University of Wrocław: Algorithms for Big Data (Fall'19) 13/01/2020

Lecture 12: Coresets

Lecturer: *Przemysław Uznański* Scribe: -

1 Coresets

Setup: given set of $P \subseteq \mathbb{R}^d$, compute $C_P() : \mathbb{R}^d \to \mathbb{R}$. For example:

- MEB, a minimal enclosing ball: $C_P(o)$ is a radius of minimal enclosing ball for set P centered at o.

$|P| = n$ is large, so storing it explicitly is out of the question. Observe that for fixed o, a single $p \in P$ is enough, that is $\forall_o \exists_{p \in P} C_P(o) = C_{\{p\}}(o)$. Can we generalize this observation so it works for all $o \in \mathbb{R}^2$ at once?

Definition 1. *We say that $S \subseteq P$ is a c-coreset for P if for any o and any $T \subseteq \mathbb{R}^d$ there is:*

$$C_{S\cup T}(o) \leq C_{P\cup T}(o) \leq c \cdot C_{S\cup T}(o)$$

Note: this is a stronger definition than just requiring that $f_S(o)$ is a c-approximation to $C_P(o)$ (take $T = \emptyset$).

Observe that in case of MEB we have for $A \subseteq B$: $C_A(o) \leq C_B(o)$, so the first inequality is "for free".

Lemma 2 (Merge property)**.** *If S is a c-coreset for P, and S' is a c-coreset for P', then $S \cup S'$ is a c^2-coreset for $P \cup P'$.*

Lemma 3 (Reduce property)**.** *If S is a c-coreset for P and P is a c-coreset for Q, then S is a c^2-coreset for Q.*

We sometimes want a stronger property (which for example MEB satisfies)

Property 4 (Disjoint merge)**.** *If S is a c-coreset for P and S' is a c-coreset for P' and $P \cap P' = \emptyset$, then $S \cup S'$ is a c-coreset for $P \cup P'$.*

Exercise: proof for MEB.

Theorem 5. *Assume that a problem is supported by a $(1+\alpha)$-coreset of size $f(\alpha)$, computable in linear space, with disjoint merhe property. Then there is a streaming algorithm with $1+\varepsilon$ guarantee, with space $\mathcal{O}(f(\varepsilon/\log n)\log n)$.*

Proof. Sketch of a proof: put stream of n elements into binary tree. Each node stores coreset for the range below it.

For two sibling nodes N_1, N_2 covering sets A_1, A_2, at level i, and parent node N at level $i+1$, there is:

- N_1 is $(1+\alpha)^i$-coreset for A_1 (and the same for N_2, A_2)

- $N_1 \cup N_2$ is $(1+\alpha)^i$-coreset for $A_1 \cup A_2$
- N is constructed as $(1+\alpha)$-coreset for $N_1 \cup N_2$
- thuse N is $(1+\alpha)^{i+1}$-coreset for $A_1 \cup A_2$

As a end-result, we have in the root $(1+\alpha)^{\log n}$-coreset for whole input. Selecting $\alpha = \mathcal{O}(\varepsilon/\log n)$ is enough. □

1.1 Coreset for MEB

Construction goes as follow. Choose dense set of directions $\{v_i\}_{i=1}^m$, such that for any other direction u, there is always some v_i such that angle$(u, v_i) \leq \alpha$: this is $\sim \alpha$-net on unit-ball (up to trigonometry). We can choose such set of $m = (1/\alpha)^{\mathcal{O}(d)}$ directions.

Claim 6. *For any direction v_i, pick $p_i \in P$ that is extremal in that direction. Set $S = \{p_i\}_{i=1}^m$ is a $(1+\mathcal{O}(\alpha^2))$-coreset for P.*

Proof. Pick arbitrary T ($T = \emptyset$ w.l.o.g. in this proof), and P and constructed set S. Fix o. Pick furthest point $x \in P$, and close direction v_i, and maximal in this direction point $p_i \in S$. The angle x-o-p_i is small (at most α), so the stretch is upperbounded by $\frac{1}{\cos\alpha} = 1 + \mathcal{O}(\alpha^2)$. □

1.2 Coreset for median

Approximate median: given sequence of numbers A of number $a_1, \ldots, a_n$, return a such that $(1/2 \pm \varepsilon)n$ elements in A are smaller/larger than a.

Alternative formulation: find a that minimizes $C_A(a) = \max(|\{i : a_i \geq a\}|, |\{i : a_i \leq a\}|)$.

Coreset of size $1/\varepsilon$: pick every εn element from sorted A. Easy to see that

$$C_{A\cup T}(a) \leq C_{A\cup S}(a) \leq (1+\varepsilon) C_{A\cup T}(a)$$

Plugging into the theorem, we obtain streaming median computation in space $\mathcal{O}(\log^2 n/\varepsilon)$. In fact this works for any quantile computation (but the error is additive). Improvement: pre-filter and keep only $1/\varepsilon^2$ elements (randomly), so space becomes $\mathcal{O}(\log^2(1/\varepsilon)/\varepsilon)$.

2 Graph algorithms

2.1 Certificates

Graph-theoretic approach, for decision problems.

Definition 7. *For property $\mathcal{P}$, and a graph G, we say that G' is a strong certificate for G if: for any H, $G \cup H$ is in $\mathcal{P}$ iff $G' \cup H$ is in $\mathcal{P}$.*

Examples:

- connectivity: any spanning forest
- non-bipartiteness: any spanning forest + single odd-cycle inducing edge
- edge/vertex connectivity

2.2 Spanners

Subgraph approximately preserving distances:

Definition 8. *H, an edge subgraph of G, is a t-spanner of G, if for any u, v there is*

$$d_H(u, v) \leq t \cdot d_G(u, v)$$

Theorem 9. *Any unweighted G contains t-spanner with $\mathcal{O}(n^{1+2/(t-1)})$ edges, and it can be computed in one pass.*

Proof. Maintain subgraph. Process edges one-by-one. If an edge causes a cycle of length $\leq t$, do not insert it.

1. This constructs a t-spanner. (Nice easy exercise)
2. The graph is sparse enough:
   - Let $d = 2m/n$ be average degree.
   - There is subgraph H of G with minimum degree $d' = d/2$: keep removing vertices with degree smaller than d'. We cannot end with no vertices, since then we removed less than m edges.
   - H has every vertex of degree at least m/n, and no cycle of length t or less.
   - BFS tree of depth $(t-1)/2$ has no cycles in H
   - $(m/n - 1)^{(t-1)/2} \leq |H| \leq n$, which implies bound on m

□

For weighted graphs: partition edges into $\log W$ classes, t-spanner of size $\mathcal{O}(n^{1+2/(t-1)} \log W)$

3 Massively Parallel Computing

Modern distributed computing model (captures map-reduce, hadoop, spark, etc.).

- input of size n
- M machines, each of space S, $S = n^{1-\delta}$, $M \cdot S = \mathcal{O}(n)$.
- output (might be too large, e.g. size n that does not fit on a single machine)

Computation:

- computation happens over R rounds
- each machine: near linear computation per round, so total computation cost $\mathcal{O}(n^{1+o(1)}R)$
- each machine communicates $\sim S$ bits per round, so total communication cost $\mathcal{O}(nR)$

goal: minimize R

3.1 Sorting (Tera-Sort)

Intuition: if we partition input onto machines, so each machine receives contiguous fragment of size S, then we are done in a single round (each machine sorts and outputs its own part, output == concatenation of outputs).

Idea:

- each machine receives input
- each machine samples randomly from its input
- each sample is sent to single machine (1 round)
- 1 machine gathers all the samples, sorts locally, and sents back to everyone approximate histogram
- machines use approximate histogram to decide how to partition locally their input and sent it to proper receivers
- then everyone sorts their parts

Total sample size $k = \mathcal{O}(S)$, and whp histogram is with $\pm\varepsilon n$ error, where $\frac{\log n}{\varepsilon^2} \leq k$. We are ok if error is $\mathcal{O}(S)$. This is satisfied when $S^{3/2} = \Omega(n\sqrt{\log n})$, or $S = \Omega(n^{2/3}(\log n)^{1/3})$ (thus whp we can sort in $\mathcal{O}(1)$ rounds).

To be continued