Center Based Clustering: A Foundational Perspective

Pranjal Awasthi and Maria-Florina Balcan
Princeton University and Carnegie Mellon University

November 10, 2014

Abstract

In the first part of this chapter we detail center based clustering methods, namely methods based on finding a "best" set of center points and then assigning data points to their nearest center. In particular, we focus on k-means and k-median clustering which are two of the most widely used clustering objectives. We describe popular heuristics for these methods and theoretical guarantees associated with them. We also describe how to design worst case approximately optimal algorithms for these problems. In the second part of the chapter we describe recent work on how to improve on these worst case algorithms even further by using insights from the nature of real world clustering problems and data sets. Finally, we also summarize theoretical work on clustering data generated from mixture models such as a mixture of Gaussians.

1 Approximation algorithms for k-means and k-median

One of the most popular approaches to clustering is to define an objective function over the data points and find a partitioning which achieves the optimal solution, or an approximately optimal solution to the given objective function. Common objective functions include center based objective functions such as k-median and k-means where one selects k center points and the clustering is obtained by assigning each data point to its closest center point. Here closeness is measured in terms of a pairwise distance function $d()$, which the clustering algorithm has access to, encoding how dissimilar two data points are. For instance, the data could be points in Euclidean space with $d()$ measuring Euclidean distance, or it could be strings with $d()$ representing an edit distance, or some other dissimilarity score. For mathematical convenience it is also assumed that the distance function $d()$ is a metric. In k-median clustering the objective is to find center points $c_1, c_2, \cdots c_k$, and a partitioning of the data so as to minimize $\Phi_{k-median} = \sum_x \min_i d(x, c_i)$. This objective is historically very useful and well studied for facility location problems [16, 43]. Similarly the objective in k-means is to minimize $\Phi_{k-means} = \sum_x \min_i d(x, c_i)^2$. Optimizing this objective is closely related to fitting the maximum likelihood mixture model for a given dataset. For a given set of centers, the optimal clustering for that set is obtained by assigning each data point to its closest center point. This is known as the Voronoi partitioning of the data. Unfortunately, exactly optimizing the k-median and the k-means objectives is a notoriously hard problem. Intuitively this is expected since the objective function is a non-convex function of the variables involved. This apparent hardness can also be formally justified by appealing to the

notion of NP completeness [43, 33, 8]. At a high level the notion of NP completeness identifies a wide class of problems which are in principle equivalent to each other. In other words, an efficient algorithm for exactly optimizing one of the problems in the class on all instances would also lead to algorithms for all the problems in the class. This class contains many optimization problems that are believed to be hard[1] to exactly optimize in the worst case and not surprisingly, k-median and k-means also fall into the class. Hence it is unlikely that one would be able to optimize these objectives exactly using efficient algorithms. Naturally, this leads to the question of recovering approximate solutions and a lot of the work in the theoretical community has focused on this direction [16, 11, 29, 34, 43, 47, 48, 57, 20]. Such works typically fall into two categories, a) providing formal worst case guarantees on all instances of the problem, and b) providing better guarantees suited for for nicer, stable instances. In this chapter we discuss several stepping stone results in these directions, focusing our attention on the k-means objective. A lot of the the ideas and techniques mentioned apply in a straightforward manner to the k-median objective as well. We will point out crucial differences between the two objectives as and when they appear. We will additionally discuss several practical implications of these results.

We will begin by describing a very popular heuristic for the k-means problem known as Lloyd's method. Lloyd's method [51] is an iterative procedure which starts out with a set of k seed centers and at each step computes a new set of centers with a lower k-means cost. This is achieved by computing the Voronoi partitioning of the current set of centers and replacing each center with the center of the corresponding partition. We will describe the theoretical properties and limitations of Lloyd's method which will also motivate the need for good worst case approximation algorithms for k-means and k-median. We will see that the method is very sensitive to the choice of the seed centers. Next we will describe a general method based on local search which achieves constant factor approximations for both the k-means and the k-median objectives. Similar to Lloyd's method, the local search heuristic starts out with a set of k seed centers and at each step swaps one of the centers for a new one resulting in a decrease in the k-means cost. Using a clever analysis it can be shown that this procedure outputs a good approximation to the optimal solution [47]. This is interesting, since as mentioned above, optimizing the k-means is NP-complete, in fact it is NP-complete even for $k = 2$, for points in the Euclidean space [33][2].

In the second part of the chapter we will describe some of the recent developments in the study of clustering objectives. These works take a non-worst case analysis approach to the problem. The basic theme is to design algorithms which give good solutions to clustering problems only when the underlying optimal solution has a meaningful structure. We will call such clustering instances as *stable* instances. We would describe in detail two recently studied notions of stability. The first one called *separability* was proposed by Ostrovsky et. al [57]. According to this notion a k-clustering instance is stable if it is much more expensive to cluster the data using $(k-1)$ or fewer clusters. For such instances Ostrovsky et. al show that one can design a simple Lloyd's type algorithm which achieves a constant factor approximation. A different notion called *approximation stability* was proposed by Balcan et. al [20]. The motivation comes from the fact that often in practice optimizing an objective function acts as a proxy for the real problem of getting close to the correct unknown ground truth clustering. Hence it is only natural to assume that any good approximation to the proxy function such as k-means or k-median will also be close to the ground truth clustering in

[1]This is the famous P vs NP problem, and there is a whole area called Computational Complexity Theory that studies this and related problems [12].

[2]If one restricts centers to be data points, then k-means can be solved optimally in time $O(n^{k+1})$ by trying all possible k-tuples of centers and choosing the best. The difficulty of k-means for $k = 2$ in Euclidean space comes from the fact that the optimal centers need not be data points.

terms of structure. Balcan et. al show that under this assumption one can design algorithms that solve the end goal of getting close to the the ground truth clustering. More surprisingly this is true even in cases where it is NP-hard to achieve a good approximation to the proxy objective.

In the last part of the chapter we briefly review existing theoretical work on clustering data generated from mixture models. We mainly focus on Gaussian Mixture Models (GMM) which are the most widely studied distributional model for clustering. We will study algorithms for clustering data from a GMM under the assumption that the mean vectors of the component Gaussians are well separated. We will also see the effectiveness of spectral techniques for GMMs. Finally, we will look at recent work on estimating the parameters of a Gaussian mixture model under minimal assumptions.

2 Lloyd's method for k-means

Consider a set A of n points in the d-dimensional Euclidean space. We start by formally defining Voronoi partitions.

Definition 1 (Voronoi Partition)**.** *Given a clustering instance $\mathcal{C} \subset \mathbb{R}^d$ and k points $c_1, c_2, \cdots c_k$, a Voronoi partitioning using these centers consists of k disjoint clusters. Cluster i consists of all the points $x \in \mathcal{C}$ satisfying $d(x, c_i) \leq d(x, c_j)$ for all $j \neq i$.*[3]

Lloyd's method, also known as the k-means algorithm is the most popular heuristic for k-means clustering in the Euclidean space which has been shown to be one of the top ten algorithms in data mining [69]. The method is an iterative procedure which is described below.

Algorithm Lloyd's method

1. **Seeding:** Choose k seed points $c_1, c_2, \cdots c_k$. Set $\Phi_{old} = \infty$. Compute the current k-means cost Φ using seed points as centers, i.e.
$$\Phi_{curr} = \sum_{i=1}^{n} \min_j d^2(x_i, c_j)$$
2. **While** $\Phi_{curr} < \Phi_{old}$,
   (a) **Voronoi partitioning:** Compute the Voronoi partitioning of the data based on the centers $c_1, c_2, \cdots c_k$. In other words, create k clusters $C_1, C_2, \cdots, C_k$ such that $C_i = \{x : d(x, c_i) \leq \min_{j \neq i} d(x, c_j)\}$. Break ties arbitrarily.
   (b) **Reseeding:** Compute new centers $\hat{c_1}, \hat{c_2}, \cdots, \hat{c_k}$, where $\hat{c_j} = mean(C_j) = \frac{1}{|C_j|} \sum_{x \in C_j} x$. Set $\Phi_{old} = \Phi_{curr}$. Update the current k-means cost Φ_{curr} using the new centers.
   (c) **Output:** The current set of centers $c_1, c_2, \ldots c_k$.

[3] Ties can be broken arbitrarily.

We would like to stress that although Lloyd's method is popularly known as the k-means algorithm, there is a difference between the underlying k-means objective (which is usually hard to optimize) and the k-means algorithm which is a heuristic to solve the problem. An attractive feature of Lloyd's method is that the k-means cost of the clustering obtained never increases. This follows from the fact that for any set of points, the 1-means cost is minimized by choosing the mean of the set as the center. Hence for any cluster C_i in the partitioning, choosing mean(C_i) will never lead to a solution of higher cost. Hence if we repeat this method until there is no change in the k-means cost, we will reach a local optimum of the k-means cost function in finite time. In particular the number of iterations will be at most $n^{O(kd)}$ which is the maximum number of Voronoi partitions of a set of n points in $\Re^d$ [42]. The basic method mentioned above leads to a class of algorithms depending upon the choice of the seeding method. A simple way is to start with k randomly chosen data points. This choice however can lead to arbitrarily bad solution quality as shown in Figure 1. In addition it is also known that the Lloyd's method can take upto 2^n iterations to converge even in 2 dimensions [14, 66].

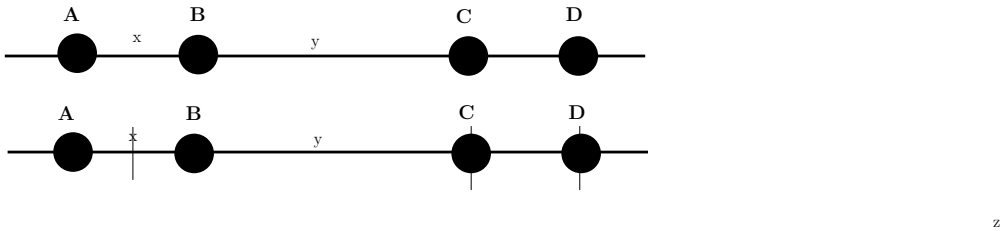

Figure 1: Consider 4 points $\{A, B, C, D\}$ on a line separated by distances x, y and z such that $z < x < y$. Let $k = 3$. The optimal solution has centers at A, B and the centroid of C, D with a total cost of $\frac{z^2}{2}$. When choosing random seeds, there is a constant probability that we choose $\{A, C, D\}$. In this case the final centers will be C, D and the centroid of A, B with a total cost of $\frac{x^2}{2}$. This ratio can be made arbitrarily bad.

In sum, from a theoretical standpoint, k-means with random/arbitrary seeds is not a good clustering algorithm in terms of efficiency or quality. Nevertheless, the speed and simplicity of k-means are quite appealing in practical applications. Therefore, recent work has focused on improving the initialization procedure: deciding on a better way to initialize the clustering dramatically changes the performance of the Lloyd's iteration, both in terms of quality and convergence properties. For example, [15] showed that choosing a good set of seed points is crucial and if done carefully can itself be a good candidate solution without the need for further iterations. Their algorithm called k-means++ uses the following seeding procedure: it selects only the first center uniformly at random from the data and each subsequent center is selected with a probability proportional to its contribution to the overall error given the previous selections. See Algorithm KMEANS++ for a formal description:

Algorithm KMEANS++

1. **Initialize:** a set S by choosing a data point at random.
2. **While** $|S| < k$,
   (a) Choose a data point x with probability proportional to $min_{z \in S} d(x, z)^2$, and add it to S.
3. **Output:** the clustering obtained by the Voronoi partitioning of the data using the centers in S.

[15] showed that Algorithm KMEANS++ is an $\log k$ approximation algorithm for the k-means objective. We say that an algorithm is an α-approximation for a given objective function Φ if for every clustering instance the algorithm outputs a solution of expected cost at most α times the cost of the best solution. The design of approximation algorithms for NP-hard problems has been a fruitful research direction and has led to a wide array of tools and techniques. Formally, [15] show that:

Theorem 1 ([15])**.** *Let S be the set of centers output by the above algorithm and $\Phi(S)$ be the k-means cost of the clustering obtained using S as the centers. Then $E[\Phi(S)] \leq O(\log k)\mathrm{OPT}$, where* OPT *is the cost of the optimal k-means solution.*

We would like to point out that in general the output of k-means++ is not a local optimum. Hence it might be desirable in practice to run a few steps of the Lloyd's method starting from this solution. This could only lead to a better solution.

Subsequent work of [6] introduced a streaming algorithm inspired by the k-means++ algorithm that makes a single pass over the data. They show that if one is allowed to cluster using a little more than k centers, specifically $O(k \log k)$ centers, then one can achieve a constant-factor approximation in expectation to the k-means objective. The approximation guarantee was improved in [5]. Such approximation algorithms which use more than k centers are also known as bi-criteria approximations.

As mentioned earlier, Lloyd's method can take up to exponential iterations in order to converge to a local optimum. However [13] showed that the method converges quickly on an "average" instance. In order to formalize this, they study the problem under the smoothed analysis framework of [65]. In the smoothed analysis framework the input is generated by applying a small Gaussian perturbation to an adversarial input. [65] showed that the simplex method takes polynomial number of iterations on such smoothed instances. In a similar spirit, [13] showed that for smoothed instances Lloyd's method runs in time polynomial in n, the number of points and $\frac{1}{\sigma}$, the standard deviation of the Gaussian perturbation. However, these works do not provide any guarantee on the quality of the final solution produced.

We would like to point out that in principle the Lloyd's method can be extended to the k-median objective. A natural extension would be to replace the mean computation in the Reseeding step with computing the median of a set of points X in the Euclidean space, i.e., a point $c \in \Re^d$ such that $\sum_{x \in X} d(x, c)$ is minimized. However this problem turns out to be NP-complete [53]. For this reason, the Lloyd's method is typically used only for the k-means objective.

3 Properties of the k-means objective

In this section we provide some useful facts about the k-means clustering objective. We will use $\mathcal{C}$ to denote the set of n points which represent a clustering instance. The first fact can be used to show that given a Voronoi partitioning of the data, replacing a given center with the mean of the corresponding partition can never increase the k-means cost.

Fact 2. *Consider a finite set $X \subset \mathbb{R}^d$ and c =mean(X). For any $y \in \mathbb{R}^d$, we have that, $\sum_{x\in X} d(x,y)^2 = \sum_{x\in X} d(x,c)^2 + |X|d(c,y)^2$.*

Proof. Representing each point in the coordinate notation as $x = (x_1, x_2, \cdots, x_d)$, we have that

$$\begin{aligned}
\sum_{x\in X} d(x,y)^2 &= \sum_{x\in X}\sum_{i=1}^{d} |x_i - y_i|^2 \\
&= \sum_{x\in X}\sum_{i=1}^{d} (|x_i - c_i|^2 + |c_i - y_i|^2 + 2(x_i - c_i)(c_i - y_i)) \\
&= \sum_{x\in X} d(x,c)^2 + |X|d(c,y)^2 + \sum_{i=1}^{d} 2(c_i - y_i)\sum_{x\in X}(x_i - c_i) \\
&= \sum_{x\in X} d(x,c)^2 + |X|d(c,y)^2
\end{aligned}$$

Here the last equality follows from the fact that for any i, $c_i = \sum_{x\in X} x_i/n$. □

An easy corollary of the above fact is the following,

Corollary 3. *Consider a finite set $X \subset \mathbb{R}^d$ and let c = mean(X). We have $\sum_{x,y\in X} d(x,y)^2 = 2|X|\sum_{x\in X} d(x,c)^2$.*

Below we prove another fact which will be useful later.

Fact 4. *Let $X \subset \mathbb{R}^d$ be finite set of points. Let ${\Delta_1}^2(X)$ denote the 1-means cost of X. Given a partition of X into X_1 and X_2 such that c =mean(X), c_1 =mean(X_1) and c_2 = mean(X_2), we have that a) ${\Delta_1}^2(X) = {\Delta_1}^2(X_1) + {\Delta_1}^2(X_2) + \frac{|X_1||X_2|}{|X|} d(c_1,c_2)^2$. and b) $d(c,c_1)^2 \leq \frac{{\Delta_1}^2(X)|X_2|}{|X||X_1|}$.*

Proof. We can write ${\Delta_1}^2(X) = \sum_{x\in X_1} d(x,c)^2 + \sum_{x\in X_2} d(x,c)^2$. Using Fact 2 we can write

$$\sum_{x\in X_1} d(x,c)^2 = {\Delta_1}^2(X_1) + |X_1|d(c,c_1)^2.$$

Similarly, $\sum_{x\in X_2} d(x,c)^2 = {\Delta_1}^2(X_2) + |X_2|d(c,c_2)^2$. Hence we have

$$\Delta_1{}^2(X) = \Delta_1{}^2(X_1) + \Delta_1{}^2(X_2) + |X_1|d(c,c_1)^2 + |X_2|d(c,c_2)^2.$$

Part (a) follows by substituting $c = \frac{|X_1|c_1+|X_2|c_2}{|X_1|+|X_2|}$ in the above equation.

From Part (a) we have that

$$\Delta_1{}^2(X) \geq \frac{|X_1||X_2|}{|X|} d(c_1, c_2)^2.$$

Part (b) follows by substituting $c_2 = \frac{(|X_1|+|X_2|)}{X_2} c - \frac{|X_1|}{|X_2|} c_1$ above. □

4 Local search based algorithms

In the previous section we saw that a carefully chosen seeding can lead to a good approximation for the k-means objective. In this section we will see how to design much better (constant factor) approximation algorithms for k-means (as well as k-median). We will describe a very generic approach based on local search. These algorithms work by making local changes to a candidate solution and improving it at each step. They have been successfully used for a variety of optimization problems [7, 28, 36, 40, 58, 61]. Kanungo et. al [47] analyzed a simple local search based algorithm for k-means as described below.

Algorithm LOCAL SEARCH

1. **Initialization:** Choose k data points $\{c_1, c_2, \ldots c_k\}$ arbitrarily from the data set $\mathcal{D}$. Let this set be T. Let $\Phi(T)$ denote the cost of the k-means solution using T as centers, i.e., $\Phi(T) = \sum_{i=1}^{n} \min_j d^2(x_i, c_j)$. Set $T_{old} = \phi$, $T_{curr} = T$.
2. **While** $\Phi(T_{curr}) < \Phi(T_{old})$,
   - For $x \in T_{curr}$ and $y \in \mathcal{D} \setminus T_{curr}$:
     if $\Phi((T_{curr} \setminus \{x\}) \cup \{y\}) < \Phi(T_{curr})$, update $T_{old} = T_{curr}$ and $T_{curr} \leftarrow (T_{curr} \setminus \{x\}) \cup \{y\}$.
3. **Output:** $S = T_{curr}$ as the set of final centers.

We would like to point out that in order to make the above algorithm run in polynomial time, one needs to change the criteria in the while loop to be $\Phi(T_{curr}) < (1-\epsilon)\Phi(T_{old})$. The running time will then depend polynomially in n and $1/\epsilon$. For simplicity of analysis, we will prove the following theorem for the idealized version of the algorithm with no ϵ.

Theorem 5 ([47]). *Let S be the final set of centers returned by the above procedure. Then, $\Phi(S) \leq 50\mathrm{OPT}$.*

In order to prove the above theorem we start by building up some notation. Let T be the set of k data points returned by the local search algorithm as candidate centers. Let O be the set of k data points which achieve the minimum value of the k-means cost function among all sets of k data points. Note that the centers in O do not necessarily represent the optimal solution as the optimal centers might not be data points. However using the next lemma one can show that using data points as centers is only twice as bad as the optimal solution.

Lemma 6. *Given $\mathcal{C} \subseteq \mathbb{R}^d$, and the optimal k-means clustering of $\mathcal{C}$, $\{C_1, C_2, \cdots C_k\}$, there exists a set S of k data points such that $\Phi(S) \leq 2\text{OPT}$.*

Proof. For a given set $\mathcal{C} \subseteq \Re^d$, let ${\Delta_1}^2$ represent the 1-means cost of $\mathcal{C}$. From Fact 2 it is easy to see that this cost is achieved by choosing the mean of $\mathcal{C}$ as the center. In order to prove the above lemma it is enough to show that for each optimal cluster C_i with mean c_i, there exists a data point $x_i \in C_i$ such that $\sum_{x \in C_i} d(x, x_i)^2 \leq 2{\Delta_1}^2(C_i)$. Let x_i be the data point in C_i which is closest to c_i. Again using Fact 2 we have $\sum_{x \in C_i} d(x, x_i)^2 = {\Delta_1}^2(C_i) + |C_i| d(x, c_i)^2 \leq 2{\Delta_1}^2(C_i)$. □

Hence it is enough to compare the cost of the centers returned by the algorithm to the cost of the optimal centers using data points. In particular, we will show that $\Phi(T) \leq 25\Phi(O)$. We start with the simple observation that by the property of the local search algorithm, for any $t \in T$, and $o \in O$, swapping t for o results in an increase in cost. In other words

$$\Phi(T - t + o) - \Phi(T) \geq 0 \tag{4.1}$$

The main idea is to add up Equation 4.1 over a carefully chosen set of swaps $\{o, t\}$ to get the desired result. In order to describe the set of swaps chosen we start by defining a cover graph

Definition 2. *A cover graph is a bipartite graph with the centers in T on one side and the centers in O on the other side. For each $o \in O$, let t_o be the point in T which is closest to o. The cover graph contains edges of the form o, t_o for all $o \in O$.*

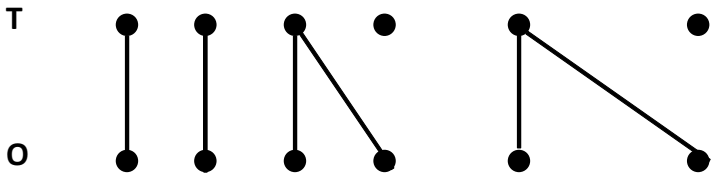

Figure 2: An example cover graph.

Next we use the cover graph to generate the set of useful swaps. For each $t \in T$ which has degree 1 in the cover graph, we output the swap pair $\{t, o\}$ where o is the point connected to t. Let T' be the degree 0 vertices in the cover graph. We pair the remaining vertices $o \in O$ with the vertices in T' such that each vertex in O has degree 1 and each vertex in T' has degree at most 2. To see that such a pairing will exist notice that for any $t \in T$ of degree $k > 1$ there will exist $k - 1$ distinct zero vertices in T; these vertices can be paired to vertices in O connected to t maintaining the above property. We then output all the edges in this pairing as the set of useful swaps.

4.1 Bounding the cost of a swap

Consider a swap $\{o, t\}$ output by using the cover graph. We will apply Equation 4.1 to this pair. We will explicitly define a clustering using centers in $T - t + o$ and upper bound its cost.

We will then use the lower bound of $\Phi(T)$ from Equation 4.1 to get the kind of equations we want to sum up over. Let the clustering given by centers in T be $C_1, C_2, \cdots C_k$. Let ${C_o}^*$ be the cluster corresponding to center o in the optimal clustering given by O. Let o_x be the closest point in O to x. Similarly let t_x be the closest point in T to x. The key property satisfied by any output pair $\{o, t\}$ is the following

Fact 7. *Let $\{o, t\}$ be a swap pair output using the cover graph. Then we have that for any $x \in C_t$ either $o_x = o$ or $t_{o_x} \neq t$.*

Proof. Assume that for some $x \in C_t$, $o_x = o' \neq o$. By the procedure used to output swap pairs we have that t has degree 1 or 0 in the cover graph. In addition, if t has degree 1 then $t_o = t$. In both the cases we have that $t_{o'} \neq t$. □

Next we create a new clustering by swapping o for t and assigning all the points in ${C_o}^*$ to o. Next we reassign points in $C_t \setminus {C_o}^*$. Consider a point $x \in C_t \setminus {C_o}^*$. Clearly $o_x \neq o$. Let t_{o_x} be the point in T which is connected to o_x in the cover graph. We assign x to t_{o_x}. One needs to ensure here that $t_{o_x} \neq t$ which follows from Fact 7. From Equation 4.1 the increase in cost due to this reassignment must be non-negative. In other words we have

$$\sum_{x \in {C_o}^*} (d(x,o)^2 - d(x,t_x)^2) + \sum_{x \in C_t \setminus {C_o}^*} (d(x,t_{o_x})^2 - d(x,t)^2) \geq 0 \tag{4.2}$$

We will add up Equation 4.2 over the set of all good swaps.

4.2 Adding it all up

In order to sum up over all swaps notice that in the first term in Equation 4.2 every point $x \in \mathcal{C}$ appears exactly once by being in ${C_o}^*$ for some $o \in O$. Hence the sum over all swaps of the first term can be written as $\sum_{x \in \mathcal{C}} (d(x,o_x)^2 - d(x,t_x)^2)$. Consider the second term in Equation 4.2. We have that $(d(x,t_{o_x})^2 - d(x,t)^2)) \geq 0$ since x is in C_t. Hence we can replace the second summation over all $x \in C_t$ without affecting the inequality. Also every point $x \in \mathcal{C}$ appears at most twice in the second term by being in C_t for some $t \in T$. Hence the sum over all swaps of the second term is at most $\sum_{x \in \mathcal{C}} (d(x,t_{o_x})^2 - d(x,t_x)^2)$. Adding these up and rearranging we get that

$$\Phi(O) - 3\Phi(T) + 2R \geq 0 \tag{4.3}$$

Here $R = \sum_{x \in \mathcal{C}} d(x,t_{o_x})^2$.

In the last part we will upper bound the quantity R. R represents the cost of assigning every point x to a center in T but not necessarily the closest one. Hence, $R \geq \Phi(T) \geq \Phi(O)$. However we next show that this reassignment cost is not too large.

Notice that R can also be written as $\sum_{o \in O} \sum_{x \in {C_o}^*} d(x,t_o)^2$. Also $\sum_{x \in {C_o}^*} d(x,t_o)^2 = \sum_{x \in {C_o}^*} d(x,o)^2 + |{C_o}^*| d(o,t_o)^2$. Hence we have that $R = \sum_{o \in O} \sum_{x \in {C_o}^*} (d(x,o)^2 + d(o,t_o)^2)$.

Also note that $d(o, t_o) \leq d(o, t_x)$ for any x. Hence

$$\begin{aligned} R &\leq \sum_{o \in O} \sum_{x \in C_o{}^*} (d(x,o)^2 + d(o,t_x)^2) \\ &= \sum_{x \in \mathcal{C}} (d(x,o_x)^2 + d(o_x,t_x)^2) \end{aligned}$$

Using triangle inequality we know that $d(o_x, t_x) \leq d(o_x, x) + d(x, t_x)$. Substituting above and expanding we get that

$$R \leq 2\Phi(O) + \Phi(T) + 2\sum_{x \in \mathcal{C}} d(x,o_x)d(x,t_x) \tag{4.4}$$

The last term in the above equation can be bounded using Cauchy-Schwarz inequality as $\sum_{x \in \mathcal{C}} d(x,o_x)d(x,t_x) \leq \sqrt{\Phi(O)}\sqrt{\Phi(S)}$. So we have that $R \leq 2\Phi(O) + \Phi(T) + 2\sqrt{\Phi(O)}\sqrt{\Phi(S)}$. Substituting this in Equation 4.3 and solving we get the desired result that $\Phi(T) \leq 25\Phi(O)$. Combining this with Lemma 6 proves Theorem 5.

A natural generalization of Algorithm LOCAL SEARCH is to swap more than one centers at each step. This could potentially lead to a much better local optimum. This multi-swap scheme was analyzed by [47] and using a similar analysis as above one can show the following

Theorem 8. *Let S be the final set of centers by the local search algorithm which swaps upto p centers at a time. Then we have that $\Phi(S) \leq 2(3 + \frac{2}{p})^2\mathrm{OPT}$, where* OPT *is the cost of the optimal k-means solution.*

For the case of k-median the same algorithm and analysis gives [16]

Theorem 9. *Let S be the final set of centers by the local search algorithm which swaps upto p centers at a time. Then we have that $\Phi(S) \leq (3 + \frac{2}{p})\mathrm{OPT}$, where* OPT *is the cost of the optimal k-median solution.*

This approximation factor for k-median has recently been improved to $(1 + \sqrt{3} + \epsilon)$ [50]. For the case of k-means in Euclidean space [48] give an algorithm which achieves a $(1 + \epsilon)$ approximation to the k-means objective for any constant $\epsilon > 0$. However the runtime of the algorithm depends exponentially in k and hence it is only suitable for small instances.

5 Clustering of *stable* instances

In this part of the chapter we delve into some of the more modern research in the theory of clustering. In recent past there has been an increasing interest in designing clustering algorithms that enjoy strong theoretical guarantees on non-worst case instance. This is of significant interest for two reasons: a) From a theoretical point of view, this helps us understand and characterize the class of problems for which one can get optimal or close

to optimal guarantees, b) From a practical point of view, real world instances often have additional structure that could be exploited to get better performance. Compared to worst case analysis, the main challenge here is to formalize well motivated and interesting additional structures of clustering instances under which good algorithms exist. In this section we present two popular interesting notions.

5.1 ϵ-separability

This notion of stability was proposed by Ostrovsky et al.[57]. Given an instance of k-means clustering, let $\mathrm{OPT}(k)$ denote the cost of the optimal k-means solution. We can also decompose $\mathrm{OPT}(k)$ as $\mathrm{OPT} = \sum_{i=1}^{k} \mathrm{OPT}_i$, where OPT_i denotes the 1-means cost of cluster C_i, i.e., $\sum_{x \in C_i} d(x, c_i)^2$. Such an instance is called *ϵ-separable* if it satisfies $\mathrm{OPT}(k-1) > \frac{1}{\epsilon^2}\mathrm{OPT}(k)$.

The definition is motivated by the following issue: when approaching a clustering problem, one typically has to decide how many clusters one wants to partition the data in, i.e., the value of k. If the k-means objective is the underlying criteria being used to judge the quality of a clustering, and the optimal $(k-1)$-means clustering is comparable to the optimal k-means clustering, then one can in principle also use $(k-1)$ clusters to describe the data set. In fact this particular method is a very popular heuristic to find out the number of hidden clusters in the data set. In other words choose the value of k at which there is a significant increase in the k-means cost when going from k to $k-1$. As an illustrative example consider the case of a mixture of k spherical unit variance Gaussians in d dimensions whose pair wise means are separated by a distance $D >> 1$. Given n points from each Gaussian, the optimal k-means cost with high probability is nkd. On the other hand, if we try to cluster this data using $(k-1)$ clusters, the optimal cost will now become $n(k-1)d + n(D^2 + d)$. Hence, taking the ratio of the two costs, this instance will be ϵ-separable for $\frac{1}{\epsilon^2} = \frac{(k-1)d + D^2 + d}{kd} = 1 + \frac{D^2}{kd}$), so $\epsilon = (1 + \frac{D^2}{kd})^{-1/2}$. Hence, if $D \gg \sqrt{kd}$, then the instance will be highly separable (the separability parameter ϵ will be $o(1)$).

It was shown by Ostrovsky et al. [57] that one can design much better approximation algorithms for ϵ-separable instances.

Theorem 10 ([57]). *There is a polynomial time algorithm which given any ϵ-separable 2-means instance returns a clustering of cost at most $\frac{OPT}{1-\rho}$ with probability at least $1 - O(\rho)$ where $c_2\epsilon^2 \leq \rho \leq c_1\epsilon^2$ for some constants $c_1, c_2 > 0$.*

Theorem 11 ([57]). *There is a polynomial time algorithm which given any ϵ-separable k-means instance a clustering of cost at most $\frac{\mathrm{OPT}}{1-\rho}$ with probability $1 - O((\rho)^{1/4})$ where $c_2\epsilon^2 \leq \rho \leq c_1\epsilon^2$ for some constants $c_1, c_2 > 0$.*

5.2 Proof Sketch and Intuition for Theorem 10

Notice that the above algorithm does not need to know the value of ϵ from the separability of the instance. Define r_i to be the radius of cluster C_i in the optimal k-means clustering,

i.e., ${r_i}^2 = \frac{\text{OPT}_i}{|C_i|}$. The main observation is that under the ϵ-separability condition, the optimal k-means clustering is "spread out". In other words, the radius of any cluster is much smaller than the inter cluster distances. This can be formulated in the following lemma

Lemma 12. $\forall i, j,\ d(c_i, c_j)^2 \geq \frac{1-\epsilon^2}{\epsilon^2} max({r_i}^2, {r_j}^2)$.

Proof. Given an ϵ-separable instance of k-means, consider any two clusters C_i and C_j in the optimal clustering with centers c_i and c_j respectively. Consider the $(k-1)$ clustering obtained by deleting c_j and assigning all the points in C_j to C_i. By ϵ-separability, the cost of this new clustering must be at least $\frac{\text{OPT}}{\epsilon^2}$. However the increase in the cost will be exactly $|C_j| d(c_i, c_j)^2$. This follows from the simple observation stated in Fact 2. Hence we have that $|C_j| d(c_i, c_j)^2 > (\frac{1}{\epsilon^2} - 1)\text{OPT}$. This gives us that ${r_j}^2 = \frac{\text{OPT}}{|C_j|} \leq \frac{\epsilon^2}{1-\epsilon^2} d(c_i, c_j)^2$. Similarly, if we delete c_i and assign all the points in C_i to C_j we get that ${r_i}^2 \leq \frac{\epsilon^2}{1-\epsilon^2} d(c_i, c_j)^2$. □

When dealing with the two means problem, if one could find two initial candidate center points which are close to the corresponding optimal centers, then we could hope to run a Lloyd's type step and improve the solution quality. In particular if we could find $\bar{c_1}$ and $\bar{c_2}$ such that $d(c_1, \bar{c_1})^2 \leq \alpha {r_1}^2$ and $d(c_2, \bar{c_2})^2 \leq \alpha {r_2}^2$, then we know from Fact 2 that using these center points will give us a $(1+\alpha)$ approximation to OPT. Lemma 12 suggests the following approach: pick data points x, y with probability proportional to $d(x, y)^2$. We will show that this will lead to seed points $\hat{c_1}$ and $\hat{c_2}$ not too far from the optimal centers. Applying a Lloyd type reseeding step will then lead us to the final centers which will be much closer to the optimal centers. We start by defining the core of a cluster.

Definition 3 (Core of a cluster)**.** *Let $\rho < 1$ be a constant. We define $X_i = \{x \in C_i : d(x, c_i)^2 \leq \frac{{r_i}^2}{\rho}\}$. We call X_i as the core of the cluster C_i.*

We next show that if we pick initial seeds $\{\hat{c_1}, \hat{c_2}\} = \{x, y\}$ with probability proportional to $d(x, y)^2$ then with high probability the points lie within the core of different clusters.

Lemma 13. *For sufficiently small ϵ and $\rho = \frac{100\epsilon^2}{1-\epsilon^2}$, we have $Pr[\{\hat{c_1}, \hat{c_2}\} \cap X_1 \neq \emptyset$ and $\{x, y\} \cap X_2 \neq \emptyset] = 1 - O(\rho)$.*

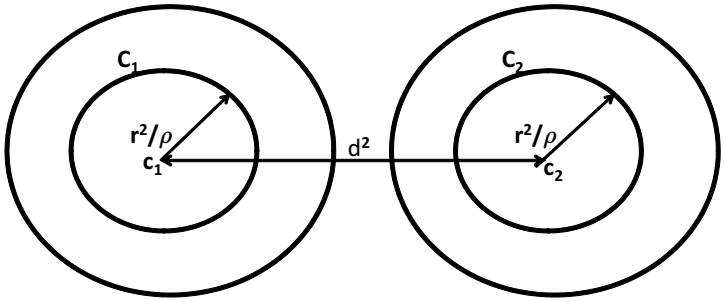

Figure 3: An ϵ-separable 2-means instance

Proof Sketch. For simplicity assume that the sizes of the two clusters is the same, i.e., $|C_i| = |C_j| = n/2$. In this case, we have ${r_1}^2 = {r_2}^2 = \frac{2OPT}{n} = r^2$. Also, let $d^2(c_1, c_2) = d^2$.

From ε-separability, we know that $d^2 > \frac{1-\epsilon^2}{\epsilon^2} r^2$. Also, from the definition of the core, we know that at least $(1-\rho)$ fraction of the mass of each cluster lies within the core. Hence, the clustering instance looks like the one showed in Figure 3. Let $A = \sum_{x \in X_1, y \in X_2} d(x,y)^2$ and $B = \sum_{x,y \subset \mathcal{C}} d(x,y)^2$. Then the probability of the event is exactly $\frac{A}{B}$. Let's analyze quantity B first. The proof goes by arguing that the pairwise distances between X_1 and X_2 will dominate B. This is because of Lemma 12 which says that d^2 is much greater than r^2, the average radius of a cluster. More formally, From Corollary 3 and from Fact 4 we can get that $B = n{\Delta_1}^2(\mathcal{C}) = n{\Delta_2}^2(\mathcal{C}) + n^2/4d^2$. In addition ϵ-separability tells us that ${\Delta_1}^2(\mathcal{C}) > 1/\epsilon^2 {\Delta_2}^2(\mathcal{C})$. Hence we get that $B \leq \frac{n^2}{4(1-\epsilon^2)} d^2$.

Let's analyze $A = \sum_{x \in X_1, y \in X_2} d(x,y)^2$. From triangle inequality, we have that for any $x \in X_1$, $y \in X_2$, $d^2(x,y) \geq (d - 2r/\sqrt{\rho})^2$. Hence $A \geq \frac{1}{4}(1-\rho)^2 n^2 (d - 2r/\sqrt{\rho})^2$. Substituting these bounds and using the fact that $\rho O(\epsilon^2)$, gives us that $A/B \geq (1 - O(\rho))$.

□

Using these initial seeds we now show that a single step of a Lloyd's type method can yield good a solution. Define $r = d(\hat{c_1}, \hat{c_2})/3$. Define $\bar{c_1}$ as the mean of the points in $B(\hat{c_1}, r)$ and $\bar{c_2}$ as the mean of the points in $B(\hat{c_2}, r)$. Notice that instead of taking the mean of the Voronoi partition corresponding to $\hat{c_1}$ and $\hat{c_2}$, we take the mean of the points within a small radius of the given seeds.

Lemma 14. *Given $\hat{c_1} \in X_1$ and $\hat{c_2} \in X_2$, the clustering obtained using $\bar{c_1}$ and $\bar{c_2}$ as centers has 2-means cost at most $\frac{OPT}{1-\rho}$.*

Proof. We will first show that $X_1 \subseteq B(\hat{c_1}, r) \subseteq C_1$. Using Lemma 12 we know that $d(\hat{c_1}, c_1) \leq \frac{\epsilon}{\rho(1-\epsilon^2)} d(c_1, c_2) \leq d(c_1, c_2)/10$ for sufficiently small ϵ. Similarly $d(\hat{c_2}, c_2) \leq d(c_1, c_2)/10$. Hence we get that $4/5 \leq r \leq 6/5$. So for any $z \in B(\hat{c_1}, r)$, $d(z, c_1) \leq d(c_1, c_2)/2$. Hence $z \in C_1$. Also for any $z \in X_1$, $d(z, \hat{c_1}) \leq 2\frac{{r_1}^2}{\rho} \leq r$. Similarly one can show that $X_2 \subseteq B(\hat{c_2}, r) \subseteq C_2$. Now applying Fact 4 we can claim that $d(\bar{c_1}, c_1) \leq \frac{\rho}{1-\rho} {r_1}^2$ and $d(\bar{c_2}, c_2) \leq \frac{\rho}{1-\rho} {r_2}^2$. So using $\bar{c_1}$ and $\bar{c_2}$ as centers we get a clustering of cost at most $\text{OPT} + \frac{\rho}{1-\rho}\text{OPT} = \frac{\text{OPT}}{1-\rho}$. □

Summarizing the discussion above, we have the following simple algorithm for the 2-means problem.

Algorithm 2-MEANS

1. **Seeding:** Choose initial seeds x, y with probability proportional to $d(x,y)^2$.
2. Given seeds $\hat{c_1}, \hat{c_2}$, let $r = d(\hat{c_1}, \hat{c_2})/3$. Define $\bar{c_1} = mean(B(\hat{c_1}, r))$ and $\bar{c_2} = mean(B(\hat{c_2}, r))$.
3. **Output:** $\bar{c_1}$ and $\bar{c_2}$ as the cluster centers.

5.3 Proof Sketch and Intuition for Theorem 11

In order to generalize the above argument to the case of k clusters, one could follow a similar approach and start with k initial seed centers. Again we start by choosing x, y with probability proportional to $d(x,y)^2$. After choosing a set of U of points, we choose the next point z with probability proportional to $min_{\hat{c}_i \in U} d(z, \hat{c}_i)^2$. Using a similar analysis as in Lemma 13 one can show that if we pick k seeds then with probability $(1 - O(\rho))^k$ they will lie with the cores of different clusters. However this probability of success is exponentially small in k and is not good for our purpose. The approach taken in [57] is to sample a larger set of points and argue that with high probability it is going to contain k seed points from the "outer" cores of different clusters. Here we define outer core of a cluster as $X_i^{out} = \{x \in C_i : d(x, c_i)^2 \leq \frac{{r_i}^2}{\rho^3}\}$ – so this notion is similar to the core notion for $k = 2$ except that the radius of the core is bigger by a factor of $1/(\rho)$ than before. We would like to again point out a similar seeding procedure as the one described above is used in the k-means++ algorithm [15](See Section 2). One can show that using k seed centers in this way gives an $O(\log(k))$-approximation to the k-means objective in the worst case.

Lemma 15 ([57]). *Let $N = \frac{2k}{1-5\rho} + \frac{2\ln(2/\delta)}{(1-5\rho)^2}$, where $\rho = \sqrt{\epsilon}$. If we sample N points using the sampling procedure then $Pr[\forall j = 1 \cdots k$, there exists some $\hat{x}_i \in {X_j}^{out}] \geq 1 - \delta$*

Since we sample more than k points in the first step, one needs to extract k good seed points out of this set before running the Lloyd step. This is achieved by the following greedy procedure:

Algorithm GREEDY DELETION PROCEDURE

1. Let S denote the current set of candidate centers. Let $\Phi(S)$ denote the k-means cost of the Voronoi partition using S. Similarly, for $x \in S$ denote $\Phi(S_x)$ be the k-means cost of the Voronoi partition using $S \setminus \{x\}$.
2. **While** $|S| > k$,
   - remove a point x from S, such that $\Phi(S_x) - \Phi(S)$ is minimum.
   - For every remaining point $x \in S$, let $R(x)$ denote the Voronoi set corresponding to x. Replace x by $mean(R(x))$.
   - **Output:** S.

At the end of the greedy procedure we have the following guarantee

Lemma 16. *For every optimal center c_i, there is a point $\hat{c}_i \in S$, such that $d(c_i, \hat{c}_i) \leq \frac{D_i}{10}$. Here $D_i = min_{j \neq i}\, d(c_i, c_j)$.*

Using the above lemma and applying the same Lloyd step as in the 2-means problem we get a set of k good final centers. These centers have the property that for each i, $d(c_i, \bar{c}_i) \leq \frac{\rho}{1-\rho} {r_i}^2$. Putting the above argument formally we get the desired result.

5.4 Approximation stability

In [20] Balcan et al. introduce and analyze a class of approximation stable instances for which they provide polynomial time algorithms for finding accurate clustering. The starting point of this work, is that for many problems of interest to machine learning, such as as clustering proteins by function, images by subject, or documents by topic, there is some unknown correct target clustering. In such cases the implicit hope when pursuing an objective based clustering approach (k-means or k-median) is that approximately optimizing the objective function will in fact produce a clustering of low clustering error, i.e. a clustering that is point wise close to the target clustering. Balcan et al. have shown that by making this implicit assumption explicit, one can efficiently compute a low-error clustering even in cases when the approximation problem of the objective function is NP-complete! This is quite interesting since it shows that by exploiting the properties of the problem at hand one can solve the desired problem and bypass worst case hardness results. A similar stability assumption, regarding additive approximations, was presented in [54]. The work of [54] studied sufficient conditions under which the stability assumption holds true.

Formally, the *approximation stability* notion is defined as follows:

Definition 4 (($(1+\alpha,\epsilon)$-approximation-stability)). *Let X be a set of n points residing in a metric space $\mathcal{M}$. Given an objective function Φ (such as k-median, k-means, or min-sum), we say that instance $(\mathcal{M}, X)$* satisfies $(1+\alpha,\epsilon)$-approximation-stability for Φ *if all clusterings $\mathcal{C}$ with $\Phi(\mathcal{C}) \leq (1+\alpha)\cdot \mathrm{OPT}_\Phi$ are ϵ-close to the target clustering $\mathcal{C}_T$ for $(\mathcal{M}, S)$.*

Here the term "target" clustering refers to the ground truth clustering of X which one is trying to approximate. It is also important to clarify what we mean by an ϵ-close clustering. Given two k clusterings $\mathcal{C}$ and $\mathcal{C}^*$ of n points, the distance between them is measured as $dist(\mathcal{C},\mathcal{C}^*) = \min_{\sigma\in S_k} \frac{1}{n}\sum_{i=1}^{k} |C_i \setminus {C^*}_{\sigma(i)}|$. We say that $\mathcal{C}$ is ϵ-close to $\mathcal{C}^*$ if the distance between them is at most ϵ. Interestingly, this approximation stability condition implies a lot of structure about the problem instance which could be exploited algorithmically. For example, we can show the following.

Theorem 17. *[[20]] If the given instance $(\mathcal{M}, S)$ satisfies $(1+\alpha,\epsilon)$-approximation-stability for the k-median or the k-means objective, then we can efficiently produce a clustering that is $O(\epsilon+\epsilon/\alpha)$-close to the target clustering $\mathcal{C}_T$.*

Notice that the above theorem is valid even for values of α for which getting a $(1+\alpha)$-approximation to k-median and k-means is NP-hard! In a recent paper, [4] show that running the kmeans++ algorithm for approximation stable instances of k-means gives a constant factor approximation with probability $\Omega(\frac{1}{k})$. In the following we will provide a sketch of the proof of Theorem 17 for k-means clustering.

5.5 Proof Sketch and Intuition for Theorem 17

Let $C_1, C_2, \cdots, C_k$ be an optimal k-means clustering of $\mathcal{C}$. Let $c_1, c_2, \cdots, c_k$ be the corresponding cluster centers. For any point $x \in \mathcal{C}$, let $w(x)$ be the distance of x to its cluster center. Similarly let $w_2(x)$ be the distance of x to the second closest center. The value of the optimal solution can then we written as $\mathrm{OPT} = \sum_x w(x)^2$. The main implication of

approximation stability is that most of the points are much closer to their own center than to the centers of other clusters. Specifically:

Lemma 18. *If the instance $(\mathcal{M}, X)$ satisfies $(1+\alpha, \epsilon)$-approximation-stability then less than $6\epsilon n$ points satisfy $w_2(x)^2 - w(x)^2 \leq \frac{\alpha \text{OPT}}{2\epsilon n}$.*

Proof. Let $\mathcal{C}^*$ be the optimal k-means clustering. First notice that by approximation-stability $dist(\mathcal{C}^*, \mathcal{C}_T) = \epsilon^* \leq \epsilon$. Let B be the set of points that satisfy $w_2(x)^2 - w(x)^2 \leq \frac{\alpha \text{OPT}}{2\epsilon n}$. Let us assume that $|B| > 6\epsilon n$. We will create a new clustering $\mathcal{C}'$ by transferring some of the points in B to their second closest center. In particular it can be shown that there exists a subset of size $|B|/3$ such that for each point reassigned in this set, the distance of the clustering to $\mathcal{C}^*$ increases by $1/n$. Hence we will have a clustering $\mathcal{C}'$ which is 2ϵ away from $\mathcal{C}^*$ and at least ϵ away from $\mathcal{C}_T$. However the increase in cost in going from $\mathcal{C}^*$ to $\mathcal{C}'$ is at most αOPT. This contradicts the approximation stability assumption. □

Let us define $d_{crit} = \sqrt{\frac{\alpha \text{OPT}}{50\epsilon n}}$ as the critical distance. We call a point x *good* if it satisfies $w(x)^2 < {d_{crit}}^2$ and $w_2(x)^2 - w(x)^2 > 25{d_{crit}}^2$. Otherwise we call x as a *bad* point. Let B be the set of all bad points and let G_i be the good points in target cluster i. By Lemma 18 at most $6\epsilon n$ points satisfy $w_2(x)^2 - w(x)^2 > 25d_{crit}^2$. Also from Markov's inequality at most $\frac{50\epsilon n}{\alpha}$ points can have $w(x)^2 > {d_{crit}}^2$. Hence $|B| = O(\epsilon/\alpha)$.

Given Lemma 18, if we then define the *τ-threshold graph* $G_\tau = (S, E_\tau)$ to be the graph produced by connecting all pairs $\{x, y\} \in \binom{\mathcal{C}}{2}$ with $d(x, y) < \tau$, and consider $\tau = 2d_{crit}$ we get the following two properties:

(1) For $x, y \in {C_i}^*$ such that x and y are good points, we have $\{x, y\} \in E(G_\tau)$.

(2) For $x \in {C_i}^*$ and $y \in {C_j}^*$ such that x and y are good points, $\{x, y\} \notin E(G_\tau)$.

(3) For $x \in {C_i}^*$ and $y \in {C_j}^*$, x and y do not have any good point as a common neighbor.

Hence the threshold graph has the structure as shown in Figure 4, where each G_i is a clique representing the set of good points in cluster i. This suggests the following algorithm for k-means clustering. Notice that unlike the algorithm for ϵ-separability, the algorithm for approximation stability mentioned below needs to know the values of the stability parameters α and ϵ[4].

[4]This is specifically for the goal of finding a clustering that nearly matches an unknown target clustering, because one may not in general have a way to identify which of two proposed solutions is preferable. On the other hand, if the goal is to find a solution of low cost, then one does not need to know α or ϵ: one can just try all possible values for d_{crit} in the algorithm and take the solution of least total cost.

Algorithm k-MEANS ALGORITHM

Input: $\epsilon \leq 1$, $\alpha > 0$, k.

1. **Initialization:** Define $d_{crit} = \sqrt{\frac{\alpha \mathrm{OPT}}{50\epsilon n}}$ [a]
2. Construct the τ-threshold graph G_τ with $\tau = 2d_{crit}$.
3. **For** $j = 1$ to k do:
   Pick the vertex v_j of highest degree in G_τ.
   Remove v_j and its neighborhood from G_τ and call this cluster $C(v_j)$.
4. **Output:** the k clusters $C(v_1), \ldots, C(v_{k-1}), S - \cup_{i=1}^{k-1} C(v_i)$.

[a] For simplicity we assume here that one knows the value of OPT. If not, one can run a constant-factor approximation algorithm to produce a sufficiently good estimate.

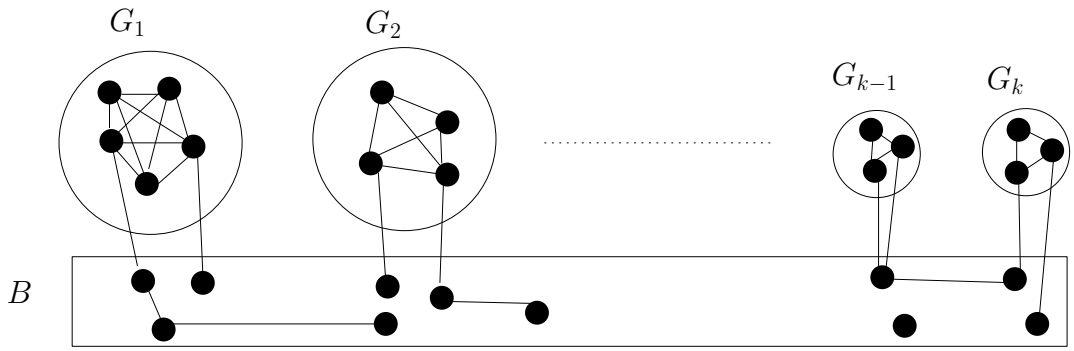

Figure 4: The structure of the threshold graph.

The authors in [20] use the properties of the threshold graph to show that the greedy method of Step 3 of the algorithm produces an accurate clustering. In particular, if the vertex v_j we pick is a good point in some cluster C_i, then we are guaranteed to extract the whole set G_i of good points in that cluster and potentially some bad points as well (see Figure 5(a)). If on the other hand the vertex v_j we pick is a bad point, then we might extract only a part of a good set G_i and miss some good points in G_i, which might lead to some errors. (Note that by property (3) we never extract parts of two different good sets G_i and G_j). However, since v_j was picked to be the vertex of the highest degree in G_τ, we are guaranteed to extract at least as many bad points as the number of missed good points in G_i see Figure 5(b). These than implies that overall we can charge the errors to the bad points, so the distance between the target clustering and the resulting clustering is $O(\epsilon/\alpha)n$, as desired.

5.6 Other notions of stability and relations between them

This notion of ϵ-separability, is in fact related to (c, ϵ)-approximation-stability. Indeed, in Theorem 5.1 of their paper, [57] show that their ϵ-separatedness assumption implies that any near-optimal solution to k-means is $O(\epsilon^2)$-close to the k-means optimal clustering. However,

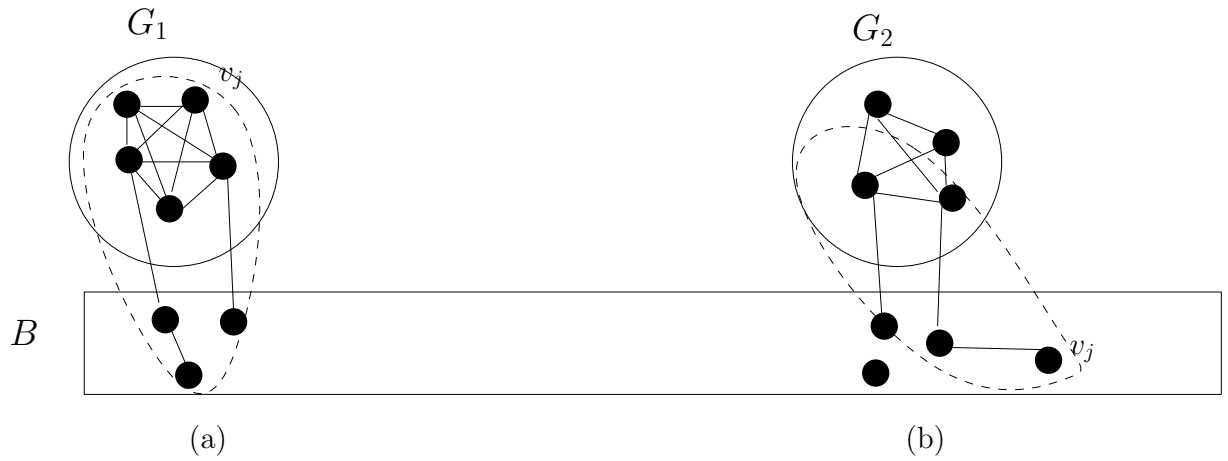

Figure 5: If the greedy algorithm chooses a good vertex v_j as in (a), we get the entire good set of points from that cluster. If v_j is a bad point as in (b), the missed good points can be charged to bad points.

the converse is not necessarily the case: an instance could satisfy approximation-stability without being ϵ-separated.[5] [21] presents a specific example of points in Euclidean space with $c = 2$. In fact, for the case that k is much larger than $1/\epsilon$, the difference between the two properties can be more substantial. See Figure 6 for an example. In addition, algorithms for approximation stability have been successfully applied in clustering problems arising in computational biology [68] (See Section 5.8 for details).

[17] study center based clustering objectives and define a notion of stability called *α-weak deletion* stability. A clustering instance is stable under this notion if in the optimal clustering merging any two clusters into one increases the cost by a multiplicative factor of $(1 + \alpha)$. This a broad notion of stability that generalizes both the ϵ-separability notion studied in section 5.1 and the approximation stability in the case of large cluster sizes. Remarkably, [17] show that for such instances of k-median and k-means one can design a $(1+\epsilon)$ approximation algorithm for any $\epsilon > 0$. This leads to immediate improvements over the works of [20] (for the case of large clusters) and of [57]. However, the runtime of the resulting algorithm depends polynomially in n and k and exponentially in the parameters $1/\alpha$ and $1/\epsilon$, so the simpler algorithms of [17] and [20] are more suitable for scenarios where one expects the stronger properties to hold. See Section 5.8 for further discussion. [3] also study various notions of clusterability of a dataset and present algorithms for such stable instances.

Kumar and Kannan [49] consider the problem of recovering a target clustering under deterministic separation conditions that are motivated by the k-means objective and by Gaussian and related mixture models. They consider the setting of points in Euclidean space, and show that if the projection of any data point onto the line joining the mean of its cluster in the target clustering to the mean of any other cluster of the target is $\Omega(k)$ standard deviations closer to its own mean than the other mean, then they can recover the target clusters in polynomial time. This condition was further analyzed and reduced by work of [18]. This separation condition is formally incomparable to approximation-stability (even restricting to the case of k-means with points in Euclidean space). In particular,

[5] [57] shows an implication in this direction (Theorem 5.2); however, this implication requires a substantially stronger condition, namely that data satisfy (c, ϵ)-approximation-stability for $c = 1/\epsilon^2$ (and that target clusters be large). In contrast, the primary interest of [21] in the case where c is below the threshold for existence of worst-case approximation algorithms.

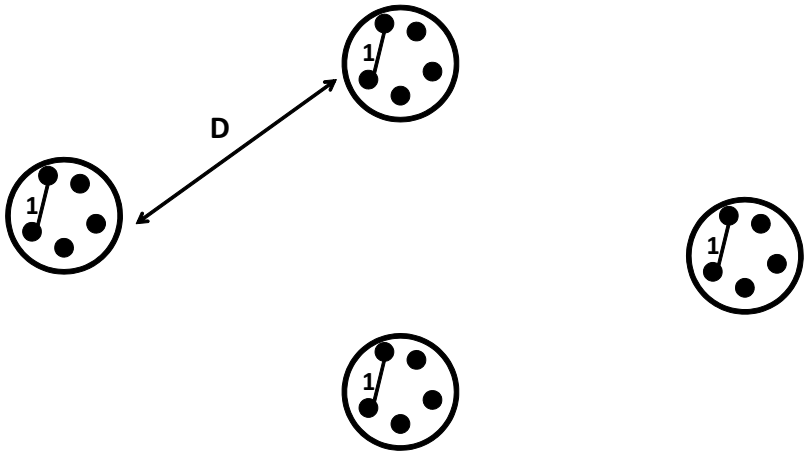

Figure 6: Suppose ϵ is a small constant, and consider a clustering instance in which the target consists of $k = \sqrt{n}$ clusters with $\sqrt{n}$ points each, such that all points in the same cluster have distance 1 and all points in different clusters have distance $D + 1$ where D is a large constant. Then, merging two clusters increases the cost additively by $\Theta(\sqrt{n})$, since D is a constant. Consequently, the optimal $(k-1)$-means/median solution is just a factor $1 + O(1/\sqrt{n})$ more expensive than the optimal k-means/median clustering. However, for D sufficiently large compared to $1/\epsilon$, this example satisfies $(2, \epsilon)$-approximation-stability or even $(1/\epsilon, \epsilon)$-approximation-stability – see [21] for formal details.

if the dimension is low and k is large compared to $1/\epsilon$, then this condition can require more separation than approximation-stability (e.g., with k well-spaced clusters of unit radius approximation-stability would require separation only $O(1/\epsilon)$ and independent of k – see [21] for an example). On the other hand if the clusters are high-dimensional, then this condition can require less separation than approximation-stability since the ratio of projected distances will be more pronounced than the ratios of distances in the original space.

Bilu and Linial [25] consider inputs satisfying the condition that the optimal solution to the objective remains optimal even after bounded perturbations to the input weight matrix. This condition is known as *perturbation resilience*. Bilu and Linial [25] give an algorithm for a different clustering objective known as maxcut. The maxcut objective asks for a 2 partitioning of a graph such the total number of edges going between the two pieces is maximized. The authors show that the maxcut objective is easy under the assumption that the optimal solution is stable to $O(n^{2/3})$-factor multiplicative perturbations to the edge weights. The work of Makarychev et al. [52] subsequently reduced the required resilience factor to $O(\sqrt{\log n})$. In [18] the authors study perturbation resilience for center-based clustering objectives such as k-median and k-means, and give an algorithm that finds the optimal solution when the input is stable to only factor-3 perturbations. This factor is improved to $1 + \sqrt{2}$ by [22], who also design algorithms under a relaxed (c, ϵ)-stability to perturbations condition in which the optimal solution need not be identical on the c-perturbed instance, but may change on an ϵ fraction of the points (in this case, the algorithms require $c = 4$). Note that for the k-median objective, (c, ϵ)-approximation-stability with respect to $\mathcal{C}^*$ implies (c, ϵ)-stability to perturbations because an optimal solution in a c-perturbed instance is guaranteed to be a c-approximation on the original instance;[6] so, (c, ϵ)-stability to per-

[6]In particular, a c-perturbed instance $\tilde{d}$ satisfies $d(x, y) \leq \tilde{d}(x, y) \leq cd(x, y)$ for all points x, y. So, using Φ to denote cost in the original instance, $\tilde{\Phi}$ to denote cost in the perturbed instance and using $\tilde{\mathcal{C}}$ to denote the optimal clustering under $\tilde{\Phi}$, we

turbations is a weaker condition. Similarly, for k-means, (c, ϵ)-stability to perturbations is implied by (c^2, ϵ)-approximation-stability. However, as noted above, the values of c known to lead to efficient clustering in the case of stability to perturbations are larger than for approximation-stability, where any constant $c > 1$ suffices.

5.7 Runtime Analysis

Below we provide the run time guarantees of the various algorithms discussed so far. While these may be improved with appropriate data structures, we assume here a straightforward implementation in which computing the distance between two data points takes time $O(d)$, as does adding or averaging two data points. For example, computing a step of Lloyd's algorithm requires assigning each of the n data points to its nearest center, which in turn requires taking the minimum of k distances per data point (so $O(nkd)$ time total), and then resetting each center to the average of all data points assigned to it (so $O(nd)$ time total). This gives Lloyd's algorithm a running time of $O(nkd)$ per iteration. The k-means++ algorithm has only a seed-selection step, which can be run in time $O(nd)$ per seed by remembering the minimum distances of each point to the previous seeds, so it has a total time of $O(nkd)$.

For the ϵ-separability algorithm, to obtain the sampling probabilities for the first two seeds one can compute all pairwise distances at cost of $O(n^2d)$. Obtaining the rest of the seeds is faster since one only needs to compute distances to previous seeds, so this takes time $O(ndk)$. Finally there is a greedy deletion procedure at time $O(ndk)$ per step for $O(k)$ steps. So the overall time is $O(n^2d + ndk^2)$.

For the approximation-stability algorithm, creating a graph of distances takes time $O(n^2d)$, after which creating the threshold graph takes time $O(n^2)$ if one knows the value of d_{crit}. For the rest of the algorithm, each step takes time $O(n)$ to find the highest-degree vertex, and then time proportional to the number of edges examined to remove the vertex and its neighbors. Over the entire remainder of the algorithm this takes time $O(n^2)$ total. If the value of d_{crit} is not known, one can try $O(n)$ values, taking the best solution. This gives an overall time of $O(n^3 + n^2d)$.

Finally, for local search, one can first create a graph of distances in time $O(n^2d)$. Each local swap step has $O(nk)$ pairs (x, y) to try, and for each pair one can compute its cost in time $O(nk)$ by computing the minimum distance of each data point to the proposed k centers. So, the algorithm can be run in time $O(n^2k^2)$ per iteration. The total number of iterations is at most $poly(n)$ [7] so the overall running time is at most $O(n^2d + n^2k^2poly(n))$. As can be seen from the table below the algorithms become more and more computationally expensive if one needs formal guarantees on a larger instance space. For example, the local search algorithm provides worst case approximation guarantees on all instances but is very slow. On the other hand Lloyd's method and k-means++ are very fast but provide bad worst case guarantees, especially when the number of clusters k is large. Algorithms based on stability notions aim to provide the best of both worlds by being fast and provably good on well behaved instances. In the conclusion section 7 we outline a guideline for practitioners when working with the various clustering assumptions.

have $\Phi(\tilde{C}) \leq \tilde{\Phi}(\tilde{\mathcal{C}}) \leq \tilde{\Phi}(\mathcal{C}^*) \leq c\Phi(\mathcal{C}^*)$.

[7] The actual number of iterations depend upon the cost of the initial solution and the stopping condition.

Method	Runtime
Lloyd's	$O(nkd)) \times (\#iterations)$
k-means++	$O(nkd)$
ϵ-separability	$O(n^2d + ndk^2)$
Approximation stability	$O(n^3 + n^2d)$
Local search	$O(n^2d + n^2k^2poly(n))$

Table 1: A run time analysis of various algorithms discussed in the chapter. The running time degrades as one requires formal guarantees on larger instance spaces.

5.8 Extensions

Variants of the k-means objective
k-means clustering is the most popular methods for vector quantization which is used in encoding speech signals and data compression [39]. There have been variants of the k-means algorithm called fuzzy kmeans which allow each point to have a degree of membership into various clusters [24]. This modified k-means objective is popular for image segmentation [1, 64]. There have also been experiments on speeding up the Lloyd's method by updating centers at each step by only choosing a random sample of the entire dataset [37]. [26] present an empirical study on the convergence properties of Lloyd's method. Rasmussen [60] contains a discussion of k-means clustering for information retrieval. [35] present an empirical comparison of k-means and spectral clustering methods. [59] study a modified k-means objective with an additional penalty for the number of clusters chosen. They motivate the new objective as a way to solve the cluster selection problem. This approach is inspired by the Bayesian model selection procedures [62]. For further details on the applications of k-means, refer to Chapters 1.2 and 2.3.

k-means++: Streaming and Parallel versions of k-means
As we saw in section 2, careful seeding is crucial in order for the Lloyd's method to succeed. One such method is proposed in the k-means++ algorithm. Using the seed centers output by k-means++ one can immediately guarantee an $O(\log k)$ approximation to the k-means objective. However k-means++ is an iterative method which needs to be repeated k times in order to get a good set of seed points. This makes it undesirable for use in applications involving massive datasets with thousands of clusters. This problem is overcome in [19] where the authors propose a scalable and parallel version of kmeans++. The new algorithm runs in much fewer iterations and chooses more than one seed point at each step. The authors experimentally demonstrate that this leads to much better computational performance in practice without losing out on the solution quality. In [6] the authors design an algorithm for k-means which makes a single pass over the data. This makes it much more suitable for applications where one needs to process data in the streaming model. The authors show that if one is allowed to store a little more than k centers ($O(k \log k)$) then one can also achieve good approximation guarantees and at the same time have an extremely efficient algorithm. They experimentally demonstrate that the proposed method is much faster than known implementations of the Lloyd's method. There has been subsequent work on improving the approximation factors and making the algorithms more practical [63].

Approximation Stability in practice
Motivated by clustering applications in computational biology, [68] analyze (c, ϵ)-approximation-stability in a model with unknown distance information where one can only make a limited number of *one versus all* queries. [68] design an algorithm that given (c, ϵ)-approx-

imation-stability for the k-median objective finds a clustering that is very close to the target by using only $O(k)$ one-versus-all queries in the large cluster case, and in addition is faster than the algorithm we present here. In particular, the algorithm for the large clusters case described in [20] (similar to the one we described in Section 5.4 for the k-means objective) can be implemented in $O(|S|^3)$ time, while the one proposed in [68] runs in time $O(|S|k(k + \log|S|))$. [68] use their algorithm to cluster biological datasets in the Pfam [38] and SCOP [56] databases, where the points are proteins and distances are inversely proportional to their sequence similarity. This setting nicely fits the one-versus all queries model because one can use a fast sequence database search program to query a sequence against an entire dataset. The Pfam [38] and SCOP [56] databases are used in biology to observe evolutionary relationships between proteins and to find close relatives of particular proteins. [68] find that for one of these sources they can obtain clusterings that almost exactly match the given classification, and for the other the performance of their algorithm is comparable to that of the best known algorithms using the full distance matrix.

6 Mixture Models

In the previous sections we saw worst case approximation algorithms for various clustering objectives. We also saw examples of how assumptions on the nature of the optimal solution can lead to much better approximation algorithms. In this section we will study a different assumption on how the data is generated in the first place. In the machine learning literature, such assumptions take the form of a probabilistic model for generating a clustering instance. The goal is to cluster correctly (with high probability) an instance generated from the particular model. The most famous and well studied example of this is the Gaussian Mixture Model (GMM)[46]. This will be the main focus of this section. We will illustrate conditions under which datasets arising from such a mixture model can be provably clustered.

Gaussian Mixture Model A univariate Gaussian random variable X, with mean μ and variance σ^2 has the density function $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{\sigma^2}}$. Similarly, a multivariate Gaussian random variable, $\mathbf{X} \in \Re^n$ has the density function

$$f(\mathbf{x}) = \frac{1}{|\Sigma|^{1/2}(2\pi)^{n/2}} e^{\left(\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}.$$

Here $\boldsymbol{\mu} \in \Re^n$ is called the mean vector and Σ is the $n \times n$ covariance matrix. A special case is the spherical Gaussian for which $\Sigma = \sigma^2 I_n$. Here σ^2 refers to the variance of the Gaussian in any given direction. Consider k n-dimensional Gaussian distributions, $\mathcal{N}(\boldsymbol{\mu_1}, \Sigma_1), \mathcal{N}(\boldsymbol{\mu_2}, \Sigma_2), \cdots, \mathcal{N}(\boldsymbol{\mu_k}, \Sigma_k)$. A Gaussian mixture model $\mathcal{M}$ refers to the distribution obtained from a convex combination of such Gaussian. More specifically

$$\mathcal{M} = w_1\mathcal{N}(\boldsymbol{\mu_1}, \Sigma_1) + w_2\mathcal{N}(\boldsymbol{\mu_2}, \Sigma_2) + \cdots w_k\mathcal{N}(\boldsymbol{\mu_k}, \Sigma_k).$$

Here $w_i \geq 0$, are called the mixing weights and satisfy $\sum_i w_i = 1$. One can think of a point being generated from $\mathcal{M}$ by first choosing a component Gaussian i, with probability w_i, and then generating a point from the corresponding Gaussian distribution $\mathcal{N}(\boldsymbol{\mu_i}, \Sigma_i)$. Given a data set of m points coming from such a mixture model, a fairly natural question

is to recover the individual components of the mixture model. This is a clustering problem where one wants to cluster the points into k clusters such that the points drawn from the same Gaussian are in a single partition. Notice that unlike in the previous sections, the algorithms designed for mixture models will have probabilistic guarantees. In other words, we would like the clustering algorithm to recover, with high probability, the individual components. Here the probability is over the draw of the m sample points. Another problem one could ask is to approximate the parameters (mean, variance) of each individual component Gaussian. This is known as the parameter estimation problem. It is easy to see that if one could solve the clustering problem approximately optimally, then estimating the parameters of each individual component is also easy. Conversely, after doing parameter estimation one can easily compute the Bayes optimal clustering. To study the clustering problem, one typically assumes separation conditions among the component Gaussians which limit the amount of overlap between them. The most common among them is to assume that the mean vectors of the component Gaussians are far apart. However, there are also scenarios when such separation conditions do not hold (consider two Gaussian which are aligned in an 'X' shape), yet the data can be clustered well. In order to do this, one first does parameter estimation which needs much weaker assumptions. After estimating the parameters, the optimal clustering can be recovered. This is an important reason to study parameter estimation. In the next section we will see examples of some separation conditions and the corresponding clustering algorithms that one can use. Later, we will also look at recent work on parameter estimation under minimal separation conditions.

6.1 Clustering methods

In this section we will look at distance based clustering algorithms for learning a mixture of Gaussians. For simplicity, we will start with the case of k spherical Gaussians in $\Re^n$ with means $\{\boldsymbol{\mu_1}, \boldsymbol{\mu_2}, \cdots, \boldsymbol{\mu_k}\}$ and variance $\Sigma = \sigma^2 I_n$. The algorithms we describe will work under the assumption that the means are far apart. We will call this as the center separation property:

Definition 5 (Center Separation)**.** *A mixture of k identical spherical Gaussians satisfies center separation if $\forall i \neq j$,*

$$\Delta_{i,j} = ||\mu_i - \mu_j|| > \beta_{i,j}\sigma$$

The quantity $\beta_{i,j}$ typically depends on k the number of clusters, n the dimensionality of the dataset and w_{min}, the minimum mixing weight. If the spherical Gaussians have different variances σ_i's, the R.H.S. is replaced by $\beta_{i,j}(\sigma_i + \sigma_j)$. For the case of general Gaussians, σ_i will denote the maximum variance of Gaussian i in any particular direction. One of the earliest results using center separation for clustering is by Dasgupta [32]. We will start with a simple condition that $\beta_{i,j} = C\sqrt{n}$, for some constant $C > 4$ and will also assume that $w_{min} = \Omega(1/k)$. Let's consider a typical point x from a particular Gaussian $\mathcal{N}(\boldsymbol{\mu_i}, \sigma^2 I_n)$. We have $E[||X - \boldsymbol{\mu_i}||^2] = E[\sum_{d=1}^{n} |x_d - \mu_{id}|^2] = n\sigma^2$. Now consider two typical points x and y from two different Gaussians $\mathcal{N}(\boldsymbol{\mu_i}, \sigma^2 I_n)$ and $\mathcal{N}(\boldsymbol{\mu_j}, \sigma^2 I_n)$. We have

$$\begin{aligned} E[||X - Y||^2] &= E[||X - \boldsymbol{\mu_i} + \boldsymbol{\mu_i} - \boldsymbol{\mu_j} - (Y - \boldsymbol{\mu_j})||^2] \\ &= E[||X - \boldsymbol{\mu_i}||^2] + E[||Y - \boldsymbol{\mu_j}||^2] + ||\mu_i - \mu_j||^2 \\ &\geq 2n\sigma^2 + C^2\sigma^2 n \end{aligned}$$

For C large enough (say $C > 4$), we will have that for any two typical points x, y in the same cluster, $||x-y||^2 \leq 2\sigma^2 n$. And for any two points in different clusters $||x-y||^2 > 18\sigma^2 n$. Using standard concentration bounds we can say that for a sample of size $poly(n)$, with high probability, all points from a single Gaussian will be closer to each other, than to points from other Gaussians. In this case one could simply create a graph by connecting any two points x, y such that $||x-y||^2 \leq 2\sigma^2 n$. It is easy to see that the connected components in this graph will correspond precisely to the individual components of the mixture model. If C is smaller, say 2, one needs a stronger concentration result [10] mentioned below

Lemma 19. *If x, y are picked independently from $N(\boldsymbol{\mu_i}, \sigma^2 I_n)$, then with probability $1-1/n^3$, $||x-y||^2 \in [2\sigma^2 n(1 - \frac{4\log(n)}{\sqrt{n}}), 2\sigma^2 n(1 + \frac{5\log(n)}{\sqrt{n}})]$.*

Also, as before, one can show that with high probability, for x and y from two different Gaussians, we have $||x-y||^2 > 2\sigma^2 n(1 + \frac{4\log(n)}{\sqrt{n}})$. From this it follows that if r is the minimum distance between any two points in the sample, then for any x in Gaussian i and any y in the same Gaussian, we have $||x-y||^2 \leq (1 + \frac{4.5\log(n)}{\sqrt{n}})r$. And for a point z in any other Gaussian we have $||x-z||^2 > (1+\frac{4.5\log(n)}{\sqrt{n}})r$. This suggests the following algorithm

Algorithm CLUSTER SPHERICAL GAUSSIANS

1. Let $\mathcal{D}$ be the set of all sample points.
2. **For:** $i = 1$ to k,
   (a) Let x_0 and y_0 be such that $\|x_0 - y_0\|^2 = r = \min_{x,y \in \mathcal{D}} \|x-y\|^2$.
   (b) Let $T = \{y \in \mathcal{D} : \|x_0 - y\|^2 \leq r(1 + \frac{4.5\log n}{\sqrt{n}})$.
   (c) **Output:** T as one of the clusters.

Handling smaller c

For smaller values of C, for example $C < 1$, one cannot in general say that the above strong concentration will hold true. In fact, in order to correctly classify the points, we might need to see points which are much closer to the center of a Gaussain (say at distance less than $\frac{1}{2}\sigma\sqrt{n}$). However, most of the mass of a Gaussian lies in a thin shell around radius of $\sigma\sqrt{n}$. Hence, one might have to see exponentially many samples in order to get a good classification. Dasgupta [32] solves this problem by first projecting the data onto a random $d = O(\log(k)/\epsilon^2)$ dimensional subspace. This has the effect that the center separation property is still preserved up to a factor of $(1-\epsilon)$. One can now do distance based clustering in this subspace as the number of samples needed will be proportional to 2^d instead of 2^n.

General Gaussians The results of Dasgupta were extend by Arora and Kannan [10] to the case of general Gaussians. They also managed to reduce the required separation between means. They assumed that $\beta_{i,j} = \Omega(\log(n))(R_i + R_j)(\sigma_i + \sigma_j)$. As mentioned before, σ_i denotes the maximum variance of Gaussian i in any direction. R_i denotes the median radius of Gaussian i[8]. For the case of spherical gaussians, this separation becomes $\Omega(n^{1/4}\log(n)(\sigma_i +$

[8]The radius such that the probability mass within R_i equals 1/2

$\sigma_j)$). Arora and Kannan use isoperimetric inequalities to get strong concentration results for such Gaussians. In particular they show that

Theorem 20. *Given $\beta_{i,j} = \Omega(\log(n)(R_i + R_j))$, there exists a polynomial time algorithm which given at least $m = \frac{n^2k^2}{\delta^2 {w^6}_{min}}$ samples from a mixture of k general Gaussians, solves the clustering problem exactly with probability $(1 - \delta)$.*

Proof Intuition: The first step is to generalize Lemma 19 for the case of general Gaussians. In particular one can show that for x, y are picked at random from a general Gaussian i, with median radius R_i and maximum variance σ_i, we have with high probability

$$2{R_i}^2 - 18\log(n)\sigma_i R_i \leq ||x - y||^2 \leq 2(R_i + 20\log(n)\sigma_i)^2$$

Similarly, for x, y from different Gaussians i and j, we have with high probability

$$||x - y||^2 > 2min({R_i}^2, {R_j}^2) + 120\log(n)(\sigma_i + \sigma_j)(R_i + R_j) + \Omega((\log(n))^2({\sigma_i}^2 + {\sigma_j}^2).$$

The above concentration results imply (w.h.p.) that pairwise distances within points from a Gaussian i lie in an interval I_i and distances between Gaussians $I_{i,j}$ lie in the interval $I_{i,j}$. Furthermore, $I_{i,j}$ will be disjoint from the interval corresponding to the Gaussian with smaller value of R_i (Need a figure here). In particular, if one looks at balls of increasing radius around a point from the Gaussian with minimum radius, σ_i, there will be a stage when there exists a gap: i.e., increasing the radius slightly does not include any more points. From the above lemmas, this gap will be roughly $\Omega(\sigma_i)$. Hence, at this stage, we can remove this Gaussian from the data and recurse. This property suggests the following algorithm outline.

Algorithm Cluster General Gaussians

1. Let r be the smallest radius such that $|B(x, r)| > \frac{3}{4} w_{min}|S|$, for some $x \in \mathcal{D}$. Here $|S|$ is the size of dataset and $B(x, r)$ denotes the ball of radius r around x.
2. Let σ denote the maximum variance of the Gaussian with the least radius. Let $\gamma = O(\sqrt{w_{min}\sigma})$.
3. **While** $\mathcal{D}$ is non-empty,
   (a) Let s be such that $|B(x, r + s\gamma)| \cap \mathcal{D} = |B(x, r + (s - 1)\gamma)|$.
   (b) Remove a set T containing all the points from S which are in $B(x, r + s\gamma\log(n))$.
   (c) **Output:** T as one of the cluster.

One point to mention is that one does not really know beforehand the value of sigma at each iteration. Arora and Kannan [10] get around this by estimating the variance from the data in the ball $B(x, r)$. They then show that this estimate is good enough for the algorithm to work.

6.2 Spectral Algorithms

The algorithms mentioned in the above section need the center separation to grow polynomially with n. This is prohibitively large especially in cases when $k \ll n$. In this section, we look at how spectral techniques can be used to only require the separation to grow with k instead of n.

Algorithmic Intuition In order to remove the dependence on n we would like to project the data such that points from the same Gaussian become much closer while still maintaining the large separation between means. One idea is to do a random projection. However, random projections from n to d dimensions scale each squared distance equally (by factor d/n) and will not give us any advantage. However, consider the case of two spherical gaussians with means $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ and variance $\sigma^2 I_n$. Consider projecting all the points to the line joining $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$. Now consider any random point x from the first Gaussian. For any unit vector along the line joining $\boldsymbol{\mu_1}$ and $\boldsymbol{\mu_2}$ we have that $(x - \boldsymbol{\mu_1}).v$ behaves like a 1-dimensional Gaussian with mean 0 and variance σ^2. Hence the expected distance of a point x from its mean becomes σ^2. This means that for any two points in the same Gaussian, the expected squared distance becomes $4\sigma^2$ (as opposed to $2n\sigma^2$). However, the distance between the means remains the same. In fact the above claim is true if we project onto any subspace containing the means. This subspace is exactly characterized by the Singular Value Decomposition (SVD) of the data matrix. This suggests the following algorithm

Algorithm Spectral Clustering

1. Compute the SVD decomposition of the data.
2. Project the data onto the space of top-k right singular vectors.
3. Run a distance based clustering method in this projected space.

Such spectral algorithms were proposed by Vempala and Wang [67] who reduced the separation for spherical Gaussians to $\beta_{i,j} = \Omega(k^{1/4}(\log(n/w_{min}))^{1/4})$. The case of general Gaussians was studied in [2] who give efficient clustering algorithms for $\beta_{i,j} = (\frac{1}{\sqrt{min(w_i,w_j)}} + \sqrt{k\log(kmin(2^k,n))})$. [45] give algorithms for general Gaussians for $\beta_{i,j} = \frac{k^{3/2}}{{w_{min}}^2}$.

6.3 Parameter Estimation

In the previous sections we looked at the problem of clustering points from a Gaussian Mixture Model. Another important problem is that of estimating the parameters of the component Gaussians. These parameters refer to the mixture weights w_i's, mean vectors μ_i's and the covariance matrices Σ_i's. As mentioned before, if one could do efficiently get a good clustering, then the parameter estimation problem is solved by simply producing empirical estimates from the corresponding clusters. However, there could be scenarios when it is not possible to produce a good clustering. For, ex. consider two one dimensional gaussians with mean 0 and variance σ^2 and $2\sigma^2$. These gaussians have a large overlap and any clustering method will inherently have a large error. On the other hand, let's look at the statistical distance between the two gaussians, i.e., $\int_x |f_1(x) - f_2(x)|dx$. This measures

how much one distribution dominates the other one. It is easy to see that in this case the Gaussian with the higher variance will dominate the other Gaussian almost everywhere. Hence the statistical distance is close to 1. This suggests that information theoretically, one should be able to estimate the parameters of these two mixtures. In this section, we will look at some recent work of Kalai, Moitra, Valiant [44] and Moitra, Valiant [55] in efficient algorithms for estimating the parameters of a Gaussian mixture model. These works make minimal assumption on the nature of the data, namely, that the component gaussians have noticeable statistical distance. Similar results were proven in [23] who also gave algorithms for more general distributions.

The case of two Gaussians:
We will first look at the case of 2 Gaussians in $\Re^n$. We will assume that the statistical distance between the gaussians, $D(\mathcal{N}_1, \mathcal{N}_2)$ is noticeable, i.e., $\int_x |f_1(x) - f_2(x)|dx > \alpha$. Kalai et. al [44] show the following theorem

Theorem 21. *Let $\mathcal{M} = w_1\mathcal{N}_1(\boldsymbol{\mu_1}, \Sigma_1) + w_2\mathcal{N}_2(\boldsymbol{\mu_2}, \Sigma_2)$ be an isotropic GMM where $D(\mathcal{N}_1, \mathcal{N}_2) > \alpha$. Then, there is an algorithm which outputs $\mathcal{M}' = w_1'\mathcal{N}'_1(\boldsymbol{\mu_1'}, \Sigma_1') + w_2'\mathcal{N}'_2(\boldsymbol{\mu_2'}, \Sigma_2')$ such that for some permutation $\pi : \{0,1\} \mapsto \{0,1\}$ we have,*

$$\begin{aligned} |w_i - w'_{\pi(i)}| &\leq \epsilon \\ ||\boldsymbol{\mu_i} - \boldsymbol{\mu'_{\pi(i)}}|| &\leq \epsilon \\ ||\Sigma_i - \Sigma'_{\pi(i)}|| &\leq \epsilon \end{aligned}$$

The algorithm runs in time poly$(n, 1/\epsilon, 1/\alpha, 1/w_1, 1/w_2)$.

The condition on the mixture being isotropic is necessary to recover a good additive approximation for the means and the variances since otherwise, one could just scale the data and the estimates will scale proportionately.

Reduction to a one dimensional problem
In order to estimate the mixture parameters, Kalai et. al, reduce the problem to a series of one dimensional learning problems. Consider an arbitrary unit vector v. Suppose we project the data onto the direction of v and let the means of the Gaussians in this projected space be μ_1' and μ_2'. Then we have that $\mu_1 = E[x.v] = E[(x - \boldsymbol{\mu_1}).v] = \boldsymbol{\mu_1}.v$. Hence, the parameters of the original mean vector are linearly related to the mean in the projected space. Similarly, let's perturb v to get $v' = v + \epsilon(e_i + e_j)$. Here e_i and e_j denote the basis vectors corresponding to coordinates i and j. Let ${\sigma_1'}^2$ be the variance of the gaussian in the projected space v'. Then writing ${\sigma_1'}^2 = E[(x.v')^2]$ and expanding, we get that $E[x_i x_j]$ will be linearly related to ${\sigma_1'}^2$, ${\sigma_1}^2$ and the μ_i's. Hence, by estimating the parameters correctly over a series of n^2, one dimensional vectors, one can efficiently recover the original parameters (by solving a system of linear equations).

Solving the one dimensional problem
The one dimensional problem is solved by the method of moments. In particular, define $L_i[\mathcal{M}]$ to be the ith moment for the mixture model $\mathcal{M}$, i.e., $L_i[\mathcal{M}] = E_{x\sim\mathcal{M}}[x^i\mathcal{M}(x)]$. Also define $\hat{L}_i$ to be the empirical ith moment of the data. The algorithm in [44] does a brute force search over the parameter space for the two Gaussians and for a given candidate model $\mathcal{M}'$ computes the first 6 moments. If all the moments are within ϵ of the empirical moments, then

the analysis in [44] shows that the parameters will be $\epsilon^{1/67}$ close to the parameters of the two gaussians. The same claim is also true for learning a mixture of k 1 dimensional gaussians if one goes upto $(4k-2)$ moments [55]. The search space however will be exponential in k. It is shown in [55] that for learning k one dimensional gaussians, this exponential dependence is unavoidable.

Solving the labeling problem
As noted above, the learning algorithm will solve n^2, 1-dimensional problems and get parameter estimates for the two gaussians for each 1-dimensional problem. In order to solve for the parameters of the original gaussians, we need to identify for each gaussian, the corresponding n^2 parameters for each of the subproblems. Kalai et. al do this by arguing that if one projects the two gaussians onto a random direction v, with high enough probability, the corresponding parameters for the two projected gaussians will differ by $poly(\alpha)$. Hence, if one takes small random perturbations of this vector v, the corresponding parameter estimates will be easily distinguishable.

The overall algorithm has the following structure

Algorithm Learning a mixture of two Gaussians

1. Choose a random vector v and choose n^2 random perturbations $v_{i,j}$.
2. For each i, j, project the data onto $v_{i,j}$ and solve the one dimensional problem using the method of moments.
3. Solve the labeling problem to identify the n^2 parameter sets corresponding to a single Gaussian.
4. Solve a system of linear equations on this parameter set to obtain the original parameters.

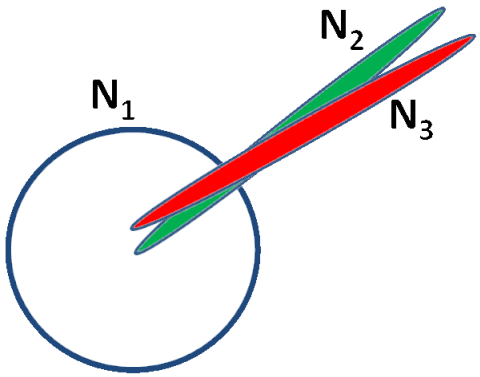

Figure 7: The case of 3 Gaussians.

For the case of more than 2 Gaussians, Moitra and Valiant [55] extend the ideas mentioned above to provide an algorithm for estimating the parameters of a mixture of k Gaussians. For the case of k Gaussians, additional complications arise as it is not true anymore that projecting the k Gaussians to a random 1-dimensional subspace, maintains the statistical distance. For example, consider Figure 7. Here, projecting the data onto a random direction,

will almost surely collapse components 2 and 3. [55] solve this problem by first running a clustering algorithm to separate components 2 and 3 from component 1 and recursively solving the two sub-instances. Once, 2 and 3 have been separated, one can scale the space to ensure that they remain separated over a random projection. The algorithm from [55] has the sample complexity which depends exponentially on k. They also show that this dependence is necessary. One could use the algorithm from [55] to also cluster the points into component Gaussians under minimal assumptions. The sample complexity however, will depend exponentially in k. In contrast, one could algorithms from previous sections to cluster in polynomial time under stronger separation assumptions. The work of [41, 9] removes the exponential dependence on k and designs polynomial time algorithms for clustering data from a GMM under minimal separation assuming only that the mean vectors span a k dimensional subspace. However, their algorithm which is based on Tensor decompositions only works in the case when all the component Gaussians are spherical. It is an open question to get similar result for general Gaussians. There has also been work on clustering points from a mixture of other distributions. [31, 30] gave algorithms for clustering a mixture of heavy tailed distributions. [27] gave algorithms for clustering a mixture of 2 Gaussians assuming only that the two distributions are separated by a hyperplane. The recent work of [49] studies a deterministic separation condition on a set of points and show that any set of points satisfying this condition can be clustered accurately. Using this they easily derive many previously known results for clustering mixture of Gaussians as a corollary.

7 Conclusion

In this chapter we presented a selection of recent work on clustering problems in the computer science community. As is evident, the focus of all these works is on providing efficient algorithms with rigorous guarantees for various clustering problems. In many cases, these guarantees depend on the specific structure and properties of the instance at hand which are captured by stability assumptions and/or distributional assumptions. The study of different stability assumptions also provide insights into the structural properties of real world data and in some cases also lead to practically useful algorithms [68]. As discussed in Section 5.6 different assumptions are suited for different kinds of data and they relate to each other in interesting ways. For instance, perturbation resilience is a much weaker assumption than both ϵ-separability and approximation stability. However, we have algorithms with much stronger guarantees for the latter two. As a practitioner one is often torn between using algorithms with formal guarantees (which are typically slower) vs. fast heuristics like the Lloyd's method. When dealing with data which may satisfy any of the stability notions proposed in this chapter, a general rule of thumb we suggest is to run the algorithms proposed in this chapter on a smaller random subset of the data and use the solution obtained to initialize fast heuristics like the Lloyd's method. Current research on clustering algorithms continues to explore more realistic notions of data stability and their implications for practical clustering scenarios.

References

[1] Gibbs random fields, fuzzy clustering, and the unsupervised segmentation of textured images. *CVGIP: Graphical Models and Image Processing*, 55(1):1 – 19, 1993.

[2] D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *Proceedings of the Eighteenth Annual Conference on Learning Theory*, 2005.
[3] Margareta Ackerman and Shai Ben-David. Clusterability: A theoretical study. *Journal of Machine Learning Research - Proceedings Track*, 5, 2009.
[4] Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. k-means++ under approximation stability. *The 10th annual conference on Theory and Applications of Models of Computation*, 2013.
[5] Ankit Aggarwal, Amit Deshpande, and Ravi Kannan. Adaptive sampling for k-means clustering. In *Proceedings of the 12th International Workshop and 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, APPROX '09 / RANDOM '09, 2009.
[6] N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In *Advances in Neural Information Processing Systems*, 2009.
[7] Paola Alimonti. Non-oblivious local search for graph and hypergraph coloring problems. In *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science. 1995.
[8] Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Mach. Learn.*
[9] Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. Technical report, http://arxiv.org/abs/1210.7559, 2012.
[10] S. Arora and R. Kannan. Learning mixtures of arbitrary gaussians. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, 2005.
[11] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*. 1999.
[12] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
[13] David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *Journal of the ACM*, 58(5), October 2011.
[14] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, 2006.
[15] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.
[16] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.
[17] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 2010.
[18] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2), January 2012.
[19] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. In *Proceedings of the 38th International Conference on Very Large Databases*, 2012.
[20] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2009.
[21] M.-F. Balcan, A. Blum, and A. Gupta. Clustering under approximation stability. In *Journal of the ACM*, 2013.
[22] Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *Proceedings of the 39th International Colloquium on Automata, Languages and Program-*

ming, 2012.

[23] Mikhail Belkin and Kaushik Sinha. Polynomial learning of distribution families. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[24] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[25] Yonatan Bilu and Nathan Linial. Are stable instances easy? In *Proceedings of the First Symposium on Innovations in Computer Science*, 2010.

[26] Lon Bottou and Yoshua Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*, pages 585–592. MIT Press, 1995.

[27] Spencer Charles Brubaker and Santosh Vempala. Isotropic PCA and affine-invariant clustering. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008.

[28] Barun Chandra, Howard Karloff, and Craig Tovey. New results on the old k-opt algorithm for the tsp. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, 1994.

[29] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoy. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, 1999.

[30] Kamalika Chaudhuri and Satish Rao. Beyond gaussians: Spectral methods for learning mixtures of heavy-tailed distributions. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

[31] Kamalika Chaudhuri and Satish Rao. Learning mixtures of product distributions using correlations and independence. In *Proceedings of the 21st Annual Conference on Learning Theory*, 2008.

[32] S. Dasgupta. Learning mixtures of gaussians. In *Proceedings of The 40th Annual Symposium on Foundations of Computer Science*, 1999.

[33] S. Dasgupta. The hardness of k-means clustering. Technical report, University of California, San Diego, 2008.

[34] W. Fernandez de la Vega, Marek Karpinski, Claire Kenyon, and Yuval Rabani. Approximation schemes for clustering problems. In *Proceedins of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, 2003.

[35] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

[36] Doratha E. Drake and Stefan Hougardy. Linear time local improvements for weighted matchings in graphs. In *Proceedings of the 2nd international conference on Experimental and efficient algorithms*, 2003.

[37] Vance Faber. Clustering and the Continuous k-Means Algorithm. 1994.

[38] R.D. Finn, J. Mistry, J. Tate, P. Coggill, A. Heger andJ.E. Pollington, O.L. Gavin, P. Gunesekaran, G. Ceric, K. Forslund, L. Holm, E.L. Sonnhammer, S.R. Eddy, and A. Bateman. The pfam protein families database. *Nucleic Acids Research*, 38:D211–222, 2010.

[39] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[40] Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 1990.

[41] Daniel Hsu and Sham M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*, 2013.

[42] 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, 1994.

[43] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, 2002.

[44] Adam Tauman Kalai, Ankur Moitra, and Gregory Valiant. Efficiently learning mixtures of two gaussians. In *Proceedings of the 42th ACM Symposium on Theory of Computing*, 2010.

[45] R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. In *Proceedings of The Eighteenth Annual Conference on Learning Theory*, 2005.

[46] Ravi Kannan and Santosh Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4), 2009.

[47] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, New York, NY, USA, 2002. ACM.

[48] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1 + \epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, Washington, DC, USA, 2004.

[49] Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[50] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, 2013.

[51] S.P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 28(2):129–137, 1982.

[52] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-linial stable instances of max cut and minimum multiway cut. In *SODA*, pages 890–906. SIAM, 2014.

[53] N. Megiddo and K. Supowit. On the complexity of some common geometric location problems. *SIAM Journal on Computing*, 13(1):182–196, 1984.

[54] Marina Meilă. The uniqueness of a good optimum for K-means. In *Proceedings of the International Machine Learning Conference*, pages 625–632, 2006.

[55] Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of gaussians. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[56] A.G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[57] R. Ostrovsky, Y. Rabani, L. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

[58] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *Proceedings of the 32nd annual symposium on Foundations of computer science*, 1991.

[59] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[60] Edie M. Rasmussen. Clustering algorithms. In *Information Retrieval: Data Structures & Algorithms*, pages 419–442. 1992.

[61] Petra Schuurman and Tjark Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS J. on Computing*, 2007.

[62] Gideon Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*,

6(2):461–464, 1978.

[63] Michael Shindler, Alex Wong, and Adam Meyerson. Fast and accurate k-means for large datasets. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*, 2011.

[64] A. H S Solberg, T. Taxt, and A.K. Jain. A markov random field model for classification of multisource satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 1996.

[65] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3), May 2004.

[66] Andrea Vattani. k-means requires exponentially many iterations even in the plane. In *Proceedings of the 25th annual symposium on Computational geometry*, 2009.

[67] S. Vempala and G. Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68(2):841–860, 2004.

[68] K. Voevodski, M. F. Balcan, H. Roeglin, S. Teng, and Y. Xia. Efficient clustering with limited distance information. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[69] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. The top ten algorithms in data mining. *Knowledge and Information Systems*, 2008.