CS 15-859: Algorithms for Big Data	Fall 2017
Lecture 8 - 2 — 10/26/2017	
Prof. David Woodruff	*Scribe: Anish Sevekari*

These notes continue the discussion on ℓ_2 heavy hitters. At this point, we can approximate x_i for all i simultaneously up to an additive error of $\mathcal{O}\left(\frac{|X|_2}{\sqrt{B}}\right)$

Tail Guarantee for ℓ_2 heavy hitters

We can approximate each x_i simultaneously up to an additive factor of $\mathcal{O}\left(\frac{|X|_2}{\sqrt{B}}\right)$. But if one of the x_i is much larger than others, then we get very bad approximations for all other x_i. One way to fix this is to argue that with high probability, none of the large value end up in same bin as x_i, then we can get a better approximation for x_i.

Theorem 1 (Tail Guarantee for CountSketch)**.** *CountSketch approximates every x_i simultaneously up to an additive error of $\mathcal{O}\left(\frac{\left|X_{-B/4}\right|_2}{\sqrt{B}}\right)$ where $X_{-B/4}$ denotes X after 0-ing out the top $B/4$ entries of X in magnitude.*

Proof. For a fixed i, we claim that with probability $3/4$, none of the top $B/4$ entries hash into the same bucket as x_i. For any j, probability that x_j hashes into the same bin as x_i is $1/B$. Taking the union bound over top $B/4$ entries, probability that at least one of them colloid with x_i is at most $1/4$, which gives us the required probability bound.

Now we can condition on hash function satisfying this condition, and analyze the estimator $\hat{X}_i = \sigma_i C_{h(i)}$.

$$\hat{X}_i = x_i + \sum_{\substack{i' \neq i \\ i' \text{ not in top } B/4}} \sigma_i \sigma_{i'} x_{i'} + \sum_{\substack{i' \neq i \\ i' \text{ in top } B/4}} \sigma_i \sigma_{i'} x_{i'}$$

Note that the second term is 0 after conditioning on hast function. And we can analyze the first time using just pairwise independence. Also, the top $B/4$ terms don't contribute to the variance of $\hat{X}_i$. Therefore,

$$\mathbb{E}[\hat{X}_i] = x_i$$

and

$$\mathbb{E}[\hat{X}_i^{\,2}] \leq \frac{\left|X_{-B/4}\right|_2}{\sqrt{B}}$$

Therefore, with constant probability, we get an additive error of $\mathcal{O}\left(\frac{\left|X_{-B/42}\right|}{\sqrt{B}}\right)$ for each x_i. We can repeat the process for $\mathcal{O}(\log n)$ times and take the median to get this error bound with $1 - 1/\operatorname{poly}(n)$ probability. And then union bound gives us the tail guarantee for ℓ_2 heavy hitters.

Remark. If x is $B/4$ sparse, then we can recover entire x accurately with high probability!

∎

Finding top k heavy hitters

Consider a complete binary tree with height $\lg n$. There are 2^i nodes in i^{th} level. For each node in i^{th} level, we can associate a subset of $[n]$ of size $n/2^i$, with the same i-bit prefix. Prefixes associated to nodes are such that if prefix corresponding to a node is $p_1 \dots p_i$ then the prefix associated to its children are $p_1 \dots p_i 0$ and $p_1 \dots p_i 1$.

For each node, we keep track of 2-norm of all the entries corresponding to that node. Algorithm to find top k heavy hitters goes as follows:

- Start at level with $2k$ nodes. Hash these $2k$ nodes into $]Oh(k)$ buckets and use ℓ_2 heavy hitters algorithm to find k nodes that have largest 2-norm. We can hash $\mathcal{O}(\log k)$ times independently to the probability guarantee.
- In the next level, we have to look at only the $2k$ children of top k nodes that we found in previous level, and repeat the same procedure.
- We can repeat the process until we hit bottom-most level, which gives us the top k heavy hitters.

Main advantage is that at each point, we are running the ℓ_2 approximation algorithm for only $\mathcal{O}(k)$ nodes instead of $\mathcal{O}(n)$ nodes. And repeat this at most $\mathcal{O}(\lg n)$ times. Therefore, we get a factor of $\mathcal{O}(\lg n)$ instead of $\mathcal{O}(n)$ for the time complexity.

Remark. Each update also take $\mathcal{O}(\lg n)$ time since we have to update $\lg n$ nodes, corresponding to all of the prefixes.

ℓ_1 heavy hitters

Recall: ℓ_1 guarantee:

- output a set of numbers j such that $|x_j| \geq \phi |x|_1$
- the set should not contain any j with $|x_j| \leq (\phi - \varepsilon)|x|_1$

ℓ_2 guarantee:

- output a set of numbers j such that $x_j^2 \geq \phi |x|_2^2$
- the set should not contain any j with $x_j^2 \leq (\phi - \varepsilon)|x|_2^2$

Why care about ℓ_1 guarantee

ℓ_2 guarantee implies ℓ_1 guarantee, since

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

But, ℓ_1 guarantee can be solved deterministically, while there is a lower bound for ℓ_2 guarantee.

Deterministic ℓ_1 heavy hitters

Definition. An $s \times n$ matrix S is called ε-*incoherent* if

- for all column S_j of S, $|S_j|_2 = 1$
- for all pairs i and j, $|\langle S_i, S_j \rangle| \leq \varepsilon$
- entries of S can be specified with $\mathcal{O}(\log n)$ bits.

Geometrically, columns of S are unit vectors which are almost orthogonal. If we have such a matrix S, we can maintain Sx using $\mathcal{O}(s \log n)$ space. Further, we claim that for any i, $\hat{X}_i = S_i^T S_i x$ computes x_i with $\varepsilon |x|_1$ error.

Proof.

$$\begin{aligned} \hat{X}_i &= \sum_{j=1}^{n} \langle S_i, S_j \rangle) x_j \\ &= |S_i|_2^2 x_i \pm \sum_{j \neq i} |\langle S_j, S_i \rangle| |x_j| \\ &= x_i \pm \varepsilon |x|_1 \end{aligned}$$

■

Then, we can figure out which i satisfy ℓ_1 guarantee.

Existence of ε-incoherent matrices

Consider prime $q = \Theta((\log n)/\varepsilon)$. Let $d = \varepsilon q$. Note that $d = \mathcal{O}(\log n)$ We consider polynomials $P_1, \ldots, P_n$ over the field $\mathbb{F}_q$ of degree less than or equal to d. There are $q^d - 1$ such polynomials, so, we have to choose constants the Θ notation such that $q^d > n$.

Let $s = q^2$. Divide rows into q groups containing q rows each. We associate P_i with i^{th} column. In j^{th} group, the i^{th} column has exactly one non-zero entry. The $P_i(j)^{\text{th}}$ entry in i^{th} column is $1/\sqrt{q}$. Note that norm of each column is 1, since it contains exactly q non-zero entries, each of which is $1/\sqrt{q}$. Further, if two columns i and j share more d common entries, then P_i and P_j agree on more than d values! Since they have degree less than or equal to d, they must be same! But we chose all P_i's to be distinct. Therefore, this cannot happen. Therefore, for any i and j,

$$|\langle S_1, S_2 \rangle| \leq d \cdot 1/q \leq \varepsilon$$

This proves that all matrices in this family are ε-incoherent.

Estimating Number of non-zero entries

Definition. $|x|_0 = |\{i \text{ such that } x_i \neq 0\}|$

We want to find an ε approximation to $|x|_0$, that is, a output a number Z such that

$$(1-\varepsilon)Z \leq |x_0| \leq (1+\varepsilon)$$

Sparse Case

Suppose $|x|_0 = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$. Then we can use k-sparse vector recovery algorithm to get number of non-zero entries exactly. Another way is to use CountSketch to recover non-zero entries of x.

Reducing error in 2-approximation

Suppose we can find Z such that $Z \leq |x|_0 \leq 2Z$ then we can increase accuracy by sampling. Let $p = \frac{100}{Z\varepsilon^2}$. We sample each coordinate independently by probability p. Let Y_i be random variable indicating if i^{th} coordinate was sampled or not. Let y be x restricted to only those coordinates with $Y_i = 1$

$$\mathbb{E}[|y|_0] = \sum_{\substack{i \\ x_i \neq 0}} \mathbb{E}[Y_i] = p|x|_0 > \frac{100}{\varepsilon^2}$$

$$\mathbf{Var}[|y|_0] = \sum_{\substack{i \\ x_i \neq 0}} \mathbf{Var}[Y_i] \leq \frac{200}{\varepsilon^2}$$

Therefore, Chebyshev's inequality gives us a bound:

$$\mathbf{Pr}\left[||y|_0 - \mathbb{E}[|y|_0]| > \frac{100}{\varepsilon}\right] \leq \frac{1}{50}$$

Therefore, we get a relative error of ε in $|y|_0$ with probability 49/50. Multiplying by $1/p$, we can get x_0 with an relative error of ε

Algorithm for the general case

We cannot get a 2-approximation to $|x|_0$ as of yet. But, if we go through all powers of 2 less than n, one of them satisfies the 2-approximation property. We can do the following:

- guess Z in powers of 2. There are $\mathcal{O}(\log n)$ of them.
- for i^{th} guess, we can sample probability $p = \min\left(1, \frac{100}{2^i\varepsilon^2}\right)$
- We do a nested sampling instead of sampling every time, so $[n] = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_{\log n}$
- Run the previous algorithm to estimate $|x|_0$ for each i.

One of the Z's satisfy $Z \leq |x|_0 \leq 2Z$ and for that i, we will get an ε approximation for $|x|_0$. So, we are left with guessing which one works.

Claim. Largest $Z = 2^i$ for which $\frac{400}{\varepsilon^2} \leq |y|_0 \leq \frac{3200}{\varepsilon^2}$ works!

Proof. Let y_i denote vector x after sampling coordinates in set S_i. Note that $\mathbb{E}[|y_i|_0] = \frac{|x|_0}{2^i \varepsilon^2}$. Therefore, note that $\mathbb{E}[|y_i|]$ is strictly decreasing, and so is $|y_i|_0$, since we do a nested sampling. Let i' be such that

$$\frac{800}{\varepsilon^2} \leq \mathbb{E}[|y_{i'}|_0] \leq \frac{1600}{\varepsilon^2}$$

then by Chebyshev's inequality,

$$\frac{400}{\varepsilon^2} \leq |y_{i'}|_0 \leq \frac{3200}{\varepsilon^2}$$

with probability at least 49/50. Similarly, following holds for $i' + 3$

$$\frac{100}{\varepsilon^2} \leq \mathbb{E}[|y_{i'+3}|_0] \leq \frac{200}{\varepsilon^2}$$

then

$$|y_{i'+3}|_0 \leq \frac{400}{\varepsilon^2}$$

with probability at least 49/50. Lets assume that both of these events hold, which happens with probability 48/50. Note that i is the largest index such that $\frac{400}{\varepsilon^2} \leq |y_i|_0 \leq \frac{3200}{\varepsilon^2}$. Sunce i' also satisfies this, $i \geq i'$. But, since $|y_{i+3}| \leq \frac{400}{\varepsilon^2}$ we get that $i' + 3 > i \geq i'$, therefore, i can take only 3 different values. For each of these 3 values, $|y_i|_0 = (1 \pm \varepsilon)\mathbb{E}[|y_i|_0]$ with probability 49/50. Again, taking an union bound, with probability 47/50, i gives us an ε approximation for $|x|_0$ for all the three values of i. Therefore, with probability at least $1 - 2/50 - 3/50 = 9/10$, we get an ε approximation to $|x|_0$ for the chosen value of i. ∎

Space Complexity

Since we are using k-sparse recovery algorithm for $k = \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$, it takes $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ space. We repeat this $\mathcal{O}(\log n)$ many times, so total space complexity is $\mathcal{O}\left(\frac{(\log n)^2}{\varepsilon^2}\right)$, ignoring the randomness.

For sampling and randomness, we can keep a pairwise independent hash function $h : [n] \to [n]$, and pick j in S_i if and only if $h(j) \leq \frac{n}{2^i \varepsilon^2}$. This in fact gives us the nested sampling as required. Further, probability bound is obtained using Chebyshev's inequality, which requires only pairwise independence. The hash function can be stored using $\mathcal{O}(\log n)$ bits.

We can improve space complexity to $\mathcal{O}\left(\frac{\log n \left(\log\left(\frac{1}{\varepsilon}\right) + \log\log n\right)}{\varepsilon^2}\right)$. This improvement comes from decreasing complexity of k-sparse recovery counters. In the levels that we care about, there are only $\mathcal{O}(1/\varepsilon^2)$ counters, each counter has $\mathcal{O}(\log n)$ bits. Instead, we can store the counter modulo a prime q that does not divide the counter value, since we are only going to check if it is non-zero. There are at most $\mathcal{O}\left(\frac{\log n}{\varepsilon^2}\right)$ which can divide any of these counters. Therefore, if we choose a random prime $q = \mathcal{O}\left(\frac{\log n \log\log n}{\varepsilon^2}\right)$ then with high probability, it does no divide any of the counters. We can store the entire sparse recovery structure modulo q, which takes $\mathcal{O}\left(\log\log n + \log\frac{1}{\varepsilon}\right)$ bits instead of $\mathcal{O}(\log n)$