

Foundations of Data Science

Lecture 11: Streaming Data

MING GAO

DaSE@ECNU

(for course related communications)

mgao@sei.ecnu.edu.cn

Dec. 9, 2016

Outline

- 1 Overview of Data Steaming
- 2 Counting Problem
- 3 Item Frequencies
- 4 Counting Distinct Elements

Motivations

Sensor data

Most of the algorithms are mining a database, where all our data is available when and if we want it. But...

- Imagine a temperature sensor bobbing about in the ocean, sending back to a base station a reading of the surface temperature each hour (not very interesting since the data rate is so low).
- Given the GPS sensor, and let it report surface height instead of temperature. If it sends a 4-byte real number every tenth of a second, then it produces 3.5 megabytes per day (not very interesting since # sensors is only one).
- To learn something about ocean behavior, we might want to deploy a million sensors (is not very many since there would be one for every 150 square miles), each sending back a stream, at the rate of ten per second. Now we have 3.5 terabytes arriving every day.

Motivations cont.

Image data

- Satellites often send down to earth streams consisting of many terabytes of images per day.
- Surveillance cameras produce images with lower resolution than satellites, but there can be many of them, each producing a stream of images at intervals like one second.
- London has six million such cameras, each producing a stream.

Internet traffic

- A switching node in the middle of the Internet receives streams of IP packets from many inputs and routes them to its output.
- Normally, the job of it is to transmit data and not to retain it or query it.
- But there is a tendency to put more capability into the switch, e.g., the ability to detect denial-of-service attacks or the ability to reroute packets based on information about congestion in the network.

Motivations cont.

Internet traffic

- Web sites receive streams of various types. Many interesting things can be learned from these streams.
- For example, an increase in queries like “sore throat” enables us to track the spread of viruses.
- A sudden increase in the click rate for a link could indicate some news connected to that page, or it could mean that the link is broken and needs to be repaired.

Many applications

Videos, email messages, webpages, chats, search queries, shopping history, GPS trials, financial transactions, stock exchange data, electricity consumption, Astronomy, Physics, medical imaging, weather measurements, maps, telephony data, audio tracks and songs, etc.

Streaming data

Characters

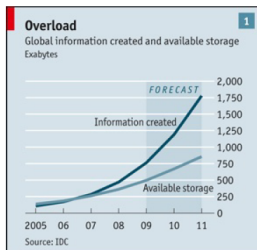
A sequence $\sigma = \langle a_1, a_2, \dots, a_m \rangle$, where the elements of the sequence are drawn from the universe $[n] := \{1, 2, \dots, n\}$.

- Continuously arriving
- Large volume
- High speed

Algorithms for data stream

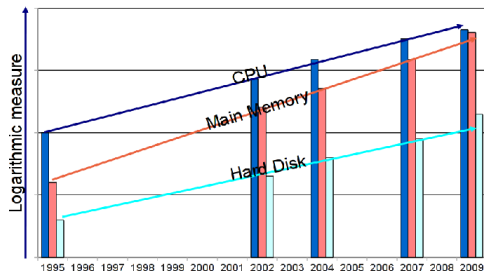
- Our central goal will be to process the input stream using a small amount of space s .
- We do not have random access to the tokens. We can only scan the sequence following the given order in p passes (some “small” integer p , for example $p = 1$).

The need for streaming data algorithm



Data growth

- IDC reports that enterprise data stores will grow an average of 60% annually.
- Moore law is the observation that the number of transistors in a dense integrated circuit doubles approximately every two years.

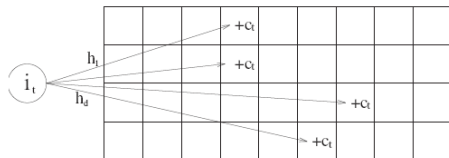


Streaming algorithm VS. traditional algorithm

Sketch

A sketch $C(\sigma)$ of some data σ with respect to some function ϕ is a compression of σ that allows us to compute or approximately compute, ϕ given access only to $C(\sigma)$.

- There exist exact solutions for some trivial tasks, such as count items, sum values, sample, find min or max.
- For non-trivial task, different problems maybe design different sketches to solve them.
- We are expected to give a bound for the estimation.



The quality of an algorithm's answer

Algorithm

We shall typically seek to compute only an approximation of the true value of $\phi(\sigma)$, because many basic functions can be provably not be computed exactly using sublinear space.

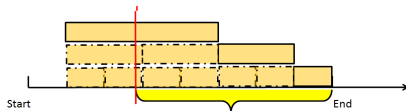
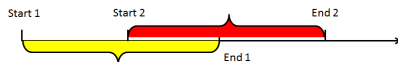
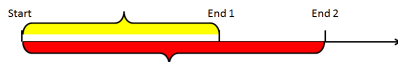
- Definition 1: let $\mathcal{A}(\sigma)$ denote the output of a randomized streaming algorithm \mathcal{A} on input σ ; note that this is a random variable. Let ϕ be the function that \mathcal{A} is supposed to compute. We say that the algorithm (ϵ, δ) -approximation ϕ if we have

$$P\left[\left|\frac{\mathcal{A}(\sigma)}{\phi(\sigma)} - 1\right| > \epsilon\right] \leq \delta.$$

- Definition 2: In the above setup, the algorithm (ϵ, δ) -additively-approximation ϕ if we have

$$P[|\mathcal{A}(\sigma) - \phi(\sigma)| > \epsilon] \leq \delta.$$

Streaming data modeling



Model

- Landmark model:
- Sliding window model:
- Exponential histogram model:

Counting problem

Problem definition

The algorithm must monitor a sequence of events, then at any given time output of the number of events thus far. More formally, this is a data structure maintaining a single integer n and supporting the following two operations:

- **update()**: increments n by 1;
- **query()**: must output (an estimate of) n .

Morris algorithm [Morris 1978]

- 1: initialize $X \leftarrow 0$;
- 2: for each update, increment X with probability $\frac{1}{2^X}$;
- 3: for a query, output $\hat{n} = 2^X - 1$.

Analysis of Morris' algorithm

Analysis

Let X_N denote X in Morris' algorithm after N updates. Then, we have:

- $E2^{X_N} = N + 1$.

Proof:

$$\begin{aligned}
 E2^{X_{N+1}} &= \sum_{j=0}^{\infty} P(X_N = j) E(2^{X_{N+1}} | X_N = j) \\
 &= \sum_{j=0}^{\infty} P(X_N = j) \left(2^j \left(1 - \frac{1}{2^j} \right) + \frac{1}{2^j} 2^{j+1} \right) \\
 &= \sum_{j=0}^{\infty} P(X_N = j) 2^j + \sum_{j=0}^{\infty} P(X_N = j) \\
 &= E2^{X_{N+1}} + 1 = (N + 1) + 1
 \end{aligned}$$

- $E2^{2X_N} = \frac{3}{2}N^2 + \frac{3}{2}N + 1$.

Analysis of Morris' algorithm cont.

Bound

It is now clear why we output our estimate of n as $\hat{n} = 2^X - 1$ since it is an unbiased estimator of n .

- Note that $E(\hat{n} - n)^2 = \frac{1}{2}n^2 - \frac{1}{2}n - 1 < \frac{1}{2}n^2$.
- If r.v. X has mean and variance $\mu = E(X)$ and $\sigma^2 = E[(X - \mu)^2]$, then $P(|X - \mu| > a) \leq \frac{\sigma^2}{a^2}$ or $P(|X - \mu| > aE(X)) \leq \frac{\sigma^2}{a^2 E(X)^2}$.
- $P(|\hat{n} - n| > \epsilon n) < \frac{n^2}{2\epsilon^2 n^2} = \frac{1}{2\epsilon^2}$.
- It is not particularly meaningful since the right hand side is only better than $\frac{1}{2}$ failure probability when $\epsilon \geq 1$.

Morris+

Boosting success probability

To decrease the failure probability of Morris' basic algorithm, we instantiate s independent copies of Morris' algorithm and average their outputs.

- That is, we obtain independent estimators $\hat{n}_1, \dots, \hat{n}_s$ from independent instantiations of Morris' algorithm, and our output to a query is

$$\hat{n} = \sum_{i=1}^s \hat{n}_i.$$

- $P(|\hat{n} - n| > \epsilon n) < \frac{1}{2s\epsilon^2} < \delta$ for $s > \frac{1}{2\epsilon^2\delta}$.

Morris++

Tug of War

It turns out there is a simple technique to reduce the dependence on the failure probability δ from $\frac{1}{\delta}$ to $\log 1/\delta$.

- We run t instantiations of Morris+, each with failure probability $\frac{1}{3}$, that is $s = O(1/\epsilon^2)$.
- We then output the median estimate from all the t Morris instantiations.
- Note that the expected number of Morris+ instantiations that succeed is at least $\frac{2t}{3}$.
- For the median to be a bad estimate, at most half the Morris+ instantiations can succeed, implying the number of succeeding instantiations deviated from its expectation by at least $\frac{t}{6}$.

Morris++ cont.

Tug of War

- Define $Y_i = \begin{cases} 1, & \text{if the } i\text{th Morris+ instantiation succeeds;} \\ 0, & \text{otherwise.} \end{cases}$
- Let X_i be a sequence of independent r.v.s with $P(X_i = 1) = p_i$ and $P(X_i = 0) = 1 - p_i$. r.v. $X = \sum_{i=1}^n X_i$. Then, we have $P(|X - \mu| \geq \epsilon\mu) \leq 2 \exp(-\mu\epsilon^2/3)$, where $\mu = \sum_{i=1}^n p_i$ and $\epsilon \in (0, 1)$ (Chernoff bound).
- Note that $\mu = E \sum_i Y_i = \frac{2t}{3}$. Then by the Chernoff bound,

$$\begin{aligned} P\left(\sum_i Y_i \leq \frac{t}{2}\right) &\leq P\left(\left|\sum_i Y_i - \frac{2t}{3}\right| \geq \frac{t}{6}\right) = P\left(\left|\sum_i Y_i - \mu\right| \geq \frac{1}{4}\mu\right) \\ &\leq 2 \exp(-2t/3(1/4)^2/3) < 2 \exp(-t/3) < \delta, \end{aligned}$$

for $t = O(\log 1/\delta)$.

- Finally, we can get an (ϵ, δ) -approximation in complexity $O(\frac{\log 1/\delta}{\epsilon^2})$.

Item frequencies

Problem definition

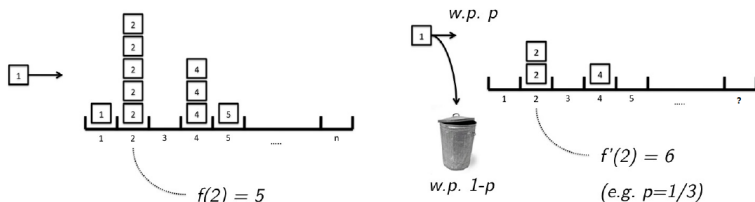
We have a stream $\sigma = \langle a_1, a_2, \dots, a_m \rangle$, with each $a_i \in [n]$ and this implicitly defines a frequency vector

$$f = (f_1, f_2, \dots, f_n).$$

Note that $\sum_{i=1}^n f_i = m$. In the Majority problem, our task is as follows: if $\exists j : f_j > \frac{m}{2}$, then output j , otherwise output \perp .

- General problem: given a parameter k , output the set $\{j : f_j > \frac{m}{k}\}$. Alternatively, given a parameter ψ , output the set $\{j : f_j > \psi m\}$.
- Frequency estimation problem: the task is to process σ to produce a data structure that can provide an estimate \hat{f}_a for the frequency f_a of a given token $a \in [n]$, then return the top- k results.
- For example, $\sigma = \langle a, b, a, c, c, a, b, d \rangle$, we have $f_a = 3$, $f_b = 2$, $f_c = 2$, $f_d = 1$. For $k = 4$, the frequent items are a, b, c . And for $\psi = 0.3$, the frequent item is a .

Naive approach



Sampling

Let $\sigma = \langle a_1, a_2, \dots, a_m \rangle$ be a stream, with each $a_i \in [n]$. Pick each element a_i from σ independently into our set $\hat{\sigma}$ with probability $p = \frac{M}{m}$.

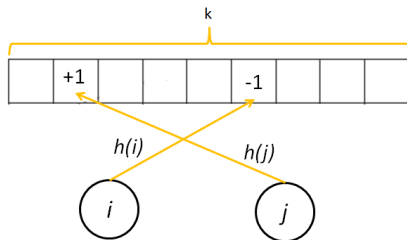
Let r.v. $X_i = \begin{cases} 1, & \text{element } a_i \text{ is picked;} \\ 0, & \text{otherwise.} \end{cases}$

Analysis of naive approach

Analysis

- The number of picked elements is $\sum_{i=1}^m X_i$. Its expectation is $E[\sum_{i=1}^m X_i] = \sum_{i=1}^m E[X_i] = \sum_{i=1}^m \frac{M}{m} = M$.
- g_i be the frequency of i in $\hat{\sigma}$, that is $g_i = \sum_{j:a_j=i} X_j$. We define $\hat{f}_i = \frac{g_i}{p}$.
- We have $E[\hat{f}_i] = \frac{E[g_i]}{p} = \frac{pf_i}{p} = f_i$, that is $E[g_i] = pf_i$.
- In terms of Chernoff bound, we have $P(|\hat{f}_i - f_i| \geq \epsilon f_i) = P(|g_i - pf_i| \geq \epsilon pf_i) = P(|\frac{g_i}{f_i} - p| \geq \epsilon p) \leq 2 \exp(-p\epsilon^2/3) < \delta$.
- Thus the space requirement is therefore $M = O(\frac{\log m/\delta}{\epsilon^2})$.

Basic count sketch



Algorithm

- 1: $C[1\dots k] \leftarrow \overleftarrow{0}$, where $k = \frac{3}{\epsilon^2}$;
- 2: Choose a random hash function $h : [n] \rightarrow [k]$ (uniformly);
- 3: Choose a random hash function $g : [n] \rightarrow \{-1, 1\}$ (uniformly);

Process item (j, c) , where $c = 1$:

- 4: $C[h(j)] \leftarrow C[h(j)] + cg(j)$;

Output:

- 5: On query a , report $\hat{f}_a = g(a)C[h(a)]$;

Count sketch analysis

Analysis

- Fix an arbitrary a and consider the output $X = \hat{f}_a$ on query a . For each token $j \in [n]$, let Y_j be the indicator for the event $h(j) = h(a)$ (collision).
- Token j contributes to $C[h(a)]$ iff $h(j) = h(a)$, and the amount of the contribution is its frequency f_j times the random sign $g(j)$.
- Thus

$$X = g(a) \sum_{j=1}^n f_j g(j) Y_j = f_a + \sum_{j \in [n] \setminus \{a\}} f_j g(a) g(j) Y_j.$$

- Since g and h are independent, we have $E[g(j) Y_j] = E[g(j)] E[Y_j] = 0 E[Y_j] = 0$.
- Thus $E[X] = f_a$, that is, the output is an unbiased estimator for f_a .

Count sketch analysis cont.

Analysis

We still need to show that X is unlikely to deviate too much from its means. Thus, we need to analyze its variance.

- For $j \in [n] \setminus \{a\}$, we have $E[Y_j^2] = E[Y_j] = P[h(j) = h(a)] = \frac{1}{k}$.
- For $i \neq j \in [n]$, we have
 $E[g(i)g(j)Y_iY_j] = E[g(i)]E[g(j)]E[Y_iY_j] = 0E[Y_iY_j] = 0..$
- Thus

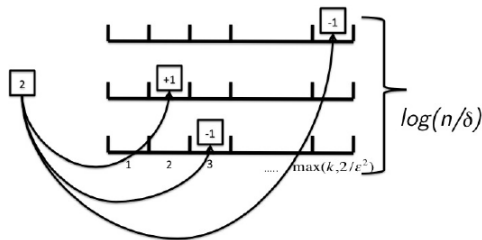
$$\begin{aligned}
 \text{Var}(X) &= 0 + g(a)^2 \text{Var}\left[\sum_{j \in [n] \setminus \{a\}} f_j g(j) Y_j\right] \\
 &= E\left[\sum_{j \in [n] \setminus \{a\}} f_j^2 Y_j^2 + \sum_{i \neq j \in [n] \setminus \{a\}} f_i f_j g(i) g(j) Y_i Y_j\right] - \left(\sum_{j \in [n] \setminus \{a\}} f_j E[g(j) Y_j]\right)^2 \\
 &= \sum_{j \in [n] \setminus \{a\}} \frac{f_j^2}{k} + 0 - 0 = \frac{\|f\|_2^2 - f_a^2}{k} = \frac{\|f_{-a}\|_2^2}{k}
 \end{aligned}$$

Count sketch analysis cont.

Using Chebyshev's inequality

$$\begin{aligned}
 P[|\hat{f}_a - f_a| \geq \epsilon \sqrt{\|f_{-a}\|_2}] &= P[|X - E(X)| \geq \epsilon \sqrt{\|f_{-a}\|_2}] \\
 &\leq \frac{\text{Var}(X)}{\epsilon^2 \|f_{-a}\|_2} = \frac{1}{k\epsilon^2} = \frac{1}{3}.
 \end{aligned}$$

Boosting the probability (Tug of war)



Boosting the probability

Tug of war

- 1: $C[1\dots t][1\dots k] \leftarrow \overleftarrow{0}$, where $k = \frac{3}{\epsilon^2}$ and $t = O(\log(1/\delta))$;
- 2: Choose t random hash functions $h_1, \dots, h_t : [n] \rightarrow [k]$ (uniformly);
- 3: Choose t random hash functions $g_1, \dots, g_t : [n] \rightarrow \{-1, 1\}$ (uniformly);

Process item (j, c) , where $c = 1$:

- 4: for i to t do $C[i][h_i(j)] \leftarrow C[i][h_i(j)] + cg_i(j)$;

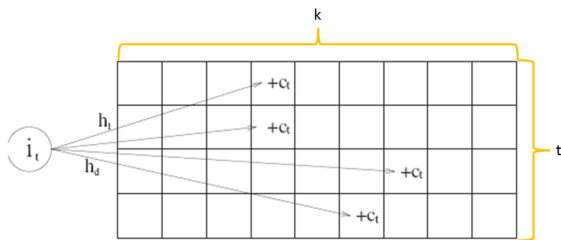
Output:

- 5: On query a , report $\hat{f}_a = \text{median}_{1 \leq i \leq t} g_i(a) C[i][h_i(a)]$;

- Thus the space requirement is therefore

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

Count min or Cormode-Muthukrishnan sketch



Algorithm

- 1: $C[1\dots t][1\dots k] \leftarrow \overleftarrow{0}$, where $k = \frac{2}{\epsilon}$ and $t = \lceil \log(1/\delta) \rceil$;
- 2: Choose t independent hash functions $h_1, h_2, \dots, h_t : [n] \rightarrow [k]$;

Process item (j, c) , where $c = 1$:

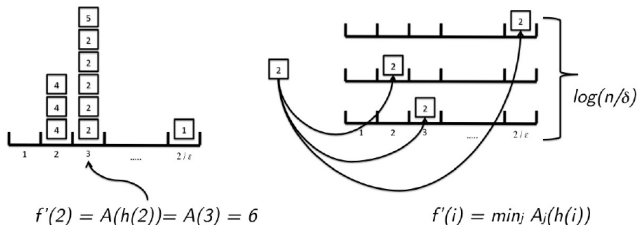
- 3: for $i = 1$ to t do $C[i][h_i(j)] \leftarrow C[i][h_i(j)] + c$;

Output:

- 4: On query a , report $\hat{f}_a = \min_{1 \leq i \leq t} C[i][h_i(a)]$;

CM sketch analysis

Analysis



- Clearly, for each i , we immediately have $f(a) \leq \text{count}[i, h_i(a)]$. However, the bound may be poor.
- To get a better estimator, we will take the minimum over all the rows in count.

CM sketch analysis cont.

Analysis

- For a fixed a , we now analyze the collision in one such counter, say in $\text{count}[i, h_i(a)]$. Let r.v. X_i denote this collision.
- For $j \in [n] \setminus \{a\}$, let $Y_{i,j}$ be the indicator of the event $h_i(j) = h_i(a)$. Notice that j makes a contribution to the counter iff $Y_{i,j} = 1$.
- Thus, we have $X_i = \sum_{j \in [n] \setminus \{a\}} f_j Y_{i,j}$. By linearity of expectation,

$$E[X_i] = \sum_{j \in [n] \setminus \{a\}} \frac{f_j}{k} = \frac{\|f\|_1 - f_a}{k} = \frac{\|f_{-a}\|_1}{k}.$$

Note that $E(Y_{i,j}) = \frac{1}{k}$.

- Since each $f_j \geq 0$, we have $X_i \geq 0$, and we can apply Markov's inequality to get $P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{\|f_{-a}\|_1}{k \epsilon \|f_{-a}\|_1} = \frac{1}{\epsilon}$ by our choice of k .

CM sketch analysis cont.

Analysis

- The above probability is for one counter. We have t such counters, mutually independent. The excess in the output $\hat{f}_a - f_a$, is the minimum of excesses X_i over all $i \in [t]$. Thus

$$\begin{aligned} P[\hat{f}_a - f_a \geq \epsilon \|f_{-a}\|_1] &= P[\min\{X_1, \dots, X_t\} \geq \epsilon \|f_{-a}\|_1] \\ &= \prod_{i=1}^t P[X_i \geq \epsilon \|f_{-a}\|_1] \leq \frac{1}{2^t}. \end{aligned}$$

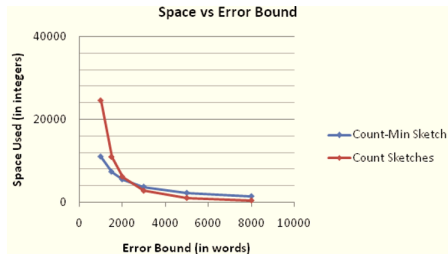
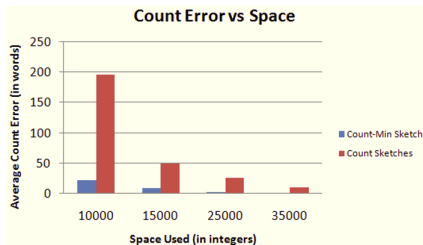
- Using our choice of t , this probability is at most δ . Thus, we have shown that, with high probability,

$$f_a \leq \hat{f}_a \leq f_a + \epsilon \|f_{-a}\|_1.$$

- Thus the space requirement is therefore $M = O\left(\frac{\log 1/\delta}{\epsilon} (\log m + \log n)\right)$.

Experimental comparison

Analysis



- It is possible to see that as the word count error bound decreases, the Count-Min sketch algorithm performs better.
- It is possible to see the Count sketches algorithm is a little bit slower in practice, probably because it runs twice as many hash calculations than the Count-Min sketch.

Take-home messages

- Overview of data streaming
 - Motivation
 - Example
 - Model
- Counting problem
 - Morris
 - Morris +
 - Morris ++
- Item frequency
 - Sampling
 - Count sketch
 - Count min sketch
-