COMPSCI 634: Geometric Algorithms April 10, 2014

Lecture # 24: Clustering

Lecturer: Pankaj Agarwal *Scribe: Nat Kell*

1 Overview

In this lecture, we briefly survey problems in *geometric clustering*. We define the problem formally, give the most commonly used objective functions, discuss the trade-offs between these objectives, and then give some of the more well-known algorithms and results in this area.

2 Introduction

2.1 Problem Definition

One of the most widely-studied problems in geometric optimization is that of *clustering*. As input, we are typically given a finite point set $S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^d$ and an integer $k \geq 1$. Our goal then is to partition S into k subsets $s_1, \ldots, s_k$, or *clusters*, such that each point exists in exactly one subset. Many times in practice, k is not given as part of the input, but for the algorithms described in this lecture we will assume that k is known. In some cases, we may assume k is fixed for all instances of the problem (for example, we will discuss algorithms for the case where $k = 1$ later on).

One can use a variety of objectives functions, but at a high level, we would like for points placed in the same cluster to be close to one another, while points in different clusters should be spaced far apart; these two criteria are referred to as *intra-cluster* and *inter-cluster* distances, respectively. One cannot hope to optimize both these types of distances simultaneously, so a given objective will focus on meeting only one of these criterion. In this lecture, we will look at objectives that optimize intra-cluster distances.

Clustering has a wide array of applications. Any area where data, objects, or structures need to be classified into groups will likely use some form of clustering analysis, e.g., bioinformatics, image analysis, pattern recognition, and data compression. Note that in some cases, it may not be possible to embed the objects of interest into a d-dimensional euclidean space (for example, protein-structures); however, in such instances we can still define a metric over the objects and instead solve the problem in a metric-space setting. Also, for d-dimensional data sets where d is very large, it is unreasonable to expect for points to be separated in every dimension. Therefore in this case one does *projective clustering*, where points are projected into a lower dimensional space and clustered there instead.

For a more comprehensive introduction to clustering algorithms and their applications, see [1, 2].

2.2 Objective Functions

To define an objective that captures the goodness of intra-cluster distances, we need a way of scoring each individual cluster. One way to do this is to, for a given cluster s_j, take the maximum distance between any pair of points in s_j as its score. However, a much more common approach is to pick a representative point for each cluster, usually deemed as the cluster's *center*, and measure distances to this point to produce the score. For our purposes, we will allow for a center to reside outside of S (although, an algorithm we describe later on will always pick centers from points in S).

Thus, the goal the algorithm is to pick k centers $C = \{c_1, \ldots, c_k\}$, where cluster s_j corresponds to center c_j. A point p_i is then placed in cluster j if $j = \arg\min_{1 \le j \le k} \|p_i - c_j\|$. Given this setup, the most straight forward objective is to pick C in order to minimize the following.

$$\text{KCEN}(C) = \max_{1 \le i \le n} \min_{1 \le j \le k} \|p_i - c_j\|$$

The problem of optimizing this objective is known as the *k-centers problem*. Intuitively, if we think of centers in C as post offices and points in S as homes, then $\text{KCEN}(C)$ is the farthest any homeowner has to walk to get to their closest post office if we place post office optimally (given they can always walk a euclidean distance). The k-centers problem is related to another problem known as *the facility location problem*. Here the objective is almost equivalent, but now the algorithm picks the number of centers (or facilities) and pays an extra cost for each facility it places/opens.

As we have seen in previous lectures, any objective that takes the max over all points (or L_∞-norm) will be sensitive to outliers. Thus an alternative objective would be to instead sum the best case point-to-center distances instead; more formally, we pick C in order to minimize:

$$\text{KMED}(C) = \sum_{1 \le i \le n} \min_{1 \le j \le k} \|p_i - c_j\|.$$

Optimizing this objective is known as the *k-medians problem*. However, the pitfall with this approach is that the square root in each euclidean norm term $\|p_i - c_j\|$ makes the problem difficult to work with, so instead we can look at an objective where we square each distance. Formally,

$$\text{KMEAN}(C) = \sum_{1 \le i \le n} \min_{i \le j \le k} \|p_i - c_j\|^2.$$

This final version of the problem (that we will look at) is known as the *k-means* problem. The trade-off here is that squaring each term smooths the objective a bit (so in some sense we have lost some precision).

Note that all three of these problem versions are NP-hard. Thus, one must either fix the value of k or approximate the solution in order to make the problem solvable in polynomial time. We will look at such approaches in the following two sections.

3 Clustering when $k = 1$

In this section, we examine the case where every point is in the same cluster. Here, the problem boils down to optimizing the placement of the only center c.

3.1 $k = 1$ for k-centers

Even when k is left unfixed, a more intuitive formulation of the k-centers problem is the following: place k congruent balls in our space such that each point in S is covered by at least one disk and the radius of each ball, r, is minimized. The reader can verify that these are indeed identical formulations.

Therefore, in the case $k = 1$, the problem becomes that of finding the *minimum enclosing ball* of the point set. Formally defined:

Definition 1. *(Minimum Enclosing Ball Problem,* MEB*): Given a point set $S = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$, output the minimum $r \in \mathbb{R}$ such that for some center $c \in \mathbb{R}^d$ we have $\|p_i - c\| \le r$ for $1 \le i \le n$.*

To define an algorithm MEB, it first helps to modify the problem formulation a bit, which will allow us to then use convex programming techniques. Namely, we would like to formulate the problem such that all of the constraints are linear and the objective is convex. Although we did not argue it formally, the linear programming algorithms we saw in a previous lecture will work for such a convex program and will have the same running-time guarantees

The current problem formulation is given by

$$\begin{aligned} &\min\ r \\ &\text{s.t. } \|p_i - c\| \leq r\ \forall\ i = 1,\ldots,n. \end{aligned} \tag{1}$$

Note that squaring every term in the formulation does not change the optimal solution. Since $\|p_i - c\|^2 = \|p_i\|^2 + \|c\|^2 - 2\langle p_i, c\rangle$ (where $\langle\cdot,\cdot\rangle$ denotes the inner-product of two vectors), we now express the problem as

$$\begin{aligned} &\min\ r^2 \\ &\text{s.t. } \|p_i\|^2 + \|c\|^2 - 2\langle p_i, c\rangle \leq r^2\ \forall\ i = 1,\ldots,n. \end{aligned} \tag{2}$$

This still gives us quadratic terms on our decision variables in both our constraints and objective, and so more manipulation is required. Let $z = r^2 - \|c\|^2$ (and so now we our program decides on z instead of r, but clearly given z and c we can determine r). With this relabeling, the formulation becomes

$$\begin{aligned} &\min\ z + \|c\|^2 \\ &\|p_i\|^2 \leq 2\langle p_i, c\rangle + z\ \forall\ i = 1,\ldots,n. \end{aligned} \tag{3}$$

Observe that now all the constraints are linear, and since we only have a $\|c\|^2$ term in the objective, the objective is convex, as desired. Based on the above discussion, we obtain the following theorem.

Theorem 1. *There exists a polynomial-time algorithm that solves* MEB *in $O(d^3 n + e^{\alpha\sqrt{d}}\log n)$ time, where d is the dimension S, $|S| = n$, and α is a constant.*

Note that our polynomial running time claim is contingent on d being a constant (or ideally small). In cases where d is large, one will typically use a *core-set* S' of the points in S (discussed in a previous lecture) and then cluster points S'. Although we will not argue it here, the following is also a theorem.

Theorem 2. *There exists a corset $S' \subset S$ of size $1/\varepsilon$ such that* $\text{MEB}(S) \leq (1+\varepsilon)\text{MEB}(S')$*, where MEB$(\cdot)$ denotes the radius of the minimum enclosing ball of each point set, respectively.*

3.2 $k = 1$ for k-medians and k-means

As mentioned previously, the k-medians problem is difficult because we must sum a sequence of square rooted terms. In fact, the optimal value of $\text{KMED}(C)$ may not even be algebraic. Therefore, approximate solutions are needed even in the $k = 1$ case.

In contrast, the $k = 1$ case for k-means is very straight forward. Here, the optimal solution is always the centroid (i.e., the average coordinate of all the points in S). Therefore, this version of the problem is solved trivially in $O(dn)$ time.

4 Clustering when k is arbitrary

Now we will examine the case where k is given to algorithm as part of the input. We will first define and analyze a natural greedy algorithm for the k-centers problem and show it is a 2-approximation. Then, we will give a commonly used expectation-maximization (EM) algorithm for the k-means problem.

4.1 2-approximation for k-centers

Informally, our algorithm works as follows. We build a solution by adding one center at a time. Given that we have already placed $h < k$ centers, there will be some point $p \in S$ who is the farthest away from its current center c (and thus $\|p-c\|$ determines our current objective). Ideally, we would like to place the next center c' so p is closer to this new center, i.e., $\|p-c'\| < \|p-c\|$. The most greedy way of doing this is to just make our next center p (therefore for this algorithm, every center in our solution will be a point in S).

The algorithm GREEDY is formally given as follows (where $D[i]$ stores the distance between p_i and its current closest center):

Algorithm 1: GREEDY

```
Set D[i] = ∞ for 1 ≤ i ≤ n
for j = 1 up to k do
    c_j = arg max_{p_i ∈ S} D[i]
    for i = 1 up to n do
        D[i] = min(D[i], ||p_i − c_j||)
    end for
end for
return (C = {c_1, ..., c_k}, max_i D[i])
```

Theorem 3. *The approximation ratio of* GREEDY *is 2.*

Proof. We analyze the optimal solution of a given instance from the disk placing perspective of the problem described in Section 3.1. Consider the set of congruent disks in the optimal placement; denote these disks as $D^* = \{D_1^*, \ldots, D_k^*\}$ where each disk has radius r^* and disk D_j^* has center c_j^* (and thus r^* is the objective that GREEDY is competing against). Based on the placement of centers by GREEDY, there are two cases (refer to Figure 1 to see an illustration of both these cases).

Case 1, *for all $D_j^* \in D$, there exists exactly one center in C that lies in D_j^**: Fix a disk D_j^* and let $c_h \in C$ be the center that lies in D_j^*. We know that for all $p_i \in D_j^*$, $\|p_i - c_j^*\| \leq r^*$. But $c_h \in D_j^*$ as well, so $\|c_h - c_j^*\| \leq r^*$. Therefore by the triangle inequality, $\|p_i - c_h\| \leq \|p_i - c_j^*\| + \|c_j^* - c_h\| \leq 2r^*$. Since this reasoning can be applied to every $D_j^* \in D$ and D covers every point in S (by definition), every point in S can be at most $2r^*$ away from its closest center placed by GREEDY.

Case 2, *there exists a j' such that $D_{j'}^*$ contains at least two centers from C:* Let c_h and c_t denote two of the centers contained in $D_{j'}^*$, and without loss of generality, let $h < t$ (i.e., GREEDY placed c_h before c_t). Note that since c_t and c_h are both contained in D_j^*, $\|c_t - c_h\| \leq 2r^*$ by the triangle inequality.

Consider when c_t was picked to be a center by GREEDY, and let $t(p)$ denote the center used for point $p \in S$ before c_t was placed. Since c_t was picked greedily, it must have been the point farthest from its current

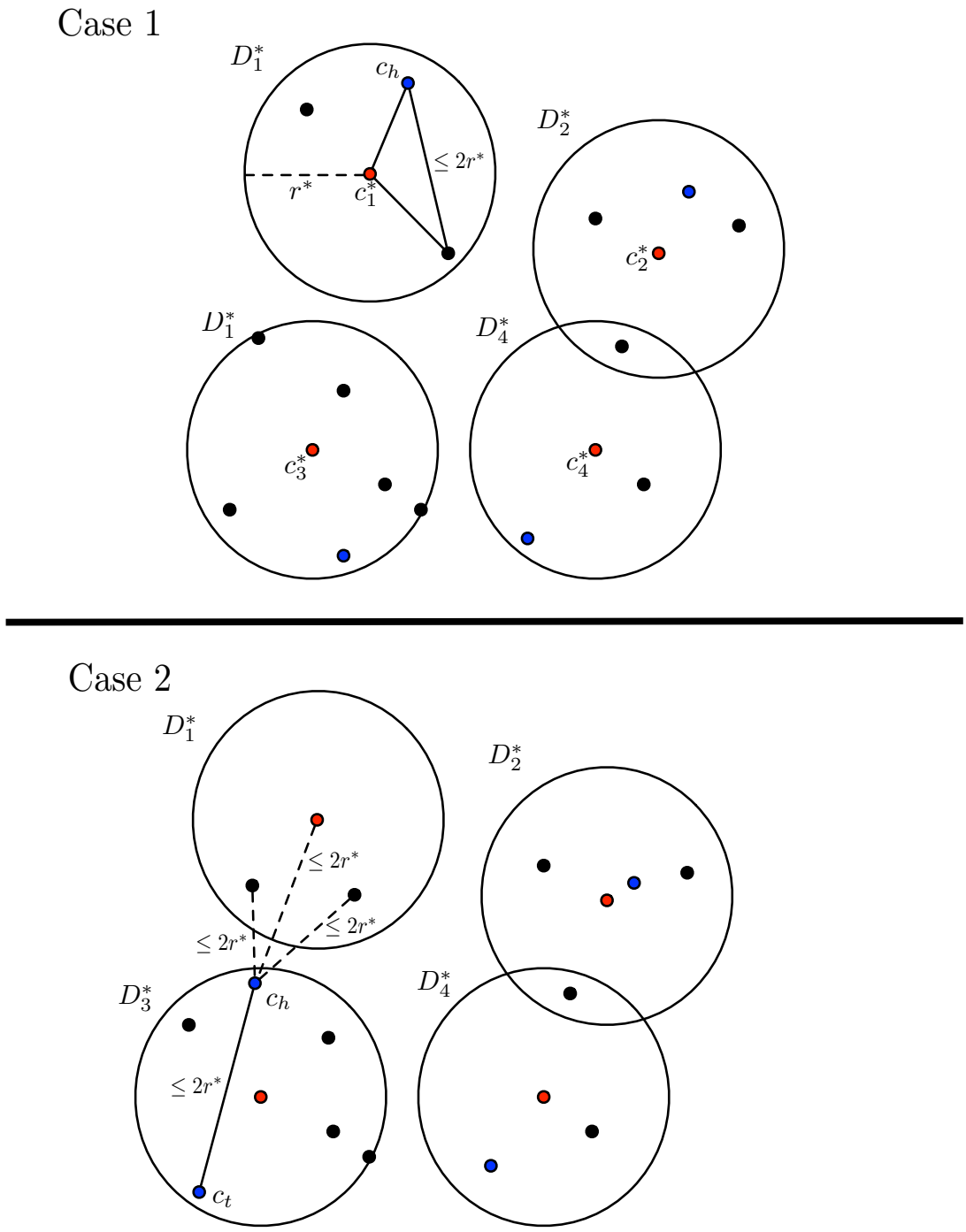

Figure 1: Illustration of Theorem 3. The red and blue points represent centers from the optimal and greedy solutions, respectively. In Case 1, there is exactly one greedy center in each optimal disk $D_j^* \in D$, and thus points cannot be more than $2r^*$ away from their nearest greedy center. In Case 2, there may be optimal disks that do no include any greedy centers (here it is D_1^*). However, since points in D_1^* must have been closer to their center than c_t was to its center when c_t was chosen, points in D_1^* must be within $2r^*$ of their nearest center since $\|c_t - c_h\| \leq 2r^*$ (note that in this example c_h serves as the nearest center for all these points, but in general this is not necessarily the case).

center $t(c_t)$. We also know that center c_h had already been placed, and so the distance between c_t and $t(c_t)$ must be no bigger than the distance between c_t and c_h. Putting these facts together, for all points $p_i \in S$ we obtain

$$\begin{aligned} \|p_i - c_t(p_i)\| &\leq \|c_t - t(c_t)\| && \text{(by greediness)} \\ &\leq \|c_t - c_h\| && \text{(definition of a point's center)} \\ &\leq 2r^* && \text{(since } |c_t - c_h\| \leq 2r^*\text{).} \end{aligned}$$

Since a point can never become farther away from its current center as the algorithm progresses, it follows that every point in S is within $2r^*$ of its final center when the algorithm finishes, as desired.

Finally note that these two cases are sufficient since D^* must cover every point, and therefore if Case 1 does not hold, Case 2 holds by the pigeonhole principle. □

4.2 EM Algorithm for k-means

Finally, we describe an expectation-maximization algorithm for the k-means problem. Although there is no provable bound on the running time or approximation ratio of this algorithm, it is widely used since it tends to work well in practice.

Informally, on a given step we will have a current partition of the points into clusters $s_1, \ldots s_k$. We then compute the centroid $\overline{c_j}$ of cluster s_j for every $j = 1, \ldots, k$ (expectation step). We then use these centroids as our new set of centers, and reassign points to clusters such that a point p lies in the cluster that corresponds to p's nearest centroid (maximization step). We then alternate these two steps until we converge to a fixed solution (we will stop when we reach a local minimum in the solution space, but again, we cannot make any guarantees on how good this solution will be or how quickly we will obtain it).

5 Summary

In this lecture, we gave a brief overview of geometric clustering. We described three different objectives which yield three different versions of the problem: k-centers, k-medians, and k-means. Each of these versions are NP-hard, so we discussed variations that make the problems tractable. In the case where $k = 1$, we gave a convex programming approach which gives a polynomial-time algorithm for the 1-center(s) problem. We then described a 2-approximate greedy algorithm for k-centers problem and then briefly described a commonly used EM-algorithm for the k-means problem, both for the cases where k is arbitrary.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[2] Rui Xu and D. Wunsch, II. Survey of clustering algorithms. *Trans. Neur. Netw.*, 16(3):645–678, May 2005.