ℓ₁ heavy hitters: State-of-the-art

Larsen, Nelson, Nguyen, Thorup [Larsen et al., 2016]
Another suggestion for the class project (contains a cool graph clustering problem, probably of independent interest)

Today we will use Jelani’s exposition for Count-Min sketch from 2016 TUM Summer School.
**$\ell_1$ point queries**

**Setting:** Strict Turnstile

- $x \in \mathbb{R}^n$
- $x \leftarrow 0$ initially
- At step $i$ we see an update $(i, \Delta)$ which causes the change
  \[ x_i \leftarrow x_i + \Delta. \]
- $\Delta$ can be negative, $x_i \geq 0$ at all times for all $i \in [n]$
  When $x_i$s can also be negative this is called the general turnstile model.

- **QUERY**(i): Return value $\tilde{x}_i$ in the range $x_i \pm \varepsilon \cdot \|x\|_1$. 
Heavy Hitters

- **HEAVYHITTER**: Return a set $L \subseteq [n]$ such that
  1. $|x_i| \geq \varepsilon \|x\|_1 \Rightarrow i \in L$
  2. $|x_i| < \frac{\varepsilon}{2} \|x\|_1 \Rightarrow i \notin L$

Today, we present the CountMin sketch [Cormode and Muthukrishnan, 2005], which solves $\ell_1$ point query in the general turnstile model.

- C code available by G. Cormode
from streamlib import CountMin
cm = CountMin()
cm.processBatch([0, 0, 1, 1, 1, 2, 4])
for i in xrange(5):
    print i, '↓', cm.estimate(i)
Observation

Claim: If we can solve point query in small space then we can solve heavy hitters in small space as well (though not necessarily efficient run-time).

• Run point query with $\varepsilon/4$ on each $i \in [n]$

• Output the set of indices $i$ for which we had large estimate of $x_i$, i.e. at least $(3\varepsilon/4)\|x\|_1$
Count-Min sketch

1. We store hash functions $h_1, \ldots, h_L : [n] \rightarrow [t]$, each chosen independently from a 2-wise independent family.

2. We store counters $C_{a,b}$ for $a \in [s]$, $b \in [t]$ with $s = \lceil 2/\varepsilon \rceil$, $t = \lceil \log_2(1/\delta) \rceil$.

3. Upon an update $(i, \Delta)$, we add $\Delta$ to all counters $C_{a,h_a(i)}$ for $a = 1, \ldots, s$.

4. To answer $\text{query}(i)$, we output $\min_{1 \leq a \leq s} C_{a,h_a(i)}$. 
Count-Min sketch

Source: Count-Min Sketch by G. Cormode

Note that our total memory consumption, in words is
\[ m = O(st) = O(\varepsilon^{-1} \log(1/\delta)). \]
Claim: \( \text{query}(i) = x_i \pm \varepsilon \|x\|_1 \) w.p \( \geq 1 - \delta \).

Proof: Fix \( i \), let \( Z_j = 1 \) if \( h_r(j) = h_r(i) \), \( Z_j = 0 \) otherwise. Now note that for any \( r \in [s] \), \( C_{r,h_r(i)} = x_i + \sum_{j \neq i} x_j Z_j \). We have

\[
E(\mathbb{E}) = \sum_{j \neq i} |x_j| \mathbb{E} Z_j = \sum_{j \neq i} |x_j| / t \leq \varepsilon / 2 \cdot \|x\|_1.
\]

Thus by Markov’s inequality, \( \mathbb{P}(E > \varepsilon \|x\|_1) < 1/2 \). Thus by independence of the \( s \) rows of the CountMin sketch, \( \mathbb{P}(\min_r C_{r,h_r(i)} > x_i + \varepsilon \|x\|_1) < 1/2^L = \delta \).
Count-Min sketch

Theorem

There is an algorithm solving the $\ell_1 \epsilon$-heavy hitter problem in the strict turnstile model with failure probability $\delta$, space $O(\epsilon^{-1} \log(n/\delta))$, update time $O(\log(n/\delta))$, and query time $O(n \log(n/\delta))$.

Proof.

We can instantiate a point query data structure with failure probability $\delta/n$. Then we point query every $i \in [n]$ and include in our output list $L$ only those $i$ for which query returned a value at least $(3\epsilon/4)\|x\|_1$.

Question: Can we improve the query time?
Count-Min sketch: Speeding up Query Time

\[
\begin{align*}
\{1, 2, \ldots n\} \\
\{1, 2, \ldots \frac{n}{2}\} & \quad \{\frac{n}{2} + 1, \ldots n\} \\
& \quad \vdots & \quad \vdots \\
& \quad 1 & \quad 2 \ldots & \quad \ldots n - 1 & \quad n
\end{align*}
\]
Count-Min sketch: Speeding up Query Time

• Tree has $1 + \lg n$ levels

• We store in memory is $1 + \lg n$ CM sketches, one per level

• Upon an update, we feed that update to the appropriate coordinate at the CM sketch at every level.

For $\alpha$-HHs, and final failure probability $\delta$, each CM sketch has error parameter $\varepsilon = \alpha/4$ and failure probability $\eta = \delta \alpha/(4 \lg n)$.

• Insight: The value at any ancestor of a node is at least as big as the value at that node.
Count-Min sketch: Speeding up Query Time

• **Insight**: The value at any ancestor of a node is at least as big as the value at that node.

• There can only be at most \(2/\alpha\) indices that are \(\alpha/2\) heavy hitters

• We move down the tree starting from the root (definitely a HH)

• At each level \(j\) of the tree, we keep track of a list \(L_j\) of heavy hitters at that level (definitely anything that is \(\alpha\)-HH, nothing below \(\frac{\alpha}{2} - HH\)).
Count-Min sketch: Speeding up Query Time

- For any node in $L_j$ we point query its two children.

- If a child has point query output at least $(3\alpha/4)\|x\|_1$, we include it in $L_{j+1}$.

- Finally, our final output list $L$ is simply the list corresponding to the bottom-most level of the tree.
Count-Min sketch: Speeding up Query Time

For failure probability $\delta$:

- **Words of space:** $O(\varepsilon^{-1} \lg n \lg((\lg n)/(\alpha \delta)))$

- **Update time:** $O(\lg n \lg(1/\eta)) = O(\lg n \lg((\lg n)/(\alpha \delta)))$

- **Query time:** $O(\varepsilon^{-1} \lg n \lg((\lg n)/(\alpha \delta)))$
\(\ell_1\) heavy hitters: State-of-the-art

Larsen, Nelson, Nguyen, Thorup [Larsen et al., 2016]

For failure probability \(\delta\)

- **Words of space:** \(O(\varepsilon^{-1} \lg(n/\delta))\)
- **Update time:** \(O(\lg(n/\delta))\)
- **Query time:** \(O(\varepsilon^{-1} \lg(n/\delta) \text{ poly}(\lg n))\)


In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*.