CS 229r Information Theory in Computer Science March 25, 2016

Lecture 16

Lecturer: Madhu Sudan *Scribe: Jack Murtagh*

1 Streaming Lower Bounds - Introduction to Model

Streaming algorithms are typically used in settings where we want to compute or approximate a function on data that is too large to fit in one computer. We should think of the data as so large that the algorithm can only make one pass over the input, i.e. it is "streaming" in, one element at a time. The goal of streaming algorithms is to minimize the space used (as a function of the input size) while approximating the target function as best as possible. One example application of streaming algorithms is a router, where we have a large amount of data flowing through a small device and we may want to track some statistics about the data. We will survey some fundamental results in this area and discuss lower bounds that can be proved with techniques from communication complexity.

To make this more precise, let $x_1, x_2, \ldots, x_m$ be the input, which we will imagine as very large and streaming into our algorithm one element at a time. Our goal is to compute or approximate a function $f(x_1, x_2, \ldots, x_m)$ while only using $O(\log^c m)$ space for some constant c. One of the great surprises in streaming algorithms is that there is actually a relatively large collection of tasks for which this task can be accomplished effectively.

2 Algorithms for Frequency Estimation

One of the seminal papers in the theory of streaming algorithms is due to Alon, Matias and Szegedy [1] on the frequency estimation problem. Given input $x_1, x_2, \ldots, x_m$, where $x_i \in [n]$ for all i, define the frequency vector $(f_1, f_2, \ldots, f_n)$ where f_i is the number of occurrences of i in the data stream. In other words

$$f_i \triangleq |\{j \mid x_j = i\}|$$

The goal is to compute the kth frequency moment F_k of the stream, defined as

$$F_k(x_1, x_2, \ldots, x_m) \triangleq \sum_{i=1}^{n} f_i^k$$

Note that

- F_1 is always equal to m and this is trivially countable in logspace.
- $F_0 = |\{i \mid \exists j \text{ s.t } x_j = i\}|$ = the number of distinct elements in the stream. For this interpretation to really make sense, we should think of $F_0 = \lim_{k \to 0} F_k$. There is a known randomized algorithm for approximating F_0 [4].

Alon, Matias, Szegedy gave an elegant randomized algorithm for approximating F_2, which can be thought of as the variance of the data stream. For all the algorithms presented here, we will solve a weaker version of the problems by assuming that the streaming algorithms have access to some very powerful hash function that is more or less random. Everything shown here can be cleaned up though by using other hash function constructions. This is important because we do not want to use a hash function that requires us to store a huge table because the whole point of these algorithms is to conserve space.

The streaming algorithm works as follows. Associate with $[n]$ random bits $R_1, R_2, \ldots, R_n$ with $R_i \in \{-1, +1\}$ for all $i \in [n]$. Instead of directly computing $\sum_{i=1}^{n} f_i^2$, we will compute

$$A = \sum_{j=1}^{m} R_{x_j} \cdot x_j$$

This is just a linear function and can clearly be computed efficiently assuming that we have oracle access to the sequence of random bits. Then we just output A^2. The point is that

$$\begin{aligned} A^2 &= \left(\sum_{j=1}^{m} R_{x_j} \cdot x_j \right)^2 \\ &= \sum_{1 \le i,k \le n} R_i R_k f_i f_k \\ &= \sum_{i=1}^{n} f_i^2 + \sum_{i \neq k} R_i R_k f_i f_k \end{aligned}$$

The first term of the last line is exactly what we wish to compute and we can view the second term as noise in our approximation. When you analyze the expectation and variance of the noise quantity and apply Chebyshev's inequality, you find that the term goes to zero. In [1] they also show lower bounds for frequency estimation. For example, they prove that F_p estimation for $p > 2$ requires $\Omega(n^{1-2/p})$ space, showing that estimating higher moments incurs higher space complexity. Note that F_p estimation is trivial using $O(n)$ space because you can just store the entire frequency vector.

In 1985, Flajolet and Martin gave a beautiful algorithm for F_0 estimation [4]. Begin by picking a random hash function

$$h \colon [n] \to [0, 1].$$

In the sequence $h(x_1), h(x_2), \ldots h(x_m)$, let

$$h_{\min} \triangleq \min_{j \in [m]} h(x_j)$$

and output $\frac{1}{h_{\min}}$. This is easy to compute because as the stream comes in we compute $h(x_i)$ for all i and we only need to maintain the minimum. If the hash value of the incoming element is smaller than our current stored minimum, we just replace it with the new value. This works because if we have say, elements from the set $\{1, 2, \ldots, 10\}$ then the expected minimum hash value in $[0, 1]$ should be around $1/10$. Outputting the reciprocal of this then gives us the number of distinct elements in the stream. The original versions of this algorithm got results of the form:

$$\Pr\left[\text{output } \notin [\frac{1}{c} \cdot F_0, c \cdot F_0]\right] \le \frac{1}{c}$$

We can actually do better if we don't just track the minimum hash value but rather the t smallest hash values of the stream [2]. Then if h_t is the tth smallest hash value, output $\frac{h_t}{h_{\min}}$. This estimator is more tightly concentrated around the target value than the first approach of just outputting the minimum. It turns out that this algorithm gives a $\theta(1/\sqrt{t})$ approximation to F_0. Setting t to $1/\epsilon^2$ then gives us a $(1 \pm \epsilon)$ approximation to F_0 and the algorithm uses space $O\left(\frac{1}{\epsilon^2} \cdot \text{polylog}(m)\right)$. The latest results are able to shave the polylog(m) in the space complexity to a polyloglog(m) but unfortunately we will show in the next section that the $1/\epsilon^2$ factor for a $(1 \pm \epsilon)$ approximation is optimal.

3 F_0-Estimation Lower Bounds via Communication Complexity

In general, lower bounds in communication complexity naturally translate to lower bounds in streaming algorithms. These are often proved using the contrapostive - that upper bounds in streaming imply upper bounds in communication complexity. To draw the connection between the two, think of streaming as an m-player game where player x_1 sends s_1 to player x_2, who sends s_2 to player x_3 etc. We want s_m to be a good approximation of $f(x_1, \ldots, x_m)$, the function we wish to compute. Amazingly, we can simplify this to the case of just two player communication by splitting the stream in half and we still get strong results. Specifically, we will think of Alice receiving as input $x_1, \ldots, x_{m/2}$ and Bob receiving $x_{m/2+1}, \ldots x_m$. Alice then computes some function of her inputs and sends a message to Bob, who must then output a value for the function. This further restriction is called one-way communication and the idea is that a space S algorithm for the streaming problem implies an S bit protocol for the analogous communication problem.

3.1 Reduction to Gap Hamming

We will prove the desired lower bound by a reduction to the Gap Hamming Problem [5]. In this problem, Alice recieves $x \in \{-1, +1\}^n$ and Bob receives $y \in \{-1, +1\}^n$. The goal is to decide whether:

$$\begin{aligned} \langle x, y \rangle &\geq +g \\ \text{or } \langle x, y \rangle &\leq -g, \end{aligned}$$

where $\langle x, y \rangle$ is the inner product between x and y:

$$\langle x, y \rangle \triangleq \sum_{i=1}^{n} x_i y_i$$

Obviously the inner product lies between $-n$ and n and roughly what this question is asking is how often the strings x and y agree. The larger the inner product, the more the two strings agree, the smaller it is, the more they disagree, and the closer it is to 0, the less correlation the strings have. There is a fairly straightforward reduction from the disjointness problem that implies that the Gap Hamming Problem with $g = 1$ requires $\Omega(n)$ communication.

For upper bounds, consider the extreme case where $g = \epsilon \cdot n$. A reasonable protocol is to sample a subset of the coordinates using correlated randomness and check the number of coordinates in the sample on which x and y agree (Alice just sends the x values on these coordinates). If we pick a large enough sample, the sample probability of agreement should closely approximate the total probability of agreement between the strings. Choosing $1/\epsilon^2$ coordinates suffices for good approximations so the communication complexity is $1/\epsilon^2$ when $g = \epsilon \cdot n$. In other words CC is $O((n/g)^2)$. We can also always take the naive approach where Alice just sends x to Bob to get CC of $\min\{O((n/g)^2), n\}$. This is an easy upper bound and it turns out this is actually tight, which means in the regime of $g = \sqrt{n}$, we get a communication complexity lower bound of $\Omega(n)$ [5]. This linear lower bound was for the one way version of the Gap Hamming Problem, which will suffice for our purposes but later on it was proven that in fact the general two way version of the problem requires $\Omega(n)$ communication [3, 6].

Now we show that a Gap Hamming lower bound implies a lower bound on F_0 estimation. The idea is that Alice has $x \in \{-1, +1\}$ and Bob has $y \in \{-1, +1\}$. Alice computes $S \subseteq [n] = \{i \mid x_i = 1\}$ and Bob computes $T \subseteq [n] = \{j \mid y_j = 1\}$. We want to write $\langle x, y \rangle$ as a function of $|S|, |T|$, and $|S \cup T|$, which will imply that if we can compute F_0 estimations, then we can approximate the sizes of these sets, which in turn allows us to compute the inner product. Notice that:

$$\langle x, y \rangle = |S \cap T| + (n - |S \cup T|) - |S \setminus T| - |T \setminus S|$$

After expressing each of the terms above in terms of just $|S|, |T|$, and $|S \cup T|$, we get the desired result and in fact our F_0 estimation will be $|S \cup T|$. Now we just need to show how Alice and Bob can estimate the size of $|S \cup T|$. The protocol starts with Alice sending $|S|$ to Bob, only using $O(\log n)$ bits. Then she creates a stream of elements of S followed by elements of T and Bob can calculate $|S \cup T|$. This is saying that if we can get a $\frac{\sqrt{n}}{n}$ approximation to $|S \cup T|$ in space $o(n)$, it would imply a $o(n)$ one-way communication protocol for the Gap Hamming Problem, which would contradict the known lower bound.

3.2 Reduction to Indexing

Finally, we will show the one-way communication lower bound on the Gap Hamming Problem. The idea is to reduce from the Indexing Problem. In this problem, Alice gets as input a string $z \in \{0,1\}^n$ and Bob gets an index $i \in [n]$. Their goal is to output z_i. This is clearly very easy to solve with two-way communication. Bob simply sends his coordinate to Alice using $\log n$ bits and she can output z_i. It turns out that this problem is hard for one-way communication.

Given an instance of Indexing, we want to show how to solve it using a protocol for Gap Hamming. Let R be a large $n \times n$ matrix of correlated randomness containing elements of $\{-1, +1\}$. Alice will compute $Rz = \tilde{x}$, a column vector of length n and then convert this to a vector x by taking the sign of each element in $\tilde{x}$ (if a coordinate is 0, just call it $+1$). Bob does the same thing with the vector e_i, containing all zeroes except a one in the ith coordinate. He computes $Re_i = y$. The claim is that:

$$\text{If } z_i = 1 \text{ then } \langle x, y \rangle \text{ is large w.h.p}$$
$$\text{If } z_i = 0 \text{ then } \langle x, y \rangle < 0 \text{ w.p } 1/2$$

To see this, we will start with the second case. Notice that y is just the ith column of R. So if $z_i = 0$ then x is completely independent of column i in R. So x and y are independent and therefore their inner product is negative with probability $1/2$.

If $z_i = 1$, then x now also picks up column i in the multiplication and we are asking what the probability is that the new vector x is correlated with column i = vector y. Let $R^{(i)}$ be R with the ith column removed and $z^{(i)}$ be z with the ith coordinate removed. Now we can write $\tilde{x}$ in the form

$$\tilde{x} = (R \cdot R^{(i)}) \cdot z^{(i)} + y$$

Again we convert $\tilde{x}$ to x by taking the sign of every element. We want to know how correlated this is to y and with what probability. Another way of asking this question is, when we add column y above, what is the probability that that flips the sign of coordinate j in x? Each entry of x before adding y is just the sum of independent random signs and so you expect the distribution to look like a Gaussian. So the probability that we actually flip the sign of coordinate j when adding y is $1/\sqrt{n}$ for every j. We are adding up n of these so we get that $\langle x, y \rangle = \frac{n}{\sqrt{n}}$ with high probability.

3.3 Complexity of Indexing

To complete the lower bound on F_0 estimation, we should observe the lower bound on the Indexing problem. First consider what can be done with a deterministic protocol. Alice sends some a message m to Bob. Suppose g is the optimal function for Bob to use on Alice's message. Bob can compute $g(m,1) = \tilde{z}_1, g(m,2) = \tilde{z}_2, \ldots, g(m,n) = \tilde{z}_n$. Clearly the only way to guarantee that Bob learns z_i, is if $\tilde{z}_j = z_j$ for all j. So there must be a one to one correspondence between possible z vectors and possible $\tilde{z}$ vectors. So if $m < n$ then by the Pigeonhole principle, there cannot be such a correspondence. This gives us the one-way deterministic lower bound of $\Omega(n)$.

If we allow randomness and some error we still have to have the mutual information between z and $\tilde{z}$ be large. Specifically $I(z, \tilde{z}) = \Omega(n)$. Since $H(m) \geq I(z, \tilde{z})$, we get that the length of Alice's message must be $\Omega(n)$ bits.

References

[1] 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

[2] Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. *Randomization and Approximation Techniques in Computer Science*. 1-10, Springer Berlin Heidelberg, 2002

[3] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM Journal on Computing* 41(5):1299-1317, 2012

[4] Philippe Flajolet and Nigel G. Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31.2:182-209, 1985

[5] Piotr Indyk and David P. Woodruff. Tight lower bounds for the distinct elements problem. *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science* 283289, 2003

[6] Alexander A. Sherstov. The communication complexity of gap hamming distance. *Theory of Computing*. 8(1):197-208, 2012.