CS 15-859: Algorithms for Big Data Fall 2020

Lecture 6.1 — October 21

Prof. David Woodruff *Scribe: Tarun Chiruvolu*

1 Introduction to the Streaming Model

Now, we turn our focus away from sketching algorithms for linear regression and low rank approximation to another important topic in modern computer science - streaming algorithms. The streaming model formalizes a situation where we are presented inputs piecemeal, and storing all of them and analyzing them would be prohibitive. Instead, we seek algorithms that can perform useful computations approximately by making a single pass over the data stream. Streaming algorithms have myriad applications in the real world, ranging from financial analysis to scientific applications such as processing genomic data.

1.1 Turnstile Streaming Model

In this class, we begin by focusing on the turnstile streaming model, which is specified by a few crucial components.

1. We have an underlying n-dimensional vector $\mathbf{x}$ with all of its components initialized to 0 that serves as the state of our data.
2. We are presented with a long stream of updates of the form $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta_j$, where $\mathbf{x}_i$ is the i-th component of the vector and Δ_j is the update we receive at time j. In addition, we are guaranteed that each Δ_j is an element of the set $\{-M, -M+1, \cdots, M-1, M\}$, where $M \leq \text{poly}(n)$.
3. Furthermore, throughout the stream, $\mathbf{x}$ is promised to be in $\{-M, -M+1, \cdots, M-1, M\}^n$.
4. We are given a particular function $f(\mathbf{x})$ that we'd like to approximate, and we can choose to use a randomized strategy to do so.

Our goal (for now) is to use as few bits as possible when approximating the function. Note that in this problem setting, we assume n is very large, so trivially storing the entirety of $\mathbf{x}$ and exactly computing the result at the end would be impossible.

2 Estimating Norms in the Streaming Model

The first application of streaming algorithms we choose to consider is the problem of computing the norm of the vector $\mathbf{x}$. Formally, suppose we'd like to estimate the ℓ_p norm of the vector $\mathbf{x}$ at the end of the streaming process. We want to output a scalar z such that

$$(1-\epsilon)\|\mathbf{x}\|_p^p \leq z \leq (1+\epsilon)\|\mathbf{x}\|_p^p$$

with probability at least 9/10. We see that particular values of p have useful properties that can arise in natural applications:

- $p = 1$: Choosing $p = 1$ can correspond to the total variation distance between two probability distributions.
- $p = 2$: The ℓ_2 norm is frequently used in linear algebra and has several useful geometric properties in Euclidean space.
- $p = \infty$: When $p = \infty$, the norm estimation problem corresponds to applications like anomaly detection.

2.1 Example Problem: Euclidean Norm

Restating the problem, we'd like to output a scalar z such that

$$(1-\epsilon)\|\mathbf{x}\|_2^2 \leq z \leq (1+\epsilon)\|\mathbf{x}\|_2^2$$

with probability at least 9/10. To solve this problem, we turn to a familiar tool - using sketching. Recall that for any vector $\mathbf{x}$, multiplying by a CountSketch matrix $\mathbf{S}$ with $O(1/\epsilon^2)$ rows gives us the above property with constant probability. In particular,

$$\|\mathbf{Sx}\|_2^2 \in (1 \pm \epsilon)\|\mathbf{x}\|_2^2$$

with probability 9/10. To update the vector, we'd like to perform $\mathbf{Sx} \leftarrow \mathbf{S}(\mathbf{x} + \Delta_j \mathbf{e}_i)$, where $\mathbf{e}_i$ is the i-th standard basis vector. By linearity, this is equal to

$$\mathbf{Sx} + \Delta_j \mathbf{S}_i$$

since multiplying the i-th basis vector is the same as extracting the i-th column of a matrix. This gives us a relatively convenient form for computing updates, since we only need to add scaled columns of $\mathbf{S}$, which we already have, to refresh our state.
Each entry of the vector $\mathbf{Sx}$ is bounded from above by poly(n) since each entry of $\mathbf{x}$ is promised to be within poly(n). That means we can store $1/\epsilon^2$ such entries in $O(\log n)$, giving us $O(n/\epsilon^2)$ space complexity for storing $\mathbf{Sx}$. Note that we'd also like to store $\mathbf{S}$ efficiently, and naively, storing the matrix in its full form would take $O(n)$ space.
However, in the CountSketch lecture, we mentioned how $\mathbf{S}$ could be represented as a 2-wise independent hash function $h : [n] \to [1/\epsilon^2]$ and a 4-wise independent hash function $\sigma : [n] \to \{-1, 1\}$. Therefore, we can store $\mathbf{S}$ in $O(\log n)$ bits as well.

2.2 Example Problem: ℓ_1 Norm

Now, we'd like to output a scalar z such that

$$(1-\epsilon)\|\mathbf{x}\|_1 \leq z \leq (1+\epsilon)\|\mathbf{x}\|_1.$$

Borrowing the idea from ℓ_1 regression, we decide to sample a matrix $\mathbf{S}$ of Cauchy random variables and use it as a sketching matrix. In terms of the space complexity, there are ways to store $\mathbf{S}$ with

$O(1/\epsilon)$ words of space, and as before, we can store $\mathbf{Sx}$ using $1/\epsilon^2$ words of $O(\log n)$ bits.
Our intuition would lead us to arrive at outputting $\|\mathbf{Sx}\|_1$, as this would not only be a natural choice but it would also copy our strategy for $p = 2$. If we examine this value more closely, we can observe that each $(\mathbf{Sx})_i$ is the dot product of a vector of Cauchy random variables and a vector, so it follows a Cauchy distribution scaled by the ℓ_1 norm of $\mathbf{x}$. Since Cauchy random variables have poor concentration, we can't expect this value to behave well, and we have to look at alternative methods for approximating the ℓ_1 norm.

2.2.1 Median as an ℓ_1 norm Estimator

Instead of directly outputting $\|\mathbf{Sx}\|_1$, we'll instead output the median of the absolute values of the entries of $\mathbf{Sx}$. For a Cauchy random variable C, the pdf of its absolute value $|C|$ is given by the function

$$f(z) = \frac{2}{\pi(1+z^2)}.$$

Integrating over x the cdf becomes

$$F(z) = \int_0^z \frac{2}{\pi(1+z^2)}dz = \frac{2}{\pi}\arctan(z)$$

which gives us a relatively nice functional form to work with. The median is given when $F(z) = 1/2$. Solving for z gives us $z = 1$, so the median of the absolute value of a Cauchy random variable is 1.
Claim 1. If we take $r = \frac{\log(1/\delta)}{\epsilon^2}$ samples from F, and we take X to be the median of the samples X_i, then $F(X) \in [1/2 - \epsilon, 1/2 + \epsilon]$ with probability $1 - \delta$.
In other words, if we take r samples, we can guarantee that the probability of the empirical median will be close to the probability of the true median of the distribution. This can be shown using a Chernoff bound. However, we aren't immediately guaranteed that the value of X is close to the true median since $|C|$ has heavy tails. To show this, we'll consider the inverse cdf of $|C|$

$$F^{-1}(z) = \tan\left(\frac{\pi z}{2}\right).$$

Plugging in $1/2+\epsilon$ and $1/2-\epsilon$ gives us $\tan(\pi/4 + \pi\epsilon/2)$ and $\tan(\pi/4 - \pi\epsilon/2)$. It can be shown using a Taylor expansion that these values are bounded from both sides by $[1-4\epsilon, 1+4\epsilon]$, so we are also guaranteed that the empirical median is close to the true median not only in probability but also in value.
Therefore, taking a matrix $\mathbf{S}$ of Cauchy random variables and performing an update scheme identical to the ℓ_2 case gives us a vector of $O(1/\epsilon^2)$ components where each value is $\langle \mathbf{S}_i, x\rangle \sim |C_i|\|\mathbf{x}\|_1$. Taking the median of these components gives us a scalar z such that

$$(1-4\epsilon)\|\mathbf{x}\|_1 \le z \le (1+4\epsilon)\|\mathbf{x}\|_1$$

as desired.

2.3 General Case: ℓ_p Norm

2.3.1 $0 < p < 2$

In this case, it can be shown that $1/\epsilon^2$ words with size $O(\log n)$ each is sufficient [Kane, Nelson, Woodruff]. The proof mirrors that of the ℓ_1 norm case, and uses the existence of p-stable distributions.

Although these distributions have no closed functional form, we can take a random sample $\theta \in [-\pi/2, \pi/2]$ and $r \in [0,1]$ and compute

$$\frac{\sin(p\theta)}{\cos^{1/p}(\theta)}\left(\frac{\cos(\theta(1-p))}{\log(1/r)}\right)^{(1-p)/p}$$

which is a sample from a p-stable distribution. We can discretize these samples and store them using limited independence in a similar way to CountSketch.

2.3.2 $p > 2$

Unfortunately, p-stable distributions do not even exist for $p > 2$. Furthermore, we have a lower bound of $\Omega(n^{1-2/p})$ of space required for norm estimation in this case. Nonetheless, we can match the lower bound up to logarithmic factors by providing a $\tilde{O}(n^{1-2/p})$ algorithm for estimating the norm. To do so, we will leverage some useful properties of exponential random variables. Our goal is to get a constant factor approximation, and the sketch matrix will be of the form PD, where P is a regular CountSketch matrix, and D is an $n \times n$ diagonal matrix where the entries are reciprocals of independent exponential variables raised to the $1/p$ power. Although we did not cover the algorithm in lecture, we briefly comment on some useful properties of exponential random variables.

Stability of Exponential Random Variables: Recall that an exponential random variable is parameterized by a rate λ and can be used to model continuous arrival rates. The pdf of an exponential random variable with rate λ is

$$f(x) = \lambda e^{-\lambda x}$$

with support on the nonnegative reals. The cdf, by integration of the pdf, is then given by

$$F(x) = 1 - e^{-\lambda x}.$$

If we scale the variable by $t \geq 0$, the cdf becomes

$$F(x) = 1 - e^{-\lambda x/t}.$$

Also, consider a vector $\mathbf{y} = (y_1, y_2, \cdots y_n)$ living in $\mathbb{R}^n$ and a collection of independent exponential random variables $\{E_i\}_{i=1}^n$. Then, let $q = \min\{E_1/|y_1|^p, E_2/|y_2|^p, \cdots, E_n/|y_n|^p\}$. It is a well known property of exponential random variables that a random variable given by the minimum of a set of independent random variables is also an exponential random variable with a rate given by the sum of the rates of its constituent random variables. We can see this by writing

$$\mathbb{P}[q > x] = \mathbb{P}\left[\forall i,\ \frac{E_i}{|y_i|^p} \geq x\right] = \prod_{i=1}^n e^{-x|y_i|^p} = e^{-x\|\mathbf{y}\|_p^p}$$

so q is also an exponential random variable.

3 Other Example Problems

3.1 Testing if $\mathbf{x} = \mathbf{0}^n$

In this problem, our goal is to check whether the vector $\mathbf{x}$ is 0 after the updates with probability at least 9/10. While we have been using randomized strategies until this point, it is natural to wonder whether we can use a deterministic strategy and achieve similar improvements in space complexity. A trivial deterministic algorithm would be to store all of the entries of the vector in $\Theta(n \log n)$ space and compute the result. However, can we do better in terms of space complexity deterministically?

Theorem 1. *Any deterministic algorithm for this problem requires $\Omega(n \log n)$ space.*

Proof. Assume for sake of contradiction that there exists some algorithm that uses $o(n \log n)$ space. Consider a stream whose first half corresponds to the vector $\mathbf{a} = \{0, 1, 2, \cdots, \text{poly}(n)\}^n$. Let $S(\mathbf{a})$ denote the state of the algorithm at this point. Since $|S(\mathbf{a})| = o(n \log n)$, there must be at least one other vector $\mathbf{a}'$ such that $S(\mathbf{a}) = S(\mathbf{a}')$ by the pigeonhole principle. Then, let the second half of the stream correspond to the vector $\mathbf{b} = \{-0, -1, -2, \cdots, -\text{poly}(n)\}^n$. Then we will get the same answer for $S(\mathbf{a}+\mathbf{b})$ and $S(\mathbf{a}'+\mathbf{b})$, when only one of them can be 0, resulting in a contradiction. ∎

This result forces us to use randomization to do better. We turn again to CountSketch matrices to solve this problem. Since we get multiplicative error, if $\mathbf{x} = 0$, then CountSketch will necessarily output the zero vector, and our norm will be 0. On the other hand, if $\mathbf{x} \neq 0$, then we get $\|\mathbf{Sx}\| \in (1 \pm \epsilon)\|\mathbf{x}\|_2^2$ with probability 9/10, so $\mathbf{Sx} \neq \mathbf{0}$. Once again, the space complexity remains the same.

3.2 Recovering a k-sparse vector

Suppose we are given some vector $\mathbf{x}$ that has at most k nonzero entries at the end of the stream. We'd like to recover the indices and values of the k nonzero entries with high probability. Our goal is to see whether we can do it with $O(k \log n)$ bits of space as well as provide an alternative deterministic algorithm.

Claim 2. We claim that if we have a matrix $\mathbf{A}$ where every set of $2k$ columns is linearly independent, then we can maintain $\mathbf{Ax}$ in the stream and recover the nonzero values, as well as their locations.

Proof. Suppose, towards a contradiction, that there exist two vectors $\mathbf{x}, \mathbf{y}$ such that $\mathbf{Ax} = \mathbf{Ay}$. Then, $\mathbf{A}(\mathbf{x} - \mathbf{y}) = 0$. The vector $\mathbf{x} - \mathbf{y}$ can have at most $2k$ nonzero entries, when none of the nonzero entries line up. However, since any set of $2k$ columns is linearly independent, multiplying by the nonzero entries guarantees that we cannot have an output of the 0 vector, resulting in a contradiction. ∎

Now, we need to provide an example of a matrix that satisfies this property. We can use the Vandermonde matrix

$$\begin{pmatrix} 1 & 1 & 1 & \cdots \\ 1 & 2 & 3 & \cdots \\ 1 & 4 & 9 & \cdots \\ 1 & 8 & 27 & \cdots \end{pmatrix}$$

where each entry (i, j) is j^{i-1}. When we compute the determinant of any $2k \times 2k$ submatrix with columns $\{i_1, \cdots, i_{2k}\}$, the formula is given as $\prod_j i_j \prod_{j<j'} (i_j - i_{j'}) \neq 0$, so the determinant cannot be 0. Note that storing the entries can be exponential in n, so instead, we just take $\mathbf{Ax}$ modulo $p = \text{poly}(n)$ where p is some sufficiently large prime number. If p is large enough, then none of the entries of $\mathbf{S}$ will be divisible by p. Also, since the entries of $\mathbf{x}$ are bounded, we can assume the same for it. Finally, we don't want the determinant to go to 0, but looking at the product formulation for the determinant, it is clear to see that it will not be divisible $\mod p$.

4 References

On the Exact Space Complexity of Sketching and Streaming Small Norms. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. Proceedings of the 2010 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). 2010, 1161-1178