CS 229r: Algorithms for Big Data Fall 2015

Lecture 03, September 11th

Prof. Jelani Nelson *Scribe: Vasileios Nakos*

1 Overview

In the last lecture we looked at distinct elements, k-wise independence, geometric subsampling of streams.

In this lecture we will see lower bounds on exact and approximate deterministic algorithms on data streams, as well as the AMS sketch and Indyk's algorithm for the F_p moments, when $0 \leq p \leq 2$.

2 Main Section

2.1 Space lower bounds for streaming algorithms

We begin by an impossibility result. We consider the distinct elements problems, where you want to find the number of distinct elements in a stream, where queries and additions are permitted. We will denote by s the space of the algorithm, n the size of the universe from which the elements come from, and m the length of the stream. We have the following result [1].

Theorem 1. *There is no deterministic exact algorithm for computing number of distinct elements in $O(minn, m)$ space.*

Proof. We are going to make an information-theoretic argument. Using a streaming algorithm with space s for the current problem we are going to show how to encode $\{0,1\}^n$ using only s bits. In other words, we are going to construct an injective mapping from $\{0,1\}^n$ to $\{0,1\}^s$. So, this implies that s must be at least n and we are done. We look for procedures Dec,Enc such that $\forall x Dec(Enc(x)) = x$ and $Enc(x)$ is a function from $\{0,1\}^n$ to $\{0,1\}^s$.
For the encoding procedure, given a string x, create a stream containing and append i at the end of the stream if $x_i = 1$. Then $Enc(x)$ is the memory content of the algorithm on that stream.
For the decoding procedure, we are going to look at each i and append it at the end of the stream(feed it to the streaming algorithm) and query then the number of distinct elements. If the number of distinct elements increases this implies that $x_i = 0$, otherwise it implies that $x_i = 1$. So we can recover x completely and this finishes the proof.

□

We move on by showing that even approximate algorithms are hopeless for this problem.

Theorem 2. *Any deterministic F_0 algorithm that provides* 1.1 *approximation requires $\Omega(n)$ space*

Proof. Suppose we had a colection C satisfying:

- $|C| \geq 2^{cn}$, for some constant $c < 1$.
- $\forall S \in C, |S| = \frac{n}{100}$
- $\forall S \neq T \in C, |S \cap T| \leq \frac{n}{2000} \leq \frac{1}{20}|S|$

We leave the proof of existence of such a set later and we are moving to our lower bound. We are going to use the algorithm to encode vectors $x_S \forall S \in C$, where x_S is the indicator vector of set S. The lower bound follows as before since we must have $s \geq cn$ The encoding procedure is going to be the same as before.
For the decoding procedure, we are going to iterate over all sets and test for each set S if it corresponds to our initial encoded set(remember we are just doing an information-theoretic argument and we do not care about the running running- we only care that this map is an injection). For that, we keep at each time the memory contents of M of the streaming algorithm after having inserted our initial string. Then for each S, we initialize our algorithm with memory contents M and then feed element i if $i \in S$. It is easy to see that if S equals the initial encoded set, the number of distinct elements does not increase by much, whereas if it is not it almost doubles. Taking also into account the approximation guarantee of the algorithm we see for example that if S is not our initial set then the number of distinct elements grows by $\frac{3}{2}$(we can tune the parametersif needed).

We now only need to prove the existence of such a family of sets C. We are going to argue via probabilistic method. We partition n into $\frac{n}{100}$ intervals of length 100 each in the obvious way. To form a set S we pick one number from each interval uniformly at random. Obviously, such a set has size exactly $\frac{n}{100}$. For two sets S, T chosen uniformly at random as before let U_i be the random variable that equals 1 if they have the same number chosen from interval i. Obviously $P[U_i = 1] = \frac{1}{100}$. So the expected size of the intersection is just $E \sum_{i=1}^{\frac{n}{100}} = \frac{n}{100} \cdot \frac{1}{100}$. The probability that this interesction is bigger than five times its mean is smaller than $e^{-c'n}$ for some constant c', by a standard Chernoff bound. We can then apply a union bound over all possible intersections and get the desired result.

□

2.2 Linear Sketches and upper bounds

2.2.1 What is a Linear Sketch?

We introduce the turnstile model in streaming algorithms. In this model we have a vector $x \in \mathbb{R}^n$ that starts as the all zero vector and then a sequence of updates comes. Each update is of the form (i, Δ), where $\Delta \in \mathbb{R}$ and $i \in \{1, .., n\}$. This corresponds to the operation $x_i \leftarrow x_i + \Delta$.
Given a function f, we want to compute or approximate $f(x)$.For example in the problem of distinct elements Δ is always 1 and $f(x) = |i : x_i \neq 0$.
The common/only technique for designing turnstile algorithms is **linear sketching**. The idea is to maintain in memory $y = \Pi x$, where $\Pi \in \mathbb{R}^{m \times n}$, a matrix that is short and fat. We care that $m < n$, usually much smaller. We can see that y is m-dimensional so we can store it efficiently but what about Π? If we need to store the whole Π in memory this will not lead to a better algorithm in terms of space. So, there are two common ways in constructing and storing Π. The one is that Π is deterministic and so we can easily compute Π_{ij} without keeping the whole matrix in memory.

The other way is that Π is defined by k-wise independent hash functions for some small k, so we can afford storing the hash functions and computing Π_{ij}.

Let's see now how updates happen when we have a linear sketch. Let Π^i be the i-th column of the matrix Π. Then $\Pi_x = \sum_{i=1}^n \Pi^i x_i$. So by storing $y = \Pi x$ when the update (i, Δ) comes we have that the new y equals $\Pi(x + \Delta e_i) = \Pi x + \Delta \Pi^i$. Observe that the first summand is the old y and the second summand is just some multiple of the i-th column of Π. This means that if I can tell which is the i-th column of Π in small space I can also perform updates in small space.

2.2.2 Moment Estimation for $p = 2$

This problem was investigated by Alon,Matis, Szegedy at StoC 96 [1]. Let $F_p = \|x\|_p^p = \sum_{i=1}^p |x_i|^p$. We want to estimate the space needed to solve the moment estimation problem as p changes. There is a transition point in complexity of F_p.

- $0 \leq p \leq 2, poly(\frac{logn}{\epsilon})$ space is achievable for $(1+\epsilon)$ approximation with $\frac{2}{3}$ success probability [1, 3].
- For $p > 2$ then we need exactly $\Theta(n^{1-\frac{2}{p}} poly(\frac{logn}{\epsilon}))$ bits of space for $(1+\epsilon)$ space with $\frac{2}{3}$ success probability [2, 4].

We not look at the case $p = 2$ which is the AMS sketch. Let $\sigma_i \in \{-1, 1\}^n$ be random signs coming from a 4-wise independent family. We need $O(logn)$ bits to represent such a family. Then set $y_i = \sum_{j=1}^n \sigma_{ij} x_j$. This means that our matrix Π we were referring before has rows the vectors σ_i. Observe that we can get columns because we just need to evaluate hash functions. Our algorithm is going to output $\frac{1}{m}\|y\|_2^2$ as estimator of $\|x\|_2^2$.

It holds that $\mathbb{E}y_i^2 = \mathbb{E}(\sum_{j=1}^n \sigma_{ij} x_j)^2 = \|x\|_2^2 + \mathbb{E}\sum_{j \neq j'} \sigma_{ij}\sigma_{ij'} x_j x_{j'}$. Moreover we need to estimate the variance and apply Chebyshev's inequality. Observe that $\mathbb{E}y_i^4 = \mathbb{E}(\sum_j \sigma_{ij} x_j)^4 = \mathbb{E}\sum_{j_1,j_2,j_3,j_4 \in [n]^4} \sigma_{ij_1}\sigma_{ij_2}\sigma_{ij_3}\sigma_{ij_4} x_{ij_1} x_{ij_2} x_{ij_3} x_{ij_4} = \sum_{j=1}^n x_j^4 + (1/2)\binom{4}{2} \sum_{j \neq j'} x_j^2 x_{j'}^2$, because when we take the expectation if a term in a summand appears odd number of times its expectation is going to be zero and cancel the whole summand, so we are left only with the summands in which all terms appear an even number of times. Calculating the variance and walking through the calculations we get the desired result.

Facts: We can also use gaussians instead of random signs. The analysis would be similar, but we would have to discretize the gaussians in order to avoid the need of infinite precision. We can also have only one hash function $\sigma : [m] \times [n] \to \{-1, 1\}$, as long as it is 4-wise independent. The difference now is that we need to compute expectation and variance of the whole estimator $\frac{1}{m}\sum_{i=1}^m y_i^2$ instead of each term individually.

2.2.3 Moment Estimation for $p \leq 2$

We are going to describe an idealized algorithm with infinite precision, given by Indyk [3]. Call a distribution $\mathbb{D}$ over $\mathbb{R}$ $p-stable$ if for $z_1, .., z_n$ iid from this distribution and for all $x \in \mathbb{R}^n$ we have that $\sum_{i=1}^n z_i x_i$ is a random variable with distribution $\|x\|_p \mathbb{D}$. An example of such a distribution are the gaussians for $p = 2$ and for $p = 1$ the Cauchy distribution, which has probability density function $pdf(x) = \frac{1}{\pi(x+1)^2}$.

Note: Observe that by the Central Limit Theorem an average of d samples from a distribution approaches a gaussian as d goes to infinity. But, if we pick a p-stable distribution for $p < 2$, such as Cauchy distribution, why does this not violate the Central Limit Theorem? Actually, for CLT to hold we also need bounded moments of the distribution. But for example second moment of Cauchy is unbounded so the theorem does not apply. Intuitively, one can think that p-stable distributions cannot exist for $p > 2$ because they violate Central Limit Theorem. Indeed, this is the case as was proved by Lévy in the 20's [5]. In fact, he proved that p-stable distributions exist if and only if $0 \leq p \leq 2$.

We get back to Indyk's algorithm. In fact the algorithm is very simple. Let the i-th row of Π be z_i, as before, where z_i comes from a p-stable distribution. Then consider $y_i = \sum_{j=1}^n z_{ij} x_j$. When a query comes, output the median of all the y_i. We can assume that without loss of generality a p-stable distribution has median equal to 1, which in fact means that for z from this distribution $\mathbb{P}(-1 \leq z \leq 1) \leq \frac{1}{2}$.

2.3 Next time

In the next lecture we are going to analyze Indyk's algorithm.

References

[1] Noga Alon, Yossi Matias, Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[2] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68(4):702–732, 2004.

[3] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53(3):307–323, 2006.

[4] Piotr Indyk, David P. Woodruff. Optimal approximations of the frequency moments of data streams. *STOC*, pages 202–208, 2005.

[5] Paul Lévy. Calcul des probabilités. Gauthier-Villars, Paris, 1925.