CS 15-859: Algorithms for Big Data Fall 2020

Lecture 10 — 11/12/2020

Prof. David Woodruff *Scribe: Zheng Xu*

1 Estimating Number of Non-zero Entries

We have a problem where we want to find $\|x\|_0 = |\{ \text{ i such that } x_i \neq 0\}|$ (usually called *0 norm*). As usual, we want to output Z such that $(1-\varepsilon)\|x\|_0 \leq Z \leq (1+\varepsilon)\|x\|_0$ with prob. 9/10 and use $O((\log n)/\varepsilon)$ bits of space. For a simpler case where we suppose $\|x\|_0 = O(\frac{1}{\varepsilon^2})$, we can either use CountSketch or use Vandermonde matrix to recover a k-sparse vector where $k = O(\frac{1}{\varepsilon^2})$. But, we also want an algorithm when $|x|_0 >> \frac{1}{\varepsilon^2}$. We will use some technique to sample coordinates of x and solve the problem on sampled x.

1.1 Estimation with given rough estimate

Suppose we are given an estimate Z such that $Z \leq \|x\|_0 \leq 2Z$. Given this information we want to sample coordinates of x so that the number of non-zero entries in sampled x is $O\left(\frac{1}{\varepsilon^2}\right)$. We start with an example where we independently sample each coordinate of x with probability $p = \frac{100}{Z\varepsilon^2}$. Let Y_i be an indicator random variable if coordinate i is sampled and let y be the vector restricted to coordinates i for which $Y_i = 1$.

$$\mathbb{E}[\|y\|_0] = \sum_{i \text{ such that } x_i \neq 0} \mathbb{E}[Y_i] = p\|x\|_0 = \frac{100}{Z\varepsilon^2}\|x\|_0 \geq \frac{100}{\varepsilon^2} \tag{1}$$

$$\text{Var}[Y_i] \leq \mathbb{E}[Y_i^2] = \mathbf{Pr}[Y_i = 1] = p \tag{2}$$

$$\text{Var}[\|y\|_0] = \sum_{\text{i such that } x_i \neq 0} \text{Var}[Y_i] \leq p\|x\|_0 \leq \frac{200}{\varepsilon^2} \tag{3}$$

$$\tag{4}$$

Here we use Chebyshev's inequality

$$\mathbf{Pr}[|\|y\|_0 - \mathbb{E}[\|y\|_0]| > \frac{100}{\varepsilon}] \leq \frac{\text{Var}[\|y\|_0]\varepsilon^2}{100^2} \leq \frac{1}{50} \tag{5}$$

Thus, $\|y\|_0 \in \frac{100}{\varepsilon^2} \pm \frac{100}{\varepsilon}$ with probability $\frac{49}{50}$. We calculate $\|y\|_0$ using sparse recovery or CountSketch to compute $\|y\|_0$ exactly, then we multiply it by $\frac{1}{p}$ we get $\frac{1}{p}\|y\|_0 \in (1 \pm \varepsilon)\|x\|_0$.

1.2 Find the appropriate Z

Because $0 \leq \|x\|_0 \leq n$, there are $O(\log(n))$ powers of 2 in the range. So we can guess Z in power of 2 and run the algorithm in parallel $O(\log(n))$ times. The i-th guess $Z = 2^i$ corresponds to sampling each coordinate with probability $p = \min(1, \frac{100}{2^i\varepsilon^2})$. Let S_i be the set of coordinates sampled for

$Z = 2^i$. We can ensure that $S_0 \supseteq S_1 \supseteq \ldots S_{O(\log(n))}$. Note that the sets S_i, S_j aren't independent for $i \neq j$ but for each set S_i the coordinates in S_i are sampled independently. We compute an estimate of $\|x\|_0$ using each set S_i. Among the guesses, one of them is correct.
The problem comes down to which one of the answers is correct. Let y^i denote the vector x restricted to the coordinates S_i. Because of the nesting property of the subsets S_i, we have

$$\|y^0\|_0 \geq \|y^1\|_0 \geq \ldots \|y^{\log(n)}\|_0.$$

The rule is we use the largest guess i for which $\frac{400}{\varepsilon^2} \leq \|y^i\|_0 \leq \frac{3200}{\varepsilon^2}$. Let this be i^*.

For any i, we have by Chebyshev's inequality

$$\mathbb{P}[|\|y^i\|_0 - \mathbb{E}[\|y^i\|_0]| \geq t] \leq \frac{\mathrm{Var}[\|y^i\|_0]}{t^2} \leq \frac{\mathbb{E}[\|y^i\|_0]}{t^2}.$$

Here the last inequality follows from $\|y^i\|_0$ is a sum of independent binary random variables. Let i' be such that $\frac{800}{\varepsilon^2} \leq \mathbb{E}[\|y^{i'}\|_0] \leq \frac{1600}{\varepsilon^2}$. Picking $t = 400/\varepsilon^2$, we obtain from above that

$$\frac{400}{\varepsilon^2} \leq \|y^{i'}\|_0 \leq \frac{2000}{\varepsilon^2}$$

with probability $\geq 1 - O(\varepsilon^2)$. We also have that $100/\varepsilon^2 \leq \mathbb{E}[\|y^{i'+3}\|_0] \leq 200/\varepsilon^2$ from which we obtain similar to above that $\|y^{i'+3}\|_0 \leq 400/\varepsilon^2$ with probability $\geq 1 - O(\varepsilon^2)$. Therefore using union bound, we have with probability $\geq 1 - O(\varepsilon^2)$, that $2000/\varepsilon^2 \geq \|y^{i'}\|_0 \geq 400/\varepsilon^2$ and $\|y^{i'+3}\|_0 \leq 400/\varepsilon^2$ simultaneously. Thus $i' \leq i^* \leq i' + 3$ using monotonicity of $\|y^i\|_0$ and the fact that i^* is the largest index with $400/\varepsilon^2 \leq \|y^{i^*}\| \leq 3200/\varepsilon^2$. Now it is easy to check that for all four guesses $Z = 2^{i'}, 2^{i'+1}, 2^{i'+2}, 2^{i'+3}$, the algorithm using Countsketch gives $1 \pm O(\varepsilon)$ approximation and therefore the estimate computed using i^* is a $1 + O(\varepsilon)$ approximation.

1.3 Overall Space Complexity

We are doing k-sparse recovery $\log(n)$ times, each k-sparse recovery uses $O(k \log(n))$ bits. $k = O(\frac{1}{\varepsilon^2})$ here, so overall $O(\frac{\log^2(n)}{\varepsilon^2})$ total bits of space will be used ignoring random bits.
Now we consider the amount of randomness we need for the algorithm to work. We only used Chebyshev's inequality in our algorithm, and Chebyshev's inequality only requires pairwise independence. We will now determine how we can achieve pairwise independence.
Suppose we have a function $h : [n] \to [n]$ and we want to sample each coordinate with probability $\frac{1}{2}$. We can use $h(i) \mod 2$ to determine if we sample this coordinate or not, 0 to sample, 1 not to. Now consider the use case on S_i. We sample the coordinate when the last i digits in binary representation of $h(i)$ are zero. This method of sampling also satisfies nested subset. We need $O(\log(n))$ bits of space for randomness.
The overall memory is $O(\frac{\log(n)(\log \frac{1}{\varepsilon} + \log(\log(n)))}{\varepsilon^2})$. But we can still improve by reducing the size of counters.

1.4 Reducing Counter Size

In sparse recovery we use a Vandermonde matrix, and we can reduce memory size of the matrix by doing mod L for a sufficiently large prime L.

In sampling levels that we care about, we have $O(\frac{1}{\varepsilon^2})$, each of $O(\log n)$ bits. To start with, we want a prime that does not divide any of the $O(\frac{1}{\varepsilon^2})$ counters. There are a total of $O(\frac{\log(n)}{\varepsilon^2})$ primes that can divide any of the counters. We want to choose a prime from $\{2, \ldots, L\}$ that avoids these primes. In $\{2, \ldots, L\}$, there are a total of $\Theta(\frac{L}{\log(L)})$ primes. We choose $L = O(\frac{\log(n)(\log(\log(n))+\log(\frac{1}{\varepsilon}))}{\varepsilon^2})$, such that $\frac{L}{\log(L)} > O(\frac{\log(n)}{\varepsilon^2})$ and it is unlikely that L divides any counter. In this case, we just need to maintain our sparse recovery structure mod L, which is $O(\frac{\log\frac{1}{\varepsilon}+\log(\log(n))}{\varepsilon^2})$ for each counter, and a total of $O(\log(n))$ counters.