CS 395T: Sublinear Algorithms Fall 2016

Lecture 10 — September 27, 2016

Prof. Eric Price *Scribes: Quinten McNamara & William Hoza*

1 Overview

In this lecture, we focus on constructing *coresets*, which are concise representations of a larger dataset. We will be particularly interested in coresets for datasets in $\mathbb{R}^2$; as an application, we will see a streaming algorithm for the k-median problem in $\mathbb{R}^2$ in the insertion-only model.

2 The k-median problem

The k-median problem can be thought of in terms of the facility location problem: given a set of consumers that need to be provided service, open a set of k stores that minimizes the cost (total distance traveled by the consumers.)

Problem definition Let Δ be a positive integer. Given a set of points $P = (p_1, \ldots, p_n) \in ([\Delta]^2)^n$, find centers $C = (c_1, \ldots, c_k) \in ([\Delta]^2)^k$ that minimize

$$d(P, C) := \sum_{i \in [n]} \min_{j \in [k]} \|p_i - c_j\|.$$

Here, $\|\cdot\|$ could denote the 1-norm or the 2-norm.

Offline complexity of the k-median problem Let C^* be the minimizer, and let $\mathsf{OPT}(P, k) = d(P, C^*)$. The bad news is that it is NP-hard to exactly compute C^*. This means we should aim for a $(1+\epsilon)$-approximation algorithm, i.e. we should try to construct C such that

$$d(P, C) \leq (1+\epsilon) \cdot \mathsf{OPT}(P, k).$$

The good news is that in 1999, Kolliopoulos and Rao [KR99] provided a $(1+\epsilon)$-approximation algorithm for the k-median problem in $\mathbb{R}^2$ with runtime $O((1/\epsilon)^{O(1/\epsilon)} n \log n \log k)$. The runtime of the [KR99] algorithm depends badly on ϵ, but for large values (say $\epsilon = 1/2$) the runtime is reasonable. There's nothing particularly "sublinear" about the [KR99] algorithm, so we won't go through it in class. But we will use it as a black box.

2.1 Comparison to k-means

The better-known *k-means problem* is very similar to the k-median problem; the difference is that we minimize

$$\sum_{i \in [n]} \min_{j \in [k]} \|p_i - c_j\|_2^2.$$

To see why the problems are called "k-median" and "k-means", observe that we can break each problem into two subproblems:

1. Choose a partition $P = P^{(1)} \cup \cdots \cup P^{(k)}$.
2. Choose k centers $C = (c_1, \ldots, c_k)$.

The goal is to minimize $\sum_{j\in[k]} \sum_i \|P_i^{(j)} - c_j\|$ in the k-median problem or $\sum_{j\in[k]} \sum_i \|P_i - c_j\|_2^2$ in the k-means problem.

Let's suppose that we've already chosen a partition. Then for each part, we need to choose a center. One can show that for the k-means problem, the best choice of c_j is the mean of the points in $P^{(j)}$. For the k-median problem, if $\|\cdot\|$ is the 1-norm, then the best choice of c_j is the coordinate-wise median of the points in $P^{(j)}$. This is easiest to see in one dimension: given $p_1, \ldots, p_n \in \mathbb{R}$, the c minimizing $\sum_i (p_i - c)^2$ is the mean of the p_is, and the c minimizing $\sum_i |p_i - c|$ is the median of the p_is.

3 Coresets

If $S = (s_1, \ldots, s_m) \in ([\Delta]^2)^m$ and $W = (w_1, \ldots, w_m) \in \mathbb{N}^m$, we will think of (S, W) as a *weighted* list of points, i.e. w_j is the weight of s_j.

Definition 1 (Coreset)**.** *A (k, ϵ)-*coreset *for $P = (p_1, \ldots, p_n)$ is a weighted list of points (S, W) such that there exists $\pi : [n] \to [m]$ such that $w_i = |\pi^{-1}(i)|$, and $d(P, S) \le \epsilon \cdot \mathsf{OPT}(P, k)$.*

Observe that without loss of generality, $\pi(i)$ gives the index of the closest s_i to p_i, in which case

$$d(P, S) = \sum_{i=1}^{n} \|p_i - s_{\pi(i)}\|.$$

Observe also that for $\epsilon < 1$, we certainly will need $m > k$, by the definition of $\mathsf{OPT}(P, k)$.

3.1 Reducing the k-median problem to coreset construction

Given a (k, ϵ)-coreset (S, W) for P, we can construct an approximate solution to the k-median problem for P via the following algorithm: Choose $\tilde{C}$ which minimizes

$$d((S, W), \tilde{C}) := \sum_{i\in[m]} w_i \cdot \min_{j\in[k]} \|s_i - c_j\|.$$

Claim 2. *$\tilde{C}$ is a $(1 + 2\epsilon)$-approximate solution to the k-median problem for $\mathbf{p}$.*

Proof. For any $C = (c_1, \ldots, c_k)$,

$$\begin{aligned} d(P, C) &= \sum_{i=1}^{n} \min_{1 \le j \le k} \|p_i - c_j\| \\ &= \sum_{i=1}^{n} \min_{1 \le j \le k} (\|s_{\pi(i)} - c_j\| \pm \|p_i - s_{\pi(i)}\|) \quad \text{by the triangle inequality} \\ &= d((S, W), C) \pm d(P, S) \\ &= d((S, W), C) \pm \epsilon \cdot \mathsf{OPT}(P, k). \end{aligned}$$

By applying this fact first with $C = \tilde{C}$ and then with $C = C^*$, we find that

$$\begin{aligned} d(P, \tilde{C}) &\le d((S, W), \tilde{C}) + \epsilon \cdot \mathsf{OPT}(P, k) \\ &\le d((S, W), C^*) + \epsilon \cdot \mathsf{OPT}(P, k) \quad \text{by the definition of } \tilde{\mathbf{c}} \\ &\le d(P, C^*) + 2\epsilon \cdot \mathsf{OPT}(P, k) \\ &= (1 + 2\epsilon) \cdot \mathsf{OPT}(P, k). \end{aligned}$$

□

Of course, this reduction is not efficient yet, because finding $\tilde{C}$ is just another instance of the k-median problem. But a similar analysis shows that if $\tilde{C}$ merely $(1+\epsilon)$-approximately minimizes $d((S, W), \tilde{C})$, it will still be a $(1 + O(\epsilon))$-approximate solution solution to the k-median problem for P. And as discussed, such a $\tilde{C}$ can be constructed reasonably efficiently using the algorithm from [KR99].

4 Constructing coresets

4.1 The offline setting

Let's suppose all the data is available to us.

$(1, \epsilon)$-coreset in one dimension As a warm-up, how can we construct a $(1, \epsilon)$-coreset in one dimension? In other words, we have numbers $p_1, \ldots, p_n \in \mathbb{R}$ with median c^*, and we would like to find a small number of points $s_1, \ldots, s_m$ and a map $\pi : [n] \to [m]$ so that if we move each p_i to $s_{\pi(i)}$, then the total distance all the points traveled is small compared to the total distance all the points would travel if we moved them all to c^*.

Construction:

$$S = \{c^*\} \cup \{c^* \pm (1 + \epsilon)^i : i \in \{0, 1, \ldots, \log_{1+\epsilon} \Delta\}\}.$$

Proof of correctness: Without loss of generality, consider p_i between $c^* + (1+\epsilon)^i$ and $c^* + (1+\epsilon)^{i+1}$. The distance moved by p_i is at most $|(1 + \epsilon)^{i+1} - (1 + \epsilon)^i|$, which is $\epsilon \cdot (1 + \epsilon)^i$, which is at most $\epsilon \cdot |p_i - c^*|$. It follows that

$$d(P, (S, W)) \le \epsilon \cdot \mathsf{OPT}(P, k).$$

The number of points m is $O(\frac{1}{\epsilon} \log \Delta)$.

(k, ϵ)-coreset in one dimension We can generalize the construction to a (k, ϵ)-coreset easily enough: Let $C^* = (c_1^*, \ldots, c_k^*)$ be the optimal solution to the k-median problem for P. Repeat the $k = 1$ construction for each c_j^* and overlay the coresets. The total number of points m will be $O(\frac{k}{\epsilon} \log \Delta)$.

$(1, \epsilon)$-coreset in two dimensions We can generalize the construction to a $(1, \epsilon)$-coreset for a two-dimensional dataset as follows. Let c^* be the solution to the 1-median problem for P. Consider concentric disks $D_1, D_2, \ldots, D_{\log \Delta}$ centered at c^*, where disk D_j has radius 2^j. Cover each D_j with disks of radius $2^{j-1}\epsilon$. Our coreset consists of the centers of these covering disks. To prove correctness, observe that if j is the smallest value such that $p_i \in D_j$, then $\|p_i - c^*\| \geq 2^{j-1}$, and hence the distance traveled by p_i is at most $2^{j-1}\epsilon \leq \epsilon \cdot \|p_i - c^*\|$. How many disks does it take to cover each D_j? Area considerations make it clear that we will need at least $\Omega(1/\epsilon^2)$ disks, and it turns out that $O(\epsilon^2)$ disks suffice. (This can be shown by considering a simple lattice of disks.) Therefore, the number of points in our coreset m is $O(\frac{1}{\epsilon^2} \log \Delta)$.

Two dimensions, arbitrary k Once again, generalizing to arbitrary k is straightforward (just repeat the construction for each center and overlay the coresets.) The number of points is $O(\frac{k}{\epsilon^2} \log \Delta)$.

Computational efficiency Unfortunately, our constructions have all relied on knowing the optimal k-median centers. What happens if we only have a $(1 + \epsilon)$-approximation, e.g. from [KR99]? Given C, we've shown how to efficient construct a weighted list (S, W) of $m \leq O(\frac{k}{\epsilon^2} \log \Delta)$ points such that $d(P, S) \leq \epsilon \cdot d(P, C)$. So as long as C is a 2-approximation to the k-median of P, then (S, W) will be a $(k, 2\epsilon)$ coreset for P.

4.2 The streaming setting: Merge and reduce

We now present a streaming approximation algorithm for constructing a (k, ϵ)-coreset. The general approach is to construct coresets of small batches of the data as they come through the stream, and then aggregate the coresets to create a coreset for the entire data.

To be more precise, we imagine a *binary forest* of coresets. The leaves of the forest are batches of a few input points. Each node is the coreset of its children. Whenever two trees appear with the same height, a new root node is added to merge them into one tree. The algorithm only actually has to store a single coreset at each level of this forest. This paradigm is called the *merge and reduce* paradigm.

Here is an explicit description of the algorithm. The algorithm maintains a list of coresets (S_j, W_j), all initially undefined.

1. Read the input stream $P = (p_1, \ldots, p_n)$ a few points a time. For each batch of a few points:
   (a) Condense the points into a $(k, \epsilon/\log n)$-coreset (S, W).
   (b) Initialize $j = 1$. While (S_j, W_j) is defined:
      i. Set (S, W) to be a $(k, \epsilon/\log n)$-coreset of $(S, W) \cup (S_j, W_j)$.
      ii. Delete (S_j, W_j) so it is no longer defined.

iii. Increment j.

(c) Define $(S_j, W_j) := (S, W)$.

2. Output a $(k, \epsilon/\log n)$-coreset of all currently defined (S_j, W_j).

Efficiency For simplicity, we can take the batch size to be, say, 1 point. Then the space used by the algorithm is only $O(\frac{k}{\epsilon^2}\log\Delta\log^3 n)$, since each coreset requires $O(\frac{k}{(\epsilon/\log n)^2}\log\Delta)$ space. (In practice, it would be beneficial to have a larger batch size – after all, we might as well take $\frac{k}{\epsilon^2}\log\Delta\log^3 n$ points at a time.)

Correctness We need to bound the error accumulation:

Lemma 3. *Assume $\epsilon \le 1/2$. Suppose (S_1, W_1) is a (k,ϵ)-coreset for P_1 and (S_2, W_2) is a (k,ϵ)-coresets for P_2. Suppose $(\bar{S}, \bar{W})$ is a (k,ϵ')-coreset for $(S_1, W_1) \cup (S_2, W_2)$. Then $(\bar{S}, \bar{W})$ is a $(k, 2\epsilon' + \epsilon)$-coreset for $P_1 \cup P_2$.*

Proof. We need to bound the distance traveled when we move from $P_1 \cup P_2$ to $(\bar{S}, \bar{W})$. By the triangle inequality, this is at most the distance traveled when we move to $(S_1, W_1) \cup (S_2, W_2)$ plus the distance traveled when we move from $(S_1, W_1) \cup (S_2, W_2)$ to $(\bar{S}, \bar{W})$:

$$\begin{aligned} d(P_1 \cup P_2, \bar{S}) &\le \epsilon \cdot \mathsf{OPT}(P_1 \cup P_2, k) + d((\bar{S}, \bar{W}), (S_1, W_1) \cup (S_2, W_2)) \\ &\le \epsilon \cdot \mathsf{OPT}(P_1 \cup P_2, k) + \epsilon' \cdot \mathsf{OPT}((S_1, W_1) \cup (S_2, W_2), k). \end{aligned}$$

Recall that earlier, when reducing the k-median problem to that of constructing coresets, we showed that

$$\mathsf{OPT}((S_1, W_1) \cup (S_2, W_2), k) \le (1 + 2\epsilon) \cdot \mathsf{OPT}(P_1 \cup P_2, k).$$

Therefore,

$$\begin{aligned} d(P_1 \cup P_2, \bar{S}) &\le (\epsilon + \epsilon' \cdot (1 + 2\epsilon)) \cdot \mathsf{OPT}(P_1 \cup P_2, k) \\ &\le (\epsilon + 2\epsilon') \cdot \mathsf{OPT}(P_1 \cup P_2, k). \end{aligned}$$

□

Based on this lemma, one can show by induction that the coresets maintained by the algorithm are all (k,ϵ)-coresets, completing the proof of correctness.

References

[KR99] S. G. Kolliopoulos and S. Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem. In *European Symposium on Algorithms*, 1643:378–389, 1999.