Lecture 9: Streaming and Sketching II

> We will follow for this chapter the (excellent) lecture notes by Amit Chakrabarti [AC], available at `https://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf`.

In the previous lecture, we introduced streaming algorithms, and saw a few examples: the Misra–Gries algorithm for Frequent Elements ("Heavy Hitters"), the Morris Counter for F_1 estimation, and the Tidemark and BJKST algorithms for Distinct Elements (F_0 estimation). The way we framed the corresponding question was in terms of a single stream σ of m items being provided, in an arbitrary order, to an algorithm A, which, at the end of the stream, had to output an answer $A(\sigma)$.

This is all good, but one can easily imagine settings where, after running A on a stream of inputs σ, getting an answer $y = A(\sigma)$ and clearing all the internal state and memory used by A, one gets a follow-up stream σ', and would like to update the solution y to get a solution $y' = A(\sigma \circ \sigma')$ to the "full stream." Or scenarios where Alice runs A on her input σ_A to get some solution y_A, Bob runs A on his input σ_B to get some solution y_B, and afterwards they would like to use y_A andy_B compute some "joint solution" to the full input $\sigma_A \circ \sigma_B$. This would be very natural, *but is not captured by the definition of streaming algorithms we saw earlier.* Fortunately, this is what a very specific type of streaming algorithms do provide: these are called *sketching algorithms.*

While sketching algorithms are a strict subset of streaming algorithms, many important streaming algorithms belong to this subset.

Chapter 5.2 of [AC]

Sketching

A sketching algorithm is the response to the following very natural question:

> *If I run two instances of a streaming algorithm for problem $\mathcal{P}$ on a stream σ_1 and on a stream σ_2, can I combine their outputs to get the same result as if I had run my algorithm for $\mathcal{P}$ on the stream $\sigma_1 \circ \sigma_2$?*

This sounds very desirable, as this allows to stop an algorithm, distribute it across multiple servers or subsets of a stream, and still

be able to recombine everything:

$$A(\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_k) = f(A(\sigma_1), A(\sigma_2), \cdots, A(\sigma_k))$$

(where f is some prespecified function which allows to combine the *sketches* of the k individual streams into an overall sketch for the whole stream).

What is even *more* appealing is a *linear* sketching algorithm, where the (sketch) output of the algorithm is just a linear function of the input stream (into a lower-dimensional space), and where "combining their outputs" just means... summing them up:

$$A(\sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_k) = A(\sigma_1) + A(\sigma_2) + \cdots + A(\sigma_k)$$

(small catch here: the summation $+$ is typically in a suitable vector space, so not always the usual sum over the reals. That being said, it often just *is* the usual sum over real-valued vectors.)

Back to Frequent Elements!

Remember that in the previous lecture, we saw the MISRA–GRIES algorithm which allowed us to compute deterministically, in one pass, an additive approximation of all the n frequencies of a given stream σ:

$$f_j - \varepsilon m \leq \widehat{f}_j \leq f_j, \qquad \text{for all } j \in [n]$$

using space $s = O(\log(mn)/\varepsilon)$. We will see in the tutorial that this is actually already a sketching algorithm!

It suffers from two possible issues, however: first, it only works in the "cash register" streaming model, where items come in the stream but are never removed ("numbers can only go up"). Second, the approximation guarantee it provides is rather weak: since

Cash register model: "numbers go up and only up"

$$\|f\|_1 = f_1 + f_2 + \cdots + f_n = m$$

for every stream σ, you can think of it as providing an ℓ_1-estimate of the stream:

$$-\varepsilon\|f\|_1 \leq \widehat{f}_j - f_j \leq 0 \qquad \text{for all } j \in [n] \tag{59}$$

which implies

$$\|\widehat{f} - f\|_\infty \leq \varepsilon\|f\|_1 \tag{60}$$

We could ask for other types of approximation: for instance, what if our error was with respect to another norm, say, ℓ_2? ℓ_p?

Recall that $\|x\|_2 \leq \|x\|_1$ for every vector $x \in \mathbb{R}^d$, so one approximation implies the other.

Count-Sketch

Chapter 5.3 of [AC]

We start with the CountSketch algorithm, due to Charikar, Chen and Farach–Colton, which works in the *turnstile* streaming model and provides exactly that: an ℓ_2 guarantee.

Turnstile model: "numbers go up, or down"

Fact 43.1. COUNTSKETCH *is a linear sketching algorithm, provided the sketches C_1, C_2 are built using the same hash functions h, g.*

Can you see why?

Algorithm 19: The CountSketch algorithm

Input: Parameter $\varepsilon \in (0,1]$

Set $k \leftarrow O(1/\varepsilon^2)$, and initialize an array C of size k to zero

Pick $h\colon [n] \to [k]$ from a strongly universal hashing family

Pick $g\colon [n] \to \{-1,1\}$ from a strongly universal hashing family

for all $1 \le i \le m$ **do**

  Get item $a_i = (j,c) \in [n] \times \{-B,\dots,B\}$ ▷ Assume $B = O(1)$

  $C[h(j)] \leftarrow C[h(j)] + c \cdot g(j)$

Output: On query $j \in [n]$, **return** $\widehat{f}_j \leftarrow g(j) \cdot C[h(j)]$

Some notation: for a vector $x \in \mathbb{R}^n$ and $j \in [n]$, denote by $x_{-j} \in \mathbb{R}^n$ the same vector, but with j-coordinate set to 0.

Theorem 44. *The (median trick version of the)* CountSketch *algorithm is a* randomised *one-pass sketching algorithm which, for any given parameters $\varepsilon, \delta \in (0,1]$, provides a (succinctly represented) estimate $\widehat{f}$ of frequency vector f of the stream such that, for every $j \in [n]$*

$$\Pr\left[\left|\widehat{f}_j - f_j\right| \le \varepsilon \|f_{-j}\|_2\right] \ge 1 - \delta$$

with space complexity

$$s = O\left(\left(\log n + \frac{1}{\varepsilon^2}\log m\right)\log\frac{1}{\delta}\right) = O\left(\left(\frac{\log(nm)}{\varepsilon^2}\right)\log\frac{1}{\delta}\right).$$

To compare it to Eq. (60), we obtain the following:

Corollary 44.1. *The (median trick version of the)* CountSketch *algorithm is a* randomised *one-pass sketching algorithm which, for any given parameters $\varepsilon, \delta \in (0,1]$, provides a (succinctly represented) estimate $\widehat{f}$ of frequency vector f of the stream such that*

$$\Pr\left[\|\widehat{f} - f\|_\infty \le \varepsilon \|f\|_2\right] \ge 1 - \delta$$

with space complexity

$$s = O\left(\frac{\log(nm)}{\varepsilon^2}\log\frac{n}{\delta}\right).$$

Proof of Theorem 44. As in previous arguments, it suffices to establish the theorem for constant error probability, as the median trick will then allow us to amplify the success probability to $1-\delta$ at the cost of a $O(\log(1/\delta))$ in the space complexity.

To begin, note that the space requirement comes from storing (1) the two hash functions h, g, and (2) an array of k values, each between $-B \cdot m$ and $B \cdot m$. Assuming we are using "good" (that is, small enough) hash families such as the ones seen earlier in the course, the total is

$$\begin{aligned} s &= O(\max(\log n, \log k)) + O(\log n) + O(k \cdot \log(Bm)) \\ &= O\left(\log n + \log\frac{1}{\varepsilon} + \frac{\log B + \log m}{\varepsilon^2}\right) \\ &= O\left(\log n + \frac{\log m}{\varepsilon^2}\right) \end{aligned}$$

(since we assume $B = O(1)$).

That was space. For correctness, the use of strongly universal hash families (*i.e.*, pairwise independent) hints at a variance-based argument, and so a natural idea is to compute the expectation and variance of each $\widehat{f}_j$ in view of applying Chebyshev's inequality. Let's proceed: fix any $j \in [n]$.

- Writing $a_i = (j_i, c_i)$ for each item a_i of the stream, observe that item a_i affects the value of $\widehat{f}_j$ if, and only if, $h(j_i) = h(j)$ (since then $C[h(j)]$ is modified when processing item a_i). Furthermore, for any $j \in [n]$, the contribution of j to the sketch C is limited to the cell $C[h(j)]$, and is equal to its frequency f_j, since

$$f_j = \sum_{i=1}^{m} c_i \mathbb{1}_{j_i = j}$$

As a result, the expectation of $\widehat{f}_j$ can be computed as

$$\begin{aligned}
\mathbb{E}\left[\widehat{f}_j\right] &= \mathbb{E}[g(j) \cdot C[h(j)]] \\
&= \mathbb{E}\left[g(j) \sum_{i=1}^{m} \mathbb{1}_{h(j_i)=h(j)} c_i \cdot g(j_i)\right] \\
&= \mathbb{E}\left[g(j) \sum_{j' \in [n]} \mathbb{1}_{h(j')=h(j)} \cdot g(j') f_{j'}\right] \\
&= \mathbb{E}\left[g(j)^2 f_j + \sum_{j' \in [n] \setminus \{j\}} \mathbb{1}_{h(j')=h(j)} \cdot g(j) g(j') f_{j'}\right]
\end{aligned}$$

By linearity of expectation, and since $g(j)^2 = 1$, we get

$$\begin{aligned}
\mathbb{E}\left[\widehat{f}_j\right] &= f_j + \sum_{j' \in [n] \setminus \{j\}} \mathbb{E}\left[\mathbb{1}_{h(j')=h(j)} g(j) g(j')\right] \cdot f_{j'} \\
&\qquad\qquad (g(j)^2 = 1, \text{ linearity of expectation}) \\
&= f_j + \sum_{j' \in [n] \setminus \{j\}} \mathbb{E}\left[\mathbb{1}_{h(j')=h(j)}\right] \cdot \mathbb{E}\left[g(j) g(j')\right] \cdot f_{j'} \\
&\qquad\qquad (h, g \text{ are independent}) \\
&= f_j + \sum_{j' \in [n] \setminus \{j\}} \Pr_h\left[h(j') = h(j)\right] \cdot \mathbb{E}[g(j)] \mathbb{E}\left[g(j')\right] \cdot f_{j'} \\
&\qquad\qquad (\text{pairwise independence of } g) \\
&= f_j + \sum_{j' \in [n] \setminus \{j\}} \Pr_h\left[h(j') = h(j)\right] \cdot 0 \cdot 0 \cdot f_{j'} \\
&\qquad\qquad (g(j), g(j') \text{ are uniformly distributed}) \\
&= f_j . \qquad (61)
\end{aligned}$$

(In the end we invoked the fact, proven in the tutorials, that drawing g from a strongly universal hash family implies that $g(x)$ is uniformly distributed, for every fixed x.) Great: $\widehat{f}_j$ has the right expectation.

- In order to give an upper bound on its variance $\mathrm{Var}[\widehat{f}_j] = \mathbb{E}\left[\widehat{f}_j^2\right] - \mathbb{E}\left[\widehat{f}_j\right]^2$, we expand the square to compute the first

term, using the same properties of h and g:

$$\begin{aligned}
\mathbb{E}\left[\widehat{f}_j^2\right] &= \mathbb{E}\left[\left(g(j)\sum_{j'\in[n]} \mathbb{1}_{h(j')=h(j)}\cdot g(j')f_{j'}\right)^2\right] \\
&= \mathbb{E}\left[g(j)^2\sum_{j',j''\in[n]} \mathbb{1}_{h(j')=h(j'')=h(j)}\cdot g(j')\cdot g(j'')f_{j'}f_{j''}\right] \\
&= \sum_{j',j''\in[n]} \mathbb{E}\left[\mathbb{1}_{h(j')=h(j'')=h(j)}\cdot g(j')\cdot g(j'')\right]f_{j'}f_{j''} \\
&\qquad\qquad (g(j)^2 = 1 \text{ and linearity of expectation}) \\
&= \sum_{j',j''\in[n]} \mathbb{E}\left[\mathbb{1}_{h(j')=h(j'')=h(j)}\cdot g(j')\cdot g(j'')\right]f_{j'}f_{j''} \\
&\qquad\qquad (g(j)^2 = 1 \text{ and linearity of expectation}) \\
&= \sum_{j',j''\in[n]} \Pr\left[\,h(j') = h(j'') = h(j)\,\right]\cdot \mathbb{E}\left[g(j')\cdot g(j'')\right]f_{j'}f_{j''} \\
&\qquad\qquad (h, g \text{ are independent}) \\
&= \sum_{j',j''\in[n]} \Pr\left[\,h(j') = h(j'') = h(j)\,\right]\cdot \mathbb{1}_{j'=j''}\cdot f_{j'}f_{j''} \\
&\qquad\qquad (\text{as before, } \mathbb{E}[g(j')\cdot g(j'')] = 0 \text{ if } j' \neq j'') \\
&= \sum_{j'\in[n]} \Pr\left[\,h(j') = h(j)\,\right]\cdot f_{j'}^2 \\
&= 1\cdot f_j^2 + \sum_{j'\in[n]\setminus\{j\}} \frac{1}{k}\cdot f_{j'}^2 \\
&= f_j^2 + \frac{\|f_{-j}\|_2^2}{k},
\end{aligned}$$

where the second-to-last step comes from drawing h from a strongly independent hash family: of course, if $j = j'$ then $h(j) = h(j')$ (always), but otherwise this happens with probability $1/k$ since $h(j')$ is uniformly distributed. This tells us that

$$\begin{aligned}
\mathrm{Var}[\widehat{f}_j] &= \mathbb{E}\left[\widehat{f}_j^2\right] - \mathbb{E}\left[\widehat{f}_j\right]^2 \\
&= f_j^2 + \frac{\|f_{-j}\|_2^2}{k} - f_j^2 \\
&= \frac{\|f_{-j}\|_2^2}{k}. \qquad (62)
\end{aligned}$$

Having the expectation and variance, we can conclude, by Chebyshev's inequality, that, for any fixed $j \in [n]$,

$$\begin{aligned}
\Pr\left[\left|\widehat{f}_j - f_j\right| > \varepsilon\|f_{-j}\|_2\right] &= \Pr\left[\left|\widehat{f}_j - \mathbb{E}\left[\widehat{f}_j\right]\right| > \varepsilon\|f_{-j}\|_2\right] \\
&\qquad\qquad (\text{Eq. (61)}) \\
&\leq \frac{\mathrm{Var}[\widehat{f}_j]}{\varepsilon^2\|f_{-j}\|_2^2} \\
&= \frac{1}{k\varepsilon^2} \qquad (\text{Eq. (62)}) \\
&\leq \frac{1}{3}
\end{aligned}$$

the last inequality for any choice of $k \geq \lceil 3/\varepsilon^2 \rceil$. □

Count-Min-Sketch

Chapter 5.4 of [AC]

Let us now cover a different (but similar-looking) algorithm with different guarantees, due to Cormode and Muthukrishnan, stated here in the cash register model: CountMinSketch.

Algorithm 20: The CountMinSketch algorithm

Input: Parameters $\varepsilon, \delta \in (0, 1]$

1: Set $k \leftarrow O(1/\varepsilon)$ and $T \leftarrow O(\log(1/\delta))$, and initialize a two-dimensional array C of size $T \times k$ to zero

2: Pick $h_1, \dots, h_T \colon [n] \to [k]$ independently from a strongly universal hashing family

3: **for all** $1 \leq i \leq m$ **do**

4: Get item $a_i = (j, c) \in [n] \times \{0, \dots, B\}$ ▷ Assume $B = O(1)$

5: **for all** $1 \leq t \leq T$ **do**

6: $C[t][h_t(j)] \leftarrow C[t][h_t(j)] + c$

Output: On query $j \in [n]$, **return** $\widehat{f}_j \leftarrow \min_{1 \leq t \leq T} C[t][h_t(j)]$

Can you see why?

Fact 44.1. CountMinSketch *is a linear sketching algorithm, provided the sketches C_1, C_2 are built using the same hash functions $h_1, \dots, h_T$.*

No median trick! The "probability amplification" is built-in, can you see where?

Theorem 45. *The* CountMinSketch *algorithm is a* randomised *one-pass sketching algorithm which, for any given parameters $\varepsilon, \delta \in (0, 1]$, provides a (succinctly represented) estimate $\widehat{f}$ of frequency vector f of the stream such that, for every $j \in [n]$*

$$\Pr\left[\left|\widehat{f}_j - f_j\right| \leq \varepsilon \|f_{-j}\|_1\right] \geq 1 - \delta$$

with space complexity

$$s = O\left(\frac{\log(nm)}{\varepsilon} \log \frac{1}{\delta}\right).$$

(Moreover, $\widehat{f}_j$ is always an overestimate: $\widehat{f}_j \geq f_j$ for all $j \in [n]$.)

But... is that not basically a similar guarantee as the Misra–Gries algorithm, but strictly worse? More space, and now it has a probability of failure!

> True, but compared to the Misra–Gries algorithm,
>
> - CountMinSketch is *much* faster and simpler
> - It provides a *linear* sketch, much easier to combine
> - It can be extended to the *strict* turnstile model.

See tutorial for this last point!

Proof of Theorem 45 (Sketch). The key steps of the analysis are as follows:

- the space complexity is

$$\begin{aligned} s &= T \cdot (O(\max(\log n, \log k)) + O(k \cdot \log(Bm))) \\ &= O(Tk \log(nm)) \end{aligned}$$

accounting for the T hash functions and storing T arrays of k numbers between 0 and Bm.

- In the cash register model, each update in Line 6 is a non-negative number: and so, for every $1 \le t \le T$ and every $j \in [n]$

$$C[t][h_t(j)] = f_j + \underset{\substack{\text{elements } j' \text{ with} \\ h_t(j')=h_t(j)}}{\text{contributions from other}} \ge f_j$$

and so

$$\widehat{f}_j = \min_{1 \le t \le T} C[t][h_t(j)] \ge f_j \,.$$

- Fix any $1 \le t \le T$. For every $j \in [n]$,

$$\begin{aligned} \mathbb{E}[C[t][h_t(j)]] &= \mathbb{E}\left[f_j + \sum_{j' \in [n] \setminus \{j\}} \mathbb{1}_{h(j')=h(j)} f_{j'} \right] \\ &= f_j + \sum_{j' \in [n] \setminus \{j\}} \Pr\left[h(j') = h(j) \right] f_{j'} \\ &= f_j + \frac{1}{k} \sum_{j' \in [n] \setminus \{j\}} f_{j'} \\ &= f_j + \frac{\|f_{-j}\|_1}{k} \end{aligned}$$

using that h is a strongly universal family, and therefore $h(j')$ is uniformly distributed in $[k]$ for any j'.

- For any fixed $j \in [n]$ and $1 \le t \le T$, let $X_{t,j} := \mathbb{E}[C[t][h_t(j)]] - f_j \ge 0$. We just showed that

$$\mathbb{E}\left[X_{t,j}\right] = \frac{\|f_{-j}\|_1}{k} \,.$$

By Markov's inequality (importantly, using $X_{t,j} \ge 0$ as proven above), this implies that

$$\Pr\left[\widehat{f}_j - f_j \ge \varepsilon \|f_{-j}\|_1 \right] \le \frac{1}{\varepsilon k} \le \frac{1}{2} \qquad (\star)$$

the last inequality as long as $k \ge \lceil 2/\varepsilon \rceil$.

- Fix $j \in [n]$. We want to bound the quantity

$$\Pr\left[|\widehat{f}_j - f_j| \ge \varepsilon \|f_{-j}\|_1 \right] = \Pr\left[\widehat{f}_j - f_j \ge \varepsilon \|f_{-j}\|_1 \right] = \Pr\left[\min_{1 \le t \le T} X_{j,t} \ge \varepsilon \|f_{-j}\|_1 \right]$$

A nice fact about the minimum of T random variables is that for the minimum to be greater than some value, *all T of them* need to be greater than this value. And since our T random variables are

independent, the expression simplifies a lot:

$$\begin{aligned}
\Pr\left[\min_{1\le t\le T} X_{j,t} \ge \varepsilon\|f_{-j}\|_1\right] &= \Pr\left[\forall 1 \le t \le T,\ X_{j,t} \ge \varepsilon\|f_{-j}\|_1\right] \\
&= \prod_{t=1}^{T} \Pr\left[X_{j,t} \ge \varepsilon\|f_{-j}\|_1\right] \\
&\qquad\qquad \text{(by independence)} \\
&\le \prod_{t=1}^{T} \frac{1}{2} = \frac{1}{2^T} \qquad\qquad \text{(by } (\star)\text{)} \\
&\le \delta \qquad\qquad \text{(by our choice of } T\text{)}
\end{aligned}$$

This proves the theorem. □