Algorithms for Big Data (III)

Chihao Zhang

Shanghai Jiao Tong University

Sept. 29, 2019

Review of the Last Lecture

Last time, we proved a few useful concentration inequalities.

We introduced the notion of universal families of Hash functions.

We constructed a 2-universal universal family of Hash functions.

Review: the construction

Let m, n be two integer and $p \geq m$ be a prime.

The family

$$\mathcal{H} = \{h_{a,b} : 1 \leq a \leq p-1, 0 \leq b \leq p-1\},$$

where each $h_{a,b} : [m] \to [n]$ is defined as

$$h_{a,b}(x) = (ax + b \mod p) \mod n.$$

We proved that for every $x \neq y$,

$$\mathbf{Pr}_{h \in \mathcal{H}}[h(x) = h(y)] \leq \frac{1}{n}.$$

So $\mathcal{H}$ is a 2-universal Hash function family.

Strongly 2-Universal Hash Family

Recall that if we further require that for any u, v,

$$\mathbf{Pr}_{h\in\mathcal{H}}\left[h(x) = u \wedge h(y) = v\right] = \frac{1}{n^2},$$

then $\mathcal{H}$ is called strongly 2-universal family of Hash functions.

When $m = n = p$ are primes, the we can modify the previously constructed $\mathcal{H}$ to get a strong 2-universal family.

In this case, we have

$$\mathcal{H} = \left\{h_{a,b}(x) = ax + b \mod p \,:\, 0 \le a, b \le p - 1\right\}.$$

Proof

Lemma

The equation $ax + b = 0 \mod p$ has unique solution (in $\mathbb{F}_p$) if $a \neq 0$ and p is a prime.

The equations $h_{a,b}(x_1) = y_1$ and $h_{a,b}(x_2) = y_2$ are equivalent to

$$ax_1 + b = y_1 \mod p, \quad ax_2 + b = y_2 \mod p.$$

They have a unique solution

$$a = \frac{y_2 - y_1}{x_2 - x_1} \mod p, \quad b = y_1 - ax_1 \mod p.$$

Therefore,

$$\mathbf{Pr}_{h_{a,b} \in H}[h_{a,b}(x_1) = y_1 \wedge h_{a,b}(x_2) = y_2] = \frac{1}{p^2}.$$

The General Case

The Hash family we just constructed has the restriction that $m = n$

We can naturally generalize $m = p$ to $m = p^k$.

Write every number x in base p:

$$x = x_0 + x_1 \cdot p + x_2 \cdot p^2 + \ldots x_{k-1} \cdot p^{k-1}.$$

For every $\bar{a} = (a_0, a_1, \ldots, a_{k-1})$, with $0 \le a_i \le p - 1$ and $0 \le b \le p - 1$, define

$$h_{\bar{a},b}(x) = \left(\sum_{i=0}^{k-1} a_i x_i + b \right) \mod p.$$

Then

$$\mathcal{H} = \{ h_{\bar{a},b} : 0 \le a_i \le p - 1, 0 \le b \le p - 1 \}.$$

Proof

Assuming $x \neq y$ and they differ on the position i ($x_i \neq y_i$).

For every $u, v \in \{0, 1, \ldots, p-1\}$, we have equations

$$\begin{cases} a_i x_i + b = \left(u - \sum_{j \neq i} a_j x_j\right) \mod p \\ a_i y_i + b = \left(v - \sum_{j \neq i} a_j y_j\right) \mod p \end{cases}$$

For fixed x, y, u, v and $\left\{a_j\right\}_{j \neq i}$, a unique pair (a_i, b) (out of p^2 pairs) is determined.
Therefore,

$$\mathbf{Pr}_{h_{\bar{a},b} \in \mathcal{H}}\left[h_{\bar{a},b}(x) = u \wedge h_{\bar{a},b}(y) = v\right] = \frac{1}{p^2}.$$

Counting Distinct Elements

Back to the streaming model, we are given a sequence of numbers $\langle a_1, \ldots, a_m \rangle$ where each $a_i \in [n]$.

It defines a frequency vector $\mathbf{f} = (f_1, \ldots, f_n)$ where $f_i = |\{k \in [m] \;:\; a_k = i\}|$.

We want to compute the number $d = |\{i \in [n] \;:\; f_i > 0\}|$.

The value d is the number of distinct elements in the stream.

The AMS Algorithm

The algorithm is named after Alon, Matias and Szegedy.

For every integer $p > 0$, we use $\mathrm{zero}(p)$ to denote number of trailing zeros of p in binary.

$$\mathrm{zero}(p) \triangleq \max \left\{ i : 2^i \text{ divides } p \right\} .$$

Algorithm AMS Algorithm for Counting Distinct Elements

Init:

A random Hash function $h : [n] \to [n]$ from a 2-universal family

$Z \leftarrow 0$

On Input y**:**

if $zero(h(y)) > Z$ **then**

$\quad Z \leftarrow zero(h(y))$

end if

Output:

$\widehat{d} = 2^{Z+\frac{1}{2}}$.

Intuition

After applying the Hash function, $h(y)$ is uniform is $[n]$.

The probability that it has more than t trailing zeros is at most 2^{-t}.

Therefore, at least in expectation, if we have d distinct numbers, one of them may have $\log_2 d$ trailing zeros.

We now turn this intuition into a rigorous proof.

For every $0 \le r \le n$, we use a random variable Y_r to denote the number of $h(a_i)$ with trailing zero at least r.

The sequence of variables $\{Y_r\}_{0\le r\le n}$ determines the variable Z since $Z = \max_r \{Y_r > 0\}$.

This motivates us to understand the behavior of Y_r.

For every $k \in [n]$, we denote $X_{k,r}$ as the indicator that $h(k)$ has at least r trailing zeros, then $Y_r = \sum_{k\in[n]:f_k>0} X_{k,r}$.

Using this decomposition, it is not hard to see that $\mathbf{E}[Y_r] = \frac{d}{2^r}$ and $\mathbf{Var}[Y_r] \le \frac{d}{2^r}$.

Applying Markov's inequality, we obtain

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

Applying Chebyshev's inequality, we obtain

$$\mathbf{Pr}\left[Y_r = 0\right] \leq \mathbf{Pr}\left[\left|Y_r - \mathbf{E}\left[Y_r\right]\right| \geq \frac{d}{2^r}\right] \leq \frac{2^r}{d}.$$

We know that $Y_r > 0$ for all $r \leq Z$ and $Y_r = 0$ for all $r > Z$.

Therefore, Z cannot be too far from $\log_2 d$:

- if $Z \ll \log_2 d$, we can find a small r with $Y_r = 0$, which happens with small probability;
- if $Z \gg \log_2 d$, we can find a big r with $Y_r > 0$, which happens with small probability.

If $\widehat{d} \le \frac{d}{3}$, let r be the largest integer with $2^{r+\frac{1}{2}} \le \frac{d}{3}$.

$$\mathbf{Pr}\left[\widehat{d} \le \frac{d}{3}\right] = \mathbf{Pr}\left[Z \le r\right] = \mathbf{Pr}\left[Y_{r+1} = 0\right] \le \frac{2^{r+1}}{d} \le \frac{\sqrt{2}}{3}.$$

If $\widehat{d} \ge 3d$, let r be the smallest integer with $2^{r+\frac{1}{2}} \ge 3d$.

$$\mathbf{Pr}\left[\widehat{d} \ge 3d\right] = \mathbf{Pr}\left[Z \ge r\right] = \mathbf{Pr}\left[Y_r > 0\right] \le \frac{d}{2^r} \le \frac{\sqrt{2}}{3}.$$

The algorithm costs $O(\log n)$ bits of memory.

Median

We can apply the standard Median trick to the AMS algorithm. (Excercise)

Using $O(\log \frac{1}{\delta} \log n)$ bits of memory, we can obtain

$$\mathbf{Pr}\left[\frac{d}{3} \leq \widehat{d} \leq 3d\right] \geq 1 - \delta.$$

The BJKST Algorithm

The following improvement of AMS is due to Bar-Yossef, Jayram, Kumar, Sivakumar and Trevisan.

Algorithm BJKST Algorithm for Counting Distinct Elements

Init: Random Hash functions $h : [n] \to [n]$, $g : [n] \to [b\varepsilon^{-4} \log^2 n]$, both from 2-universal families; $Z \leftarrow 0$, $B \leftarrow \varnothing$

On Input y**:**

if $zero(h(y)) \geq Z$ **then**

  $B \leftarrow B \cup \{(g(y), zeros(h(y)))\}$

  while $|B| \geq c/\varepsilon^2$ **do**

    $Z \leftarrow Z + 1$

    Remove all (α, β) with $\beta < Z$ from B

  end while

end if

Output: $\widehat{d} = |B|\, 2^Z$
