

Computational Frameworks: Streaming

(Part 2)

OUTLINE

- ① Introduction to the streaming model (Part 1)
- ② Warm-up: finding the majority element (Part 1)
- ③ Sampling (Part 1)
 - Uniform sampling with reservoir sampling
 - Finding (approximate) frequent items
- ④ Sketching
 - Counting distinct elements with probabilistic counting
 - Estimating individual frequencies and the second moment
- ⑤ Filtering: Bloom filters for membership problem

Sketching

Sketch: space-efficient data structure that can be used to provide (typically **probabilistic**) estimates of (statistical) characteristics of a data stream.

Frequency Moments

Consider a stream $\Sigma = x_1, x_2, \dots, x_n$, whose elements belong to a universe U .

For each $u \in U$ occurring in Σ define its (*absolute*) frequency

$$f_u = |\{j : x_j = u, 1 \leq j \leq n\}|,$$

i.e., the number of occurrences of u in Σ

Definition: Frequency Moments

For every $k \geq 0$, the k -th frequency moment F_k of Σ is

$$F_k = \sum_{u \in U} f_u^k.$$

(We assume $0^0 = 0$)

Frequency Moments

Relevant informations from frequency moments:

- $F_0 = \text{number of distinct items in } \Sigma$. It is useful, for instance, in the analysis of web-logs.
- $F_1 = |\Sigma|$: trivial to maintain with a counter.
- $1 - F_2/|\Sigma|^2 = \text{Gini index of } \Sigma$. It provides info on data skew: the closer to 0 and the higher is the skew. It is used for decision tree induction.

Σ = m elementi con n/m occorrenze

$$F_2 = \sum_{u \in U} f_u^2 = \sum_{u \in U} \left(\frac{n}{m}\right)^2 = m \frac{n^2}{m^2} = \frac{m^2}{m}$$

$$\text{Gini index} = 1 - \frac{F_2}{|\Sigma|^2} = 1 - \frac{n^2}{m} \cdot \frac{1}{n^2} = 1 - \frac{1}{m}$$

Σ = 1 elemento, n occorrenze; $|U| = m$

$$F_2 = \sum_{u \in U} f_u^2 = n^2; \quad \text{Gini index} = 1 - \frac{n^2}{n^2} = 0$$

Estimating F_0 for Σ (i.e., distinct elements)

Exact computation: use $|U|$ counters or dictionary with $|F_0|$ elements
(not suited for streaming setting).

Approximation: Probabilistic counting algorithm

([Flajolet, Martin 1983])

- Working memory $O(\log |U|)$. \Rightarrow severe upper bound set U per allocation
- F_0 estimated within a factor c with probability $\geq 1 - 2/c$
(accuracy-confidence tradeoff).
- Main idea:
 - Map each $u \in U$ to a random integer $h(u) \in [0, |U| - 1]$.
 $\Rightarrow h(u)$ is a $O(\log |U|)$ -bit binary string.
 - The more distinct elements in Σ , the more likely to have a $u \in \Sigma$ mapped to a string with many trailing 0's.

\Rightarrow funzione hash

Probabilistic counting algorithm

We need the following ingredients:

- Array C of $\lceil \log |U| \rceil + 1$ bits, all initialized to zero.
- Hash Function $h : U \rightarrow [0..|U| - 1]$. We assume:
 - For every $u \in U$, $h(u)$ has uniform distribution in $[0..|U| - 1]$
 - All $h(u)$'s are pairwise independent.
- Notation: for $i \in [0..|U| - 1]$, define

$tr(i) = \text{number of trailing zeroes in binary representation of } i$

e.g., $12 = (1100)_2 \Rightarrow tr(12) = 2$

Algorithm: For each $x_j \in \Sigma$ do $C[tr(h(x_j))] \leftarrow 1$

After processing x_n , estimate F_0 as

$$\tilde{F}_0 = 2^R,$$

where R is the largest index of C with $C[R] = 1$.

Example

$$\Sigma = A, D, A, A, C, B, F, F, B, A, E, C$$

	$h \Rightarrow$ map to 5 bit
A	10110
B	11000
C	10101
D	11001
E	10011
F	01000

$$A \rightarrow h(A) = 10110 \Rightarrow t_5(A) = 1$$

$$D \rightarrow h(D) = 11001 \Rightarrow t_5(D) = 0$$

:

$$C \rightarrow h(C) = 10101 \Rightarrow t_5(C) = 0$$

$$B \rightarrow h(B) = 11000 \Rightarrow t_5(B) = 3$$

$$F \rightarrow h(F) = 01000 \Rightarrow t_5(F) = 3$$

$\therefore E \rightarrow h(E) = 10011 \Rightarrow t_5(E) = 0$

$$C = \begin{bmatrix} 1 & 1 & 1 \\ \emptyset & \emptyset & 0 \\ 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

\Downarrow

$$R=3$$

$$\tilde{F}_0 = 2^3 = 8$$

Idea: ogni volta che ho ruoto d., ripetere alg. x volte
in parallelo, poi prendo mediana

Why does it work? (Intuition)

For simplicity, assume $|U|$ a power of 2.

For $x \in \Sigma$, what is the probability that $h(x)$ has at least j trailing zeros?

- $\text{Prob}(tr(h(x)) \geq 1) = \frac{1}{2}$
- $\text{Prob}(tr(h(x)) \geq 2) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$
-
- $\text{Prob}(tr(h(x)) \geq j) = \left(\frac{1}{2}\right)^j$

m d. distinti

$$x_j = \# d. \text{ con } \ln(x_i) \geq j$$

$$x_j = \sum_{u \in V} Y_{u,j}, \quad Y_{u,j} = \begin{cases} 1 & \text{if } \ln(u) \geq j \\ 0 & \text{else} \end{cases} \Rightarrow \mathbb{E}[Y_{u,j}] = \frac{1}{2^j}$$
$$= \sum_{u \in \text{insiemi}} Y_{u,j}$$

$$\mathbb{E}[x_j] = \mathbb{E}\left[\sum Y_{u,j}\right] = \sum \mathbb{E}[Y_{u,j}] = \sum \frac{1}{2^j} = \frac{m}{2^j}$$

Ie $2^j > m$, $\frac{m}{2^j} < 1 \Rightarrow$ non mi aspetto che posizione j sia 1

$$\mathbb{E}[x_1] = m/2^1 = m/2$$

$$\mathbb{E}[x_1] = m/2^2 = m/4$$

\vdots
 $\mathbb{E}[x_i] = m/2^i \sim 1 \Rightarrow$ si aspettiamo che $\log_2 m$ sia posizione eguale a 1
+ a delta

$$\mathbb{E}[x_{j+1}] = m/2^{j+1} < 1$$

comme aere + escavazioni

Probabilistic guarantees

The following theorem summarizes the properties of the algorithm.

Theorem

For a stream Σ of n elements, the **probabilistic counting algorithm** returns a value \tilde{F}_0 such that, for any $c > 2$:

$$\Pr(\tilde{F}_0 < F_0/c) \leq 1/c \quad \text{and} \quad \Pr(\tilde{F}_0 > cF_0) \leq 1/c,$$

hence

$$\Pr(F_0/c \leq \tilde{F}_0 \leq cF_0) \geq 1 - 2/c.$$

The algorithm requires a working memory of $O(\log |U|)$ bits, 1 pass, and $O(\log |U|)$ time per element.

Exercises

Exercise: High Probability Guarantees (median trick)

Consider a stream Σ with elements from a universe U and let F_0 be the number of distinct elements in Σ . Suppose that you run ℓ independent instances of the probabilistic counting algorithm, obtaining ℓ estimates $\tilde{F}_0^{(j)}$ for F_0 , with $1 \leq j \leq \ell$. Assume ℓ odd and let \tilde{F}_0 be the median value of the $\tilde{F}_0^{(j)}$'s. Show that with a suitable choice of $\ell = \Theta(\log |U|)$ we have:

$$\Pr(\tilde{F}_0 < F_0/16) \leq 1/|U| \quad \text{and} \quad \Pr(\tilde{F}_0 > 16F_0) \leq 1/|U|.$$

Hint: Use the previous theorem and the Chernoff bound.

Estimating individual frequencies and F_2

Consider a stream $\Sigma = x_1, x_2, \dots, x_n$, whose elements belong to U , with $|U| = M$. Recall that for each $u \in U$ its frequency in Σ is

$$f_u = |\{j : x_j = u, 1 \leq j \leq n\}|,$$

OUR OBJECTIVE: in one pass over Σ we want to compute a small sketch that enables to derive unbiased estimates of

- f_u for any given $u \in U$ (*individual frequencies*)
- $F_2 = \sum_{u \in U} (f_u)^2$ (*second moment*)

with provable space-accuracy tradeoffs

Observation: clearly, the exact computation of all f_u 's or of F_2 might require space proportional to $|\Sigma|$.

Count-min sketch

The first approach we consider is based on the **count-min sketch** invented by [Cormode, Muthukrishnan 2003].

non-i unbiased estimator

Main ingredients

- $d \times w$ array C of counters ($O(\log n)$ bits each)
- d hash functions: h_0, h_1, \dots, h_{d-1} , with

$$h_j : U \rightarrow \{0, 1, \dots, w-1\},$$

for every j .



Note that d and w are design parameters that regulate the space/time-accuracy tradeoff.

Count-min sketch: algorithm

Initialization: $C[j, k] = 0$, for every $0 \leq j < d$ and $0 \leq k < w$.

For each x_t in Σ **do**

For $0 \leq j \leq d - 1$ **do** $C[j, h_j(x_t)] \leftarrow C[j, h_j(x_t)] + 1$;

At the end of the stream: for any $u \in U$, its frequency f_u can be estimated as:

$$\tilde{f}_u = \min_{0 \leq j \leq d-1} C[j, h_j(u)].$$

Example: $n = 15, d = 3, w = 3$

$\Sigma = A, B, C, B, D, A, C, D, A, B, D, C, A, A, B$

u, f_u	h_0	h_1	h_2
A, 5	1	2	2
B, 4	2	3	2
C, 3	1	1	3
D, 3	2	2	3

Array C

h_0	0x2	0x2	0
h_1	01	01	0x2
h_2	0	0x23	01

1 2 3

- $\tilde{f}_A =$

- $\tilde{f}_B =$

- $\tilde{f}_C =$

- $\tilde{f}_D =$

Count-min sketch: analysis

We assume:

- the d hash functions h_1, h_2, \dots, h_d are mutually independent
- for each $j \in [0, d - 1]$ and each $u, v \in U$ with $u \neq v$, $h_j(u)$ and $h_j(v)$ are independent random variables uniformly distributed in $[0, w - 1]$.

Theorem

Consider a $d \times w$ count-min sketch for a stream Σ of length n , where $d = \log_2(1/\delta)$ and $w = 2/\epsilon$, for some $\delta, \epsilon \in (0, 1)$. The sketch ensures that for any given $u \in U$ occurring in Σ

$$\tilde{f}_u - f_u \leq \epsilon \cdot n,$$

with probability $\geq 1 - \delta$.

Obs.: The bias in the estimated frequencies discourages their use to estimate the second moment F_2 .

Count sketch

The **count sketch** was invented by [Charikar,Chen,Farach-Colton 2002], and can be seen as an **unbiased variant of the count-min sketch**.

IDEA: for each item $u \in U$ multiply its contributions to each row by a value in $\{-1, +1\}$ randomly selected, so to **cancel out collisions**.

Main ingredients

- $d \times w$ array C of counters ($O(\log n)$ bits each)
- d hash functions: h_0, h_1, \dots, h_{d-1} , with

$$h_j : U \rightarrow \{0, 1, \dots, w-1\},$$

for every j .

- d hash functions: g_0, g_1, \dots, g_{d-1} , with

$$g_j : U \rightarrow \{-1, +1\},$$

for every j .

Count sketch: algorithm

Initialization: $C[j, k] = 0$, for every $0 \leq j < d$ and $0 \leq k < w$.

For each x_t in Σ **do**

For $0 \leq j \leq d - 1$ **do** $C[j, h_j(x_t)] \leftarrow C[j, h_j(x_t)] + g_j(x_t)$;

At the end of the stream: for any $u \in U$ and $0 \leq j < d$, let

$$\tilde{f}_{u,j} = g_j(u) \cdot C[j, h_j(u)].$$

The frequency of u can be estimated as:

$$\tilde{f}_u = \text{median of the } \tilde{f}_{u,j}'s$$

Example: $n = 15, d = 3, w = 3$

$\Sigma = A, B, C, B, D, A, C, D, A, B, D, C, A, A, B$

u, f_u	h_0	g_0	h_1	g_1	h_2	g_2
A, 5	1	+1	2	+1	2	+1
B, 4	2	-1	3	+1	2	-1
C, 3	1	-1	1	-1	3	+1
D, 3	2	-1	2	+1	3	+1

Array C		

- $\tilde{f}_A =$

- $\tilde{f}_B =$

- $\tilde{f}_C =$

- $\tilde{f}_D =$

Count sketch: analysis

Assumptions: for both sets of hash functions (the h_j 's and the g_j 's) we make the same assumptions of **independence and uniform distribution**, which we made for the h_j 's in the analysis of the count-min sketch.

Theorem

Consider a $d \times w$ count sketch for a stream Σ of length n , where $d = \log_2(1/\delta)$ and $w = O(1/\epsilon^2)$, for some $\delta, \epsilon \in (0, 1)$. The sketch ensures that for any given $u \in U$ occurring in Σ :

- $E[\tilde{f}_{u,j}] = f_u$, for any $j \in [0, d - 1]$, i.e., $\tilde{f}_{u,j}$ is an unbiased estimator of f_u ;
- With probability $\geq 1 - \delta$,

$$|\tilde{f}_u - f_u| \leq \epsilon \cdot \sqrt{F_2},$$

where $F_2 = \sum_{u \in U} (f_u)^2$ (true second moment).

Intuition. Due to the random signs, on average the “noise” created by several items colliding on the same column as a u , cancel out.

Count-min sketch vs count sketch

Estimation of F_2

Given a $d \times w$ count sketch for Σ , define

$$\tilde{F}_{2,j} = \sum_{k=1}^w (C[j, k])^2 \quad \text{for } 0 \leq j < d$$

We can derive the following estimate for the true second moment F_2 :

$\tilde{F}_2 = \text{median of the } \tilde{F}_{2,j} \text{'s}$

Example (same as before)

$$\Sigma = A, B, C, B, D, A, C, D, A, B, D, C, A, A, B$$

$$F_2 = (f_A)^2 + (f_B)^2 + (f_C)^2 + (f_D)^2 = 5^2 + 4^2 + 3^2 + 3^2 = 59.$$

Array C from before		

- Estimate from row $j = 0$:
- Estimate from row $j = 1$:
- Estimate from row $j = 2$:

$$\Rightarrow \tilde{F}_2 =$$

Analysis of \tilde{F}_2

The following theorem can be proved under the same assumptions made for the analysis of the count sketch

Theorem

Consider a $d \times w$ count sketch for a stream Σ of length n , where $d = \log_2(1/\delta)$ and $w = O(1/\epsilon^2)$, for some $\delta, \epsilon \in (0, 1)$. The sketch ensures that:

- $E[\tilde{F}_{2,j}] = F_2$ for any $0 \leq j < d$. That is, any $\tilde{F}_{2,j}$ is an unbiased estimator of F_2 .
- With probability $\geq 1 - \delta$,

$$|\tilde{F}_2 - F_2| \leq \epsilon \cdot \sqrt{F_2}.$$

In the following slides, we show that $E[\tilde{F}_{2,j}] = F_2$ for every j , while we skip the proof of the second bullet point.

Analysis of performance metrics

Both count-min and count sketches can be computed in **1 pass**

To assess space and time performance, we assume:

- Each hash function can be applied in constant time
- The space occupied by the sketch dominates over the one needed to store the hash functions

For both sketches we have

- **Working memory:** $O(d \cdot w)$, which becomes $O(\log(1/\delta)/\epsilon)$, for the count-min sketch, and $O(\log(1/\delta)/\epsilon^2)$, for the count sketch, in order to attain the probabilistic accuracy stated before.
- **Processing time per element:** $O(d) = O(\log(1/\delta))$,

Moreover, given the sketch, the estimates \tilde{f}_u 's (individual frequencies) and \tilde{F}_2 (second moment) can be computed in $O(d)$ and $O(d \cdot w)$ time, respectively.

Filtering

Motivation

For many applications, processing a data stream $\Sigma = x_1, x_2, \dots$ entails essentially the identification of the x_i 's which meet a certain criterion.

Some criteria can be checked very easily with a minimum cost in terms of space and time. However, this is not always the case.

Example. Suppose that the x_i 's are email addresses and that when x_i arrives we need to check whether it belongs to a set S of verified addresses. If S is very large (e.g., 1 billion addresses of approximately 20 bytes each), we face two issues:

- If S does not fit into main memory, it must be stored on disk.
- Standard exact techniques to check $x_i \in S$, especially if S is on disk, may be time consuming and not compatible with a high arrival rate.

Can we check membership efficiently with reasonable accuracy?

Bloom filter

Approximate membership problem

Given a stream $\Sigma = x_1, x_2, \dots$ of elements from some universe U , and let S be a set of m elements from U . Store S into a compact data structure that, for any given x_i , allows to check whether $x_i \in S$ with

- no error, when $x_i \in S$ (No false negatives)
- small probability error, when $x_i \notin S$ (Small false positive rate)

A solution to the problem comes from the **Bloom filter**, introduced in [Bloom 1970]. Its **main ingredients** are:

- Array A of n bits, all initially 0.
- k hash functions: h_0, h_1, \dots, h_{k-1} , with

$$h_j : U \rightarrow \{0, 1, \dots, n-1\} \quad \text{for every } 0 \leq j < k$$

Note that n and k are *design parameters* that regulate the tradeoff between space/time and accuracy.

Bloom filter

Initialization:

For each $e \in S$ do

For $0 \leq j < k$ do $A[h_j(e)] \leftarrow 1$;

Membership test: for any x_i in Σ if

$$x_i \in S \Leftrightarrow A[h_0(x_i)] = A[h_1(x_i)] = \dots = A[h_{k-1}(x_i)] = 1$$

Straighforward properties:

- The approach ensures that there are no false negatives
- Assuming that $k \ll n$, and that the hash functions can be stored compactly, the required working memory is dominated by the storage of $A \Rightarrow n$ bits.
- Assuming that each hash function can be applied in $O(1)$ time, the membership test requires $O(k)$ time.

Example

- $S = \{A, B, C\}$
- $n = 12$
- $k = 2$

	A	B	C
h_0	0	1	4
h_1	4	9	9

The resulting array is:

0	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	1	0	0	0	0	1	0	0

Bloom filter: analysis of false positive rate

Assumptions: for the set of hash functions (the h_j 's) we make the same assumptions of **independence and uniform distribution**, which we made in the analysis of the count-min sketch.

Theorem

Suppose that n is sufficiently large. For any given x_i which does not belong to S , the probability that x_i is erroneously claimed to be in S is

$$\Pr(A[h_j(x_i)] = 1 \text{ for each } 0 \leq j < k) \simeq (1 - e^{-km/n})^k$$

This probability is referred to as *false positive rate*.

Email example: In the case of email addresses mentioned before, $m = 10^9$ and storing the entire set S would require **20GB** (assuming that each email takes 20 bytes). Using a Bloom filter with $n = 8m$ (hence $|A| = 1\text{GB}$), and $k = 6$, the false positive rate is about 2.15%.

Exercise

Consider a stream $\Sigma = x_1, x_2, \dots, x_n$ of n measurements from sensors. Each measurement x_i is a pair (k_i, w_i) , where k_i is the ID of a sensor and w_i is the value of the measurement (an integer). For a given sensor u occurring in Σ define

$$f_u = \sum_{(k_i, w_i) \in \Sigma : k_i = u} w_i,$$

i.e., the aggregate measurements taken by u .

- ① Briefly describe a space-efficient unbiased estimator for $\sum_u (f_u)^2$, where the sum is over all sensors occurring in the stream.
- ② What can you say about the unbiasedness of your estimator?

Exercise

Consider a Bloom filter built to assess membership for a set S of m elements. The Bloom filter consists of a n -bit array A and k hash functions h_0, h_1, \dots, h_{k-1} , mutually independent and with values uniformly distributed in $[0, n - 1]$. Assume that n is even, and that each hash function h_i is such that, for every $e \in S$, $h_i(e) \bmod n/2$ is uniformly distributed in $[0, n/2 - 1]$.

- ① Show how to transform, in $O(n)$ time, the given Bloom filter into a new Bloom filter based on an $n/2$ -bit array B , and describe how to assess membership with the new Bloom filter.
- ② Compute the probability that a given cell $B[i]$ of the new array is 0.

References

- [LRU14] J. Leskovec, A. Rajaraman and J. Ullman. **Mining Massive Datasets**. Cambridge University Press, 2014. Chapter 4 (Sections 4.3-4.5) (pdf provided in Moodle)
- [DF08] C. Demetrescu and I. Finocchi. **Algorithms for Data Streams**. In *Handbook of Applied Algorithms*, Chapter 8, Section 3. Wiley-IEEE Press, 2008. (pdf provided in Moodle)
- [CGHJ12] G. Cormode, M.N. Garofalakis, P.J. Haas, C. Jermaine. **Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches**. Foundations and Trends in Databases 4(1-3): 1-294, 2012. Chapter 5 (Sections 5.1-5.3) (pdf provided in Moodle)