CS 368 (Stanford, Spring 2022) Lecture 3

Adapted from CS 369G, Spring 2016. Scribes: Stephen Mussman, Spencer Yee, Michael Xie, Paris Syminelakis.

Lower bounds for streaming; Frequency moments

1 Overview

In this lecture, we present a proof technique to prove lower bounds for streaming algorithms. We define the concept of a frequency moment and propose an algorithm to estimate F_2. Finally, we discuss how to implement efficient hash functions for the AMS second frequency moment estimator and the notion of linear sketches.

1.1 Proving Lower Bounds on Streaming Algorithms

In this section we describe a common technique for deriving bounds for streaming problems. For this technique, we specify $I_1, ..., I_N$ different streams. If we can show that the algorithm must have a unique state after processing each stream, $\Omega(\log(N))$ is lower bound on the space since the algorithm must distinguish the N different streams.

In order to show that the algorithm must have different states after processing two streams I_i and I_j, we can construct an additional stream I' such that the algorithm must give a different result for $I_i \cup I'$ than for $I_j \cup I'$.

Using this technique, we can prove a lower bound for a deterministic, approximate algorithm for the distinct element problem. Let $\{S_i\}_{i=1}^N$ be subsets of $[n]$ that satisfy

$$\forall i : |S_i| = \frac{n}{10} \tag{1}$$

$$\forall i \neq j : |S_i \cap S_j| \leq \frac{n}{20} \tag{2}$$

With a probabilistic construction, we can find 2^{cN} subsets that meet these requirements.

Note that the number of distinct elements in $S_i \cup S_i$ is $(n/10)$ while the number of distinct elements in $S_i \cup S_j$ (for $i \neq j$) is at least $(3/2)(n/10)$. Thus, any deterministic, approximate algorithm must distinguish the N streams corresponding to the sets S_i. Thus, $\Omega(\log(2^{cN})) = \Omega(n)$ is a lower bound on the required space.

1.2 Algorithms for Frequency Moments

Define f_i as the number of times that element i appears in a stream.

The t^{th} frequency moment is the quantity,

$$F_t = \sum_i f_i^t \tag{3}$$

Note that F_0 is the number of distinct elements (assuming $0^0 = 0$). F_1 is simply the number of elements in the stream, and thus is trivial to compute. F_2 is a measure of skewness and is the problem we will next examine, which has a solution given in [AMS99].

Suppose we have a hash function $h : U \to \{\pm 1\}$. Let $Y = \sum_i h(x_i)$ be a random variable. This variable is a sketch because we can simply add the sums of two different sets. Further, Y^2 is an unbiased estimate of F_2.

To see this, define the random variable $X_i = h(x_i)$ and note that $Y = \sum_i f_i X_i$

$$\mathbb{E}[Y^2] = \mathbb{E}[(\sum_i f_i X_i)^2] \tag{4}$$

$$= \mathbb{E}[\sum_{i,j} f_i f_j X_i X_j] \tag{5}$$

$$= \sum_{i,j} f_i f_j \mathbb{E}[X_i X_j] \tag{6}$$

$$= \sum_i f_i^2 \tag{7}$$

The last equation follows from the fact that $\mathbb{E}[X_i X_j] = 0$ if $i \neq j$.

We need a bound on the variance in order to obtain a concentration inequality.

$$\mathbb{E}[Y^4] = \mathbb{E}[(\sum_i f_i X_i)^4] = \sum_i f_i^4 + 6 \sum_{i,j} f_i^2 f_j^2 \tag{8}$$

Thus,

$$Var[Y^2] = \mathbb{E}[Y^4] - \mathbb{E}[Y^2]^2 \tag{9}$$

$$= [\sum_i f_i^2 + 6 \sum_{i,j} f_i^2 f_j^2] - [\sum_i f_i^2 + 2 \sum_{i,j} f_i^2 f_j^2] \tag{10}$$

$$= 4 \sum_{i,j} f_i^2 f_j^2 \leq 2F_2^2 = 2\mathbb{E}[Y^2]^2 \tag{11}$$

Thus, we can use Chebyshev's inequality to establish concentration around the mean and then use the median of means technique.

2 Second Frequency Moments

In the previous section, we saw an elegant estimator for the second frequency moment for a stream. We will review the details of this estimator. We require a hash function $h : [n] \to \{\pm 1\}$ which we assume to be random, and we define the frequency f_i as the number of copies of element i in a stream, so the second frequency moment F_2 is Σf_i^2. Start with a counter Y initialized to 0, and add $h(i)$ to Y for each element i in the stream. In the end, $Y = \Sigma f_i h(i)$, and Y^2 is our estimator for F_2.

For now, think of $h(i)$ as a random variable x_i. Then $Y = \Sigma f_i x_i$, and as shown in the previous lecture, $E[Y^2] = F_2$ and $\text{Var}[Y^2] \leq 2F_2^2$, the desired properties for our estimator. The key observations that yielded those results were $E[x_i^2] = 1$, $E[x_i] = 0$, $E[x_i x_j] = 0$, and what the value of $E[x_{i_1} x_{i_2} x_{i_3} x_{i_4}]$ is, depending on the number of distinct x_i. Ignoring the problem of how to implement h, Y requires very little space, has the right expectation, and a small variance, so we can apply median of means. This means that in $O\left(\frac{1}{\epsilon^2} \log\left(\frac{1}{\delta}\right)\right)$ space, we can estimate F_2 within ϵ error with probability at least $1 - \delta$.

Now we address the issue of how to implement the hash function. We've assumed that the hash function is random, but in order to implement a random hash function, we essentially need to store the hash value for every element in the set, which is n bits. So we can't use a completely random hash function. Looking back at our key observations, though, the properties of randomness that we require can be satisfied with a 4-wise independent hash function, where

$$\Pr[h(x_1) = a_1, h(x_2) = a_2, h(x_3) = a_3, h(x_4) = a_4] = \prod_{i=1}^{4} \Pr[h(x_i) = a_i].$$

There are 4-wise independent hash functions that only require $O(\log n)$ space and $O(\log n)$ evaluation time, so we can indeed estimate F_2 with a small amount of space.

2.1 Linear Sketches

This is an example of a linear sketch. We can think of the input stream as a vector $x \in \mathbb{R}^n$ where the i-th element of x is f_i, where each element in the stream updates x. Then we can represent that d copies of Y that we are storing as Ax, where $A \in \mathbb{R}^{d \times n}$. Each row of A represents a different 4-wise independent hash function, and the i-th entry in a row representing hash function h is simply $h(i)$. We don't actually store the entire matrix A; instead, we have an implicit representation that allows us to compete each coordinate when needed. Every time an element in the stream appears, we update x with Δx, a vector with 1 in one coordinate and 0s in the others. The sketch is incremented by $A\Delta x$, which is equivalent to updating our copies of Y with the appropriate hash values of the new element. The fact that this is a linear sketch is useful, since we can use operations such as addition and subtraction. Combining linear sketches is very simple, since $Ax + Ay = A(x + y)$.

We can also think of the input as being updates to coordinates rather than elements. In the turnstile model, the input is x, and updates are changes to one coordinate of x, which need not be +1. Linear sketches support this. We can also have negative updates, which are also supposrted by linear sketches.

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating the frequency moments". In: *Journal of Computer and system sciences* 58.1 (1999), pp. 137–147.