Algorithms for data streams

Maria Serna

Spring 2024

Frequent items

- We have a stream $x_1, \ldots, x_m$, where $x_i \in \Sigma$.
- This implicitly defines a frequency vector $f_1, \ldots, f_n$, where $n = |\Sigma|$ with $f_1 + \cdots + f_n = m$.
- Frequent items problem:
  Given k, output the set $\{j \mid f_j > m/k\}$.
- Frequency estimation problem:
  Process the stream to get a data structure that can provide an estimate $\hat{f}_i$ of f_i, for a given $i \in [n]$.

Frequency estimation: Naive approach

- Exact algorithm:

```
procedure FREQ(int n, stream s)
    int j, F[n] = 0
    while not s.end() do
        j = s.read()
        F[j]++
```

- Computes the frequency vector.
- One pass, using $O(n \log m)$ memory and $O(1)$ time per item.

Frequency Estimation: Misra-Gries algorithm

- The algorithm has an additional parameter k.
- Uses an associative array with n potential keys.
- The associative array can be implemented using a balanced binary search tree.

Frequency Estimation: Misra-Gries algorithm

procedure MISRA-GRIES(int n, stream s,int k)
  int A empty associative array
  while not $s.end()$ **do**
    $j = s.read()$
    if $j \in keys(A)$ **then**
      $A[j]{+}{+}$
    else
      if $|keys(A)| < k - 1$ **then**
        $A[j] = 1$
      else
        for $\ell \in keys(A)$ **do**
          $A[\ell]$- -
          if $A[\ell] == 0$ **then**
            remove ℓ from A
  On query a, if $a \in keys(A)$, report $\hat{f}_a = A[a]$, else report 0.

Misra-Gries algorithm: cost analysis

- Only one pass.
- Each key requires $O(\log n)$ bits and each value $O(\log m)$ bits.
- There are at most $k-1$ key/value pairs, the total space is $O(k(\log m + \log n))$.
- The time per element is $O(k)$.
- Quality of the solution?

Misra-Gries algorithm: quality analysis

- Let's see A as a vector with $A[i] = 0$ when $i \notin keys(A)$
- $A[j]$ is incremented only when j appears in s, so $\hat{f}_j \leq f_j$.
- Whenever $A[j]$ is decremented, we decrement the values of other $k - 1$ keys.
  The decrement is witnessed by k tokens including j, assuming that $A[j]$ first goes to 1 and then down to 0.
- Since the stream has m tokens there can be at most m/k such decrements. Therefore, $\hat{f}_j \geq f_j - m/k$.
- Putting all together

$$f_j - \frac{m}{k} \leq \hat{f}_j \leq f_j$$

Frequent items using Misra-Gries algorithm

- By the analysis, if one key j has $f_j > m/k$, $\hat{f}_j > 0$.
- However, there might be elements for which $\hat{f}_j > 0$ but $f_j \leq m/k$.
- Perform a second pass on the stream, counting exactly the frequencies of the values $i \in keys(A)$. And extracting only those verifying the property.
- 2 pass algorithm, using $O(k(\log m + \log n))$ space, and $O(k)$ time per element.

1. Finding frequent items
2. Counting values
3. Sketches

Counting the number of distinct elements

- Distinct elements problem: output $|\{j \mid f_j > 0\}|$.
- This is a simplification of the Frequent items problem:
- In order to solve the problem using sublinear space we need to use probabilistic algorithms/data structure and some adequate notion of approximation.

An (ϵ, δ)-approximation

- Let $\mathcal{A}(s)$ denote the output of a randomized streaming algorithm $\mathcal{A}$ on input s; note that this is a random variable.
- Let $\Phi(s)$ be the function that $\mathcal{A}$ is supposed to compute.
- $\mathcal{A}$ is a (ϵ, δ)-approximation to Φ if we have

$$Pr\left[\left|\frac{\mathcal{A}(s)}{\Phi(s)} - 1\right| > \epsilon\right] \leq \delta.$$

- $\mathcal{A}$ is a (ϵ, δ)-additive approximation to Φ if we have

$$Pr\left[|\mathcal{A}(s) - \Phi(s)| > \epsilon\right] \leq \delta.$$

- When $\delta = 0$, $\mathcal{A}$ must be deterministic.
  When $\epsilon = 0$, $\mathcal{A}$ must be an exact algorithm.

Randomized data structures

- We need hashing and in particular hash functions selected at random from a universal hash family.
- Recall that a family of functions

$$H = \{h : U \to [m]\}$$

is called a 2-universal family if, $\forall x, y \in U,\ x \neq y$,

$$\Pr_{h \in H}[h(x) = h(y)] \leq \frac{1}{m}.$$

- A hash function can be easily selected at random from a 2-universal hash family.

Values from the binary representation

- For an integer $p > 0$, let zeros(p) be the number of zeros at the end of the binary representation of p.

$$\text{zeros}(p) = \max\{i \mid 2^i \text{ divides } p\}.$$

Counting distinct elements

Algorithm: Flajolet and Martin, 1983

```
procedure COUNT-DIF(stream s)
    Choose a random hash function h : [n] → [n]
    from a universal family
    int z = 0
    while not s.end() do
        j = s.read()
        if zeros(h(j)) > z then
            z = zeros(h(j))
    Return ⌊2^(z+1/2)⌋
```

- Assuming that there are d distinct elements, the algorithm computes $\max \text{zeros}(h(j))$ as a good approximation of $\log d$.
- 1 pass, $O(\log n)$ memory and $O(1)$ time per item.

Counting the number of distinct elements: Quality

- For $j \in [n]$ and $r \geq 0$, let $X_{r,j}$ be the indicator r.v. for
  zeros$(h(j)) \geq r$.
- Since $h(j)$ is uniformly distributed over the $\log n$-bit strings,

$$E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = \frac{1}{2^r}$$

- Let $Y_r = \sum_{j|f_j>0} X_{r,j}$ and let t denote the final value of z.
- $Y_r > 0$ iff $t \geq r$, or equivalently $Y_r = 0$ iff $t \leq r - 1$.

Counting the number of distinct elements: Quality

$$E[X_{r,j}] = Pr[\text{zeros}(h(j)) \geq r] = Pr[2^r \text{ divides } h(j)] = \frac{1}{2^r}.$$

$$E[Y_r] = \sum_{j|f_j>0} E[X_{r,j}] = \frac{d}{2^r}$$

- Random variables Y_r are pairwise independent, as they come from a universal hash family.

$$Var[Y_r] = \sum_{j|f_j>0} Var[X_{r,j}] \leq \sum_{j|f_j>0} E[X_{r,j}^2] = \sum_{j|f_j>0} E[X_{r,j}] = \frac{d}{2^r}$$

Counting the number of distinct elements: Quality

- $E[Y_r] = Var[Y_r] = d/2^r$
- Using Markov's and Chebyshev's inequalities,

$$Pr[Y_r > 0] = Pr[Y_r \geq 1] \leq \frac{E[Y_r]}{1} = \frac{d}{2^r}.$$

$$Pr[Y_r = 0] = Pr[|Y_r - E[Y_r]| \geq \frac{d}{2^r}] \leq \frac{Var[Y_r]}{(d/2^r)^2} \leq \frac{2^r}{d}.$$

Counting the number of distinct elements: Quality

- $Pr[Y_r > 0] \leq \frac{d}{2^r}$ and $Pr[Y_r = 0] \leq \frac{2^r}{d}$.
- Let $\hat{d}$ be the estimate of d, $\hat{d} = 2^{t+\frac{1}{2}}$.
- Let a be the smallest integer so that $2^{a+\frac{1}{2}} \geq 3d$,

$$Pr[\hat{d} \geq 3d] = Pr[t \geq a] = Pr[Y_a = 0] \leq \frac{d}{2^a} \leq \frac{\sqrt{2}}{3}.$$

- Let b be the largest integer so that $2^{b+\frac{1}{2}} \leq 3d$,

$$Pr[\hat{d} \leq 3d] = Pr[t \leq b] = Pr[Y_{b+1} = 0] \leq \frac{2^{b+1}}{d} \leq \frac{\sqrt{2}}{3}.$$

Counting the number of distinct elements: Quality

- $Pr[\hat{d} \geq 3d] \leq \frac{\sqrt{2}}{3}$ and $Pr[\hat{d} \leq 3d] \leq \frac{\sqrt{2}}{3}$.
- Thus the algorithm provides a $(2, \frac{\sqrt{2}}{3})$-approximation.
- How to improve the quality of the approximation?
- Usual technique: run k independent copies of the algorithm and take the best information from them, in this case, the median of the k answers.

  If the median exceed $3d$ at least $k/2$ of the runs do.
- By standard Chernoff bounds, the median exceed $3d$ with probability $2^{-\Omega(k)}$ and the median is below $3d$ with probability $2^{-\Omega(k)}$.
- Choosing $k = \Theta(\log(1/\delta))$, we can make the sum to be at most δ. So we get a $(2, \delta)$-approximation. However, the used memory is now $O(log(1/\delta) \log n)$.

Sketches

- We require some property for the data structure computed while processing a stream s.
- A data structure $DS(s)$ is called a sketch when there is a space efficient combining algorithm $\textsc{comb}$ such that
$$\textsc{comb}(DS(s_1), DS(s_2)) = DS(s_1 \cdot s_2).$$
- A sketching algorithm $\textsc{sk}$ is an algorithm that computes a sketch on input s
- A sketching algorithm $\textsc{sk}$ is a linear sketch if for each s over a token univers $[n]$, $\textsc{sk}(s)$ takes values in a linear space of dimension $\ell(n)$, and $\textsc{sk}(s)$ is a linear function.
- For linear sketches the combination algorithm is simply to add the sketches in the appropriate vector space.

The count sketch

- Charikar, Chen and Farach-Colton, 2004
- We assume that the stream is formed by items in the form (j, c) where c is a non negative value ($c \geq 0$).
- This is a modification on the basic model for a stream. We get different interpretations depending on the possible values of c.
- We analyze sketches for the Frequency Estimation problem.
- For this problem, the sketches are just vectors on k components for some $k \leq n$.

The count sketch

procedure CSKETCH(int n, stream s, real ϵ)
  Choose a random hash function $h : [n] \to [k]$ from a 2-universal family
  Choose a random hash function $g : [n] \to \{-1, 1\}$ from a 2-universal family
  int j, $C[k] = 0$
  while not $s.end()$ **do**
    $(j, c) = s.read()$
    $C[h(j)] = C[h(j)] + c\ g(j)$
  On query a, report $\hat{f}_a = g(a)C[h(a)]$

Is CSKETCH a linear sketch? provided h and g are the same, yes!

Frequency Estimation: CSketch

- The algorithm has an additional parameter ϵ that controls the accuracy of the solution.
- For Frequency estimation we take always $c = 1$.
- One pass, $O(k \log n)$ memory, $O(1)$ ops per element
- Fix an arbitrary a, and consider the random variable $X = \hat{f}(a)$.
- For each $j \in [n]$, let Y_j be the indicator of $h(j) = h(a)$.
- $j \in [n]$ contributes to $C[h(a)]$ only when $h(j) = h(a)$, an the contribution is $g(j) f_j$
- Thus, $X = g(a) \sum_{j \in [n]} g(j) f_j Y_j$

Frequency Estimation: CSketch

- $X = g(a)\sum_{j\in[n]} g(j) f_j Y_j = f_a + \sum_{j\in[n]-\{a\}} g(a)g(j)f_j Y_j$ as $g(a)g(a) = 1$
- Since g and h are independent

$$E[g(j)Y(j)] = E[g(j)]E[Y_j] = 0\, E[Y_j] = 0$$

- Therefore, $E[X] = f_a$
- $\hat{f}_a$ is an unbiased estimator of f_a.
- Quality of the solution? again we need a concentration bound

CSketch: quality analysis

- We analyze the variance of $X = f_a + \sum_{j\in[n]-\{a\}} g(a)g(j)f_j Y_j$
- By the 2-universality of the families of hash functions

  For $j \in [n] - \{a\}$,

$$E[Y_j^2] = E[Y_j] = Pr[h(j) = h(a)] = \frac{1}{k}$$

  For $i \neq j$,

$$E[g(i)g(j)Y_i Y_j] = E[g(i)]E[g(j)]E[Y_i Y_j] = 0$$

- Now, we compute the variance

CSketch: quality analysis

$$
\begin{aligned}
Var[X] &= 0 + g(a)^2 Var\left[\sum_{j\in[n]-\{a\}} g(j) f_j Y_j\right] \\
&= E\left[\sum_{j\in[n]-\{a\}} f_j^2 Y_j^2 + \sum_{i,j\in[n]-\{a\} i\neq j} g(i)g(j) f_i f_j Y_i Y_j]\right] \\
&\quad - \left(\sum_{j\in[n]-\{a\}} f_j E[g(j)Y_j]\right)^2 \\
&= \sum_{j\in[n]-\{a\}} \frac{f_j^2}{k} = \frac{(\sum_{j\in[n]} f_j^2) - f_a^2}{k}
\end{aligned}
$$

CSketch: quality analysis

- Recall $||f||_2^2 = \sum_{j\in[n]-\{a\}} f_j^2$
- Using Chebyshev's inequality

$$Pr\left[|f_a - \hat{f}_a| \geq \epsilon\sqrt{||f||_2^2 - f_a^2}\right] = Pr\left[|X - E[X]| \geq \epsilon\sqrt{||f||_2^2 - f_a^2}\right]$$
$$\leq \frac{Var[X]}{\epsilon^2(||f||_2^2 - f_a^2)} = \frac{1}{k\epsilon^2} = \frac{1}{3}$$

CSketch: quality analysis

- For $j \in [n]$, define f_{-j} to be the $(n-1)$-dimensional vector eliminating the j-th component of f.

$$||f_{-a}||_2^2 = ||f||_2^2 - f_a^2$$

- We have shown that

$$Pr\left[|f_a - \hat{f}_a| \geq \epsilon||f_{-a}||_2^2\right] \leq \frac{1}{3}$$

- CSketch is an $(\epsilon, 1/3)$ approximation.
- Still not too good! What with the median trick?

The Count sketch

procedure COUNT SKETCH(int n,α, stream s, double ϵ, δ)
  Choose t independent random hash function $h_1, \ldots, h_t : [n] \to [k]$ from a 2-universal family
  Choose t random hash function $g_1 \ldots g_t : [n] \to \{-1, 1\}$ from a 2-universal family
  double $k = 3/\epsilon^2$, $t = \alpha \log(1/\delta)$
  int j, $C[t][k] = 0$
  while not $s.end()$ **do**
    $(j, c) = s.read()$
    for do $i = 1, \ldots, t$
      $C[i][h_i(j)] = C[i][h_i(j)] + cg_i(j)$
  On query a, report $\hat{f}_a = \text{median}_{1 \leq i \leq t} g_i(a) C[i][h_i(a)]$

COUNT SKETCH is a linear sketch provided h's and g's are the same

The Count sketch

- 1 pass, $O(1)$ time per item.
- Memory:
  $O(t \log n)$ space for storing the hash functions
  $O(\log m)$ per counter
  overall $O(t \log n + tk \log m)$ which is

$$O\left(\frac{1}{\epsilon^2} \log \frac{1}{\delta} (\log m + \log n)\right)$$

- Quality: We need Chernoff like bounds for medians.

Chernoff for median

Let X be a random variable with $E[X] = \mu$. For any $\epsilon, \delta > 0$, let $t = C \log \frac{1}{\delta}$ and $k = \frac{3Var(X)}{\epsilon^2 \mu^2}$, where C is some universal constant. Let X_{ij}, $i \in [t]$ and $j \in [k]$, be independent random variables with the same distribution as X. Let $Z = \frac{1}{k} \text{median}_{i \in [t]} \left(\sum_{j=1}^{k} X_{ij} \right)$

Then, $E[Z] = \mu$ and $Pr(|Z - \mu| \geq \epsilon\mu) \leq \delta$

- Thus, for the sketch

$$Pr\left[|f_a - \hat{f}_a| \geq \epsilon ||f_{-a}||_2^2\right] \leq \delta$$

- Count Sketch is an (ϵ, δ) approximation when α is a suitable big enough constant.