Lecture 14

1 Overview

So far in the course we have studied several problems under a streaming model that consider insertions and deletions. There is a big qualitative difference when we restrict the streaming model by allowing only insertions. These differences include not only better algorithms, but also different algorithmical approaches (typically randomized algorithms in the former case, and typically deterministic in the later case).

We have seen an algorithm for the Heavy Hitters problem (also known as *most frequent elements*) in the insertion and deletion streaming model. For the $L_1$ norm we learned a $O(1/\varphi \log m)$ space algorithm. We will see in this lecture that we can use $O(1/\varphi)$ space when only insertions are allowed.

We will also introduce the metric $k$–median problem. Given a set $n$ points (clients), we want to find a set of $k$ points (facilities) that minimizes the average distance of the clients to their closest facility. We will see a $O(1)$–approximation and $O(kn^\alpha)$–space algorithm to this problem in the insertion only model, for all $\alpha > 0$.

2 Heavy hitters

We define the Heavy Hitters of a vector (for the $L_1$ norm) as follows:

$$ HH_\varphi(x) = \{ i \text{ s.t. } |x_i| > \varphi \|x\|_1 \} $$

(note: for convenience, we are using strict inequality today). In Lecture 4 we discussed how to get approximations of the heavy hitters set: given $\varphi > \varphi'$, we want to find a set of coordinates $S$ such that $HH_\varphi(x) \subseteq S \subseteq HH_{\varphi'}(x)$. We learned a $O(1/\varphi \log m)$ space algorithm for this problem in the insertion and deletion model.

Now we will find out another approximation for heavy hitters providing an algorithm that uses only $O(1/\varphi)$ space in the insertion only model. This approximation is defined by the following problem:

APROX-HH

**INPUT:** $\varphi \in [0, 1]$

**OUTPUT:** a set of coordinates $S$ such that $S$ contains $HH_\varphi(x)$ and its size is $O(1/\varphi)$.

Note that this approximation is weaker than the previous one, when we take for example, $\varphi = \varphi' + \epsilon$ in the original approximation.

**Theorem 1.** There is a deterministic algorithm for APROX-HH [1, 3].

**Proof.** First, we will see how the algorithm works for the case $\varphi = 1/2$.

Algorithm ($\varphi = 1/2$)

- Set $c = 0$, $e = \text{NULL}$.
• For each stream element $a$:
  - If $c = 0$ then $e = a$.
  - If $e = a$ then $c = c + 1$, else $c = c - 1$.
• Report $e$.

Suppose there is a majority element $m$, that is, it appears more than half of the total number of elements. We have that between any two consecutive times that $c$ takes the value 0, the number of times that $m$ appears is at most one half of the total of numbers read. Therefore, whenever $c = 0$ we know that $m$ still is a majority element in the remaining of the stream. The last time the algorithm set $c = 0$, necessarily it also has to set $e = m$, and the algorithm reports the element $m$. When there is no majority element, the algorithm gives (incorrectly) one element as output.

Now, we will see the extension of the previous algorithm for the general case. Suppose $\phi = 1/L + 1$.

Algorithm ($\phi = 1/L + 1$)
• Set $S = \emptyset$.
• For each stream element $a$:
  1. If $a \notin S$ and $|S| < L$ then add $a$ to $S$ and set $c_a = 0$.
  2. If $a \in S$ then set $c_a = c_a + 1$, else set $c_{a'} = c_{a'} - 1$ for all $a' \in S$.
  3. For all $a \in S$: if $c_a = 0$ then remove $a$ from $S$.
• Report $S$.

Note that set $S$ obtained at the end of the procedure has at most $L$ elements, so its size is $O(1/\phi)$. Also note that whenever the else condition in step 2 of the main loop is executed, the set $S$ has $L$ elements, so exactly $L$ counters $c_{a'}$, $a' \in S$ are decremented.

If $m$ is a heavy hitter element, then it appears more then $n/(L + 1)$ times. Let’s considering the following events:

1. At step 2 of the main loop we read $m$ and execute the “then” branch, that is, we increment $c_m$. Let $n_i$ be the number of times this situation occurs.
2. At step 2 of the main loop we read $m$, and execute the “else” branch, that is, we decrement $L$ counters in $c$ (of course, different from $c_m$). Let $n_d$ be the number of times this situation occurs.
3. At step 2 of the main loop we don’t read $m$, we execute the “else” branch and we decrement $c_m$. Note that in this case $c_m$ is at most decremented by one, and it is the only case when $c_m$ is decremented. Let $n_e$ be the number of times this situation occurs.

Note that $n_i + n_d$ is the total number of occurrences of $m$, therefore, $n_i + n_d > n/(L + 1)$. Also, every time we execute the “else” branch we decrement $L$ counters, and we are also reading the current element without incrementing any counter. Moreover, each of those $L + 1$ events can
be uniquely “charged” to one of the stream elements, which implies that \((L + 1)(n_d + n_e) \leq n\).

Therefore,

\[ c_m = n_i - n_e > n/(L + 1) - n_d - n_e \geq 0, \]

so \(m\) is reported by the algorithm.

### 3 Metric \(k\)-median

The metric \(k\)-median is a version of the facility location problem. The setup is the following: we have oracle access to a distance function \(d(\cdot, \cdot)\) between pairs of points in certain space \(X\). We have a stream of points \(S = \{p_1, p_2, \ldots, p_n\} \subset X\), and a fixed number \(k \leq n\). We want to find a set \(Q \subset S\) of \(k\) points that minimize the average distance of the points in \(S\) to the set \(Q\). Specifically, the distance \(d(p, Q)\) of a point \(p\) to a set \(Q\) is defined as follows:

\[ d(p, Q) = \min_{q \in Q} d(p, q), \]

and the problem is to find \(Q \subset S\) of size \(k\) that minimizes \(\sum_{p \in S} d(p, Q)\). Note that we are using uppercase letters for sets and lower case for points, in order to use \(d\) with different meanings. For simplicity, we will use \(d(S, Q)\) to represent \(\sum_{p \in S} d(p, Q)\), and \(\text{cost}(S, Q)\) to represent \(\min_{Q' \subseteq Q, |Q'| = k} d(S, Q)\).

We will show an \(O(1)\)-approximation algorithm for this problem using \(O(\sqrt{kn})\) space, which can be found in [4]. By using recursion, it is possible to reduce the space to \(O(kn^\alpha)\) for any constant \(\alpha > 0\).

First we will prove some lemmas that will be useful in the analysis of the algorithm.

**Lemma 2.** For all set \(S\) and for all set \(Q\), \(\text{cost}(S, S) \leq 2\text{cost}(S, Q)\).

**Proof.** Let \(Q' \subset Q\) be a set of size \(k\) such that \(d(S, Q')\) is minimum. We can construct a set of \(k\) medians using points in \(S\) just by taking the nearest neighbor \(\overline{q} \in S\) of each point of \(q \in Q'\). For every \(p \in S\) whose closest median in \(Q'\) is \(q\) we have \(d(p, \overline{q}) \leq d(p, q) + d(q, \overline{q})\). By summing over all \(p \in S\) and bounding, we can easily show that this solution has cost at most \(2\text{cost}(S, Q)\).

**Lemma 3.** For all sets \(S\) and for all partitions \(\{S_i\}\) of \(S\), \(\sum_i \text{cost}(S_i, S_i) \leq 2\text{cost}(S, S)\).

**Proof.** By Lemma 2, we have that \(\text{cost}(S_i, S_i) \leq 2\text{cost}(S_i, S)\). After summing over \(i\) the result follows.

Now, we will show the algorithm.

**Theorem 4.** There is a deterministic \(O(1)\)-approximation algorithm for the \(k\)-median problem that runs in \(O(\sqrt{kn})\) space.

**Proof.** The following algorithm uses a divide and conquer approach. First, it solves the \(k\)-medians in smaller subsets of points, and then merges the solutions by applying \(k\)-medians again over the \(k\)-medians found. We will assume the existence of an algorithm [2] for \(k\)-medians that uses linear space and which is known to be \(b\)-approximate, for some constant \(b > 1\).

**Algorithm:**
1. Partition the stream in blocks $S_1, S_2, \ldots, S_L$ of the same size, with $L = \sqrt{n/k}$

2. For each block $S_i$
   
   (a) Find $Q_i \subset S_i$ of size $k$ that solves $k$-median for the block $S_i$, using the $b$-approximation algorithm.
   
   (b) Compute, for all $q \in Q_i$, the number $w^q$ of points $p \in S_i$ such that its nearest point $q_p \in Q_i$ is $q_p = q$.

3. Using the $b$-approximation algorithm, find $Q$ of size $k$ that solves $k$-median for the set of points in the multiset $\cup Q_i$, but weighting each element $q \in Q_i$ by a factor $w_q$.

4. Return $Q$

Step 3 is only find $k$-medians in the set of $kL$ $k$-medians found in step 2, but assigning weights to each element in the set. This means that every element is repeated as many times as its weight. Since it is possible to solve the problem in space linear in the number of different elements, it is clear that this algorithm uses $O(\sqrt{k}n)$ space.

Let’s analyze the approximation guarantee. At the end of Step 2, the algorithm finds $k$ medians for each $S_i$. Since we are using a $b$-approximation algorithm, we get that the cost of the solution found by this algorithm is at most $b$ times the optimum value $\text{cost}(S_i, S_i)$. Therefore, if we use the $Lk = \sqrt{n/k}$ medians in the multiset $\cup Q_i$ as medians for the entire set $S$, we find a solution for this problem with cost at most $b \sum_i \text{cost}(S_i, S_i) \leq 2b\text{cost}(S, S)$, by Lemma 3.

Now, in the Step 3 of the algorithm we reduce the $Lk$ medians to a set $Q$ of only $k$-medians. Note that the distance of any point $p \in S$ to the set $Q$ can be bounded by the distance from $p$ to the nearest median $q_p$ in $Q_i$ plus the distance from $q_p$ to $Q$:

$$d(p, Q) \leq d(p, q_p) + d(q_p, Q),$$

for all $p \in S$.

Therefore,

$$d(S, Q) \leq \sum_i d(S_i, Q_i) + \sum_{p \in S} d(q_p, Q) \leq 2b\text{cost}(S, S) + \sum_{p \in S} d(q_p, Q).$$

The last term $\sum_{p \in S} d(q_p, Q)$ is just the cost of the solution $Q$ for the weighted $k$ median problem solved in step 3. We will bound this cost as follows: the optimal solution $Q^*$ for $k$-medians on $S$ (using points in $S$) can be used as medians for the multiset $\cup Q_i$ (weighted according to the coefficients $w_i^q$). The cost of this solution is at most $2bd(S, S) + d(S, S)$, because if $p^*$ is the nearest neighbor in $Q^*$ of $p$, then $d(q_p, p^*) \leq d(q_p, p) + d(p, p^*)$, so, adding this for every $p \in S$, we get that the cost of the solution is at most $\sum_i d(S_i, Q_i) + \text{cost}(S, S) \leq 2b\text{cost}(S, S) + \text{cost}(S, S)$. Therefore, the optimal cost of the weighted problem using only points in the multiset has a cost at most $2(2b\text{cost}(S, S) + \text{cost}(S, S))$ by Lemma 2.

Hence,

$$d(S, Q) \leq 2b\text{cost}(S, S) + 2b(2b\text{cost}(S, S) + \text{cost}(S, S)) \leq 4b(b + 1)\text{cost}(S, S),$$

which completes the proof.
References


