CS-621 Theory Gems November 20, 2012

Lecture 18

Lecturer: Aleksander Mądry *Scribes: Abdallah Elguindy and Milan Korda*

1 Introduction

So far, we have seen streaming algorithms for two important variants of L_p-norm estimation problem: L_0-norm estimation (the distinct elements problem) and L_2-norm estimation. We also noted that the L_1-norm estimation problem (at least, when we do not allow element deletions) corresponds to just computing the length of the stream and thus can be trivially solved in $O(\log n)$ space. Therefore, the next natural step would be to try to approach the task of L_∞-norm estimation, i.e., estimating the frequency of the most frequently occurring element.

Unfortunately, as mentioned last time, one can show (and we will see it soon) that L_∞-norm estimation requires $\Omega(m)$ space, which is completely prohibitive from the-point of view of streaming algorithms. This is rather disappointing, as ability to efficiently compute this very fundamental statistic of the data streams would be very valuable.

Fortunately, despite this negative result, there is still hope for getting something useful here. Namely, as we will see later, the L_∞-norm estimation problem instances that are used in the $\Omega(m)$ lowerbound are all corresponding to a situation in which every element (including the most frequent one) has only very few (in fact, at most two) occurrences in the stream. However, this kind of instances are not really that interesting from the-point of view of the intended applications of an L_∞-norm estimation algorithm. The scenarios that we would really like to address are the ones in which the most frequent element appears with substantial frequency (think, e.g., about a router trying to detect a Denial-of-Service attack).

As it turns out, once we reformulate our problem to allow it to not provide any meaningful answer when there is no very frequently appearing elements, the $\Omega(m)$ lowerbound does not hold anymore and one can obtain very satisfactory algorithms for such scenarios.

2 ℓ_q-point Query Problem

The precise problem that we want to purse today is the *ℓ_q-point query* problem. In this problem, given a stream $\boldsymbol{y} = (y_1, \ldots, y_n)$, the desired accuracy parameter $\varepsilon > 0$ and failure probability upperbound $\delta > 0$, we want to devise a streaming algorithm that produces frequency estimates $\hat{x}_j$, for each element $j \in [m]$, such that, with probability at least $1 - \delta$, we have

$$\hat{x}_j = x_j \pm \varepsilon \|x\|_q,$$

for *all* j. Here, x_j is the true frequency of the element j, i.e., the number of occurrences of this element in the stream $\boldsymbol{y}$, and $\|x\|_q$ is the ℓ_q-norm of the true frequency vector $\boldsymbol{x}$.

Note that due to the additive nature of the allowed error, the frequency estimate $\hat{x}_j$ is only meaningful whenever j is a relatively frequently occurring element, i.e., whenever $x_j > \varepsilon \|x\|_q$. So, even though in principle one can use these estimates to perform L_∞-norm estimation, this estimation will be most likely incorrect when elements appear only few times and thus the $\Omega(m)$ lowerbound that we discussed above does not apply here anymore.

Also, it is worth observing that, as we always have that $\|x\|_p \geq \|x\|_{p'}$ whenever $1 \leq p \leq p'$, the larger q the better the quality of the corresponding estimates $\hat{x}_j$ is.

3 ℓ_1-point Query Problem

It is natural to start our investigation of streaming algorithm for the ℓ_q-point query problem by considering the case of $q = 1$.

3.1 Count Algorithm

First, let us consider the following simple (and deterministic!) algorithm that is parametrized by a parameter r.

Algorithm Count:

1. Maintain a counter $c(j)$ for each element j. Initially, all $c(j)$s are equal to zero and we will make sure that at any-point of time at most r of them is non-zero.
2. In each step t, examine the t-th element y_t of the stream.
   - If $c(y_t) > 0$ or the number of non-zero counters is less than r, increment $c(y_t)$ by one;
   - Otherwise, *decrement* all (r) non-zero counters by one.
3. At the end, return an estimate $\hat{x}_j = c(j)$ for each element $j \in [m]$.

Clearly, as it is always the case that at most r counters is non-zero, this algorithm can be implemented using only $O(r \log n)$ space.

It is easy to see that at the end we always have that

$$x_j \geq \hat{x}_j,$$

for each $j \in [m]$. So, our estimates are always a lowerbound on the true frequencies.

We also claim that we always have that

$$\hat{x}_j \geq x_j - \frac{n}{r}.$$

To see why this is the case, let x_j^t (resp. $c(j)^t$) be the number of hitherto occurrences of element j (resp. the value of the counter $c(j)$) at the end of t-th step. Clearly, we have that $x_j^0 - c(j)^0$ is zero. Furthermore, this difference never decreases and can increase only in steps in which the decrement of r non-zero counters was triggered. But, each such decrement decreases the sum of all $c(j)$s by r and this sum starts at zero, never becomes negative, and can be increment by at most one in each step. So, as there is n steps overall, the total number of steps in which decrements of r non-zero counters occurs is at most n/r. Therefore, we can conclude that

$$x_j - \hat{x}_j = x_j^n - c(j)^n \leq \frac{n}{r},$$

as desired.

Thus, we see that for any parameter r, the Count algorithm provides estimates of element frequencies up to an additive error of $\frac{n}{r}$ and that it has space complexity of $O(r \log n)$. As in the insertions-only regime, we have that $\|x\|_1 = n$, setting $r = \frac{1}{\varepsilon}$ provides us with a (deterministic) $(\varepsilon, 0)$-approximation to the ℓ_1-point query problem that works in $O(\varepsilon^{-1} \log n)$ space.

3.2 Count-Min Sketch

Even though the Count algorithm presented above provides already a very satisfactory solution to ℓ_1-point query problem, we want to design now a different algorithm for that problem.

This algorithm will not be deterministic anymore (so, it will be incorrect with some non-zero probability) and it will have a slightly worse space complexity. However, its main advantage will be that it will work also in the turnstile model, i.e., also when elements can be deleted from the stream, not only inserted. This distinction is especially important in the context of ℓ_1-point query problem, as in turnstile model $\|x\|_1$ can be much smaller than n (and thus the resulting additive error can be appropriately smaller too).

Now, let $\varepsilon > 0$ by our desired accuracy parameter and $\delta > 0$ be the desired error probability bound. Consider the following algorithm:

Algorithm Count-Min:

1. Maintain $d = \log \frac{m}{\delta}$ arrays A_l of size r and d 2-wise independent hash functions $h_l : [m] \to [r]$, where $r = 2/\varepsilon$. Initially, all the entries of these arrays are zero.
2. In each step t, examine the t-th element y_t of the stream and for all $l \in [d]$
   (a) Increment $A_l[h_l(y_t)]$ by one, if y_t was inserted;
   (b) Decrement $A_l[h_l(y_t)]$ by one, if y_t was deleted.
3. At the end, output $\hat{x}_j = \min_l A_l[h_l(j)]$, for all $j \in [m]$.

Before we proceed with the analyze, we want to note that although this algorithm works in turnstile model, it requires the stream to be *well-formed*, i.e., that all the cumulative frequencies x_j are always non-negative. To get rid of this restriction, one just needs to increase d and r slightly (by a certain constant) and make all $\hat{x}_j$s correspond to the median of $A_l[h_l(j)]$s and not their minimum. (We will not cover the analysis of this modification here, but it is a quite straightforward extension of the ideas we will need to make the current version work. Also, cf. the analysis of the Count-Sketch algorithm below.)

It is not hard to see that the Count-Min algorithm can be implemented in $O(rd \log n) = O(\varepsilon^{-1} \log n \log \frac{m}{\delta})$ space. (Recall from the last lecture that a 2-wise independent hash function that takes values from a set of size T can be stored in $O(\log T)$ space.)

Now, to establish that indeed this algorithm delivers an (ε, δ)-approximation to the ℓ_1-point query problem, we prove the following lemma.

Lemma 1 *With probability at least $(1 - \delta)$, all $\hat{x}_j$s returned by Count-Min algorithm satisfy*

$$x_j \leq \hat{x}_j \leq x_j + \varepsilon \|x\|_1.$$

Proof Note that for any element $j \in [m]$ and $l \in [d]$, we have

$$A_l[h_l(j)] = \sum_{j' : h_l(j') = h_l(j)} x_{j'},$$

i.e., $A_l[h_l(j)]$ is the sum of frequencies of all the elements that have the same hash value as j with respect to the hash function h_l.

So, we have that, for each j,

$$\hat{x}_j = \min_l A_l[h_l(j)] = \min_l \sum_{j' : h_l(j') = h_l(j)} x_{j'} \geq \min_l x_j = x_j,$$

where we use the fact that all $x_{j'} \geq 0$, as the stream is well-formed.

Now, we claim that, for each $j \in [m]$ and $l \in [d]$, we also have that with probability at least $\frac{1}{2}$,

$$A_l[h_l(j)] \leq x_j + \varepsilon \|x\|_1.$$

Note that once we establish this claim, we our proof will be concluded. This is so, as we would have then that – due to the fact that all hash functions h_l are independent of each other – for any $j \in [m]$, with probability at least $1 - (\frac{1}{2})^d = 1 - \frac{\delta}{m}$, there is at least one $l^* \in [d]$ such that $A_{l^*}[h_{l^*}(j)] \leq x_j + \varepsilon \|x\|_1$ and therefore

$$\hat{x}_j = \min_l A_l[h_l(j)] \leq A_{l^*}[h_{l^*}(j)] \leq x_j + \varepsilon \|x\|_1.$$

So, by taking union bound over all m possible values of j, we get the desired approximation guarantee with probability at least $(1 - \delta)$.

To establish that remaining claim, let us fix some $j \in [m]$ and $l \in [d]$ and note that

$$A_l[h_l(j)] = \sum_{j':h_l(j')=h_l(j)} x_{j'} = x_j + \sum_{j' \neq j} x_{j'} Y_{j'},$$

where $Y_{j'}$ is a random variable that is 1 if $h_l(j') = h_l(j)$ and 0 otherwise.

By observing that, due to h_l being 2-wise independent hash function, we have that $Pr[Y_{j'} = 1] = 1/r$ for $j' \neq j$, we get

$$\mathbb{E}[A_l[h_l(j)]] = x_j + \frac{1}{r} \sum_{j' \neq j} x_{j'} \leq x_j + \frac{||x||_1}{r} = x_j + \frac{\varepsilon}{2}||x||_1.$$

Now, by Markov's inequality we can conclude that

$$Pr(A_l[h_l(j)] - x_j \geq \varepsilon||x||_1) \leq 1/2,$$

as desired. ■

4 ℓ_2-point Query Problem – Count-Sketch Algorithm

So far, we have seen two solutions to the ℓ_1-point query problem. How about ℓ_2-point query problem?

It turns out that a relatively simple extension of the idea behind the Count-Min algorithm can give us a solution for this problem (and thus better quality frequency estimates) too. Unfortunately, this will be at the expense of increased space complexity.

Consider the following algorithm (it works in turnstile model and even when the stream is not well-formed):

Algorithm Count-Sketch :

1. Maintain $d = \Theta(\log \frac{m}{\delta})$ arrays A_l of size $r = 4/\varepsilon^2$. Also, associate with each array two 2-wise independent hash functions $h_l : [m] \to [r]$ and $g_l : [m] \to \{-1, 1\}$. Initially, all entries of these arrays are zero.
2. In each step t, examine the t-th element y_t of the stream and for all $l \in [d]$
   (a) Add $g_l(y_t)$ to $A_l[h_l(y_t)]$ if y_t was inserted
   (b) Subtract $g_l(y_t)$ from $A_l[h_l(y_t)]$ if y_t was deleted
3. At the end, for each element $j \in [m]$, output $\hat{x}_j = \text{median}_{l \in [d]} \{g_l(j) A_l[h_l(j)]\}$.

It is easy to see that this algorithm can be implemented in $O(rd \log n) = O(\varepsilon^{-2} \log n \log \frac{m}{\delta})$ space. Now, to prove that it indeed provides us with an (ε, δ)-approximation to the ℓ_2-point query problem, we establish the following lemma.

Lemma 2 *With probability at least $(1 - \delta)$, all the estimates $\hat{x}_j$ output by the Count-Sketch algorithm satisfy*

$$x_j - \varepsilon||x||_2 \leq \hat{x}_j \leq x_j + \varepsilon||x||_2.$$

Proof We claim that for any fixed $j \in [m]$ and $l \in [d]$, the estimator $g_l(j) A_l[h_l(j)]$ is withing an additive $\varepsilon||x||_2$ error of x_j with probability at least 3/4.

Once this claim is established, the lemma will follow by first applying Chernoff bounds to show that, for a given $j \in [m]$, the probability that at least half of the estimates $g_l(j) A_l[h_l(j)]$s is by more than $\varepsilon||x||_2$ away from x_j, is at most $\frac{\delta}{m}$; and then using union bound over all m possible values of j.

To prove the needed claim, let us fix some $j \in [m]$ and $l \in [d]$ and note that

$$g_l(j)A_l[h_l(j)] = g_l(j) \sum_{j':h_l(j')=h_l(j)} g_l(j')x_{j'} = (g_l(j))^2 x_j + \sum_{j' \neq j} g_l(j)g_l(j')x_{j'}Y_{j'},$$

where $Y_{j'}$ is, again, a random variable that is equal to 1 if $h_l(j) = h_l(j')$ and 0 otherwise.

Now, if we examine the expected value of our estimator, we obtain

$$\mathbb{E}[g_l(j)A_l[h_l(j)]] = [g_l(j)]^2 x_j + \sum_{j' \neq j} \mathbb{E}[g_l(j)g_l(j')Y_{j'}]x_{j'} = x_j + \sum_{j' \neq j} \mathbb{E}[g_l(j)g_l(j')]\mathbb{E}[Y_{j'}]x_{j'} = x_j,$$

where we used the fact that the random variables $g_l(j)g_l(j')$ and $Y_{j'}$ are independent and that $\mathbb{E}[g_l(j)g_l(j')] = 0$ when $j \neq j'$ (we are using here the 2-wise independence of g_l and the fact that $\mathbb{E}[g_l(j')] = 0$).

So, the expectation of our estimator is a correct one. To establish its concentration around this expected value (and thus bound its probability of failure), we will compute its variance and then appeal to the Chebyshev's inequality. To this end, note that

$$\mathrm{V}ar[g_l(j)A_l[h_l(j)]] = \mathrm{V}ar[g_l(j)A_l[h_l(j)] - x_j],$$

as subtracting a constant term form a random variable does not change its variance.

Now, since the expectation of $g_l(j)A_l[h_l(j)] - x_j$ is zero, its variance is equal to the expectation of its square. Therefore, we have

$$\begin{aligned}
\mathrm{V}ar[g_l(j)A_l[h_l(j)]] &= \mathrm{V}ar[g_l(j)A_l[h_l(j)] - x_j] = \mathrm{V}ar\left[\sum_{j' \neq j} g_l(j)g_l(j')x_{j'}Y_{j'}\right] \\
&= \mathbb{E}[(\sum_{j' \neq j} g_l(j)g_l(j')x_{j'}Y_{j'})^2] \\
&= \sum_{j',j'' \neq j} \mathbb{E}[g_l(j')g_l(j'')Y_{j'}Y_{j''}]x_{j'}x_{j''}.
\end{aligned}$$

Note that, again, the random variables $g_l(j')g_l(j'')$ and $Y_{j'}Y_{j''}$ are independent and $\mathbb{E}[g_l(j')g_l(j'')] = 0$ when $j' \neq j''$. So, we have that $\mathbb{E}[g_l(j')g_l(j'')Y_{j'}Y_{j''}] = 0$ whenever $j' \neq j''$ and thus

$$\begin{aligned}
\mathrm{V}ar[g_l(j)A_l[h_l(j)]] &= \sum_{j',j'' \neq j} \mathbb{E}[g_l(j')g_l(j'')Y_{j'}Y_{j''}]x_{j'}x_{j''} \\
&= \sum_{j' \neq j} \mathbb{E}[Y_{j'}^2]x_{j'}^2 \leq \frac{\|x\|_2^2}{r},
\end{aligned}$$

as $j \neq j'$ and $\mathbb{E}[Y_{j'}^2] = \mathbb{E}[Y_{j'}] = \frac{1}{r}$ by 2-wise independence of h_l.

Now, applying Chebychev's inequality (see Theorem 2 in Lecture 17) to $g_l(j)A_l[h_l(j)]$ with $\gamma = \varepsilon\|x\|_2$, we get that

$$Pr[|g_l(j)A_l[h_l(j)] - x_j| \geq \varepsilon||x||_2] \leq \frac{\mathrm{V}ar[g_l(j)A_l[h_l(j)]]}{\varepsilon^2||x||_2^2} \leq \frac{1}{4}.$$

This completes the proof. ■