CS 229r Information Theory in Computer Science April 11, 2019

Lecture 20

Instructor: Madhu Sudan *Scribe: Boriana Gjura*

1 Today: Streaming Algorithms

Moving away from communication complexity, in this lecture we explore other applications of information theory. In particular, we introduce streaming algorithms and illustrate the model by elegant algorithms that compute the frequency of moments of an input stream. We conclude by proving lower bounds, relying on results from multi-party lower bounds of set-disjointness.

2 Model

Streaming algorithms process an input stream, given as a sequence of variables, which can be examined in *only a few passes*. Unless otherwise noted, we will consider single-pass algorithms only. Such algorithms find many practical applications in networking, such as in routers for internet traffic, computing popular IT addresses, etc.

We are mainly interested in the computational power of this model when $S = poly(\log m, \log n)$.

```
Streaming Model:
    Given: An input stream x_1, x_2, ..., x_m, where x_i ∈ [n] for all i ∈ [m].
    Goal:  Compute f(x_1, ..., x_m) for some f : [n]^m → Γ.
    Restriction: Small space S.
        Algorithm:
            State update:    A : {0,1}^S × [n] → {0,1}^S
            Output function: g : {0,1}^S → Γ
```

A few notes on randomness.

The state update algorithm A is wlog deterministic, for if necessary, we can include the random strings that we used in the description of the respective states. We do not worry about how the computational complexity of A.

We want to derive $f(x_1, \ldots, x_m)$ from σ_m, such that $f(x_1, \ldots, x_m) = g(\sigma_m)$, where

$$\begin{aligned} \sigma_0 &= \overline{0}, \\ \sigma_i &= A(\sigma_{i-1}, x_i), \quad i \in [m]. \end{aligned}$$

In randomized streaming algorithms we can allow σ_0 to be random, in which case we will require only that the algorithm is correct most of the time, say:

$$\mathbb{P}_{\sigma_0}[f(x_1, \ldots, x_m) = g(\sigma_m)] \leq \frac{2}{3}.$$

3 Example algorithms: frequency moments

The frequency of $i \in [n]$ is defined naturally as $f_i(x_1, \ldots, x_m) = |\{j \mid x_j = i\}|$. We want to compute the k^{th} moment of frequency in the streaming model, where $k \in \mathbb{N}$ for simplicity (roughly, most arguments given here should work for fractional k as well.)

`K-Moment of Frequency:`
`Compute` $F_k(x_1, \ldots, x_m) = \sum_{i \in [n]} f_i^k(x_1, \ldots, x_m)$.

We first present elegant algorithms for $k = 0, 2$, and then give a clever algorithm for higher values of k. We will use a lot of randomness in the following algorithms (you will see this when we pick uniformly random hash functions!). This is not necessary and can be fixed.

Trivially, $F_1 = \sum_{i \in [n]} f_i(x_1, \ldots, x_m) = \sum_{i \in [n]} |\{j \mid x_j = i\}| = m$ corresponds to *the number of elements in the input stream.* A more interesting example is the 0^{th} moment, which corresponds to *the number of distinct elements in the stream*, defined as:

$$\begin{aligned} F_0(x_1, \ldots, x_m) &= \lim_{k \to 0} F_k(x_1, \ldots, x_m) \\ &= |\{i \mid f_i > 0\}| \end{aligned}$$

Can we compute k^{th} moments with non-trivial space?

3.1 Quick overview

The algorithms we will see today have the following general structure.

- Find an unbiased estimator of F_k, which might have a high variance.
- Run the algorithm a (constant) $\times$ variance times.
- Report the median, using concentration inequalities this gives us the bounds we need.

Brief summary		
trivial	$F_1 = m$	
'85, Flajolet-Martin, [1]	approximate F_0	polylog(n,m)
'86, Alon-Matias-Szegedy, [2]	approximate F_2	polylog(n,m)
'98, Alon-Matias-Szegedy, [2]	approximate F_k	$n^{1-1/k}$
'03, BJKS,[3]	approximate F_k, single pass	lower bound: $n^{1-2/k}$
'03, BJKS, [3]	approximate F_k, multi pass	lower bound: $n^{1-3/k}$

From now on, we will fix a stream $x_1, \ldots, x_m$.

3.2 Algorithm for F_0.

`[Flajolet-Martin '85]`
`Pick random` $h : [n] \to [0,1]$ `uniformly at random.`
`Compute` $\min_{j\in[m]}\{h(x_j)\} = h_{min}$.
`Output:` $\frac{1}{h_{min}} - 1$.

The intuition is that if h is uniform, the m inputs are (roughly) uniformly distributed in $[0,1]$, which then implies that the number of distinct elements is, in expectation, $\frac{1}{h_{min}} - 1$. Formally, h_{min} is equal to $0 \le h \le 1$ with probability $h(1-h)^{F_0-1}$, that is, $h_{min} = h$ and all other $F_0 - 1$ positive values are greater than h. Then

$$\mathbb{E}\left[h_{min}\right] = F_0 \int_0^1 \frac{1}{h(1-h)^{F_0-1}} dh = F_0 \frac{1}{F_0(F_0+1)} = \frac{1}{F_0+1}.$$

Can we get rid of the expectation? Remember that the exponential distribution corresponds is the probability distribution that describes the time between events in a Bernoulli process. This way of thinking about it is useful because the process is memoryless (nothing from the past carries to the future). This is particularly useful when we are doing minimum analysis, because the minimum will be distributed as $exp(1/p)$, where p is the success probability of the Bernoulli process. To make the above algorithm cleaner, we can take $h(i)$ i.i.d. $\sim exp(1)$, where 1 is the mean of the exponential random variable. With k distinct elements, the minimum of the $h(i)$ is distributed as $exp(k)$, which has expectation $\frac{1}{k}$, so that the algorithm outputs $\frac{1}{1/k} = k$.

This leads to an $O(\frac{1}{\varepsilon^2}\log^c n)$ algorithm to output $(1\pm\varepsilon)$-approximation to F_0. To get a sense of how efficient this algorithm is: Gap-Hamming distance lower bounds imply a lower bound of $\Omega(\frac{1}{\varepsilon^2})$ for computing $(1\pm\varepsilon)$-approximate F_0.

Slightly better algorithm? Instead of h_{min}, we can pick the t^{th} smallest and output $\frac{t}{h_{t_{min}}}$.

3.3 Algorithm for F_2

In a similar spirit, we have the following algorithm for F_2.

`[Alon-Matias-Szegedy]`
`Pick` $h : [n] \to \{+1,-1\}$ `uniformly at random.`
`Compute` $V = \sum_{j=1}^m h_{min}$.
`Output:` V^2.

The calculation below shows that $F_2(x_1,\ldots,x_m) = \mathbb{E}[V^2]$.

$$\begin{aligned}
\mathbb{E}[V^2] &= \mathop{\mathbb{E}}_h\Big[\big(\sum_{j=1}^m h(x_j)\big)\big(\sum_{k=1}^m h(x_k)\big)\Big] = \sum_{j=1}^m\sum_{k=1}^m \mathbb{E}_h\big[h(x_j)h(x_k)\big]\Big] \\
&= \sum_{i=1}^n\sum_{j=1}^m\sum_{k=1}^m \mathbb{1}_{x_j=x_k=i} = \sum_{i=1}^n\Big(\sum_{j=1}^m \mathbb{1}_{x_j=i}\Big)^2 \\
&= \sum_{i=1}^n f_i^2 = F_2
\end{aligned}$$

Note that above $\mathbb{E}[h(x_j)h(x_k)] = 1$ if $x_j = x_k$, 0 otherwise.

3.4 Arbitrary k

What about F_3, F_4 and so on? [1] We cannot use the same tricks as earlier, since now when we take higher powers, we will have to worry about a lot of low-order terms. (Remember that for $k = 2$, $\mathbb{E}[h(x_j)h(x_k)] = 0$ unless $x_j = x_k$, which made our argument go through.) The algorithm before has a different flavor.

```
[AMS Algorithm]
    Pick j ∈ [m] uniformly at random.
    Let i = x_j.
    Let r_i = |{j' | x_j' = i, j' ≥ j}| \\count how many times it appears later
    Output: X_j = m(r_i^k − (r_i − 1)^k).
```

We can verify that $\mathbb{E}[X_j] = F_k$. A somewhat intuitive way to view this AMS estimator is to interpret $r^k - (r-1)^k$ as the rate of change of F_k at time j, and then "integrate" backwards from $j = m$ to $j = 0$ (which, in expectation, is the same as sampling $j \sim [m]$ uniformly). The final sum telescopes so we get

$$\mathbb{E}[X_j] = \sum_{i\in[n]} (1^k - 0) + (2^k - 1^k) + \cdots + (r_i^k - (r_i - 1)^k) = \sum_{i\in[n]} f_i^k = F_k,$$

One can also check that $V[X] \approx n^{1-\frac{1}{k}}$, which explains the running time. We give a proof sketch from 3.1, and refer the reader to the paper for a full proof.

$$\frac{\mathbb{E}[X_j^2]}{m} = \sum_{i\in[n]} 1^{2k} + (2^k - 1^k)^2 + \cdots + (f_i^k - (f_i - 1)^k)^2$$

This series is not telescoping anymore, but we can turn it into one by using the inequality

$$a^k - b^k \le (a-b)ka^{k-1}, \quad \forall a > b > 0,$$

which follows directly from the identity $a^k - b^k = (a-b)\sum_{t=1}^{k-1} a^t b^{k-t}$. This gives

$$\begin{aligned}\frac{\mathbb{E}[X_j^2]}{k} &= m\Big(\sum_{i\in[n]} 1^{2k-1} + 2^{k-1}(2^k - 1^k) + \cdots + f_i^{k-1}(f_i^k - (f_i-1)^k)\Big) = m\sum_{i\in[n]} f_i^{2k-1} \\ &= F_1 F_{2k-1} \le n^{1-1/k} F_k^2\end{aligned}$$

where the last inequality is an application of the Hölder Inequality.

As described earlier, we reduce the variance by repeating the algorithm $\sim kn^{1-\frac{1}{k}}$ times, and choosing the mean.

4 Lower bounds

We now shift our attention to lower bounds for the problems we just considered. We rely heavily on the lower bounds of set disjointness shown in [3].

[1] This is well-defined for fractional k as well, but in this lecture we will take $k \in \mathbb{N}$ for simplicity.

4.1 t -party communication model

The model consists of t parties: $P_1, P_2, \ldots, P_t$ that share a common randomness R. P_1 has access to x_1, and communicates m_1 to P_2. For all other i, P_i has access to X_i and m_{i-1}, and communicates m_i to P_{i+1}. At the end, P_t communicates m_t. It is natural to think of $x_1, \ldots, x_t$ as coming from an input stream, and of m_t as $m_t = f(x_1, \ldots, x_t)$.

We say the model is one way if the communication happens in this form: $P_1 \to P_2 \to \ldots P_t$. We say the model is r-pass one-way if the communication looks like $P_1 \to \ldots P_t \to P_1 \to \ldots P_t \ldots P_1 \to \ldots P_t$, r times.

A streaming algorithm can be simulated as a t-party protocol, where the stream into t pieces and P_i gets the i^{th} piece. We will see how we can simulate the computation of F_k in this model later. First, we introduce the general version of set-disjointness and related hardness results which we will later use to show our lower bounds.

4.2 t-party lower bounds of t-set-disjointness

We will now define a stronger notion of set disjointness.

[t-set-disjointness]
YES: $x_1, \ldots, x_t \in \{0,1\}^n$
and $\exists$ i such that $x_1^{(i)} \ldots x_t^{(i)} = 1$.
and $\forall i' \neq i$, $\sum_{j=1}^{t} x_j^{(i')} \leq 1$, otherwise disjoint.
NO: $x_1, \ldots, x_t \in \{0,1\}^k$, $\forall i : \sum_{j=1}^{t} x_j^{(t)} \leq 1$. \\ strongly disjoint.
Distinguish YES from NO.

We will use the following theorem, without proof.

Theorem. One way CC $\geq \Omega(\frac{n}{t})$. Arbitrary CC $\geq \Omega(\frac{n}{t^2})$. (Theorem 7.1, [3])

4.3 Streaming Lower Bound for F_3

Take any single-pass streaming algorithm that computes F_k and uses space c. Set $t = (2n)^{1/k}$, we will see shortly why this is useful. As we described above, we can describe the streaming algorithm as a t-party t-set-disjointness communication problem, with communication bounded by ct.

When the t-set-disjointness algorithm outputs YES: $F_k \geq t^k$, i.e. there is some element in all t sets, and therefore $F_k \geq t^k \geq 2n$ by our choice of t. Since the communication complexity ct is lower bounded by $\Omega(\frac{n}{t})$, $c \geq \Omega(\frac{n}{t^2})$. Again by our choice of t, we get the lower bound:

$$c \geq \Omega(n^{1-\frac{2}{k}}.)$$

On the other end, when the t-set-disjointness algorithm outputs NO: $t_i \leq 1$ for all i, which implies $F_k \leq n$.

This completes our analysis of lower bounds.

References

[1] Philippe Flajolet and Nigel G. Martin. Probabilistic counting algorithms for data base applications.*Journal of computer and system sciences* 31.2:182-209, 1985
`http://algo.inria.fr/flajolet/Publications/src/FlMa85.pdf`

[2] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences.* 58(1):137147, 1999
`https://dl.acm.org/citation.cfm?id=237823`

[3] Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences* 68 (2004) 702–732
`http://people.seas.harvard.edu/~madhusudan/courses/Spring2016/papers/BJKS.pdf`