CS 15-851: Algorithms for Big Data Spring 2025

Lecture 17 — March 20

Prof. David Woodruff *Scribe: Kefan Cao*

1 Estimating Heavy Hitters in a Stream

1.1 l_1 and l_2-Heavy Hitters

Definition. l_1-guarantee:

- Output a set cibtaububg all items j for whih $|x_j| \geq \phi|x|_1$.
- The set should not contain any j with $|x_j| \leq (\phi - \epsilon)|x|_1$.

Definition. l_2-guarantee:

- Output a set cibtaububg all items j for whih $x_j^2 \geq \phi|x|_2^2$.
- The set should not contain any j with $x_j^2 \leq (\phi - \epsilon)|x|_2^2$.
- The l_2-guarantee is much stronger than the l_1-guarantee.

If $|x_j| \geq \phi|x|_1$, then $x_j^2 \geq \phi^2|x|_1^2 \geq \phi^2|x|_2^2$ by squaring both sides.

For example, in a stream $x = (\sqrt{n}, 1, 1, 1, \ldots, 1)$, the first item is a l_2 heavy hitter for constant ϕ, ϵ, but not a l_1 heavy hitter.

1.2 An Deterministic Algorithm to Find i

Suppose you are promised at the end of the stream, $x_i = n$, and $x_j \in \{0, 1\}$ for $j \in \{1, 2, \ldots, n\}$ with $j \neq i$.

Let $A_i \subset [n]$ be the set of indices with $x_j = 0$, and B_i be the set of indices with $x_j = 1$.

For each j in $\{1, 2, 3, \ldots, \log n\}$, let $A_j \subset [n]$ be the set of indices with j-th bit in their binary representation equal to 0, and B_j be the set with j-th bit equal to 1.

Compute $a_j = \sum_{i \in A_j} x_i$ and $b_j = \sum_{i \in B_j} x_i$ for each j in $\{1, 2, \ldots, \log n\}$, and then we can read off the bianry representation of i. Memory used: $O(\log^2 n)$ bits.

$$\begin{bmatrix} \underline{a_1} & \underline{a_2} & \cdots & \underline{a_{\log n}} \\ \underline{b_1} & \underline{b_2} & \cdots & \underline{b_{\log n}} \end{bmatrix}$$

Memory used: $O(\log^2 n)$ bits.

Instead, now suppose you are promised at the end of the stream, $x_i = 100\sqrt{n \log n}$, and $x_j \in \{0, 1\}$ for $j \in \{1, 2, \ldots, n\}$ with $j \neq i$. Then this might not work because the sums can be larger than the sum with x_i. A quick solution is to use use countsketch (random signed sum).

The additive Chernoff bound implies magnitude of noise in a count is at most $\sqrt{n \log(n)}$ w.h.p.

Remark 1. The l_2-heavey hitter is a set instead of 1 element. We could possibly have $1/\phi$ l_2 heavy hitters.

Now remove the assumptions about x's.

1.3 CountSketch Achieves l_2-guarantee

CountSketch does a random signed sum.

Assign each coordinate i a random sign $\sigma_i \in \{-1, 1\}$. If we add $\leq n$ random signs, $\sigma_1 + \sigma_2 + \ldots + \sigma_n$, the variance will be $O(n)$. The standard deviation will be $O(\sqrt{n})$ with constant probability. The sum will be $\Theta(\sqrt{n \log n})$ with high probability (1-1/n).

Randomly partition coordinates into B buckets, maintain $c_j = \sum_{i:h(i)=j} x_i \cdot \sigma_i$ in the j-th bucket.

$$\sigma_i c_{h(i)} = \sigma_i \sum_{j,h(j)=h(i)} x_j \cdot \sigma_j = \sigma_i \sigma_i x_i + \sigma_i \sum_{j \neq i, h(j)=h(i)} x_j \cdot \sigma_j$$

The expectation of the first term is x_i and the expectation for the second part is 0. Thus,

$$\mathbb{E}[\sigma_i c_{h(i)}] = x_i$$

Suppose we independently repeat this hashing scheme $O(\log n)$ times and output the median of the estimates across the $\log n$ repetitions. The noise in a bucket is $\sigma_i \cdot \sum_{i' \neq i, h(i')=h(i)} \sigma_{i'} \cdot x_{i'}$. The expectation of the noise is 0. The variance of the noise is:

$$\begin{aligned}
\mathbb{E}\left[\sigma_i \sum_{i' \neq i, h(i')=h(i)} \sigma_{i'} x_{i'}\right]^2 &= \mathbb{E}\left[\sum_{i' \neq i} \delta(h(i') = h(i)) \cdot \sigma_{i'} x_{i'}\right]^2 \\
&= \mathbb{E}\left[\sum_{i' \neq i, i'' \neq i} \delta(h(i') = h(i)) \cdot \delta(h(i'') = h(i)) \cdot \sigma_{i'} \cdot \sigma_{i''} \cdot x_{i'} \cdot x_{i''}\right] \\
&= \sum_{i' \neq i} \mathbb{E}\left[\delta(h(i') = h(i)) \cdot x_{i'}^2\right] \\
&\leq \frac{|x|_2^2}{B}
\end{aligned}$$

So with constant probability, the noise in a bucket is $O\left(\frac{|x|_2}{\sqrt{B}}\right)$.

Since the $\log n$ repetitions are independent, this ensures that our estimate $\sigma_i c_{h(i)}$ will equal $x_i \pm O\left(\frac{|x|_2}{\sqrt{B}}\right)$ with high probability $\left(1 - \frac{1}{poly(n)}\right)$. This is because in order for a median to be bad, we need at least half of the repetitions to be bad.

We can use a union bound for every i. Hence we can approximate every x_i simultaneously up to additive error $O\left(\frac{|x|_2}{\sqrt{B}}\right)$.

Remark 2. Memory used: $O(B \log^2 n)$ bits.

1.4 Tail Guarantee

CountSketch approximates every x_i simultaneously up to additive error $O\left(\frac{|x|_2}{\sqrt{B}}\right)$. Then how should we choose B?

We want to find all $x_i^2 \geq \phi |x|_2^2$ and no $x_i^2 \leq (\phi - \epsilon)|x|_2^2$. Our goal is to estimate x_i up to additive error $(\sqrt{\phi} - \sqrt{\phi - \epsilon})|x|_2 \approx \frac{|x|_2}{\sqrt{B}}$.

Then we want $\sqrt{\phi} - \sqrt{\phi - \epsilon} = \frac{1}{\sqrt{B}}$. For example, here we set $\epsilon = \phi/2$, then $\sqrt{\phi} - \sqrt{\phi - \epsilon} = \frac{1}{\sqrt{B}}$, $O(\sqrt{\phi}) = \frac{1}{\sqrt{B}}$, $B = O(1/\phi)$.

Remark 3. The memory used now is $O(\log^2 n/\phi)$ bits.

Claim 1. Tail Guarantee: CountSketch approximates every x_i simultaneously up to additive error $O\left(\frac{|x_{-B/4}|_2}{\sqrt{B}}\right)$ where $x_{-B/4}$ is x after zero-ing out its top $B/4$ coordinates in magnitude.

Proof. Let j be one of the top $B/4$ items. The probability that j collides with a specific item is $1/B$. Using a union bound, the probability that there exists a coordinate j among the top $B/4$ items such that j collides with i is bounded by $\leq B/4 \cdot 1/B = 1/4$.

Thus with probability at least $3/4$, in each repetition the top $B/4$ coordinates of x in magnitude do not land in the same hash bucket as x_i.

Given that we do not collide with the top $B/4$ coordinates, the noise in the bucket is $O\left(\frac{|x_{-B/4}|_2}{\sqrt{B}}\right)$. ■

Remark 4. If we know x is $B/4$-sparse, then we can perfectly recover x. This is another algorithm to recover sparse vector.

1.5 Deterministic l_1-Heavy Hitters

Definition. An $s \times n$ matrix S is ϵ-incoherent if:

- for all columns S_i, $|S_i|_2 = 1$.
- for all pairs of columns S_i and S_j, $|\langle S_i, S_j \rangle| \leq \epsilon$.
- Entries can be specified with $O(\log n)$ bits of space.

Can compute Sx in a stream using $O(s \log n)$ bits of space.

Estimate $\hat{x_i}$:

$$\begin{aligned}\hat{x_i} =& \sum_{j=1}^{n} \langle S_i, S_j \rangle x_j \\ =& \langle S_i, S_i \rangle x_i + \sum_{j=1, j \neq i}^{n} \langle S_i, S_j \rangle x_j \\ =& |S_i|_2^2 x_i \pm \max_{i,j} |\langle S_i, S_j \rangle| |x|_1 \\ =& x_i \pm \epsilon |x|_1\end{aligned}$$

1.6 A Simple Construction of ϵ-Incoherent Matrices

Consider a prime $q = \Theta((\log n)/\epsilon)$. Let $d = \epsilon \cdot q = O(\log n)$.

Consider n distinct non-zero polynomials $p_1, \ldots, p_n$, each of degree less than d:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_d x^d$$

Consider polynomials mod q.

The number of distinct polynomials is q^d. We want $q^d - 1 > n$.

Then we want to associate the i-th column of S with p_i.

S has q^2 rows and n columns. We can partition S into q groups with q rows in each group. We can associate the i with column with polynomial p_i. In the j-th group and i-th column, we put a 1 on $p_i(j)$ and 0 elsewhere.

Each column has norm $|S_i|_2 = 1$.

The dot product between any pair of distinct columns is bounded by d (because there are at most d intersections for two polynomials of degree $d = \epsilon q$). So $|\langle S_i, S_j \rangle| \leq \epsilon$.