CS 388R: Randomized Algorithms Fall 2017

Lecture 17 — Nov. 2, 2017

Prof. Eric Price *Scribe: Daniel Liang, Josh Vekhter*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Overview

In this lecture, we consider the problem of maintaining a minimum count of the number of distinct elements in a set that we access in a streaming fashion (i.e. space poly log in the size of the set, we only get to see each element once).

More formally we denote this as the problem of estimating the L_0 norm of a set $\|x\|_0$. In particular, we will build to the problem of estimating this in the "turnstile model" where elements can be both inserted and deleted in the stream (but the total count of each element never passes below zero).

2 L_0 Sampling

2.1 Easy Problem

Suppose $\|x\|_0 = 1$ in the turnstile model. Can we find the non-zero index?

Solution: Track $a = \sum ix_i$ and $b = \sum x_i$. We note that if $\|x\|_0 = 1$ then $a = ix_i$ and $b = x_i$ where i is the index of the non-zero element. Thus we can output $i = \frac{a}{b}$.

2.2 Medium Problem

What if $\|x\|_0 = r$, can we sample uniformally from the non-zero indices?

Solution: Pick a random hash function $h : [n] \to [r]$. If h is a good hash function, we expect only one index to hash to zero. If we define $H = \{i \mid h(i) = 0\}$ then we can track $a = \sum_{i\in H} ix_i$ and $b = \sum_{i\in H} x_i$. We see that if $\|\text{Supp}(x) \cap H\|_0 = 1$ then $\frac{a}{b}$ is this single element in the intersection.

Of course, we aren't guaranteed that we only get one element in the intersection. For a fixed $i \in \text{Supp}(x)$ we want to find the probability that $h(i) = 0$ and the rest don't hash to zero so that our output is correct.

- If h is fully independent then we get $\frac{1}{r}\left(1 - \frac{1}{r}\right)^{r-1} \approx \frac{1}{er}$ as our probability for a given index. Thus the probability that we output anything is $\frac{1}{e}$.

- If $h : [n] \to [B]$ is pairwise independent then we can see that

$$\mathbb{P}\left[\text{output } i\right] \leq \frac{1}{B}$$

$$\begin{aligned}
\mathbb{P}\left[\text{output } i\right] &\geq \frac{1}{B}\mathbb{P}\left[\nexists j : h \in H \mid i \in H\right] \\
&= \frac{1}{B}\left(1 - \mathbb{P}\left[\exists j : h \in H \mid i \in H\right]\right) \\
&\geq \frac{1}{B}\left(1 - \sum_j \mathbb{P}\left[j \in H \mid i \in H\right]\right) \\
&= \frac{1}{B}\left(1 - \frac{r-1}{B}\right) \\
&\geq \frac{1}{B}\left(1 - \frac{r}{B}\right)
\end{aligned}$$

If we set $B = \frac{r}{\epsilon}$ then

$$(1-\epsilon)\frac{\epsilon}{r} \leq \mathbb{P}\left[\text{output } i\right] \leq \frac{\epsilon}{r}$$

With this in mind we can try $\mathcal{O}(\frac{1}{\epsilon}\log n)$ different pairwise-independent hash functions to output a value with high probability, since fully independent hash functions are costly.

2.3 Medium-Hard Problem

We aren't guaranteed to find an index for a given h, so how can we tell if our output is a correct value or not? That is, can we determine if $\|\text{Supp}(x) \cap H\|_0 = 1$ for a given hash function h?

Solution: Let us define

$$x'_i = \begin{cases} x_i & h(i) = 0 \\ 0 & h(i) \neq 0 \end{cases}$$

such that $\|x'\| = 1$ iff $\|\text{Supp}(x) \cap H\|_0 = 1$. Let us also define a new hash function $g : [n] \to \{0, 1\}$. If we define $G = \{i \mid g(i) = 0\}$ then we can track $c = \sum_{i \in G} x'_i$ and $d = \sum_{i \notin G} x'_i$ so that we know how many non-zero elements hashed to 0 and 1 respectively. We can see that if $\|x'\|_0 = 1$ then one of c or d will be non-zero and the other will be zero. On the other hand, if $\|x'\|_0 > 1$ then $\mathbb{P}\left[c \neq 0 \wedge d \neq 0\right] \geq \frac{1}{2}$. Thus, if we select $\mathcal{O}(\log n)$ different $g : [n] \to \{0, 1\}$ we succeed with high probability.

It turns out in the non-strict turnstile model, where the x_i can be negative, that if we define $S(i) = \{\pm 1\}$ as a 4-wise independent function that $c = \sum_{i \in G} x'_i S(i)$ and $d = \sum_{i \notin G} x'_i S(i)$ works with high probability as well. However, this means that we need to check that $x \neq \vec{0}$ first, which can be done in $\mathcal{O}(\log n)$

2.4 Hard Problem

What happens if we don't know r? Then we don't know what the image size of our hash function h should be to give us the best probability of successful output.

Solution: Try $\mathcal{O}(\log n)$ different r values: $1, 2, 4, 8, \cdots$ and use the r value that works the best. We don't take much of a hit to our output success probability since we will try an r value within a factor of 2 of the real r value.

Taken together the overall space complexity of this "hard" problem is:

$$\underbrace{\log n}_{\text{check if } x_{b,g}=0} \times \underbrace{\log n}_{\text{check if } \|x_n\|_1=1} \times \underbrace{\frac{1}{\epsilon}\log n}_{\text{output successfully w.h.p}} \times \underbrace{\log n}_{\text{try different } r \text{ values}} = O(\text{polylog}(n))$$

The state of the art result for $\|x\|_0$ estimation is $\mathcal{O}\left((\log n)^2\right)$.

3 Heavy Hitters

Now suppose we want to find the set of elements x_i that appear more than α fraction of the time (more formally, we'd like to find $S = \{i \in [n] \mid x_i \geq \alpha\|x\|_1\}$).

To make life a bit easier, we would actually like to find a "small" S' such that $S \subseteq S'$.

3.1 Insertion-Only model

Solution: First note that for a fixed α, there are at most $\frac{1}{\alpha}$ distinct heavy hitters. For this model, simply use reservoir sampling on s items, where $s \geq \frac{2}{\alpha}$ to give us an appropriate amount of leeway.

Now let us analyze the probability that a heavy hitter, i, is part of our output. Let $T = \{j \in [m] \mid v_j = i\}$ where v_j is the j-th element seen. For all $j \in T$ let Z_j be the indicator function for j being in the output sample, such that if $\sum Z_j \geq 1$ then v_j is in the output. We can see that $\mathbb{E}[Z_j] = \frac{s}{m}$ due to reservoir sampling, as $m = \|x\|_1$. Since we have a heavy hitter, $\mathbb{E}\left[\sum_j Z_j\right] \geq \alpha m \frac{s}{m} = \alpha s \geq 2$. Using the multiplicative Chernoff bound, we find that

$$\mathbb{P}\left[\sum_j Z_j < 1\right] \leq \mathbb{P}\left[\sum_j Z_j < \left(1 - \frac{1}{2}\right)\mathbb{E}\left[\sum_j Z_j\right]\right] \leq e^{-\frac{\frac{1}{4}\alpha s}{2}} = e^{-\frac{\alpha s}{8}}$$

If we set $s = \frac{1}{\alpha}\log n$ then i is in the output set with high probability.

3.2 Count-min sketch: Turnstiles with Insertions and Deletions

Solution: We initialize a bloom filter on $r = O(\log n)$ pairwise independent hash functions, each of which maps $h_r : [n] \to [B]$, where $|B| = O(\frac{1}{\alpha})$.

Because we do not allow entries to be negative, we can implement insertions by incrementing all of the buckets which get mapped to by each hash function, and deletions by decrementing each bucket.

In this framework, note that we can bound

$$x_i \leq \hat{x}_i = min_r Y_{r,h_r}(i)$$

where $Y_{r,y} = \sum_{i:h_r(i)=u} x_i$ is the number of items in each cell.

The next natural question to then ask is: how tight is this bound? To do this, we look at the expected number of collisions between elements in the bloom filter:

$$E[Y_{r,h_r}(i) - x_i] = E\left[\sum_{j \neq i} x_j \cdot \mathbb{1}_{h_r(j)=h_r(i)}\right] \leq \frac{\|x\|_1}{B}$$

By markov's inequality, we then have that $Y_{r,h_r}(i) - x_i \leq \frac{2\|x\|_1}{B}$ with probabilty $\frac{1}{2}$, which implies that $\hat{x}_i \leq x_i + \frac{2\|x\|_1}{B}$ w.h.p. ($1 - \frac{1}{2^r}$ by a union bound over the sum) - pretty tight! If we set $B = \frac{4}{\alpha}$ then w.h.p. $x_i \leq \hat{x}_i \leq x_i + \frac{\alpha}{2}\|x\|_1$. Our output is then

$$\{i \in [n] \mid \hat{x}_i \geq \alpha\|x\|_1\}$$

where $\|x\|_1$ is computed in parallel. Because $x_i \leq \hat{x}_i$ w.h.p we should output all heavy hitters w.h.p.

This leads naturally to related questions like attempting to count a median sketch, and counting a "correct" sketch in the L_2 sense instead of L_1. Another line of questioning which this leads to is problems where elements can take on negative edge weights, which is useful for implementing certain online graph algorithms to compute things like streaming flows and cuts. This line of inquiry is (w.h.p) outside the scope of this class.

References

[CM1] G. Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *Journal of Algorithms, 55(1):5875*, 2005.

[CM2] G. Cormode and S. Muthukrishnan. Whats hot and whats not: Tracking most frequent items dynamically. *In Proceedings of ACM Principles of Database Systems, pages 296-306, 2003.*