Approximation Algorithms **Spring 2019**

Lecture 2: A Greedy 2-approximation for k-center

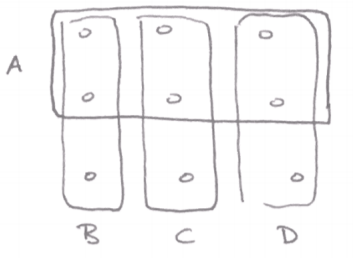
Scribe: Ida Huihan Liu 2/15/2019

In this note, we begin our discussion on greedy approximation algorithms by considering the k-center problem. We provide a 2-approximation algorithm due to Gonzalez [1].

2.1 What is a Greedy Algorithm?

Greedy algorithms construct a solution step by step. At each step of the algorithm, it constructs the next part of the solution by making some decision that is locally the best possible. Consequently, the decision made by the greedy algorithm at each step will always be in the final solution. How does a greedy algorithm make decisions? Usually it is by some information we can retrieve from the structure of the problem. This guiding information is called the heuristics, which is how greedy algorithm choose a certain action to take or not to take. Choosing the right heuristic is central to the correctness and effectiveness of greedy algorithm.

Problems that can be successfully solved by the greedy algorithm typically admit special structure in the form of an optimal solution that is *composable*. For example, Kruskal's Algorithm for finding the minimum spanning tree of the graph exploits the fact that the smallest edge across any cut will always be part of some minimum spanning tree. Iteratively choosing the minimum cost edge without introducing a cycle is thus guaranteed the optimal solution. Not all problems admit this structure however. Last week, we discussed a greedy approximation for set cover. The following example shows that the greedy heuristic does not always find the optimal solution: whereas the optimal choice is to cover the following nine points with sets B, C, and D, the greedy algorithm will choose A and then be forced to select the rest of the sets.



Though greedy approaches may not optimally solve a problem, they can still provide approximations that are *provably* good, and even optimal under certain hardness assumptions such as P = NP. Moreover, greedy algorithms are often very simple to describe and efficient to implement. To understand both the power and limitations of the greedy algorithm, let us examine its application on a classical problem: k-center.

2.2 The k-center Problem

Let's say we have a large amount of data, some of them are similar, some are dissimilar. We want to group similar data together, possibly in a certain number of groups. We want to select some data points among all our data to be the cluster centers so that we can match each data to the nearest cluster center, hence find the clusters themselves.

This is a fairly reasonable problem to study because it tells some innate structure of our data, and how the data points relate to each other. It also arises in many real world situations: for example, Netflix users like different kinds of movies. A way of recommending similar moves may involve grouping customers that have similar movie interests together.

2.2.1 Problem Definition

In the k-center problem, we are given a set of vertices V, a distance function $d : V \times V \to \mathbb{R}_{\geq 0}$, and an integer $k > 0$ where the distance function induces a *metric* on elements of V. More precisely, $d(\cdot, \cdot)$ must satisfy the following properties.

(1) Positive semidefiniteness: $d(x, y) \geq 0$ for all $x, y \in V$ and $d(x, y) = 0$ if and only if $x = y$.

(2) Symmetric: $d(x, y) = d(y, x)$.

(3) Triangle inequality: $d(x, y) \leq d(x, z) + d(z, y)$.

Our objective is to find a set S of k vertices, called *cluster centers*, such that the maximum distance of any vertex to its cluster center is minimized where $i \in V$ is assigned to the cluster centered around the closest $s \in S$. We will write the distance of i to its center in short-hand as

$$d(i, S) = \min_{s \in S} d(i, s)$$

Given our cluster centers S, we will also denote the *radius* of S as the following:

$$r = \max_{i \in V} d(i, S)$$

We can then restate our objective as finding S that minimizes the radius of S. That is we find S subject to

$$\min_{S \subseteq V : |S| = k} \max_{i \in V} d(i, S)$$

Why ask only for the cluster centers, and not the clusters themselves? The objective function allows us to efficiently recover the clusters given the clusters via a simple scan: for each vertex i, iterate through k cluster centers to find the closest one, and place i in the partition belonging to the chosen center.

2.3 A Greedy Algorithm

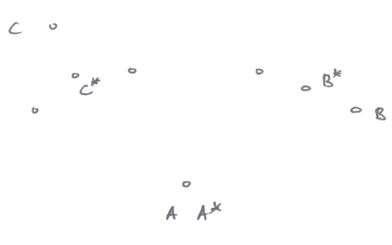
What will be a good heuristics for the k-centering problem? Intuitively, we want the centers we pick to be as spread out as possible, so that (1) all vertices in the graph possess a chance of having a nearby vertex as their centers, and (2) anomaly vertices that are far away from other vertices do not contribute too much to the maximum distance.

Our algorithm is thus; first pick a vertex $i \in V$ arbitrarily and put it in our set S of cluster centers. It then makes sense for the next cluster center to be as far away as possible from all the other cluster centers. While $|S| < k$, repeatedly find a vertex $j \in V$ for which the distance $d(j, S)$ is maximized (or in other words, which determines the diameter of set S). Add it to S. Once $|S| = k$, we stop and return S.

The Greedy Algorithm for k-Centering
Pick arbitrary $s \in V$ and initialize $S = \{s\}$. Do while $|S| < k$:

1. $s \leftarrow \text{argmax}_{s \in V}\, d(s, S)$
2. Update $S \leftarrow S \cup \{s\}$

It is certainly the case that this algorithm is not optimal. Consider the following example for $k = 3$ where the points are placed according to euclidean distance.



We see that cluster centers achieving minimal radius are given by A^*, B^*, and C^*, while, if A is chosen as the first cluster center, the greedy algorithm will choose A, B, and C.

2.3.1 Approximation Analysis

How good of an approximation does the greedy algorithm return? We can compare the greedy solution returned by the algorithm to an optimal solution. That is to say, we measure the effectiveness of this algorithm by bounding the approximation ratio.

Theorem 2.1. *The greedy algorithm produces a 2-approximation algorithm for the k-clustering problem.*

Proof. Let $S^* = \{s_1, \ldots, s_k\}$ denote the optimal solution, and let r^* denote its radius. This optimal solution partitions the nodes V into clusters $V_1^*, \ldots, V_k^*$, where each point $i \in V$ is placed in V_ℓ^* if it is closest to s_ℓ among all of the points in S^*. Ties are broken arbitrarily.

First, observe that each pair of points i and j in the same cluster V_ℓ^* are at most $2r^*$ apart. According to the triangle inequality, the distance $d(i,j)$ between them is at most the sum of $d(i,s_\ell)$, the distance from i to the center s_ℓ, and $d(j,s_\ell)$, the distance from the center s_ℓ to j. Both distances are bounded by r^* thus:

$$d(i,j) \leq d(i,s_\ell) + d(j,s_\ell) \leq r^* + r^* = 2r^*$$

Now consider the set $S \subseteq V$ of points selected by the greedy algorithm and consider the first iteration where the algorithm selects a point $i \in V_\ell^*$ to add to S, even though the algorithm had already selected a point $i' \in V_\ell^*$ in an earlier iteration. Prior to i' being added, each center added to S were chosen from separate optimal clusters of S^*. For all points j covered by centers added prior to i', we must have $d(j,S) \leq r^*$ by the argument above. Because the greedy algorithm always selects points furthest away from the current set of points in S, every other point $j \in V$ where j has not been added to S must have distance bounded by

$$d(j,S) \leq d(i',i)$$

otherwise, j should have been added to S prior to i'. However, i and i' belong to the same optimal cluster V_ℓ^* hence $d(i',i) \leq 2r^*$. Hence for all points covered after i' is added to the cluster centers, the distance between it and its nearest center is also bounded by at most $2r^*$. We have for all points $i \in V$, the distance is bounded by $d(i,S) \leq 2r^*$. If r represents the radius of S returned by our greedy solution, we can conclude that:

$$r^* \leq r \leq 2 \cdot r^*$$

implying the algorithm returns a 2-approximation as required. □

2.3.2 Hardness of Approximation

The problem of finding optimal S for an instance of k-center is **NP**-hard hence why we seeked an approximation algorithm to begin with. However, we can ask an equally important question: how hard is it to approximate k-center within some approximation ratio? Admittedly, the greedy algorithm for k-center is not terribly difficult to explain nor implement. However, can there be an efficient algorithm that provides a better approximation? We will now show that this mission is impossible, unless $\mathbf{P} = \mathbf{NP}$.

Theorem 2.2. *There is no α-approximation algorithm for the k-center problem for $\alpha < 2$ unless $\mathbf{P} = \mathbf{NP}$.*

In general, how do we prove that a problem is "hard"? One common way is to reduce another hard problem to it. Recall that, given two problems A and B and a black-box solver B, if we can produce a procedure that transforms an input of A to B, then we say that A reduces to B. Critically, this procedure must run in *polynomial time*, allowing us to conclude that, if there is an efficient algorithm for B, then there must be an efficient algorithm for A. In that sense, we think of problem B to be *at least as hard as* A. For k-center, if an **NP**-hard problem can be reduced to producing an α-approximation for k-center problem where $\alpha < 2$, then finding an α-approximation to k-center problem with $\alpha < 2$ will be **NP**-complete.

Consider the **NP**-hard problem of finding a dominating set. In the dominating set problem, we are given a graph $G = (V,E)$ and an integer k, and we must decide if there exists a set $S \subset V$ of size k such that each vertex is either in S, or adjacent to a vertex in S. We will now prove by reducing dominating set to finding an α-approximate k-center problem for $\alpha < 2$.

Proof. Let us illustrate how to reduce dominating set to k-center. Given an instance of dominating set, we can define an instance of k-center by setting the distance between adjacent vertices to 1, and non-adjacent vertices to 2. These distances satisfy properties of a metric and, in particular, the triangle inequality as for any three vertices $x, y, z \in V$, for any choice of $d(x, y)$, $d(y, z)$, and $d(x, z)$ equal to 1 or 2, we satisfy

$$d(x, z) \leq d(x, y) + d(y, z)$$

Observe there is a dominating set of size k if and only if the optimal radius for this k-center instance is 1. If there is a dominating set D of size k, then choosing the cluster centers $S = D$ will achieve radius 1 as every vertex is adjacent to a vertex in D and hence be distance 1 away from a cluster center. If there is a choice of k cluster centers S with radius 1, then by construction of the metric, each vertex v is adjacent to $s \in S$ meaning S is a dominating set.

The objective for k-center is to minimize the maximum distance of a vertex to its cluster center. The radius cluster centers found from our reduction can only be 2 or 1. Now suppose we have an algorithm that returns an α-approximation for k-center. If r^* denotes the optimal cluster radius and r the radius of the approximate cluster centers, the algorithm guarantees:

$$r^* \leq r \leq \alpha \cdot r^*$$

If there is a dominating set in G with k vertices, then the radius of the optimal cluster centers will be $r^* = 1$. The approximation algorithm will guarantee:

$$1 \leq r \leq \alpha$$

If there is not a dominating set, then the optimal radius will be $r^* = 2$ and the algorithm guarantees:

$$2 \leq r \leq \alpha \cdot 2$$

For α such that these two ranges are non-overlapping, then the α-approximation for k-center will correctly distinguish whether or not G has a dominating set of size k. These two ranges are non-overlapping when $\alpha < 2$, which means if there is an α-approximation for k-center, then there exists a polynomial time algorithm for dominating set. We conclude that unless $\mathbf{P} = \mathbf{NP}$, there is no polynomial time α-approximation for k-center where $\alpha < 2$. □

2.4 Conclusion

Greedy approaches often tend to be simple to describe and implement, but this should not be discounted as we demonstrated, perhaps surprisingly, that our algorithm for k-center achieves an optimal approximation ratio of 2 unless $\mathbf{P} = \mathbf{NP}$. We will continue our study of greedy approximation algorithms by discussing yet another famous **NP**-hard problem next – traveling salesman.

References

[1] Gonzalez, T. F. (1985). "Clustering to minimize the maximum intercluster distance." In *Theoretical Computer Science*, 38, 293-306.