Lecture 15

1 Overview

We continue our study of the insertion-only streaming model. Now, we will see an approach for solving problems in this model via a framework known as coresets. Essentially, given an optimization problem whose input is a set of points $P \subseteq \mathbb{R}^d$, a coreset $S$ for this problem is a subset of $P$ such that we can get an approximate solution solving the same problem on the coreset. Also, coresets satisfy some kind of merge property, that allows to solve the problem via decomposition in smaller subproblems.

As examples of the use of this framework, we will apply this technique to the Minimum Enclosing Ball and the Approximate $\epsilon$-median problem.

Note that there are several definitions of core-sets out there, and they do not always mean the same thing. The definition below is perhaps the simplest one, and demonstrates connections to the notion of sparse graph certificates, which we will cover in the next lecture.

For more on core-sets, see the survey [2].

2 Coresets

Suppose we want to minimize a function $C_P : \mathbb{R}^d \to \mathbb{R}$ parametrized by a collection of points $P \subseteq \mathbb{R}^d$. We say that $S \subseteq P$ is a $c$-coreset for $P$ if for any set $T$ of points and for any point $p$, we have the following inequality:

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

Note that this approximation guarantee is stronger than just a good bound for the optimal solution.

In this lecture, we will only deal with monotone functions $C$, that is, $A \subseteq B \Rightarrow C_P(A) \leq C_P(B)$. Therefore, the first inequality holds for free. In some cases we also allow the coreset to be a multiset, as in the example we will see in Section 4.

The main topic of this lecture is how to use coresets in streaming problems. The following two lemmas are basic properties of coresets that will be useful for this purpose. Both are very simple and easy to prove.

**Lemma 1.** [Merge Property] If $S$ is a $c$-coreset for $P$, and $S'$ is a $d$-coreset for $P'$, then $S \cup S'$ is a $cd$-coreset for $P \cup P'$.

*Proof.* The property follows from:

$$C_{S \cup S' \cup T}(p) \leq c \cdot C_{P \cup S' \cup T}(p) \leq cd \cdot C_{P \cup P' \cup T}(p).$$

**Lemma 2.** [Reduce Property] If $S$ is a $c$-coreset for $P$, and $P$ is a $d$-coreset for $Q$, then $S$ is a $cd$-coreset for $Q$. 
Proof. The property follows from:

\[ C_{S \cup T}(p) \leq c \cdot C_{P \cup T}(p) \leq cd \cdot C_{Q \cup T}(p). \]

Now we will establish the connection between coresets and streaming algorithms. Essentially, the following result implies the existence of an streaming algorithm when a problem admits a coreset.

**Theorem 3.** Suppose that a problem admits a \((1 + \delta)\)-coreset of size \(f(\delta)\) that is computable in linear space in the number of input points. Then, there exist a streaming \((1 + O(\epsilon))\)-approximation algorithm for the problem that runs in \(O(f(\epsilon/\log n) \log n)\) space.

Proof. Let \(P = p_1, p_2, \ldots, p_r\) be the sequence of points seen so far. Let \(\delta = \epsilon/\log n\). The streaming algorithm partition \(P\) into \(O(\log n)\) sets \(P_0, P_1, \ldots, P_t\), such that either \(P_i\) is empty, or \(|P_i| = 2^i M\), where \(M\) is a constant. Also, the algorithm keeps a collection of coresets \(Q_0, Q_1, Q_2, \ldots, Q_t\). The algorithm works as follows.

Initially, all \(P_i\) and \(Q_i\) are empty sets. When a new element \(p_u\) arrives, we add it to \(P_0\) and \(Q_0\). If \(P_0\) has less than \(M\) elements, then we are done, and we wait for the next element in the stream. Otherwise, we do the following loop:

- Move all the elements in \(P_i\) to the next level \(P_{i+1}\).
- Move all the elements in \(Q_i\) to the next level \(Q_{i+1}\). If \(Q_{i+1}\) was empty before, then we are done and we exit the loop. If \(Q_{i+1}\) was not empty we replace \(Q_{i+1}\) with a \((1 + \delta)\)-coreset for \(Q_i\) and we loop again with \(i = i + 1\).

Note that this procedure ensures that whenever \(Q_i\) is non empty, then it is a coreset for \(P_i\) that has size \(2^i M\) (except maybe for \(P_0\)). The set \(\cup Q_i\) is a coreset for the stream \(P\) that we can use to solve the problem in \(P\) approximately.

The algorithm only requires to keep the coresets \(Q_i\). Since there is one per each value of \(i\), the space used is \(O(f(\delta) \log n)\). The approximation factor of the coreset \(\cup Q_i\) is \(O((1 + \delta)^{\log n})\). Taking \(\delta = \epsilon/\log n\) we obtain the desired guarantee.

The previous theorem allow us to prove the existence of an algorithm for a streaming problem just by proving that the problem admits a coreset. Moreover, a slightly modified version of the theorem can be applied when the linear space requirement in the computation of the coreset doesn’t hold.

There are several applications of the use of this technique, for example:

- Find the diameter, minimum enclosing ball, or width of a collections of points [1].
- Find the \(k\)-medians or the \(k\)-centers of a collection of points [5, 4].

We will see two examples of the use of this technique in the next sections.
3 Coresets for the Minimum Enclosing Ball (MEB)

We are ready to see a first application of coresets to streaming algorithms. We will see a purely
geometric problem, which is a kind of natural application of this framework.

The minimum enclosing ball (MEB) of a set of points $P$ in $\mathbb{R}^d$ is the smallest ball that encloses
all the points in $P$. We are interested in the radius of the ball, and also in the center $p$.

We can define $C_P(p)$ as the smallest radius of a ball centered in $p$ containing $P$, and we have
the following result:

**Theorem 4.** There is a $(1 + O(\alpha^2))$-coreset for MEB of size $O(1/\alpha^{d-1})$.

**Proof.** To construct the coreset, the idea is to choose directions $v_1, v_2, \ldots, v_k$ with the following
property: for any non zero vector $u$ there is a vector $v_i$ such that the angle between $v_i$ and $u$ is
at most $\alpha$. It turns out that in $\mathbb{R}^d$, it is possible to do this with $k = O(1/\alpha)^{d-1}$ vectors. We will
assume this fact without a proof. Note that this fact is obvious for $d = 2$.

Then, for every direction $v_i$ we project all the points of $P$ in this direction and we include the
extremal points in the coreset. Let $Q$ be the coreset of extremal points obtained. We will see that
this is a $1 + O(\alpha^2)$ coreset for $P$.

Let us take any set $T$ of points and let us compare the radius $R_Q$ and $R_P$ of the MEB of $Q \cup T$
and $P \cup T$ when a center $p$ is given. The difference between both radii is maximum when a point
$q \in P - Q$ is as far as possible from $p$. The worst case occurs when $q$ forms an angle of $\alpha/2$ with
some $v_i$ and it is as far from $p$ as possible. Since the projection of $q$ on $v_i$ is at distance at most
$R_Q$ from $p$, it turns out that $R_P \cdot \cos(\alpha/2) \leq R_Q$. Using standard approximations we get

\[ R_P \leq R_Q \left(1 + O(\alpha^2)\right), \]

which is the desired result.

\[ \Box \]

Now we have proved the existence of a coreset for MEB, by Theorem 3 we can conclude the
existence of a $(1+O(\epsilon))$-approximation streaming algorithm that uses $O(\text{polylog}(n)/\epsilon^{(d-1)/2})$ space.

4 The $\epsilon$-approximate median problem.

In this section we show how the concept of coreset can be used to solve a non-geometric problem.

The median of a sequence of numbers $A = \{a_1, a_2, \ldots, a_n\}$ is the number in the middle of the
sequence after sorting its elements in increasing order (to make life easier, we assume that $n$ is an
odd number, so that the middle is well-defined). This problem can be formulated as an optimization
problem, that is, finding $a$ such that:

\[ C_A(a) \equiv \max\{|\{i : a_i \geq a\}|, |\{i : a_i \leq a\}|\}, \]

is minimized.

An approximate version of this problem is the following: given a sequence of numbers $A = \{a_1, a_2, \ldots, a_n\}$, the $\epsilon$-approximate median problem is to find an element $a_i$ such that at least
$1/2 - \epsilon$ of the elements in the sequence are greater than $a_i$, and at least $1/2 - \epsilon$ of the elements in
the sequence are less than $a_i$.

The following result establish the existence of coresets for this problem.
Theorem 5. There is an \((1 + \delta)\)-coreset of size \(O(1/\delta)\) for the problem of finding a median.

Proof. The coreset for a sequence \(A = a_1, a_2, \ldots, a_n\) is the set of elements \(a'_1, a'_{\delta n+1}, a'_{2\delta n+1}, \ldots, a'_n\), where \(a'_1, a'_2, \ldots, a'_n\) is the sequence \(A\) ordered increasingly. Each each element is repeated \(\delta n\) times in the coreset (In this case, we allow the coreset to be a multiset, but the size of the coreset is just the number of distinct elements, that is, \(O(1/\delta)\)).

It is easy to see that:

\[
C_{\text{AUT}}(a) \leq C_{\text{SLT}}(a) \leq (1 + O(\delta))C_{\text{AUT}}(a),
\]

therefore it is a coreset with the desired property. 

From this, and using Theorem 3, we conclude the existence of a \((1 + O(\delta))\) approximation algorithm for finding a median in the streaming model, that uses \(O(\log^2 n/\delta)\) space. This can be seen as an algorithm that solves the \(\epsilon\)-approximate median using \(O(\log^2 n/\epsilon)\) space.

Note that a \(O(\log^2 n/\epsilon)\)-space for the median problem has been effectively known for almost three decades [6], well before the idea of core-sets was introduced. Interestingly enough, the algorithm in [6] is quite similar to the one presented in this lecture. However, it is not optimal - one can reduce the space bound by a factor of \(\log n\) [3].

References


