CS 388R: Randomized Algorithms Fall 2015

Lecture 19 — Nov 11, 2015

Prof. Eric Price *Scribe: Xiangru Huang, Qi Lei*

1 Overview

In previous lectures, we introduced Count-Min Sketch for finding l_1 heavy hitters.

In this lecture, we will introduce Count Sketch and l_0 sampling.

2 Review Count-Min sketch

First let's review Count-Min sketch.

We have a stream of insertions and deletions $(u, \Delta x_u)$ within "strict" turnstile model(i.e. $x_u \geq 0$) and want to recover $\hat{x}_u$.

Count-Min sketch works by maintaining $r = \log \frac{|U|}{\delta}$ bloom filters with size $O(k)$. And get

$$|\hat{x}_u - x_u| \leq \frac{\|x_{-k}\|_1}{k}$$

where x_{-k} is the vector after deleting the top k elements from x.

3 Count sketch

Before introducing Count sketch, let's consider a medium case algorithm.

3.1 Count-Median sketch

Recall that in Count-Min sketch, we have hash function h_i and counters in bloom filters looks like

$$y_{i,v} = \sum_{u, h_i(u)=v} x_u$$

we get one estimate from each bloom filter

$$\tilde{x}_u^{(i)} = y_{i,h_i(u)}$$

and each estimate is good, since with probability $\frac{1}{2}$

$$|\tilde{x}_u^{(i)} - x_u| \leq \frac{\|x_{-k}\|_1}{k} \tag{1}$$

So then we take the min of all estimates

$$\tilde{x}_u = \min_i \tilde{x}_u^{(i)}$$

we have with high probability

$$|\tilde{x}_u - x_u| \leq \frac{\|x_{-k}\|_1}{k}$$

Now for turnstile model(not necessary strict), we can not use min any more, and a nature idea would be to change min to median.

However, if we change **min** to **median**, we will not get the same bound, since the expectation of number of estimates not satisfy Eqn. (1) is $\frac{r}{2}$, which effectively makes the median not good. But since the size of each bloom filter is $O(k) = Ck$, we can choose parameter C larger so that the probability for each estimate to satisfy Eqn. (1) is $\frac{3}{4}$. Then this modified algorithm (Count-Median) will work similarly as Count-Min.

3.2 Count sketch

The idea of Count sketch is to symmetrize the noise(in Count-Min we have all positive noise). As in Count sketch, we define

- pair-wise independent hash functions $h_i : U \to [c]$, and random signs $s_i(u) \in \{\pm 1\}$
- counters in bloom filter
$$y_{i,v} = \sum_{u, h_i(u)=v} s_i(u) x_u$$
- estimate from one bloom filter $\tilde{x}_u^{(i)} = s_i(u) y_{i,h_i(u)}$
- estimate of x_u is $\tilde{x}_u = \text{ median}_i \ \tilde{x}_u^{(i)}$

Now we give the bounds of Count sketch.

For each u, first we condition on u not colliding with any top $|H|$ elements by using h_i, which is true with probability $1 - \frac{|H|}{c}$. Then

$$(\tilde{x}_u^{(i)} - x_u)^2 = \sum_{u' \notin H \cup \{u\}} x_{u'}^2 \mathbb{1}_{h_i(u)=h_i(u')} + \sum_{u_1, u_2 \notin H \cup \{u\}} s_i(u_1) s_i(u_2) \mathbb{1}_{h_i(u_1)=h_i(u)=h_i(u_2)} x_{u_1} x_{u_2}$$

The expectation of the second term is zero since

$$\mathbb{E}[s_i(u_1) s_i(u_2)] = 0$$

Hence we have

$$\begin{aligned} \mathbb{E}(\tilde{x}_u^{(i)} - x_u)^2 &= \mathbb{E}[\sum_{u' \notin H \cup \{u\}} x_{u'}^2] \frac{1}{c} \\ &\leq \frac{\|x_{-k}\|^2}{c} \end{aligned}$$

Therefore $(\tilde{x}_u^{(i)} - x_u)^2 \leq \frac{8\|x_{-k}\|^2}{c}$ with probability $\geq \frac{7}{8}$.

3.3 Comparison with Count-Min sketch

In generaly, l_2 norm is better than l_1 norm.

For example, consider power law distribution with parameter α, the i-th largest element has frequency proportional to $i^{-\alpha}$. With this kind of distribution, Count-Min is good for $\alpha > 1$, Count sketch is good for $\alpha > \frac{1}{2}$.

4 l_0 sampling

Given a stream of items $v_1, v_2, \ldots$, we would like to sample one element from U, where U contains all unique elements appeared in the stream. To be specific, we want to design an online algorithm using sublinear space. Therefore storing U is not realistic.

Here we consider three version of this problem.

4.1 First version

In the first version, each elements appears once. We can use Reservoir Sampling[1], which basically works as follows

- Set the first element as candidate.
- For $i \geq 2$, switch i-th element with the candidate with probability $\frac{1}{i}$
- Output candidate after the stream.

4.2 Second version

In the second version, we have duplicates of the same elements in the stream. For this version, we can pick a uniformly random hash function h such that

$$h : U \to [0, 1]$$

Then we store the element v that minimize $h(v)$, and output v.

If h is fully independent, we get perfect performance

- $O(1)$ words
- $O(1)$ time for each element in the stream and storing and evaluating h
- x is uniformly random from U

But fully independence is not practical. So we have an alternative reasoning

[1] `https://en.wikipedia.org/wiki/Reservoir_sampling#Reservoir_with_Random_Sort`

- We can use min-wise independence from [BCF00], which basically means

$$\forall S, x \in S, \ \mathbb{P}[h(x) = \min_{x' \in S} h(x')] = \frac{1 \pm \epsilon}{|S|}$$

- Theorem 1.1 in [I01] shows that $O(\log \frac{1}{\epsilon})$-wise independence implies approximate min-wise independence for $|S| < \epsilon |U|$

4.3 Third version

In this version, we have deletions in the stream. To solve this problem, we first introduce a few building blocks. The high level idea is to subsample the stream so that each substream contains one unique element with good probability. And recover the element back using certain simple techniques.

4.3.1 subsampling

Similar to the way we recover shortest path in lecture 15, we first get a 2-approximation of $\|x\|_0$ by choosing $\log |U|$ different r to be $1, 2, 4, \ldots, \log |U|$. For each r, we choose $\log |U|$ different hash function h_i where $h_i : U \to [r]$ and keep the samples with $h_i(v) = 0$. Each hash function corresponds to one subsample set.

As we did in lecture 15, if r is a 2-approximation to the true $\|x\|_0$, i.e., size of distinct elements of the stream, with probability at least $\frac{1}{2e}$, you have exactly one unique element in one subsample set. Since we choose $O(\log |U|)$ subsample sets, with high probability $1 - |U|^{-c}$, we will get one unique element in some subsample set.

4.3.2 Check if only one element remains

Although each time with probability $\geq \frac{1}{2e}$ only one element remains from the sketching process, we need to check whether it is actually the case so that we can recover and output the element. And unlike in lecture 15, this process is much tricky.

Now let x be a subsample vector(corresponding to the number of occurrence of each element in this subsample set).

We want to be able to check if $\|x\|_0 = 1$ w.h.p to verify if we succeed in subsampling. This can be further divided into two tasks.

- We first check if $\|x\|_0 \geq 1$. To do so, we randomly pick $v \in \{\pm 1\}^{|U|}$ and check if $v^T x = 0$. For some $x_i \neq 0$

$$v^T x = 0 \Leftrightarrow -v_{-i}^T x_{-i} = v_i x_i \quad (v_{-i} \text{ is vector v without coordinate i})$$

  since v_i is randomly picked from $\{\pm 1\}$, the probability of a false positive(i.e. $v^T x = 0$ for some $x \neq 0$) is at most $\frac{1}{2}$. Therefore by repeating $\log |U|$ times, we succeed with high probability.

- Suppose we know $\|x\|_0 \geq 1$, we want to know if there are more than one element in this subsample set. To do this, we randomly split x into x_1 and x_2 and use the method described in the first part above. If both $v_1^T x_1 \neq 0$ and $v_2^T x_2 \neq 0$ happens, we are certain that $\|x\|_0 \geq 2$. And we make mistake with probability at most $\frac{7}{8}$. [2] So again by repeating this $\log |U|$ times, we succeed in this task with high probability.

4.3.3 recover the index of the unique element

If for some subsample vector x, $\|x\|_0 = 1$, we can use a simple technique to recover the non-zero coordinate(i.e. the unique element) i^*

- Pick $v' = (1, \ldots, 1)$ and $v = (1, 2, 3, \ldots, |U|)$
- Maintain $v^T x$ and $(v')^T x$ during the stream.
- We have
$$v^T x = x_{i^*} i^*, \quad (v')^T x = x_{i^*}$$
So we output $i^* = \frac{(v^T x)}{(v')^T x}$

References

[I01] Piotr Indyk A small approximately min-wise independent family of hash functions. *Journal of Algorithms*, 38.1 (2001): 84-90.

[BCF00] Broder, Andrei Z., et al. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60.3 (2000): 630-659.

[2] To be specific, with probability $\frac{1}{2}$, two unique element are not both in x_1 or x_2. Then with probability $\frac{1}{4}$ there is no false positive on both x_1 and x_2. This means we succeed with probability $\frac{1}{8}$.