| | |
|---|---|
| **CS 15-859: Algorithms for Big Data** | Fall 2022 |

Lecture 8 Part 2 — 11/03/22

*Prof. David Woodruff* — *Scribe: Emma Hu*

# 1 $p$-Norm Estimator for $p > 2$

Recall from the first half of the lecture that when estimating the $p$-th norm of a vector in the streaming model where $p > 2$, we apply two matrices $P$ and $D$, where $P$ is a regular CountSketch matrix and $D$ is a $n \times n$ diagonal matrix. The $i$-th entry of $D$ is $D_{ii} = \frac{1}{E_i^{1/p}}$ where each $E_i$ is an exponential random variable.

**Lemma 1.** *Assume we have $n$ independent exponential random variables $E_1, E_2, ..., E_n$ and scalars $|y_1|, ..., |y_n|$, let $q = \min(\frac{E_1}{|y_1|^p}, ..., \frac{E_n}{|y_n|^p})$, then $q$ is also an exponential random variable.*

*Proof.* Note that

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

which is an exponential random variable with $\lambda = |y|_p^p$. In another word, $q \equiv \left(\frac{1}{|y|_p^p}\right) E$. ■

Using the above lemma, we know that

$$|Dy|_\infty^p = \max_i \frac{|y_i|^p}{E_i} = \frac{1}{\min_i \frac{E_i}{|y_i|^p}} = \frac{1}{E \cdot \frac{1}{|y|^p}} = \frac{|y|_p^p}{E}.$$

Also,

$$\mathbb{P}[E \in \left[\frac{1}{10}, 10\right]] = (1 - e^{-1}) - (1 - e^{-\frac{1}{10}}) \geq \frac{4}{5},$$

i.e., with probability at least $\frac{4}{5}$,

$$|Dy|_\infty \in \left[\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p\right].$$

This essentially means that $|Dy|_\infty$ is a good estimate of $|y|_p$. However, $|Dy|$ is an $n$-dimensional vector, so we apply an additional CountSketch matrix $P$ to $Dy$. Intuitively, $P$ is hashing coordinates of $Dy$ into buckets and taking a signed sum of the entries. Our goal now is to show that with high probability,

$$|PDy|_\infty \approx |Dy|_\infty.$$

## 1.1 Analysis of $|PDy|_\infty$

Let's first establish the following notations:

- Let $s$ be the number of rows of $P$, which we can think of as the number of hash buckets for elements in $Dy$.
- Let $h : [n] \to [s]$ and $\sigma : [n] \to \{-1, 1\}$ be the two hash functions that describes the CountSketch matrix $P$. For our analysis we assume $h$ and $\sigma$ are truly random, although they could be de-randomized in practice.

Then by the above notation,

$$(PDy)_i = \sum_j \delta(h(j) = i)\sigma_j (Dy)_j.$$

We already know that $|Dy|_\infty \in \left[\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p\right]$ with probability at least $\frac{4}{5}$. To achieve $|PDy|_\infty \approx |Dy|_\infty$ with good probability, we want

- in each bucket $i$ not containing the coordinate $j$ for which $|(Dy)_j| = |Dy|_\infty$ (i.e. not containing the max), we have $|(PDy)_i|) \leq \frac{|y|_p}{100}$;
- in the bucket $i$ containing the coordinate $j$ for which $|(Dy)_j| = |Dy|_\infty$ (i.e. containing the max), we have $|(PDy)_i| - |Dy|_\infty) \leq \frac{|y|_p}{100}$.

We start by calculating the expectation and variance of $(PDy)_i$. Let $\delta$ be the indicator random variable, i.e. $\delta(E) = 1$ if event $E$ holds, and $\delta(E) = 0$ otherwise.

**Lemma 2.** $\mathbb{E}[(PDy)_i] = 0$.

*Proof.* Recall the above expression $(PDy)_i = \sum_j \delta(h(j) = i)\sigma_j (Dy)_j$, and note that $\mathbb{E}[\sigma_j] = 0$ for all $j$ since it's the sign function. ■

**Lemma 3.** $\mathbb{E}[(PDy)_i^2] = O(\frac{1}{s})(n^{1-2/p}|y|_p^2)$.

*Proof.* To begin with,

$$\begin{aligned}
\mathop{\mathbb{E}}_P[(PDy)_i^2] &= \sum_{j,j'} \mathbb{E}[\delta(h(j) = i)\delta(h(j') = i)\sigma_j\sigma_{j'}](Dy)_j(Dy)_{j'} \\
&= \sum_j \mathbb{E}[\delta(h(j) = i)](Dy)_j^2 \\
&= \frac{1}{s}|Dy|_2^2.
\end{aligned}$$

Then,

$$\mathop{\mathbb{E}}_D[|Dy|_2^2] = \sum_i y_i^2\, \mathbb{E}[D_{i,i}^2]$$

Now, let's bound $\mathbb{E}[D_{i,i}^2]$. Recall that $D_{i,i} = \frac{1}{E_i^{1/p}}$, and $E_i^{2/p} = \int_{t\geq 0} e^{-t}t^{-2/p}dt$ we obtain

$$\begin{aligned}
\mathbb{E}[D_{i,i}^2] &= \int_{t\geq 0} t^{-2/p}e^{-t}dt \\
&= \int_{0\leq t\leq 1} t^{-2/p}e^{-t}dt + \int_{t>1} t^{-2/p}e^{-t}dt \\
&\leq \int_{0\leq t\leq 1} t^{-2/p}dt + \int_{t>1} e^{-t}dt \\
&= O(1)
\end{aligned}$$

Finally, from Holder's Inequality, we get

$$\begin{aligned}
|y|_2^2 &= \sum y_i^2 \cdot 1 \\
&\leq (\sum (y_i^2)^{p/2})^{2/p} \cdot (\sum 1^{1/(1-2/p)})^{1-2/p} \\
&= n^{1-2/p}|y|_p^2
\end{aligned}$$

Putting everything together, we establish the claim. ■

Now, we introduce the Bernstein's Bound to further bound $(PDy)_i^2$.

**Theorem 1.** *Bernstein's Bound: Suppose $R_1, ..., R_n$ are independent, and for all $j$, $|R_j| \leq K$, and $Var[\sum_j R_j] = \sigma^2$, there are constants $C$, $c$ such that for all $t > 0$,*

$$\mathbb{P}\left[|\sum_j R_j - \mathbb{E}[\sum_j R_j]| > t\right] \leq C(e^{-\frac{ct^2}{\sigma^2}} + e^{-\frac{ct}{K}}).$$

While it's tempting to directly apply this bound to $(PDy)_i = \sum_j R_j$ where each

$$R_j = \delta(h(j) = i)\sigma_j(Dy)_j,$$

we require however, that $R_j \leq K$. We cannot simply define $K$ to be the maximum entry since then we can only get a constant bound instead. This defeats our purpose since in the end we need to take a union bound over all entries in $PDy$. As a result, we proceed to divide into two cases, and only apply Bernstein's Bound to the *small* entries in $PDy$.

## 1.2 Understanding the Large Elements

Recall $(PDy)_i = \sum_j \delta(h(j) = i)\sigma_j(Dy)_j$, and set $R_j = \delta(h(j) = i)\sigma_j(Dy)_j$. We will separately handle those $R_j$ for which $|R_j| > \frac{\alpha|y|_p}{\log n}$ for a sufficiently small constant $\alpha > 0$. Note that if $|R_j| > \frac{\alpha|y|_p}{\log n}$, then necessarily $|(Dy)_j| > \frac{\alpha|y|_p}{\log n}$. We call such a $j$ *large* if $|(Dy)_j| > \frac{\alpha|y|_p}{\log n}$, otherwise $j$ is *small*.

**Lemma 4.** *The expected number of large $j$ is $O(\log^p n)$, with constant probability.*

*Proof.* Recall that $|(Dy)_j| = \frac{|y_j|}{E_j^{1/p}}$. Then

$$\begin{aligned}
\mathbb{P}[j \text{ is } \textit{large}] &= \mathbb{P}\left[\frac{|y_j|}{E_j^{1/p}} \geq \frac{\alpha|y|_p}{\log n}\right] \\
&= \mathbb{P}\left[\frac{|y_j|^p \log^p n}{\alpha^p |y|_p^p} \geq E_j\right] \\
&= 1 - e^{-\frac{|y_j|^p \log^p n}{\alpha^p |y|_p^p}} \\
&\leq \frac{|y_j|^p \log^p n}{\alpha^p |y|_p^p}.
\end{aligned}$$

So adding up the probabilities, the expected number of *large* $j$ is $\frac{\log^p n}{\alpha^p} = O(\log^p n)$. Finally, we can derive the conclusion after applying Markov inequality. ■

Now, conditioning on $|Dy|_\infty \in \left[\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p\right]$ and that the number of *large* $j$ is in $O(\log^p n)$, we know that with high probability, there is no collision for the *large* elements in $h$ (i.e. all the *large* $j$ get perfectly hashed into separate hash buckets by $P$). We can easily verify this since $s \geq n^{1-2/p}$ which is way larger than the number of *large* elements (can think of buckets as bins and *large* elements as balls, and $\mathbb{P}[\text{exists a collision}](1/s)\binom{(\log n)^p}{2} \ll \frac{1}{100}$). Now, with large elements dealt with separately, we are ready to apply Bernstein's bound for each hash bucket and wrap up our analysis!

### 1.3 Applying Bernstein's bound

Recall that we conditioned on:

- $|Dy|_\infty \in \left[\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p\right]$.
- The expected number of *large* elements is $O(\log^p n)$.
- All *large* elements are perfectly hashed in $h$.

In other words, in every single bucket, there is either no *large* element, or only one *large* element. Setting the *large* elements aside, we can assume $K = \max_j |R_j| \leq \frac{\alpha|y|_p}{\log n}$. Now, setting $t = \frac{|y|_p}{100}$ and $s = \Theta(n^{1-2/p}\log n)$, we finally apply the Bernstein's bound

$$\mathbb{P}\left[\left|\sum_{\textit{small } j} \delta(h(j) = i)\sigma_j (Dy)_j\right| > \frac{|y|_p}{100}\right] \leq \frac{1}{n^2}$$

By a union bound over all the $s$ buckets, the 'signed sum' of *small* $j$ in every bucket will be at most $\frac{|y|_p}{100}$.

### 1.4 Wrapping up

From the above analysis, we now know that for all $i$,

- $|(PDy)_i| \leq \frac{|y|_p}{100}$ if there is no *large* indices in the $i$-th bucket.
- $|(PDy)_i| = |\sigma_j(Dy)_j| \pm \frac{|y|_p}{100}$ if there is exactly one *large* index in the $i$-th bucket.
- No bucket contains more than one *large* index.

Since we also conditioned on $|Dy|_\infty \in \left[\frac{|y|_p}{10^{1/p}}, 10^{1/p}|y|_p\right]$, we conclude that

$$|PDy|_\infty \leq 10^{1/p}|y|_p + \frac{|y|_p}{100},$$
$$|PDy|_\infty \geq \frac{|y|_p}{10^{1/p}} - \frac{|y|_p}{100}.$$

Therefore, $|PDy|_\infty$ is indeed a good estimate of $|y|_p$, so we can simply output $|PDy|_\infty$. The overall space complexity is thus $s = O(n^{1-2/p}\log n)$ words, which is $O(n^{1-2/p}\log^2 n)$ bits.

# 2 Heavy Hitters in a Stream

The goal of this problem is finding the *large* entries of a vector $x$. There are different sets of guarantees to define *large* entries.

1. $l_1$-guarantee:
   - output a set containing all items $j$ for which $|x_j| \geq \phi|x|_1$
   - the set should not contain any $j$ with $|x_j| \leq (\phi - \varepsilon)|x|_1$
2. $l_2$-guarantee:
   - output a set containing all items $j$ for which $x_j^2 \geq \phi|x|_2^2$
   - the set should not contain any $j$ with $x_j^2 \leq (\phi - \varepsilon)|x|_2^2$

Note that $l_2$-guarantee can be much stronger than the $l_1$-guarantee. For example, let's consider the vector $x = (\sqrt{n}, 1, 1, ..., 1)$. Item 1 is an $l_2$-heavy hitter for constant $\phi = 1/2$, but not an $l_1$-heavy hitter. More specifically, if an item is an $l_1$-heavy hitter, it is also an $l_2$-heavy hitter, since

$$|x_j| \geq \phi|x|_1 \Rightarrow x_j^2 \geq \phi^2|x|_1^2 \geq \phi^2|x|_2^2$$

## 2.1 Heavy Hitter Intuition

Suppose you are promised at the end of the stream, $x_i = n$ and $x_j \in \{0, 1\}$ for $j \in \{1, 2, ..., n\}$ with $j \neq i$. How do we find $i$? One thing we could do is to try to reconstruct its binary representation. For each $j \in \{1, 2, ..., \log n\}$, let $A_j \subset [n]$ be the set of indices with $j$-th bit in their binary representation equal to 0, and $B_j$ be the set of indices with $j$-th bit in their binary representation equal to 1. Then, for each $j \in \{1, 2, ..., \log n\}$, compute $a_j = \sum_{i \in A_j} x_i$ and $b_j = \sum_{i \in B_j} x_i$. Finally, we can obtain the identity of $i$: if $a_j > b_j$, then the $j$-th bit of $i$ is 0; otherwise, it's 1. This is a deterministic algorithm with $O(\log^2 n)$ bits of memory.