CS 15-859: Algorithms for Big Data Fall 2022

Lecture 9 Part 2 — 11/11

Prof. David Woodruff *Scribe: Yudong Liu*

1 Heavy Hitters ℓ_1 Guarantee

Definition: An $s \times n$ matrix S is ϵ-incoherent if $\forall S_i$, $|S_i|_2 = 1$ and $\forall$ column pairs S_i, S_j we have $\langle S_i, S_j \rangle < \epsilon$, and also, entries can be specified with $O(\log n)$ bits of space.
We want to computer a stream of $S\dot{x}$ using $O(s \log n)$ bits of space using an estimate $\hat{x}_i = S_i^T Sx$. According to ϵ-incoherence we have

$$\hat{x} = \sum_{j=1,\cdots n} \langle S_i, S_j \rangle x_j = ||S_i||_2^2 x_i \pm \max_{i,j} |\langle S_i, S_j \rangle| ||x||_1 = x_i \pm \epsilon \tag{1}$$

Which solves the ℓ_1 Heavy Hitter problem.
Now we hope to find ϵ-incoherent matrices. The reason we care about them is that they're "deterministic". The step is as follows:
Consider prime $q = \Theta(\log n/\epsilon)$. Let $d = \epsilon - q = \Theta(\log n)$. For n distinct polynomials $p_1, p_2, \cdots, p_n$ each of degree less than or equal to d. For each there're $q^d - 1 > n$ possible cases.
Now to construct the a ϵ-incoherent matrix S, let its number of rows to be $s = q^2$. associate each p_i with ith column of matrix S to group the rows of S into q groups , each group consists of q consecutive rows. For each jth group, ith column consists of a single non-zero entry on the $p_i(j)$-th location, such that the entry is $\frac{1}{\sqrt{q}}$. Therefore, $\forall i, ||S_i||_2 = \sqrt{q * (\frac{1}{\sqrt{q}})^2} = 1$.
Therefore, the dot product between two columns $\langle S_i, S_j \rangle$ would be the number of entries k such that $p_i(k) = p_j(k)$. Notice that the number of such groups is at most $d = \epsilon q$. Therefore, we have $|\langle S_i, S_j \rangle| \leq \epsilon$. Therefore, Thus we have achieved an ϵ-incoherent matrix with $\mathrm{O}(\frac{1}{\epsilon^2} \log^2(n))$ rows.

2 Finding Top-K Heavy Hitters Quickly

If we were to perform the previous streaming algorithm, which searches among the estimates of $x_i \in X$, then the total runtime requires at least $O(n)$ to conclude. We want to the time complexity to be sub-linear to n (for example linear to k) to find the top hitters in a vector of elements X. Therefore we propose the following algorithm:

1. Construct a binary tree such that there're 2^i nodes on the i-th level of tree. Each node consists to a subset of $[n]$ of size $n/2^i$ with the same i-bit prefix. We start at the level where there're at least $2k$ nodes

2. On each of the ith level, there're 2^i groups where each group's size is $\frac{n}{2^i}$. Think of each group as a metacoordinate and define each group vector as y_i. Use count sketch matrix S to compute Sy_i, as an approximation. Such that $\forall i, ||Sy_i||_2^2 = (1 \pm \frac{1}{100})||y_i||_2^2$

3. Set $O(k)$ buckets and hash each Sy_i to a bucket. In each bucket j, the value is stored as $\sum_{h(i)=j} Sy_i$, where h denotes the hash function.
For a $\frac{1}{k}$ heavy hitter x_j. Assume that it appears in group y_i on some level l. Then obviously y_i is also a $\frac{1}{k}$ heavy hitter among $\{y_0, y_1, \cdots, y_{2^l}\}$, due to the fact that $||y_i||_2^2 \geq x_j^2 \geq \frac{1}{k}||X||_2^2$.
Assume y_i is hashed to some bucket B, denote other groups hashed to the bucket as $B = \{y^{i_1}, y^{i_2}, \cdots, y^{i_q}\}$, where q is the size of the bucket. By triangular inequality we have

$$||S(y_i+y^{i_1}+y^{i_2}+\cdots, y^{i_q})||_2 = (1\pm\frac{1}{100})||(y_i+y^{i_1}+y^{i_2}+\cdots, y^{i_q})||_2 \leq (1\pm\frac{1}{100})(||y_i||_2+||(y^{i_1}+y^{i_2}+\cdots, y^{i_q})||_2)$$

Since in expectation the square length gets evenly distributed among buckets in expectation, therefore we have $\mathbb{E}(||(y^{i_1} + y^{i_2} + \cdots, y^{i_q})||_2^2) = \frac{||X||_2^2}{CK}$, where Ck is the number of buckets linear to k. Thus with high probabilty $||(y^{i_1} + y^{i_2} + \cdots, y^{i_q})||_2 \leq \frac{||x||_2}{\sqrt{Ck}}$. Therefore, it's easy to conclude that if there's a heavy hitter in the bucket, the noise would be small compared to the size of the heavy hitter. Therefore, a bucket would only be large if there's heavy hitter in the bucket.

4. We want to estimate each y_i with a heavy-hitter by decoding each bucket using the count sketch. To do that we take the median of all buckets that y_i is hashed to . There're two sources of errors, one is the error caused by count sketch, the other is the euclidean norm error caused by the noises in the same bucket. If there's a heavy hitter in the bucket, then the estimate would be large since the euclidean norm of noises would be small compared to the heavy-hitter $||x_j||_2^2 \geq \frac{||X||_2^2}{k}$. Notice that there're at most k coordinates satisfying $||y_j||_2^2 \geq ||x_j||_2^2 \geq \frac{1}{k}||X||_2^2$. Therefore we find at most k heavy hitters after decoding.

5. Now we know on each of the ith level, there're 2^i groups where each group's size is $\frac{n}{2^i}$, among them only k of the coordinates that contains the heavy hitters are large after approximation using count sketch and hashing. Since there're only $O(k)$ number of hashing bins to consider. We only need to spend $O(k)$ time approximating the coordinates containing the heavy hitters. As we move onto the next level, we know the heavy hitters would only be splitted between the left or right child node of the coordinate containing heavy hitters on the previous layer. Thus, although the total number of groups are doubled in the next layer, we only have at most $2k$ nodes to consider as well upon entering a new layer.
Thus, as we decode level by level, notice that the runtime for each layer is $O(k)$. Thus the total time is $O(k \log n)$.

3 Estimating the number of NonZero Entries (ℓ_0 norm)

Definition:
$||x||_0 = |\{i : x_i \neq 0\}|$
We want to output a number Z such that $(1-\epsilon)Z \leq |x|_0 \leq (1+\epsilon)Z$.
Suppose $|x_0| = O(\frac{1}{\epsilon^2})$. We want our algorithm to recover a k-sparse vector such that $k = O(\frac{1}{\epsilon^2})$
General Knowledge:
If we have D distinct items, we could sample each element of index $\{1, 2, \cdots, n\}$ with probability p independently. Therefore we have $p * D$ expected number of sampled items. Construct Y_i such that

$Y_i = 1$ if item i is sampled, and 0 otherwise. Let $X = \sum_i Y_i$. We have

$$\mathbb{E}(X) = PD \tag{2}$$

$$Pr[|X - PD| \geq \lambda] \leq \frac{Var[X]}{\lambda^2} \leq \frac{PD}{\lambda^2} \tag{3}$$

$$Var(X) = P(1-P)D \tag{4}$$

Let $P = \Theta(\frac{1}{\epsilon^2 n})$. The number of coordinates sampled is in expectation $\Theta(\frac{1}{\epsilon^2})$
Case 1:
Suppose we somehow had an estimate Z with $Z \leq |x|_0 \leq 2Z$. We use the following preocedures to estimate $|x|_0$ within a multiplicative factor of ϵ:

1. We choose sampling probability $p = \frac{100}{Z\epsilon^2}$, and perform independent sampling across each coordinates
2. Using the above intuition, construct Y such that $Y_i = 1$ if the ith element is sampled, and 0 otherwise.
3. Use sparse recovery or CountSketch to compute $|Y|_0$.
4. Output $\frac{|Y|_0}{p}$ as an estimate for the ℓ_0 norm of x.

We can show that the produced estimation is within $(1 \pm \epsilon)$ multiplicative factor of $||x||_0$:
Notice that

$$E(||Y||_0) = P||x||_0 = \frac{100}{Z\epsilon^2}||x||_0 \in [\frac{100}{\epsilon^2}, \frac{200}{\epsilon^2}] \tag{5}$$

and

$$Var(||Y||_0) = \sum_i Var(Y_i) \leq \frac{200}{\epsilon^2}. \tag{6}$$

let $\lambda = \frac{100}{\epsilon}$, therefore we have

$$Pr[|||Y||_0 - E(||Y||_0)| \geq \lambda] \leq \frac{Var(||Y||_0)\epsilon^2}{100^2} \leq \frac{1}{50} \tag{7}$$

Therefore, with probability of at least $\frac{49}{50}$, we have $(1-\epsilon)E(||Y||_0) \leq ||Y||_0 \leq (1+\epsilon)E(||Y||_0)$

Case 2:
Suppose we don't have the estimate Z such that $Z \leq |x|_0 \leq 2Z$. Then we can guess Z in powers of 2. Since $0 \leq ||x||_0 \leq n$, there're $O(\log n)$ guesses. The ith guess $Z = 2^i$ corresponds to sampling each coordinate with probability $p = \min(1, \frac{100}{2^i\epsilon^2})$. Then sample coordinates as nested subsets

$$[n] = S_0 \supseteq S_1 \supseteq S_2 \supseteq \cdots S_{\log n}$$

We then run the previous algorithm for each $Z = 2^i$. Eventually one of our guess satisfies $Z = 2^i \leq ||x||_0 \leq 2Z = 2^{i+1}$. Then we could preform the algorithm in Case 1 using that guess.
Conclusion:
A good option is to use a largest guess $Z = 2^i$ for which $\frac{400}{\epsilon^2} \leq ||y||_0 \leq \frac{3200}{\epsilon^2}$.
If $\frac{800}{\epsilon^2} \leq E(||Y||_0) \leq \frac{1600}{\epsilon^2}$, then by Chebyshev's inequality,

$$\frac{400}{\epsilon^2} \leq ||Y||_0 \leq \frac{3200}{\epsilon^2} \tag{8}$$

with probability $1 - O(\epsilon^2) = \frac{49}{50}$.
If $\frac{100}{\epsilon^2} \leq E(||Y||_0) \leq \frac{200}{\epsilon^2}$, then by Chebyshev's inequality,

$$||Y||_0 \leq \frac{400}{\epsilon^2} \tag{9}$$

with probability $1 - O(\epsilon^2) = \frac{49}{50}$.
Thus, in conclusion, with probability $\frac{98}{100}$, we choose i satisfies $\frac{200}{\epsilon^2} \leq E(||y||_0) \leq \frac{1600}{\epsilon^2}$.
Notice there are only 4 such indices i, and all 4 of them satisfy $||Y||_0 = (1 \pm \epsilon)E(||Y||_0)$ simultaneously with probability $1 - \frac{4}{50} = \frac{46}{50}$.
Overall our success probability is $1 - \frac{2}{50} - \frac{4}{50} = \frac{44}{50} > \frac{4}{5}$