**CSE 522: Sublinear (and Streaming) Algorithms** **Spring 2014**

# Lecture 5: More Heavy Hitters: Count Sketch

April 14, 2014

*Lecturer: Paul Beame* *Scribe: Paul Beame*

We recall that for the turnstile and cash-register versions of streaming input are of the form $(x_i, c_i)$ for $x_i \in [M]$ and $c_i \in \mathbb{Z}$ and the frequency vector $f$ is given by $f_j = \sum_{i:\ x_i=j} c_i$. Recall:

**The COUNT-MIN Sketch Algorithm**

<u>Initialize:</u><br>
$k \leftarrow \lceil 2/\varepsilon \rceil$<br>
$t \leftarrow \lceil \log_2(1/\delta) \rceil$<br>
$C \leftarrow t \times k$ integer array, initially 0<br>
Choose $h_1, \ldots h_t : [M] \rightarrow [k]$ independently from a 2-universal family of hash functions<br>
<u>Process:</u><br>
**for** each $i$ **do**<br>
    **for** $s = 1$ to $t$ **do**<br>
        $C[s, h_s(x_i)] \leftarrow C[s, h_s(x_i)] + c_i$<br>
    **end for**<br>
**end for**<br>
<u>Output:</u> $\tilde{f} \leftarrow (C, h_1, \ldots, h_t)$<br>
$\tilde{f}_j = \min\{C[s, h_s(j)] \ : \ s = 1, \ldots, t\}$

The total space of the sketch is $O(\frac{1}{\varepsilon}\log(1/\delta)(\log n + \log M))$. By our construction, for each fixed $j$, $f_j \leq \tilde{f}_j \leq f_j + \varepsilon||f_{-j}||_1$

In order to get a guarantee that is closer to that of the Misra-Gries sketch we may actually want this level of approximation for every $j$ with a fixed failure probability $\delta$. To do this we can apply the COUNT-MIN Sketch with $\delta/M$ instead of $\delta$ and take a union bound. This would replace $\log(1/\delta)$ with $\log(1/\delta) + \log M$ in our space bound.

**COUNT-MIN as a Linear Map** If we view the $C$ matrix as a length $tk$ vector, we can see that it is a *linear* function of the data as follows. (The linearity of this sketch is why it can handle the turnstile model, though it does require that the final $f$ vector is non-negative.)

Define a $tk \times M$ matrix $A_{h_1,\ldots,h_t}$ depending on $h_1, \ldots, h_t$ of the COUNT-MIN Sketch where for $s \in [t]$ and $\ell$ the $(s, \ell)$ entry corresponds to row $(s-1)k + \ell$ of $A$. For each block of $k$ rows, which corresponds to a single hash function, there will be precisely one 1 in each column; the rest will be 0. Alternative, the row corresponding to the pair $(s, \ell)$ will be the characteristic vector of $h_s^{-1}(\ell)$, the set of all elements of $[M]$ that $h_1$ maps to $\ell$. Since $M$ is enormous, this will be a huge matrix with many more columns than rows. It is, however, very succinctly specified.

The input $(x_i, c_i)$ corresponds to a vector of length $M$, with 0's everywhere except for a single $c_i$ in the $x_i$-th entry, a vector $c_i \; e_{x_i}$ where $e_{x_i}$ is the $x_i$-th unit vector. The $t$ updates to $C$ add $A_{h_1,\ldots h_t} c_i \; e_{x_i}$ to the current vector representing the $tk$ elements of $C$.

$$\left[\begin{array}{cccccccccccccc}
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \cdots & \cdots & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 & 1 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \cdots & \cdots & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 1 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \cdots & \cdots & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 & 1 & 0
\end{array}\right] \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ c_i \\ \vdots \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The total sketch is then $\sum_{i=1}^n c_i \; A_{h_1,\ldots,h_t} \cdot e_{x_i} = A_{h_1,\ldots,h_t} \cdot (\sum_{i=1}^n c_i \; e_{x_i})$. But the $j$-th coordinate of $\sum_{i=1}^n c_i \; e_{x_i}$ is precisely $f_j = \sum_{i:\, x_i = j} c_i$ so $\sum_{i=1}^n c_i \; e_{x_i} = f$. Therefore, the straightened out version of the final version of $C$ is

$$A_{h_1,\ldots,h_t} \cdot f.$$

**The** COUNT **Sketch** By our construction, the COUNT-MIN Sketch produces a $\tilde{f}$ such that for each fixed $j$, $f_j \leq \tilde{f}_j \leq f_j + \varepsilon ||f_{-j}||_1$. The COUNT Sketch, which was actually defined earlier than the simpler COUNT-MIN Sketch, will give a more accurate approximation in that the error will be based on the $\ell_2$ norm, rather than the $\ell_1$ norm of $f$. This will allow us to find $(\gamma, 2)$ heavy hitters rather than just $(\gamma, 1)$-heavy hitters.

As before, we begin first with a basic version with large failure probability. The main difference is that instead of always adding $c_i$ at the location $h(x_i)$, depending on $x_i$ we may subtract $c_i$ rather than adding it. Consider a single $j^*$. For every $i$ such that $x_i = j^*$ the value $c_i$ will consistently either be subtracted or added to location $C[h(j^*)]$. To recover the contribution for $j^*$ we flip the sign depending on $j^*$. The big advantage versus the COUNT-MIN sketch where collisions always bias things in the same direction, is that the other values of $j$ that collide with $j^*$ under the hash function will tend to cancel each other out because of the random directions of their signs.

**The Basic** COUNT **Sketch**

<u>Initialize:</u>
$k \leftarrow \lceil 3/\varepsilon^2 \rceil$
$C \leftarrow$ length $k$ integer array, initially 0
Choose $h : [M] \to [k]$ from a 2-universal family of hash functions
Choose $g : [M] \to \{1, -1\}$ from a pairwise independent family of hash functions
<u>Process:</u>
**for** each $i$ **do**
  **for** $s = 1$ to $t$ **do**
    $C[h(x_i)] \leftarrow C[h(x_i)] + c_i \; g(x_i)$
  **end for**
**end for**
<u>Output:</u> $\tilde{f} \leftarrow (C, h, g)$
$\tilde{f}_j = g(j) \cdot C[h(j)]$

**Analysis** Fix $j^* \in [M]$. Let

$$Y_j = \begin{cases} 1 & \text{if } h(j) = h(j^*) \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$\begin{aligned} \tilde{f}_{j^*} &= g(j^*) \cdot C[h(j^*)] \\ &= g(j^*) \cdot \sum_{i=1}^{n} c_i \cdot g(x_i) Y_{x_i} \\ &= g(j^*) \cdot \sum_{j=1}^{M} f_j \; g(j) Y_j \\ &= g(j^*)^2 f_{j^*} Y_{j^*} + \sum_{j \neq j^*} f_j \; g(j^*) g(j) Y_j \\ &= f_{j^*} + \sum_{j \neq j^*} f_j \; g(j^*) g(j) Y_j \qquad () \end{aligned}$$

since $g(j^*)^2 = 1$ and $Y_{j^*} = 1$. Therefore, by linearity of expectation,

$$\mathbb{E}(\tilde{f}_{j^*}) = f_{j^*} + \sum_{j \neq j^*}^{M} f_j \; \mathbb{E}(g(j^*) g(j) Y_j).$$

Now since $g$ is pairwise independent and is independent of $Y_j$ which is a function solely of $h$, for $j \neq j^*$ we have $\mathbb{E}(g(j^*)g(j)Y_j) = \mathbb{E}(g(j^*))\mathbb{E}(g(j)Y_j) = 0 \cdot \mathbb{E}(g(j)Y_j) = 0$. Therefore

$$\mathbb{E}(\tilde{f}_{j^*}) = f_{j^*};$$

i.e, $\tilde{f}_{j^*}$ is an unbiased estimator of $f_{j^*}$. We now compute its variance

$$\begin{aligned}
Var(\tilde{f}_{j^*}) &= \mathbb{E}((\tilde{f}_{j^*} - f_{j^*})^2) \\
&= \mathbb{E}((\sum_{j \neq j^*} f_j \; g(j^*)g(j)Y_j)^2) \qquad \text{by } (1) \\
&= \mathbb{E}(\sum_{i \neq j^*} \sum_{j \neq j^*} f_i \; f_j \; g(i)g(j)Y_iY_j) \qquad \text{since } g(j^*)^2 = 1 \\
&= \sum_{i \neq j^*} \sum_{j \neq j^*} f_i \; f_j \; \mathbb{E}(g(i)g(j)Y_iY_j)
\end{aligned}$$

Observe that for $i \neq j$, since $g$ is pairwise independent and independent of $h$, $\mathbb{E}(g(i)g(j)Y_iY_j) = \mathbb{E}(g(i))\mathbb{E}(g(j)Y_iY_j) = 0$. Therefore the only terms in the variance that are survive are when $i = j$, so

$$Var(\tilde{f}_{j^*}) = \sum_{j \neq j^*} f_j^2 \mathbb{E}(Y_j^2).$$

Now $\mathbb{E}(Y_j^2) = \mathbb{E}(Y_j)$ since $Y_j$ is an indicator and $\mathbb{E}(Y_j) = \mathrm{P}[h(j) = h(j^*)] = 1/k$ by the 2-universal property of $h$. Therefore

$$Var(\tilde{f}_{j^*}) = \sum_{j \neq j^*} f_j^2/k = \frac{||f||_2^2 - f_{j^*}^2}{k} = \frac{||f_{-j^*}||_2^2}{k},$$

where $f_{j^*} = (f_1, \ldots, f_{j^*-1}, f_{j^*+1}, \ldots, f_M)$. By Chebyshev's inequality,

$$\begin{aligned}
\mathbb{P}[\; |\tilde{f}_{j^*} - f_{j^*}| \geq \varepsilon ||f_{-j^*}||_2 \;] &\leq \frac{Var(\tilde{f}_{j^*})}{\varepsilon^2 ||f_{-j^*}||_2^2} \\
&\leq \frac{1}{\varepsilon^2 k} \leq 1/3
\end{aligned}$$

by our choice of $k$. Since this is bounded below 1/2 we can apply the median trick to get the final COUNT Sketch which is given below.

**The COUNT Sketch Algorithm**

1: <u>Initialize:</u>
2: $k \leftarrow \lceil 3/\varepsilon^2 \rceil$
3: $t \leftarrow \lceil c \log_2(1/\delta) \rceil$ for some constant $c$.
4: $C \leftarrow t \times k$ integer array, initially 0
5: Choose $h_1, \ldots h_t : [M] \to [k]$ independently from a 2-universal family of hash functions

Choose $g_1, \ldots, g_t : [M] \rightarrow \{1, -1\}$ independently from a pairwise independent family of hash functions
Process:
**for** each $i$ **do**
    **for** $s = 1$ to $t$ **do**
        $C[s, h_s(x_i)] \leftarrow C[s, h_s(x_i)] + c_i \; g_s(x_i)$
    **end for**
**end for**
Output: $\tilde{f} \leftarrow (C, h_1, \ldots, h_t, g_1, \ldots, g_t)$
$\tilde{f}_j = \text{Median}\{g_s(j) \cdot C[s, h_s(j)] \ : \ s = 1, \ldots, t\}$

Therefore for each $j$, with probability at least $1 - \delta$,

$$f_j - \varepsilon||f_{-j}||_2 \leq \tilde{f}_j \leq f_j + \varepsilon||f_{-j}||_2.$$

The total space is $O(\frac{1}{\varepsilon^2}\log(1/\delta)(\log n + \log M))$ which pays for the sharper bound for the $\ell_2$ rather than $\ell_1$ norm at the space cost of a $1/\varepsilon^2$ factor instead of a $1/\varepsilon$ factor. In matrix form, the representation of this sketch is something like

$$\left[\begin{array}{cccccccccccccc}
0 & 0 & -1 & 0 & 0 & +1 & 0 & -1 & \cdots & \cdots & -1 & 0 & 0 & 0 \\
0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
+1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & -1 & 0 \\
\hline
0 & 0 & -1 & 0 & 0 & +1 & 0 & -1 & \cdots & \cdots & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & -1 & 0 \\
+1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & -1 & 0 & 0 \\
\hline
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\hline
0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & -1 & 0 \\
+1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & +1 & 0 & -1 & \cdots & \cdots & -1 & 0 & 0 & 0
\end{array}\right].$$

**Practical frequency estimation** There are many reasons in practice to be able to maintain the The three algorithms we have seen for frequency estimation and heavy hitters are interesting for practice. There is a nice survey on the subject by Cormode and Hadjelefhteriou in a CACM 2009 article. This also discusses a deterministic algorithm, the Space-Saving algorithm of Metwally et al. from 2006, which is very similar to the Misra-Gries Algorithm but has some other nice properties, and is discussed in a problem on the first homework. Though we have given worst-case bounds, the hard cases are when the data is very uniformly distributed; one can show that if there is sufficient skew in the data, some of these algorithms will find the heavy hitters with certainty.