CS 229r: Algorithms for Big Data Fall 2013

Lecture 5 — September 17, 2013

Prof. Jelani Nelson *Scribe: Cameron Musco*

1 Overview

In this lecture we will continue going over turnstyle streams. The set up here is a vector $\boldsymbol{x} \in \mathbb{R}^n$ initialized to $\mathbf{0}$ and receiving updates of the form $x_i \leftarrow x_i + v$ where we can have $v \leq 0$ or $v > 0$. We will consider three algorithms under using this model:

1. Point Query: given i output $x_i \pm error$
2. Heavy Hitters: output/estimate $\phi - HH^p = \{i : |x_i|^p \geq \phi \|\boldsymbol{x}\|_p^p\}$
3. Sparse Approximation: recover $\tilde{\boldsymbol{x}}$ sparse such that $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|$is small.

We will use 2 algorithms to solve all these problems: Count Min Sketch and Count Sketch

2 Count Min Sketch

An algorithm for Point Query first given in: [2] by Cormode and Muthukrishnan.

Algorithm:

- Maintain at $t \times w$ matrix of counters. For each of our t rows of counters, we have an associated hash function $h_i : [n] \to [w]$. $h_1, ...h_t$ are chosen independently at random from a 2-wise independent family.
- Upon receiving the increment of value v to index i, hash i using each of our t hash functions. Add v to the counter $C_{j,h_j(i)}$ for each $j \in [t]$.
- Output $PointQuery(i) = \min_{j \in [t]} C_{j,H_j(i)}$

For the analysis, we assume that $\forall i x_i \geq 0$. That is, although we can have negative values of v, none of the counters ever drops below 0. This is also known as the **strict turnstile assumption**.

Claim 1. *If $t \geq \log_2(\frac{1}{\delta})$ and $w \geq \frac{2}{\epsilon}$ then $\mathbb{P}(PointQuery(i) \in [x_i - \epsilon\|\boldsymbol{x}\|_1, x_i + \epsilon\|\boldsymbol{x}\|_1]) \geq 1 - \delta$*

Proof. For any $j \in [m]$,

$$\begin{aligned} C_{j,H_j(i)} &= x_i + \sum_{r:h_j(r)=h_j(i), r\neq i} x_r \\ &= x_i + \underbrace{\sum_{r\neq i} \delta_r x_r}_{noise} \end{aligned}$$

where δ_r is the indicator function with value 1 if $h_j(r) = h_j(i)$, 0 otherwise.

Using the fact that our hash functions come from a 2-wise independent family we have:

$$\mathbb{E}\sum_{r\neq i}\delta_r x_r = \frac{\sum_{r\neq i} x_r}{w} \leq \frac{\epsilon}{2}\|\boldsymbol{x}\|_1$$

Applying Markov's Inequality (and using the assumption that each x_i is nonnegative) gives:

$$\mathbb{P}(noise > \epsilon\|\boldsymbol{x}\|_1) \leq \frac{1}{2}$$

So, $C_{j,H_j(i)} \geq x_i$ and with probability $> 1/2$, $C_{j,H_j(i)} \leq \epsilon\|\boldsymbol{x}\|_1$

Since we are repeating $t = \log_2(\frac{1}{\delta})$ times,

$$\begin{aligned}
\mathbb{P}(\min_{j\in[t]} C_{j,H_j(i)} > x_i + \epsilon\|\boldsymbol{x}\|_1) &= \mathbb{P}(\forall j \in [t],\ C_{j,H_j(i)} > \epsilon\|\boldsymbol{x}\|_1) \\
&< \frac{1}{2^t} \\
&< \delta
\end{aligned}$$

□

Note: The error guarantee is only really meaningful if $x_i > \epsilon\|\boldsymbol{x}\|$, so only really meaningful for $\frac{1}{\epsilon}$ of the values in $\boldsymbol{x}$.

Note 2: If we throw out the strict turnstile assumption and let the x_i's be negative, we can use a similar algorithm except output the median of our t counters for x_i. Setting w to something like $\frac{\epsilon}{3}$, lets us use Markov's to bound the noise to be less than $\epsilon\|\boldsymbol{x}\|_1$ with probabilitiy $> 2/3$. We can then apply the Chernoff bound to show that our median will fall within ϵ error with high probability.

3 Heavy Hitters - with Count Min

Definition 2. $\phi - HH^1 = \{i : |x_i| \geq \phi\|\boldsymbol{x}\|_1\}$

Goal: Output a list $L \subseteq [n]$ such that

- $\phi - HH^1 \subseteq L$
- if $i \in L$, $i \in \frac{\phi}{2} - HH^1$

Easy but slow to compute L algorithm:

- Use Count Min, setting $\delta < \frac{\gamma}{n}$ and $\epsilon = \phi/4$. Run $PointQuery(i)$ for each i, and add i to L if $PointQuery(i) > \frac{3}{4}\phi\|\boldsymbol{x}\|_1$. (Can get $\|\boldsymbol{x}\|_1$ simply by summing one of the rows of counters)

Claim 3. *Algorithm satisfies the goal conditions with probability* $> 1 - \gamma$

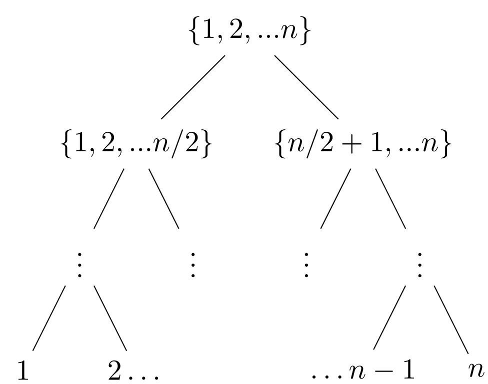
We add all actual ϕ heavy hitters to L and only add a $< \phi/2$ heavy hitter with probability at most $\delta = \frac{\gamma}{n}$ So, by a union bound, with our n point queries we only add a less than $\frac{\phi}{2}$ heavy hitter with probability at most γ

Space: $t = \log_2(\frac{1}{\delta}) = \log(\frac{n}{\gamma})$ and $w = \frac{2}{e} = O(\frac{1}{\phi})$ so our total space (the size of our counter matrix is $\boldsymbol{O(\frac{\log(n/\gamma)}{\phi})}$

Time: The downside. Runtime to output L is $\boldsymbol{\Theta(n \log n)}$

Faster ϕ-HH algorithm:

Create a perfect binary tree using our n vector elements as the leaves.



Define I^J to be the partition of $[n]$ into buckets of size 2^j: $\{\{1, 2, ...2^j\}, \{2^j + 1, 2^j + 2, ...2^{j+1}\}, ...\}$. At the j^{th} row of our binary tree where $j \in [0, \ldots, \log_2(n)]$ we have $n/2^j$ buckets. We can view these as forming a vector $x^j \in \mathbb{R}^{n/2^j}$ where

$$(x^j)_i = \sum_{r \in i^{th} \ partition \ of \ I^j} x_r$$

Now our algorithm is:

- Run Count Min Sketch $\log_2(n) + 1$ times - once on each vector x^j, where $j \in [0, ... \log_2(n) + 1]$. Run with error $\epsilon = \frac{\phi}{4}$ and $\delta = \frac{\gamma\phi}{\log(n)}$
- Move down the tree starting from the root. For each node, run $PointQuery$ for each of its two children. If a child is a heavy hitter, i.e. $PointQuery$ returns $\geq \frac{3}{4}\phi\|x\|_1$, continue moving down that branch of the tree.
- Add to L any leaf of the tree that you point query and that has $PointQuery(i) \geq \frac{3}{4}\phi\|x\|_1$.

Correctness: If a leaf is a heavy hitter, then all of its ancestors must also be heavy hitters. So we will eventually point query every leaf that is a heavy hitter and at it to L. On each level of the tree we can have only $O(\frac{1}{\phi})$ heavy hitters. So we make $O(\frac{\log(n)}{\phi})$ point queries total. Again using a union bound, we have a $< \delta = \frac{\gamma\phi}{\log(n)}$ chance of failing on each of these queries so a $< \gamma$ chance of failing at all.

Time to Recover L: We improved from n point queries to $\log(n)/\phi$ point queries. The total time is the number of point queries times $t = \log(\frac{1}{d})$. So the total time is: $O\left(\frac{\log(n)}{\phi} * \log\left(\frac{\log(n)}{\phi\gamma}\right)\right)$

Space: $O(\frac{1}{\epsilon}\log(\frac{1}{\delta}) * \log(n)) = O\left(\frac{\log(n)}{\phi} * [\log(\frac{1}{\phi\gamma}) + \log\log(n)]\right)$

Note: Why do we have to use more space to make recovering L faster? Jelani doesn't know. Possible final project idea.

4 Sparse Approximation

Goal: Recover k-sparse $\tilde{x} \in \mathbb{R}^n$ such that $\|x - \tilde{x}\|_\infty \le \alpha \|x_{tail(k)}\|_1$ where $\alpha > 1$.

Definition 4. $x_{tail(k)}$ *is* x *but with the heaviest* k *coordinates in magnitude zero'd out.*

Claim 5. *PointQuery on Count Min with* $\delta = \frac{1}{\gamma n}$ *and* $w = O(k)$ *works to solve* k*-sparse recovery.*

Proof. Define L to be the top k coordinates in x by magnitude. $L \subseteq [n]$, and $|L| = k$.

$$C_{j,H_j(i)} = x_i + \sum_{r \in L, r \neq i} x_r \delta_r + \sum_{r \notin L, r \neq i} x_r \delta_r$$

We can bound the error arising from $r \notin k$ as before. $\mathbb{E}(error) = \frac{\|x_{tail(k)}\|_1}{w}$. And if $w = O(\frac{ck}{\alpha})$ we expect something like α collisions with heavy elements in L. With big enough c, by Markov's inequality, we are very likely not to have a collision at all. So, with high probability, our error on each element is $O(\frac{\|x_{tail(k)}\|_1}{w}) = O(\frac{\|x_{tail(k)}\|_1}{k})$, giving us the guarantee we were looking for.

5 Count Sketch

Given in [1].

Basically, keep a table of counters as in Count Min Sketch. With each associated row $i \in [t]$ we have a hash function $h_i : [n] \to [w]$ as before. We also have a hash function $\sigma_i : [n] \to \{-1, 1\}$, with each σ chosen independently at random.

$C_{i,j} = \sum_{r:h_i(r)=j} \sigma_i(r) * x_r$

Basically, doing a similar analysis to problem 3 of Pset 1, we can show that $C^2_{j,h_j(i)} = x_i^2 + noise$ and can bound the noise and show that taking the medians of the counters gives good estimates with high probability.

$\square$

References

[1] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, January 2004.

[2] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.