, and $d_3 = 1$, and set $|C_1| = 2^{2\alpha^*}|C_2|$, then the probability of sampling a 4th center in the third clique for $\alpha = \alpha^* + \delta$ is equal to

$$\frac{|C_3 \cup C_4|}{|C_3 \cup C_4| + (2^{\alpha^* + \delta} + 2^{\alpha^* - \delta})|C_2|}.$$

This is maximized when $\delta = 0$.

¹ In Appendix B, we show that if the dataset satisfies a stability assumption called separability [26, 38], then (α, β) -Lloyds++ outputs the same clustering as farthest-first traversal with high probability when $\alpha > \log n$.

Now we add local minima traps for Lloyd’s method as follows. In the first two cliques, we add three centers so that the 2-clustering cost is only slightly better than the 1-clustering cost. In the third clique, which consists of $C_3 \cup C_4$, add centers so that the 2-clustering cost is much lower than the 1-clustering cost. We also show that since all cliques are far apart, it is not possible for a center to move between clusters during Lloyd’s method.

Finally, we add three centers c_5, b_5, b'_5 to the last cluster C_5 . We set the rest of the points so that c_5 minimizes the ℓ_{β^*} objective, while b_5 and b'_5 favor $\beta = \beta^* \pm \epsilon$. Therefore, (α^*, β^*) performs the best out of all pairs (α, β) . \square

Sample efficiency. Now we give sample complexity bounds for learning the best algorithm from the class of (α, β) -Lloyds++ algorithms. We analyze the phases of Algorithm 1 separately. For the first phase, our main structural result is to show that for a given clustering instance, with high probability over the draw of $\vec{Z} \sim \text{Uniform}([0, 1]^k)$, the number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ as we vary $\alpha \in [\alpha_\ell, \alpha_h]$ is $O(nk \log(n) \log(\alpha_h/\alpha_\ell))$. Our analysis crucially harnesses the randomness of \vec{Z} to achieve this bound. For instance, if we ignore the distribution of \vec{Z} and use a purely combinatorial approach as in prior algorithm configuration work, we would only achieve a bound of $n^{O(k)}$, which is the total number of sets of k centers. For completeness, we give a combinatorial proof of $O(n^{k+3})$ discontinuities in Appendix B (Theorem 15). Similarly, for the second phase of the algorithm, we show that for any clustering instance \mathcal{V} , initial set of centers C , and any maximum number of iterations T , the function $\beta \mapsto \text{lloyds}_\beta(\mathcal{V}, C, T)$ has at most $O(\min(n^{3T}, n^{k+3}))$ discontinuities.

We begin by analyzing the number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$. Before proving the $O(nk \log(n) \log(\alpha_h/\alpha_\ell))$ upper bound, we define a few concepts used in the proof. Assume we start to run Algorithm 1 without a specific setting of α , but rather a range $[\alpha_\ell, \alpha_h]$, for some instance \mathcal{V} and vector \vec{Z} . In some round t , if Algorithm 1 would choose the same center c_t for every setting of $\alpha \in [\alpha_\ell, \alpha_h]$, then we continue normally. However, if the algorithm would choose a different center depending on the specific value of α used from the interval $[\alpha_\ell, \alpha_h]$, then we fork the algorithm, making one copy for each possible next center. In particular, we partition $[\alpha_\ell, \alpha_h]$ into a finite number of sub-intervals such that the next center is constant on each interval. The boundaries between these intervals are “breakpoints”, since as α crosses those values, the next center chosen by the algorithm changes. Our goal is to bound the total number of breakpoints over all k rounds in phase 1 of Algorithm 1, which bounds the number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$.

A crucial step in the above approach is determining where the breakpoints are located. Recall that in round t of Algorithm 1, each datapoint v_i is assigned an interval in $[0, 1]$ of size $d_i^\alpha/D_n(\alpha)$, where d_i is the minimum distance from v_i to the current set of centers, and $D_j(\alpha) = d_1^\alpha + \dots + d_j^\alpha$. The interval for point v_i is $[\frac{D_{i-1}(\alpha)}{D_n(\alpha)}, \frac{D_i(\alpha)}{D_n(\alpha)})$ (see Figure 2). WLOG, we assume that the algorithm sorts the points on each round so that $d_1 \geq \dots \geq d_n$. We prove the following nice structure about these intervals.

Lemma 3. *Assume that v_1, \dots, v_n are sorted in decreasing distance from a set C of centers. Then for each $i = 1, \dots, n$, the function $\alpha \mapsto \frac{D_i(\alpha)}{D_n(\alpha)}$ is monotone increasing and continuous along $[0, \infty)$. Furthermore, for all $1 \leq i < j \leq n$ and $\alpha \in [0, \infty)$, we have $\frac{D_i(\alpha)}{D_n(\alpha)} \leq \frac{D_j(\alpha)}{D_n(\alpha)}$.*

This lemma guarantees two crucial properties. First, we know that for every (ordered) set C of $t \leq k$ centers chosen by phase 1 of Algorithm 1 up to round t , there is a single interval (as opposed to a more complicated set) of α -parameters that would give rise to C . Second, for an interval

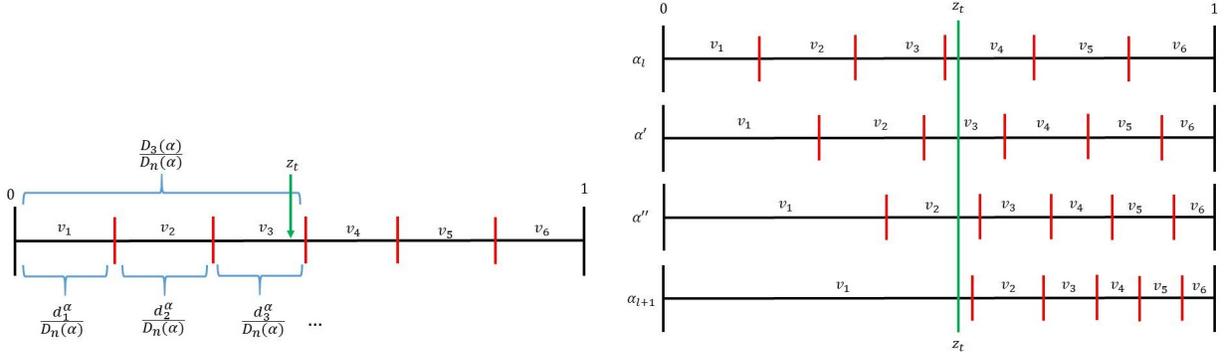


Figure 2: The algorithm chooses v_3 as a center (left). In the interval $[\alpha_\ell, \alpha_{\ell+1}]$, the algorithm may choose v_4, v_3, v_2 , or v_1 as a center, based on the value of α (right).

$[\alpha_\ell, \alpha_h]$, the set of possible next centers is exactly $v_{i_\ell}, v_{i_\ell+1}, \dots, v_{i_h}$, where i_ℓ and i_h are the centers sampled when α is α_ℓ and α_h , respectively (see Figure 2).

Now we are ready to prove our main structural result, which bounds the number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ in expectation over $\vec{Z} \sim \text{Uniform}([0, 1]^k)$. We prove two versions of the result: one that holds over parameter intervals $[\alpha_\ell, \alpha_h]$ with $\alpha_\ell > 0$, and a version that holds when $\alpha_\ell = 0$, but that depends on the largest ratio of any pairwise distances in the dataset.

Theorem 4. Fix any clustering instance \mathcal{V} and let $\vec{Z} \sim \text{Uniform}([0, 1]^k)$. Then:

1. For any parameter interval $[\alpha_\ell, \alpha_h]$ with $\alpha_\ell > 0$, the expected number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ on $[\alpha_\ell, \alpha_h]$ is at most $O(nk \log(n) \log(\alpha_h/\alpha_\ell))$.
2. For any parameter interval $[0, \alpha_h]$, the expected number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ on $[0, \alpha_h]$ is at most $O(nk \log(n) \log(\alpha_h \log(R)))$, where R is the largest ratio between any pair of non-zero distances in \mathcal{V} .

Proof sketch. Consider round t in the run of phase 1 in Algorithm 1 on instance \mathcal{V} with vector \vec{Z} . Suppose at the beginning of round t , there are L possible states of the algorithm; that is, L α -intervals such that within each interval, the choice of the first $t - 1$ centers is fixed. By Lemma 3, we can write these sets as $[\alpha_0, \alpha_1], \dots, [\alpha_{L-1}, \alpha_L]$, where $0 = \alpha_0 < \dots < \alpha_L = \alpha_h$. Given one interval, $[\alpha_\ell, \alpha_{\ell+1}]$, we claim the expected number of new breakpoints, denoted by $\#I_{t,\ell}$, introduced by choosing a center in round t starting from the state for interval ℓ is bounded by

$$\min \{2n \log(R)(\alpha_{\ell+1} - \alpha_\ell), n - t - 1, 4n \log(n)(\log \alpha_{\ell+1} - \log \alpha_\ell)\}.$$

Note that $\#I_{t,\ell} + 1$ is the number of possible choices for the next center in round t using α in $[\alpha_\ell, \alpha_{\ell+1}]$.

The claim gives three different upper bounds on the expected number of new breakpoints, where the expectation is only over $z_t \sim \text{Uniform}([0, 1])$, and the bounds hold for any given configuration of $d_1 \geq \dots \geq d_n$ (i.e., it does not depend on the centers that have been chosen on prior rounds). To prove the first statement in Theorem 4, we only need the last of the three bounds, and to prove the second statement, we need all three bounds.

First we show how to prove the first part of the theorem assuming the claim, and later we will prove the claim. We prove the first statement as follows. Let $\#I$ denote the total number of discontinuities of $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ for $\alpha \in [\alpha_\ell, \alpha_h]$. Then we have

$$\begin{aligned}
\mathbb{E}_{\bar{Z}}[\#I] &\leq \mathbb{E}_{\bar{Z}} \left[\sum_{t=1}^k \sum_{\ell=1}^{L-1} (\#I_{t,\ell}) \right] \\
&= \sum_{t=1}^k \sum_{\ell=0}^{L-1} \mathbb{E}_{\bar{Z}}[\#I_{t,\ell}] \\
&\leq \sum_{t=1}^k \sum_{\ell=0}^{L-1} 4n \log(n) (\log \alpha_{\ell+1} - \log \alpha_{\ell}) \\
&= \sum_{t=1}^k 4n \log(n) (\log \alpha_h - \log \alpha_{\ell}) \\
&= O \left(nk \log n \log \frac{\alpha_h}{\alpha_{\ell}} \right)
\end{aligned}$$

Now we prove the second part of the theorem. Let ℓ^* denote the largest value such that $\alpha_{\ell^*} < \frac{1}{\log R}$. Such an ℓ^* must exist because $\alpha_0 = 0$. Then we have $\alpha_{\ell^*} < \frac{1}{\log R} \leq \alpha_{\ell^*+1}$. We use three upper bounds for three different cases of alpha intervals: the first ℓ^* intervals, interval $[\alpha_{\ell^*}, \alpha_{\ell^*+1}]$, and intervals $\ell^* + 2$ to L . Let $\#I$ denote the total number of discontinuities of $\alpha \mapsto \mathbf{seed}_{\alpha}(\mathcal{V}, \bar{Z})$ for $\alpha \in [0, \alpha_h]$.

$$\begin{aligned}
\mathbb{E}_{\bar{Z}}[\#I] &\leq \mathbb{E}_{\bar{Z}} \left[\sum_{t=1}^k \sum_{\ell=1}^{L-1} (\#I_{t,\ell}) \right] \\
&= \sum_{t=1}^k \sum_{\ell=0}^{L-1} \mathbb{E}_{\bar{Z}}[\#I_{t,\ell}] \\
&= \sum_{t=1}^k \left(\sum_{\ell=0}^{\ell^*-1} \mathbb{E}_{\bar{Z}}[\#I_{t,\ell}] + \mathbb{E}_{\bar{Z}}[\#I_{t,\ell^*}] + \sum_{\ell=\ell^*+1}^{L-1} \mathbb{E}_{\bar{Z}}[\#I_{t,\ell}] \right) \\
&\leq \sum_{t=1}^k \left(\sum_{\ell=0}^{\ell^*-1} (2n \log R (\alpha_{\ell+1} - \alpha_{\ell})) + (n - t - 1) + \sum_{\ell=\ell^*+1}^{L-1} (4n \log n (\log \alpha_{\ell+1} - \log \alpha_{\ell})) \right) \\
&\leq \sum_{t=1}^k (2n \log R \cdot \alpha_{\ell^*} + n + 4n \log n (\log \alpha_h - \log \alpha_{\ell^*})) \\
&\leq \sum_{t=1}^k \left(2n \log(R) \cdot \frac{1}{\log R} + n + 4n \log(n) \left(\log \alpha_h - \log \left(\frac{1}{\log R} \right) \right) \right) \\
&= O(nk \log(n) (\log(\alpha_h \log(R))))
\end{aligned}$$

Now we will prove the claim. Given $z_t \in [0, 1]$, let x and y denote the minimum indices s.t. $\frac{D_x(\alpha_{\ell})}{D_n(\alpha_{\ell})} > z_t$ and $\frac{D_y(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} > z_t$, respectively. Then from Lemma 3, the number of breakpoints for $\alpha \in [\alpha_{\ell}, \alpha_{\ell+1}]$ is exactly $\#I_{t,\ell} = x - y$ (see Figure 3). Therefore, our goal is to compute $\mathbb{E}_{z_t}[x - y]$. One method is to sum up the expected number of breakpoints for each interval I_v by bounding the maximum possible number of breakpoints given that z_t lands in I_v . However, this will sometimes lead to a bound that is too coarse. For example, if $\alpha_{\ell+1} - \alpha_{\ell} = \epsilon \approx 0$, then for each bucket I_{v_j} , the maximum number of breakpoints is 1, but we want to show the expected number of breakpoints is

proportional to ϵ . To tighten up this analysis, we will show that for each bucket, the probability (over z_t) of achieving the maximum number of breakpoints is low.

Assuming that z_t lands in a bucket I_{v_j} , we further break into cases as follows. Let i denote the minimum index such that $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} > \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$. Note that i is a function of j, α_ℓ , and $\alpha_{\ell+1}$, but it does not depend on z_t . If z_t is less than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then we have the maximum number of breakpoints possible, since the algorithm chooses center v_{i-1} when $\alpha = \alpha_{\ell+1}$ and it chooses center v_j when $\alpha = \alpha_\ell$. The number of breakpoints is therefore $j - i + 1$, by Lemma 3. We denote this event by $E_{t,j}$, i.e., $E_{t,j}$ is the event that in round t , z_t lands in I_{v_j} and is less than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$. If z_t is instead greater than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then the algorithm chooses center v_i when $\alpha = \alpha_{\ell+1}$, so the number of breakpoints is $\leq j - i$. We denote this event by $E'_{t,j}$ (see Figure 3). Note that $E_{t,j}$ and $E'_{t,j}$ are disjoint and $E_{t,j} \cup E'_{t,j}$ is the event that $z_t \in I_{v_j}$.

Within an interval I_{v_j} , the expected number of breakpoints is

$$P(E_{t,j}) \cdot (j - i + 1) + P(E'_{t,j}) \cdot (j - i) = P(E_{t,j} \cup E'_{t,j}) \cdot (j - i) + P(E_{t,j}).$$

We will bound $j - i$ and $P(E_{t,j})$ separately.

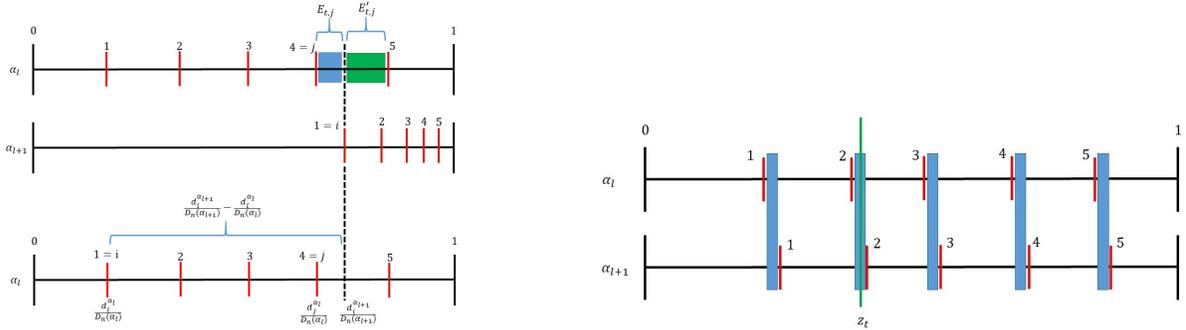


Figure 3: Definition of $E_{t,j}$ and $E'_{t,j}$, and details for bounding $j - i$ (left). Intuition for bounding $P(E_{t,j})$, where the blue regions represent $E_{t,j}$ (right).

First we upper bound $P(E_{t,j})$. Recall this is the probability that z_t is in between $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$ and $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, which is

$$\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} \leq \frac{D_j(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}.$$

Therefore, we can bound this quantity by bounding the derivative $\left| \frac{\partial}{\partial \alpha} \left(\frac{D_j(\alpha)}{D_n(\alpha)} \right) \right|$, which we show is at most $\min \left\{ \frac{2}{\alpha} \log n, \log(R) \right\}$ in Appendix B.

Now we upper bound $j - i$. Recall that $j - i$ represents the number of intervals between $\frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}$ and $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$ (see Figure 3). Note that the smallest interval in this range has width $\frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)}$, and

$$\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)} \leq \frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}.$$

Again, we can use the derivative of $\frac{D_i(\alpha)}{D_n(\alpha)}$ to bound $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}$. To finish off the proof of the claim, we have

$$\begin{aligned}
\mathbb{E}[\#I_{t,\ell}] &\leq \sum_j (P(E'_{t,j}) \cdot (j-i) + P(E_{t,j}) \cdot (j-i+1)) \\
&= \sum_j (P(E'_{t,j} \cup E_{t,j}) \cdot (j-i) + P(E_{t,j})) \\
&= \sum_j P(z_t \in I_{v_j}) \cdot (j-i) + \sum_j P(E_{t,j}) \\
&\leq \sum_j \left(\frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)} \right) \left(\frac{D_n(\alpha_\ell)}{d_j^{\alpha_\ell}} \cdot \frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \right) + \sum_j \left(\frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \right) \\
&\leq 2n \left(\frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \right) \\
&\leq 2n \min(2 \log n(\log \alpha_{\ell+1} - \log \alpha_\ell), \log D(\alpha_{\ell+1} - \alpha_\ell))
\end{aligned}$$

This accounts for two of the three upper bounds in our claim. To complete the proof, we note that $\mathbb{E}[\#I_{t,\ell}] \leq n-t-1$ simply because there are only $n-t$ centers available to be chosen in round t of the algorithm (and therefore, $n-t-1$ breakpoints). \square

Now we analyze phase 2 of Algorithm 1. Since phase 2 does not have randomness, we use combinatorial techniques. Recall that $\text{lloyd}_\beta(\mathcal{V}, C, T)$ denotes the cost of the outputted clustering from phase 2 of Algorithm 1 on instance \mathcal{V} with initial centers C , and a maximum of T iterations.

Theorem 5. *Given $T \in \mathbb{N}$, a clustering instance \mathcal{V} , and a fixed set C of initial centers, the number of discontinuities of $\text{lloyd}_\beta(\mathcal{V}, C, T)$ as a function of β on instance \mathcal{V} is $O(\min(n^{3T}, n^{k+3}))$.*

Proof sketch. Given \mathcal{V} and a set of initial centers C , we bound the number of discontinuities introduced in the Lloyd's step of Algorithm 1. First, we give a bound of n^{k+3} which holds for any value of T . Recall that Lloyd's algorithm is a two-step procedure, and note that the Voronoi partitioning step is independent of β . Let $\{C_1, \dots, C_k\}$ denote the Voronoi partition of V induced by C . Given one of these clusters C_i , the next center is computed by $\min_{c \in C_i} \sum_{v \in C_i} d(c, v)^\beta$. Given any $c_1, c_2 \in C_i$, the decision for whether c_1 is a better center than c_2 is governed by $\sum_{v \in C_i} d(c_1, v)^\beta < \sum_{v \in C_i} d(c_2, v)^\beta$. By a consequence of Rolle's theorem, this equation has at most $2n+1$ roots. This equation depends on the set C of centers, and the two points c_1 and c_2 , therefore, there are $\binom{n}{k} \cdot \binom{n}{2}$ equations each with $2n+1$ roots. We conclude that there are n^{k+3} total intervals of β such that the outcome of Lloyd's method is fixed.

Next we give a different analysis which bounds the number of discontinuities by n^{3T} , where T is the maximum number of Lloyd's iterations. By the same analysis as the previous paragraph, if we only consider one round, then the total number of equations which govern the output of a Lloyd's iteration is $\binom{n}{2}$, since the set of centers C is fixed. These equations have $2n+1$ roots, so the total number of intervals in one round is $O(n^3)$. Therefore, over T rounds, the number of intervals is $O(n^{3T})$. \square

By combining Theorem 4 with Theorem 5, and using standard learning theory results, we can bound the sample complexity needed to learn near-optimal parameters α, β for an unknown distribution \mathcal{D} over clustering instances. Recall that $\text{clus}_{\alpha,\beta}(\mathcal{V})$ denotes the expected cost of the clustering outputted by (α, β) -Lloyds++, with respect to the target clustering, taken over both the random draw of a new instance and the algorithm randomness. Let H denote an upper bound on $\text{clus}_{\alpha,\beta}(\mathcal{V})$.

Theorem 6. Let \mathcal{D} be a distribution over clustering instances with k clusters and at most n points, and let $\mathcal{S} = \{(\mathcal{V}^{(i)}, \vec{Z}^{(i)})\}_{i=1}^m$ be an i.i.d. sample from $\mathcal{D} \times \text{Uniform}([0, 1]^k)$. For any parameters α, β , let $L_{\mathcal{S}}(\alpha, \beta) = \frac{1}{m} \sum_{i=1}^m \text{clus}_{\alpha, \beta}(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ denote the sample loss, and $L_{\mathcal{D}}(\alpha, \beta) = \mathbb{E}[\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})]$ denote the expected loss. The following statements hold:

1. For any $\epsilon > 0$, $\delta > 0$, and parameter interval $[\alpha_\ell, \alpha_h]$ with $\alpha_\ell > 0$, if the sample is of size $m = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\min(T, k) \log n + \log k + \log \frac{1}{\delta} + \log(\log(\frac{\alpha_h}{\alpha_\ell}))\right)\right)$ then with probability at least $1 - \delta$, for all $(\alpha, \beta) \in [\alpha_\ell, \alpha_h] \times [1, \infty]$, we have $|L_{\mathcal{S}}(\alpha, \beta) - L_{\mathcal{D}}(\alpha, \beta)| < \epsilon$.
2. Suppose that the maximum ratio of any non-zero distances is bounded by R for all clustering instances in the support of \mathcal{D} . Then for any $\epsilon > 0$, $\delta > 0$, and parameter interval $[0, \alpha_h)$, if the sample size is $m = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\min(T, k) \log n + \log k + \log \frac{1}{\delta} + \log(\log(\alpha_h \log(R)))\right)\right)$, then with probability at least $1 - \delta$, for all $(\alpha, \beta) \in [0, \alpha_h] \times [1, \infty]$, we have that $|L_{\mathcal{S}}(\alpha, \beta) - L_{\mathcal{D}}(\alpha, \beta)| < \epsilon$.

Proof. We begin by proving the first statement, which guarantees uniform convergence for parameters $(\alpha, \beta) \in [\alpha_\ell, \alpha_h] \times [1, \infty]$.

First, we argue that with probability at least $1 - \delta/2$, for all sample indices $i \in [m]$, the number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ for $\alpha \in [\alpha_\ell, \alpha_h]$ is $O(mnk \log(n) \log(\alpha_h/\alpha_\ell)/\delta)$. By Theorem 4, we know that for any sample index i , the expected number of discontinuities of $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ over the draw of $Z^{(i)} \sim \text{Uniform}([0, 1]^d)$ is $O(nk \log(n) \log(\alpha_h/\alpha_\ell))$. Applying Markov's inequality with failure probability $\delta/(2m)$, we have that with probability at least $1 - \delta/(2m)$ over $\vec{Z}^{(i)}$, the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ has at most $O(mnk \log(n) \log(\alpha_h/\alpha_\ell)/\delta)$ discontinuities. The claim follows by taking the union bound over all m functions. We assume this high probability event holds for the rest of the proof.

Next, we construct a set of $N = O(m^3 nk \log(n) \log(\alpha_h/\alpha_\ell) \min(n^{3T}, n^{k+3})/\delta)$ parameter values $(\alpha_1, \beta_1), \dots, (\alpha_N, \beta_N)$ that exhibit all possible behaviors of the (α, β) -Lloyds++ algorithm family on the entire sample of clustering instances. Taking the union of all $O(m^2 nk \log(n) \log(\alpha_h/\alpha_\ell)/\delta)$ discontinuities of the functions $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ for $i \in [m]$, we can partition $[\alpha_\ell, \alpha_h]$ into $O(m^2 nk \log(n) \log(\alpha_h/\alpha_\ell)/\delta)$ intervals such that for each interval I and any $\alpha, \alpha' \in I$, we have $\text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)}) = \text{seed}_{\alpha'}(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ for all $i \in [m]$. In other words, on each interval and each sample instance, the initial centers chosen by phase 1 of the algorithm is fixed. Now consider any interval I in this partition. For each instance $(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ and any $\alpha \in I$, the set of initial centers chosen by phase 1 of the algorithm is fixed. Therefore, Theorem 5 guarantees that the number of discontinuities of the function $\beta \mapsto \text{lloyds}_{\alpha, \beta}(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ is at most $O(\min(n^{3T}, n^{k+3}))$. By a similar argument, it follows that we can partition the beta parameter space $[1, \infty]$ into $O(m \min(n^{3T}, n^{k+3}))$ intervals such that for each interval the output clustering is constant for all instances (when run with any parameter $\alpha \in I$). Combined, it follows that we can partition the joint parameter space $[\alpha_\ell, \alpha_h] \times [1, \infty]$ into $N = O(m^3 nk \log(n) \log(\alpha_h/\alpha_\ell) \min(n^{3T}, n^{k+3})/\delta)$ rectangles such that for each rectangle and every instance $(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$, the clustering output by (α, β) -Lloyds++ (and therefore the loss) is constant for all (α, β) values in the rectangle. Let $(\alpha_1, \beta_1), \dots, (\alpha_N, \beta_N)$ be a collection of parameter values obtained by taking one pair from each rectangle in the partition.

Finally, to prove the uniform convergence guarantee, we bound the empirical Rademacher complexity of the family of loss functions $\mathcal{F} = \{f_{\alpha, \beta} : (\mathcal{V}, \vec{Z}) \mapsto \text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z}) \mid \alpha \in [\alpha_\ell, \alpha_h], \beta \in [1, \infty]\}$ on the given sample of instances. The empirical Rademacher complexity is defined by

$$\hat{R}(\mathcal{F}, \mathcal{S}) = \frac{1}{m} \mathbb{E}_\sigma \left[\sup_{f_{\alpha, \beta} \in \mathcal{F}} \sum_{i=1}^m \sigma_i f_{\alpha, \beta}(\mathcal{V}^{(i)}, \vec{Z}^{(i)}) \right],$$

Algorithm 2 Dynamic algorithm configuration

Input: Instance $\mathcal{V} = (V, d, k)$, vector $\vec{Z} \in [0, 1]^k$, $\alpha_\ell, \alpha_h, \epsilon > 0$

1. Initialize Q to be an empty queue, then push the root node $(\langle \rangle, [\alpha_\ell, \alpha_h])$ onto Q .
 2. While Q is non-empty
 - (a) Pop node (C, A) from Q with centers C and alpha interval A .
 - (b) For each point u_i that can be chosen as the next center, compute $A_i = \{\alpha \in A : u_i \text{ is the sampled center}\}$ up to error ϵ and set $C_i = C \cup \{u_i\}$.
 - (c) For each i , if $|C_i| < k$, push (C_i, A_i) onto Q . Otherwise, output (C_i, A_i) .
-

where σ is a vector of m i.i.d. Rademacher random variables. The above arguments imply that we can replace the supremum over all of \mathcal{F} by a supremum only over the loss functions with parameters in the finite set $\{(\alpha_j, \beta_j)\}_{j=1}^N$:

$$\hat{R}(\mathcal{F}, \mathcal{S}) = \frac{1}{m} \mathbb{E}_\sigma \left[\sup_{j \in [N]} \sum_{i=1}^m \sigma_i f_{\alpha_j, \beta_j}(V^{(i)}, \vec{Z}^{(i)}) \right].$$

Define the vector $a^{(j)} = (f_{\alpha_j, \beta_j}(\mathcal{V}^{(1)}, \vec{Z}^{(1)}), \dots, f_{\alpha_j, \beta_j}(\mathcal{V}^{(m)}, \vec{Z}^{(m)})) \in [0, H]^m$ for all $j \in [N]$ and let $A = \{a^{(j)} \mid j \in [N]\}$. We have that $\|a_j\|_2 \leq H\sqrt{m}$ for all $j \in [N]$. Applying Massart's Lemma [33] gives

$$\hat{R}(\mathcal{F}, \mathcal{S}) = \frac{1}{m} \mathbb{E}_\sigma \left[\sup_{a \in A} \sum_{i=1}^m \sigma_i a_i \right] \leq H \sqrt{2 \log(N)/m}.$$

From this, the final sample complexity guarantee follows from standard Rademacher complexity bounds [10] using the remaining $\delta/2$ failure probability.

The proof of the second statement follows a nearly identical argument. The only step that needs to be modified is the partitioning of the α parameter space. In this case, we use the second statement from Theorem 4, which guarantees that for each index $i \in [m]$, the expected number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ for $\alpha \in [0, \alpha_h]$ (over the draw of $\vec{Z} \sim \text{Uniform}([0, 1]^k)$) is $O(nk \log(n) \log(\alpha_h \log(R)))$. Applying Markov's inequality and the union bound, we have that with probability at least $1 - \delta/2$, for all $i \in [m]$, the number of discontinuities of $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}^{(i)}, \vec{Z}^{(i)})$ is $O(mnk \log(n) \log(\alpha_h \log(R))/\delta)$. Now the rest of the argument is identical to the proof for the first statement, except the number of rectangles in the partition is $N = O(m^3 nk \log(n) \log(\alpha_h \log(R)) \min(n^{3T}, n^{k+3})/\delta)$, which replaces $1/\alpha_\ell$ by $\log(R)$. \square

Computational efficiency In this section, we present an algorithm for tuning α whose running time scales with the true number of discontinuities over the sample. Combined with Theorem 4, this gives a bound on the expected running time of tuning α .

The high-level idea of our algorithm is to directly enumerate the set of centers that can possibly be output by d^α -sampling for a given clustering instance \mathcal{V} and pre-sampled randomness \vec{Z} . We know from the previous section how to count the number of new breakpoints at any given state in the algorithm, however, efficiently solving for the breakpoints poses a new challenge. From the previous section, we know the breakpoints in α occur when $\frac{D_i(\alpha)}{D_n(\alpha)} = z_t$. This is an exponential

equation with n terms, and there is no closed-form solution for α . Although an arbitrary equation of this form may have up to n solutions, our key observation is that if $d_1 \geq \dots \geq d_n$, then $\frac{D_i(\alpha)}{D_n(\alpha)}$ must be monotone decreasing (from Lemma 3), therefore, it suffices to binary search over α to find the unique solution to this equation. We cannot find the exact value of the breakpoint from binary search (and even if there was a closed-form solution for the breakpoint, it might not be rational), however we can find the value to within additive error ϵ for all $\epsilon > 0$. Now we show that the cost function $\text{clus}_{\alpha,\beta}(\mathcal{V})$ is $(Hnk \log R)$ -Lipschitz in α for a constant-size interval, therefore, it suffices to run $O\left(\log \frac{Hn \log R}{\epsilon}\right)$ rounds of binary search to find a solution whose expected cost is within ϵ of the optimal cost. This motivates Algorithm 2.

Lemma 7. *Given any clustering instance \mathcal{V} with maximum non-zero distance ratio R , $\epsilon > 0$, and $\alpha \in (0, \infty) \cup \{\infty\}$, $P_{\vec{Z}}(\text{seed}_{\alpha}(\mathcal{V}, \vec{Z}) \neq \text{seed}_{\alpha+\epsilon}(\mathcal{V}, \vec{Z})) \leq \min(2nk \log n \log(\frac{\alpha+\epsilon}{\alpha}), \epsilon nk \log R)$.*

Proof. Given a clustering instance \mathcal{V} , $\epsilon > 0$, we will show that, over the draw of $\vec{Z} \sim \text{Uniform}([0, 1]^k)$, there is low probability that $\text{seed}_{\alpha}(\mathcal{V}, \vec{Z})$ outputs a different set of centers than $\text{seed}_{\alpha+\epsilon}(\mathcal{V}, \vec{Z})$. Assume in round t of d^{α} -sampling and $d^{\alpha+\epsilon}$ -sampling, both algorithms have C as the current list of centers. Given we draw $z_t \sim [0, 1]$, we will show there is only a small chance that the algorithms choose different centers in this round. Let I_1, \dots, I_n be the intervals such that the next center chosen by the algorithm with parameter α is v_i whenever $z_t \in I_i$ and I'_1, \dots, I'_n be the intervals for the algorithm with parameter $\alpha + \epsilon$. First we will show that $P_{\vec{Z}}(\text{seed}_{\alpha}(\mathcal{V}, \vec{Z}) \neq \text{seed}_{\alpha+\epsilon}(\mathcal{V}, \vec{Z})) \leq \epsilon nk \log R$. Since the algorithms have an identical set of current centers, the distances $d(v, C)$ are the same, but the breakpoints of the intervals, $\frac{\sum_{j=1}^i d(v_j, C)^{\alpha}}{\sum_{j=1}^n d(v_j, C)^{\alpha}}$ differ slightly. If $z_t \sim [0, 1]$ lands in $I_i \cap I'_i$, the d^{α} and $d^{\alpha+\epsilon}$ will both choose v_i as the next center. Thus, we need to bound the size of $\sum_{i=1}^n (I_i \setminus I'_i) \cup (I'_i \setminus I_i)$. Recall the endpoint of interval i is $\frac{D_i(\alpha)}{D_n(\alpha)}$, where $D_i = \sum_{j=1}^i d(v_j, C)^{\alpha}$. Thus, we want to bound $\left| \frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)} \right|$, and we can use Lemma 16, which bounds the derivative of $\frac{D_i(\alpha)}{D_n(\alpha)}$ by $\log R$, to show $\left| \frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)} \right| \leq \epsilon \log R$.

Therefore, we have

$$\begin{aligned} \sum_{i=1}^n (I_i \setminus I'_i) \cup (I'_i \setminus I_i) &\leq \sum_{i=1}^n \left| \frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)} \right| \\ &\leq \sum_{i=1}^n \epsilon \cdot \log R \\ &\leq \epsilon n \log R \end{aligned}$$

Therefore, assuming d^{α} -sampling and $d^{\alpha+\epsilon}$ -sampling have chosen the same centers so far, the probability that they choose different centers in round t is $\leq \epsilon n \log R$. Over all rounds, the probability the outputted set of centers is not identical, is $\leq \epsilon nk \log R$.

Now we will show that $P_{\vec{Z}}(\text{seed}_{\alpha}(\mathcal{V}, \vec{Z}) \neq \text{seed}_{\alpha+\epsilon}(\mathcal{V}, \vec{Z})) \leq 2nk \log n \log(\frac{\alpha+\epsilon}{\alpha})$. We will bound $\left| \frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)} \right|$ a different way, again using Lemma 16. We have that

$$\left. \frac{D_j(\alpha)}{D_n(\alpha)} \right|_{\alpha_{\ell}}^{\alpha_{\ell+1}} \leq 2 \log n \int_{\alpha_{\ell}}^{\alpha_{\ell+1}} \frac{1}{\alpha} d\alpha \leq 2 \log n (\log \alpha) \Big|_{\alpha_{\ell}}^{\alpha_{\ell+1}} = 2 \log n (\log \alpha_{\ell+1} - \log \alpha_{\ell}).$$

Using this inequality and following the same steps as above, we have

$$\sum_{i=1}^n (I_i \setminus I'_i) \cup (I'_i \setminus I_i) \leq \sum_{i=1}^n \left| \frac{D_i(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha+\epsilon)}{D_n(\alpha+\epsilon)} \right|$$

$$\begin{aligned}
&\leq \sum_{i=1}^n 2 \log n (\log \alpha_{\ell+1} - \log \alpha_{\ell}) \\
&\leq 2n \log n \log \left(\frac{\alpha + \epsilon}{\alpha} \right)
\end{aligned}$$

Over all rounds, the probability the outputted set of centers is not identical, is $\leq nk \log n \log \left(\frac{\alpha + \epsilon}{\alpha} \right)$. This completes the proof. \square

In order to analyze the runtime of Algorithm 2, we consider the *execution tree* of d^α -sampling run on a clustering instance \mathcal{V} with randomness \vec{Z} . This is a tree where each node is labeled by a state (i.e., a sequence C of up to k centers chosen so far by the algorithm) and the interval A of α values that would result in the algorithm choosing this sequence of centers. The children of a node correspond to the states that are reachable in a single step (i.e., choosing the next center) for some value of $\alpha \in A$. The tree has depth k , and there is one leaf for each possible sequence of k centers that d^α -sampling will output when run on \mathcal{V} with randomness \vec{Z} . Our algorithm enumerates these leaves up to an error ϵ in the α values, by performing a depth-first traversal of the tree.

Theorem 8. *Let \mathcal{D} be a distribution over clustering instances with k clusters and at most n points and let $\mathcal{S} = \{(\mathcal{V}^{(i)}, \vec{Z}^{(i)})\}_{i=1}^m$ be an i.i.d. sample from $\mathcal{D} \times \text{Uniform}([0, 1]^k)$. Fix any $\epsilon > 0$, $\delta > 0$, a parameter $\beta \in [1, \infty]$, and a parameter interval $[\alpha_\ell, \alpha_h]$, and run Algorithm 2 on each sample instance and collect the breakpoints (boundaries between the intervals A_i). Let $\bar{\alpha}$ be the lowest cost breakpoint across all instances. Then the following statements hold:*

1. *If $\alpha_\ell > 0$ and the sample is of size $m = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\log\left(\frac{nk}{\delta}\right) + \log\left(\log\left(\frac{\alpha_h}{\alpha_\ell}\right)\right)\right)\right)$, then with probability at least $1 - \delta$ we have $|\text{clus}_{\bar{\alpha}, \beta}(\mathcal{S}) - \min_{\alpha_\ell \leq \alpha \leq \alpha_h} \text{clus}_{\alpha, \beta}(\mathcal{S})| < \epsilon$ and the total running time of finding the best breakpoint is $O\left(mn^2k^2 \log\left(\frac{\alpha_\ell}{\alpha_h}\right) \log\left(nH \log\left(\frac{\alpha_\ell}{\alpha_h}\right) / \epsilon\right)\right)$.*
2. *If $\alpha_\ell = 0$ but the maximum ratio of any non-zero distances for all instances in the support of \mathcal{D} is bounded by R and the sample is of size $m = O\left(\left(\frac{H}{\epsilon}\right)^2 \left(\log\left(\frac{nk}{\delta}\right) + \log\left(\log(\alpha_h \log(R))\right)\right)\right)$, then with probability at least $1 - \delta$ we have $|\text{clus}_{\bar{\alpha}, \beta}(\mathcal{S}) - \min_{0 \leq \alpha \leq \alpha_h} \text{clus}_{\alpha, \beta}(\mathcal{S})| < \epsilon$ and the total running time of finding the best breakpoint is $O\left(mn^2k^2 (\log(\alpha_h \log(R))) \log\left(nH \log(\log(\alpha_h \log(R))) / \epsilon\right)\right)$.*

Proof. We argue that one of the breakpoints outputted by Algorithm 2 on the sample is approximately optimal over all $\alpha \in [\alpha_\ell, \alpha_h]$. Formally, denote $\hat{\alpha}$ as the value with the lowest empirical cost over the sample, and $\bar{\alpha}$ as the value with the lowest empirical cost over the sample, among the set of breakpoints returned by the algorithm. We define α^* as the value with the minimum true cost over the distribution. We also claim that for all breakpoints α , there exists a breakpoint $\hat{\alpha}$ outputted by Algorithm 2 such that $|\alpha - \hat{\alpha}| < \frac{\epsilon}{5n^2k \log n \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}$. We will prove this claim at the end of the proof. Assuming the claim is correct, we denote α' as a breakpoint outputted by the algorithm such that $|\hat{\alpha} - \alpha'| < \frac{\epsilon}{5n^2k \log n \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}$.

For the rest of the proof, denote $\mathbb{E}_{\mathcal{V} \sim \mathcal{D}} [\text{clus}_{\alpha, \beta}(\mathcal{V})] = \text{true}(\alpha)$ and $\frac{1}{m} \sum_{i=1}^m \text{clus}_{\alpha, \beta}(\mathcal{V}^{(i)}, \vec{Z}^{(i)}) = \text{sample}(\alpha)$ since beta, the distribution, and the sample are all fixed.

By construction, we have $\text{sample}(\hat{\alpha}) \leq \text{sample}(\alpha^*)$ and $\text{sample}(\bar{\alpha}) \leq \text{sample}(\alpha')$. By Theorem 6, with probability $> 1 - \delta$, for all α (in particular, for $\bar{\alpha}$, $\hat{\alpha}$, α^* , and α'), we have $|\text{sample}(\alpha) - \text{true}(\alpha)| < \epsilon/5$. Finally, by Lemma 7, we have

$$|\hat{\alpha} - \alpha'| < \frac{\epsilon}{5n^2k \log n \log\left(\frac{\alpha_h}{\alpha_\ell}\right)} \implies |\text{true}(\hat{\alpha}) - \text{true}(\alpha')| < \epsilon/5.$$

Using these five inequalities for α' , $\hat{\alpha}$, $\bar{\alpha}$, and α^* , we can show the desired outcome as follows.

$$\begin{aligned}
\text{true}(\bar{\alpha}) - \text{true}(\alpha^*) &\leq (\text{true}(\bar{\alpha}) - \text{sample}(\bar{\alpha})) + \text{sample}(\bar{\alpha}) - (\text{true}(\alpha^*) - \text{sample}(\alpha^*)) - \text{sample}(\alpha^*) \\
&\leq \epsilon/5 + \text{sample}(\alpha') + \epsilon/5 - \text{sample}(\hat{\alpha}) \\
&\leq (\text{sample}(\alpha') - \text{true}(\alpha')) + (\text{true}(\alpha') - \text{true}(\hat{\alpha})) + (\text{true}(\hat{\alpha}) - \text{sample}(\hat{\alpha})) + \frac{2\epsilon}{5} \\
&\leq \epsilon.
\end{aligned}$$

Now we will prove the claim that for all breakpoints α , there exists a breakpoint $\hat{\alpha}$ outputted by Algorithm 2 such that $|\alpha - \hat{\alpha}| < \frac{\epsilon}{5n^2k \log n \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}$. Denote $\epsilon' = \frac{\epsilon}{5n^2k \log n \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}$. We give an inductive proof. Recall that the algorithm may only find the values of breakpoints up to additive error ϵ' , since the true breakpoints may be irrational and/or transcendental. Let \hat{T}_t denote the execution tree of the algorithm after round t , and let T_t denote the *true* execution tree on the sample. That is, T_t is the execution tree as defined earlier this section, \hat{T}_t is the execution tree with the algorithm's ϵ' imprecision on the values of alpha. Note that if a node in T_t represents an α -interval of size smaller than ϵ' , it is possible that \hat{T}_t does not contain the node. Furthermore, \hat{T}_t might contain spurious nodes with alpha-intervals of size smaller than ϵ' .

Our inductive hypothesis has two parts. The first part is that for each breakpoint α in T_t , there exists a breakpoint $h(\alpha)$ in \hat{T}_t such that $|\alpha - h(\alpha)| < \epsilon'$. For the second part of our inductive hypothesis, we define $B_t = \bigcup_{\alpha \text{ breakpoint}} ([\alpha, h(\alpha)] \cup [h(\alpha), \alpha])$, the set of “bad” intervals. Note that for each α , one of $[\alpha, h(\alpha)]$ and $[h(\alpha), \alpha]$ is empty. Then define $G_t = [\alpha_\ell, \alpha_h] \setminus B_t$, the set of “good” intervals. The second part of our inductive hypothesis is that the set of centers for α in T_t is the same as in \hat{T}_t , as long as $\alpha \in G_t$. That is, if we look at the leaf in T_t and the leaf in \hat{T}_t whose alpha-intervals contain α , the set of centers for both leaves are identical. Now we will prove the inductive hypothesis is true for round $t+1$, assuming it holds for round t . Given T_t and \hat{T}_t , consider a breakpoint α from T_{t+1} introduced in round $t+1$.

Case 1: $\alpha \in G_t$. Then the algorithm will recognize there exists a breakpoint, and use binary search to output a value $h(\alpha)$ such that $|\alpha - h(\alpha)| < \epsilon'$. The interval $[\alpha, h(\alpha)] \cup [h(\alpha), \alpha]$ is added to B_{t+1} , but the good intervals to the left and right of this interval still have the correct centers.

Case 2: $\alpha \in B_t$. Then there exists an interval $[\alpha', h(\alpha')] \cup [h(\alpha'), \alpha]$ containing α . By assumption, this interval is size $< \epsilon'$, therefore, we set $h(\alpha') = h(\alpha)$, so there is a breakpoint within ϵ' of α .

Therefore, for each breakpoint α in T_{t+1} , there exists a breakpoint $\hat{\alpha}$ in \hat{T}_{t+1} such that $|\alpha - \hat{\alpha}| < \epsilon'$. Furthermore, for all $\alpha \in G_{t+1}$, the set of centers for α in T_{t+1} is the same as in \hat{T}_{t+1} . This concludes the inductive proof.

Now we analyze the runtime of Algorithm 2. Let (C, A) be any node in the algorithm, with centers C and alpha interval $A = [\alpha_\ell, \alpha_h]$. Sorting the points in \mathcal{V} according to their distance to C has complexity $O(n \log n)$. Finding the points sampled by d^α -sampling with α set to α_ℓ and α_h costs $O(n)$ time. Finally, computing the alpha interval A_i for each child node of (C, A) costs $O\left(n \log \frac{nH \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}{\epsilon}\right)$ time, since we need to perform $\log \frac{nkH \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}{\epsilon}$ iterations of binary search on $\alpha \mapsto \frac{D_i(\alpha)}{D_n(\alpha)}$ and each evaluation of the function costs $O(n)$ time. We charge this $O\left(n \log \frac{nH \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}{\epsilon}\right)$ time to the corresponding child node. If there are N nodes in the execution tree, summing this cost over all nodes gives a total running time of $O\left(N \cdot n \log \frac{nH \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}{\epsilon}\right)$. If we let $\#I$ denote the total number of α -intervals for \mathcal{V} , then each layer of the execution tree has at most $\#I$ nodes, and the depth is k , giving a total running time of $O\left(\#I \cdot kn \log \frac{nH \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}{\epsilon}\right)$.

For the case when $\alpha_\ell > 0$, Theorem 4 guarantees that we have $\mathbb{E}[\#I] \leq 8nk \log(n) \log\left(\frac{\alpha_h}{\alpha_\ell}\right)$.

Therefore, the expected runtime of Algorithm 2 is $O\left(n^2k^2 \log\left(\frac{\alpha_h}{\alpha_\ell}\right) \left(\log\frac{nH \log\left(\frac{\alpha_h}{\alpha_\ell}\right)}{\epsilon}\right)\right)$. The case where $\alpha_\ell = 0$ but there is a bound on the maximum ratio between any nonzero pair of distances is similar, except we use the appropriate statements from Theorem 4 and Lemma 7. \square

Since we showed that d^α -sampling is Lipschitz as a function of α in Lemma 7, it is also possible to find the best α parameter with sub-optimality at most ϵ by finding the best point from a discretization of $[0, \alpha_h]$ with step-size $s = \epsilon/(Hn^2k \log R)$. The running time of this algorithm is $O(n^3k^2H \log R/\epsilon)$, which is significantly slower than the efficient algorithm presented in this section. Intuitively, Algorithm 2 is able to binary search to find each breakpoint in time $O(\log\frac{nkH \log R}{\epsilon})$, whereas a discretization-based algorithm must check all values of alpha uniformly, so the runtime of the discretization-based algorithm increases by a multiplicative factor of $O\left(\frac{nH \log R}{\epsilon} \cdot \left(\log\frac{nH \log R}{\epsilon}\right)^{-1}\right)$.

Generalized families Theorems 4 and 8 are not just true specifically for (α, β) -Lloyds++; they can be made much more general. In this section, we show these theorems are true for any family of randomized initialization procedures which satisfies a few simple properties. First we formally define a parameterized family of randomized initialization procedures.

Definition 9. *Given a clustering instance $\mathcal{C} = (V, d)$ and a function $p : V \times 2^V \mapsto [0, 1]$, a p -randomized initialization method is an iterative method such that \mathcal{C} is initialized as empty and one new center is added in each round, and in round t , each point v is chosen as a center with probability $p(v, C)$. Note that for all $C \subseteq V$ such that $|C| \leq k$, we must have $\sum_{v \in V} p(v, C) = 1$. An α -parameterized family is a set of p -randomized initialization methods $p_\alpha : V \times 2^V \mapsto [0, 1]$.*

We will show how to prove general upper bounds on α -parameterized families of p -randomized initialization methods as long as the functions p_α satisfy a few simple properties.

Just as with (α, β) -Lloyds++, we specify precisely how we execute the randomized steps of the algorithm. We assume the algorithm draws a vector $\vec{Z} = \{z_1, \dots, z_k\}$ from $[0, 1]^k$ uniformly at random. Then in round t , the algorithm partitions $[0, 1]$ into n intervals, where there is an interval I_{v_i} for each v_i with size equal to the probability $p_\alpha(v_i, C)$ of choosing v_i in round t where C is the set of centers at the start of round t . Then the algorithm chooses the point v_i as a center, where $z_t \in I_{v_i}$.

Now we will show how to upper bound the number of discontinuities of the output as a function of α , which will lead to sample-efficient and computationally-efficient meta-algorithms. We start with a few definitions. Without loss of generality, we assume that in each round we rename the points so that they satisfy $p_{\alpha_\ell}(v_1, C) \geq \dots \geq p_{\alpha_\ell}(v_n, C)$. For a given C and v_i , we define the partial sums $S_{i,C}(\alpha) = \sum_{j=1}^i p_\alpha(v_j, C)$. Let the set of permissible values of α be the interval $[\alpha_\ell, \alpha_h]$. Let $\text{seed}_\alpha(\mathcal{V}, \vec{Z}, p)$ denote the sequence of centers returned with randomness $\vec{Z} \in [0, 1]^k$ and parameter α . Let $D_p = \max_{i,C,v,\alpha} \left(\frac{\partial S_{i,C}(\alpha)}{\partial \alpha}\right)$.

Theorem 10. *Given an α -parameterized family such that (1) for all $1 \leq i \leq k$ and $C \subseteq V$ such that $|C| \leq k$, each $S_{i,C}(\alpha)$ is monotone increasing and continuous as a function of α , and (2) for all $1 \leq i \leq j \leq n$ and $\alpha \in (\alpha_\ell, \alpha_h)$, $S_{i,C}(\alpha) \leq S_{j,C}(\alpha)$, then the expected number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z}, p)$ as a function of α is $O(nkD_p(\alpha_h - \alpha_\ell))$.*

Note this theorem is a generalization of Theorem 4, since (α, β) -Lloyds++ is an α -parameterized property with the properties (due to Lemma 3) and $D_p = 4 \log n$. We give the proof of Theorem 10

in Appendix B, since it is similar to the proof of Theorem 4. Intuitively, a key part of the argument is bounding the expected number of discontinuities in an arbitrary interval $[\alpha_\ell, \alpha_{\ell+1}]$ for a current set of centers C . If the algorithm chooses v_j as a center at α_ℓ and v_i at α_ℓ , then the only possible centers are v_i, \dots, v_j . We show this key fact follows for any partial sum of probability functions as long as they are monotone, continuous, and non-crossing. Furthermore, $j - i$ is bounded by the maximum derivative of the partial sums over $[\alpha_\ell, \alpha_{\ell+1}]$, so the final upper bound scales with D_p . We can also achieve a generalized version of Theorem 8.

Theorem 11. *Given parameters $0 \leq \alpha_\ell < \alpha_h$, $\epsilon > 0$, a sample \mathcal{S} of size*

$$m = O\left(\left(\frac{H}{\epsilon}\right)^2 \log\left(\frac{\alpha_h n D_p}{\delta}\right)\right)$$

from $(\mathcal{D} \times [0, 1]^k)^m$, and an α -parameterized family satisfying properties (1) and (2) from Theorem 10, run Algorithm 2 on each sample and collect all breakpoints (i.e., boundaries of the intervals A_i). With probability at least $1 - \delta$, the breakpoint $\bar{\alpha}$ with lowest empirical cost satisfies $|\text{clus}_{\bar{\alpha}, \beta}(\mathcal{S}) - \min_{0 \leq \alpha \leq \alpha_h} \text{clus}_{\alpha, \beta}(\mathcal{S})| < \epsilon$. The total running time to find the best breakpoint is $O(mn^2 k^2 \alpha_h D_p \log\left(\frac{nH}{\epsilon}\right) \log n)$.

5 Experiments

In this section, we empirically evaluate the effect of the α and β parameters on clustering cost for real-world and synthetic clustering domains. We find that the optimal α and β parameters vary significantly from domain to domain. We also find that the number of possible initial centers chosen by d^α -sampling scales linearly with n and k , suggesting our Theorem 4 is tight up to logarithmic factors. Finally, we show the empirical distribution of α -interval boundaries.

Experiment Setup. Our experiments evaluate the (α, β) -Lloyds++ family of algorithms on several distributions over clustering instances. Our clustering instance distributions are derived from classification datasets. For each classification dataset, we sample a clustering instance by choosing a random subset of k labels, sampling N examples belonging to each of the k chosen labels. The clustering instance then consists of the kN points, and the target clustering is given by the ground-truth labels. This sampling distribution covers many related clustering tasks (i.e., clustering different subsets of the same labels). We evaluate clustering performance in terms of the Hamming distance to the optimal clustering, or the fraction of points assigned to different clusters by the algorithm and the target clustering. Formally, the Hamming distance between the outputted clustering $\{C_1, \dots, C_k\}$ and the optimal clustering $\{C_1^*, \dots, C_k^*\}$ is measured by $\min_\sigma \frac{1}{n} \sum_{i=1}^k C_i \setminus C_{\sigma(i)}^*$, where the minimum is taken over all permutations σ of the cluster indices. In all cases, we limit the algorithm to performing 3 iterations of β -Lloyds. We use the following datasets:

MNIST: We use the raw pixel representations of the subset of MNIST [31]. For MNIST we set $k = 5$, $N = 100$, so each instance consists of $n = 500$ points.

CIFAR-10: The CIFAR-10 dataset [28] is an image dataset with 10 classes. Following [29] we include 50 randomly rotated and cropped copies of each example. We extract the features from the Google Inception network [41] using layer in4d. For CIFAR10 we set $k = 5$, $N = 100$, so each instance consists of $n = 500$ points.

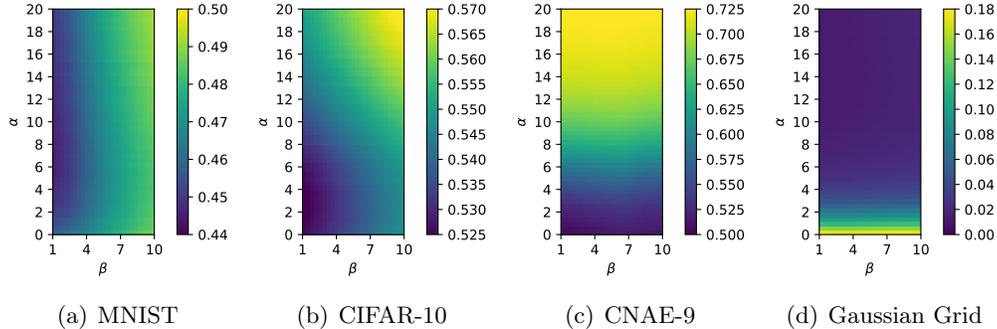


Figure 4: Average Hamming error of (α, β) -Lloyds++ as a function of α and β .

CNAE-9: The CNAE-9 [16] dataset consists of 1080 documents describing Brazilian companies categorized into 9 categories. Each example is represented by an 875-dimensional vector, where each entry is the frequency of a specific word. For CNAE-9 we set $k = 4$ and $N = 100$, so each clustering instance has $n = 400$ points.

Gaussian Grid: We also use a synthetic 2-dimensional clustering instance where points are sampled from a mixture of 9 standard Gaussians arranged in a 3×3 grid with a grid stride of 5. For the Gaussian Grid dataset, we set $k = 4$ and $N = 120$, so each clustering instance has $n = 500$ points.

Parameter Study. Our first experiment explores the effect of the α and β parameters. For each dataset, we sampled $m = 50,000$ sample clustering instances and run (α, β) -Lloyds++ for all combinations of 50 values of α evenly spaced in $[0, 20]$ and 25 values of β evenly spaced in $[1, 10]$. Figure 4 shows the average Hamming error on each dataset as a function of α and β . The optimal parameters vary significantly across the datasets. On MNIST and the Gaussian grid, it is best to set α to be large, while on the remaining datasets the optimal value is low. Neither the k -means++ algorithm nor farthest-first traversal have good performance across all datasets. On the Gaussian grid example, k -means++ has Hamming error 6.8% while the best parameter setting only has Hamming error 1.3%.

Next, we use Algorithm 2 to tune α without discretization. In these experiments we set $\beta = 2$ and modify step 6 of Algorithm 1 to compute the mean of each cluster, rather than the point in the dataset minimizing the sum of squared distances to points in that cluster (as is usually done in Lloyd’s method). This modification improves running time and also the Hamming cost of the resulting clusterings. For each dataset, we sample $m = 50,000$ sample clustering instances and divide them evenly into testing and training sets (for MNIST we set $m = 250,000$ instead). We plot the average Hamming cost as a function of α on both the training and testing sets. The optimal value of α varies between the different datasets, showing that tuning the parameters leads to improved performance. Interestingly, for MNIST the value of α with lowest Hamming error is $\alpha = 4.1$ which does not correspond to any standard algorithm. Moreover, in each case the difference between the training and testing error plots is small, supporting our generalization claims.

Number of α -Intervals. Next we report the number of α -intervals in the above experiments. On average, MNIST had 826.1 intervals per instance, CIFAR-10 had 994.5 intervals per instance, CNAE-9 had 855.9 intervals per instance, and the Gaussian grid had 953.4 intervals per instance.

In Figure 6 we evaluate how the number of α intervals grows with the clustering instance size n . For $n \in \{50, 100, 150, \dots, 1000\}$, we modify the above distributions by setting $N = n/k$ and plot

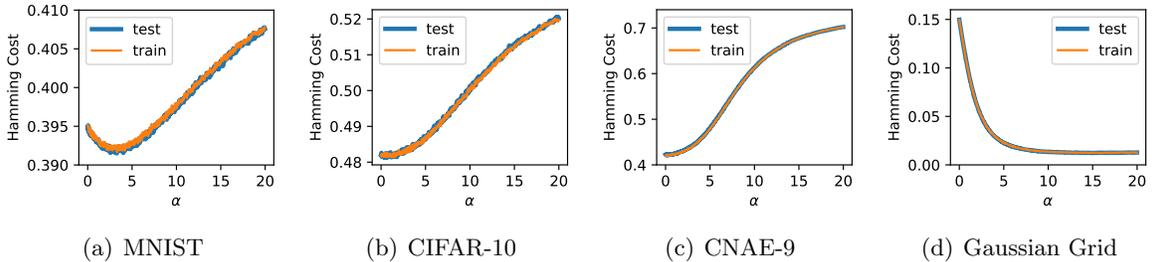


Figure 5: Average Hamming error of (α, β) -Lloyds++ as a function of α for $\beta = 2$.

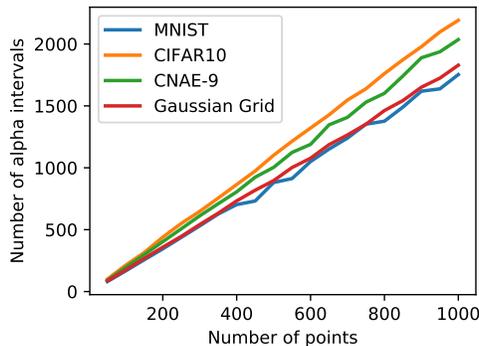


Figure 6: The number of α -intervals as a function of clustering instance size.

the average number of α -intervals on $m = 5000$ samples. For all four datasets, the average number of intervals grows nearly linearly with n .

Distribution of α -decision Points. Finally, for each dataset, Figure 7 shows a histogram of the distribution of the α -interval boundaries.

6 Conclusion

We define an infinite family of algorithms generalizing Lloyd’s method, with one parameter controlling the initialization procedure, and another parameter controlling the local search procedure. This family of algorithms includes the celebrated k -means++ algorithm, as well as the classic

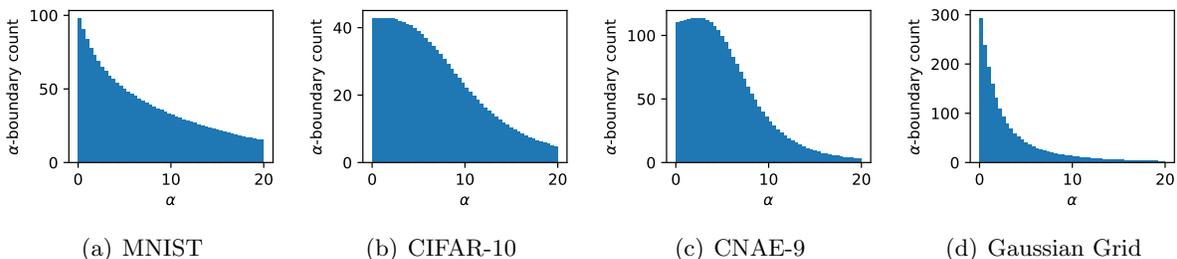


Figure 7: Distribution of α -decision points.

farthest-first traversal algorithm. We provide a sample efficient and computationally efficient algorithm to learn a near-optimal parameter over an unknown distribution of clustering instances, by developing techniques to bound the expected number of discontinuities in the cost as a function of the parameter. We give a thorough empirical analysis, showing that the value of the optimal parameters transfer to related clustering instances. We show the optimal parameters vary among different application domains, and the optimal parameters often significantly improve the error compared to existing algorithms such as k -means++ and farthest-first traversal.

7 Acknowledgments

This work was supported in part by NSF grants CCF-1535967, IIS-1618714, an Amazon Research Award, a Microsoft Research Faculty Fellowship, a National Defense Science & Engineering Graduate (NDSEG) fellowship, and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

References

- [1] Margareta Ackerman, Shai Ben-David, and David Loker. Towards property-based classification of clustering paradigms. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 10–18, 2010.
- [2] Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k -means and euclidean k -median by primal-dual algorithms. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, 2017.
- [3] Kohei Arai and Ali Ridho Barakbah. Hierarchical k -means: an algorithm for centroids initialization for k -means. *Reports of the Faculty of Science and Engineering*, 36(1):25–31, 2007.
- [4] David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k -means method. *Journal of the ACM (JACM)*, 58(5):19, 2011.
- [5] David Arthur and Sergei Vassilvitskii. How slow is the k -means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153. ACM, 2006.
- [6] David Arthur and Sergei Vassilvitskii. k -means++: The advantages of careful seeding. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.
- [7] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k -median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
- [8] Hassan Ashtiani and Shai Ben-David. Representation learning for clustering: a statistical framework. In *Proceedings of the Annual Conference on Uncertainty in Artificial Intelligence*, pages 82–91, 2015.
- [9] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 213–274, 2017.
- [10] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

- [11] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Annual Symposium on Discrete Algorithms (SODA)*, pages 737–756, 2015.
- [12] Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 1–10, 1999.
- [13] Michael B Cohen, Yin Tat Lee, Gary Miller, Jakub Pachocki, and Aaron Sidford. Geometric median in nearly linear time. In *Proceedings of the Annual Symposium on Theory of Computing (STOC)*, pages 9–21. ACM, 2016.
- [14] Sanjoy Dasgupta and Philip M Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555–569, 2005.
- [15] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society*, pages 1–38, 1977.
- [16] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.
- [17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.
- [18] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.
- [19] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
- [20] Sarel Har-Peled and Bardia Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.
- [21] Richard E Higgs, Kerry G Bemis, Ian A Watson, and James H Wikel. Experimental designs for selecting molecules from large chemical databases. *Journal of chemical information and computer sciences*, 37(5):861–870, 1997.
- [22] Wenhao Jiang and Fu-lai Chung. Transfer spectral clustering. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, pages 789–803, 2012.
- [23] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.
- [24] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.
- [25] Jon M Kleinberg. An impossibility theorem for clustering. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 463–470, 2003.
- [26] Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. An online hierarchical algorithm for extreme clustering. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

- [27] Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001.
- [28] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
- [29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.
- [30] Stuart Lloyd. Least squares quantization in pcm. *transactions on information theory*, 28(2):129–137, 1982.
- [31] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, pages 301–320, 2007.
- [32] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [33] Pascal Massart. Some applications of concentration inequalities to statistics. In *Annales-Faculte des Sciences Toulouse Mathematiques*, volume 9, pages 245–303. Université Paul Sabatier, 2000.
- [34] Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1):7–12, 1960.
- [35] Rafail Ostrovsky, Yuval Rabani, Leonard J Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM (JACM)*, 59(6):28, 2012.
- [36] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the Annual Conference on Knowledge Discovery and Data Mining (KDD)*, pages 277–281, 1999.
- [37] José M Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.
- [38] Kim D Pruitt, Tatiana Tatusova, Garth R Brown, and Donna R Maglott. Nebi reference sequences (refseq): current status, new features and genome annotation policy. *Nucleic acids research*, 40(D1):D130–D135, 2011.
- [39] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 759–766, 2007.
- [40] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2110–2118, 2016.
- [41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

- [42] Timo Tossavainen. On the zeros of finite sums of exponential functions. *Australian Mathematical Society Gazette*, 33(1):47–50, 2006.
- [43] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.
- [44] Andrea Vattani. K-means requires exponentially many iterations even in the plane. *Discrete & Computational Geometry*, 45(4):596–616, 2011.
- [45] Qiang Yang, Yuqiang Chen, Gui-Rong Xue, Wenyuan Dai, and Yong Yu. Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Conference on Natural Language Processing*, pages 1–9, 2009.

Table 1: Notation table

Symbol	Description
$\mathcal{V} = (V, d, k)$	Clustering instance of points V with distance metric d , number of clusters k
n	$n = V $, the size of the point set
k	k =number of clusters in \mathcal{V}
$\mathcal{C}^* = \{C_1^*, \dots, C_k^*\}$	Optimal clusters according to an objective such as k -means
(α, β) -Lloyds++	Clustering algorithm with parameters $\alpha \in [0, \infty)$ and $\beta \in [1, \infty)$
\vec{Z}	$\vec{Z} = \{z_1, \dots, z_k\} \in [0, 1]^k$ is the random seed for the (α, β) -Lloyds++ algorithm
$\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})$	Cost of the clustering outputted by (α, β) -Lloyds++ with randomness $\vec{Z} \in [0, 1]^k$
$\text{clus}_{\alpha, \beta}(\mathcal{V})$	$\text{clus}_{\alpha, \beta}(\mathcal{V}) = \mathbb{E}_{\vec{Z} \sim [0, 1]^k} [\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})]$
$\text{seed}_{\alpha}(\mathcal{V}, \vec{Z})$	the output centers of phase 1 of Algorithm 1 run on \mathcal{V} with vector \vec{Z}
$\text{lloyds}_{\beta}(\mathcal{V}, C, T)$	cost of the outputted clustering from phase 2 of Algorithm 1 on instance \mathcal{V} with initial centers C , and a maximum of T iterations.
H	H = maximum loss for any clustering instance \mathcal{V}
s	s = the minimum ratio $\frac{d_1}{d_2}$ between two distances $d_1 > d_2$ in the point set
d_j	$d_j = d(c, j)$ only when the center c is clear from context
R	$R = \max_{u, v, x, y \in V} \frac{d(u, v)}{d(x, y)}$ (the ratio of the largest to smallest distances)
$D_i(\alpha)$	$D_i(\alpha) = \sum_{j=1}^i d_j^\alpha$
$\#I$	The number of breakpoints of (α, β) -Lloyds++ (the number of discontinuities in $\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})$)
$\#I_{t, \ell}$	The number of breakpoints in round t and interval $[\alpha_\ell, \alpha_{\ell+1}]$ (the number of times the choice of t 'th center changes as we vary α along $[\alpha_\ell, \alpha_{\ell+1}]$)
$E_{t, j}$	the event that $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} < z_t < \frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$ (we achieve the maximum number of breakpoints given v_i is the t 'th center)
$E'_{t, j}$	the event that $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} < z_t < \frac{D_{j+1}(\alpha_\ell)}{D_n(\alpha_\ell)}$ (we do not achieve the max number of breakpoints given v_i is the t 'th center)

A Table of Notation

B Details from Section 4

In this section, we give details and proofs from Section 4.

Lemma 1. *For any clustering instance $\mathcal{V} = (V, d, k)$ and $\delta > 0$, if $\alpha > \log(\frac{nk}{\delta}) / \log s$, where s denotes the minimum ratio d_1/d_2 between two distances $d_1 > d_2$ in the point set, then $P_{\vec{Z}}(\text{seed}_{\alpha}(\mathcal{V}, \vec{Z}) = \text{seed}_{\infty}(\mathcal{V}, \vec{Z})) \geq 1 - \delta$.*

Proof. Given such a clustering instance $\mathcal{V} = (V, d, k)$ and α , first we note that farthest-first traversal (i.e. d^∞ -sampling) and d^α -sampling both start by picking a center uniformly at random from V . Assume both algorithms have chosen initial center v_1 , and let $C = \{v_1\}$ denote the set of current centers. In rounds 2 to n , farthest-first traversal deterministically chooses the center u

which maximizes $d_{\min}(u, C)$ (breaking ties uniformly at random). We will show that with high probability, in every round, d^α -sampling will also choose the center maximizing $d_{\min}(u, C)$ or break ties at random. In round t , let $d_t = \max_{u \in V} d_{\min}(u, C)$ (assuming C are the first $t-1$ centers chosen by farthest-first traversal). Let d'_t denote the largest distance smaller than d_t , so by assumption, $d_t > s \cdot d'_t$. Assume there are x points whose minimum distance to C is d_t . Then the probability that d^α -sampling will fail to choose one of these points is at most

$$\begin{aligned} \frac{(n-x)d_t^\alpha}{(n-x)d_t^\alpha + x(sd'_t)^\alpha} &\leq \frac{n-x}{n-x+x \cdot s^\alpha} \\ &\leq \frac{n}{s^\alpha} \end{aligned}$$

Over all k rounds of the algorithm, the probability that d^α -sampling will deviate from farthest-first traversal (assuming they start with the same first choice of a center and break ties at random in the same way) is at most $\frac{nk}{s^\alpha}$, and if we set this probability smaller than δ and solve for α , we obtain

$$\alpha > \frac{\log\left(\frac{nk}{\delta}\right)}{\log s}.$$

□

Now we define a common stability assumption called separability [26, 38], which states that there exists a value r such that all points in the same cluster have distance less than r and all points in different clusters have distance greater than r .

Definition 12. *A clustering instance satisfies $(1+c)$ -separation if $(1+c) \max_{i|u,v \in C_i} d(u,v) < \min_{j \neq j' | u \in C_j, v \in C_{j'}} d(u,v)$.*

Now we show that under $(1+c)$ -separation, d^α -sampling will give the same output as farthest-first traversal with high probability if $\alpha > \log n$, even for $c = .1$.

Lemma 13. *Given a clustering instance \mathcal{V} satisfying $(1+c)$ -separation, and $0 < \delta$, then if $\alpha > \frac{1}{c} \cdot (\log n + \log \frac{1}{\delta})$, with probability $> 1 - \delta$, d^α -sampling with Lloyd's algorithm will output the optimal clustering.*

Proof. Given \mathcal{V} satisfying $(1+c)$ -separation, we know there exists a value r such that for all i , for all $u, v \in C_i$, $d(u, v) < r$, and for all $u \in C_j, v \in C_{j' \neq j}$, $d(u, v) > (1+c)r$. WLOG, let $r = 1$. Consider round t of d^α -sampling and assume that each center v in the current set of centers C is from a unique cluster in the optimal solution. Now we will bound the probability that the center chosen in round t is not from a unique cluster in the optimal solution. Given a point u from a cluster already represented in C , there must exist $v \in C$ such that $d(u, v) < 1$, so $d_{\min}(u, C) < 1$. Given a point u from a new cluster, it must be the case that $d_{\min}(u, C) > (1+c)$. The total number of points in represented clusters is $< n$. Then the probability we pick a point from an already represented cluster is $\leq \frac{tn}{tn+c^\alpha}$. If we set $\frac{tn}{tn+c^\alpha} \leq \frac{\delta}{k}$ and solve for α , we obtain $\alpha > \frac{\log n + \log \frac{1}{\delta}}{\log c} \leq \frac{1}{c} \cdot (\log n + \log \frac{1}{\delta})$. Since this is true for an arbitrary round t , and there are k rounds in total, we may union bound over all rounds to show the probability d^α -sampling outputting one center per optimal clustering is $> 1 - \delta$. Then, using $(1+c)$ -separation, the Voronoi tiling of these centers must be the optimal clustering, so Lloyd's algorithm converges to the optimal solution in one step. □

Next, we give the formal proof of Theorem 2.

Theorem 2 (restated). For $\alpha^* \in [.01, \infty) \cup \{\infty\}$ and $\beta^* \in [1, \infty) \cup \{\infty\}$, there exists a clustering instance \mathcal{V} whose target clustering is the optimal ℓ_{β^*} clustering, such that $\text{clus}_{\alpha^*, \beta^*}(\mathcal{V}) < \text{clus}_{\alpha, \beta}(\mathcal{V})$ for all $(\alpha, \beta) \neq (\alpha^*, \beta^*)$.

Proof. Consider $\alpha^*, \beta^* \in [0, \infty) \cup \{\infty\}$. The clustering instance consists of 6 clusters, C_1, \dots, C_6 . The target clustering will be the optimal ℓ_{β^*} objective. The basic idea of the proof is as follows.

First, we show that for all β and $\alpha \neq \alpha^*$, $\text{clus}_{\alpha^*, \beta}(\mathcal{V}) < \text{clus}_{\alpha, \beta}(\mathcal{V})$. We use clusters C_1, \dots, C_4 to accomplish this. We set up the distances so that d^{α^*} sampling is more likely to sample one point per cluster than any other value of α . If the sampling does not sample one point per cluster, then it will fall into a high-error local minima trap that β -Lloyd's method cannot escape, for any value of β . Therefore, d^{α^*} sampling is more effective than any other value of α .

Next, we use clusters C_5 and C_6 to show that if we start with one center in C_5 and one in C_6 , then β^* -Lloyd's method will strictly outperform any other value of β . We accomplish this by adding three choices of centers for C_5 . Running β^* -Lloyd's method will return the correct center, but any other value of β will return suboptimal centers which incur error on C_5 and C_6 . Also, we show that Lloyd's method returns the same centers on C_1, \dots, C_4 , independent of β .

For the first part of the proof, we define three cliques (see Figure 1). The first two cliques are C_1 and C_2 , and the third clique is $C_3 \cup C_4$. C_1 contains w_1 points at distance $x > 1$, and C_2 contains w_2 points at distance $\frac{1}{x}$. We set $w_2 = x^{2\alpha^*} w_1$. The last clique contains w points at distance 1. Since the cliques are very far apart, the first three sampled centers will each be in a different clique, with high probability (for $\alpha > .01$). The probability of sampling a 4th center x_4 in the third clique, for $\alpha = \alpha^* + \delta$ is equal to

$$\frac{w}{w + (x^{\alpha^* + \delta} + x^{\alpha^* - \delta})w_2}.$$

Since $x^{\alpha^* + \delta} + x^{\alpha^* - \delta}$ is minimized when $\delta = 0$, this probability is maximized when $\alpha = \alpha^*$. Now we show that the error will be much larger when x_4 is not in the third clique. We add center c_1 which is distance $x - \epsilon$ to all points in C_1 . We also add centers b_1 and b'_1 which are distance $x - 2\epsilon$ to B_1 and B'_1 such that B_1 and B'_1 form a partition of C_1 . Similarly, we add centers c_2, b_2 , and b'_2 at distance $\frac{1}{x} - \epsilon$ and $\frac{1}{x} - 2\epsilon$ to C_2, B_2 , and B'_2 , respectively, such that B_2 and B'_2 form a partition of C_2 . Finally, we add c_3 and c_4 which are distance $.5$ to C_3 and C_4 , respectively, and we add b_3 which is distance $1 - \epsilon$ to $C_3 \cup C_4$. Then the optimal centers for any β must be c_1, c_2, c_3, c_4 , and this will be the solution of β -Lloyd's method, as long as the sampling procedure returned one point in the first two cliques, and two points in the third clique. If the sampling procedure returns two points in the first clique or second clique, then β -Lloyd's method will return b_1, b'_1, c_2, b_3 or c_1, b_2, b'_2, b_3 , respectively. This will incur error $w/2 + w_1/2$ or $w/2 + w_2/2$, since we set the target clustering to be the optimal ℓ_{β^*} objective which is equal to $\{C_1, C_2, C_3, C_4\}$. Note that we have set up the distances so that Lloyd's method is independent of β . Therefore, the expected error is equal to

$$\frac{w}{w + (x^{\alpha^* + \delta} + x^{\alpha^* - \delta})w_2} \cdot (w/2 + \min(w_1, w_2)/2).$$

This finishes off the first part of the proof. Next, we construct C_5 and C_6 so that β^* -Lloyd's method will return the best solution, assuming the sampling returned one point in C_5 and C_6 . Later we will show how adding these clusters does not affect the previous part of the proof. We again define three cliques. The first clique is size $\frac{2}{3}w_5$ and distance $.1$, the second clique is size $\frac{1}{3}w_5$ and distance $.1$, and the third clique is size w_6 and distance $.1$. The first two cliques are C_5 , and the

second clique is C_6 . The distance between the first two cliques is .2, and the distance between the first two and the third clique is 1000. Now imagine the first two cliques are parallel to each other, and there is a perpendicular bisector which contains possible centers for C_5 . I.e., we will consider possible centers c for C_5 where the ℓ_β cost of c is $\frac{2}{3}w_5 \cdot z^\beta + \frac{1}{3}w_5(.2 - z)^\beta$ for some $0 \leq z \leq .2$. For $\beta \in (0, \infty)$, the β which minimizes the expression must be in $[0, .2]$. We set c_5 corresponding to the z which minimizes the expression for β^* , call it z^* . Therefore, β^* -Lloyd's method will output c_5 . We also set centers b_5 and b'_5 corresponding to $z^* - \epsilon$ and $z^* + \epsilon$. Therefore, any value of β slightly above or below β^* will return a different center. We add a center C_6 at distance $.1 - \epsilon$ to the third clique. This is the only point we add, so it will always be chosen by β -Lloyd's method for all β . Finally, we add two points p_1 and p_2 in between the second and third cliques. We set the distances as follows. $d(c_5, p_1) = d(c_6, p_2) = 500 - \epsilon$, $d(c_5, p_2) = d(c_6, p_1) = 500$, $d(b_5, p_1) = 500 + \epsilon$, and $d(b'_5, p_2) = 500 - 2\epsilon$. Since the weight of these two points are very small compared to the cliques, these points will have no effect on the prior sampling and Lloyd's method analyses. The optimal clustering for the ℓ_{β^*} objective is to add p_1 to C_5 and p_2 to C_6 . However, running β -Lloyd's method for β smaller or larger than β^* will return center b_5 or b'_5 and incur error 1 by mislabeling p_1 or p_2 .

Since all cliques from both parts of the proof are 1000 apart, with high probability, the first 5 sampled points will be in cliques $C_1, C_2, C_3 \cup C_4, C_5$, and C_6 . Since the cliques from the second part of the proof are distance .2, while in the first part they were $> \frac{1}{x}$ apart, we can set the variables x, w_1, w_2, w, w_5, w_6 so that with high probability, the sixth sampled point will not be in C_5 or C_6 . Therefore, the constructions in the second part do not affect the first part. This concludes the proof. \square

Now we upper bound the number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$. Recall that $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ denotes the outputted centers from phase 1 of Algorithm 1 on instance \mathcal{V} with randomness \vec{Z} . For the first phase, our main structural result is to show that for a given clustering instance and value of β , with high probability over the randomness in Algorithm 1, the number of discontinuities of the cost function $\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})$ as we vary $\alpha \in [0, \alpha_h]$ is $O(nk(\log n)\alpha_h)$. Our analysis crucially harnesses the randomness in the algorithm to achieve this bound. In contrast, a combinatorial approach would only achieve a bound of $n^{O(k)}$, which is the total number of sets of k centers. For completeness, we start with a combinatorial proof of $O(n^{k+3})$ discontinuities. Although Theorem 15 is exponential as opposed to Theorem 4, it holds with probability 1 and has no dependence on α_h . First, we need to state a consequence of Rolle's Theorem.

Theorem 14 (ex. [42]). *Let f be a polynomial-exponential sum of the form $f(x) = \sum_{i=1}^N a_i b_i^x$, where $b_i > 0$, $a_i \in \mathbb{R}$, and at least one a_i is non-zero. The number of roots of f is upper bounded by N .*

Now we are ready to prove the combinatorial upper bound.

Theorem 15. *Given a clustering instance \mathcal{V} and vector $\vec{Z} \in [0, 1]^k$, the number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ as a function of α over $[0, \infty) \cup \{\infty\}$ is $O(\min(n^{k+3}, n^2 2^n))$.*

Proof. Given a clustering instance \mathcal{V} and a vector \vec{Z} , consider round t of the d^α seeding algorithm. Recall that the algorithm decides on the next center based on $z_t \in [0, 1]$, α , and the distance from each point to the set of current centers. The idea of this proof will be to count the number of α intervals such that within one interval, all possible decisions the algorithm makes are fixed. In round t , there are $\binom{n}{t-1}$ choices for the set of current centers C . We denote the points $V = \{v_1, \dots, v_n\}$ and WLOG assume the algorithm orders the intervals $I_{v_1}, I_{v_2}, \dots, I_{v_n}$. Given a point $v_i \in V$, it will

be chosen as the next center if and only if z_t lands in its interval, formally,

$$\frac{\sum_{j=1}^{i-1} d_{\min}(v_j, C)^\alpha}{\sum_{j=1}^n d_{\min}(v_j, C)^\alpha} < z_t < \frac{\sum_{j=1}^i d_{\min}(v_j, C)^\alpha}{\sum_{j=1}^n d_{\min}(v_j, C)^\alpha}$$

By Theorem 14, these two equations have at most $n + 1$ roots each. Therefore, in the $n + 2$ intervals of α between each root, the decision whether or not to choose v_i as a center in round t is fixed. Note that the center set C , the point v_i , and the number z_t fixed the coefficients of the equation. Since in round t , there are $\binom{n}{t}$ choices of centers, then there are $\sum_{t=1}^k \binom{n}{k} \cdot n \cdot 2$ total equations which determine the outcome of the algorithm. Each equation has at most $n + 1$ roots, so it follows there are $1 + \sum_{t=1}^k \binom{n}{k} \cdot n \cdot 2(n + 1) \in O(n^3 \cdot n^k)$ total intervals of α along $[0, \infty) \cup \{\infty\}$ such that within each interval, the entire outcome of the algorithm, $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$, is fixed. Note that we used $k \cdot n^k$ to bound the total number of choices for the set of current centers. We can also bound this quantity by 2^n , since each point is either a center or not a center. This results in a final bound of $O(\min(n^{k+3}, n^2 2^n))$. \square

Lemma 3 (restated). *Assume that v_1, \dots, v_n are sorted in decreasing distance from a set C of centers. Then for each $i = 1, \dots, n$, the function $\alpha \mapsto \frac{D_i(\alpha)}{D_n(\alpha)}$ is monotone increasing and continuous along $[0, \infty)$. Furthermore, for all $1 \leq i \leq j \leq n$ and $\alpha \in [0, \infty)$, we have $\frac{D_i(\alpha)}{D_n(\alpha)} \leq \frac{D_j(\alpha)}{D_n(\alpha)}$.*

Proof. Recall that $D_i(\alpha) = \sum_{j=1}^i d_{\min}(v_j, C)^\alpha$, where v_1, \dots, v_n are the points sorted in decreasing order of distance to the set of centers C .

First we show that for each i , the function $\alpha \mapsto D_i(\alpha)/D_n(\alpha)$ is monotone increasing. Given $\alpha_1 < \alpha_2$, we must show that for each i ,

$$\frac{D_i(\alpha_1)}{D_n(\alpha_1)} \leq \frac{D_i(\alpha_2)}{D_n(\alpha_2)}.$$

This is equivalent to showing

$$D_i(\alpha_1)D_n(\alpha_2) \leq D_i(\alpha_2)D_n(\alpha_1).$$

Using the shorthand notation $d_j = d(v_j, C)$, we have

$$\begin{aligned} D_i(\alpha_1)D_n(\alpha_2) &= \left(\sum_{j=1}^i d_j^{\alpha_1} \right) \left(\sum_{j=1}^n d_j^{\alpha_2} \right) \\ &= \left(\sum_{j=1}^i d_j^{\alpha_1} \right) \left(\sum_{j=1}^i d_j^{\alpha_2} \right) + \left(\sum_{j=1}^i d_j^{\alpha_1} \right) \left(\sum_{j=i+1}^n d_j^{\alpha_2} \right) \\ &= \left(\sum_{j=1}^i d_j^{\alpha_1} \right) \left(\sum_{j=1}^i d_j^{\alpha_2} \right) + \sum_{j=1}^i \sum_{k=i+1}^n d_j^{\alpha_1} d_k^{\alpha_2} \\ &\leq \left(\sum_{j=1}^i d_j^{\alpha_1} \right) \left(\sum_{j=1}^i d_j^{\alpha_2} \right) + \sum_{j=1}^i \sum_{k=i+1}^n d_j^{\alpha_1} d_k^{\alpha_2} \left(\frac{d_j}{d_k} \right)^{\alpha_2 - \alpha_1} \\ &= \left(\sum_{j=1}^i d_j^{\alpha_1} \right) \left(\sum_{j=1}^i d_j^{\alpha_2} \right) + \sum_{j=1}^i \sum_{k=i+1}^n d_j^{\alpha_2} d_k^{\alpha_1} \end{aligned}$$

$$\begin{aligned}
&= \left(\sum_{j=1}^i d_j^{\alpha_2} \right) \left(\sum_{j=1}^n d_j^{\alpha_1} \right) \\
&= D_i(\alpha_2) D_n(\alpha_1),
\end{aligned}$$

as required.

Next we show that $\alpha \mapsto D_i(\alpha)/D_n(\alpha)$ is continuous along $[0, \infty)$. $D_i(\alpha)$ and $D_n(\alpha)$ are both sums of simple exponential functions, so they are continuous. $D_n(\alpha)$ is always at least n along $[0, \infty)$, therefore, $D_i(\alpha)/D_n(\alpha)$ is continuous.

Finally, we show that for all $1 \leq i \leq j \leq n$, we have $\frac{D_i(\alpha)}{D_n(\alpha)} \leq \frac{D_j(\alpha)}{D_n(\alpha)}$. Given $1 \leq i \leq j \leq n$, then

$$\frac{D_j(\alpha)}{D_n(\alpha)} - \frac{D_i(\alpha)}{D_n(\alpha)} = \frac{d_{i+1}^\alpha + \cdots + d_j^\alpha}{D_n(\alpha)} \geq 0$$

This completes the proof. \square

Now to work up to the proof of Theorem 4, we bound the derivative of $\frac{D_i(\alpha)}{D_n(\alpha)}$.

Lemma 16. *Given distances $d_1 \geq \cdots \geq d_n$, for any index i and $\alpha > 0$, we have*

$$\left| \frac{\partial}{\partial \alpha} \left(\frac{D_i(\alpha)}{D_n(\alpha)} \right) \right| \leq \min \left\{ \frac{2}{\alpha} \log n, \log \frac{d_1}{d_n} \right\}.$$

Proof. For any $i \in [n]$, the derivative of $D_i(\alpha)$ is given by $D_i'(\alpha) = \sum_{j=1}^i d_j^\alpha \log d_j$. With this,

$$\begin{aligned}
\frac{\partial}{\partial \alpha} \left(\frac{D_i(\alpha)}{D_n(\alpha)} \right) &= \frac{D_i'(\alpha) D_n(\alpha) - D_n'(\alpha) D_i(\alpha)}{(D_n(\alpha))^2} \\
&= \frac{\left(\sum_{x=1}^i d_x^\alpha \log d_x \right) \left(\sum_{y=1}^n d_y^\alpha \right) - \left(\sum_{x=1}^i d_x^\alpha \right) \left(\sum_{y=1}^n d_y^\alpha \log d_y \right)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \\
&= \frac{\sum_{x=1}^i \sum_{y=1}^n (d_x^\alpha d_y^\alpha \log d_x) - \sum_{x=1}^i \sum_{y=1}^n (d_x^\alpha d_y^\alpha \log d_y)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \\
&= \frac{\sum_{x=1}^i \sum_{y=i+1}^n (d_x^\alpha d_y^\alpha \log d_x) - \sum_{x=1}^i \sum_{y=i+1}^n (d_x^\alpha d_y^\alpha \log d_y)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \\
&= \frac{\sum_{x=1}^i \sum_{y=i+1}^n \left(d_x^\alpha d_y^\alpha \log \left(\frac{d_x}{d_y} \right) \right)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \tag{1}
\end{aligned}$$

At this point, because $d_1 \geq \cdots \geq d_n \geq 0$, we achieve our second bound as follows.

$$\frac{\sum_{x=1}^i \sum_{y=i+1}^n \left(d_x^\alpha d_y^\alpha \log \left(\frac{d_x}{d_y} \right) \right)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \leq \frac{\sum_{x=1}^i \sum_{y=i+1}^n d_x^\alpha d_y^\alpha}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \cdot \log \frac{d_1}{d_n} \leq \log \frac{d_1}{d_n}. \tag{2}$$

Recall our goal is to bound the derivative by a minimum of two quantities. Equation 2 gives us the second bound. To achieve the first bound, we bound Equation 1 a different way, as follows.

$$\frac{\sum_{x=1}^i \sum_{y=i+1}^n \left(d_x^\alpha d_y^\alpha \log \left(\frac{d_x}{d_y} \right) \right)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \leq \frac{1}{\alpha} \cdot \frac{\sum_{x=1}^i d_x^\alpha \left(\sum_{y=i+1}^n d_y^\alpha \log \left(\frac{d_x}{d_y} \right) \right)}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \tag{3}$$

To upper bound (3), we will show that $\sum_{y=i+1}^n d_y^\alpha \log \left(\frac{d_1}{d_y} \right) \leq 2 \log n \sum_{y=1}^n d_y^\alpha$. We bound each term of the sum in one of two cases:

Case 1: If y is such that $d_y^\alpha \geq \frac{d_1^\alpha}{n^2}$, then $d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \leq 2d_y^\alpha \log n$.

Case 2: If y is such that $d_y^\alpha < \frac{d_1^\alpha}{n^2}$, then $d_y^\alpha \log\left(\frac{d_1^\alpha}{d_y^\alpha}\right) \leq d_1^\alpha \left(\frac{d_y^\alpha}{d_1^\alpha} \log \frac{d_1^\alpha}{d_y^\alpha}\right) \leq \frac{1}{n} \cdot d_1^\alpha$, where the last inequality follows because $\frac{1}{x} \log x \leq \frac{1}{n}$ for all $x > n^2$.

Let $C_1 = \{y \geq i+1 \mid d_y^\alpha \geq \frac{d_1^\alpha}{n^2}\}$ and $C_2 = \{y \geq i+1 \mid d_y^\alpha < \frac{d_1^\alpha}{n^2}\}$ be the values of y corresponding to Cases 1 and 2, respectively. Then we have

$$\sum_{y=i+1}^n d_y^\alpha \cdot \log \frac{d_x^\alpha}{d_y^\alpha} = \sum_{y \in C_1} d_y^\alpha \cdot \log \frac{d_x^\alpha}{d_y^\alpha} + \sum_{y \in C_2} d_y^\alpha \cdot \log \frac{d_x^\alpha}{d_y^\alpha} \leq 2 \log(n) \sum_{y=2}^n d_y^\alpha + d_1^\alpha \leq 2 \log(n) \sum_{y=1}^n d_y^\alpha.$$

Substituting this into (3), we have that

$$\frac{\partial}{\partial \alpha} \frac{D_i(\alpha)}{D_n(\alpha)} \leq \frac{2}{\alpha} \log(n) \cdot \frac{\sum_{x=1}^i \sum_{y=i+1}^n d_x^\alpha d_y^\alpha}{\sum_{x=1}^n \sum_{y=1}^n d_x^\alpha d_y^\alpha} \leq \frac{2}{\alpha} \log n,$$

as required. □

Theorem 4. Fix any clustering instance \mathcal{V} and let $\vec{Z} \sim \text{Uniform}([0, 1]^k)$. Then:

1. For any parameter interval $[\alpha_\ell, \alpha_h]$ with $\alpha_\ell > 0$, the expected number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ on $[\alpha_\ell, \alpha_h]$ is at most $O(nk \log(n) \log(\alpha_h/\alpha_\ell))$.
2. For any parameter interval $[0, \alpha_h]$, the expected number of discontinuities of the function $\alpha \mapsto \text{seed}_\alpha(\mathcal{V}, \vec{Z})$ on $[0, \alpha_h]$ is at most $O(nk \log(n) \log(\alpha_h \log(R)))$, where R is the largest ratio between any pair of non-zero distances in \mathcal{V} .

Proof. Given \mathcal{V} , we will show that $\mathbb{E}[\#I] \leq nk \log n \log \frac{\alpha_h}{\alpha_\ell}$ over $[\alpha_\ell, \alpha_h]$ and $\mathbb{E}[\#I] \leq 4nk \log n (1 + \log \alpha_h + \log \log D)$ over $[0, \alpha_h]$, where $\#I$ denotes the total number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ and the expectation is over the draw of $\vec{Z} \in [0, 1]^k$. Consider round t of a run of Algorithm 1. Suppose at the beginning of round t , there are L possible states of the algorithm, e.g., L sets of α such that within a set, the choice of the first $t-1$ centers is fixed. By Lemma 3, we can write these sets as $[\alpha_0, \alpha_1], \dots, [\alpha_{L-1}, \alpha_L]$, where $0 = \alpha_0 < \dots < \alpha_L = \alpha_h$. Given one interval, $[\alpha_\ell, \alpha_{\ell+1}]$, we claim the expected number of new breakpoints $\#I_{t,\ell}$ by choosing a center in round t is bounded by

$$\min(2n \log R(\alpha_{\ell+1} - \alpha_\ell), n - t - 1, 4n \log n (\log \alpha_{\ell+1} - \log \alpha_\ell)).$$

Note that $\#I_{t,\ell} + 1$ is the number of possible choices for the next center in round t using α in $[\alpha_\ell, \alpha_{\ell+1}]$.

The claim gives three different upper bounds on the expected number of new breakpoints, where the expectation is only over z_t (the uniformly random draw from $[0, 1]$ used by Algorithm 1 in round t), and the bounds hold for any given configuration of $d_1 \geq \dots \geq d_n$. To prove the first statement in Theorem 4, we only need the last of the three bounds, and to prove the second statement, we need all three bounds. First we show how to prove these statements assuming the claim, and later we will prove the claim. We prove the first statement as follows. Let $\#I$ denote the total number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ along $[\alpha_\ell, \alpha_h]$.

$$E_{Z \in [0,1]^k}[\#I] \leq E_{Z \in [0,1]^k} \left[\sum_{t=1}^k \sum_{\ell=1}^{L-1} (\#I_{t,\ell}) \right]$$

$$\begin{aligned}
&\leq \sum_{t=1}^k \sum_{\ell=0}^{L-1} E_{Z \in [0,1]^k} [\#I_{t,\ell}] \\
&\leq \sum_{t=1}^k \sum_{\ell=0}^{L-1} E_{Z \text{ in } [0,1]^k} (4n \log n (\log \alpha_{\ell+1} - \log \alpha_\ell)) \\
&\leq \sum_{t=1}^k (4n \log (\log \alpha_h - \log \alpha_\ell)) \\
&\leq k \left(4n \log n \log \frac{\alpha_h}{\alpha_\ell} \right) \\
&\leq nk \log n \log \frac{\alpha_h}{\alpha_\ell}
\end{aligned}$$

Now we prove the second statement of Theorem 4. Let ℓ^* denote the largest value such that $\alpha_{\ell^*} < \frac{1}{\log R}$. Such an ℓ^* must exist because $\alpha_0 = 0$. Then we have $\alpha_{\ell^*} < \frac{1}{\log R} \leq \alpha_{\ell^*+1}$. We use three upper bounds for three different cases of alpha intervals: the first ℓ^* intervals, interval $[\alpha_{\ell^*}, \alpha_{\ell^*+1}]$, and intervals $\ell^* + 2$ to L . Let $\#I$ denote the total number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ along $[0, \alpha_h]$.

$$\begin{aligned}
E_{Z \in [0,1]^k} [\#I] &\leq E_{Z \in [0,1]^k} \left[\sum_{t=1}^k \sum_{\ell=1}^{L-1} (\#I_{t,\ell}) \right] \\
&\leq \sum_{t=1}^k \sum_{\ell=0}^{L-1} E_{Z \in [0,1]^k} [\#I_{t,\ell}] \\
&\leq \sum_{t=1}^k \left(\sum_{\ell=0}^{\ell^*-1} E_{Z \in [0,1]^k} [\#I_{t,\ell}] + E_{Z \in [0,1]^k} [\#I_{t,\ell^*}] + \sum_{\ell=\ell^*+1}^{L-1} E_{Z \in [0,1]^k} [\#I_{t,\ell}] \right) \\
&\leq \sum_{t=1}^k \left(\sum_{\ell=0}^{\ell^*-1} (2n \log D(\alpha_{\ell+1} - \alpha_\ell)) + (n - t - 1) + \sum_{\ell=\ell^*+1}^{L-1} (4n \log n (\log \alpha_{\ell+1} - \log \alpha_\ell)) \right) \\
&\leq \sum_{t=1}^k (2n \log D \cdot \alpha_{\ell^*} + n + 4n \log n (\log \alpha_h - \log \alpha_{\ell^*})) \\
&\leq \sum_{t=1}^k \left(2n \log D \cdot \frac{1}{\log D} + n + 4n \log n \left(\log \alpha_h - \log \left(\frac{1}{\log D} \right) \right) \right) \\
&\leq k (2n + n + 4n \log n (\log \alpha_h + \log \log D)) \\
&\leq 4nk \log n (1 + \log \alpha_h + \log \log D).
\end{aligned}$$

Now we will prove the claim. Recall that for α -sampling, each point x receives an interval along $[0, 1]$ of size $\frac{d_x^\alpha}{D_n(\alpha)}$, so the number of breakpoints in round t along $[\alpha_\ell, \alpha_{\ell+1}]$ corresponds to the number of times z_t switches intervals as we increase α from α_ℓ to $\alpha_{\ell+1}$. By Lemma 3, the endpoints of these intervals are monotone increasing, continuous, and non-crossing, so the number of breakpoints is exactly $x - y$, where x and y are the minimum indices s.t. $\frac{D_x(\alpha_\ell)}{D_n(\alpha_\ell)} > z_t$ and $\frac{D_y(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} > z_t$, respectively (see Figure 3). We want to compute the expected value of $x - y$ for z_t uniform in $[0, 1]$ (here, x and y are functions of z_t).

We take the approach of analyzing each interval individually. One method for bounding $E_{z_t \in [0,1]}[x - y]$ is to compute the maximum possible number of breakpoints for each interval I_{v_j} , for all $1 \leq j \leq n$. Specifically, if we let i denote the minimum index such that $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} < \frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then

$$\begin{aligned} E[\#I_{t,\ell}] &\leq \sum_{j=1}^n P\left(\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} < z_t < \frac{D_{j+1}(\alpha_\ell)}{D_n(\alpha_\ell)}\right) \cdot (j - i + 1) \\ &\leq \sum_{j=1}^n \frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)} \cdot (j - i + 1). \end{aligned}$$

In this expression, we are using the worst case number of breakpoints within each bucket, $j - i + 1$.

We cannot quite use this expression to obtain our bound; for example, when $\alpha_{\ell+1} - \alpha_\ell$ is extremely small, $j - i + 1 = 1$, so this expression will give us $E[\#I_{t,\ell}] \leq 1$ over $[\alpha_\ell, \alpha_{\ell+1}]$, but we need to show the expected number of breakpoints is proportional to ϵ to prove the claim. To tighten up this analysis, we will show that for each bucket, the probability (over z_t) of achieving the maximum number of breakpoints is low.

Assuming that z_t lands in a bucket I_{v_j} , we further break into cases as follows. Let i denote the minimum index such that $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} > \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$. Note that i is a function of j, α_ℓ , and $\alpha_{\ell+1}$, but it is independent of z_t . If z_t is less than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then we have the maximum number of breakpoints possible, since the algorithm chooses center v_{i-1} when $\alpha = \alpha_{\ell+1}$ and it chooses center v_j when $\alpha = \alpha_\ell$. The number of breakpoints is therefore $j - i + 1$, by Lemma 3. We denote this event by $E_{t,j}$, i.e., $E_{t,j}$ is the event that in round t , z_t lands in I_{v_j} and is less than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$. If z_t is instead greater than $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, then the algorithm chooses center v_i when $\alpha = \alpha_{\ell+1}$, so the number of breakpoints is $\leq j - i$. We denote this event by $E'_{t,j}$ (see Figure 3). Note that $E_{t,j}$ and $E'_{t,j}$ are disjoint and $E_{t,j} \cup E'_{t,j}$ is the event that $z_t \in I_{v_j}$.

Within an interval I_{v_j} , the expected number of breakpoints is

$$P(E_{t,j})(j - i + 1) + P(E'_{t,j})(j - i) = P(E_{t,j} \cup E'_{t,j})(j - i) + P(E'_{t,j}).$$

We will show that $j - i$ and $P(E_{t,j})$ can both be bounded using Lemma 16, which finishes off the claim.

First we upper bound $P(E_{t,j})$. Recall this is the probability that z_t is in between $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$ and $\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})}$, which is

$$\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} \leq \frac{D_j(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} = \frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}}.$$

Recall that Lemma 16 states that $\left| \frac{\partial}{\partial \alpha} \left(\frac{D_i(\alpha)}{D_n(\alpha)} \right) \right| \leq \min \left(\frac{2}{\alpha} \log n, \log \left(\frac{d_1}{d_n} \right) \right)$. If we use the second part of the min expression (and use $\frac{d_1}{d_n} \leq R$) we get $\frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \leq \log D(\alpha_{\ell+1} - \alpha_\ell)$. If we use the first part of the min expression, we get

$$\frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \leq 2 \log n \int_{\alpha_\ell}^{\alpha_{\ell+1}} \frac{1}{\alpha} \cdot d\alpha \leq 2 \log n (\log \alpha) \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} = 2 \log n (\log \alpha_{\ell+1} - \log \alpha_\ell).$$

Now we upper bound $j - i$. Recall that $j - i$ represents the number of intervals between $\frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}$ and $\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)}$ (see Figure 3). Note that the smallest interval in this range is $\frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)}$, and

$$\frac{D_j(\alpha_\ell)}{D_n(\alpha_\ell)} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)} \leq \frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)}.$$

Therefore, the expected number of breakpoints is at most $\frac{D_n(\alpha_\ell)}{d_j^{\alpha_\ell}} \cdot \left(\frac{D_i(\alpha_{\ell+1})}{D_n(\alpha_{\ell+1})} - \frac{D_i(\alpha_\ell)}{D_n(\alpha_\ell)} \right)$, and we can bound the second half of this fraction by again using Lemma 16. To finish off the proof, we have

$$\begin{aligned}
E[\#I_{t,\ell}] &\leq \sum_j (P(E'_{t,j}) \cdot (j-i) + P(E_{t,j}) \cdot (j-i+1)) \\
&\leq \sum_j (P(E'_{t,j}) \cdot (j-i) + P(E_{t,j}) \cdot (j-i) + P(E_{t,j})) \\
&\leq \sum_j (P(E'_{t,j} \cup E_{t,j}) \cdot (j-i) + P(E_{t,j})) \\
&\leq \sum_j (P(z_t \in I_{v_j}) \cdot (j-i)) + \sum_j P(E_{t,j}) \\
&\leq \sum_j \left(\frac{d_j^{\alpha_\ell}}{D_n(\alpha_\ell)} \right) \left(\frac{D_n(\alpha_\ell)}{d_j^{\alpha_\ell}} \cdot \frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \right) + \sum_j \left(\frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \right) \\
&\leq 2n \left(\frac{D_j(\alpha)}{D_n(\alpha)} \Big|_{\alpha_\ell}^{\alpha_{\ell+1}} \right) \\
&\leq 2n \min(2 \log n(\log \alpha_{\ell+1} - \log \alpha_\ell), \log D(\alpha_{\ell+1} - \alpha_\ell))
\end{aligned}$$

This accounts for two of the three upper bounds in our claim. To complete the proof, we note that $E[\#I_{t,\ell}] \leq n-t-1$ simply because there are only $n-t$ centers available to be chosen in round t of the algorithm (and therefore, $n-t-1$ breakpoints). \square

In fact, we can also show that the worst-case number of discontinuities is exponential.

Lemma 17. *Given n , there exists a clustering instance \mathcal{V} of size n and a vector \vec{Z} such that the number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ as a function of α over $[0, 2]$ is $2^{\Omega(n)}$.*

Proof. We construct $\mathcal{V} = (V, d, k)$ and $\vec{Z} = \{z_1, \dots, z_k\}$ such that $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ has $2^{k/3}$ different intervals in α which give different outputs.

Here is the outline of the construction. At the start, we set z_1 so that one point, v , will always be the first center chosen. Then we add points $a_1, b_1, \dots, a_k, b_k$ such that in round i , either a_i or b_i will be chosen as centers. We carefully set the distances so that for each combinations of centers, there is an α interval which achieves this combination of centers. Therefore, the total number of α intervals such that the output of the sampling step is fixed, is 2^{k-1} . Our construction also uses points $a'_1, b'_1, \dots, a'_k, b'_k$ and v_1, \dots, v_k which are never chosen as centers, but will be crucial in the analysis.

Next, we describe the distances between the points in our clustering instance. Almost all distances will be set to 100, except for a few distances: for all i , $d(a_i, a'_i) = d(b_i, b'_i) = \epsilon$, $d(b_i, v_i) = 100 - o_i$, $d(a_i, b_i) = 2o_i$, $d(v, a_1) = d(v, b_1) = 99$, and $d(a_{i-1}, a_i) = d(a_{i-1}, b_i) = d(b_{i-1}, a_i) = d(b_{i-1}, b_i) = 100 - o_i$, for $0 \leq \epsilon, o_1, \dots, o_k \leq 1$ to be specified later. At the end, we will perturb all other distances by a slight amount ($< \epsilon$) away from 100, to break ties.

Now we set up notation to be used in the remainder of the proof. We set $z_i = \frac{1}{2}$ for all i . For $1 \leq i \leq k$, given $\vec{x} \in \{0, 1\}^{i-1}$, let $E_{\vec{x}}$ denote the equation in round i which determines whether a_i or b_i is chosen as the next center, in the case where for all $1 \leq j < i$, $a_j \in C$ if $\vec{x}_j = 0$, or else $b_j \in C$ (and let E' denote the single equation in round 2). Specifically, $E_{\vec{x}}$ is the following expression

$$\frac{100^\alpha \binom{n-(i-1)}{2}}{100^\alpha(n-2(i-1)) + (100-o_i)^\alpha + \sum_{j=1}^{i-1}(100-\vec{x}_j o_j)^\alpha}.$$

Let $\alpha_{\vec{x}}$ denote the solution to equation $E_{\vec{x}} = \frac{1}{2}$ in $[1, 3]$, if it exists. In the rest of the proof, we must show there exist parameters $\epsilon, o_0, \dots, o_k$ which admit an ordering to the values $\alpha_{\vec{x}}$ which ensures that each $\alpha_{\vec{x}}$ falls in the correct range to split up each interval, thus achieving 2^{k-1} intervals. The ordering of the $\alpha_{\vec{x}}$'s can be specified by two conditions: (1) $\alpha_{[\vec{x} \ 0]} < \alpha_{[\vec{x}]} < \alpha_{[\vec{x} \ 1]}$ and (2) $\alpha_{[\vec{x} \ 0 \ \vec{y}]} < \alpha_{[\vec{x} \ 1 \ \vec{z}]}$ for all $\vec{x}, \vec{y}, \vec{z} \in \bigcup_{i < k} \{0, 1\}^i$ and $|\vec{y}| = |\vec{z}|$. To prove the $\alpha_{\vec{x}}$'s follow this ordering, we use an inductive argument. We must show the following claim: there exist $0 < o_1, \dots, o_k < 1$ such that if we solve $E_{\vec{x}} = \frac{1}{2}$ for $\alpha_{\vec{x}}$ for all $\vec{x} \in \bigcup_{i < k} \{0, 1\}^i$, then the α 's satisfy $\alpha_{[\vec{x} \ 0]} < \alpha_{[\vec{x}]} < \alpha_{[\vec{x} \ 1]}$ and for all $i < k$, $\alpha_{[\vec{x} \ 1]} < \alpha_{[\vec{y} \ 0]}$ for $\vec{x}, \vec{y} \in \{0, 1\}^i$ and $x_1 \dots x_i < y_1 \dots y_i$.

Given $\vec{x} \in \{0, 1\}^i$, for $1 \leq i \leq k-1$, let $p(\vec{x}), n(\vec{x}) \in \{0, 1\}^i$ denote the vectors which sit on either side of $\alpha_{\vec{x}}$ in the desired ordering, i.e., $\alpha_{\vec{x}}$ is the only $\alpha_{\vec{y}}$ in the range $(\alpha_{p(\vec{x})}, \alpha_{n(\vec{x})})$ such that $|\vec{y}| = i$. If $\vec{x} = [1 \dots 1]$, then set $\alpha_{n(\vec{x})} = 3$, and if $\vec{x} = [0 \dots 0]$, then set $\alpha_{p(\vec{x})} = 1$.

Given $1 \leq i \leq k-2$, assume there exist $0 < o_1, \dots, o_i < 1$ such that the statement is true. Now we will show the statement holds for $i+1$. Given $\vec{x} \in \{0, 1\}^i$, by assumption, we have that the solution to $E_{\vec{x}} = \frac{1}{2}$ is equal to $\alpha_{\vec{x}}$. First we consider $E_{[\vec{x} \ 0]} = \frac{1}{2}$. Note there are two differences in the equations $E_{\vec{x}}$ and $E_{[\vec{x} \ 0]}$. First, the number of 100^α terms in the numerator decreases by 1, and the number of terms in the denominator decreases by 2. WLOG, at the end we set a constant $c = \frac{n}{k}$ large enough so that this effect on the root of the equation is negligible for all n . Next, the offset in the denominator changes from $(100-o_i)^\alpha$ to $(100-o_{i+1})^\alpha$. Therefore, if $0 < o_{i+1} < o_i$, then $\alpha_{[\vec{x} \ 0]} < \alpha_{\vec{x}}$. Furthermore, there exists an upper bound $0 < z_{i+1} < o_i$ such that for all $0 < o_{i+1} < z_{i+1}$, we have $\alpha_{[\vec{x} \ 0]} \in (\alpha_{p(\vec{x})}, \alpha_{\vec{x}})$. Next we consider $E_{[\vec{x} \ 1]} = \frac{1}{2}$. As before, the number of 100^α terms decrease, which is negligible. Note the only other change is that an 100^α term is replaced with $(100-o_{i+1})^\alpha$. Therefore, as long as $0 < o_{i+1} < o_i$, then $\alpha_{\vec{x}} < \alpha_{[\vec{x} \ 1]}$, and similar to the previous case, there exists an upper bound $0 < z'_{i+1} < o_i$ such that for all $0 < o_{i+1} < z'_{i+1}$, we have $\alpha_{[\vec{x} \ 1]} \in (\alpha_{\vec{x}}, \alpha_{n(\vec{x})})$. We conclude that there exists $0 < o_{i+1} < \min(z_i, z'_i) < o_i$ such that $\alpha_{p(\vec{x})} < \alpha_{[\vec{x} \ 0]} < \alpha_{\vec{x}} < \alpha_{[\vec{x} \ 1]} < \alpha_{n(\vec{x})}$, thus finishing the inductive proof.

Now we have shown that there are $2^{k'}$ nonoverlapping α intervals, such that within every interval, d^α -sampling chooses a unique set of centers, for $k' = k-1$. To finish our structural claim, we will show that after β -Lloyd's method, the cost function $\text{clus}_{\alpha, \beta}(\mathcal{V}, \vec{Z})$ alternates $2^{k'}$ times above and below a value r as α increases. We add two points, a and b , so that $d(v, a) = d(v, b) = 100$, $d(a_k, a) = d(b_k, b) = 100 - \epsilon$, and the distances from all other points to a and b are length $100 + \epsilon$. Then we add many points in the same location as v , a_i , and b_i , so that any set c returned by d^α -sampling is a local minima for β -Lloyd's method, for all β . Furthermore, these changes do not affect the previous analysis, as long as we appropriately balance the terms in the numerator and denominator of each equation $E_{\vec{x}}$ (and for small enough ϵ). Finally, we set v and a to have label 1 in the target clustering, and all points are labeled 2. Therefore, as d^α -sampling will alternate between $a_k \in C$ and $b_k \notin C$ as we increase α , a and v alternate being in the same or different clusters, so the function $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ will alternate between different outputs $2^{\Omega(n)}$ times as a function of α . \square

Now we give the details for Theorem 5.

Theorem 5 (restated). *Given $T \in \mathcal{N}$, a clustering instance \mathcal{V} , and a fixed set C of initial centers, the number of discontinuities of $\text{lloyd}s_\beta(\mathcal{V}, C, T)$ as a function of β on instance \mathcal{V} is $O(\min(n^{3T}, n^{k+3}))$.*

Proof. Given a clustering instance \mathcal{V} and a vector \vec{Z} , we bound the number of possible intervals created by the Lloyd's step, given a fixed set of initial centers. Define $\text{lloyd}_\beta(\mathcal{V}, C)$ as the cost of the clustering outputted by the β -Lloyd iteration algorithm on \mathcal{V} using initial centers C . Note that the Voronoi partitioning step is only dependent on C , in particular, it is independent of β . Let $\{C_1, \dots, C_k\}$ denote the Voronoi partition of V induced by C . Given one of these clusters C_i , the next center is computed by $\min_{c \in C_i} \sum_{v \in C_i} d(c, v)^\beta$. Given any $c_1, c_2 \in C_i$, the decision for whether c_1 is a better center than c_2 is governed by $\sum_{v \in C_i} d(c_1, v)^\beta < \sum_{v \in C_i} d(c_2, v)^\beta$. Again by Theorem 14, this equation has at most $2n + 1$ roots. Notice that this equation depends on the set C of centers, the choice of a cluster C_i , and the two points $c_1, c_2 \in C_i$. Then there are $\binom{n}{k} \cdot n \cdot \binom{n}{2}$ total equations which fix the outcome of the Lloyd's method, and there are $\binom{n}{k} \cdot n \cdot \binom{n}{2} \cdot (2n + 1) \leq n^{k+4}$ total intervals of β such that the outcome of Lloyd's method is fixed.

Next we give a different analysis which bounds the number of discontinuities by n^{3T} , where T is the maximum number of Lloyd's iterations. By the same analysis as the previous paragraph, if we only consider one round, then the total number of equations which govern the output of a Lloyd's iteration is $\binom{n}{2}$, since the set of centers C is fixed. These equations have $2n + 1$ roots, so the total number of intervals in one round is $O(n^3)$. Therefore, over T rounds, the number of intervals is $O(n^{3T})$. \square

Now we give the details for the proofs of the generalized results, Theorems 10 and 11.

Theorem 10 (restated). *Given an α -parameterized family such that (1) for all $1 \leq i \leq k$ and $C \subseteq V$ such that $|C| \leq k$, each $S_{i,C}(\alpha)$ is monotone increasing and continuous as a function of α , and (2) for all $1 \leq i \leq j \leq n$ and $\alpha \in (\alpha_\ell, \alpha_h)$, $S_{i,C}(\alpha) \leq S_{j,C}(\alpha)$, then the expected number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z}, p)$ as a function of α is $O(nkD_p(\alpha_h - \alpha_\ell))$.*

Proof. Given \mathcal{V} and $[0, \alpha_h]$, we will show that $\mathbb{E}[\#I] \leq nk \log n \cdot \alpha_h$, where $\#I$ denotes the total number of discontinuities of $\text{seed}_\alpha(\mathcal{V}, \vec{Z})$ and the expectation is over the randomness $Z \in [0, 1]^k$ of the d^α -sampling algorithm. Consider round t of a run of the algorithm. Suppose at the beginning of round t , there are L possible states of the algorithm, e.g., L sets of α such that within a set, the choice of the first $t - 1$ centers is fixed. From the monotonicity property, we can write these sets as $[\alpha_0, \alpha_1], \dots, [\alpha_{L-1}, \alpha_L]$, where $0 = \alpha_0 < \dots < \alpha_L = \alpha_h$. Recall that $D_p = \max_{i,C,v,\alpha} \left(\frac{\partial S_{i,C}(\alpha)}{\partial \alpha} \right)$. Given one interval, $[\alpha_\ell, \alpha_{\ell+1}]$, we claim the expected number of new breakpoints $\#I_{t,\ell}$ by choosing a center in round t is bounded by $nD_p(\alpha_{\ell+1} - \alpha_\ell)$. Note that $\#I_{t,\ell} + 1$ is the number of possible choices for the next center in round t using α in $[\alpha_\ell, \alpha_{\ell+1}]$.

The claim gives an upper bound on the expected number of new breakpoints, where the expectation is only over z_t (the uniformly random draw from $[0, 1]$ used by the initialization algorithm in round t), and the bound holds for any set of centers and points. Assuming the claim, we can finish off the proof by using linearity of expectation as follows.

$$\begin{aligned} E_{Z \in [0,1]^k}[\#I] &\leq E_{Z \in [0,1]^k} \left[\sum_{t=1}^k \sum_{\ell=1}^L (\#I_{t,\ell}) \right] \\ &\leq \sum_{t=1}^k \sum_{\ell=1}^L E_{Z \in [0,1]^k}[\#I_{t,\ell}] \\ &\leq \sum_{t=1}^k \sum_{\ell=1}^L nD_p(\alpha_{\ell+1} - \alpha_\ell) \end{aligned}$$

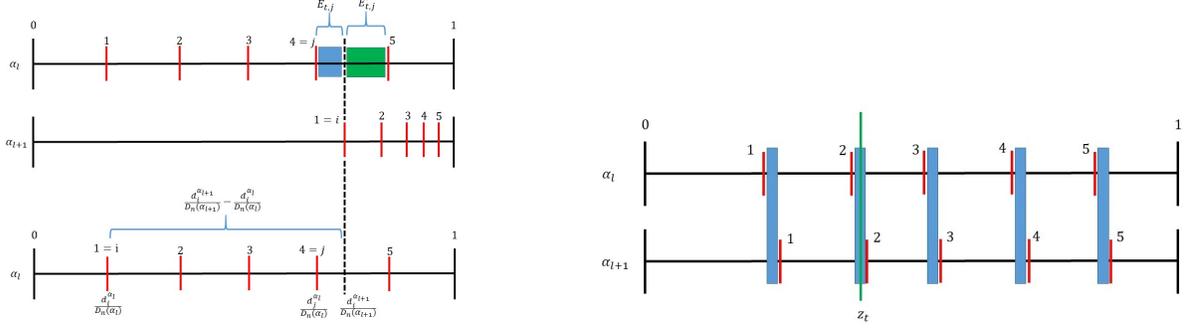


Figure 3: (Repeated) Definition of $E_{t,j}$ and $E'_{t,j}$, and details for bounding $j - i$ (left). Intuition for bounding $P(E_{t,j})$, where the blue regions represent $E_{t,j}$ (right).

$$\leq nkD_p \cdot \alpha_h$$

Now we will prove the claim. Recall that for α -sampling, each point v_i receives an interval along $[0, 1]$ of size proportional to $p_\alpha(v_i, C) \frac{d_x^\alpha}{D_n(\alpha)}$, so the number of breakpoints in round t along $[\alpha_\ell, \alpha_{\ell+1}]$ corresponds to the number of times z_t switches intervals as we increase α from α_ℓ to $\alpha_{\ell+1}$. Note that $S_{i,C}(\alpha)$ corresponds to the division boundary between the interval of v_i and v_{i+1} . By assumption, these divisions are monotone increasing, so the number of breakpoints is exactly $x - y$, where x and y are the minimum indices s.t. $\frac{S_{x,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} > z_t$ and $\frac{S_{y,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} > z_t$, respectively. We want to compute the expected value of $x - y$ for z_t uniform in $[0, 1]$ (here, x and y are functions of z_t).

We take the approach of analyzing each interval individually. One method for bounding $E_{z_t \in [0,1]}[x - y]$ is to compute the maximum possible number of breakpoints for each interval I_{v_j} , for all $1 \leq j \leq n$. Specifically, if we let i denote the minimum index such that $\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} < \frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}$, then

$$\begin{aligned} E[\#I_{t,\ell}] &\leq \sum_{j=1}^n P\left(\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} < z_t < \frac{S_{j+1,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}\right) \cdot (j - i + 1) \\ &\leq \sum_{j=1}^n \frac{p_{\alpha_\ell}(v_j, C)}{S_{n,C}(\alpha_\ell)} \cdot (j - i + 1). \end{aligned}$$

In this expression, we are using the worst case number of breakpoints within each bucket, $j - i + 1$.

We cannot quite use this expression to obtain our bound; for example, when $\alpha_{\ell+1} - \alpha_\ell$ is extremely small, $j - i + 1 = 1$, so this expression will give us $E[\#I_{t,\ell}] \leq 1$ over $[\alpha_\ell, \alpha_{\ell+1}]$, which is not sufficient to prove the claim. Therefore, we give a more refined analysis by further breaking into cases based on whether z_t is smaller or larger than $\frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}$. In case 1, when $z_t > \frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}$, the number of breakpoints is $j - i$, and we will show that $j - i \leq nD_p(\alpha_{\ell+1} - \alpha_\ell)$. In case 2, when $z_t < \frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}$, the number of breakpoints is $j - i + 1$, but we will show the probability of this case is low.

For case 2, the probability that z_t is in between $\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}$ and $\frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}$ is $\frac{S_{j,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} - \frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} \leq \frac{S_{j,C}(\alpha_{\ell+1}) - S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_{\ell+1})}$. Therefore, we can bound this quantity by bounding the derivative $\left| \frac{\partial}{\partial \alpha} \left(\frac{S_{j,C}(\alpha)}{S_{n,C}(\alpha)} \right) \right|$, which is at most D_p by definition.

For case 1, recall that $j-i$ represents the number of intervals between $\frac{S_{i,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}$ and $\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}$. Note that the smallest interval in this range is $\frac{p_{\alpha_\ell}(v_i, C)}{S_{n,C}(\alpha_\ell)}$, and $\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} - \frac{S_{i,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} \leq \frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} - \frac{S_{i,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}$. Therefore, the expected number of breakpoints is at most $\frac{S_{n,C}(\alpha_\ell)}{p_{\alpha_\ell}(v_j, C)} \cdot \left(\frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} - \frac{S_{i,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} \right)$, and we can bound the second half of this fraction by again using the derivative of $S_{i,C}(\alpha)$.

Putting case 1 and case 2 together, we have

$$\begin{aligned}
E[\#I_{t,\ell}] &\leq \sum_j P\left(\frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} < z_t < \frac{S_{j+1,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}\right) \cdot (j-i) + \sum_j P\left(\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} < z_t < \frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}\right) \cdot (j-i+1) \\
&\leq \sum_j P\left(\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} < z_t < \frac{S_{j+1,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}\right) \cdot (j-i) + \sum_j P\left(\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} < z_t < \frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}\right) \\
&\leq \sum_j \frac{p_{\alpha_\ell}(v_j, C)}{S_{n,C}(\alpha_\ell)} \cdot \frac{S_{n,C}(\alpha_\ell)}{p_{\alpha_\ell}(v_j, C)} \cdot \left(\frac{S_{j,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} - \frac{S_{i,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}\right) + \sum_j P\left(\frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)} < z_t < \frac{S_{j,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})}\right) \\
&\leq \sum_j \left(\frac{S_{i,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} - \frac{S_{i,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}\right) + \sum_j \left(\frac{S_{j,C}(\alpha_{\ell+1})}{S_{n,C}(\alpha_{\ell+1})} - \frac{S_{j,C}(\alpha_\ell)}{S_{n,C}(\alpha_\ell)}\right) \\
&\leq \sum_j D_p(\alpha_{\ell+1} - \alpha_\ell) + \sum_j D_p(\alpha_{\ell+1} - \alpha_\ell) \\
&\leq 2nD_p(\alpha_{\ell+1} - \alpha_\ell)
\end{aligned}$$

This concludes the proof. \square

Theorem 11 (restated). *Given parameters $0 \leq \alpha_\ell < \alpha_h$, $\epsilon > 0$, a sample \mathcal{S} of size*

$$m = O\left(\left(\frac{H}{\epsilon}\right)^2 \log\left(\log\frac{\alpha_h n D_p}{\delta}\right)\right)$$

from $(\mathcal{D} \times [0, 1]^k)^m$, and an α -parameterized family satisfying properties (1) and (2) from Theorem 10, run Algorithm 2 on each sample and collect all break-points (i.e., boundaries of the intervals A_i). With probability at least $1 - \delta$, the break-point $\bar{\alpha}$ with lowest empirical cost satisfies $|\text{clus}_{\bar{\alpha}, \beta}(\mathcal{S}) - \min_{0 \leq \alpha \leq \alpha_h} \text{clus}_{\alpha, \beta}(\mathcal{S})| < \epsilon$. The total running time to find the best break point is $O(mn^2 k^2 \alpha_h D_p \log(\frac{nH}{\epsilon}) \log n)$.

The proof is almost identical to the proof of Theorem 8.

Proof. First we argue that one of the breakpoints output by Algorithm 2 on the sample is approximately optimal. Formally, denote $\bar{\alpha}$ as the breakpoint returned by the algorithm with the lowest empirical cost over the sample, and denote α^* as the value with the minimum true cost over the distribution. We define $\hat{\alpha}$ as the empirically optimal value over the sample. We also claim that for all breakpoints α , there exists a breakpoint $\hat{\alpha}$ outputted by Algorithm 2 such that $|\alpha - \hat{\alpha}| < \frac{\epsilon}{5n^2 k L \log n}$. We will prove this claim at the end of the proof. Assuming the claim is correct, we denote α' as a breakpoint outputted by the algorithm such that $|\hat{\alpha} - \alpha'| < \frac{\epsilon}{5n^2 k L \log n}$.

For the rest of the proof, denote $\mathbb{E}_{V \sim \mathcal{D}}[\text{clus}_{\alpha, \beta}(V)] = \text{true}(\alpha)$ and $\frac{1}{m} \sum_{i=1}^m \text{clus}_{\alpha, \beta}(V^{(i)}, \vec{Z}^{(i)}) = \text{sample}(\alpha)$ since beta, the distribution, and the sample are all fixed.

By construction, we have $\text{sample}(\hat{\alpha}) \leq \text{sample}(\alpha^*)$ and $\text{sample}(\bar{\alpha}) \leq \text{sample}(\alpha')$. By Theorem 6, with probability $> 1 - \delta$, for all α (in particular, for $\bar{\alpha}$, $\hat{\alpha}$, α^* , and α'), we have $|\text{sample}(\alpha) - \text{true}(\alpha)| < \epsilon/5$. Finally, by Lemma 7, we have

$$|\hat{\alpha} - \alpha'| < \frac{\epsilon}{5n^2kL \log n} \implies |\text{true}(\hat{\alpha}) - \text{true}(\alpha')| < \epsilon/5.$$

Using these five inequalities for α' , $\hat{\alpha}$, $\bar{\alpha}$, and α^* , we can show the desired outcome as follows.

$$\begin{aligned} \text{true}(\bar{\alpha}) - \text{true}(\alpha^*) &\leq (\text{true}(\bar{\alpha}) - \text{sample}(\bar{\alpha})) + \text{sample}(\bar{\alpha}) - (\text{true}(\alpha^*) - \text{sample}(\alpha^*)) - \text{sample}(\alpha^*) \\ &\leq \epsilon/5 + \text{sample}(\alpha') + \epsilon/5 - \text{sample}(\hat{\alpha}) \\ &\leq (\text{sample}(\alpha') - \text{true}(\alpha')) + (\text{true}(\alpha') - \text{true}(\hat{\alpha})) + (\text{true}(\hat{\alpha}) - \text{sample}(\hat{\alpha})) + \frac{2\epsilon}{5} \\ &\leq \epsilon. \end{aligned}$$

Now we will prove the claim that for all breakpoints α , there exists a breakpoint $\hat{\alpha}$ outputted by Algorithm 2 such that $|\alpha - \hat{\alpha}| < \frac{\epsilon}{5n^2kL \log n}$. Denote $\epsilon' = \frac{\epsilon}{5n^2kL \log n}$. We give an inductive proof. Recall that the algorithm may only find the values of breakpoints up to additive error ϵ' , since the true breakpoints may be irrational and/or transcendental. Let \hat{T}_t denote the execution tree of the algorithm after round t , and let T_t denote the *true* execution tree on the sample. That is, T_t is the execution tree as defined earlier this section, \hat{T}_t is the execution tree with the algorithm's ϵ' imprecision on the values of alpha. Note that if a node in T_t represents an alpha-interval of size smaller than ϵ' , it is possible that \hat{T}_t does not contain the node. Furthermore, \hat{T}_t might contain spurious nodes with alpha-intervals of size smaller than ϵ' .

Our inductive hypothesis has two parts. The first part is that for each breakpoint α in T_t , there exists a breakpoint $h(\alpha)$ in \hat{T}_t such that $|\alpha - h(\alpha)| < \epsilon'$. For the second part of our inductive hypothesis, we define $B_t = \bigcup_{\alpha \text{ breakpoint}} ([\alpha, h(\alpha)] \cup [h(\alpha), \alpha])$, the set of “bad” intervals. Note that for each α , one of $[\alpha, h(\alpha)]$ and $[h(\alpha), \alpha]$ is empty. Then define $G_t = [\alpha_\ell, \alpha_h] \setminus B_t$, the set of “good” intervals. The second part of our inductive hypothesis is that the set of centers for α in T_t is the same as in \hat{T}_t , as long as $\alpha \in G_t$. That is, if we look at the leaf in T_t and the leaf in \hat{T}_t whose alpha-intervals contain α , the set of centers for both leaves are identical. Now we will prove the inductive hypothesis is true for round $t+1$, assuming it holds for round t . Given T_t and \hat{T}_t , consider a breakpoint α from T_{t+1} introduced in round $t+1$.

Case 1: $\alpha \in G_t$. Then the algorithm will recognize there exists a breakpoint, and use binary search to output a value $h(\alpha)$ such that $|\alpha - h(\alpha)| < \epsilon'$. The interval $[\alpha, h(\alpha)] \cup [h(\alpha), \alpha]$ is added to B_{t+1} , but the good intervals to the left and right of this interval still have the correct centers.

Case 2: $\alpha \in B_t$. Then there exists an interval $[\alpha', h(\alpha')] \cup [h(\alpha'), \alpha]$ containing α . By assumption, this interval is size $< \epsilon'$, therefore, we set $h(\alpha') = h(\alpha)$, so there is a breakpoint within ϵ' of α .

Therefore, for each breakpoint α in T_{t+1} , there exists a breakpoint $\hat{\alpha}$ in \hat{T}_{t+1} such that $|\alpha - \hat{\alpha}| < \epsilon'$. Furthermore, for all $\alpha \in G_{t+1}$, the set of centers for α in T_{t+1} is the same as in \hat{T}_{t+1} . This concludes the inductive proof.

Now we analyze the runtime of Algorithm 2. Let (C, A) be any node in the algorithm, with centers C and alpha interval $A = [\alpha_\ell, \alpha_h]$. Sorting the points in \mathcal{V} according to their distance to C has complexity $O(n \log n)$. Finding the points sampled by d^α -sampling with α set to α_ℓ and α_h costs $O(n)$ time. Finally, computing the alpha interval A_i for each child node of (C, A) costs $O(n \log \frac{nH}{\epsilon})$ time, since we need to perform $\log \frac{nkH \log n}{\epsilon}$ iterations of binary search on $\alpha \mapsto \frac{D_i(\alpha)}{D_n(\alpha)}$ and each evaluation of the function costs $O(n)$ time. We charge this $O(n \log \frac{nH}{\epsilon})$ time to the corresponding child node. If there are N nodes in the execution tree, summing this cost over all nodes gives a

total running time of $O(N \cdot n \log \frac{nH}{\epsilon})$. If we let $\#I$ denote the total number of α -intervals for \mathcal{V} , then each layer of the execution tree has at most $\#I$ nodes, and the depth is k , giving a total running time of $O(\#I \cdot kn \log \frac{nH}{\epsilon})$.

From Theorem 4, we have $\mathbb{E}[\#I] \leq 8nk \log n \cdot \alpha_h$. Therefore, the expected runtime of Algorithm 2 is $O(n^2 k^2 \alpha_h (\log n) (\log \frac{nH}{\epsilon}))$. This completes the proof. \square