CS 395T: Sublinear Algorithms Fall 2014

Lecture 3 — Sept. 4, 2014

Prof. Eric Price *Scribe: Zhao Song*

In today's lecture, we will discuss the following problems:

1. Distinct elements
2. Turnstile model
3. AMS - sketch

1 Distinct elements

Given $1, 5, 4, 4, 19, \cdots, \ \in [n]$.

Goal Estimate $k (= \#$ distinct elements) up to factor $(1 \pm \epsilon)$ with $1 - \delta$ probability.

In order to solve the above problem, let's look at the following basic question:

Given t.

Goal Ask either $k \leq t$ or $k \geq 2t$?

Choose a subset $S \subseteq [n]$, then $\forall i \in [n], i \in s$ with probability $\frac{1}{t}$. Record whether the intersection of "stream" and set S is empty, let x denote this event (stream $\cap S) \neq \emptyset$. (Note that S is chosen before you see a stream of integers).

$$Pr[x \ is \ true] = Pr[x] = 1 - (1 - \frac{1}{t})^k$$

(Note that, $Pr[x]$ is a monotonically increasing function on k when t is fixed)

For $k \leq t$, we have

$$Pr[x | k \leq t] \leq 1 - (1 - \frac{1}{t})^t \approx 1 - \frac{1}{e} \approx 0.63$$

For $k \geq 2t$, we have

$$Pr[x | k \geq 2t] \geq 1 - (1 - \frac{1}{t})^{2t} \approx 1 - \frac{1}{e^2} \approx 0.8$$

Repeat to get $x_1, x_2, \cdots, x_m$ independent samples. Return whether $\sum_i x_i \geq 0.7m$. Since $x_i \in \{0, 1\}$ is subgaussian for $\sigma = \frac{1}{2}$, we have that $\sum_i x_i$ is also a subgaussian with $\sigma = \frac{\sqrt{m}}{2}$.

$$Pr[\sum_i x_i \geq \mu + t] \leq e^{\frac{t^2}{2\sigma^2}} = e^{\frac{2t^2}{m}}$$

$$Pr[\sum_i x_i \geq 0.63m + 0.07m] \leq e^{2 \cdot 0.07^2 m}$$

Therefore with $m = \Theta(\log(1/\delta))$, we can distinguish the to cases with $1 - \delta$ probability.

Question: But: how do we store a concise description of S? There can be 2^n such sets, and roughly $\binom{n}{k}$ "likely" sets. So storing S would dominate the space complexity.

Answer 1: Crypto h : SHA-256 [SHA256], or any other crypto hash, and choose roughly $S = \{i | \frac{h(i)}{2^{256}} < \frac{1}{t}\}$. Then there would exist streams that break the algorithm, but it's (hopefully) computationally intractible to find them.

Answer 2: h : pair-wise independent, $s = \{i | h(i) < \frac{1}{t}\}$

Let's look at the general definition of some hash functions first,

Definition 1.1. *Family H of functions from $[n] \to [m]$ is pair-wise independent if with probability*

$$\Pr_{h \in H \;\; x,y \in [n] \;\; c,d \in [m]}[h(x) = c \text{ and } h(y) = d] = \frac{1}{m^2}$$

Example 1.2. *Canonical example: $h(x) = ax + b \pmod m$, where $(a, b) \in [m]$ pair-wise independent if m is a prime $\geq n$.*

Let's consider an algorithm that uses pair-wise independent hash function:

Algorithm: Let H denote a pairwise-independent hash function family, choose $h \in H$ such that $h : [n] \to [B]$, where $B = \Theta(t)$(the constant will be decided later). Consider the set $S = \{i | h(i) = 0\}$.

Then, for the probability of any $x \in S$, we have an upper bound by the union bound:

$$Pr[any\ x \in s] \leq \sum_i Pr[i \in s] = \frac{k}{B}$$

And we have a lower bound by Inclusion-Exclusion[1]:

$$\begin{aligned} Pr[any\ x \in s] &\geq \sum_i Pr[i \in s] - \sum_{i,j} Pr[i \in s \text{ and } j \in s] \\ &= \frac{k}{B} - \frac{k(k-1)}{2B^2} \\ &= \frac{k}{B}(1 - \frac{k-1}{B}) \end{aligned}$$

Let's set $B = 4t$, for $k \leq t$, we have

$$Pr[any\ x \in S] \leq \frac{t}{B} = \frac{1}{4}$$

For $k \geq 2t$, we have

$$Pr[any\ x \in S] \geq \frac{1}{2}(1 - \frac{1}{4}) = \frac{3}{8}$$

For any t, do $\log(\frac{1}{\delta})$ independent samples/examples, each uses $O(\log n)$ spaces. Since there are $O(\log n)$ different ts, then $O(\frac{1}{\epsilon^2} \cdot \log(\frac{\log n}{\delta}))$ total space is used to perform distinct elements.

[1] http://en.wikipedia.org/wiki/Inclusion-exclusion_principle

Idealized streaming algorithm[2]

We now explain the LogLog algorithm of [DF03], which improves the space complexity from roughly $O(\frac{1}{\epsilon^2}\log n)$ to $O(\frac{1}{\epsilon^2}\log\log n)$. One algorithm you could use for distinct elements is the following:

1. Pick a random hash function $h : [n] \to [0, 1]$
2. Define $z = \min\limits_{i \in \text{stream}} h(i)$, then $\frac{1}{z} - 1 \approx k$.

The observation is that you don't need to store z exactly; you only need to remember which of $\log n$ different scales z lies in.

LogLog algorithm

1. Pick a random hash function $h : [n] \to \{0, 1\}$. (Note that h is able to convert a stream of integers to a binary string.)
2. For a string $x \in \{0, 1\}^\infty$, define $\rho(x)$ to be the number of leading zeros from left. (In [DF03], they defined $\rho(x)$ in a similar way, where $\rho(x)$ denotes the position of its first 1-bit, e.g. $\rho(1\cdots) = 1$ and $\rho(001\cdots) = 3$.)
3. Separate elements into m buckets (Analysis in [DF03] shows that $\epsilon = \frac{1.3}{\sqrt{m}}$, here; $\epsilon = \frac{1.05}{\sqrt{m}}$, for HyperLogLog.)
4. Let $m = 2^t$, then the first t binary bits of x denote the index of one of m buckets.
5. Let $\mathcal{M}$ denote the multiset of hashed values, define $z(\mathcal{M}) = \max\limits_{x \in \mathcal{M}} \rho(x)$.
6. For each bucket j, ignore the first t bits and compute z_j.
7. Output $\alpha_m m 2^{\frac{1}{m}\sum z_j}$ to approximate n, where α_m is a constant value (defined in [DF03]).

Total Space : $O(\frac{1}{\epsilon^2}\log\log n + \log n)$, where the first term $\frac{1}{\epsilon^2}\log\log n$ is caused by m buckets and the second term $\log n$ is from hash function.

There exists a better algorithm :

Theorem 1.3. *[KNW10] For a stream of indices in $\{1, 2, \cdots, n\}$, the algorithm computes $(1 \pm \epsilon)$-approximation using an optimal $O(\frac{1}{\epsilon^2} + \log(n))$ bits of space with $\frac{2}{3}$ success probability, where $0 < \epsilon < 1$.*

[2] The details of ISA can be found in Lecture 2 of Course Algorithm for Big Data at Harvard. http://people.seas.harvard.edu/ minilek/cs229r/lec/lec2.pdf

2 Turnstile model

1. Pick a vector $x \in \mathbb{R}^n$, start at 0.
2. Read a stream of updates $(\cdots, (i, \alpha_i), \cdots)$, where $i \in [n]$, α_i is the number of elements to be added or deleted.
3. For each (i, α_i), we update $x_i \leftarrow x_i + \alpha_i$.
4. Compute $f(x)$.

A further restriction is the "strict" turnstile model, where x_i is always ≥ 0, which means the count of any item can not be negative at any time.

What are examples of f that you might want to compute? Well, distinct elements corresponds to

$$f(x) = (\#i | x_i \neq 0) = \|x\|_0 \quad \text{(also called the "sparsity of x")} \tag{1}$$

One may also ask about other norms, e.g. $\|x\|_1$ and $\|x\|_2$, or finding spanning tree, or finding the largest entries.

Estimate $\|x\|_2$ in turnstile model

Let $A \in \mathbb{R}^{m \times n}$ be a Johnson-Lindenstrauss matrix, where $A_{ij} \sim \mu(0, \frac{1}{m})$, $m = O(\frac{1}{\epsilon^2} \log(\frac{1}{\delta}))$, $\|Ax\|_2^2 = (1 \pm \epsilon)\|x\|_2^2$ with probability $1 - \delta$.

Given update (i, α), then we have :

$$\begin{aligned} x &\leftarrow x + \alpha \cdot e_i \\ Ax &\leftarrow Ax + A \cdot e_i \cdot \alpha \\ y &\leftarrow y + \alpha \cdot (column\ i \in A) \end{aligned}$$

where e_i is the "elementary unit vector", a vector of length n with $e_i = \underbrace{00\cdots0}_{i-1}1\underbrace{00\cdots0}_{n-i}$. This means we can maintain the linear "sketch" $y = Ax$ under streaming updates to x.

This would let us estimate $\|x\|_2$ from a small space sketch Ax. The problem is, to do so requires us to remember A, which takes more than mn bits. So how do we solve this? The same way we solved not being able to store S for distinct elements – with hashing and limited independence.

3 AMS - sketch [AMS99]

Definition 3.1. *H is a k-wise independent hash family if*

$$\forall i_1 \neq i_2 \neq \cdots i_k \in [n] \text{ and } \forall j_1, j_2, \cdots, j_k \in [m]$$

$$\Pr_{h \in H}[h(i_1) = j_1 \wedge \cdots \wedge h(i_k) = j_k] = \frac{1}{m^k}$$

AMS Algorithm[3]:

1. Pick a random hash function $h : [n] \to \{-1, +1\}$ from a four-wise independent family.
2. Let $v_i = h(i)$.
3. Let $y =< v, x >$, output y^2.
4. From Lemma 3.1 and 3.2, we know that y^2 is an unbiased estimator with variance big-Oh of the square of its expectation.
5. Sample y^2 $m_1 = O(\frac{1}{\epsilon^2})$ independent times : $\{y_1^2, y_2^2, \cdots, y_{m_1}^2\}$. Use Chebyshev's inequality to obtain a $(1 \pm \epsilon)$ approximation with $\frac{2}{3}$ probability.
6. Let $\overline{y} = \frac{1}{m_1}\sum_{i=1}^{m_1} y_i^2$.
7. Sample $\overline{y}$ $m_2 = O(\log(\frac{1}{\delta}))$ independent times : $\{\overline{y}_1, \overline{y}_2, \cdots, \overline{y}_{m_2}\}$. Take the median to get $(1 \pm \epsilon)$-approximation with probability $1 - \delta$.

Space Analysis : Each of the hash function takes $O(\log n)$ bits to store, and there are $O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$ hash functions in total.

Lemma 3.2. $E[y^2] = \|x\|_2^2$

Proof.

$$\begin{aligned}
E[y^2] &= E[(< v, x >)^2] \\
&= E[\sum_{i=1}^{n} v_i^2 x_i^2 + \sum_{i \neq j} v_i v_j x_i x_j] \\
&= E[\sum_{i=1}^{n} v_i^2 x_i^2] + E[\sum_{i \neq j} v_i v_j x_i x_j] \\
&= \sum_{i=1}^{n} x_i^2 + 0 \\
&= \|x\|_2^2
\end{aligned}$$

where $E[v_i v_j] = E[v_j] \cdot E[v_k] = 0$ since pair-wise independence. □

[3]More details also can be found in : Lecture 2 of Course Algorithm for Big Data at Harvard. http://people.seas.harvard.edu/ minilek/cs229r/lec/lec2.pdf ; Lecture 2 of Course Sublinear Algorithms for Big Datasets at the University of Buenos Aires. http://grigory.github.io/files/teaching/sublinear-big-data-2.pdf

Lemma 3.3. $E[(y^2 - E[y^2])^2] \leq 2\|x\|_2^4$

Proof.

$$\begin{aligned} E[(y^2 - E[y^2])^2] &= E[(\sum_{i \neq j} v_i v_j x_i x_j)^2] \\ &= E[4\sum_{i<j} v_i^2 v_j^2 x_i^2 x_j^2 + 4 \sum_{i \neq j \neq k} v_i^2 v_j v_k x_i^2 x_j x_k + 24 \sum_{i<j<k<l} v_i v_j v_k v_l x_i x_j x_k x_l] \\ &= 4\sum_{i<j} x_i^2 x_j^2 + 4 \sum_{i \neq j \neq k} E[v_i^2 v_j v_k x_i^2 x_j x_k] + 24E[\sum_{i<j<k<l} v_i v_j v_k v_l x_i x_j x_k x_l] \\ &= 4\sum_{i<j} x_i^2 x_j^2 + 0 + 0 \\ &\leq 2\|x\|_2^4 \end{aligned}$$

where $E[v_i^2 v_j v_k] = E[v_j] \cdot [v_k] = 0$ since pair-wise independence,

and $E[v_i v_j v_k v_l] = E[v_i]E[v_j]E[v_k]E[v_l] = 0$ since four-wise independence. □

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[DF03] Marianne Durand and Philippe Flajolet. Loglog Counting of Large Cardinalities. *ESA*, LNCS 2832:605–617, 2003.

[KNW10] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* 2010.

[SHA256] Descriptions of SHA-256, SHA-384, and SHA-512. *NIST*, 2014-09-07, http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf