Approximate Clustering

Ragesh Jaiswal and Sandeep Sen
Department of Computer Science and Engineering
Indian Institute of Technology Delhi
New Delhi, India, 110016
E-mail: {rjaiswal, ssen}@cse.iitd.ac.in

1.1 Introduction

Clustering is the task of partitioning a given set of objects so that similar objects belong to the same group. This general idea is extremely useful in unsupervised learning where little to no prior knowledge about the data is available. Clustering is usually the first data analysis technique employed when analysing big data. The importance of clustering and its applications may be found in a number of books (e.g., [37]) on the subject and we avoid the detailed discussion here. Instead, we focus on formulating clustering and analyzing some mathematically well-defined problems. The first issue that we need to address when formulating the clustering problem is: how is the data represented? In case it is possible to study and measure the defining properties and attributes of the objects that need to be clustered, then individual objects may be represented as points in some Euclidean space $\mathbb{R}^d$, where the dimension d denotes the number of attributes that define objects. In this scenario, the similarity or dissimilarity between objects may be defined as a function of the Euclidean distance between the points corresponding to the objects. In many situations, representing data as points in Euclidean space may not be possible and all that is known is how similar or dissimilar a pair of objects are. Since the goal of clustering is to group similar objects in the same group, this pairwise similarity/dissimilarity information suffices. In such cases, what is given is a distance function $D : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, where $\mathcal{X}$ denotes the set of objects. In most practical scenarios, $(\mathcal{X}, D)$ is a *metric*. Any metric $(\mathcal{X}, D)$ satisfies the following three properties: (i) $\forall x \in \mathcal{X}, D(x, x) = 0$, (ii) $\forall x, y \in \mathcal{X}, D(x, y) = D(y, x)$ (symmetry), and (iii) $\forall x, y, z \in \mathcal{X}, D(x, z) \leq D(x, y) + D(y, z)$ (triangle inequality). The Euclidean setting may be regarded as a special case of the metric setting where $\mathcal{X} = \mathbb{R}^d$ and where the distance $D(x, y)$ between two points $x, y \in \mathcal{X}$ is the Euclidean distance (or squared Euclidean distance).

Given any set of objects $X \subseteq \mathcal{X}$, the goal of clustering is to partition X into sets $X_1, ..., X_k$ (these sets are called *clusters*) for some k such that for any pair (x, y) of data points in the same cluster, $D(x, y)$ is small and for any pair of points in different clusters, $D(x, y)$ is large. In many scenarios, the number of clusters k is known. In such scenarios, it makes sense to define the clustering problem in one of the following ways:

- **Clustering problem #1**: Partition the given dataset X into k clusters $X_1, ..., X_k$ such that the following cost function is minimized: $\Psi_{sum}(X_1, ..., X_k) \stackrel{def.}{=} \sum_{i=1}^{k} \sum_{x,y \in X_i} D(x, y)$.
- **Clustering problem #2**: Partition the given dataset X into k clusters $X_1, ..., X_k$ such that the following cost function is minized: $\Psi_{max}(X_1, ..., X_k) \stackrel{def.}{=} \sum_{i=1}^{k} \max_{x,y \in X_i} D(x, y)$.

Note that the goal of minimizing the above objective functions is aligned with the goal of putting similar objects in the same cluster and different objects in different clusters.[1] So, these could be the problems most relevant for clustering. However, the above problem statements are in terms of partitions of the given dataset X. Partitions are complex objects (in terms of description) and it would nicer if there was an alternate formulation where the solution is much simpler to describe compared to a partition. Consider the following idea: suppose we are given k points $\{c_1, ...c_k\} \subseteq \mathcal{X}$ (let us call them *centers*) such that each of

[1] However, note that the distances *across* clusters has no contribution in the cost function.

these centers *represent* a cluster. That is, all points that have the closest centers as c_i are in the cluster X_i. This is also known as *Voronoi partitioning*. Note that since D is a metric, all points in cluster X_i defined as above will be close to each other. So, these centers may be used as compact representation of the clusters that they define and then the clustering goal would be to find good centers. This motivates the following clustering problems which we will study in the remainder of this Chapter.

- **Clustering problem #3 (Metric-k-median)**: Given dataset $X \subseteq \mathcal{X}$, find k centers $C = \{c_1, ..., c_k\} \subseteq X$ such that the following cost function is minimized: $\Psi(C, X) \stackrel{def.}{=} \sum_{x \in X} \min_{c \in C} D(x, c)$.
- **Clustering problem #4 (Metric-k-center)**: Given dataset $X \subseteq \mathcal{X}$, find k centers $C = \{c_1, ..., c_k\} \subseteq X$ such that the following cost function is minimized: $\Psi_{max}(C, X) \stackrel{def.}{=} \max_{x \in X}\{\min_{c \in C} D(x, c)\}$.

In the Euclidean setting where $\mathcal{X} = \mathbb{R}^d, D(x, y) = ||x - y||$ ($||.||$ denotes Euclidean distance) and C is allowed to be any set of k-points from $\mathcal{X}$, these problems are called the Euclidean-k-median and Euclidean-k-center problem. Furthermore, $D(x, y) = ||x - y||^2$, the problem is called the Euclidean-k-Means problem. The Euclidean-k-means problem is one of the most studied clustering problems and is popularly known as the k-means problem. For the sake of clarity, we give the description of these problems below:

- **Clustering problem #5 (Euclidean-k-median)**: Given dataset $X \subseteq \mathbb{R}^d$, find k centers $C = \{c_1, ..., c_k\} \subseteq \mathbb{R}^d$ such that the following cost function is minimized: $\phi(C, X) \stackrel{def.}{=} \sum_{x \in X} \min_{c \in C} ||c - x||$.
- **Clustering problem #6 (Euclidean-k-center)**: Given dataset $X \subseteq \mathbb{R}^d$, find k centers $C = \{c_1, ..., c_k\} \subseteq \mathbb{R}^d$ such that the following cost function is minimized: $\phi_{max}(C, X) \stackrel{def.}{=} \max_{x \in X}\{\min_{c \in C} ||c - x||\}$.
- **Clustering problem #7 (k-means)**: Given dataset $X \subseteq \mathbb{R}^d$, find k centers $C = \{c_1, ..., c_k\} \subseteq \mathbb{R}^d$ such that the following cost function is minimized: $\Phi(C, X) \stackrel{def.}{=} \sum_{x \in X} \min_{c \in C} ||c - x||^2$.

Clustering problems #3, 4, 5, 6, and 7 together are known as *center based clustering* problems. Since they are related to Voronoi partitioning, a brute-force algorithm may end up generating all possible Voronoi partitioning of n points in d dimensions. It is known [34] that there are $O(n^{kd})$ possible partitioning defined by k centers. Clearly, this would be completely impractical. We will be discussing only these center based clustering problems in the remainder of this chapter. The corresponding weighted version of the above problems are also important for many applications, in particular, for many facility location problems. Each point has an associated positive real number and the objective function is weighted by the corresponding weight for each of the terms. The weighted versions can sometimes be solved by the same algorithms with some additional adjustments in the analysis. In this chapter, we will address the unweighted versions for ease of exposition.

1.2 The k-Center Problem

The two versions of the problem that we consider in this section is the Metric-k-center problem (problem #4) and the Euclidean-k-center problem (problem #6). For both these versions of the k-center problem, even approximating the solution to within a fixed constant factor is NP-hard [27, 48, 30]. For the Euclidean-k-center problem, this approximation factor has been shown to be 1.822 and for the Metric version, the approximation factor is known to be 2. Bern and Eppstein [17] gave a nice summary of these results in their book chapter about approximation algorithms for geometric problems.

Sequential-selection Here, we will focus on the simple 2-factor approximation algorithm for the Metric-k-center problem (and hence also the Euclidean-k-center) that is based on iteratively picking k centers such that the choice of the i^{th} center depends on all the previously chosen centers. Let us call such algorithms *sequential-selection* algorithms. We will see more such sequential-selection algorithms in this chapter. The nice property of these algorithms is that they are extremely simple (and hence easy to implement and

test) and in many cases run very fast in practice. When solving the k-center problem, the high-level objective is to find k centers that are well distributed within the given data points. Solutions where two centers are very close to each other may not optimise the objective function well. Consider a very simple algorithm that iteratively picks k centers that are far apart from each other. This algorithm is called the *Farthest First Traversal* algorithm [30]. This algorithm is used as an effective heuristic for many different problems such as the Traveling Salesman Problem (TSP). For the Metric-k-center problem, we will show that this algorithm gives a 2 factor approximation. The hardness of approximation guarantee discussed in the previous paragraphs tells us that this is the best approximation guarantee that can be achieved unless $\mathsf{P} = \mathsf{NP}$.

> (**Farthest-first-traversal**): Let X denote the given points and $D(.,.)$ be the distance function. Pick the first center arbitrarily from the given points. After having picked $(i-1)$ centers denoted by C_{i-1}, pick a point $p \in X$ to be the i^{th} center c_i such that p is the farthest point from all centers in C_{i-1}. That is $c_i = \arg\max_{x \in X} \left[\min_{c \in C_{i-1}} D(x, c)\right]$.

The next theorem shows that the above algorithm gives a 2-factor approximation.

Theorem 1.1 *For any metric $(\mathcal{X}, D)$ and any dataset $X \subseteq \mathcal{X}$, let C be the k centers produced by the Farthest-first-traversal algorithm and let C^* denote any optimal solution. Then $\Psi_{max}(C, X) \leq 2 \cdot \Psi_{max}(C^*, X)$.*

Proof: Let $y = \arg\max_{x \in X} \{\min_{c \in C} D(x, c)\}$. So, y is the point that is farthest from the set of centers C and let $R = \min_{c \in C} D(y, c)$ be this farthest distance. Consider the set $C' = C \cup \{y\}$. First, observe that for every point $x \in \mathcal{X}$, $D(x, C) \leq R$. Moreover, note that due to the manner in which the centers in the set C are picked, each pair of points in the set C' have distance at least R. Now, at least two points in the set C' share the same nearest center in the set C^*. Let these points be p and q and the center be c^*. Then by triangle inequality, we have $D(p, c^*) + D(q, c^*) \geq D(p, q) \geq R$. So, we have $\max(D(p, c^*), D(q, c^*)) \geq R/2$ which further gives $\Psi_{max}(C^*, X) \geq \Psi_{max}(C, X)/2$. □

Note that the approximation factor of 2 given by the above algorithm holds for both Metric and Euclidean versions of the k-center problem. As mentioned earlier, it has been shown that one cannot get a better approximation than 2 for the Metric version of the problem unless $\mathsf{P} = \mathsf{NP}$. However, for the Euclidean version the known lower bound on the approximation factor is 1.822. So, the following problem remains open to the best of our knowledge: Is it possible to design an efficient algorithm for the Euclidean k-center problem that gives approximation guarantee better than 2? Is it possible to argue that one cannot find an efficient algorithm with approximation guarantee $c > 1.822$ unless $\mathsf{P} = \mathsf{NP}$?

1.3 The k-Means/Median Problem

We discuss the k-Means (problem #7) and k-Median (problems #3 and #5) problems together in this section. The reason for similar treatment of k-means and k-median is because the statement of the these problems are similar. Recall that in the k-means problem the goal is to minimize the sum of squared distances whereas the goal in the k-median problem is to minimize the sum of distances. However, there are some crucial differences that one should note between these problems. One key difference due to squared versus non-squared distance is that in the Euclidean setting, the 1-median problem, also known as the *Geometric Median* problem or a special case of the *Fermat-Weber* problem, is a problem of unknown difficulty [2] whereas the 1-means problem is easy. In fact, there is a closed form expression for the solution of any given 1-means problem. The geometric mean of all the given points is the optimal solution for the problem. This is evident from the following well known fact.

Fact 1.1 *For any set $X \subseteq \mathbb{R}^d$ and any point $p \in \mathbb{R}^d$, we have*

$$\sum_{x \in X} ||x - p||^2 = \sum_{x \in X} ||x - \mu(X)||^2 + |X| \cdot ||p - \mu(X)||^2,$$

where $\mu(X) \stackrel{def.}{=} \frac{\sum_{x \in X} x}{|X|}$ is the geometric mean (or centroid) of the points in the set X.

[2] even though simple approximation algorithms exist.

Another, important difference is due to the triangle inequality that is satisfied when distance is being considered as in the k-median problem as opposed to squared distance that does not satisfy the triangle inequality. [3] This happens to be one of the main reasons why the techniques of Arora *et al.* [7] that work for the Euclidean k-median problem, cannot be made to work for the k-means problem.

Hardness The k-means and the Euclidean k-median problem are simple in one dimension (i.e., $d = 1$). There is a simple Dynamic Programming approach that efficiently solves the 1-dimensional version of these problems. However, the planar version ($d = 2$) and higher dimensional version ($d > 2$) of both the k-means [24, 50, 47] and k-median [48] have been shown to be NP-hard. So, the next natural question is how hard are these problems to approximate? For the k-means problem, Awasthi *et al.* [10] showed that there exists a constant $0 < c < 1$ such that it is NP-hard to approximate the k-means problem to a factor better than $(1 + c)$. For the Euclidean k-median problem, Guruswami and Indyk [32] showed that under standard complexity theoretic assumptions, one cannot obtain an efficient algorithm that gives $(1+\varepsilon)$ approximation for an arbitrary small ε. Similarly for the metric k-median problem, it was shown by Jain *et al.* [38] that there is no efficient algorithm that achieves an approximation factor better than $(1 + 2/e)$ conditioned on some standard complexity theoretic assumption.

Approximation algorithms On the positive side, there are efficient constant factor approximation algorithms for the k-means/median problems. More specifically, for the k-median problem (Metric and Euclidean versions), the best known efficient constant factor approximation algorithm is by Arya *et al.* [9] who gave a $(3 + \varepsilon)$-factor approximation algorithm. Similarly, for the k-means algorithm the best known efficient constant factor approximation algorithm is by Kanungo *et al.* [42] who gave a $(9+\varepsilon)$-factor approximation algorithm. [4] Both the above-mentioned algorithms are based on a technique known as *local search*. We discuss these techniques further in the next few subsections.

As far as approximation schemes are concerned, there are PTAS (Polynomial Time Approximation Scheme)[5] for these problems for special cases when either k or the dimension d is a fixed constant. Let us first discuss the special case where the dimension d is a fixed constant. In a significant technical breakthrough, Arora *et al.* [7] gave a PTAS for the Euclidean k-median problem when d is a constant. The running time of this algorithm was later improved by Kolliopoulos and Rao [43]. Note that the dynamic programming based approach of [7] and [43] did not extend to the k-means problem for reasons mentioned earlier and until recently the problem of obtaining a PTAS for k-means in constant dimension was open. In recent developments, Addad *et al.* [2] and Friggstad *et al.* [29] have obtained PTAS for the k-means problem in constant dimension. For the case when k is a fixed constant, PTAS for the k-means problem has been known. Kumar *et al.* [44] gave the first PTAS for the k-means problems under the assumption that k is a fixed constant. Using different techniques, Feldman *et al.* [28] and Jaiswal *et al.* [40, 41] gave algorithms with improved running time.

Techniques In the remainder of this section, we discuss the k-median and k-means problem with respect to the three main techniques or approaches that are used to obtain approximate solutions for these problems: (i) *Linear Program (LP)* (ii) *Local Search*, and (iii) *Sampling*. We discuss these three approaches in the next three subsections. We will discuss the sampling based approach more elaborately because of the simplicity of the ideas involved that makes the resulting algorithms more practical.

1.3.1 Linear Program and Rounding

Consider the Metric-k-median problem. Note that the centers C are supposed to be a subset of the given point set (i.e., $C \subseteq X \subseteq \mathcal{X}$). This allows us to write a simple Linear Program for this problem. We will denote the given n points with integers $1, ..., n$ and the distance between points i and j with $D(i, j)$. Let y_i

[3] The triangle inequality says that $\forall p, q, r \in \mathbb{R}^d, ||p - q|| + ||q - r|| \geq ||p - q||$. Note that even though the sqaured Euclidean distance does not satisfy the triangle inequality, it does satisfy an *approximate* version $\forall p, q, r \in \mathbb{R}^d, 2 \cdot ||p-q||^2 + 2 \cdot ||q-r||^2 \geq ||p - q||^2$.

[4] The running time of the algorithms in [42] and [9] is polynomial in the input parameters and in $1/\varepsilon$.

[5] Polynomial Time Approximation Schemes are $(1 + \varepsilon)$-factor approximation algorithms for any given ε. The running time of such algorithms is polynomial in the input parameters but can be exponential in $1/\varepsilon$.

Figure 1.1: *A simple one dimensional example for showing that Lloyd's algorithm may not give any approximation guarantees. Consider four points as shown above and let their labels also be their x coordinates. For $k = 3$, the optimal centers are located at a, b, and $(c+d)/2$ with cost $(d-c)^2/2$. However, if the Lloyd's algorithm picks a, c, d as the initial 3 centers, then the algorithm outputs centers $(a+b)/2, c, d$ with cost $(b-a)^2/2$ which may be arbitrarily bad compared to the optimal.*

denote whether the point i is chosen as a center (i.e., $y_i = 1$ when point i is chosen as a center, otherwise 0). Similarly, let x_{ij} denote whether point j is assigned to center i. We can write the following program with respect to these variables:

$$\text{Minimize:} \quad \sum_{i,j} D(i,j) \cdot x_{ij}$$

$$\text{Subject to:} \quad \forall j, \sum_i x_{ij} = 1 \quad \text{and} \quad \forall i,j, x_{ij} \leq y_i \quad \text{and} \quad \sum_i y_i \leq k \quad \text{and} \quad \forall i,j, y_i, x_{ij} \in \{0,1\}$$

The main idea in obtaining approximation algorithm using the above Integer Linear Program (ILP) is to solve a *relaxed* version of the program (i.e., relax the last two integer constraints to $0 \leq x_{ij} \leq 1, 0 \leq y_i \leq 1$), and then *round* the non-integer solution to obtain an approximate solution.[6] Lin and Vitter [45] developed a bicriteria rounding technique that gave a *pseudo-approximation* algorithm where the number of centers produced is $2k$ and the cost is at most four times the optimal cost (with respect to k centers). Such algorithms are also called *bi-criteria* approximation algorithms. Charikar *et al.* [23] gave a rounding algorithm that gives an approximation guarantee of $6\frac{2}{3}$. This was later improved to 3.25 by Charikar and Li [22].

1.3.2 Local Search

Local search is another very highly explored technique in the context of the center based clustering problems. This technique has given some very interesting results. The main idea of local search can be summarised as follows:

> **(Local search)**: Start with a set of k centers and in each step update the set of k centers by making a *local* change. Perform these local changes interatively until the set of centers satisfy some *stability* condition. The local change step is of the following nature: suppose at the start of the step the set of centers is C, then in the current step update the centers from C to C' by performing one (single-swap) or few (multi-swap) *swaps* such that C' is better than C.

Note that due to the nature of the local search algorithm, the centers that are produced by the algorithm are either from among the given points or from among a set of *possible* centers (that may be chosen in a pre-processing step based on the input points). Arya *et al.* [9] showed that multi-swap local search gives a $(3+\varepsilon)$-factor approximation algorithm for the metric k-median problem with polynomial running time. Using similar ideas, Kanungo *et al.* [42] showed that the multi-swap local search gives a $(9+\varepsilon)$-factor approximation for the k-means problem. In more recent developments, Addad *et al.* [2] and Friggstad *et al.* [29] showed that the local search technique gives a PTAS[7] for the k-means problem for the case when d is a fixed constant.

1.3.3 Sampling

Despite all the development in approximation algorithms for the clustering problems that we have discussed here, the algorithms that are used to solve some of these problems in practice are heuristics without any

[6] Note that it is known that the *integrality gap* of the above LP relaxation is at least 2. It is also known from the work of Archer *et al.* [6] that the integrality gap is upper bounded by 3.

[7] As explained earlier, a PTAS is a $(1+\varepsilon)$-approximation where the running time is polynomial in the input parameters but is allowed to be exponential in $1/\varepsilon$.

theoretical guarantees. The reason for using such heuristics over algorithms with guarantees is that these heuristics are extremely simple (so implementation and debugging is simple) and they produce very good results on real world input data. For instance, the heuristic that is used to solve the k-means problem in practice is called the *Lloyd's* algorithm. The algorithm works by starting with k centers picked arbitrarily, and then in a sequence of steps *improving* the k centers. A single improvement step consists of two parts: (i) perform a Voronoi partition with respect to the k centers, and (ii) update the k centers to the centroid of the k Voronoi partitions. This procedure does not give any approximation guarantees. Figure 1.1 gives a simple example to show this. However, if started with k centers with cost at most c times the optimal, then the Lloyd's algorithm trivially gives a c factor approximation since the procedure can only improve the solution. This encourages development of *seeding* algorithms that are fast and simple and that give some approximation guarantee. Such an algorithm, followed by Lloyd's procedure, would provide a "best of theoretical and practical worlds" scenario.

Sampling based techniques gives rise to such extremely simple and practical seeding algorithms. Note that even though we are calling them seeding algorithms, these algorithms are approximate clustering algorithms by themselves since they produce k centers with provably bounded cost. One of the most popular seeding algorithm is called the k-means++ seeding algorithm which in essence is a randomized *sequential-selection* algorithm. This simple sampling procedure also called the D^2-sampling procedure. We will use k-means++ and D^2-sampling interchangeably in this chapter. The description of this algorithm is given below.

> (**k-means++ seeding or D^2-sampling**): Let X denote the given points and $d(.,.)$ be the distance function. Pick the first center randomly from the given points. After having picked $(i-1)$ centers denoted by C_{i-1}, pick a point $p \in X$ to be the i^{th} center with probability proportional to $\min_{c \in C_{i-1}} d(x,c)$.

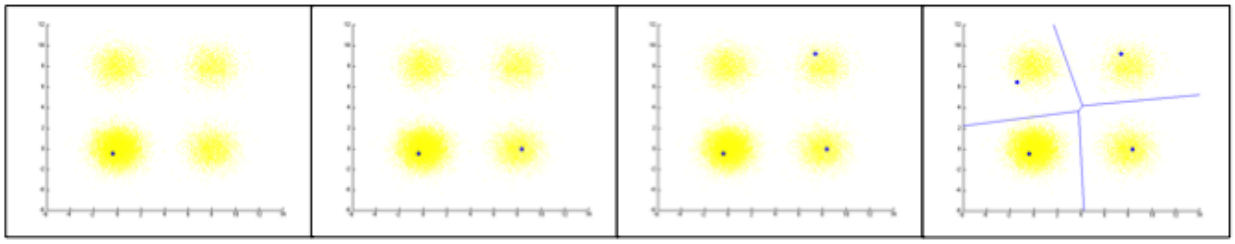

Figure 1.2: *The figure shows a two dimensional data being clustered into four clusters using the sampling algorithm. The algorithm samples the centers in four simple steps. The first center is chosen randomly. For the second center, the points that are further away from the first center is given more priority and so on.*

In the context of the k-means problem, the distance function d is the squared Euclidean distance (hence the name D^2-sampling). This sampling procedure runs fast in practice and has a worst-case running time of $O(nkd)$. Figure 1.2 demonstrates the sampling procedure on a simple two-dimensional problem instance with four clusters. Arthur and Vassilvitskii [8] showed that the above sampling procedure gives an $O(\log k)$-factor approximation guarantee in expectation. The formal statement of their result is given below.

Theorem 1.2 (Theorem 3.1 in [8]) *Let $X \subseteq \mathbb{R}^d$ denote the data points and let C denote the set of k centers produced by the k-means++ seeding algorithm. Then we have $\mathbf{E}[\Phi(C,X)] \leq 8(\ln k + 2) \cdot \Phi_{OPT}(X)$, where $\Phi_{OPT}(X)$ denotes the optimal k-means cost for the input X.*

The above result generalises for cost functions of the following form:

$$\Phi^{\ell}(C,X) = \sum_{x \in X} \min_{c \in C} ||x-c||^{\ell} \tag{1.1}$$

Note that the k-median cost function corresponds to $\ell = 1$. For such cost functions we will use D^{ℓ}-sampling instead of D^2-sampling. The more general result shows approximation guarantee for the D^{ℓ}-sampling procedure.

Theorem 1.3 (Theorem 5.1 in [8]) *Let $X \subseteq \mathbb{R}^d$ denote the data points and let C denote the set of k centers produced by the D^ℓ-sampling algorithm. Then we have $\mathbf{E}[\Phi(C, X)] \leq 2^{2\ell}(\ln k + 2) \cdot \Phi^\ell_{OPT}(X)$, where $\Phi^\ell_{OPT}(X)$ denotes the optimal cost for the input X (defined in equation (1.1)).*

The above results are obtained by making clever use of the next two lemmas in an induction based argument. These lemmas form the foundation for analysing such sampling algorithms. We state these lemmas in the context of the k-means objective function and D^2-sampling (i.e., $\ell = 2$).

Lemma 1.1 (Lemma 3.1 in [8]) *Let A denote points in any optimal k-means cluster of data X. Let C denote a randomly chosen center from A. Then $\mathbf{E}[\Phi(C, A)] \leq 2 \cdot \Delta(A)$. Here $\Delta(A)$ denotes the optimal 1-means cost of A.*

Let $X_1, ..., X_k$ denote the optimal clusters for any dataset X. From Fact 1.1, we get that $OPT(X) = \sum_{i=1}^{k} \Delta(X_i)$. The above lemma says that if a center is chosen uniformly at random from a set X_i of points, then this center gives a 2-factor approximation to the 1-means problem for the point set X_i. This means that if a set C of k points sampled uniformly at random from the dataset X happen to belong to the k optimal clusters $X_1, ..., X_k$, then C is a 2-approximate solution in expectation. However, some thinking reveals that the event that k uniformly sampled points belong to the k optimal clusters might be very unlikely for certain datasets. For example, consider a scenario where $k = 2$ and the dataset X has one large cluster and another extremely small cluster that is very far from the larger cluster. There is a very small chance that uniformly sampled points will belong to the small cluster. On the other hand, if points are sampled using D^2-sampling then the event that points are picked from both large and small cluster is more likely. The first uniformly sampled point will most likely belong to the large cluster. Now, since we are picking the second point with D^2-sampling, a point that is further away from the first point is more likely to be picked as compared to a point that is closer. Recall that given data X and center set C, sampling a center with D^2-sampling means to pick a point from $x \in X$ with probability proportional to the squared Euclidean distance of x from its nearest center in C. So, the second point is more likely to be picked from the smaller cluster. However, in this case, we cannot use Lemma 1.1 since the second sampled point is not a *uniform* sample from the smaller cluster. The next lemma helps out by making a claim similar to Lemma 1.1 but with respect to a center chosen with D^2-sampling.

Lemma 1.2 (Lemma 3.2 in [8]) *Let A denote any optimal k-means cluster of X and let C denote an arbitrary set of centers. Let c denote a center chosen using D^2-sampling with respect to center set C. Then $\mathbf{E}[\Phi(C \cup \{c\}, A) | c \in A] \leq 8 \cdot \Delta(A)$. Here $\Delta(A)$ denotes the optimal 1-means cost of A.*

The above lemma implies that the conditional expectation of the cost of A with respect to the points $C \cup \{c\}$ is at most 8 times optimal given that the sampled point is from the set A. So, as long as the sampled point happens to be from the set A, we get that the set of points $C \cup \{c\}$ gives a constant factor approximation with respect to the points in the set A. Arthur and Vassilvitskii [8] gave a clever induction based argument using Lemmas 1.1 and 1.2 and showed a $O(\log k)$ factor approximation in expectation.

Let us try to use Lemma 1.2 in a weaker but simpler manner to get more intuition into the approximation analysis of the sampling procedure. At a high level, what we can say is that during D^2-sampling, if a chosen center belongs to some optimal cluster A, then we get a constant factor approximation with respect to that cluster. In some sense we can say that A gets "covered" when this happens. So, in the event that all optimal clusters get covered, we get a constant factor approximation guarantee. However, the probability of the event that all optimal clusters get covered by picking k centers may be very small. This can be seen as follows: Consider sampling the i^{th} center after a set of $(i-1)$ centers C_{i-1} have already been chosen with D^2-sampling. Suppose the probability that the point sampled next is from an "uncovered" cluster is small, say at most $1/2$ and this is true for all i. Then the probability of covering all clusters by picking k centers will be at most $1/2^k$. The next lemma shows that in any step of D^2-sampling, the probability of sampling the next point from a currently uncovered cluster cannot be too small unless the current set of centers already give a constant factor approximation (so picking more points will make the solution only better).

Lemma 1.3 (Lemma 2.7 in [5]) *Let X denote the dataset and $X_1, ..., X_k$ denote the k optimal k-means clusters. Let C denote any set of centers. Let I denote the subset of indices of clusters that are covered*

w.r.t. center set C. That is, for every $i \in I$, we have $\Phi(C, X_i) \leq c \cdot \Delta(X_i)$ for some fixed constant c. Let $x \in X_j$ denote the next point that is sampled with D^2-sampling. Then at least one of following statements is true: (i) $\mathbf{Pr}[j \notin I] \geq 1/2$, (ii) $\Phi(C, X) \leq (2c) \cdot \Phi_{OPT}(X)$.

Proof: Suppose the first statement is false, then we will show that the second statement will be true. Let $X_c = \cup_{i \in I} X_i$ and $X_u = X \setminus X_c$. Then we have: $1/2 > \mathbf{Pr}[j \notin I] = \frac{\Phi(C, X_u)}{\Phi(C, X_u) + \Phi(C, X_c)} \Rightarrow \Phi(C, X_c) > \Phi(C, X_u)$. Using this inequality, we get $\Phi(C, X) = \Phi(C, X_u) + \Phi(C, X_c) < 2 \cdot \Phi(C, X_c) \leq (2c) \cdot \sum_{i \in I} \Delta(X_i) \leq (2c) \cdot \sum_{i=1}^{k} \Delta(X_i) \leq (2c) \cdot \Phi_{OPT}(X)$. □

The above lemma implies that the k-means++ algorithm gives constant approximation with probability at least $\frac{1}{2^{O(k)}}$. The lemma also implies that if we sample $O(k \log k)$ centers with D^2-sampling (instead of k centers) and compare the cost with the optimal cost with respect to k centers, then we get a constant factor *pseudo-approximation*. This observation was made by Ailon *et al.* [5]. With a little more work using martingale analysis, Aggarwal *et al.* [4] showed that D^2-sampling gives a constant pseudo-approximation even when only $O(k)$ centers are sampled.

Arthur and Vassilvitskii [8] had showed that D^2-sampling procedure gives an $O(\log k)$ approximation in expectation. They also support their upper bound on the approximation guarantee with a matching lower bound of $\Omega(\log k)$. That is, they give a dataset X such that the k-means++ seeding algorithm when executed on X gives a solution with expected cost $\Omega(\log k)$ times the optimal.

After the initial results on the analysis of the sampling algorithm by Arthur and Vassilvitskii [8], there has been a huge volume of work on understanding the D^2-sampling technique. We give a brief review of all these developments in the next few paragraphs.

Pseudo-approximation Unlike the Lloyd's algorithm where the number of clusters is used crucially in the algorithm, the sampling algorithm uses it just as a termination condition. That is, it terminates when it has sampled k centers. An interesting property of the sampling algorithm is that if one considers the first i centers C_i that are sampled, then C_i gives a $O(\log i)$ factor approximation for the i-means problem for the dataset. In this scenario, it makes sense to consider sampling more than k centers and then compare the solution with the optimal solution with respect to k centers. Such algorithms are known as *pseudo-approximation* algorithms. We have already mentioned the results of Ailon *et al.* [5] and Aggarwal *et al.* [4] who showed constant pseudo-approximation with $O(k \log k)$ and $O(k)$ centers respectively. In a more recent work, Wei [51] showed that if βk centers are sampled for any constant $\beta > 1$, then we get a constant factor pseudo-approximation in expectation.

Lower bound on approximation The results of Arthur and Vassilvitskii [8], left a natural open question of whether the k-means++ seeding gives better than $O(\log k)$ approximation (say constant factor) with probability that is not too small (say $poly(1/k)$). Brunsch and Röglin [20] showed that this is not possible. More specifically, they created an instance where the sampling procedure gives approximation ratio of $(2/3 - \varepsilon) \log k$ with probability that is exponentially small in k. However, the instance that they gave was a high-dimensional instance and they left an important open problem of analysing the sampling procedure for small dimensional instances. In fact, Brunsch and Röglin [20] conjectured that for constant dimensional instances, the sampling procedure gives a constant factor approximation with not too small probability. This conjecture was refuted by Bhattacharya *et al.* [19] who gave a two dimensional instance such that the sampling procedure gives an $O(\log k)$ approximation on that instance with probability that is exponentially small in k.

D^2-sampling under separation Qualitatively, the D^2-sampling procedure samples points that are far from each other. Consider the case when the optimal clusters are "separated" in some sense. If a center c from one of the optimal clusters X_i have been picked, then the probability of choosing another center from the same cluster would be low since the other points in X_i will typically be closer to c as compared to points from other clusters. So, in such cases it is likely that the sampling procedure will pick one center from each of the optimal cluster and given this, we know from Lemma 1.2 that the solution will be good. In order to formulate such results, we first need to define an appropriate notion of separation of clusters. One popular notion of separation was defined by Ostrovsky *et al.* [49]. This was in terms of the gap between

the cost of the optimal solution with respect to k centers and that with respect to $k-1$ centers. If the ratio of the latter to the former is large, then this implies some kind of separation between clusters since clustering into k cluster is much better than clustering into $k-1$ clusters. Jaiswal and Garg [39] showed that under this kind of separation guarantee, the k-means++ seeding procedure indeed gives a constant approximation with probability $\Omega(1/k)$. Another notion of separation was given by Balcan *et al.* [14] and is known as *approximation stability*. A dataset is said to satisfy approximation stability iff for every clustering with cost near the optimal cost is "close" to the target optimal clustering. The *closeness* is in terms of the number of points that need to be reassigned to resemble the optimal clustering. Agrawal *et al.* [3] showed that if the dataset X satisfies approximation stability and if all optimal clusters have some minimum size, then the D^2-sampling algorithm gives a constant factor approximation with probability $\Omega(1/k)$.

PTAS using D^2-sampling Polynomial Time Approximation Scheme (PTAS) is an algorithm that takes as input an error parameter ε in addition to the problem and outputs a solution with an approximation guarantee of $(1+\varepsilon)$. The running time of such an algorithm should be polynomial in the input parameters (but is allowed to be exponential in the error parameter ε). Such algorithms allow us to obtain solutions with cost that is arbitrarily close to the optimal cost. As we noted earlier while discussing the hardness for the k-means problem (and also the k-median problem), such algorithms are not possible. However, when k or d is not part of the input and is a fixed constant, then there do exist PTAS for these problems. We noted earlier that a local search based algorithm is a PTAS under the assumption that d is not part of the input. Here, we will see a simple D^2-sampling based PTAS under the assumption that k is not part of the input. These are results from Jaiswal *et al.* [40, 41]. We will discuss the k-means problem and later note that the techniques generalise to k-median and other settings. The main lemma that we will use in the discussion is the following lemma by Inaba *et al.* [34].

Lemma 1.4 ([34]) *Let S be a set of points obtained by independently sampling M points with replacement uniformly at random from a point set $X \subset \mathbb{R}^d$. Then for any $\delta > 0$,*

$$\mathbf{Pr}\left[\Phi(\{\Gamma(S)\}, X) \leq \left(1 + \frac{1}{\delta M}\right) \cdot \Delta(X)\right] \geq (1-\delta).$$

Here $\Gamma(S)$ denotes the geometric centroid of the set S. That is $\Gamma(S) = \frac{\sum_{s \in S} s}{|S|}$

Fact 1.1 tells us that the centroid of any point set $X \subset \mathbb{R}^d$ is the optimal 1-means solution for X. What the above lemma tells us is that the centroid of a uniform sample from any point set *approximates* the 1-means solution to within a factor of $(1+\varepsilon)$ if the size of the sample is $\Omega(1/\varepsilon)$. This implies that there exits at least one such subset among the possible $n^{O(1/\varepsilon)}$ subsets for which this holds. Worah and Sen [52] showed how to find such a subset using efficient derandomization. The situation here is different since we do not even know the clusters.

Let X denote any dataset and let $X_1, ..., X_k$ denote an optimal k-means clusters of X. Suppose we manage to isolate uniform samples $S_1, ..., S_k$ from $X_1, ..., X_k$ respectively such that $\forall i, |S_i| = \Omega(1/\varepsilon)$. Then, from the above lemma, we get that the centers $(c_1, ..., c_k)$ where $c_i = \Gamma(S_i)$ is a $(1+\varepsilon)$-approximate solution. Let us now try to think about how we can possibly perform this task. Without loss of generality, let us assume that X_1 is the largest cluster (in terms of the number of points). Isolating a uniform sample from X_1 is simple may be done in the following manner:

> Sample a set U uniformly at random (with replacement) from X such that $|U| = poly(k/\varepsilon)$. Let $P_1, ..., P_m$ denote all the $\binom{|U|}{\lceil 1/\varepsilon \rceil}$ subsets [8] of size $\lceil \frac{1}{\varepsilon} \rceil$ of the set U. Since X_1 is well represented in the sample U (note that at least $\frac{1}{k} \cdot |U| = \Omega(\frac{1}{\varepsilon})$ elements in U will belong to X_1 in expectation), it will not be too difficult to argue that at least one of the subsets $P_1, ..., P_m$ will represent a uniform sample from X_1. S_1 denotes that subset.

Note that even though we are not able to isolate a single uniform sample S_1 from X_1, we are able to produce a *list* of samples such that one of them is such a sample. This is sufficient for developing an

[8] This can be bounded by $2^{O\left(\frac{1}{\varepsilon} \log \frac{|U|}{\varepsilon}\right)}$ using the inequality $\binom{x}{y} \leq (\frac{ex}{y})^y$.

`Find-`k`-means`(X, k, ε)
- Let $N = \Theta(\frac{k}{\epsilon^2})$, $M = \Theta(\frac{1}{\varepsilon})$ and initialize $\mathcal{L}$ to $\emptyset$.
- Repeat 2^{2k} times: Make a call to `Sample-centers`$(X, k, \epsilon, 0, \{\})$.
- Return the set of k centers from $\mathcal{L}$ that has least cost.

`Sample-centers`(X, k, ϵ, i, C)
(1) If $(i = k)$ then add C to the set $\mathcal{L}$.
(2) else
  (a) Sample a multi-set S of N points with D^2-sampling (w.r.t. centers C)
  (b) For all subsets $T \subset S$ of size M:
    (i) $C \leftarrow C \cup \{\Gamma(T)\}$.
    (ii) `Sample-centers`$(X, k, \epsilon, i + 1, C)$

Figure 1.3: $(1 + \varepsilon)$-approximation algorithm for k-means. Note that $\Gamma(T)$ denotes the centroid of T.

algorithm. What we can do is calculate the centroid of each of the subsets and try each of these centers as a possible first center. The next question is: how do we compute the uniform samples $S_2, ..., S_k$ from $X_2, ..., X_k$? Note that uniformly sampling from X and trying all possible subsets (as we did for S_1) may not work since $X_2, ..., X_k$ may be tiny clusters and a uniformly sampled set from X may not have adequate representatives from $X_2, ..., X_k$. This is where D^2-sampling becomes helpful. Suppose at this stage, we sample a set U with D^2-sampling. We can now argue that there is a good chance that U has adequate representation from $X_2, ..., X_k$ and so at least one of its subsets of size $\lceil \frac{1}{\varepsilon} \rceil$ has points entirely from one of $X_2, ..., X_k$. Moreover, we can argue that if the size of U is chosen carefully, at least one of the subsets of size $\lceil \frac{1}{\varepsilon} \rceil$ will resemble a uniform sample from one of $X_2, ..., X_k$. Hence, the centroid of that set will be a good addition to the set of centers from Lemma 1.4. We continue this argument to show that one set of k centers will have a good center from each of the k-optimal clusters. Since this is a branching algorithm with a branch factor of $2^{\tilde{O}(1/\varepsilon)}$ and a depth of k, the running time of the algorithm is $O(nd \cdot 2^{\tilde{O}(k/\varepsilon)})$.[9] The detailed algorithm is given in Figure 1.3. We give an analysis of this algorithm showing $(1 + \varepsilon)$-approximation in sufficient detail below. Following is the main result that we show:

Theorem 1.4 *Let $0 < \varepsilon \leq 1/2$, k be a positive integer, and $X \subseteq \mathbb{R}^d$. Then* `Find-`k`-Means`(X, k, ε) *runs in time $O(|X|d \cdot 2^{\tilde{O}(k/\varepsilon)})$ and gives a $(1 + \varepsilon)$-approximation to the k-means problem.*

The algorithm `Find-`k`-Means` maintains a set C of centers, that is initially empty. Every recursive call to the function `Sample-centers` increments the size of C by one. In step (2) of the sub-routine `Sample-centers`, it *tries out* various candidates that can be added to C. First, it samples a multi-set S of size $N = O(k/\varepsilon^3)$ by D^2-sampling from X with respect to center set C. Then it considers all subsets of S of size $M = O(1/\varepsilon)$ - for each such subset it adds the centroid of the subset to C and makes a recursive call to itself. Thus each invocation of `Sample-centers` makes precisely $\binom{N}{M}$ recursive calls to itself. It will be useful to think of the execution of this algorithm as a tree $\mathcal{T}$ of depth k. Each node in $\mathcal{T}$ can be labeled with a set C – it corresponds to the invocation of `Sample-centers` with this set as C (and i being the depth of this node). The children of a node denote the recursive calls made by the corresponding invocation of `Sample-centers`. Finally, the leaves denote the set of candidate centers produced by the algorithm.

Let $X_1, ..., X_k$ denote the optimal k-means clustering of the given dataset X and let $OPT(X)$ denote the optimal k-means cost for dataset X. As defined before, let $\Delta(R)$ denote the optimal 1-means cost for any dataset R. Note that $OPT(X) = \sum_{r=1}^{k} \Delta(X_r)$. A node v at depth i in the execution tree $\mathcal{T}$ corresponds to a set C of size i. Let us call this set C_v. Our proof will argue inductively that for each i, there will be a node v at depth i such that the centers chosen so far in C_v are good with respect to a subset of i distinct clusters among $X_1, ..., X_k$ or the entire set X. More specifically, we will argue that the following invariant $P(i)$ is maintained during the recursive calls to **Sample-centers**:

> **P(i)**: With probability at least $\frac{1}{2^{i-1}}$, there is a node v_i at depth $(i - 1)$ in the tree $\mathcal{T}$ such that either **(a)** $\Phi(C_{v_i}, X) \leq (1 + \varepsilon) \cdot OPT(X)$, or **(b)** there is a set of $(i - 1)$ distinct clusters

[9] Here $\tilde{O}$ hides logarithmic factors in k and ε. Also, the branching structure will be more clear when we discuss the algorithm in detail below.

$X_{j_1}, X_{j_2}, ..., X_{j_{i-1}}$ such that: $\Phi(C_{v_i}, X_{j_1} \cup ... \cup X_{j_{i-1}}) \leq (1 + \frac{\varepsilon}{2}) \cdot \sum_{r=1}^{i-1} \Delta(X_{j_r})$.

Theorem 1.4 follows from the above since $P(k+1)$ holds and 2^{2k} repeated calls to the `Sample-centers` procedure are made and the best set of k centers picked. We prove the invariant using induction. Note that $P(1)$ is trivially true since for the root v of the tree, C_v is the empty set. Now assume that $P(i)$ holds for some $i \geq 1$. We will show that $P(i+1)$ also holds. For this, we condition on the event $P(i)$ (that happens with probability at least 2^{i-1}). Note that if $\Phi(C_i, X) \leq (1+\varepsilon) \cdot OPT(X)$, then $P(i+1)$ is trivially true. So, for the rest of the discussion we will assume that this does not hold. Let v_i and $X_{j_1}, ..., X_{j_{i-1}}$ be as guaranteed by the invariant $P(i)$. For ease of notation and without loss of generality, let assume that the index j_r is r and let us call C_{v_i} as just C_i. So, we have good approximation with respect to points in $X_1 \cup ...X_{i-1}$ and these cluster may be thought of as "covered" clusters. Note that the probability of D^2-sampling a point from one of the uncovered clusters $X_i, ..., X_k$, say X_r, is equal to $\frac{\Phi(C_i, X_r)}{\Phi(C_i, X)}$. Let $\bar{i} \in \{i, i+1, ..., k\}$ be the index for which $\Phi(C_i, X_{\bar{i}})$ is the largest. We can break the analysis into the following two parts – (i) $\frac{\Phi(C_i, X_{\bar{i}})}{\Phi(C_i, X)} \leq \frac{\varepsilon}{2k}$, and (ii) $\frac{\Phi(C_i, X_{\bar{i}})}{\Phi(C_i, X)} > \frac{\varepsilon}{2k}$. In the former case, we will argue that $\Phi(C_i, X) \leq (1+\varepsilon) \cdot OPT(X)$, that is, the current set of centers already gives $(1+\varepsilon)$-approximation for the entire dataset and hence $P(i+1)$ holds. In the latter case, we will argue that sufficient number of points will be sampled from $X_{\bar{i}}$ when sampling is done using D^2-sampling and the added center corresponding to at least one of the branches incident on v_i in the tree $\mathcal{T}$, will be a good center for $X_{\bar{i}}$ with probability at least $1/2$. Hence, $P(i+1)$ holds again. These two cases are discussed in the next two lemmas.

Lemma 1.5 *Let $0 < \varepsilon \leq 1/2$. If $\frac{\Phi(C_i, X_{\bar{i}})}{\Phi(C_i, X)} \leq \frac{\varepsilon}{2k}$, then $\Phi(C_i, X) \leq (1+\varepsilon) \cdot \sum_{r=1}^{k} \Delta(X_r) = (1+\varepsilon) \cdot OPT(X)$.*

Proof: Since $\frac{\Phi(C_i, X_{\bar{i}})}{\Phi(C_i, X)} \leq \frac{\varepsilon}{2k}$, we have $\frac{\sum_{r=i}^{k} \Phi(C_i, X_r)}{\sum_{r=1}^{i-1} \Phi(C_i, X_r) + \sum_{r=i}^{k} \Phi(C_i, X_r)} \leq \frac{\varepsilon}{2}$ which implies $\sum_{r=i}^{k} \Phi(C_i, X_r) \leq \frac{\varepsilon/2}{1-\varepsilon/2} \cdot \sum_{r=1}^{i-1} \Phi(C_i, X_r)$. We use this inequality to prove the lemma in the following manner:

$$\Phi(C_i, X) = \sum_{r=1}^{i-1} \Phi(C_i, X_r) + \sum_{r=i}^{k} \Phi(C_i, X_r) \leq \frac{1}{1-\frac{\varepsilon}{2}} \cdot \sum_{r=1}^{i-1} \Phi(C_i, X_r) \leq \frac{1+\frac{\varepsilon}{2}}{1-\frac{\varepsilon}{2}} \cdot \sum_{r=1}^{i-1} \Delta(X_r) \leq (1+\varepsilon) \cdot OPT(X).$$

Note that the second to last inequality follows from the invariant. □

The above lemma handles case (i). Let us now discuss case (ii). So, we have $\frac{\Phi(C_i, X_{\bar{i}})}{\Phi(C_i, X)} > \frac{\varepsilon}{2k}$. What this essentially says is that the set S in the `Find-k-Means` algorithm that is obtained by D^2-sampling with respect to center set C_i will have good representation from $X_{\bar{i}}$ as long as $N = \Theta(k/\varepsilon^2)$ (since the expected number of points in the set S from $X_{\bar{i}}$ will be $\Omega(1/\varepsilon)$). So, the important question now is that if we consider all possible subsets of S of size $M = \Theta(1/\varepsilon)$, are we guaranteed that at least one of these subsets will be a uniform sample from $X_{\bar{i}}$? If this were true, then we can straightaway apply Inaba *et al.* 's lemma (Lemma 1.4) and argue that the centroid of such a sample will "cover" $X_{\bar{i}}$ and so $P(i+1)$ holds. However, this may not be true since there may be points in $X_{\bar{i}}$ that are very close to centers in C_i and hence have very small probability of being sampled. In order to handle this issue, we partition the set $X_{\bar{i}}$ into two sets, $X_{\bar{i}}^{near}$ and $X_{\bar{i}}^{far}$ denoting the points that are near to the centers in C_i and those that are far respectively. More specifically, a point $x \in X_{\bar{i}}$ belongs to $X_{\bar{i}}^{far}$ iff $\frac{\Phi(C_i, \{x\})}{\Phi(C_i, X_{\bar{i}})} > \frac{\varepsilon}{16} \cdot \frac{1}{|X_{\bar{i}}|}$. This means that given that the sampled point is from $X_{\bar{i}}$, the conditional probability of sampling any given point $x \in X_{\bar{i}}^{far}$ is at least $\frac{\varepsilon}{16}$ fraction of the probability if it were sampled uniformly from $X_{\bar{i}}$. So, what we can argue for points in $X_{\bar{i}}^{far}$ is that if $N = |S|$ is chosen carefully, there is a good chance that one of the subsets of S will be a uniform sample from $X_{\bar{i}}^{far}$ and given this, the center corresponding to this subset will provide a $(1 + \frac{\varepsilon}{4})$ approximation for the set $X_{\bar{i}}^{far}$. We omit the details of this analysis which may be found in [40]. As for points in $X_{\bar{i}}^{near}$, we will argue that the cost of these points with respect to centers in C_i is small (at most $\frac{\varepsilon}{4} \cdot \Delta(X_{\bar{i}})$) and so they do not contribute to the overall cost (even if no further centers are chosen). So, given that a good center for $X_{\bar{i}}^{far}$ is chosen (this happens with high probability), the overall cost of $\bar{X}_i$ will be at most $(1 + \frac{\varepsilon}{4}) \cdot \Delta(X_{\bar{i}}^{far}) + \frac{\varepsilon}{4} \cdot \Delta(X_{\bar{i}}) \leq (1 + \frac{\varepsilon}{2}) \cdot \Delta(X_{\bar{i}})$. The details of this part of the analysis is given in the next lemma which is a combination of Lemmas 2 and 3 in [41]).

Lemma 1.6 $\Phi(C_i, X_{\bar{i}}^{near}) \leq \frac{\varepsilon}{4} \cdot \Delta(X_{\bar{i}})$.

Proof: Let $D(p,q)$ denote the squared Euclidian distance between p and q. Let $c_{\bar{i}}$ denote the centroid of $X_{\bar{i}}$. For any point $x \in X_{\bar{i}}^{near}$ let $c_x = \arg\min_{c \in C_i}\{D(c,x)\}$. We have:

$$\frac{\varepsilon}{16} \cdot \frac{1}{|X_{\bar{i}}|} \geq \frac{\Phi(C_i,\{x\})}{\Phi(C_i,X_{\bar{i}})} \geq \frac{D(x,c_x)}{\Phi(c_x,X_{\bar{i}})} \overset{Fact\ 1.1}{=} \frac{D(x,c_x)}{\Delta(X_{\bar{i}}) + |X_{\bar{i}}| \cdot D(c_x,c_{\bar{i}})} \geq \frac{D(x,c_x)}{\Delta(X_{\bar{i}}) + 2|X_{\bar{i}}| \cdot (D(x,c_x) + D(x,c_{\bar{i}}))}.$$

The last inequality uses the fact that for any three points $p,q,r, D(p,q) \leq 2(D(p,r) + D(r,q))$. Rearranging the terms of the above inequality, we get that: $D(x,c_x) \leq \frac{\varepsilon/16}{1-\varepsilon/16} \cdot \left(\frac{\Delta(X_{\bar{i}})}{|X_{\bar{i}}|} + D(x,c_{\bar{i}})\right) \leq \frac{\varepsilon}{8} \cdot \left(\frac{\Delta(X_{\bar{i}})}{|X_{\bar{i}}|} + D(x,c_{\bar{i}})\right)$. Using this, we can get a bound on $\Phi(C_i, X_{\bar{i}}^{near})$ as follows:

$$\Phi(C_i, X_{\bar{i}}^{near}) = \sum_{x \in X_{\bar{i}}^{near}} D(x,C_i) \leq \sum_{x \in X_{\bar{i}}^{near}} \frac{\varepsilon}{8} \cdot \left(\frac{\Delta(X_{\bar{i}})}{|X_{\bar{i}}|} + D(x,c_{\bar{i}})\right) \leq \frac{\varepsilon}{4} \cdot \Delta(X_{\bar{i}}).$$

This completes the proof of the lemma. □

This D^2-sampling technique generalizes for the Euclidean k-median problem and we have a similar result as above. Interestingly, these techniques generalizes for distance measures other than Euclidean and squared Euclidean distance such as the Mahalanobis distance and μ-similar Bregman divergences (see [1] for discussion on these distance functions). The analysis uses only simple properties of the distance measure such as (approximate) symmetry and (approximate) triangle inequality which causes the analysis to carry over to these settings.

The D^2-sampling technique also extends to the setting of *constrained clustering* which is relevant in many machine learning applications. In many clustering scenarios, optimising the k-means cost function is not the only objective. Depending on the clustering context there may be additional constraints. For example, consider the *r-gather problem* where the additional constraint is that each cluster should have at least r points. Ding and Xu [25] gave a general framework in which such algorithms can be studied and also gave a $(1+\varepsilon)$-approximation algorithm (not based on D^2-sampling). Note that in such scenarios, the separation between optimal clusters is not as well defined as in the classical k-means problem (where the clusters are optimal Voronoi partitions). However, even under such scenarios the algorithm based on D^2-sampling can be made to work with minimal changes in the overall outline. This is one of the main results by Bhattacharya *et al.* [18]. Let us go back to the outline of the algorithm and see the main issue. After finding good centers for clusters $X_1, ..., X_i$, we sample point set U using D^2-sampling. This is done to isolate a uniform sample from one of the remaining clusters $X_{i+1}, ..., X_k$. However, some of the points of these clusters may be very close to the i good centers that we have already picked and hence will have low chance of being sampled. This is handled by taking appropriate copies of the already chosen centers in the set U (which act as proxies for points in remaining clusters that are close to the already chosen centers). The analysis becomes slightly tricky than before but it goes through without changing the basic outline of the algorithm for the classical k-means problem as in [40].

Variations of k-means++ seeding Due to its simplicity, many practitioners have been interested in the D^2-sampling technique and have used it in various data mining tasks. A number of optimizations have also been suggested that make the sampling procedure more efficient in various contexts. One such context is parallel computing. With advancement in parallel architectures there is a move towards developing parallel algorithms for popular data mining tasks such as clustering. It would be nice if we can come up with a parallel version of the k-means++ seeding algorithm. However, we note that the sampling algorithm is inherently sequential. This is because the sampling of the i^{th} center depends on the previous $i-1$ centers. So, since a minimum of k sequential steps are required, parallelism cannot give arbitrary running time improvements. Bahmani *et al.* [13] gave a slightly modified version of the algorithm that they called the k-means|| (k-means *parallel*). The main idea is that in the i^{th} iteration, instead of sampling a single point, sample ℓ points (for some $\ell > 1$) in parallel and run this algorithm for t iterations (for some $t < k$). Finally, the sampled points are *pruned* to obtain k centers. On the experimental side, it was shown that the behaviour of this algorithm is similar to that of the k-means++ even when t is very small and on the theoretical side, they showed a constant approximation guarantee under some reasonable assumption on the lower bound on t.

The D^2-sampling algorithm runs in k iterations, one center being picked in each iteration. Note that the running time of each of these iterations is $O(nd)$. This is because, at the start of every iteration, we need to compute the distribution from which the next center will be sampled and for this we need to compute the distance of every point to its nearest center (from among the centers chosen until that iteration). Bachem *et al.* [11] suggest a simple Monte Carlo Markov Chain (MCMC) based technique to improve the running time in every iteration. Their algorithm is called the k-MC^2 algorithm. The main idea is that in order to sample $x \in X$ with probability $p(x)$, we perform a bounded-length random walk on a Markov chain where the states denote points in X and we make a transition from x to y with probability $\min(1, p(y)/p(x))$. Note that $p(x), p(y)$ is proportional to the squared distance of x and y from their nearest centers respectively. Since the transition only requires the ratio of these quantities, we do not need to compute the distance of every point from its nearest center. So, the running time in every iteration of the sampling algorithm is proportional to the length of the random walk performed. This may be much smaller than n in many different contexts. In a follow-up paper, Bachem *et al.* [12] gave an improvement where we can put an upper bound on the length of the random walk at a small cost to the approximation guarantee.

Streaming Algorithm In typical data mining tasks where these clustering algorithms are typically used, the data is so large that the standard random access model of computation (where data is assumed to be in the memory) is not appropriate. In such large-data scenarios the algorithm typically gets to see the data by making one pass (or a few passes) over the data. This model of computation is called the *streaming model* since the data is accessible as a *stream*. The question in this context is whether we can solve the center based clustering problems in the streaming setting where the memory available is small compared to the number of data points. The tradeoff of interest in this setting is that between the approximation guarantee and the amount of space required by the streaming algorithm. There is a standard method of using classical approximation algorithms to design approximation algorithms in the streaming setting where the memory is limited. Let A, B be an approximation algorithms for the *weighted* k-means problem that gives an approximation factor of α, β respectively and use linear amount of memory. The weighted k-means problem is a simple generalization of the k-means problem where each point has an associated weight and the goal is to minimize the sum of squared distances multiplied by the weights (instead of just the squared distances as in k-means). Using A and B we will design a streaming algorithm that uses $O(k\sqrt{n})$ memory. Imagine having a bucket that can store at most $\sqrt{n}$ data points. As we make a pass over the data, we keep filling this bucket. Once the bucket is full, we execute A on the data points in the bucket and find k centers. We empty the bucket and keep these weighted k centers separately, the weight of a center being the number of points in the bucket for which this center is the closest. We continue with the data stream, repeating the same procedure as above. Once all the n points have been seen, we go back to the weighted $O(k\sqrt{n})$ centers that have been saved and use algorithm B to cluster these weighted centers and obtain the final k centers. Note that the total memory used is $O(k\sqrt{n})$ and the algorithm just makes one pass over the data. Guha *et al.* [31] showed that the approximation guarantee obtained by using algorithms A and B in such a manner (that is, using B on the output of A) is $2\alpha + 4\beta(2\alpha + 1)$. Also, algorithm A may be a pseudo-approximation algorithm. Ailon *et al.* [5] used the sampling based pseudo-approximation algorithm in the above general technique to obtain a streaming algorithm that for any constant $\varepsilon > 0$ uses $O(n^\varepsilon)$ space and gives $O(c^{1/\varepsilon} \log k)$ approximation guarantee for some constant c. Note that this is not the best tradeoff that is known between space and approximation guarantee and there are better tradeoffs known [21] using other techniques.

1.4 Other Topics on Clustering

Even tough the title of this Chapter is Approximate Clustering, there are many important topics on clustering that we did not discuss. The simple reason for this is that the literature on clustering has now become so vast that a comprehensive coverage in a book chapter is not possible. There are a few books and surveys [33, 37, 36, 16, 35] on various clustering topics that the reader may find useful. In this chapter, we discussed center-based clustering problems such as k-center/means/median and even within this topic, our main focus was on sampling based sequential selection algorithms. Given that we have left out discussions on many clustering topics, it will be useful to mention at least a few of them here.

It is important to note that the center-based clustering problems provide just one way to partition the data into meaningful clusters. These problems make sense in contexts where the target clusters satisfy *locality property*, meaning that the points within the same cluster are all close to each other (or *tightly packed* in some sense). However, there are other contexts where the meaningful clusters do not satisfy such properties and in these scenarios, modelling the clustering problem in terms of center-based problems such as k-center/means/median may not give the best results. The Figure 1.4 (left figure), shows one such scenario. In such contexts, a *density based model* makes much more sense. The main idea is to keep neighbouring

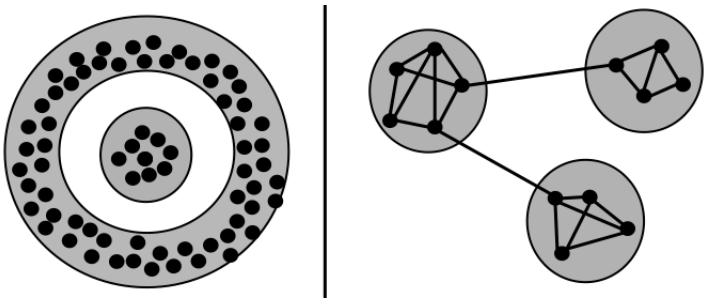

Figure 1.4: (a) The figure on the left shows a simple two dimensional dataset where the two meaningful clusters is unlikely to be found using center-based clustering algorithms, (b) The figure on the right shows a simple graph clustering problem.

points in the same cluster even if all points within a cluster are not close to each other as in the scenario above. DBSCAN [26] is a density based algorithm that is widely used such scenarios.

Data Clustering can also be modelled as a graph theoretic problem in the following manner: For a given dataset along with similarity measure, we can define a graph where the vertices correspond to the data points and there is an edge between vertices x and y iff similarity exceeds some threshold and the weight of the edge equals the similarity. Modelling in terms of a graph also makes sense in contexts where the similarity/dissimilarity between data points are known at a coarse level such as for every pair x, y all that is known is whether x and y are similar or not. The high-level goal is to find a clustering of the vertices such that the sum of weight of the edges that go across clusters is minimized (or in other words find *cuts* with small capacity). Figure 1.4 (right figure) shows a simple graph clustering problem where all edge weights are 1. Note that in this formulation, the number of clusters k may not be given as input. *Spectral techniques* [46] are popularly used for such cut problems in graphs. Correlation clustering [15] may be described as a more general version of the graph clustering problem discussed in the previous paragraph. Here the edges in the graph may have positive weight (indicating similarity) or negative weight (indicating dissimilarity). The goal is to find a partition of the vertices into clusters such that sum of weights of positive edges within clusters minus the sum of weights of negative edges across clusters is maximized.

References

[1] Marcel R. Ackermann and Johannes Blömer. Coresets and Approximate Clustering for Bregman Divergences. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'09)*, 2009.

[2] Vincent Cohen-Addad, Philip N. Klein, and Claire Mathieu. Local search yields approximation schemes for k-means and k-median in Euclidean and minor-free metrics. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS'16)*, 2016.

[3] Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. k-means under approximation stability. *Theoretical Computer Science*, Volume 588, Pages 37–51, 2015.

[4] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 5687 of *Lecture Notes in Computer Science*, pages 15–28. Springer Berlin Heidelberg, 2009.

[5] Nir Ailon, Ragesh Jaiswal, and Claire Monteleoni. Streaming k-means approximation. In *NIPS*, pages 10–18. 2009.

[6] A. Archer, R. Rajagopalan, and D. B. Shmoys. Lagrangian relaxation for the k-median problem: new insights and continuity properties. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, pages 31–42, 2003.

[7] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC '98)*. ACM, New York, NY, USA, pages 106-113, 1998

[8] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[9] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local Search Heuristics for k-Median and Facility Location Problems. *SIAM J. Comput. 33, 3 (March 2004)*, pages 544-562.

[10] Pranjal Awasthi and Moses Charikar and Ravishankar Krishnaswamy and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In *the 31st International Symposium on Computational Geometry (SoCG'15)*, pages 754–767, 2015.

[11] Approximate k-Means++ in Sublinear Time. Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. In *the 30th AAAI Conference on Artificial Intelligence*, 2016.

[12] Fast and Provably Good Seedings for k-Means. Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. In *NIPS*, 2016.

[13] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. In *Proc. VLDB Endow. 5, 7 (March 2012)*, pages 622-633, 2012.

[14] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Clustering under approximation stability. *J. ACM* 60(2) 8:1?8:34, 2013.

[15] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation Clustering. *Machine Learning*, 56: 89, 2004.

[16] P. Berkhin. A survey of clustering data mining techniques. In: *Grouping Multidimensional Data*, pp. 25 ? 71. Springer (2006)

[17] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. *Approximation Algorithms for NP-hard Problems, D. Hochbaum, ed., PWS Publishing*, pp. 296–345, 1996.

[18] Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Faster Algorithms for the Constrained k-Means Problem. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, pages 16:1–16:13, 2016.

[19] Anup Bhattacharya, Ragesh Jaiswal, Nir Ailon. Tight lower bound instances for k-means++ in two dimensions. *Theoretical Computer Science*, Volume 634, Pages 55-66, 2016.

[20] Tobias Brunsch and Heiko Röglin. A bad instance for k-means++. *Theoretical Computer Science*, 505(0):19 – 26, 2013.

[21] Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (STOC '03)*. ACM, New York, NY, USA, 30-39, 2003.

[22] Moses Charikar and Shi Li. A dependent LP-rounding approach for the k-median problem. In *Proceedings of the 39th international colloquium conference on Automata, Languages, and Programming - Volume Part I (ICALP'12)*, pages 194–205, 2012.

[23] Moses Charikar, Sudipto Guha, Eva Tardos, David B. Shmoys. A Constant-Factor Approximation Algorithm for the k-Median Problem. *Journal of Computer and System Sciences*, Volume 65, Issue 1, Pages 129-149, 2002.

[24] S. Dasgupta. The hardness of k-means clustering. *Technical Report CS2007-0890, University of California, San Diego*, 2007.

[25] Hu Ding and Jinhui Xu. A unified framework for clustering constrained data without locality property. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 1471–1490, 2015.

[26] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *Proc. KDD*, pages 226–231, 1996.

[27] Tamas Feder and Daniel Greene. Optimal algorithms for approximate clustering. In *Proceedings of the 20th ACM Symposium on Theory of Computing*, pages 434–444, 1988.

[28] Dan Feldman, Morteza Monemizadeh, and Christian Sohler. A PTAS for k-means clustering based on weak coresets. In *Symposium on Computational Geometry*, pages 11–18, 2007.

[29] Zachary Friggstad, Mohsen Rezapour, and Mohammad R. Salavatipour. Local Search Yields a PTAS for k-Means in Doubling Metrics. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS'16)*, 2016.

[30] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[31] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani,and Liadan O? Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Transactions on Knowledge and Data Engineering*, 15(3): 515?528, 2003.

[32] Venkatesan Guruswami and Piotr Indyk. Embeddings and non-approximability of geometric problems. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA'03)*, pages 537–538, 2003.

[33] J. A. Hartigan. Clustering Algorithms. *Wiley (1975).*

[34] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational geometry (SoCG'94)*, pages 332–339, New York, NY, USA, 1994.

[35] A. K. Jain: Data clustering: 50 years beyond k-means. *Pattern Recognition Letters* 31(8), 651 ? 666 (2010).

[36] A. K. Jain, M. N. Murty, P. J. Flynn. Data clustering: A review. *ACM Computing Surveys* 31(3), 264 ? 323 (1999).

[37] Anil K. Jain and Richard C. Dubes. Algorithms for Clustering Data. *Prentice-Hall, Inc., Upper Saddle River, NJ, USA,* 1988.

[38] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC'02)*, pages 731–740, New York, NY, USA, 2002.

[39] Ragesh Jaiswal and Nitin Garg. Analysis of k-means++ for separable data. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, in: Lecture Notes in Computer Science*, vol. 7408, Springer, Berlin Heidelberg, pp. 591?602, 2012

[40] Ragesh Jaiswal, Amit Kumar, and Sandeep Sen. A Simple D^2-Sampling Based PTAS for k-Means and Other Clustering Problems. *Algorithmica* Volume 70, Issue 1, pp 22–46, 2014.

[41] Ragesh Jaiswal, Mehul Kumar, Pulkit Yadav. Improved analysis of D^2-sampling based PTAS for k-means and other clustering problems. *Information Processing Letters*, Volume 115, Issue 2, February 2015, Pages 100-103.

[42] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry*, Volume 28, Issue 2, 2004, Pages 89-112.

[43] S. G. Kolliopoulos and S. Rao. A Nearly Linear-time Approximation Scheme for the Euclidean k-median problem. *SIAM J. Computing, (37)*, pages 757-782, 2007.

[44] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM, 57(2)*, 2010.

[45] Jyh-Han Lin, Jeffrey Scott Vitter. Approximation algorithms for geometric median problems. *Information Processing Letters*, Volume 44, Issue 5, Pages 245-249, 1992.

[46] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing* 17, 4 (December 2007), 395-416, 2007.

[47] Meena Mahajan, Prajakta Nimbhorkar, Kasturi Varadarajan. The planar k-means problem is NP-hard. *Theoretical Computer Science*, Volume 442, Pages 13-21, 2012.

[48] Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM Journal of Computing*, 13:182–196, 1984.

[49] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, Chaitanya Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *J. ACM* 59(6)28:1?28:22, 2013.

[50] Andrea Vattani. The hardness of k-means clustering in the plane. *Technical report, Department of Computer Science and Engineering, University of California San Diego*, 2009.

[51] Dennis Wei. A Constant-Factor Bi-Criteria Approximation Guarantee for k-means++. In *NIPS*, 2016.

[52] P. Worah and S. Sen. A linear time deterministic algorithm to find a small subset that approximates the centroid. *Information Processing Letters* 105(1): 17 – 19, Dec 2007.