Khoury College of Computer Sciences Fall 2023
Northeastern University 8 September 2023
CS 7870: Seminar in Theoretical Computer Science Scribe: Rajmohan Rajaraman

Lecture Outline:

- Roadmap of the course
- Clustering: the k-center problem
- Nets in metric spaces and nearest neighbor search

1 Roadmap of the course

This course presents a modern toolkit for algorithms and their analysis. The algorithmic techniques we plan to cover in this class include greedy algorithms, local search, linear and convex programming, algebraic methods, and variants of gradient descent. For analyzing algorithms, we will use classic worst-case methods such as approximation ratios, competitive analysis, and regret bounds, as well as newer ”beyond worst-case analysis” methods including perturbation stability, smoothed analysis, and tools for analyzing ML-augmented algorithms. Here is a high-level outline of the course units and topics.

Clustering

- k-center, k-median, mixture of Gaussians, spectral clustering
- Greedy algorithms, linear programming, local search
- Approximation algorithms, perturbation stability, high-dimensional spaces

Linear programming

- Applications and algorithms
- Geometry and duality

Dimensionality reduction

- Clustering mixture of Gaussians, low-rank approximations and sparse recovery
- Principal Component Analysis, Singular Value Decomposition

Smoothed analysis

- Simplex algorithm and local search

- Equivalence with pseudo-polynomial time complexity

Optimization

- Gradient descent and stochastic gradient descent
- Multiplicative weights method and Follow-the-Regularized-Leader

Online algorithms

- Online learning and convex optimization
- Competitive analysis
- Algorithms with ML predictions

2 Clustering

We will begin the course with a topic that is well-understood and has wide applications: clustering. The clustering problem is that of partitioning a given set of data points into clusters in which each cluster represents a collection of very similar points, while two points in different clusters are expected to be dissimilar. Clustering is a canonical technique for unsupervised learning, and has many applications. The material in this section and the next is partly based on the lecture notes of Chandra Chekuri [2, Section 9.1].

While the notion of clustering is intuitive and it is often easy for humans to visually evaluate the quality of a clustering, there is no single optimization objectiveness that captures the goodness of clustering. There are many compelling algorithms used for clustering in practice, and there are many mathematical problem formulations. The twain do not always meet. We will use clustering as a vehicle to introduce interesting algorithmic design and analysis techniques, with a motivation to bridge the gap between theory and practice.

Definition 1. The input to the Clustering problem is a set V of n points, with a distance function $d : V \times V \to \mathbb{R}$ and a non-negative integer $k \leq n$. The goal is to partition V into a collection $\mathcal{C}$ of k disjoint subsets of V, which are called *clusters*. Different clustering problems can be defined by specifying different measures of goodness of $\mathcal{C}$.

We will be primarily interested in clustering problems where the underlying distance function forms a metric. We refer to the pair (V, d) as a metric space.

Definition 2. A metric space (V, d) is a set V of points and a distance function $d : V \times V \to \mathbb{R}$ satisfying the following properties:

- (Non-negativity) $d(u, v) \geq 0$ for all u, v and $d(u, v) = 0$ if and only if $u = v$.
- (Symmetry) $d(u, v) = d(v, u)$ for all $u, v \in V$.
- (Triangle Inequality) $d(u, w) \geq d(u, v) + d(v, w)$ for all $u, v, w \in V$.

In center-based clustering, we measure the goodness of a clustering $\mathcal{C}$ by associating a center c_i with each cluster C_i of $\mathcal{C}$, and set the cost of $\mathcal{C}$ to be

$$\sum_{C_i \in \mathcal{C}} \sum_{v \in C_i} d(v, c_i)^p,$$

for some non-negative p. An alternative equivalent formulation of center-based clustering is to seek a set $X = \{c_1, \ldots, c_k\}$ of k centers so as to minimize

$$\sum_{v \in V} d(v, X)^p,$$

where for any set S, we use $d(v, S)$ to denote $\min_{s \in S} d(v, s)$ Cluster C_i associated with center c_i consists of all points for which c_i is the closest center, breaking ties arbitrarily.

Three popular choices for p yield three commonly studied center-based clustering problems.

- The *k-center problem* is obtained when p is set to ∞, where the objective can be revised as minimizing

$$\max_{v \in V} d(v, X)$$

- The *k-median problem* is obtained when p is set to 1.
- The *k-means problem* is obtained when p is set to 2.

3 The k-center problem

We consider the k-center problem when the distance function d forms a metric space; that is, d is non-negative, symmetric, and satisfies the triangle inequality. We observe below that the k-center problem in metric spaces is NP-hard. A natural approach to overcoming such evidence of intractability is to allow for approximations. The *approximation ratio* of an algorithm $\mathcal{A}$ for a given minimization problem Π is

$$\max_{\text{instance } I \text{ of } \Pi} \frac{\text{cost of } \mathcal{A} \text{ for } I}{\text{optimal cost for } I}$$

There is a rich theory of approximation algorithms, which classifies NP-complete problems according to the approximation ratios that are achievable in polynomial time, subject to conjectures in complexity theory. For more on approximation algorithms, please refer to the excellent texts by Williamson and Shmoys [7] and Vazirani [6]. For instance, the k-center problem is covered in [7, Section].

We now present the NP-hardness of k-center in metric spaces. In fact, we will show that it is NP-hard to even compute a c-approximation to k-center, for any $c < 2$.

Lemma 1. There is no polynomial-time c-approximation algorithm for metric k-center, with $c < 2$, unless P = NP.

Proof. We present a reduction from the decision version of the *minimum dominating set problem*. In the decision version of the minimum dominating set problem, we are given an undirected graph G and an integer k, and the goal is to determine if there is a subset S of k vertices in G such that every vertex is in S or adjacent to some vertex in S.

We construct the following k-center instance. We have a point for each vertex in G. The distance between any two points is given by the length of the shortest path between the two in G. We now argue that there is a dominating set of size k in G if and only if there is a k-center solution with cost strictly less than 2. If there is a dominating set of size k, then there is a solution to the k-center problem with cost 1. Any solution to the k-center problem that is better than 2-approximation has cost strictly less than 2; since all distances are integral, the cost is, in fact, at most 1. Hence, every vertex is either one of the centers or is adjacent to one of the centers. This implies the existence of a dominating set of size k.

It follows that any polynomial-time algorithm for k-center with approximation ratio strictly less than 2 can be used to solve the minimum dominating set in polynomial time. Since the minimum dominating set problem is NP-complete, the desired claim follows. $\square$

The following lemma lower bounding the optimal cost is key to establish the performance guarantee for approximation algorithms for k-center.

Lemma 2. If S is a set of $k + 1$ points such that the distance between any pair of points in S is at least $2R$, then the optimal cost is at least R.

Proof. The proof is by contradiction. Suppose the optimal k-median cost is less than R, yet there exists a set S of $k + 1$ points such that the distance between any pair of points in S is at least $2R$. Let C be the set of k centers of an optimal k-median solution. By the pigeon-hole principle, there exist two points in S that belong to the same cluster, say with center c. By the bound on optimal k-median cost, the distance between the two points is less than $2R$, a contradiction to our assumption. $\square$

3.1 Gonzalez's algorithm

A simple approach to attacking k-center is a greedy algorithm due to Gonzalez [3]. Start with an arbitrary center, and repeatedly select the next center to be the point farthest from the current centers.

Theorem 1. In any metric space, Gonzalez's algorithm yields a 2-approximation for the k-center problem.

Proof. Let C be the set of centers determined by Gonzalez's algorithm. Let R denote the cost of the solution. Let $c_1, c_2, \ldots, c_k$ denote the k centers, and let v denote a point that is farthest from C. By definition, v is at least R from each of the centers. Furthermore, the distance between c_i and $\{c_1, \ldots, c_{i-1}\}$ is at least R; otherwise c_i would not have been chosen as the ith center since v would have been farther from $\{c_1, \ldots, c_{i-1}\}$ than c_i. Thus, we have a set of $k + 1$ points such that the distance between any pair is at least R. By Lemma 2, it follows that the optimal k-center cost is at least $R/2$, implying a 2-approximation for Gonzalez's algorithm. $\square$

Remark 1. A useful feature of Gonzalez's algorithm is that it builds the k-center solution incrementally. Thus, if we run the algorithm for n iterations, then for $1 \leq k \leq n$ the partial solution after k iterations is a 2-approximate solution to the k-center problem. This means that for relevant applications we can build a good k-center solution in a dynamic manner, adding a new center whenever we have enough resources to open the next one.

3.2 Hochbaum-Shmoys's algorithm

Another greedy algorithm for k-center, due to Hochbaum and Shmoys [4] is one which first guesses the cost R^* of the optimal solution, selects an arbitrary point as the first center, and then repeatedly finds a center, if possible, that is more than $2R^*$ away from the current set of centers. (If no such center exists, the iteration terminates.) The algorithm runs for k iterations and returns the set of selected centers if there is no point at distance more than $2R^*$ away from the selected centers.

Before analyzing the approximation ratio of the algorithm, let us analyze its running time. A naive implementation of iteration i processes $(i-1)(n-i+1)$ distances; the total running time can be bounded by $O(k^2n)$. What about guessing R^*? Well, there are at most $\binom{n}{2}$ possible values for R^*, so one can conduct a binary search over these values and select the smallest value of R^* for which the algorithm returns a set of centers.

Theorem 2. In any metric space, the Hochbaum-Shmoys algorithm computes a 2-approximation to k-center.

Proof. Let R^* denote the optimal cost for the k-center instance. We argue that after selection of k centers, the distance of any point v from its closest center is at most $2R^*$. Let C be the set of k centers. By the design of the algorithm, the distance between any two centers in C is greater than $2R^*$. Suppose, for the sake of contradiction, there exists a point v that is at distance greater than $2R^*$ from C. Then, it follows that $C \cup \{v\}$ is a set of $k+1$ points such that the distance between every pair of points is greater than $2R^*$. By Lemma 2, it follows that the optimal k-center cost is greater than R^*, yielding a contradiction. $\square$

4 Nets in metric spaces

The k-center algorithms presented above compute a set of well-separated centers such that every point is within a certain distance from its closest center. Such a collection of centers in a metric space is referred to as a net, and has been extensively used in solving distance-related problems. We formally present the concept of nets in metric spaces and discuss its application in nearest neighbor search. The material for this section is partly drawn from notes due to Sanjeev Arora [1].

Formally, we define an ε-net of a metric space as follows.

Definition 3. An ε-net of a metric space (X, d) is a subset S of X satisfying the following properties:

- For any $u, v \in S$, we have $d(u, v) \geq \varepsilon$.
- For any $x \in X$, we have $d(x, S) < \varepsilon$.

It is straightforward to construct an ε-net for any finite metric space: Start with $S = \{s\}$ for an arbitrary point $s \in X$; repeatedly add x to S where x is an arbitrary point satisfying $d(x, S) \geq \varepsilon$, until there is no such x; return S. Clearly, the procedure terminates since every iteration adds a point to S and X is finite. By construction, any point added to S is at least ε away from all points to S, satisfying the first property. The second property holds at termination since there is no point x remaining that has $d(x, S) \geq \varepsilon$. The above construction also extends to infinite metric spaces (assuming Zorn's Lemma). For $\mathbb{R}$, an ε-net is any set of the form $\{x + n\varepsilon : n \in Z\}$.

Definition 4. The doubling dimension of a metric space (X, d) is the smallest integer k such that every subset $S \subseteq X$ can be covered by at most 2^k sets of diameter at most half the diameter of S.

It is easy to see that the doubling dimension of $\mathbb{R}^d$ is d.

Lemma 3. For any metric space with diameter D and doubling dimension m, the size of an ε-net is $\left(\frac{2D}{\varepsilon}\right)^m$.

Proof. We prove by induction on i that for any metric space with diameter $2^i\varepsilon$ and doubling dimension m can be covered by at most 2^{im} sets, each of diameter at most ε. The induction base, $i = 0$, is immediate. For the induction step, suppose the claim holds for $i = k$. Any metric space with diameter $2^{k+1}\varepsilon$ and doubling dimension m can be covered by at most 2^m sets, each of diameter at most $2^k\varepsilon$. Each of these sets is a metric space of diameter $2^k\varepsilon$ and can be covered by at most 2^{km} sets of diameter at most ε. Therefore, the metric space of diameter $2^{k+1}\varepsilon$ can be covered by at most $2^{(k+1)m}$ sets of diameter at most ε, completing the induction step. Therefore, any metric space with diameter D and doubling dimension m can be covered by at most $2^{(1+\log_2(D/\varepsilon))m}$ sets of diameter at most ε.

Any ε-net of the metric space has at most one point from a set of diameter at most ε. Thus, any ε-net of a metric space of diameter D and doubling dimension m has size at most $\left(\frac{2D}{\varepsilon}\right)^m$. □

5 Application of nets to nearest neighbor search

The nearest neighbor problem is the following: given a metric space (X, d), a subset $S \subseteq X$, and a point $x \in X$, return the point y in S that minimizes $d(x, y)$. The nearest neighbor problem has many applications, the most notable one being its use in machine learning. A common approach to classification is to associate a metric space among data items, and classifying each new data item by the label of the nearest neighbor in the training. Variants of nearest neighbor classification include using an average, distance-weighted average, or majority label of k nearest neighbors.

An easy solution to the nearest neighbor problem takes time $O(|S|)$ by going over all the distances from S to the new point x. But this is excessive in situations where S and the number of new points to be classified are both large. We now consider an alternative approach, using ε-nets, of calculating approximate nearest neighbors in $O(\log D)$ time, for metric spaces with diameter D and constant doubling dimension. This is due to Krauthgamer and Lee [5]. For simplicity, in this lecture, we will present a 3-approximation algorithm; using some more sophistication, we can compute a $(1 + \varepsilon)$-approximation.

The idea for the algorithm is similar to the concept of quad-trees extensively used in databases and data structures. We maintain ε-nets of different "scales". We assume, without loss of generality, that the minimum distance in the metric is 1 and the diameter of the metric is D. For simplicity, we assume that D is a power of 2. Let Y_i denote a 2^i-net, for each $1 \leq i \leq k$, where $k = \log_2 D$. Note that Y_k is a singleton set. Let q denote the query point.

The nearest neighbor algorithm maintains for each $1 \le i \le k$, the 2^i-net Y_i, and for each point $y \in Y_i$, a set $L_{y,i}$ of nearby points from Y_{i-1}, defined as follows:

$$L_{y,i} = \{z \in Y_{i-1} : d(y, z) \le c2^i\},$$

for a constant c that we will set shortly.

Given a query q, the algorithm proceeds as follows.

1. Set y to be the unique point in Y_k, and $i = k$.
2. While $d(q, L_{y,i}) \le 3 \cdot 2^{i-1}$: set y to be the point in $L_{i,y}$ nearest to q and $i = i - 1$.
3. Return y

Theorem 3. The above algorithm returns a 3-approximation to the nearest neighbor problem.

Proof. Let y denote the point returned by the algorithm and let i be such that $d(q, L_{y,i}) > 3 \cdot 2^{i-1}$. By the definition of the algorithm, we have $d(q, y) \le 3 \cdot 2^i$.

Let v denote the nearest neighbor for q. Let p denote the point in Y_{i-1} closest to v. Then, we have $d(p, v) \le 2^{i-1}$. We argue that p is in $L_{y,i}$. We have $d(p, y) \le d(p, v) + d(v, q) + d(q, y) \le 2^{i-1} + 2d(q, y) \le 6.5 \cdot 2^i$. If we set c to be at least 6.5, we have p in $L_{y,i}$.

We now show that $d(q, v) \ge 2^i$. We have $d(q, v) \ge d(q, p) - d(p, v) \ge 3 \cdot 2^{i-1} - 2^{i-1} = 2^i$. This completes the proof of the theorem. $\square$

We now analyze the running time of the above approximation algorithm for nearest neighbor search. Note that the nets Y_is and the corresponding sets $L_{y,i}$ are all pre-computed. Given a query q, the total running time is at most k times the size of each $L_{y,i}$.

We now place an upper bound on $|L_{y,i}|$ for any y and i. Each point in $L_{y,i}$ is within $6.5 \cdot 2^i$ of y, therefore the diameter of $L_{y,i}$ is at most $13 \cdot 2^i$. If the doubling dimension of the metric space is m, then L_{y_i} can be covered by at most $2^{m\lceil \log_2(26) \rceil} = 2^{5m}$ sets of diameter at most 2^{i-2}. Since the points in $L_{y,i}$ are separated from one another by at least 2^{i-1}, at most one of these points can occupy any set of diameter at most 2^{i-2}. Therefore, $|L_{y,i}|$ is at most 2^{5m}, thus bounding the time per query to be at most $2^{5m} \log(\Delta)$. For constant m, this time is, in fact, independent of the size of S and only logarithmic in the diameter of the metric space. As mentioned above, the algorithm can be refined to also achieve an $(1 + \varepsilon)$ approximation to the nearest neighbor problem, for arbitrary $\varepsilon > 0$ by choosing nets at more scales and storing distance thresholds in the algorithm more carefully.

References

[1] Sanjeev Arora. Two notions of ε-nets and applications. `https://www.cs.princeton.edu/courses/archive/fall07/cos597D/Site/lec12.pdf`.

[2] Chandra Chekuri. Approximation algorithms. `https://courses.engr.illinois.edu/cs583/fa2021/approx-algorithms-lecture-notes.pdf`.

[3] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[4] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the *k*-center problem. *Math. Oper. Res.*, 10(2):180–184, 1985.

[5] Robert Krauthgamer and James R. Lee. Navigating nets: simple algorithms for proximity search. In J. Ian Munro, editor, *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 798–807. SIAM, 2004.

[6] Vijay V. Vazirani. *Approximation algorithms.* Springer, 2001.

[7] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.