Scribes: A. Prakash, S. Rao, U. Vazirani
Last revised

1 Streaming Algorithms

The streaming model is one way to model the problem of analyzing massive data. The model assumes that the data is presented as a stream $(x_1, x_2, \cdots, x_m)$, where the items x_i are drawn from a universe of size n. Realtime data like server logs, user clicks and search queries are modeled by streams. The available memory is much less than the size of the stream, so a streaming algorithm must process a stream in a single pass using sublinear space.

We consider the problem of estimating stream statistics using $O(\log^c n)$ space. The number of occurrences of element i in the stream is denoted by m_i. The frequency moments $F_k = \sum_i m_i^k$ are natural statistics for streams.

The moment F_0 counts the number of distinct items, an algorithm that estimates F_0 can be used to find number of unique visitors to a website, by processing the stream of ip addresses. The moment F_1 is trivial as it is the length of the stream while computing F_2 is more involved. The streaming algorithms for estimating F_0 and F_2 rely on pairwise independent hash functions, which we introduce next.

1.1 Deterministic algorithm

The following algorithm estimates item frequencies f_j within an additive error of n/k using with $O(k(\log n + \log m))$ memory,

1. Maintain set S of k counters, initialize to 0. For each element x_i in stream:
2. If $x_i \in S$ increment the counter for x_i.
3. If $x_i \notin S$ add x_i to S if space is available, else decrement all counters in S.

An item in S whose count falls to 0 can be removed, the space requirement for storing k counters is $k \log n$ and one needs to store the item which is $O(k \log m)$ as there are m items in the universe, and the update time per item is $O(k)$. The algorithm estimates the count of an item as the value of its counter or zero if it has no counter.

Claim 1

The frequency estimate n_j produced by the algorithm satisfies $f_j - n/k \leq n_j \leq f_j$.

Proof: Clearly, n_j is less than the true frequency f_j. Differences between f_j and the value of the estimate are caused by one of the two scenarios: (i) The item $j \notin S$, each counter in S gets decremented, this is the case when x_j occurs in the stream but the counter for j is not incremented. (ii) The counter for j gets decremented due to an element j' that is not contained in S.

Both scenarios result in k counters getting decremented hence they can occur at most n/k times, showing that $n_j \geq f_j - n/k$. □

1.2 Counting distinct items

Exactly counting the number of distinct elements in a stream requires $O(n)$ space, we will present a randomized algorithm that estimates the number of distinct elements to a multiplicative factor of $(1 \pm \epsilon)$ with high probability using poly$(\log n, \frac{1}{\epsilon})$ space. The probabilities are over the internal randomness used by the algorithm, the input stream is deterministic and fixed in advance.

1.2.1 Exact counting requires $O(n)$ space

Suppose A is an algorithm that counts the number of distinct elements in a stream S with elements drawn from $[n]$. After executing A on the input stream S it acts as a membership tester for S. On input $x \in [n]$ the count of distinct items increases by 1 if $x \notin S$ and stays the same if $x \in S$. The internal state of A must contain enough information to distinguish between the 2^n possible subsets of $[n]$ that could have occurred in S. The algorithm requires $O(n)$ bits of storage to distinguish between 2^n possibilities.

1.2.2 A toy problem

Consider the following simpler version of approximate counting: The output should be 'yes' if the number of distinct items N is more than $2k$, 'no' if N is less than k and we do not care about the output if $k < N < 2k$.

1. Choose a uniformly random hash function $h : [n] \rightarrow [B]$, where the number of buckets $B = O(k)$.
2. Output 'yes' if there is some $x_i \in S$ such that $h(x_i) = 0$ else output 'no'.

The value $h(x)$ is uniformly distributed on $[B]$, so for all $x \in U$ we have $\Pr_{h \in \mathcal{H}}[h(x) = 0] = 1/B$. If there are at most k distinct items in the stream, the probability that none of the N items hash to 0 is,

$$\Pr[A(x) = No \mid N \leq k] = \left(1 - \frac{1}{B}\right)^N \geq \left(1 - \frac{1}{B}\right)^k$$

If the number of elements is greater than $2k$ then the probability that the algorithm outputs no is,

$$\Pr[A(x) = No \mid N > 2k] = \left(1 - \frac{1}{B}\right)^N \leq \left(1 - \frac{1}{B}\right)^{2k}$$

The gap between the probability of the output being 'no' for the two cases is a constant for $B = O(k)$.

However, specifying a random hash function requires $O(n \log B)$ bits of storage, the truth table must be stored to evaluate the hash function. The memory requirement can be reduced by choosing h from a hash function family $\mathcal{H}$ of small size having good independence properties.

2-wise independent hash functions: The property required from $\mathcal{H}$ is 2-wise independence, informally a hash function family is 2 wise independent if the hash value $h(x)$ provides no information about $h(y)$.

CLAIM 2
The family $\mathcal{H} : [n] \to \{0, \dots, p-1\}$ consisting of functions $h_{a,b}(x) = ax + b \mod p$ where p is a prime number greater than n and $a, b \in \mathbb{Z}_p$ is 2-wise independent,

$$\Pr_{a,b}[h(x) = c \wedge h(y) = d] = \frac{1}{p^2} \qquad \forall x \neq y$$

PROOF: If $h(x) = c$ and $h(y) = d$ then the following linear equations are satisfied over $\mathbb{Z}_p$,

$$ax + b = c \qquad\qquad ay + b = d$$

The linear system has a unique solution (a, b) as the determinant $(x - y) \neq 0$ for distinct x, y. The claim follows as $|H| = p^2$ and there is a unique function such that $h(x) = c$ and $h(y) = d$.
□

This construction of 2 wise independent hash function families generalizes to k wise independent families by choosing degree k polynomials. For the streaming algorithm we require a 2-wise independent hash function family $\mathcal{H} : [n] \to [B]$ where B is not a prime number, the family $h_{a,b} = (ax+b \mod p) \mod B$ for a prime larger than p is approximately 2 wise independent.

1.3 Analysis

We analyze the algorithm using a random hash function from a pairwise independent family $\mathcal{H} : [n] \to [4k]$. From claim **??**, it follows that $\Pr_{a,b}[h(x) = 0] = 1/B$ for all $x \in [U]$. If there are k elements in the stream the probability of some element being hashed to 0 can be bounded using the union bound $\Pr[\cup A_i] \leq \sum \Pr[A_i]$,

$$\Pr[A(x) = Yes \mid N < k] \leq \frac{k}{B} = \frac{1}{4} \tag{1}$$

The inclusion exclusion principle is used to show that the probability of the output being yes is large if there are more than $2k$ elements in the stream. Truncating the inclusion exclusion formula to the first two terms yields $\Pr[\cup A_i] \geq \sum \Pr[A_i] - \sum \Pr[A_i \cap A_j]$. Using pairwise independence,

$$\Pr[A(x) = Yes \mid N \geq 2k] \geq \frac{2k}{B} - \frac{2k.(2k-1)}{B^2} \geq \frac{2k}{B}(1 - \frac{k}{B}) = \frac{3}{8} \tag{2}$$

The yes and no cases are separated by a gap of 1/8, the memory used by the algorithm is $O(\log n)$ as numbers a, b need to be stored. Using a combination of standard tricks, the quality of approximation can be improved to $1 \pm \epsilon$.

1.4 A $1 \pm \epsilon$ approximation:

The probability of obtaining a correct answer is boosted to $1 - \delta$ by running the algorithm with several independent hash functions using the following simplified version of Chernoff bounds,

Claim 3
If a coin with bias b is flipped $k = O(\frac{\log(1/\delta)}{\epsilon^2})$ times, with probability $1-\delta$ the number of heads $\widehat{b}$ satisfies $bk(1-\epsilon) \leq \widehat{b} \leq bk(1+\epsilon)$.

The algorithm is run for $O(\log 1/\delta)$ independent iterations and the output is 'yes' if the fraction of yes answers is more than 5/16. Applying the claim for the yes and no cases, it follows that the correct answer is obtained with probability at least $1-\delta$.

The number of distinct items N can be approximated to a factor of 2 using the binary search trick. The algorithm is run simultaneously for the $\log n$ intervals $[2^k, 2^{k+1}]$ for $k \in [\log n]$. If $N \in [2^k, 2^{k+1}]$ then with high probability the first $k-1$ runs answer 'yes', the answer for the k-th run is indeterminate and the last $\log n - k - 1$ runs answer 'no'. The first no in the sequence of answers occurs either for $[2^k, 2^{k+1}]$ or $[2^{k+1}, 2^{k+2}]$, the left end point of the interval where the transition occurs satisfies $\frac{N}{2} \leq L \leq 2N$.

The third trick is to replace 2 by $1+\epsilon$ in equations (1), (2) and change parameters appropriately in the boosting part to approximate the number of distinct items in the stream up to a factor of $1 \pm \epsilon$.

The space requirement of the algorithm is $O(\log n . \log_{1+\epsilon} n . \frac{\log(1/\delta)}{\epsilon^2})$, the $\log n$ is the amount of memory required to store a single hash function, the $\log_{1+\epsilon} n$ is the number of intervals considered and $\frac{\log(1/\delta)}{\epsilon^2}$ is the number of independent hash functions used for each interval.

2 Estimating F_2

The hash function h is chosen from a 4-wise independent family $\mathcal{H} : [n] \to \pm 1$. The algorithm outputs $Z^2 = (\sum_i h(x_i))^2$ as an estimate for μ, the memory requirement is $O(\log n)$. The analysis will show that $E[Z^2] = F_2$ and that the variance is small. Denoting the hash value $h(j)$ by Y_j we have,

$$Z = \sum_{i \in [m]} h(x_i) = \sum_{j \in S} Y_j m_j$$

The expectation of Z^2 can be computed by squaring and using the 2 wise independence of the hash function to cancel out the cross terms,

$$E[Z^2] = \sum_j E[Y_j^2] m_j^2 + \sum_{i,j} E[Y_i] E[Y_j] m_i m_j = \sum_i m_i^2 = F_2$$

A variance calculation is required to ensure that we obtain the correct answer with sufficiently high probability. Recall that the variance of a random variable X is equal to $E[X^2] - E[X]^2$, the variance calculation requires computing the fourth moment of Z,

$$E[Z^4] = \sum_i E[Y_i^4 m_i^4] + 6 \sum_{i,j} E[Y_i^2 Y_j^2 m_i^2 m_j^2] = \sum_i m_i^4 + 6 \sum_{i,j} m_i^2 m_j^2$$

The variance of Z^2 can now be computed,

$$Var(Z^2) = E[Z^4] - E[Z^2]^2 = 4 \sum m_i^2 m_j^2 < 2F_2^2$$

The Chebyshev inequality is useful for bounding the deviation of a random variable from its mean,

$$\Pr[|X-\mu| \geq \epsilon F_2] \leq \frac{Var(X)}{\epsilon^2 F_2^2}$$

The variance is too large for Chebyshev's inequality to be useful. The variance can be reduced by running the procedure over $k = 2/\delta\epsilon^2$ independent iterations, with the output being $Z = \frac{1}{k}\sum_{i\in[k]} Z_i^2$.

The expectation $E[Z] = \mu$ by linearity and the the variance can be calculated using relations $Var[cX] = c^2 Var[X]$ and $Var(X+Y) = Var(X) + Var(Y)$ for independent random variables X and Y.

$$Var[Z] = \sum_{i\in[k]} Var\left[\frac{Z_i^2}{k}\right] \leq \frac{2F_2^2}{k}$$

Applying the Chebychev inequality for $Z = \frac{1}{k}\sum_{i\in[k]} Z_i^2$ with $k = \frac{2}{\delta\epsilon^2}$yields $\Pr[|Z-\mu| \geq \epsilon F_2] \leq \delta$. The output of the algorithm Z is therefore a $(1\pm\epsilon)$ approximation for μ with probability at least $1-\delta$. The memory requirement for the algorithm is $O(\log n/\epsilon^2)$.

Optional Material.

2.1 Count min sketch

The turnstile model allows both additions and deletions of items in the stream. The stream consists of pairs (i, c_i), where the $i \in [m]$ is an item and c_i is the number of items to be added or deleted. The count of an item can not be negative at any stage, the frequency f_j of item j is $f_j = \sum c_j$.

The following algorithm estimates frequencies of all items up to an additive error of $\epsilon|f|_1$ with probability $1-\delta$, the ℓ_1 norm $|f|_1$ is the number of items present in the data set. The two parameters k and t in the algorithm are chosen to be $(\frac{2}{\epsilon}, \log(1/\delta))$.

1. Maintain t arrays $A[i]$ each having k counters, hash function $h_i : U \rightarrow [k]$ drawn from a 2-wise independent family $\mathcal{H}$ is associated to array $A[i]$.

2. For element (j, c_j) in the stream, update counters as follows:

$$A[i, h_i(j)] \leftarrow A[i, h_i(j)] + c_j \qquad\qquad \forall i \in [t]$$

3. The frequency estimate for item j is $\min_{i\in[t]} A[i, h(j)]$.

The output estimate is always more than the true value of f_j as the count of all the items in the stream is non negative.

2.1.1 Analysis

To bound the error in the estimate for f_j we need to analyze the excess X where $A[1, h_1(j)] = f_j + X$. The excess X can be expressed as a sum of random variables $X = \sum_i Y_i$ where the indicator random variable $Y_i = f_i$ if $h_1(j) = h_1(i)$ and 0 otherwise. As $h_1 \in \mathcal{H}$ is chosen uniformly at random from a 2-wise independent hash function family, $E[Y_i] = f_i/k$.

$$E[X] = \frac{|f|_1}{k} = \frac{\epsilon |f|_1}{2}$$

Applying Markov's inequality, we have

$$Pr[X > \epsilon |f|_1] \leq \frac{1}{2}$$

The probability that all the excesses at $A[i, h_i(x_j]$ are greater than $\epsilon|f|_1$ is at most $1/2^t \leq \delta$ as t was chosen to be $\log(1/\delta)$. The algorithm estimates the frequency of item x_j up to an additive error $\epsilon|f|_1$ with probability $1 - \delta$.

The memory required for the algorithm is the sum of the space for the array and the hash functions, $O(kt \log n + t \log m) = O(\frac{1}{\epsilon} \log(1/\delta) \log n)$. The update time per item in the stream is $O(\log \frac{1}{\delta})$.

2.2 Count Sketch

We present another sketch algorithm with error in terms of the ℓ_2 norm $|f|_2 = \sqrt{\sum_j f_j^2}$. The relation between the ℓ_1 and ℓ_2 norms is $\frac{1}{\sqrt{n}}|f|_1 \leq |f|_2 \leq |f|_1$, the ℓ_2 norm is less than the ℓ_1 norm so the guarantee for this algorithm is better than that for the previous one.

1. Maintain t arrays $A[i]$ each having k counters, hash functions $g_i : U \rightarrow \{-1, 1\}$ and $h_i : U \rightarrow [k]$ drawn uniformly at random from a 2-wise independent family are associated to array $A[i]$.

2. For element (j, c_j) in the stream, update counters as follows:

$$A[i, h_i(j)] \leftarrow A[i, h_i(j)] + g_i(j)c_j \qquad \forall i \in [t]$$

3. The frequency estimate for item j is the median over the t arrays of $g_i(x_j)A[i, h(j)]$.

2.2.1 Analysis

Again, the entry $A[1, h_1(j)] = g_1(j)f_j + X$, we examine the contribution X from the other items by writing $X = \sum_i Y_i$ where the indicator variable Y_i is $\pm f_i$ if $h_1(i) = h_1(j)$ and 0 otherwise. Note that $E[Y_j] = 0$, so the expected value of $g_1(j)A[1, h(j)]$ is f_j.

The random variables Y_i are pairwise independent as h_1 is a 2-wise independent hash function, so the variance of X can be expressed as,

$$\text{Var}(X) = \sum_{i \in [m]} \text{Var}(Y_i) = \sum_{i \in [m]} \frac{f_i^2}{k} = \frac{|f|_2^2}{k}$$

We will use Chebyshev's inequality to bound the deviation of X from its expected value,

$$Pr[|X - \mu| > \Delta] \leq \frac{Var(X)}{\Delta^2}$$

The mean $\mu = 0$ and the variance is $\frac{|f|_2^2}{k}$, choosing $\Delta = \epsilon|f|_2$ and $k = 4/\epsilon^2$ we have,

$$Pr[|X - \mu| > \epsilon|f|_2] \leq \frac{1}{\epsilon^2 k} \leq \frac{1}{4}$$

For $t = \theta(\log(1/\delta))$, the probability that the median value deviates from μ by more than $\epsilon|f|_2$ is less than δ by a Chernoff bound. That is, the probability that there are fewer than $t/2$ success in a series of t tosses of a coin with success probability $3/4$ is smaller than δ for $t = O(\log(1/\delta))$.

Arguing as in the count min sketch the space required is $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$ and the update time per item is $O(\log \frac{1}{\delta})$.

2.3 Remarks

The count sketch approximates f_j within $\epsilon|f|_2$ but requires $\widetilde{O}(\frac{1}{\epsilon^2})$ space, while the count min sketch approximates f_j within $\epsilon|f|_1$ and requires $\widetilde{O}(\frac{1}{\epsilon})$ space. The approximation provided by the sketch algorithms is meaningful only for items that occur with high frequency.