CS 493: Algorithms for Massive Data Sets	Streaming Algorithms (Cont.)
03/28/2002	Scribe: Fengzhou Zheng

1 Review

1.1 Stream Model and Frequency Moments

In **stream model**, the data to be processed are given to us one by one as a sequence and cannot be stored, mainly due to the consideration of space complexity. So in this model there is not a single moment when we have the full data set in hand. However, we are still able to learn or estimate some global properties of the sequence of data, such as frequency moments.

Let $A = (a_1, a_2, \ldots, a_m)$ be a sequence of elements, where each a_i is a member of $N = \{1, 2, \ldots, n\}$. Let m_i denote the number of occurrences of element i in the sequence A, and define, for each $k \geq 0$

$$F_k = \sum_{i=1}^{n} m_i^k$$

as the **k'th frequency moment**. In particular, F_0 is the number of distinct elements appearing in the sequence, F_1 is the length of the sequence. The frequency moments of a data set represent important demographic information about the data, and are important features in the context of database applications.

1.2 Estimation in Stream Model

It is rather straightforward to maintain the (exact) frequency moments by maintaining a full histogram on the data, i.e., maintaining a counter m_i for each data value $i \in \{1, 2, \ldots, n\}$, which requires memory of size $\Omega(n)$. However, sometimes it is important to reduce the space complexity. In the previous lecture, we have learned about some randomized algorithms that can approximate F_1 and F_2 using limited memory.

The basic idea behind these randomized algorithms is to devise a random variable Y whose computation only requires limited memory in stream model and whose expectation $E[Y]$ equals to the value V we want to estimate. Using F_2 as an example, we have constructed its estimator Y as follows: given H, a family of hash functions and for each $h \in H$,

$$h : \{1, 2, \ldots, n\} \longrightarrow \{+1, -1\}$$

randomly pick an h from H and let

$$x_i = h(i) \quad i \in \{1, 2, \ldots, n\}$$

$$X = \sum_{i=1}^{n} x_i m_i$$

then, our estimator for F_2 is

$$Y = X^2$$

It is easily to see that X, thus Y, can be computed using only one counter ($O(\log m)$ bits) in stream model. Also

$$\begin{aligned}
E[Y] &= E[(\sum x_i m_i)^2] \\
&= E[\sum x_i^2 m_i^2 + \sum_{i \neq j} x_i x_j m_i m_j] \\
&= \sum m_i^2 + \sum_{i \neq j} m_i m_j E[x_i x_j] \\
&= \sum m_i^2 + \sum_{i \neq j} m_i m_j E[x_i] E[x_j] \\
&= \sum m_i^2
\end{aligned}$$

In order to proceed with the above deduction, we assume following two things about H:

- pair-wise independency, that is, $Pr[h(i) = x, h(j) = y] = Pr[h(i) = x] \cdot Pr[h(j) = y]$ so that $E[x_i x_j] = E[x_i] E[x_j]$ for all $i \neq j$
- $Pr[h(i) = +1] = Pr[h(i) = -1]$ so that $E[x_i] = 0$ for all i

if we randomly pick an h from H.

If we can further prove that $var[Y] \leq O(V^2)$, then we can come up with the following scheme:

$$\underbrace{\underbrace{Y_{11}, Y_{12}, \ldots, Y_{1C_1}}_{C_1} \; \underbrace{Y_{21}, Y_{22}, \ldots, Y_{2C_1}}_{C_1} \cdots}_{C_2}$$

where Y_{ij}'s are independent tests on the same data set. Let Y_i be the mean of $Y_{i1}, Y_{i2}, \ldots, Y_{iC_1}$ and Y be the median of $Y_1, Y_2, \ldots, Y_{C_2}$. We proved following theorem in last lecture. For more details about the proof, please refer to the note of last lecture.

Theorem 1.1 *If $E[Y_{ij}] = V$ and $var[Y_{ij}] \leq O(V^2)$, the above scheme guarantees that $Pr[|Y - V| < \epsilon V] \geq 1 - \delta$, for some $C_1 = O(1/\epsilon^2)$ $C_2 = O(\log(1/\delta))$.*

Notice that in order to prove $var[Y] \leq O(V^2)$ for F_2's case, we further assume that H is 4-wise independent, because there are terms like $E[x_i x_j x_k x_l]$ showing up in the proof.

2 More Examples of Estimation

2.1 Estimation with Multiple Streams

There are situations where we want to estimate some properties about multiple data steams. For example, let us consider the case with two data steams, say

$$f_i = \text{number of element } i \text{ in } 1^{st} \text{ stream}$$

$$g_i = \text{number of element } i \text{ in } 2^{nd} \text{ stream}$$

how do we estimate $\sum(f_i - g_i)^2$? It is easy to show that by choosing

$$X = \sum x_i(f_i - g_i) = \sum x_i f_i - \sum x_i g_i = X^{(1)} - X^{(2)}$$

$$Y = X^2$$

all the known results about F_2 can be applied here.

As a more realistic example, let's try to estimate $\sum f_i g_i$. This value has its meaning in database applications: for two tables $\{< a_i, b_i > | \; a_i \in A, b_i \in B\}$ and $\{< b_j, c_j > | \; b_j \in B, c_j \in C\}$ in a relational database, let f_i and g_i denote the number of occurrences of element i in column B of these two tables, respectively, then $\sum f_i g_i$ indicates the size of the result of joining these two tables. We can construct its estimator as follows:

$$Y = X^{(1)}X^{(2)} = (\sum x_i f_i)(\sum x_i g_i)$$

$$\begin{aligned} E[Y] &= E[(\sum x_i f_i)(\sum x_i g_i)] \\ &= E[\sum x_i^2 f_i g_i + \sum_{i \neq j} x_i x_j f_i g_j] \\ &= \sum f_i g_i \end{aligned}$$

and

$$\begin{aligned} var[Y] &= E[(X^{(1)}X^{(2)})^2] - (E[X^{(1)}X^{(2)}])^2 \\ &= E[(\sum x_i^2 f_i g_i)^2 + (\sum_{i \neq j} x_i x_j f_i g_j)^2 + \sum_{others} \ldots] - (\sum f_i g_i)^2 \\ &= E[(\sum_{i \neq j} x_i x_j f_i g_j)^2] \\ &= \sum_{i \neq j} f_i^2 g_j^2 + \sum_{i \neq j} f_i g_j f_j g_i \\ &\leq (\sum f_i^2)(\sum g_i^2) + \sum_{i \neq j} f_i g_j f_j g_i \\ &\leq (\sum f_i^2)(\sum g_i^2) + (\sum f_i g_i)^2 \\ &\leq 2(\sum f_i^2)(\sum g_i^2) \qquad \text{(Cauchy Inequality)} \end{aligned}$$

Although we have proved that $var[Y] \leq 2(\sum f_i^2)(\sum g_i^2)$, since $(\sum f_i^2)(\sum g_i^2) \geq (\sum f_i g_i)^2$, we can not use the measurement scheme described before to get the same result.

2.2 Estimating F_0

As we have said before, F_0 is the number of distinct elements in the data stream. The most straight forward method to get F_0 would be keeping track of all distinct elements in the stream, but this requires $O(n)$ memory. Here we first give a algorithm that can estimate F_0 using only $O(1)$ memory theoretically, and then the one used in practice.

Suppose there are K distinct elements, $x_1, x_2, \ldots, x_K$, in the stream and H is a family of hash function h's, such that $h : \{1, 2, \ldots, n\} \longrightarrow [0, 1]$ and $h(x_i)$ is evenly distributed between 0 and 1 for any $x_i \in \{1, 2, \ldots, n\}$ if we randomly choose h from H. Let $X = \min_{i=1}^{K} h(x_i)$. Since

$$Pr[X \in [t, t + dt]] = K(1 - t)^{K-1}dt$$

we have

$$\begin{aligned}
E[X] &= \int_0^1 t \cdot K(1-t)^{K-1}dt \\
&= K\int_0^1 (1-(1-t))(1-t)^{K-1}dt \\
&= K(\int_0^1 (1-t)^{K-1}dt - \int_0^1 (1-t)^{K}dt) \\
&= K(\frac{1}{K} - \frac{1}{K+1}) \\
&= \frac{1}{K+1} \\
E[X^2] &= \int_0^1 t^2 \cdot K(1-t)^{K-1}dt \\
&= K\int_0^1 (1-(1-t))(1-(1-t))(1-t)^{K-1}dt \\
&= K\int_0^1 (1-2(1-t)+(1-t)^2)(1-t)^{K-1}dt \\
&= K(\int_0^1 (1-t)^{K-1}dt - 2\int_0^1 (1-t)^{K}dt + \int_0^1 (1-t)^{K+1}dt) \\
&= K(\frac{1}{K} - \frac{2}{K+1} + \frac{1}{K+2}) \\
&= K(\frac{K^2+3K+2-2K^2-4K+K^2+K}{K(K+1)(K+2)}) \\
&= \frac{2}{(K+1)(K+2)} \\
var[X] &= E[X^2] - (E[X])^2 \\
&= \frac{2}{(K+1)(K+2)} - \frac{1}{(K+1)^2} \\
&= \frac{1}{(K+1)^2(1+\frac{2}{K})} \\
&= O(\frac{1}{(K+1)^2})
\end{aligned}$$

So all the analysis about F_2 in the last lecture is applicable in estimating $\frac{1}{K+1}$, and $F_0 = K$.

Although the algorithm above can estimate $\frac{1}{K+1}$ and thus F_0, it requires that the possible values of $h(x_i)$ include all real numbers between 0 and 1, which makes h difficult to implement in practice. So here we give an alternative approximation algorithm, where the ranges of hash functions are just some finite integer sets.

Consider a hash function family H, where each h maps an element in a Galois Field to another element in the same field. To be exact, $h(x) = ax + b$, where a, b are chosen randomly from $GF(2^d)$, $2^d > m$. Here we assume that each distinct data element in the stream corresponds to a distinct element in $GF(2^d)$, so we use same variables, say x, to represent both a data element and its counterpart in the field undistinguishedly. Each element in $GF(2^d)$ can be represented as a bit-vector, let $r(ax + b)$ denote the length of the starting sequence of consecutive 0's in $ax + b$, then our new estimator Y can be constructed as follows:

$$R = \max_{i=1}^{K} r(ax_i + b)$$
$$Y = 2^R$$

For Y, we can prove following theorem:

Theorem 2.1 *$Pr[Y \in [F_0/C, CF_0]] \geq 1 - 3/C$ for any appropriate constant C.*

Proof. First let us define some notations:

$$Z_r(x_i) = \begin{cases} 1 & r(ax_i + b) \geq r \\ 0 & \text{otherwise} \end{cases}$$

$$Z_r = \sum Z_r(x_i).$$

By assuming that H is fully random and pair-wise independent, we have

$$E[Z_r(x_i)] = \frac{1}{2^r}$$

$$E[Z_r] = \frac{F_0}{2^r}$$

$$var[Z_r] = \frac{F_0}{2^r}(1 - \frac{1}{2^r}) < \frac{F_0}{2^r}$$

Now let us compute $Pr[Y > CF_0]$. Let r' be the smallest r that satisfies $2^r > CF_0$.

$$\begin{aligned} Pr[Y > CF_0] &= Pr[2^R \geq 2^{r'}] \\ &= Pr[R \geq r'] \\ &= Pr[Z_{r'} \geq 1] \\ &\leq E[Z_{r'}]/1 \qquad \text{(Markov Bounds)} \\ &= F_0/2^{r'} \\ &< 1/C \end{aligned}$$

For $Pr[Y < F_0/C]$, let r'' be the largest r such that $2^r < F_0/C$.

$$\begin{aligned}
Pr[Y < F_0/C] &= Pr[2^R \leq 2^{r''}] \\
&= Pr[R \leq r''] \\
&= Pr[Z_{r''+1} = 0] \\
&\leq Pr[|Z_{r''+1} - E[Z_{r''+1}]| \geq E[Z_{r''+1}]] \\
&\leq \frac{var[Z_{r''+1}]}{(E[Z_{r''+1}])^2} \qquad \text{(Chebyshev Bounds)} \\
&< 1/\frac{F_0}{2^{r''+1}} \\
&= 2/\frac{F_0}{2^{r''}} \\
&< 2/C
\end{aligned}$$

■

2.3 Construction of H

Using the H for estimating F_2 as an example, we can construct it as an explicit set of $H = \{h_1, h_2, \ldots, h_k\}$ of $k = O(n^2)$ vectors of length n with $+1$,-1 entries, and for every four distinct coordinates $1 \leq i_1 \leq \ldots \leq i_4 \leq n$ and every choice of $\epsilon_1, \ldots, \epsilon_4 \in \{+1,-1\}$ exactly (1/16) fraction of the vectors have ϵ_j in their coordinate number i_j for $j = 1, \ldots, 4$. As described in [2] such sets can be constructed using the parity check matrices of BCH codes.

References

[1] Noga Alon, Yossi Matias, Mario Szegedy: *The Space Complexity of Approximating the Frequency Moments*, Journal of Computer and System Sciences, 58(1): 137-147 (1999).

[2] N. Alon, L. Babai, and A. Itai: *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. Algorithms 7: 567-583 (1986).