CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 17 (Nov 4, 2019): i-Sample and Coresets

Lecturer: Mohammad R. Salavatipour *Scribe: Brandon Fuller*

17.1 Selection

In this section we will finish the discussion of Munro-Patterson algorithm for selection. Recall that the algorithm would have multiple passes. In each pass over the data, it would reduce the size of the problem to an instance over $O(n \log^2 n/s)$ items using a buffer of size s, this was done by taking an i-sample in pass i with two lower/upper bound filters a_i, b_i.

Lemma 1 *Suppose $x_1 < \cdots < x_s$ is an i-sample of a population P of size $2^i s$. Then for each j, $2^i j \leq rank(x_j, P) \leq 2^i(i + j)$.*

Proof. Let L_{ij} and M_{ij} be the least/most bounds for $rank(x_j, P)$ in the ith sample of our process. Using induction on i, it is clear that when $i = 0$, $rank(x_j, P) = j$ and thus the property holds (since the i-samples are sorted). Now suppose $i > 0$ and notice that the $x_1, \ldots, x_s$ were selected from two $i-1$-samples, say $y_1, \ldots, y_s$ and $z_1, \ldots, z_s$. We notice that in our i-sample, if p elements less than x_j come from the list $y_1, \ldots, y_s$ then $j-p$ elements less than x_j come from the list $z_1, \ldots, z_s$. Thus, since the i-sample takes even elements from each list we get the following (using the induction hypothesis):

$$L_{ij} = \min_p\{L_{i-1,2p} + L_{i-1,2(j-p)}\} = \min_p\{2^{i-1}2p + 2^{i-1}2(j-p)\} = \min_p\{2^i p + 2^i j - 2^i p\} = 2^i j$$

and using a similar argument for M_{ij} we get:

$$\begin{aligned} M_{ij} &= \max_p\{M_{i-1,2p} + M_{i-1,2(j-p+1)}\} \\ &= \max_p\{2^{i-1}(i-1+2p) + 2^{i-1}(i-1+2(j-p+1))\} \\ &= \max_p\{2^{i-1}i - 2^{i-1} + 2^i p + 2^{i-1}i - 2^{i-1} + 2^i j - 2^i p + 2^i\} \\ &= 2^i i + 2^i j \\ &= 2^i(i+j) \end{aligned}$$

■

Thus if we are interested in finding a element with rank k, we want to maintain filters a_i, b_i at each level of the i-sample such that for all elements x_j, if $a_i \leq x_j \leq b_i$ then $rank(x_j, P)$ is a contender for an element with rank k. More specifically, $2^i j \leq k \leq 2^i(i+j)$. Thus we want the smallest a_i such that $M_{ia} \geq k$; choosing $a_i = \lceil \frac{k}{2_i} \rceil - i$ suffices. Similarly we want the largest b_i such that $L_{ib} \leq k$; choosing $b_i = \lfloor \frac{k}{2_i} \rfloor$ suffices. So if m_i is the number of elements between a_i and b_i we can see that $m_{i+1} = O(\frac{m_i \log^2(n)}{s})$ where n is the original population size. Thus we can see that our choice in s from the last lecture is correct.

17.2 Coresets

We now switch to some geometric problems in the streaming setting. To do so we start with the notion of coresets via a specific problem (called minimum enclosing ball MEB). Roughly speaking, coresets of a set points is a much smaller (than original input) sample that preserves a lot of properties of the input. The ideas used in this section are similar to the previous two selection algorithms in terms of combining and sparsifying sets.

Definition 1 *A* ***metric space*** *is a pair* (χ, d) *where* χ *is a non-empty set of points and* $d : \chi \times \chi \to \mathbb{R}^{\geq 0}$ *is a distance function satisfying*

1. $d(x, y) = d(y, x)$
2. $d(x, y) = 0 \Leftrightarrow x = y$
3. $d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z \in \chi$*; meaning* d *satisfies the triangle inequality.*

So, if χ is finite, we can think of this as a complete graph where the vertices are the elements of χ and the edges have edge weights corresponding to the distances between pairs of points (and thus the edge weights satisfy the triangle inequality). One example of a distance function would the the ℓ_p-norm; $d(x, y) = \|x - y\|_p$.

For a set $P \subseteq \chi$ and $q \in \chi$. We define the function $cost(P, q) = \max_{p \in P}\{d(q, p)\}$; the maximum distance between q and a point in P.

Definition 2 *Let* $P \subseteq \chi$*. A* ***coreset*** $Q \subseteq P$ *is an* ϵ*-coreset if* $\forall y \in \chi$

$$(1 - \epsilon)cost(P, y) \leq cost(Q, y) \leq (1 + \epsilon)cost(P, y)$$

Generally, a coreset's size is much smaller than the size of the original set. These definitions hold in general, but for the rest of this lecture we will consider $\chi = \mathbb{R}^d$ and $d(x, y) = \|x - y\|_2$ as the Euclidean distance.

17.2.1 Minimum Enclosing Ball and Offline Coreset

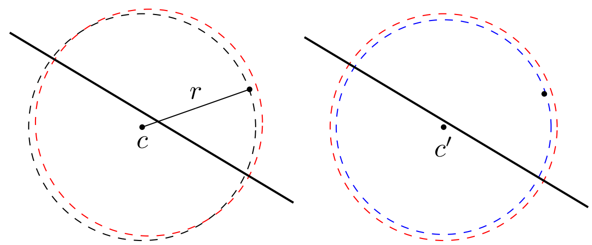
Definition 3 *Let* $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$*. The* ***minimum enclosing ball (MEB)*** *of the set* P *is defined by the point* $x \in \mathbb{R}^d$ *which minimizes* $cost(P, x)$*. The cost function is the radius of this MEB.*

We will use MEB to design an offline coreset algorithm. We will then use this algorithm as a subpart of a streaming algorithm for finding a ϵ-coreset.

Lemma 2 *Suppose* B *is a MEB of* $P \subseteq \mathbb{R}^d$ *with center* c *and radius* r*. Then any enclosed half-space that contains* c *also contains a point* $x \in P$ *such that* $d(x, c) = r$*.*

Proof.

Suppose towards a contradiction that this is not the case. Let H be the half-space and $\bar{H}$ be everything else. Clearly, if H contains no point at distance r, then $\bar{H}$ must. But this means $\exists \delta$ small enough such that a point $c' \in H$ with $d(c, c') = \delta$ but c' is closer to the half-space separation. It can be seen that c' with radius r' is also a MEB for P, where $r' < r$. Below is an (exaggerated) illustration of this for $d = 2$.



■

Theorem 1 *Given $P \subseteq \mathbb{R}_d$, $\exists\epsilon$-coreset $S \subseteq P$ of size $2\epsilon^{-1}$. Equivalently, if r_P, r_S is the radius of a MEB of P, S respectively; then $\frac{1}{1+\epsilon} r_P \leq r_S \leq r_P$.*

Proof. Consider the following algorithm:

MEB ϵ-coreset(P)

1. Let $S_1 = \{p\}$ for any arbitrary point $p \in P$.
2. For $i \leftarrow 1$ to $T = 2\epsilon^{-1}$ do:
   - $c_i \leftarrow$ center of $MEB(S_i)$.
   - $p_i \leftarrow \arg\max_{p \in P} d(c_i, p)$
   - $S_{i+1} = S_i \cup \{p_i\}$
3. Return S_T

Clearly, the return set of this algorithm returns a set of size $2\epsilon^{-1}$ so we need only show that for $S = S_T$, that for r_P, r_S defined in the statement, $\frac{1}{1+\epsilon} r_P \leq r_S \leq r_P$. Since $S \subseteq P$, it is clear that $r_S \leq r_P$ thus we need only show the former inequality. Define the following variables:

- r_i as the radius of $MEB(S_i)$
- $\lambda_i = \frac{r_i}{r_P}$
- $\delta_i = \|c_i - c_{i+1}\|$

First, we notice that $\forall i, \exists q \in P$ such that $d(c_i, q) \geq r_P$ (by definition of MEB). So, by the triangle inequality and our definitions we get that:

$$\lambda_i r_P = r_{i+1} \geq d(q, c_{i+1}) \geq d(q, c_i) - d(c_i, c_{i+1}) \geq r_P - \delta_i$$

So, if $\delta_i = 0$, then we are done. So consider $\delta_i > 0$. This means, $\exists p \in P$ such that $d(p, c_i) = r_i = \lambda_i r_P$. Thus since we are using euclidean distances:

$$r_{i+1} \geq d(c_{i+1}, p) = \sqrt{r_i^2 + \delta_i^2} = \sqrt{\lambda_i^2 r_P^2 + \delta_i^2}$$

This combined with the fact above implies that $r_{i+1} \geq \max\{r_P - \delta_i, \sqrt{\lambda_i^2 r_P^2 + \delta_i^2}\}$ which is minimized when $r_P - \delta_i = \sqrt{\lambda_i^2 r_P^2 + \delta_i^2}$. Solving for δ_i gives $\delta_i = \frac{1}{2}(1 - \lambda_i^2) r_P$. Substituting this δ_i into the first equation and solving gives us that $\lambda_{i+1} \geq \frac{1}{2}(1 + \lambda_i^2)$. Solving this recursion finally gives that $\lambda_i \geq 1 - \frac{1}{1+i/2}$. Finally, to have $\lambda_T \geq 1 - \epsilon$ (for our desired result for r_S) it is enough to set $T = 2\epsilon^{-1}$. ■

Finally, without proof, we note that there is a way to build coresets of size $O(\frac{1}{\epsilon^{(d-1)/2}})$ which is an improvement for $d = 2$.

17.2.2 Streaming Model

Now we look at solving this problem in a streaming model. First, we consider the following remarks (without proof).

Remark 1 If Q, Q' are ϵ-coresets for P, P' respectively, then $Q \cup Q'$ is an ϵ-coreset for $P \cup P'$.

Remark 2 If R is an ϵ-coreset for Q and Q is an ϵ'-coreset for P then R is an $(\epsilon + \epsilon')$-coreset for P.

So, if we split the input stream into chunks of size B, we can build a coreset for each chunk as leaves of a tree, and combine pairs of coresets using the above remarks until we have a single coreset for the whole stream. If the stream is of size m, then the tree would have a height of $\log \frac{m}{B}$. If we combine the coresets as soon as possible while building the tree, then we will need at most $O(\log m)$ coresets at any point in time. For the analysis below, let $A(\epsilon)$ be the space complexity of an ϵ-coreset. If we use our algorithm that was previously mentioned, $A(\epsilon) = O(\epsilon^{-1})$. We have two methods of getting coresets for an input stream:

- **Method 1:** At the first level of making the coresets, make δ-coresets (where $\delta = \frac{\epsilon}{\log m}$) and combine two coresets Q_1 and Q_2 by finding a δ-coreset for the set $Q_1 \cup Q_2$. Using the second remark, the final coreset of this algorithm will be an ϵ-coreset with a space complexity $O(A(\frac{\epsilon}{\log m}) \log m)$.
- **Method 2:** At the first level of making the coresets, make ϵ-coresets and combine them by using the ϵ-coreset $Q_1 \cup Q_2$. By the first remark, the final coreset of this algorithm will be an ϵ-coreset with a space complexity $O(A(\epsilon) \log^2 m)$.

In both methods if we are using the algorithm that was presented we have an algorithm for obtaining an ϵ-coreset for a stream that uses $O(\epsilon^{-1} \log^2 m)$ space.

References

AHV04 P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, Approximating Extent Measures of Points. *Journal of the ACM*, 51(4):606635, 2004.

BC03 M. Bădoiu and K. L. Clarkson, Smaller Core-sets for Balls. *In SODA '03: Proc. of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

BHI02 M. Bădoiu, S. Har-Peled, and P. Indyk, Approximate Clustering via Core-sets. *In Proc. of the 34th Annual ACM-SIAM Symp. on Theory of Computing*, 2002.

MP80 J. I. Munro and M. S. Paterson, Selection and Sorting with Limited Storage. *Theoretical Computer Science*, 12(3):315-323, 1980.