CMPUT 675: Algorithms for Streaming and Big Data **Fall 2019**

Lecture 8 (Sep 30, 2019): Applications of Counting Sketch

Lecturer: Mohammad R. Salavatipour *Scribe: Brandon Fuller*

8.1 Applications of Count and Count-Min Sketch Algorithms

In the last lecture we discussed some possible applications for the count and count-min sketch algorithms. In this lecture we more formally define the problems and describe how we might approach implementing the solutions. All problems will be queries that can be asked at any point during the algorithm. For the purpose of this lecture we will consider the examples for the count sketch algorithm.

8.1.1 Point Query

We begin with the easiest application. Point Query asks for a frequency at any point during the algorithm. This is trivial as during the algorithm we are maintaining our estimates.

Problem 1 (Point Query) *Answer query $Q(i)$: return an estimate of f_i.*

This problem is easy to answer because at any point we have a current running estimate of f_i. Whenever you receive query $Q(i)$ return $\hat{f}_i$ as the median of all the running estimates.

8.1.2 Heavy Hitters

This next query is interested in knowing which are the most frequent items you have seen so far. We say an element is frequent if it supersedes a threshold percentage of the total count.

Problem 2 (Heavy Hitters) *Fix $\alpha \in (0,1)$ before running the count sketch algorithm. Answer query Q: return the α-heavy hitters. Specifically output all $a \in [n]$ such that $f_a \geq \alpha \|f\|_1$.*

For simplicity we can assume we are working with a cash register model, meaning we only add elements and never remove them. To do this, maintain a heap. Every time we update $\hat{f}_a$ we check if $\hat{f}_a \geq \alpha \|f\|_1$. If it is, add it to the heap, or update it if it is already in the heap. If $\hat{f}_a < \alpha \|f\|_1$ and it is currently in the heap, remove it. Make sure the heap has at most $\frac{1}{\alpha}$ elements in it at any given time, removing smallest counts when necessary. The space complexity for the heap is $O(\frac{1}{\alpha}(\log n + \log m))$ and the update time is $O(\log \frac{1}{\alpha})$.

8.1.3 Range Queries

We now consider a more interesting query. The Range Query is interested in knowing the sum of a contiguous range of frequencies. A naive approach to solving this problem for any given range would be to use the answer to the Point Query for each index in the range. We will show a faster solution after stating the exact problem.

Problem 3 (Range Queries) *Answer query $Q(l, r)$: return an estimate of $\sum_{i=l}^{r} f_i$.*

For simplicity assume that n is a power of two (if it is not, round n up to the nearest power of two). For an interval $[i, j]$ if there exists a $k \geq 0$, such that $2^k|(i-1)$ and $j - i + 1 = 2^k$ then it is called a *dyadic interval*. Some examples of dyadic intervals: $[3, 3]$ has length $2^0 = 1$, $[5, 6]$ has length $2^1 = 2$ and $[1, 8]$ has length $2^3 = 8$.

Create a node for every dyadic interval that has a right endpoint less than $n = 2^r$:

- Length 2^0: $[1, 1], [2, 2], \ldots, [n, n]$
- Length 2^1: $[1, 2], [3, 4], \ldots, [n-1, n]$

  $\vdots$
- Length 2^i: $[1, 2^i], [2^i + 1, 2^{i+2}], \ldots, n - 2^i + 1, n]$

  $\vdots$
- Length 2^r: $[1, n]$

For nodes of length 2^i,$i > 0$, make it the parent node of the two nodes with intervals of length 2^{i-1} whose union is this nodes interval. For each node of each dyadic interval $[i, j]$, we will store$\sum_{k=i}^{j} \hat{f}_k$. This binary tree has exactly $2n - 1$ nodes. We can clearly see that the nodes of length 1 would store exactly the estimates $\hat{f}_k$ as before. Whenever the frequency of $\hat{f}_k$ gets updated we must also update every node which contains k in their interval, of which there are $O(\log n)$ such nodes.

Now, to determine $Q(l, r)$, we want to find some non-overlapping subset of dyadic intervals S such that $[l, r] = \cup_{P \in S} P$. Finding such a set we can use the nodes to get the desired sum. We use the following lemma to show that such a subset exists, its size, and subsequently how to obtain such a set.

Lemma 1 *Every range $[l, r]$ can be partitioned into a subset S of dyadic intervals where $|S| \leq 2 \log n$.*

Proof. Consider the following recursive algorithm which uses $I(N)$ to represent the interval of node N and $l(N), r(N)$ to represent the left and right child respectively.

RangeQuery(node N, interval P)

1. $S \leftarrow \emptyset$
2. If $I(N) = P$ then:
   - $S \leftarrow \{P\}$
3. Else if P is a non-empty interval:
   - $S \leftarrow$ **RangeQuery(** $l(N)$**,** $P \cap I(l(N))$ **)** $\cup$ **RangeQuery(** $r(N)$**,** $P \cap I(r(N))$ **)**
4. Return S

Running this algorithm starting with N being the node that has interval $[1, n]$ and $P = [l, r]$ will give the set S. It can be seen that at any level of the binary tree there are at most 2 intervals added to S. Thus since the tree has height $\log n$ we get at most $2 \log n$ intervals added to S before returning. ■

For actual applications we cannot store $O(n)$. As we kept a count sketch for the lists of length 1 we will now keep a count sketch for each list of dyadic intervals with the same length. This approach will take $O(\log n \log \frac{1}{\delta})$ for updates.

8.1.4 Sparse Recovery

For a vector $x \in \mathbb{R}^n$ suppose we are looking for a sparse vector $z \in \mathbb{R}^n$ that is an approximation of x. A vector is *sparse* if it has few non-zero entries. Specifically, a vector z is α-sparse if $\|z\|_0 \leq \alpha$; the number of non-zero elements is less than α. Let

$$err_p^\alpha(x) = \min_{z:\|z\|_0 \leq \alpha} \|x - z\|_p$$

and consider S to be the set of α indices with largest values in x. Then we get $err_p^\alpha(x) = (\sum_{i \notin S} |x_i|^p)^{\frac{1}{p}}$.

Lemma 2 *With high probability, $\forall i \in [n]$, $|\hat{f}_i - f_i| \leq \frac{\epsilon}{\sqrt{\alpha}} err_2^\alpha(f)$ if we choose $k = \frac{3\alpha}{\epsilon}$ and $t = O(\log n)$ for the count sketch algorithm.*

Proof. Let S be the α indices with largest frequencies in f and let f' be obtained from f by zeroing all indices in S. Then $err_2^\alpha(f) = \|f'\|_2$.

Fix an index i. For any ℓ, let $z_\ell = g_\ell(i)C[\ell, h(i)]$ from the count sketch algorithm. Define the event A_ℓ to be the event $\exists i' \in S \backslash \{i\}$ such that $h_\ell(i) = h_\ell(i')$. So A_ℓ is 1 if the coordinate i has a collision with an index of S under h_ℓ and 0 otherwise.

$$\begin{aligned}
\Pr[A_\ell] &= \Pr[A_\ell = 1] \\
&\leq \sum_{i' \in S \backslash \{i\}} \Pr[h_\ell(i) = h_\ell(i')] \\
&\leq \frac{|S|}{k} \\
&= \frac{\alpha}{(\frac{3\alpha}{\epsilon})} \\
&= \frac{\epsilon}{3}
\end{aligned}$$

Then we get:

$$\begin{aligned}
\Pr[|z_\ell - f_i| \geq \frac{\epsilon}{\sqrt{\alpha}} err_2^\alpha(f)] &= \Pr[|z_\ell - f_i| \geq \frac{\epsilon}{\sqrt{\alpha}} \|f'\|_2] \\
&= \Pr[A_\ell] \Pr[|z_\ell - f_i| \geq \frac{\epsilon}{\sqrt{\alpha}} \|f'\|_2 | A_\ell] + \Pr[\bar{A}_\ell] \Pr[|z_\ell - f_i| \geq \frac{\epsilon}{\sqrt{\alpha}} \|f'\|_2 | \bar{A}_\ell] \\
&\leq \Pr[A_\ell] + \Pr[|z_\ell - f_i| \geq \frac{\epsilon}{\sqrt{\alpha}} \|f'\|_2 | \bar{A}_\ell] \\
&\leq \frac{\epsilon}{3} + \frac{1}{3} \\
&\leq \frac{1}{2}
\end{aligned}$$

The second equality is gotten by splitting it up into the event A_ℓ happens and the event A_ℓ does not happen. The fourth equality is from the previous equation alongside Theorem 1 from Lecture 7. We over estimate but we can see that this estimation is good enough since this probability only considers a single hash function. If we consider the medians over all the hash functions, we must have more than half of the hash functions fail. Since they are independent, each with probability $\leq \frac{1}{2}$, this would have a probability $\leq 2^{-\frac{t}{2}}$ which is enough to finish the lemma given our choice of t. ■

Suppose that $\hat{f}$ is our estimate of f from the count sketch algorithm. We will prove in the following lemma that $\hat{z}$, being the largest α elements of $\hat{f}$, is a good estimate of an α-sparse vector for f.

Lemma 3 *Suppose $x, y \in \mathbb{R}^n$ where $\|x-y\|_\infty \leq \frac{\epsilon}{\sqrt{\alpha}} err_2^\alpha(x)$. Then if z is obtained from y by taking the largest α coordinates (zeroing all the others), then $\|x-z\|_2 \leq (1+5\epsilon) err_2^\alpha(x)$.*

Proof. For the proof, denote $err_2^\alpha(x) = E$.

Let S, T be the set of indices of the largest α coordinates in x and y respectively. Then $z_i = y_i$ if $i \in T$ and $z_i = 0$ otherwise.

$$\begin{aligned} \|x-z\|_2^2 &= \sum_{i\in T} |x_i - z_i|^2 + \sum_{i \in S\setminus T} |x_i - z_i|^2 + \sum_{i \in [n]\setminus(S\cup T)} |x_i - z_i|^2 \\ &= \sum_{i\in T} |x_i - y_i|^2 + \sum_{i \in S\setminus T} |x_i|^2 + \sum_{i \in [n]\setminus(S\cup T)} |x_i|^2 \end{aligned}$$

Since $\|x-y\|_\infty \leq \frac{\epsilon}{\sqrt{\alpha}} E$ we get

$$\begin{aligned} \sum_{i\in T} |x_i - y_i|^2 &\leq \sum_{i\in T} (\frac{\epsilon}{\sqrt{\alpha}} E)^2 \\ &= |T| \frac{\epsilon^2}{\alpha} E^2 \\ &= \alpha \frac{\epsilon^2}{\alpha} E^2 \\ &= \epsilon^2 E^2 \end{aligned}$$

Also we have $\forall i, |x_i| - |y_i| \leq |x_i - y_i| \leq \frac{\epsilon}{\sqrt{\alpha}} E$ and $|y_i| - |x_i| \leq |x_i - y_i| \leq \frac{\epsilon}{\sqrt{\alpha}} E$. Rearranging the two equations and we get $|x_i| \leq |y_i| + \frac{\epsilon}{\sqrt{\alpha}} E$ and $-|x_i| \leq -|y_i| + \frac{\epsilon}{\sqrt{\alpha}} E$. Thus for any i, j we get $|x_i| - |x_j| \leq |y_i| - |y_j| + \frac{2\epsilon}{\sqrt{\alpha}} E$.

Specifically if we consider $a = \max_{i \in S\setminus T} x_i$ and $b = \min_{i \in T\setminus S} x_i$ we get $a - b \leq \frac{2\epsilon}{\sqrt{\alpha}} E$. Also, $b \leq \frac{\|x_{T\setminus S}\|_2}{\sqrt{(|T\setminus S|)}}$ since b is the minimum of the set. Using these facts we get the following:

$$\begin{aligned}
\sum_{i\in S\setminus T}|x_i|^2 &\leq \sum_{i\in S\setminus T}a^2\\
&= a^2|S\setminus T|\\
&\leq (b+\frac{2\epsilon}{\sqrt{\alpha}}E)^2|S\setminus T|\\
&\leq (\frac{\|x_{T\setminus S}\|_2}{\sqrt{|T\setminus S|}}+\frac{2\epsilon}{\sqrt{\alpha}}E)^2|S\setminus T|\\
&= (\frac{\|x_{T\setminus S}\|_2^2}{|S\setminus T|}+\frac{4\epsilon}{\sqrt{\alpha}}\frac{\|x_{T\setminus S}\|_2}{\sqrt{|S\setminus T|}}+\frac{4\epsilon^2}{\alpha}E^2)|S\setminus T|\\
&\leq \|x_{T\setminus S}\|_2^2+4\epsilon E+4\epsilon^2E^2\\
&\leq \|x_{T\setminus S}\|_2^2+8\epsilon E^2
\end{aligned}$$

Now we have from the two above equations:

$$\begin{aligned}
\|x-z\|_2^2 &= \sum_{i\in T}|x_i-y_i|^2+\sum_{i\in S\setminus T}|x_i|^2+\sum_{i\in[n]\setminus(S\cup T)}|x_i|^2\\
&\leq \epsilon^2E^2+\|x_{T\setminus S}\|_2^2+8\epsilon E^2+\|x_{[n]\setminus(S\cup T)}\|_2^2\\
&= 9\epsilon E^2+\|x_{[n]\setminus S}\|_2^2\\
&= 9\epsilon E^2+E^2\\
&\leq (1+10\epsilon)E^2
\end{aligned}$$

Finally we get $\|x-z\|_2\leq\sqrt{(1+10\epsilon)E^2}\leq(1+5\epsilon)E$ as desired. ■

In the next lecture we will continue sparse recovery and its use in applications.

References

CCFC04 M. Charikar, K. C. Chen, and M. Farach-Colton, Finding frequent items in data streams. *Theoretical Computer Science*, 312:03-15, 2004.

CM05 G. Cormode and S. Muthukrishnan, An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58-75, 2005.

CM06 G. Cormode and S. Muthukrishnan, Combinatorial algorithms for compressed sensing. *In Proc. 40th Ann. Conf. Information Sciences and Systems, Princeton*, 2006.