Nearly Optimal Distinct Elements and Heavy Hitters on Sliding Windows

Vladimir Braverman[2], Elena Grigorescu[3], Harry Lang[2], David P. Woodruff[1], Samson Zhou[3]

Carnegie Mellon University[1], Johns Hopkins University[2], Purdue University[3]

MODEL

Streaming Model: Elements of an underlying data set arrive sequentially

- What if we should not consider "old" elements?

Sliding Window Model: Most recent n updates form the underlying data set.

Caveat: We only consider insertion-only updates in the sliding window.

Open: What about more general forms of updates?

Problem 1 (Distinct Elements): Given a parameter $0 < \epsilon < 1$ and a set S of elements in $[m]$, give a $(1+\epsilon)$-approximation to the number of items i whose frequency f_i satisfies $f_i > 0$.

Applications: Network monitoring, data mining, query optimization

Problem 2 (Heavy Hitters): Given a parameter $0 < \epsilon < 1$ and a set S of elements in $[m]$, output all items i whose frequency f_i satisfies $f_i > \epsilon (F_p)^{1/p}$ and no item i whose frequency f_i satisfies $f_i < (\epsilon - \phi)(F_p)^{1/p}$, where $F_p = \sum_{\{i=1\}}^m f_i^p$ and $\phi = c\epsilon$ for some constant $c < 1$.

Applications: Network monitoring, denial-of-service prevention, moment estimation, L_p sampling

RESULTS (Distinct Elements)

Theorem 1: Given $\epsilon > 0$, there exists an algorithm that, with probability at least $\frac{2}{3}$, provides a $(1+\epsilon)$-approximation to the number of distinct elements in the sliding window model, with space complexity (in bits) $O\left(\frac{1}{\epsilon^2}\log n \log\frac{1}{\epsilon}\log\log n + \frac{1}{\epsilon}\log^2 n\right)$.

Theorem 2: Let $0 < \epsilon < \frac{1}{\sqrt{n}}$. Any one-pass streaming algorithm that gives a $(1+\epsilon)$-approximation to the number of distinct elements in the sliding window model with probability at least $\frac{2}{3}$ requires space complexity $\Omega\left(\frac{1}{\epsilon^2}\log n \log\frac{1}{\epsilon}\log\log n + \frac{1}{\epsilon}\log^2 n\right)$.

Upper Bound	Lower Bound
$O\left(\frac{1}{\epsilon^3}\log^2 n + \frac{1}{\epsilon}\log^3 n\right)$ [KNW10, BO07]	$\Omega\left(\frac{1}{\epsilon^2} + \log n\right)$ [AMS99]
$O\left(\frac{1}{\epsilon^2}\log n\left(\log\log n \log\frac{1}{\epsilon}\right) + \frac{1}{\epsilon}\log^2 n\right)$	$\Omega\left(\frac{1}{\epsilon^2}\log n + \frac{1}{\epsilon}\log^2 n\right)$

RESULTS (Heavy Hitters)

Theorem 3: Given $\epsilon > 0$ and $0 < p \le 2$, there exists an algorithm that, with probability at least $\frac{2}{3}$, outputs all indices $i \in [m]$ for which $f_i \ge \epsilon (F_p)^{1/p}$, and no indices $i \in [m]$ for which $f_i \le \frac{\epsilon}{12}(F_p)^{1/p}$. The algorithm uses $O\left(\frac{1}{\epsilon^p}\log^2 n\left(\log^2\log n + \log\frac{1}{\epsilon}\right)\right)$ bits of space.

Theorem 4: Let $p > 0$ and $\epsilon > 0$. Any one pass streaming algorithm that finds the L_p-heavy hitters in the sliding window model with probability at least $\frac{2}{3}$ uses space $\Omega\left(\frac{1}{\epsilon^p}\log^2 n\right)$.

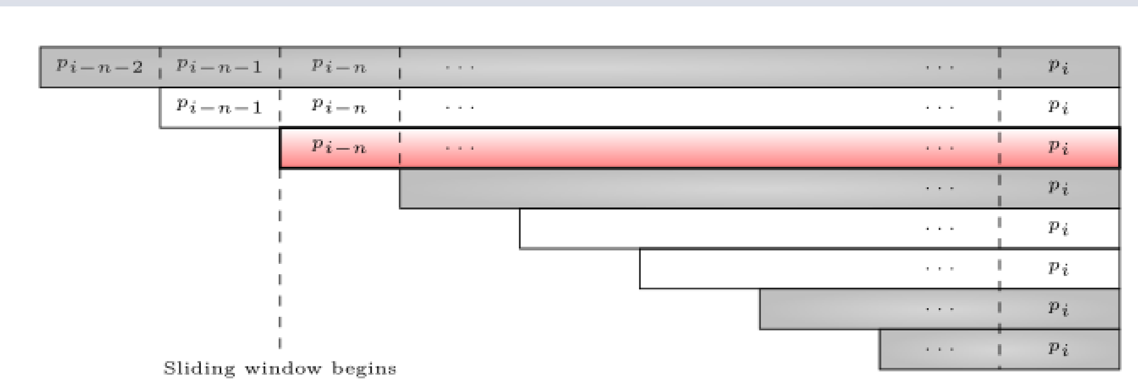
Upper Bound	Lower Bound
$O\left(\frac{1}{\epsilon^4}\log^3 n\right)$ [BGO14]	$\Omega\left(\frac{1}{\epsilon^2}\log n\right)$ [JST11]
$O\left(\frac{1}{\epsilon^2}\log^2 n\left(\log\log n + \log\frac{1}{\epsilon}\right)\right)$	$\Omega\left(\frac{1}{\epsilon^2}\log^2 n\right)$

HISTOGRAMS

Definition: A function f is (α, β)-smooth if

1. (Monotonicity) $f(A) \ge f(B)$ for B subset of A.
2. (Polynomially Bounded) $f(A) \le n^c$ for all A.
3. (Smoothness) There exists $\alpha \in (0,1)$ and $\beta \in (0, \alpha]$ so that if B is a subset of A and $(1-\beta)f(A) \le f(B)$, then $(1-\alpha)f(A \cup C) \le f(B \cup C)$ for any adjacent C.

Fact: L_p norm is $\left(\epsilon, \frac{\epsilon^p}{p}\right)$-smooth and L_0 is (ϵ, ϵ)-smooth



Framework by [BO07] for converting insertion-only streaming algorithms to sliding window algorithms for smooth functions.

Intuition:

1. Maintain algorithms for substreams where the output jumps by a factor of $(1+\epsilon)$.
2. Delete "old" algorithms, so that there is never more than one algorithm containing expired points.
3. Delete algorithms whose substreams are "too close" to each other. It the outputs of the algorithms are close, we don't need one of them!

Only need a logarithmic number of algorithms!

Caveat: We need correctness over all algorithms and all elements in the sliding window.

UPPER BOUND (Distinct Elements)

Given a hash function $h: [m] \to \{0,1\}^{\log m}$, let $S_k = \{s \in S \mid \text{lsb}(h(s)) \ge k\}$. Note that $2^k |S_k|$ is an unbiased estimator for S.

Balls into bins: Fill up a $\log n$ by $\frac{100}{\epsilon^2}$ table T. Given a hash function $h_2: [m] \to \frac{100}{\epsilon^2}$, set $T(i,j) = 0$ if $h_2(s) \ne j$ for all $s \in S_i$. Subsampling with probability $\frac{1}{2^i}$. Look at a row for which $\mathrm{E}[S_k] = \Theta\left(\frac{1}{\epsilon^2}\right)$ for number of distinct elements.

Idea: Instead of keeping a table for each instance, keep ONE table which encodes all of the tables!

Each cell stores ID of the first nonzero instance: $O\left(\frac{1}{\epsilon^2}\log n\right)$ cells, each $O\left(\log\log n + \log\frac{1}{\epsilon}\right)$

Each column in the table is monotonic, can further compress! Encode each column using $O\left(\log n \log\frac{1}{\epsilon}\right)$ bits.

1	1	1	1	2	2
3	1	3	1	2	3
4	2	3	1	3	4
4	3	3	1		
4		4	1		
			1		
			4		

Caveat: Need $O(\log\log n)$ instances for union bound.

UPPER BOUND (Heavy Hitters)

Estimator [BCINWW17]: Provides a $(1+\epsilon)$-approximation to L_2-norm using space complexity (in bits) $O\left(\frac{1}{\epsilon^2}\log n\left(\log\log n + \log\frac{1}{\epsilon}\right)\right)$.

BPTree [BCINWW17]: Returns a set of $\frac{\epsilon}{2}$ heavy hitters containing every ϵ heavy hitter using space complexity $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}\log n\right)$.

Already works in framework by [BO07] but space dependency is $\frac{1}{\epsilon^4}$. Instead use the following ideas:

1. Maintain a 2-approximation to the L_2-norm using Estimator.
2. BPTree returns a set of $\frac{\epsilon}{2}$ heavy hitters.

Problem: Reported heavy hitters may be before sliding window begins!

SmoothCounter: Provides a $(1+\epsilon)$-approximation to the frequency of a particular element in the sliding window using $O\left(\frac{1}{\epsilon}\log^2 n\right)$ bits of space.

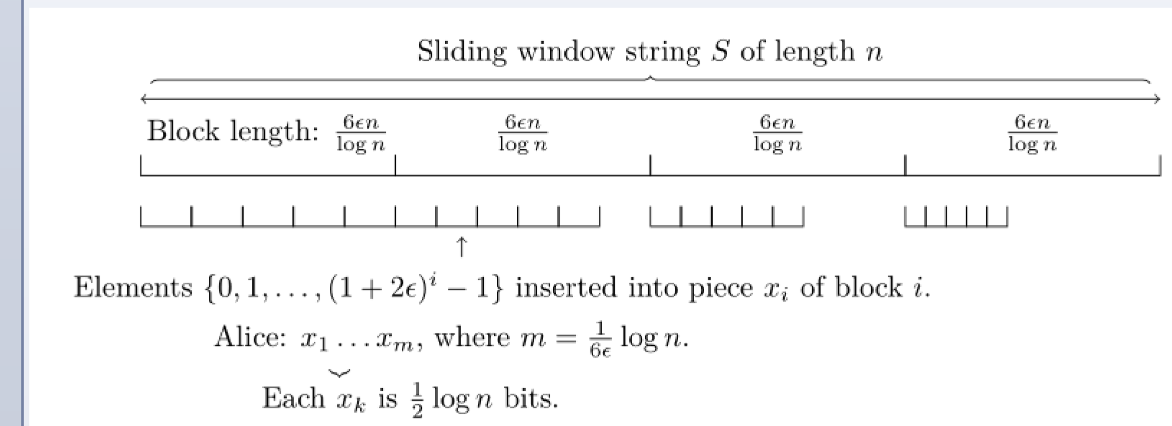
3. Keep a 2-approximation to the frequency of the reported heavy hitters in the sliding window using SmoothCounter.

Caveat: Need additional tricks to show that a union bound over $O(\log n)$ instances suffices.

LOWER BOUND (Distinct Elements)

Lower bound of $\Omega\left(\frac{1}{\epsilon}\log^2 n\right)$ from IndexGreater:

Alice has $S = x_1 x_2 \ldots x_m$, each x_i has n bits. Bob is given $i \in [m]$ and $j \in [2^n]$ and must determine if $x_i > j$.



Lower bound of $\Omega\left(\frac{1}{\epsilon^2}\log n\right)$ from GapHamming:

Alice has x and Bob has y, each binary of length n, and must determine $\mathrm{HAM}(x,y) \ge \frac{n}{2} + \sqrt{n}$ or $\mathrm{HAM}(x,y) \le \frac{n}{2} - \sqrt{n}$.

Construction: $\Omega(\log n)$ instances of GapHamming, each requires space $\Omega\left(\frac{1}{\epsilon^2}\right)$ for $\epsilon \le \frac{1}{\sqrt{n}}$.

Alice maintains a counter p for overall position and inserts p if the corresponding bit of x_i is a one. Alice passes the state of the algorithm to Bob, who does the same thing for y_i.

LOWER BOUND (Heavy Hitters)

AugmentedIndex: Alice is given $S \in [k]^n$ and Bob is given $i \in [n]$ as well as $S[1, i-1]$. Bob must output $S[i]$.

Construction: Let $a = \frac{1}{2^p \epsilon^p}\log\sqrt{n}$ and $b = \log n$ and $S = [2^a]^b$. Each $S[i]$ is a bits, which can be written as $w_1 w_2 \ldots w_t$, where $t = \frac{1}{2^p \epsilon^p}$ and each w_j has $\log\sqrt{n}$ bits.

Alice makes each w_j a heavy hitter by having geometrically decreasing frequency for each $S[i]$ and passes the state of the algorithm to Bob, who expires everything in the stream before $S[i]$ and computing the heavy hitters.

REFERENCES

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. J. Comput. Syst. Sci., 1999.

[BCINWW17] Vladimir Braverman, Stephen R. Chestnut, Nikita Ivkin, Jelani Nelson, Zhengyu Wang, and David P. Woodruff. Bptree: An L2 heavy hitters algorithm using constant memory. PODS 2017.

[BGO13] Vladimir Braverman, Ran Gelles, and Rafail Ostrovsky. How to catch L2 -heavy-hitters on sliding windows. COCOON 2013.

[BO07] Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. FOCS 2007.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. PODS 2010.

[JST11] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for Lp samplers, finding duplicates in streams, and related problems. PODS 2011.