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`.

Sketching

Chapter 5.2 of [AC]

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.

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.

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"

Algorithm 17: The CountSketch algorithm

```
Input: Parameter $\varepsilon \in (0,1]$
1: Set $k \leftarrow O(1/\varepsilon^2)$, and initialize an array $C$ of size $k$ to zero
2: Pick $h\colon [n] \to [k]$ from a strongly universal hashing family
3: Pick $g\colon [n] \to \{-1,1\}$ from a strongly universal hashing family
4: for all $1 \leq i \leq m$ do
5:     Get item $a_i = (j,c) \in [n] \times \{-B,\dots,B\}$    ▷ Assume $B = O(1)$
6:     $C[h(j)] \leftarrow C[h(j)] + c \cdot g(j)$
Output: On query $j \in [n]$, return $\widehat{f}_j \leftarrow g(j) \cdot C[h(j)]$
```

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.*

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 $j \in [n]$*

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

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*

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

with space complexity

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

Count-Min-Sketch

Chapter 5.4 of [AC]

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

Algorithm 18: 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 $\widehat{f}_j \leftarrow \min_{1 \leq t \leq T} C[t][h_t(j)]$
```

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

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 $j \in [n]$*

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

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

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:

- The algorithm 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!