Estimating Distinct Elements in a Data Stream

Among practical algorithms, 3 are state of the art

- HyperLogLog [Flajolet et al., 2007]
- K-Minimum Values (KMV)
- Adaptive Sampling

KMV + Adaptive Sampling

I will describe KMV as having space usage $O(C \log n + \frac{1}{\epsilon^2} \cdot (\log C + \log \log n))$

for $\delta = \frac{1}{5\delta}$ (say).

In theory, [KMV] gave an $O(C \log n + \frac{1}{\epsilon^2})$ space algorithm with O(1) update time.

**FOR KMV:** Let $h: [n] \rightarrow [0, 1]$ be a random function.

- Track the K-smallest hash values observed in the stream. Let $mu$ denote the K-th smallest hash value seen.
- Output the estimate $\frac{1}{mu}$

**Adaptive Sampling:** Let $h: [n] \rightarrow [0, 1]$ be a random hash function.

- Initialize $i = 0$.
- While processing stream, store all hash values $< 2^{-i}$.
- If more than K hash values are stored, set $i \leftarrow i + 1$.
- If $i$ is count hash values stored at end of stream, output $2^i - 2^i$.

**Facts about KMV and Adaptive Sampling:**

- KMV has a natural min-heap based implementation; however, KMV requires $O(C \log n)$ time per stream update, and doubles its space usage compared to just keeping the K hash values in a hash table.

Adaptive Sampling is faster (O(1)-amortized update time) and doesn't have the space cost of a min-heap. But the error of Adaptive sampling oscillates due to periodic purges.
· Both KMV and Adaptive Sampling are unbiased, we can prove for KMV

\[
\text{Var}[\text{KMV}] \leq \frac{F_0^2}{K-1}
\]

\[
\text{Var}[\text{Adaptive Sampling}] \approx \frac{1.44 F_0^2}{K-1}
\]

We will not prove either of these facts, however we note that

by Chebyshev, the variance bounds mean setting \( K = O(\frac{1}{\varepsilon^2}) \) is enough for an \( (\varepsilon, \delta) \)-multiplicative approximation.

\[
\Pr[|\text{KMV} - F_0| > \varepsilon F_0] \leq \frac{1}{4}.
\]

\[
\Omega(\text{KMV, \; with \; } k = \frac{1}{2}+\text{counters}) \text{ is about } \frac{\varepsilon F_0}{2} \approx \frac{\varepsilon}{\sqrt{4\varepsilon^2}}
\]

* What is the space usage of KMV and adaptive sampling?

* It turns out it is enough for the hash function \( h \) to be from a

  pairwise independent hash family mapping \([n]\) to a set

  of size \( O\left(\frac{1}{\varepsilon^4} \log^{1+\delta} n\right) \). Hence, \( h \) requires \( O(\log n + \log(\frac{1}{\varepsilon^4})) \)

  bits to represent, and each of the \( O(\frac{1}{\varepsilon^4}) \) hash values take

  \( O(\log \frac{1}{\varepsilon^4} + \log \log n) \) bits to represent, for a final space

  bound of \( O(\log n + \frac{1}{\varepsilon^4} \cdot \left(\log \frac{1}{\varepsilon^4} + \log \log n\right)) \).

* In practice, it can be very useful to store not just the hash

  values but also the corresponding identifiers. If you do this, the space

  usage is \( \Theta\left(\frac{\log n}{\varepsilon^4}\right) \). This lets you, e.g., obtain estimates

  for ad hoc subsets of users, where the subset of interest

  is only determined at runtime.
Hyper-loglog:

- Let \( h \) be a random function mapping \([n]\) to \([k]\).
- Let \( g \) be a random function mapping \([n]\) to \([k]\).
- For each \( j \in [k] \), track \( y_j = \max z \text{ s.t. } h(a_i) = j \).
- Define \( z_j = g(a_i) \).

Let \( Z \) be the harmonic mean of \( 2^z \).

Output: \( \alpha \cdot k \cdot Z \). \( \alpha \) is a constant meant to correct a small bias in \( k \).

Intuition: \( 2^z \) should be about \( \frac{F_0}{k} \), so \( k \cdot Z \) should be close to \( F_0 \).

To get an \( (\epsilon, \delta) \)-multiplicative approximation, can take \( k \approx \frac{1}{\epsilon^2} \), so total space usage is \( O(\log n + \log(\log n)/\epsilon^2) \).

Proof that \( k \cdot Z \) understated. Fix any item \( i \),

Fix any \( j \in [k] \) appearing one or more times in the stream.

Define \( V_j := \frac{1}{m_k} \) if \( h(j) \) is one of the \( k \) smallest hash values observed.

Let \( H_j \) denote the \( k \)-th smallest hash value of all other items in the stream other than \( j \).

Let \( m_k \) denote the \( k \)-th smallest hash value in \( H_j \), and \( m_{k-1} \) the \( k-1 \)-st smallest.

Then \( \mathbf{E}[V_j | H_j] = \Pr[h(j) < m_{k-1}] \cdot \frac{1}{m_{k-1}} \) 
\+ \Pr[m_{k-1} < h(j) < m_k] \cdot 0 
= \frac{m_{k-1}}{m_k} - \frac{1}{m_{k-1}}.

Note the KMV estimate is \( \sum \frac{1}{z_j} \), so \( \mathbf{E}[\text{KMV estimate}] = \sum \mathbf{E}[V_j] = \sum \frac{1}{z_j} = F_0 \).
Analysis of Adaptive Sampling with $K = \frac{c_N}{C}$ (to be specified later)

Let $X_{r,j} = 1$ if $h(j) \leq 2^{-r}$

and let $Y_r = \sum_j X_{r,j}$. Let $i_{\text{max}}$ denote the value of $i$ at the end of set of $h(j)$ values less than $2^{-i_{\text{max}}}$.

So the returned estimate $\hat{F}_0 = \frac{1}{i_{\text{max}}}, \hat{F}_{i_{\text{max}}}$

Exactly as last lecture, we obtain

Fact 1: $E[Y_r] = \frac{F_0}{2^r}$, $\text{Var}[Y_r] \leq \frac{F_0}{2^r}$.

Note: if $i_{\text{max}} = 0$ then the algorithm never saw more than $K$ lines, so it returns exactly $1S_1 = F_0$.

Otherwise, we need to bound the probability $|Y_{i_{\text{max}}} - \frac{F_0}{2^{i_{\text{max}}}}| \geq \frac{cF_0}{2^{i_{\text{max}}}}$, $c = \text{Call PS1 error, FAIL}$.

Let $s$ be the unique moyer s.t. $\frac{1}{2^s} \leq \frac{F_0}{d_s} \leq \frac{1}{2^{s+1}}$.

Then $\Pr[\text{FAIL}] = \sum_{r=1}^{i_{\text{max}}} \Pr \left[ \left| Y_r - \frac{F_0}{2^r} \right| \geq \frac{cF_0}{2^r}, \text{and } i_{\text{max}} = r \right]$

$= \left( \sum_{r=1}^{s+1} \Pr \left[ \left| Y_r - \frac{F_0}{2^r} \right| \geq \frac{cF_0}{2^r} \right] \frac{cF_0}{2^r} + \sum_{r=s+2}^{i_{\text{max}}} \Pr \left[ \left| Y_r - \frac{F_0}{2^r} \right| \geq \frac{cF_0}{2^r} \right] \right) + \Pr[C_{i_{\text{max}}} < k_4] + \Pr[C_{\text{max}} \geq \frac{c}{c_0}]$

$\leq \downarrow$

$\leq \downarrow$
By Chebyshev's and Fact 1, we have,
\[ \Pr \left[ \left| Y_n - \frac{F_0}{c_n} \right| \geq \frac{\varepsilon}{c_n} \right] \leq \frac{\varepsilon^2}{F_0}, \]
so the first term is at most
\[ \frac{5}{2} \frac{\varepsilon^2}{F_0} \leq \frac{5}{2} \frac{\varepsilon^2}{\varepsilon^2} = \frac{1}{2}. \]

By Markov's inequality and Fact 1, we have,
\[ \mathbb{E}[Y_{\delta-1}] = \frac{F_0}{c_0} \leq \frac{49}{c} \leq \frac{1}{18}. \]
We choose \( \varepsilon = 4 \).

In total, \( \Pr[\text{Fail}] \leq \frac{1}{12} + \frac{1}{12} = \frac{1}{6} \) so this is an \((6, \varepsilon)\)-approximation algorithm.