Estimating the number of distinct items in a stream

Scribe: Almas Abdibayev

1 Algorithm 1

Given:

1. $L = \lceil \log_2 m \rceil$
2. $Z[0 : L]$ - array of counters
3. $h : [m] \to [n]$ hash function picked uniformly at random from pairwise independent family

When item $a \in \{1, 2, ..., m\}$ arrives

- Evaluate pos_a := largest j s.t. 2^j divides $h(a)$ i.e. number of trailing 0's in binary representation of $h(a)$
- $Z[pos_a] = Z[pos_a] + 1$
- Find largest k s.t. $Z[k] > 0$
- After stream ends: output $\hat{d} = 2^k$

Theorem 1. with probability $\geq \frac{5}{8}$, $\frac{d}{16} \leq \hat{d} \leq 16d$ where d is the actual number of distinct items. We will devote next few pages to proving above theorem.

2 Proof of Theorem 1

2.1 Some preliminaries

Definitions:

1. $X_{a,j}$ =

$$\begin{cases} 1 & \text{if } 2^j \text{ is the largest power of 2 which divides } h(a) \\ 0 & otherwise \end{cases}$$

   Observe that $X_{a,j}$ is a random variable, since the choice of h is random.

2. $y_j = \sum_{distinct\ a} X_{a,j}$, where j is the index of the counter. This variable basically tells us how many distinct entries in the stream will increment the value of the counter j

Observations:

1. $P_h[X_{a,j} = 1] = \frac{1}{2^{j+1}}$ (you can think of $h(a)$ as a function that assigns a random string of 1s and 0s of length j to a. Then each digit within it has an equally likely, independent $\frac{1}{2}$ chance of appearing. Note that this breaks down when N (maximum possible n h can hash to) is small). This observation tells that probability of incrementing the counter j goes down exponentially with increase in j.

2. $E[y_j] = \frac{d}{2^{j+1}}$. This also implies that $2^{j+1}y_j$ is an unbiased estimator of d for all j.

3. In general $y_j \leq Z[j]$, this is true since two copies of a can map to the same bucket and thus overincrement the counter.

4. $y_j = 0 \iff Z[j] = 0$
   ($\implies$) none of y_j of distinct copies hash to position j
   ($\impliedby$) $Z[j] = 0$ means inequality $y_j \leq 0$

5. $$Var[y_j] \leq E[y_j]$$

   Proof.

   (a) variance is at most the expectation for indicator variables. E.g. $Var[X_{a,j}] \leq E[X_{a,j}]$

   (b) $Var[y_j] = \sum_a Var(X_{a,j}) \leq_{\text{(by above step)}} \sum_a E[X_{a,j}]$

   $Var[y_j] \leq_{\text{by linearity of an expectation}} E[\sum_a X_{a,j}] = E[y_j]$

   This inequality is important since for an unbiased estimator X we want $\frac{Var[X]}{E[X]}$ to be small. Knowing that $Var[y_j]$ is bounded by $E[y_j]$ we can conclude that variance of estimator y_j is at most $\frac{1}{E[y_j]}$. This means that if the mean of y_j is large we can obtain small variance for our estimator.

2.2 Onto actual proof

Let l be s.t. $2^l < d \leq 2^{l+1}$. Think of l as the number of bits required to describe d.
We know that for all j, $E[y_j] = \frac{d}{2^{j+1}}$, $Var[y_j] \leq E[y_j]$

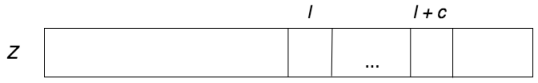

Figure 1. Array of counters Z, cell at index l and cell at index $l + c$. This figure is useful for visualization of Fact 1.

2.2.1 Fact 1

In plain English the first half of the proof says that if we go too much to the right of the array index l we do no expect to see 1s.

Define event "Bad" := $\exists\, j \geq l + c$ s.t. $Z[j] > 0$

Assert that for all $c \geq 1$,

$$P[\text{“}Bad\text{”}] \leq \delta$$

where δ is a small number is to be determined below. As the name implies this event is bad news for us, so ideally we'd want to upperbound probability of it happening.

Derivation of δ:

$$P[\text{“}Bad\text{”}] \leq_{\text{union bound}} \sum_{j \geq l+c} P[Z[j] > 0]$$

We cannot really deal with $Z[j]$s, so we'd like to instead use y_js. Recall that $y_j = 0 \iff Z[j] = 0$. This also implies that $Z[j] > 0$ implies that $y_j > 0$. If $y_j > 0$ then y_j is at least one. -

$$P[\text{“}Bad\text{”}] = \sum_{j \geq l+c} P[y_j \geq 1]$$

Remark: *Observe that as c grows we expect that $P[y_i \geq 1]$ is smaller and smaller*

$$P[\text{“}Bad\text{”}] \leq_{\text{(using Markov property)}} \sum_{j \geq l+c} \frac{E[y_j]}{1}$$

$$P[\text{“}Bad\text{”}] \leq \sum_{j \geq l+c} \frac{d}{2^{j+1}} \text{(by definition)}$$

$$P[\text{“}Bad\text{”}] \leq \sum_{j \geq l+c} \frac{2^{l+1}}{2^{j+1}} = \frac{1}{2^c} \sum_{j \geq l+c} \frac{1}{2^{j-(l+c)}} = \frac{1}{2^c}(1 + \frac{1}{2} + ...)$$

$$P[\text{“}Bad\text{”}] \leq \frac{1}{2^{c-1}}$$

2.2.2 Applying Fact 1 to a particular value of c

$$P[\ \exists\, j \geq l+4 \ \text{ s.t. } \ Z[j] > 0\ ] \leq \frac{1}{8}$$

Above implies that with probability $\geq \frac{7}{8}$, $k < l + 4$ (recall that k is the value of index that Algorithm #1 outputs)

$$d = 2^k \leq 2^{l+4} \leq 16 \cdot 2^l \leq 16d$$

2.2.3 Fact 2

In plain English the second half of the proof says that if we go too much to the left of the index l we do expect to see 1s. Conversely, the probability that we won't see 1s to the left of l is small.

$$P[y_{l-c} = 0] \leq P\Big[\mid y_{l-c} - E[y_{l-c}] \mid \geq E[y_{l-c}]\Big] \leq_{\text{using Chebyshev's inequality}} \frac{Var(y_{l-c})}{(E[y_{l-c}])^2}$$

$$P[y_{l-c} = 0] \leq_{\text{using } obs.(5)} \frac{E[y_{l-c}]}{(E[y_{l-c}])^2}$$

$$P[y_{l-c} = 0] \leq \frac{1}{E[y_{l-c}]} = \frac{2^{l-c+1}}{d} <_{(since\ d>2^l)} \frac{2^{l-c+1}}{2^l} = \frac{1}{2^{c-1}}$$

Thus,

$$P[y_{l-c} = 0] < \frac{1}{2^{c-1}}$$

2.2.4 Using Fact 1 and 2 to finish the proof of Theorem 1

Recall that by observation (4) $y_j = 0 \iff Z[j] = 0$, which implies that $P[Z[l-c] = 0] = P[y_{l-c} = 0]$

Then $P[Z[l-3] = 0] \leq \frac{1}{4} \implies$ with probability $\frac{3}{4}$, $k \geq l - 3 \implies \hat{d} = 2^k > 2^{l-3} \geq \frac{2^{l+1}}{2^4} > \frac{d}{16}$

□

Theorem 2. with probability $\geq \frac{5}{8}$, $\frac{d}{16} \leq \hat{d} \leq 16d$, where d is the actual number of distinct items

Remark: *Space taken by this algorithm equals $O(\log^2 n)$, since there are $\log n$ counters and each takes up $\log n$ bits. This can be reduced by only giving 1 bit for each counter.*

3 Algorithm 2: better way to bound $\hat{d}$

The key idea is to replace array of counters by array of sets (e.g. linked list) of bounded by constant size (to limit the memory consumption). The main observation here is that up until the threshold value, i.e. if $Z[j] < B$ then $Z[j] = y_j$.

1. $L = \lceil \log_2 m \rceil$
2. $Z[0 : L]$ - array of sets
3. $h : [m] \to [n]$ hash function picked uniformly at random from pairwise independent family
4. B - a constant bounding number of items in each set

When item $a \in \{1, 2, ..., m\}$ arrives

- Evaluate pos_a := largest j s.t. 2^j divides $h(a)$ i.e. number of trailing 0's in binary representation of $h(a)$
- $Z[pos_a] = Z[pos_a].append(a)$ if $size(Z[pos_a]) < B$
- Find smallest k s.t. $|Z[k]| < B$
- After stream ends: output $\hat{d} = 2^{k+1} \cdot |Z[k]|$.

The following remark is what our second algorithm exploits.

Remark: *Suppose Z is a random variable s.t. $Var(Z) \leq E[Z]$*

$$P\left[|z_j - E[z_j]| \geq \epsilon E[z_j]\right] \leq \frac{Var z_j}{\epsilon^2 (E[z_j]^2)} \leq \frac{1}{\epsilon^2} \frac{1}{E[z_j]}$$

$$\therefore if\, E[z_j] > \frac{10}{\epsilon^2}, this\ prob \leq \frac{1}{10}$$

The takeaway from this is that large mean and bounded variance imply good "concentration" of points. This in turn implies that the random variable is fairly close to the mean (within some approximation factor).

Question 1. Is $\hat{d}$ an unbiased estimator?

No, since k has to be fixed. In our case k is a random variable (due to the fact we are hashing). $\hat{d}$ is an underestimator (due to the bound B).

3.1 Theorem for algorithm 2

Theorem 3. with prob $\geq \frac{2}{3}$, $\frac{d}{1+\epsilon} \leq \hat{d} \leq (1+\epsilon)d$

Redefine $BAD := (\hat{d} - d) > \epsilon d$

We want to upperbound $P[BAD] = P[2^{k+1} \cdot y_k \notin (1 \pm \epsilon)d]$

Remark: *For any fixed j, $E[2^{j+1}y_j] = d$ i.e. $E[y_j] = \frac{d}{2^{j+1}}$ and $Var[y_j] \leq E[y_j]$.*
$\therefore P\left[|y_j - \frac{d}{2^{j+1}}| \geq \epsilon \frac{d}{2^{j+1}}\right] \leq \frac{1}{\epsilon^2}\frac{1}{E[y_j]} = \frac{1}{\epsilon^2}\frac{2^{j+1}}{d}$

Let l be s.t. $\frac{10}{\epsilon^2} \leq \frac{d}{2^{l+1}}_{(\text{equivalent to} E[y_l])} \leq \frac{20}{\epsilon^2}$. Why is this important? For this particular l the probability that $E[y_l]$ will be more than half the bucket size is small. This implies that bucket will fit it all of the items for this particular l with high probability.

Let's go back to $P[BAD]$.

$$P[BAD] = P[2^{k+1} \cdot y_k \notin (1 \pm \epsilon)d] = P\left[|y_k - \frac{d}{2^{k+1}}| > \epsilon \frac{d}{2^{k+1}}\right]$$

$$P[BAD] =_{\text{decoupling trick}} = \sum_{j=0}^{L} P\left[(k = j) \text{ and } |y_j - \frac{d}{2^{j+1}}| > \epsilon \frac{d}{2^{j+1}}\right]$$

Remark: *We cannot fix k (we cannot apply the previous remark) but we can sum over all possibilities to "fix" it. This only works for "small" js. For large js this is irrelevant since the probability of those buckets to have values decays exponentially.*

$$P[BAD] = \sum_{j=0}^{l} P\left[(k = j) \text{ and } |y_j - \frac{d}{2^{j+1}}| > \epsilon \frac{d}{2^{j+1}}\right] + \sum_{j=l+1}^{L} P\left[(k = j) \text{ and } |y_j - \frac{d}{2^{j+1}}| > \epsilon \frac{d}{2^{j+1}}\right]$$

$$P[BAD] \leq \sum_{j=0}^{l} P\left[|y_j - \frac{d}{2^{j+1}}| > \epsilon \frac{d}{2^{j+1}}\right] + \sum_{j=l+1}^{L} P\left[(k = j)\right]$$

$$P[BAD] \leq \sum_{j=0}^{l} \frac{1}{\epsilon^2}\frac{2^{j+1}}{d} + P\left[(k > l)\right]$$

$$P[BAD] \leq \sum_{j=0}^{l} \frac{1}{\epsilon^2}\frac{2^{j+1}}{d} + P\left[(k > l)\right]$$

$$P[BAD] \leq \frac{1}{\epsilon^2 d} \sum_{j=0}^{l} 2^{j+1} + P\Big[(k \geq B)\Big] (if \;\; k > l \implies |Z[l]| > B \implies |y_l| > B, \text{but if not } \; y_l = Z[l] \leq B)$$

$$P[BAD] \leq \frac{2^{l+2}}{\epsilon^2 d}$$

$$P[BAD] \leq \frac{1}{10} \frac{2^{l+2}}{2^{l+1}}$$

$$P[BAD] \leq \frac{1}{5}$$

Remark: *you need to keep counting until B =* $\frac{100}{\epsilon^2}$

Remark: *Space taken by this algorithm is* $O(\frac{\log^2 n}{\epsilon^2})$ *bits*