CS 368 (Stanford, Spring 2022) Lecture 2

Adapted from CS 369G, Spring 2016. Scribes: Stephen Mussmann, Paris Syminelakis.

Streaming Algorithms

1 Overview

In this lecture, we first derive a concentration inequality for an algorithm for counting distinct element in a stream using pairwise independent hash functions. Then, we present a proof technique to prove lower bounds for streaming algorithms. Finally, we define the concept of a frequency moment and propose an algorithm to estimate F_2.

2 Review

Last lecture we examined the problem of estimating the number of distinct elements in a stream. We found a solution that performed better than the brute force approach of keep an enormous hash table. The solution that was presented was a probabilistic algorithm that gave an $(1+\epsilon)$ approximation with probability $1-\delta$. Further, the algorithm required space $O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta}))$.

More precisely, we defined Y as the minimum hash value of the stream. For a fully independent hash, we found that

$$\mathbb{E}[Y] = \frac{1}{k+1} \tag{1}$$

where k is the number of distinct elements. Recall that we combined many copies of Y using independent hashes to provide an estimate. In particular, we created $O(\log(\frac{1}{\delta}))$ groups of hashes where each group had $O(\frac{1}{\epsilon^2})$ hashes. For the estimate, we computed the mean of each group, then calculated the median of these means.

3 Sketches

Informally, a data sketch is a smaller description of a stream of data that enables the calculation or estimate of a property of the data. An important attribute of sketches is that they are composable. Suppose we have data streams S_1 and S_2 with corresponding sketches $sk(S_1)$ and $sk(S_2)$. We wish there to be an efficiently computable function f where

$$sk(S_1 \cup S_2) = f(sk(S_1), sk(S_2)) \tag{2}$$

4 Bounds for Pairwise Independent Hashes

In the analysis of the distinct element sketch from last time, we relied on a fully independent family of hash functions. Unfortunately, such hash functions are not practical. Here, we examine pairwise independent hashes. Recall that a pairwise independent family of hash functions satisfies

$$\mathbb{P}_h[h(x_1) = y_1, h(x_2) = y_2] = \mathbb{P}_h[h(x_1) = y_1]\mathbb{P}_h[h(x_2) = y_2] \tag{3}$$

4.1 Example

Choose p to be a large prime number. Let $a, b \in [p]$. Let us define the following hash function $h_{a,b} : [p] \to [p]$ as

$$h_{a,b}(x) = ax + b(\text{mod p}) \tag{4}$$

This family of hash functions is pairwise independent.

In particular, for this family of hash functions, we have the following bounds

$$\mathbb{P}[Y < \frac{1}{3k}] < \frac{2}{5} \tag{5}$$

$$\mathbb{P}[Y > \frac{3}{k}] < \frac{1}{3} \tag{6}$$

We can then make $O(\log(\frac{1}{\delta}))$ copies of the hash and take the median to be an estimate that is within a factor of 3 of the true answer with probability $1 - \delta$.

The first bound has an easy proof in the continuous case since we can do a union bound on the interval $[0, \frac{1}{3k}]$ among k elements to get a probabilistic bound of $\frac{1}{3}$.

4.2 General Pairwise Independent Analysis

To get a general bound for pairwise independent hash families, we need to change the algorithm. Instead of taking the mean of the min within a group of hashes, we keep track of the smallest t hash elements. Let y_i be the i^{th} smallest element. For this setup, our estimator is t/y_t.

Theorem 1. *Fix $t = c/\epsilon^2$. With probability $2/3$,*

$$\frac{(1-\epsilon)t}{k} \leq y_t \leq \frac{(1+\epsilon)t}{k}$$

Proof. Let us first prove the second inequality first.

Let $I = [0, (1+\epsilon)\frac{t}{k}]$. Let X_i be an indicator variable for the event $h(x_i) \in I$. Let $X = \sum_i X_i$. Thus, X is the number of hash values in the interval I.

Note that $\mathbb{E}[X] = \sum_i \mathbb{E}[X_i] = k(1+\epsilon)\frac{t}{k} = (1+\epsilon)t$.

$$\mathbb{P}[y_t > \frac{(1+\epsilon)t}{k}] = \mathbb{P}[X < t] = \mathbb{P}[X - \mathbb{E}[X] < -\epsilon t] \leq \mathbb{P}[|X - \mathbb{E}[X]| > \epsilon t] \tag{7}$$

By Chebyshev's inequality,

$$\mathbb{P}[|X - \mathbb{E}[X]| > \epsilon t] \leq \frac{Var[X]}{\epsilon^2 t^2} \tag{8}$$

Let p be the probability that $X_i = 1$. Then, $\mathbb{E}[X_i] = p$ and $Var(X_i) = p(1-p)$. By linearity of expectation, $\mathbb{E}[X] = kp$ and by pairwise independence, $Var(X) = kp(1-p) \leq \mathbb{E}[X] = (1+\epsilon)t$. Thus,

$$\mathbb{P}[|X - \mathbb{E}[X]| > \epsilon t] \leq \frac{(1+\epsilon)t}{\epsilon^2 t^2} = \frac{(1+\epsilon)}{c} \tag{9}$$

We can choose the value of c so that $\frac{(1+\epsilon)}{c} \leq \frac{1}{6}$. Putting this together, we get,

$$\mathbb{P}[y_t < \frac{(1+\epsilon)t}{k}] \leq \frac{1}{6} \tag{10}$$

the proof of the other direction is the same except that the Chebyshev bound is in the other direction. □

For this scheme, we need $O(\log(\frac{1}{\delta}))$ different hashes and for each hash we need to store $t = O(\frac{1}{\epsilon^2})$ values. Thus, the memory of the algorithm will be $O(\frac{1}{\epsilon^2}\log(\frac{1}{\delta})$.

4.3 Proving Lower Bounds on Streaming Algorithms

In this section we describe a common technique for deriving bounds for streaming problems. For this technique, we specify $I_1, ..., I_N$ different streams. If we can show that the algorithm must have a unique state after processing each stream, $\Omega(\log(N))$ is lower bound on the space since the algorithm must distinguish the N different streams.

In order to show that the algorithm must have different states after processing two streams I_i and I_j, we can construct an additional stream I' such that the algorithm must give a different result for $I_i \cup I'$ than for $I_j \cup I'$.

Using this technique, we can prove a lower bound for a deterministic, approximate algorithm for the distinct element problem. Let $\{S_i\}_{i=1}^{N}$ be subsets of $[n]$ that satisfy

$$\forall i : |S_i| = \frac{n}{10} \tag{11}$$

$$\forall i \neq j : |S_i \cap S_j| \leq \frac{n}{20} \tag{12}$$

With a probabilistic construction, we can find 2^{cN} subsets that meet these requirements.

Note that the number of distinct elements in $S_i \cup S_i$ is $(n/10)$ while the number of distinct elements in $S_i \cup S_j$ (for $i \neq j$) is at least $(3/2)(n/10)$. Thus, any deterministic, approximate algorithm must distinguish the N streams corresponding to the sets S_i. Thus, $\Omega(\log(2^{cN})) = \Omega(n)$ is a lower bound on the required space.

4.4 Algorithms for Frequency Moments

Define f_i as the number of times that element i appears in a stream.

The t^{th} frequency moment is the quantity,

$$F_t = \sum_i f_i^t \tag{13}$$

Note that F_0 is the number of distinct elements (assuming $0^0 = 0$). F_1 is simply the number of elements in the stream, and thus is trivial to compute. F_2 is a measure of skewness and is the problem we will next examine, which has a solution given in [AMS99].

Suppose we have a hash function $h : U \to \{\pm 1\}$. Let $Y = \sum_i h(x_i)$ be a random variable. This variable is a sketch because we can simply add the sums of two different sets. Further, Y^2 is an unbiased estimate of F_2.

To see this, define the random variable $X_i = h(x_i)$ and note that $Y = \sum_i f_i X_i$

$$\mathbb{E}[Y^2] = \mathbb{E}[(\sum_i f_i X_i)^2] \tag{14}$$

$$= \mathbb{E}[\sum_{i,j} f_i f_j X_i X_j] \tag{15}$$

$$= \sum_{i,j} f_i f_j \mathbb{E}[X_i X_j] \tag{16}$$

$$= \sum_i f_i^2 \tag{17}$$

The last equation follows from the fact that $\mathbb{E}[X_i X_j] = 0$ if $i \neq j$.

We need a bound on the variance in order to obtain a concentration inequality.

$$\mathbb{E}[Y^4] = \mathbb{E}[(\sum_i f_i X_i)^4] = \sum_i f_i^4 + 6 \sum_{i,j} f_i^2 f_j^2 \tag{18}$$

Thus,

$$Var[Y^2] = \mathbb{E}[Y^4] - \mathbb{E}[Y^2]^2 \tag{19}$$

$$= [\sum_i f_i^2 + 6 \sum_{i,j} f_i^2 f_j^2] - [\sum_i f_i^2 + 2 \sum_{i,j} f_i^2 f_j^2] \tag{20}$$

$$= 4 \sum_{i,j} f_i^2 f_j^2 \leq 2F_2^2 = 2\mathbb{E}[Y^2]^2 \tag{21}$$

Thus, we can use Chebyshev's inequality to concentrate the mean and then use the median of means technique.

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. "The space complexity of approximating the frequency moments". In: *Journal of Computer and system sciences* 58.1 (1999), pp. 137–147.