CS-455 Topics in Theoretical Computer Science February 27, 2017

Lecture 2

Lecturer: Michael Kapralov *Scribes: Aida Mousavifar and Junxiong Wang*

1 Cont'd

In the last lecture, we introduce a stream algorithm to distinguish **Yes case** and **No case** in the *distinct elements problem*. However, by the linearity of expectation, the expected size of sample set S is $\frac{n}{t}$. To achieve an $o(n)$ space algorithm, we modify the sample methods. We firstly introduce the limited independence.

Definition 1 *A family of hash functions $H = \{h : [n] \to U\}$ is a pair-wise independent if for every pair of distinct elements $x, y \in [n]$ and every $a, b \in U$, we have:*

$$Pr_{h\in H}[h(x) = a \wedge h(y) = b] = \frac{1}{|U|^2}.$$

An example of pair-wise hash family is $\{ax + b \mod p\}$ where $a, b, x \in \mathbb{Z}_p$ and a, b are selected independently uniformly at random and p is a prime number.

1.1 Sample

(1) Choose a parameter $B = \Theta(t)$.

(2) Select a hash function h from a pair-wise independent family H.

(3) Let $S = \{i \in [n], h(i) = 1\}$

1.2 Analysis

- No Case: $k \le t$

$$\begin{aligned} Pr[\textstyle\sum x_i > 0] &\le Pr[\text{supp}(X) \cap S \ne \emptyset] \\ &\le Pr_{i\in \text{supp}(X)}[i \in S] \\ &\le \frac{k}{B} \\ &\le \frac{t}{B} \end{aligned}$$

where the second inequality fellows from *union bound*.

- Yes Case: $k \ge 2t$

$$\begin{aligned} Pr[\textstyle\sum x_i > 0] &\le \sum Pr[i \in S] - \sum_{i,j\in \text{supp}(X), i\ne j} Pr[i \in S \wedge j \in S] \\ &= \frac{k}{B} - \sum_{i,j\in \text{supp}(X), i\ne j} \frac{1}{B^2} \\ &= \frac{k}{B} - \frac{k(k-1)}{B} \\ &\le \frac{2t}{B} - \frac{(2t)^2}{B^2} \end{aligned}$$

where the first inequality fellows from *Inclusion-exclusion principle* and the second inequality fellows from the definition of *pair-wise independent hash family.*

By taking $B = 16t$, we can obtain the following inequality.

$$\begin{aligned} Pr[Yes] - Pr[No] &\leq \frac{t}{B} - \frac{4t^2}{B^2} \\ &\leq \frac{t}{B}(1 - \frac{1}{4}) \\ &\leq \frac{3}{4}\frac{t}{B} \end{aligned}$$

Once we have ALG_t that distinguish $k < t$ and $k \geq 2t$, to solve the *distinct elements problem*, we can run ALG_t for $t = 1, 2, 4, \cdots, n$. Thus overall algorithm is $O(\log n \log \frac{1}{\delta} \log n)$ with the failure probability at most $\delta \log n$.

2 Distinct Elements Problem

In this lecture, we start by counting the number of distinct elements in a stream. Later on, we focus on computing l_p norm of the frequency vector of the input stream.

2.1 Flajolet-Martin algorithm

The Flajolet-Martin algorithm is an algorithm for approximating the number of distinct elements in a stream with a single pass and space-consumption logarithmic in the maximal number of possible distinct elements in the stream. The algorithm was introduced by Philippe Flajolet and G. Nigel Martin [4]. Later it has been improved by Marianne Durand and Philippe Flajolet [2]. Flajolet et al. in [4] introduced probabilistic method of counting which was inspired from a paper by Robert Morris Counting large numbers of events in small registers. Morris in his paper says that if the requirement of accuracy is dropped, a counter n can be replaced by a counter $\log n$ which can be stored in $\log \log n$ bits [3]. Flajolet et al. in [4] improved this method by using a hash function h which is assumed to uniformly distribute the element in the hash space (a binary string of length L)

We present Flajolet-Martin algorithm, which uses $O(\log \log n)$ space to solve the approximate counting distinct elements problem. In the following algorithm, by $[n]$ we denote $\{1, \cdots, n\}$.

2.1.1 FM algorithm

(1) Pick a random hash function h: $[n] \to [0, 1]$.

(2) Maintain a counter $X = \min_{i \in \text{stream}} h(i)$.

(3) Output $\tilde{t} = \frac{1}{X} - 1$.

Let us first prove two claims regarding the expected value and the variance of X in above algorithm.

Claim 2 *Let $X_1, \cdots, X_t$ be t i.i.d random variables from the distribution UNIF$[0, 1]$. Also, let $X = \min_{1 \leq i \leq t} X_i$, then*

$$\mathbb{E}[X] = \frac{1}{t+1}.$$

Proof

$$\begin{aligned}
\mathbb{E}[X] &= \int_0^1 Pr[X \geq \lambda] d_\lambda \\
&= \int_0^1 Pr[\min_{1\leq i\leq t} X_i \geq \lambda] d_\lambda \\
&= \int_0^1 \prod_{1\leq i\leq t} (Pr[X_i \geq \lambda]) d_\lambda \\
&= \int_0^1 (1-\lambda)^t d_\lambda \\
&= \int_0^1 \lambda^t d_\lambda \\
&= \frac{1}{t+1}
\end{aligned}$$

■

Claim 3 *Let $X_1, \cdots, X_t$ be t i.i.d random variables from the distribution UNIF$[0,1]$. Also, let $X = \min_{1\leq i\leq t} X_i$, then*

$$Var[X] = O(\frac{1}{t^2}).$$

Proof Recall that, one can compute the variance of a random variable using the following formula:

$$\text{Var}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2.$$

Therefore, let us first compute $E[X^2]$.

$$\begin{aligned}
\mathbb{E}[X^2] &= \int_0^1 Pr[X^2 \geq \lambda] d_\lambda \\
&= \int_0^1 Pr[X \geq \sqrt{\lambda}] d_\lambda \\
&= \int_0^1 (1-\sqrt{\lambda})^t d_\lambda \\
&= 2\int_0^1 u^t(1-u) d_u \\
&= 2(\frac{1}{t+1} - \frac{1}{t+2}) \\
&= \frac{2}{(t+1)(t+2)}
\end{aligned}$$

So

$$\begin{aligned}
\text{Var}[X] &= E[X^2] - (E[X])^2 \\
&= \frac{2}{(t+1)(t+2)} - \frac{1}{(t+1)^2} \\
&= O(\frac{1}{t^2}).
\end{aligned}$$

■

Thus, by Chebyshev we get that the probability of failure of FM Algorithm is

$$\Pr[|\tilde{t} - \mathbb{E}[\tilde{t}]|| > \epsilon t] = \Pr[|\tilde{t} - t]| > \epsilon t] \leq \frac{Var[\tilde{t}]}{t^2\epsilon^2} = O(\frac{1}{\epsilon^2})$$

So far, we have the right mean, but a bad variance. How do we fix that? One way to remedy this problem is to run a number Q (to be determined later) of independent rounds of FM Algorithm, and then output the average as an estimate. Given that the outputs are i.i.d random variables, the variance goes down, but the mean doesn't change. In light of this observation, we introduce FM+, a modified version of FM.

2.2 FM+ algorithm

In this section we improve FM algorithm using the averaging method. FM+ algorithm provides an (ϵ, δ)-approximation by repeating FM Algorithm many times and then returning the average. We say a (randomized) algorithm provides (ϵ, δ)-approximation iff it outputs an estimate that is $(1 \pm \epsilon)$ multiplicative factor of the actual value with probability at least $(1 - \delta)$.

(1) For $Q = O(\frac{1}{\epsilon^2})$, maintain Q copies of FM Algorithm. Let $X_1, X_2 \cdots X_Q$ be the output of the Q copies.

(2) Let $X = \frac{1}{Q}\sum_{j=1}^{Q} X_j$.

(3) Output $\frac{1}{X} - 1$.

By linearity of expectation, the expected value of FM+'s estimate is still t. However, the variance becomes smaller. In particular, the variance of the estimate is now $\frac{\text{Var}(X_1)}{Q}$. Note that the space required by FM+ becomes $O(\epsilon^{-2}\log n)$. Thus, we now obtain an algorithm that requires $O(\log n)$ space and returns $(1+\epsilon)$-approximation with constant probability. In the following section, we focus on improving the success probability of the FM+ algorithm.

2.3 FM++ algorithm

As we saw in the previous lectures, we can use median trick to improve the success probability of an algorithm. Therefore, we also apply this trick to FM+ algorithm and get the FM++ algorithm. To this end, create $Q \cdot R$ hash functions and split them into R distinct groups (each of size Q). Within each group take the mean of the Q results. Finally use the median for aggregating together the R group estimates as the final estimate.

(1) For $Q = O(\frac{1}{\epsilon^2})$ and $R = O(\log\frac{1}{\delta})$, maintain $Q \cdot R$ copies of FM Algorithm. Let $X^{r,q}$ $\forall r = 1, \ldots R$, $\forall q = 1, \ldots Q$ be the output of the $Q \cdot R$ copies.

(2) Let $X^r = \frac{1}{Q}\sum_{q=1}^{Q} X^{r,q}$ $\forall r = 1, \ldots R$

(3) Let $X = median_{r=1,\ldots R}\{X^r\}$

(4) Output $\frac{1}{X} - 1$

By applying Chernoff, it's easy to show that by choosing $R = O(\log\frac{1}{\delta})$ we get a δ (any constant less than half would suffice) probability of failure. The space requirement of FM++ now becomes

$O(\log \frac{1}{\delta} \frac{1}{\epsilon^2} \log n)$. Flajolet et al. [3] further improve the space complexity of the above mentioned algorithm to $O(\frac{1}{\epsilon^2} \log \log n + \log n)$ by storing the position of the least significant bit.

In 2010 article "An optimal algorithm for the distinct elements problem",[1] Daniel M. Kane, Jelani Nelson and David P. Woodruff give an improved algorithm, which computes a $(1 \pm \epsilon)$- approximation using an optimal $O(\frac{1}{\epsilon^2} + \log n)$ bits of space with $\frac{2}{3}$ success probability.

3 Turnstile Model

Much of the streaming literature is concerned with computing statistics on frequency distributions that are too large to be stored. For this class of problems, there is a vector $x = (x_1, \dots, x_n)$ (initialized to the zero vector) that has updates presented to it in a stream. The goal of these algorithms is to compute functions of x using considerably less space than it would take to represent x precisely. There is a common models for updating such streams, called "turnstile" model. In the turnstile model each update is of the form (i, Δ_i) , so that x_i is incremented by some (possibly negative) integer Δ_i. In the "strict turnstile" model, no x_i at any time may be less than zero. So far we know how to compute $f(x) = ||x||_0 = |\{i : x_i \neq 0\}|$. How about $||x||_p = (\sum_{i\in[n]} |x_i|^p)^{\frac{1}{p}}$ for $p > 0$?

There exist algorithms for approximating $||x||_p$ with space complexity poly$(\log n)$ for $p \in [0, 2]$ and space $\Omega(n^{1-\frac{2}{p}})$ for $p > 2$. In this lecture, we present an algorithm for the case that $p = 2$. Note that FM does not apply to strict turnstile model because we are not able to capture the proper minimum after updates. A randomized algorithm seeks to approximate a function $f(x)$ with constant probability while only making a single pass over this sequence of updates and using a small amount of space. All known algorithms in this model are linear sketches: they sample a matrix A from a distribution on integer matrices in the preprocessing phase, and maintain the linear sketch $b = A \cdot x$ while processing the stream. At the end of the stream, they output an arbitrary function of $b = A \cdot x$. Li, Nguyen, Woodruff [5] show that any single pass constant probability streaming algorithm for approximating an arbitrary function $f(x)$ in the turnstile model can also be implemented by sampling a matrix A from the uniform distribution on $O(n \log m)$ integer matrices, with entries of magnitude $poly(n)$, and maintaining the linear sketch $b = A \cdot x$. Any update in the stream such as (i, Δ_i) is equivalent to $b \leftarrow b + A \cdot (\Delta_i \cdot e_i)$ where e_i is the i-th indicator vector.

4 Lower-bounds against deterministic algorithms

In this section, we first prove a lower-bound on the space complexity of any deterministic algorithm for computing the l_p norm. Later, we present a randomized algorithm for computing the l_2 norm.

Claim 4 *Any deterministic streaming algorithm that approximate $||x||_k = \sum_j x_j^k$ up to a (1 ± 0.1)-factor must use $\Omega(n)$ space for every integer $k \neq 1$.*

Proof For every large enough integer n divisible by 4, there exists a collection F of subsets of $\{1, 2, \dots, n\}$ of cardinality $\frac{n}{4}$ such that for any two distinct subsets such $G_1 \neq G_2 \in F$, $|G_1 \cap G_2| \leq \frac{n}{8}$ and $|F| \geq 2^{cn}$ for any constant c.

Toward contradiction, assume that there is a deterministic algorithm ALG(G), which uses less than cn bits for small constant c. Therefore, ALG(G) has less than 2^{cn} possible states. Thus, if one runs this algorithm on all the sets in F, by the pigeonhole principle there exist $G_1 \neq G_2 \in F$ such that the state of the algorithm ends up being the same, i.e., $ALG(G1) = ALG(G2) = S$. So if we concatenate G_1 to end of these two strings, the output of the algorithm would be the same, i.e., $ALG(G1, G1) = ALG(G2, G1)$. However, we show that the l_k-norm$(k \neq 1)$ in $G1, G1$ and $G2, G1$ are far from each other. Let x_1 and x_2 denote the frequency vector of $G1, G1$ and $G2, G1$, respectively. Now we consider the following cases:

- l_0-norm:
  $||x_1||_0 = \frac{n}{4}, ||x_2||_0 \geq 2 \cdot \frac{n}{4} - \frac{n}{8}$
  $\Rightarrow ||x_2||_0 \geq \frac{n}{4} + \frac{n}{8} > (1+0.1)\frac{n}{4} = (1+0.1)||x_1||_0$
- l_k-norm ($k \geq 2$):
  $||x_1||_k = 2^k \cdot \frac{n}{4}$
  Let $\Delta = |G_1 \cap G2|$. We know that $\Delta \leq \frac{n}{8}$
  $||x_2||_k = 2 \cdot \frac{n}{4} - 2\Delta + \Delta \cdot 2^k \leq 2 \cdot \frac{n}{4} - 2\frac{n}{8} + \frac{n}{8} \cdot 2^k \leq (2^k + 2)\frac{n}{8}$
  $\Rightarrow (1+0.1)||x_2||_k \leq 2^k \cdot \frac{n}{4} = ||x_1||_k$

This contradicts the fact that ALG is a (1 ± 0.1)-approximation algorithm.
■

5 AMS Sketch

In this section we introduce a randomized algorithm for computing l_2-norm for a frequency vector x. To this end, let us first define k-wise independent hash function.

Definition 5 *A family of hash functions $H = \{h : [n] \to U\}$ is a k-wise independent if for any k distinct elements $(x_1, \cdots, x_k) \in U^k$ and any numbers $(u_1, \cdots, u_k)$, we have:*

$$Pr_{h \in H}[h(x_1) = u_1 \wedge \cdots \wedge h(x_k) = u_k] = (\frac{1}{|U|})^k.$$

```
(1) Pick a 4-wise independent hash function: $h : [n] \to \{-1, +1\}$
(2) Let $\sigma_i = h(i)$ so $\sigma \in \{-1, 1\}^n$
(3) Maintain $Z = < \sigma, x >$
(4) Output $Z^2$
```

Let us first prove two claims regarding the expected value and the variance of Z^2 in above algorithm.

Claim 6

$$\mathbb{E}[Z^2] = ||x||_2^2$$

Proof

$$\begin{aligned}
\mathbb{E}[Z^2] &= \mathbb{E}[(\sum_{i \in [n]} \sigma_i x_i)^2] \\
&= \mathbb{E}[\sum_{i,j \in [n]} \sigma_i \sigma_j x_i x_j] \\
&= \mathbb{E}[\sum_{i \in [n]} \sigma_i^2 x_i^2] + \mathbb{E}[\sum_{i \neq j \in [n]} \sigma_i \sigma_j x_i x_j] \\
&= \mathbb{E}[\sum_{i \in [n]} x_i^2] + \sum_{i \neq j \in [n]} \mathbb{E}[\sigma_i]\mathbb{E}[\sigma_j] x_i x_j \\
&= ||x||_2^2
\end{aligned}$$

■

Claim 7

$$Var[Z^2] \leq 2||x||_2^2$$

Proof Recall that, one can compute the variance of a random variable using the following formula:

$$\text{Var}[Z^2] = \mathbb{E}[Z^4] - (\mathbb{E}[Z^2])^2.$$

Therefore, let us first compute $E[Z^4]$.

$$Z^4 = (\sum_{i\in[n]} \sigma_i x_i)(\sum_{j\in[n]} \sigma_j x_j)(\sum_{k\in[n]} \sigma_k x_k)(\sum_{l\in[n]} \sigma_l x_l)$$

Let us consider several types of terms:

- all the indexes are equal $i = j = k = l$:
  $\sum_{i\in[n]} \sigma_i^4 x_i^4 = \sum_{i\in[n]} x_i^4$
- the indexes matched 2 by 2:
  $\binom{4}{2} \sum_{i<j} (\sigma_i \sigma_j x_i x_j)^2 = 6 \sum_{i<j} x_i^2 x_j^2$
- terms with a single (unmatched) multiplier: in this case, since the value $\mathbb{E}[\sigma_i] = 0$ for any $1 \leq i \leq n$, then the coefficient of such terms are zero.

Therefore, $\mathbb{E}[Z^4] = \sum_{i\in[n]} x_i^4 + 6 \sum_{i<j} x_i^2 x_j^2$. So the variance of Z^2 is :

$$\begin{aligned}
\text{Var}[Z^2] &= \mathbb{E}[Z^4] - (\mathbb{E}[Z^2])^2 \\
&= \sum_{i\in[n]} x_i^4 + 6 \sum_{i<j} x_i^2 x_j^2 - (\sum_i x_i^2)^2 \\
&= \sum_i x_i^4 + 6 \sum_{i<j} x_i^2 x_j^2 - \sum_i x_i^4 - 2 \sum_{i<j} x_i^2 x_j^2 \\
&= 4 \sum_{i<j} x_i^2 x_j^2 \\
&\leq 2 (\sum x_i^2)^2 \\
&= 2||x||_2^4
\end{aligned}$$

■

We have $\mathbb{E}[Z^2] = ||x||_2^2$ and $\text{Var}[Z^2] \leq 2||x||_2^4$. Now we improve the precision of the estimate by repeating the algorithm for a sufficient number of times (independently) and using the average as an estimate.

(1) For $t = O(\frac{6}{\epsilon^2})$, maintain t i.i.d copies of the above algorithm. Let $Z_1, Z_2 \cdots Z_t$ be the output of the Z copies.

(2) Let $\tilde{Z} = \frac{1}{t} \sum_{i=1}^t Z_i$.

(3) Output $\tilde{Z}^2$.

By linearity of expectation, we have $\mathbb{E}[\tilde{Z}] = ||x||_2^2$. However, the variance becomes smaller. In particular, the variance of the estimate is now $\frac{\mathrm{Var}(Z_i)}{t} \leq \frac{2}{t}||x||_2^4$. By Chebyshev's inequality we get

$$\begin{aligned} Pr[|\tilde{Z}^2 - ||x||_2^2| > \epsilon||x||_2^2] &\leq \frac{(\frac{2}{t}) \cdot ||x||_2^4}{\epsilon^2||x||_2^4} \\ &\leq \frac{2}{t\epsilon^2} \\ &\leq \frac{1}{3} \end{aligned}$$

References

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 5–14. ACM, 2012.

[2] Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities. In *European Symposium on Algorithms*, pages 605–617. Springer, 2003.

[3] Philippe Flajolet. Approximate counting: a detailed analysis. *BIT Numerical Mathematics*, 25(1):113–134, 1985.

[4] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[5] Christian Konrad. Maximum matching in turnstile streams. In *Algorithms-ESA 2015*, pages 840–852. Springer, 2015.