**CS 15-859: Algorithms for Big Data** Fall 2019

Lecture 7 — October 17

*Prof. David Woodruff* *Scribe: Mayur Paralkar*

# 1 Introduction to the Streaming Model

## 1.1 Turnstile Streaming Model

Although many data stream models exist, we will focus on the Turnstile Streaming Model for now. The model begins with an underlying n-dimensional vector $x$, which is initialized to $0^n$. This vector receives a long stream of additive updates to its coordinates of the form

$$x_i \leftarrow x_i + \Delta_j$$

where $\Delta_j \in \{-M, -M+1, ..., M-1, M\}$, assuming that $M \leq \text{poly}(n)$. $x_i$ represents the $i^{th}$ coordinate of $x$ and $\Delta j$ is the $j^{th}$ update in the stream. Each $x_i$ is bounded by the same values as $\Delta_j$ throughout the stream: $x_i \in \{-M, -M+1, ..., M-1, M\}$.

At the end of the stream, we will output an approximation to $f(x)$ with high probability for some function $f$. Note that the stream is worst case, so we cannot make any distribution assumptions on the input. Our goal is to use as little space (in bits) as possible when storing values throughout the model, and no limits are placed on the time complexity of our algorithms for now. The goal is derived from the applications of the data stream model to massive datasets such as stock transactions, weather data, and genomes. As we are able to pass through updates only once in this model, the general idea for minimizing space is often to maintain a summary or sketch to help with outputting $f(x)$. Let's proceed by looking at one problem we may encounter.

## 1.2 Testing if $x = 0^n$

Following the stream of updates, we may with to output whether $x = 0^n$ with constant probability. This problem is solvable with only $O(\log n)$ bits of space. Remember that we have previously shown that for any fixed vector $x$, if $S$ is a CountSketch matrix with $O(\frac{1}{\epsilon^2})$ rows, then

$$|Sx|_2^2 = (1 \pm \epsilon)|x|_2^2$$

with constant probability. Consequently, if $x = 0^n$, then the norm for $Sx$ will be 0. Otherwise, if $x \neq 0^n$, then the norm for $Sx$ will be greater than 0. Let $\epsilon$ be some small constant fraction (e.g. $\frac{1}{2}$). Each entry (row) in $Sx$ will consist of at most $n$ non-zero elements of $x$, each of which is upper-bounded by poly(n). Therefore, an entire row of $Sx$ takes only $O(\log n)$ space to store. Since $Sx$ will have $O(1)$ rows, we can store all of it using $O(\log n)$ bits of space. Furthermore, note that we can store updates to $x$ and $Sx$ analogously:

$$x_i \leftarrow x_i + \Delta_j$$

$$Sx \leftarrow Sx + \Delta_j S_i$$

where $S_i$ is the $i^{th}$ column of $S$. Finally, we do not need to store the actual matrix $S$; it will suffice to store its associated hash and sign functions, each of which is possible with $O(\log n)$ bits.

Interestingly, any deterministic algorithm to test if $x = 0^n$ requires at least $\Omega(n \log n)$ bits of space.

*Proof.* We will use a pigeonhole argument to prove our claim.

Suppose vector $a \in \{0, 1, ..., \text{poly}(n)\}^n$ corresponds to updates to each of the elements of $x$ for the first half of updates in a stream, and let $S(a)$ be the state of the algorithm after the updates are applied. Assume for the sake of contradiction that $S(a)$ has stored less than $n \log n$ bits. The number of possible values for $a$ is $\text{poly}(n)^n$, which takes $n \log n$ bits to write. Therefore, there exists a vector $a'$ for which $S(a) = S(a')$. Now, suppose that vector $b \in \{0, -1, ..., -\text{poly}(n)\}^n$ corresponds to updates to each of the elements of $x$ during the second half of updates in the stream. The algorithm must output the same answer on $a + b$ and $a' + b$, even though the former corresponds to $0^n$ and the latter does not. Therefore, the algorithm will err in one case, and thus a contradiction is shown. ■

### 1.3 Recovering a k-Sparse Vector

Suppose we are promised that $x$ has at most $k$ non-zero entries at the end of the stream, where $k$ is relatively small. It is possible to find the indices and values of these entries with high probability deterministically and using $O(k \text{ poly}(\log n))$ bits of space.

Suppose $A$ is an $s \times n$ matrix such that any $2k$ columns are linearly independent. Throughout the stream, we will maintain $A \cdot x$. Again, updates to $x$ and $A \cdot x$ can be analogously defined:

$$x_i \leftarrow x_i + \Delta_j$$

$$A \cdot x \leftarrow A \cdot x + \Delta_j A_i$$

**Claim 1.** It is possible to recover the subset of $k$ non-zero entries and their values from $A \cdot x$.

*Proof.* Assume for the sake of contradiction that there exists vectors $x$ and $y$ each with at most $k$ non-zero entries and $A \cdot x = A \cdot y$. Then $A(x - y) = 0$. $A$ will choose $2k$ cols from vector $x - y$, $x - y$ has at most $2k$ non-zero entries, and any $2k$ columns are linearly independent. Combining the last two, $x - y = 0$. Therefore, $x = y$, which contradicts the assumption that $x$ and $y$ are two different vectors. ■

Now, one may ask whether an $A$ where any $2k$ columns are linearly independent, with a small number of rows (at least $2k$), *and* with low storage costs, exists. The answer is... yes!

### 1.3.1 Vandermonde Matrix

Let $A$ be a Vandermonde matrix with $s = 2k$ rows and $n$ columns. Then $A_{i,j} = j^{i-1}$.

$$A = \begin{bmatrix} 1 & 1 & 1 & ... \\ 1 & 2 & 3 & ... \\ 1 & 4 & 9 & ... \\ 1 & 8 & 27 & ... \\ ... & ... & ... & ... \end{bmatrix}$$

**Claim 2.** Any $2k$ columns of $A$ are linearly independent.

*Proof.* It is known that the determinant of $2k \times 2k$ submatrix of $A$ with columns $i_1, i_2, ..., i_{2k}$ is

$$\prod_j i_j \prod_{j<j'} (i_j - i_{j'}) \neq 0$$

Therefore, any $2k$ columns of $A$ are linearly independent. ■

However, one may ask how to store $A$ and $A \cdot x$ in a small amount of space, especially when the elements are exponentially increasing. Clearly, it is possible to generate any entry of $A$ on demand. But, $A \cdot x$ needs a large amount of space. Our solution will be to make the entries smaller by storing $A \cdot x \mod p$ for a prime $p$ that is larger than the maximum possible $x$ value (poly($n$)). No two vectors will result in the same value.

*Proof.* Consider two vectors $x, y$. Assume for the sake of contradiction that

$$A \cdot x \mod p = A \cdot y \mod p$$

Then

$$A(x - y) \mod p = 0$$

$A(x - y)$ still has linearly independent columns mod $p$ since the determinant is not divisible by $p$ and $A(x - y)$ is of size $2k \times 2k$. Using the determinant formula from earlier, any $2k$ columns are linearly independent. Therefore, $x \mod p = y \mod p$. Since $x < p$ and $y < p$, $x = y$. Thus, a contradiction is shown. ■

There are $2k$ rows to store, and each row needs $O(\log n)$ bits. Therefore, the overall space complexity is $O(k \log n)$.

## 2 Estimating Norms in the Streaming Model

**Definition.** (p-norm of $x$) $|x|_p^p = \sum_{i=1}^n |x_i|^p$

With constant probability, one may wish to output an approximation to the p-norm of $x$, which we will call $Z$:

$$(1 - \epsilon)|x|_p^p \leq Z \leq (1 + \epsilon)|x|_p^p$$

## 2.1 Euclidean Norm

Suppose we desire $Z$ such that

$$(1-\epsilon)|x|_2^2 \leq Z \leq (1+\epsilon)|x|_2^2$$

We will use a CountSketch matrix $S$ with $\frac{1}{\epsilon^2}$ rows in order to find $Z$. As mentioned earlier, we can use hash functions to store $S$ efficiently. Also, we will use the same analogous updates as shown in subsection 1.2 (if $x_i \leftarrow x_i + \Delta_j$ then $Sx \leftarrow Sx + \Delta_j S_i$). At the end of the stream, we will output $|Sx|_2^2$. By the properties of CountSketch, $|Sx|_2^2 = (1 \pm \epsilon)|x|_2^2$ with constant probability. Each row in $|Sx|$ takes $O(\log n)$ bits to store. Since there are $1/\epsilon^2$ rows, the overall space complexity in bits is $O(\frac{\log n}{\epsilon^2})$.

## 2.2 1-Norm

Now consider the problem of norm estimation of 1-norm. Suppose we desire $Z$ such that

$$(1-\epsilon)|x|_1 \leq Z \leq (1+\epsilon)|x|_1$$

One idea is to follow the same idea as that with the Euclidean Norm, except using a Cauchy matrix $S$ instead of a CountSketch matrix. However, Cauchy random variables have no concentration, so this estimator will not work. Instead, let us look at the shape of $Sx$.

$$Sx = \begin{bmatrix} |x|_1 \cdot C_1 \\ |x|_1 \cdot C_2 \\ ... \\ |x|_1 \cdot C_{1/\epsilon^2} \end{bmatrix}$$

Instead of taking the mean, we can use the median instead!

### 2.2.1 1-Norm Estimator

The PDF $f(x)$ of $|C|$ for a half-Cauchy random variable $C$ is

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

Therefore, the CDF $F(z)$ is

$$F(z) = \int_0^z f(x)dx = \frac{2}{\pi}\tan^{-1}(z)$$

Using the fact that $tan(\frac{\pi}{4}) = 1$, we can see that $F(1) = \frac{1}{2}$. Since exactly half the probabilities occur before 1, the median value of $|C|$ is 1.

**Lemma 1.** *For any distribution $F$, if you take $r = \frac{\log(1/\delta)}{\epsilon^2}$ independent samples $X_1, X_2, ..., X_r$ from $F$, and $X = median_i X_i$, then $F(X) \in [\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon]$ with probability $1 - \delta$.*

Notice that $F^{-1}(X) = \tan(\frac{X\pi}{2})$. We can show that $F^{-1}(X) \in [1-4\epsilon, 1+4\epsilon]$ with probability $1-\delta$ as follows. Let $Z_L$ be the point at which a $\frac{1}{2}-\epsilon$ fraction of the samples are to the left of it and let $Z_R$ be the point at which a $\frac{1}{2}-\epsilon$ fraction of the samples are to the right of it on the CDF. Let $w_i$ be the indicator if $|C_i| \leq Z_R$. Then $\mathbf{Pr}\Big[w_i = 1\Big] \geq \frac{1}{2}+\epsilon$. We know $W = \sum w_i$ and $\mathbb{E}[W] \geq \frac{r}{2} + \epsilon r$. Using a Chernoff bound, $\mathbf{Pr}\Big[W < \frac{r}{2}\Big] \leq \mathbf{Pr}\Big[|W - \mathbb{E}[W]| \geq \epsilon r\Big] \leq e^{-\Theta(\epsilon^2 r)}$. So, the median is a good estimator since the median of $\frac{\log(1/\delta)}{\epsilon^2}$ Cauchy random variables is $O(1 \pm \epsilon)$ with constant probability.

### 2.2.2 Discretizing Continuous Distributions

It seems impossible to have an upper bound on the space complexity, since the size of a single Cauchy RV can be infinite. To combat this problem, we can use rounding. We will look at an example by applying this idea to a sketching matrix $S$ of size $\frac{1}{\epsilon^2} \times n$ with Gaussians $N(0, \epsilon^2)$.

Round each non-zero entry in $S$ to the nearest $\frac{1}{n^2}$ factor. Using the PDF of a normal RV,

$$\mathbf{Pr}\Big[|N(0,1)| \geq t\Big] \leq 2e^{-t^2} \leq \frac{1}{n^2}$$

if we set $t = O(\sqrt{\log n})$. With probability $1 - \frac{1}{\epsilon^2 n}$, all entries will have absolute value at most $O(\sqrt{\epsilon \log n})$. If we consider the case when $\epsilon$ is a constant, the number of different values that the rounded Gaussians can take is $O(n^2\sqrt{\log n})$, as all entries will be in the discrete set $\{-C\sqrt{\log n}, ..., -\frac{1}{n^2}, 0, \frac{1}{n^2}, ..., C\sqrt{\log n}\}$ with high probability for some constant $C > 0$. We can enumerate all of the possibilities using $O(\log n)$ bits, so each entry in the Gaussian matrix only needs $O(\log n)$ bits to represent.

Rounding the entries in this way will not change the behavior of the sketching matrix $S$ by a large factor. Suppose $S'$ is the result of rounding the elements in $S$ and $x$ is an arbitrary vector. Then $S' \cdot x = S \cdot x + E \cdot x$, where $E$ is the error matrix from rounding. Each entry in $E$ will be at most $\frac{1}{n^2}$ due to our rounding scheme. By the triangle inequality, it follows that

$$||S' \cdot x||_2 = ||S \cdot x||_2 \pm ||E \cdot x||_2$$

Additionally, it is true that

$$||E \cdot x||_2 \leq ||E||_2 ||x||_2 \leq ||E||_F ||x||_2$$

and

$$||E||_F \leq \frac{1}{n} \cdot O(\sqrt{n/\epsilon}) = O(\sqrt{1/(\epsilon n)}$$

If $\epsilon$ is $O(1)$, the total error is $O(\sqrt{1/n}||x||_2)$.

$$O(\sqrt{1/n}||x||_2) \leq ||S \cdot x||_2 = (1 \pm \epsilon)||x||_2$$

with constant probability when using a non-rounded sketching matrix $S$. Note that this analysis is quite general; each entry could have been rounded to additive $\frac{1}{n^{100}}$ to obtain the same result.

Thus, our approximations are close and we can discretize the entries in the matrix with this procedure.