1 Overview

In this lecture we will continue going over turnstile streams. The setup here is a vector $x \in \mathbb{R}^n$ initialized to 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 $x_i \pm \text{error}$
2. Heavy Hitters: output/estimate $\phi - HH_p = \{i : |x_i|^p \geq \phi \|x\|_p^p\}$
3. Sparse Approximation: recover $\tilde{x}$ sparse such that $\|x - \tilde{x}\|$ is small.

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

2 Count Min Sketch


Algorithm:

- Maintain at $t \times w$ matrix of counters. For each of our $t$ rows of counters, we have an associated hash function $h_i : [n] \rightarrow [w]$. $h_1, \ldots, h_t$ 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 $\text{PointQuery}(i) = \min_{j \in [t]} C_{j,h_j(i)}$

For the analysis, we assume that $\forall ix_i \geq 0$. 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 $\mathbb{P}(\text{PointQuery}(i) \in [x_i - \epsilon\|x\|_1, x_i + \epsilon\|x\|_1]) \geq 1 - \delta$

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

$$C_{j,H_j(i)} = x_i + \sum_{r : h_j(r) = h_j(i), r \neq i} x_r$$

$$= x_i + \sum_{r \neq i} \delta_r x_r$$

where $\delta_r \sim \text{Noise}$. 

\[1\]
where $\delta_r$ 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:

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

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

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

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

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

$$
P(\min_{j \in [t]} C_{j,h_j(i)} > x_i + \epsilon \|x\|_1) = P(\forall j \in [t], C_{j,h_j(i)} > \epsilon \|x\|_1)
< \frac{1}{2^t}
< \delta
$$

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

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 $\frac{1}{\epsilon}$, lets us use Markov’s to bound the noise to be less than $\epsilon \|x\|_1$ with probability $> 2/3$. We can then apply the Chernoff bound to show that our median will fall within $\epsilon$ error with high probability.

3 Heavy Hitters - with Count Min

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

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

- $\phi - HH^1 \subseteq L$
- if $i \in L$, $i \in \phi - HH^1$

Easy but slow to compute $L$ algorithm:

- Use Count Min, setting $\delta < \frac{2}{n}$ and $\epsilon = \phi/4$. Run $PointQuery(i)$ for each $i$, and add $i$ to $L$ if $PointQuery(i) > \frac{3}{4} \phi \|x\|_1$. (Can get $\|x\|_1$ simply by summing one of the rows of counters)
Claim 3. Algorithm satisfies the goal conditions with probability \( > 1 - \gamma \)

We add all actual \( \phi \) 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 \( \phi/2 \) heavy hitter with probability at most \( \gamma \).

**Space:** \( t = \log_2\left(\frac{1}{\delta}\right) = \log\left(\frac{n}{\gamma}\right) \) and \( w = \frac{2}{\epsilon} = O\left(\frac{1}{\epsilon}\right) \) so our total space (the size of our counter matrix is \( \Theta\left(\frac{\log(n/\gamma)}{\phi}\right) \).

**Time:** The downside. Runtime to output \( L \) is \( \Theta(n \log n) \).

**Faster \( \phi \)-HH algorithm:**

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

\[
\begin{array}{c}
\{1, 2, \ldots, n\} \\
\{1, 2, \ldots, n/2\} & \{n/2 + 1, \ldots, n\} \\
\vdots & \vdots & \vdots & \vdots \\
1 & 2 & \ldots & n - 1 & n \\
\end{array}
\]

Define \( I^j \) to be the partition of \([n]\) into buckets of size \( 2^j \): \( \{1, 2, \ldots, 2^j\}, \{2^j + 1, 2^j + 2, \ldots, 2^{j+1}\}, \ldots \).

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

\[
(x^j)_i = \sum_{r \in \text{partition of } I^j} x_r
\]

Now our algorithm is:

- Run Count Min Sketch \( \log_2(n) + 1 \) times - once on each vector \( x^j \), where \( j \in [0, \ldots, \log_2(n) + 1] \). Run with error \( \epsilon = \frac{\gamma}{4} \) and \( \delta = \frac{\gamma \phi}{\log(n)} \).

- Move down the tree starting from the root. For each node, run \text{PointQuery} for each of its two children. If a child is a heavy hitter, i.e. \( \text{PointQuery} \) 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 \( \text{PointQuery}(i) \geq \frac{3}{4} \phi \|x\|_1 \).

**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 $O\left(\frac{1}{\phi}\right)$ heavy hitters. So we make $O\left(\frac{\log(n)}{\phi}\right)$ 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 $t = \log\left(\frac{1}{\phi}\right)$. So the total time is: $O\left(\frac{\log(n)}{\phi} \log\left(\frac{\log(n)}{\phi}\right)\right)$

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

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 $\|x - \tilde{x}\|_\infty \leq \alpha \|x_{tail(k)}\|_1$ 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. $E(error) = \frac{\|x_{tail(k)}\|_1}{w}$. And if $w = O\left(\frac{ck}{\alpha}\right)$ we expect something like $\alpha$ 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 $O\left(\frac{\|x_{tail(k)}\|_1}{w}\right) = O\left(\frac{\|x_{tail(k)}\|_1}{k}\right)$, 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 $\sigma$ 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 $O_{j,h}(i) = x_i^2 + noise$ and can bound the noise and show that taking the medians of the counters gives good estimates with high probability.
References
