L9: Distinct Elements

scribe: *Mengyang Wang*

Overview

Data stream is a very useful model in various domains such as data mining and network monitoring. A data stream can be viewed as a sequence of items. Each of them can only be seen once(or more than once if making multiple passes). In streaming model, space is restricted, so it will be infeasible to get exact answer for some problems. However, in many streaming applications, approximate answers are sufficient.

9.1 Streaming

Suppose there is a stream $A = \langle a_1, a_2, ..., a_m \rangle$. The current size of this stream is m and for each element a_i, we have $a_i \in [n]$. Note that a_i is not necessary to be an integer, but we can always map it to an integer. So it will take about $\log n$ bits to store a_i and $\log m$ space to store m, which is the number of elements we have seen by now. The goal is to do some computation using only $\mathsf{poly}(\log m, \log n)$ space.

9.2 Distinct-Element Problem

Given a stream $A = \langle a_1, a_2, ..., a_m \rangle$ with each $a_i \in [n]$. We define $f_j = |\{a_i \in A \mid a_i = j\}|$, which is the number of elements in the stream that have value j. The goal of this problem is to find the number of distinct element F_0, denote as $F_0 = |\{\ j \in [n] \mid f_j >= 0\ \}|$.

For integer p, we denote zeros$(p) = \max\{\ i \mid 2^i$ divides $p\ \}$. This function computes the number of zeros that binary form of p end with. For example, if $p = 8_{(10)} = 1000_{(2)}$, then zeros$(p) = 3$.

Algorithm 9.2.1 Distinct-Elements

Initialization:
Choose a random hash function $h : [n] \to [n]$
$z \leftarrow 0$

Stream: A
if zeros$(h(a_i)) > z$ **then**
  $z \leftarrow$ zeros$(h(a_i))$

Output: $2^{z+\frac{1}{2}}$

Let there be k distinct elements(k is just for analysis, we don't know the value of k). We expect $1/k$ distinct elements to have zeros$(a_i) \geq \log k$. And we expect no elements to have zeros$(a_i) \gg \log k$. So z can be used to estimate $\log k$.

Analysis Let $X_{r,j}$ be an indicator random variable for event zeros$(h(j)) > r$. And let $Y_r = \sum_j X_{r,j}$. Let t denote the value of variable z at end of stream. We have

$$Y_r > 0 \Leftrightarrow t \geq r$$

And it can be rewritten as

$$Y_r = 0 \Leftrightarrow t < r$$

The expectation of $X_{r,j}$ is

$$\mathbf{E}[X_{r,j}] = \mathbf{Pr}[\text{ zeros}(h(j)) \geq r\ ] = \mathbf{Pr}[\ 2^r \text{ divides } h(j)\ ] = \frac{1}{2^r}$$

The expectation and variance of Y_r are

$$\mathbf{E}[Y_r] = \sum_j \mathbf{E}[X_{r,j}] = \frac{k}{2^r}$$

$$\begin{aligned} \mathbf{Var}[Y_r] &= \sum_j \mathbf{Var}[X_r, j] \quad (\ \mathbf{Var}[X_{r,j}] = \mathbf{E}[(X_{r,j})^2)] - \mathbf{E}[X_{r,j}]^2\ ) \\ &\leq \sum_j \mathbf{E}[X_{r,j}^2] \\ &= \sum_j \mathbf{E}[X_{r,j}] \\ &= \frac{k}{2^r} \end{aligned} \tag{9.1}$$

Using Markovs inequality, we have

$$\mathbf{Pr}[Y_r > 0] = \mathbf{Pr}[Y_r \geq 1] \leq \frac{\mathbf{E}[Y_r]}{1} = \frac{k}{2^r}$$

Using Chebyshev's inequality, we get

$$\begin{aligned} \mathbf{Pr}[Y_r = 0] &= \mathbf{Pr}[\ |Y_r - E[Y_r]| \geq \frac{k}{2^r}\ ] \\ &\leq \frac{\mathbf{Var}[Y_r]}{(k/2^r)^2} \\ &\leq \frac{2^r}{k} \end{aligned} \tag{9.2}$$

The algorithm output will be $\hat{k}$, which is an estimated value of k, so $\hat{k} = 2^{t+\frac{1}{2}}$. Let a to be the smallest integer such that $2^{a+1/2} \geq 3k$, then

$$\mathbf{Pr}[\hat{k} \geq 3k] = \mathbf{Pr}[t \geq a] = \mathbf{Pr}[Y_a > 0] \leq \frac{k}{2^a} \leq \frac{\sqrt{2}}{3} < \frac{1}{2}$$

Let b to be the largest integer such that $2^{b+\frac{1}{2}} < k/3$, then

$$\mathbf{Pr}[\hat{k} \leq k/3] = \mathbf{Pr}[t \leq b] = \mathbf{Pr}[Y_{b+1} = 0] \leq \frac{2^{b+1}}{k} \leq \frac{\sqrt{2}}{3} < \frac{1}{2}$$

That possibility of $\hat{k}$ larger than $3k$ or smaller than $k/3$ is no more than 50%. This is really weak bound. And in this case $\varepsilon = 3, \delta = 1/2$, so it is a $(3, \frac{1}{2})$-approximation. One way to improve it is to use median trick, which will be discussed later.

9.3 Median Trick

The median trick can help us to reduce the error probability to any $\delta > 0$. Suppose we are running s parallel, independent hash functions on the above procedure. The output of will be $\hat{K} = \{\hat{k}_1, \hat{k}_2, ..., \hat{k}_s\}$. Let $\bar{k}$ denote the median of $\hat{K}$. Then if $\bar{k} > 3k$, there must be at least $s/2$ values in $\hat{K}$ than greater than $3k$. In $\hat{K}$, each $\hat{k}_i$ that has value no greater than $3k$ with probability of $1/2$. So $(\frac{1}{2})^{s/2} \leq \delta$. Solve for s we have $s \geq 2\log(1/\delta)$. Similarly, we can get same answer for lower bound. Using $s = 2\log(2/\delta)$, take median $\bar{k}$ is an $(\varepsilon = 3, \delta)$-approximation of the number of distinct elements.

In this algorithm, we need $O(\log\log n)$ bits to store t. We also have $O(\log(1/\delta))$ hash functions, each of them needs $O(\log n)$ space. So the space used by this algorithm will be $O(\log(1/\delta) \cdot \log n)$. There is a better algorithm that has space bound $O(\log m + 1/\varepsilon^2 \cdot \log(1/\varepsilon + \log\log m))$.

Appendix

Markov Inequality If X is a random variable and $a > 0$, then

$$\mathbf{Pr}(\ |X| \geq a) \leq \frac{\mathbf{E}[|X|]}{a}$$

Chebyshev's Inequality If Y a random variable and $b > 0$, then

$$\mathbf{Pr}(|Y - \mathbf{E}[Y]| \geq b) \leq \frac{\mathbf{Var}[Y]}{b^2}$$