Lecture IV – Mining data streams

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

CSE 547/STAT 548
Winter 2022

Data Streams

- **In many data mining situations, we do not know the entire data set in advance**
- **Stream Management** is important when the input rate is controlled **externally:**
  - Google queries
  - Twitter or Facebook status updates
- We can think of the **data** as **infinite** and **non-stationary** (the distribution changes over time)

The Stream Model

- Input **elements** enter at a rapid rate, at one or more input ports (i.e., **streams**)
  - **We call elements of the stream tuples**
- **The system cannot store the entire stream accessibly**
- **Q: How do you make critical calculations about the stream using a limited amount of (secondary) memory?**

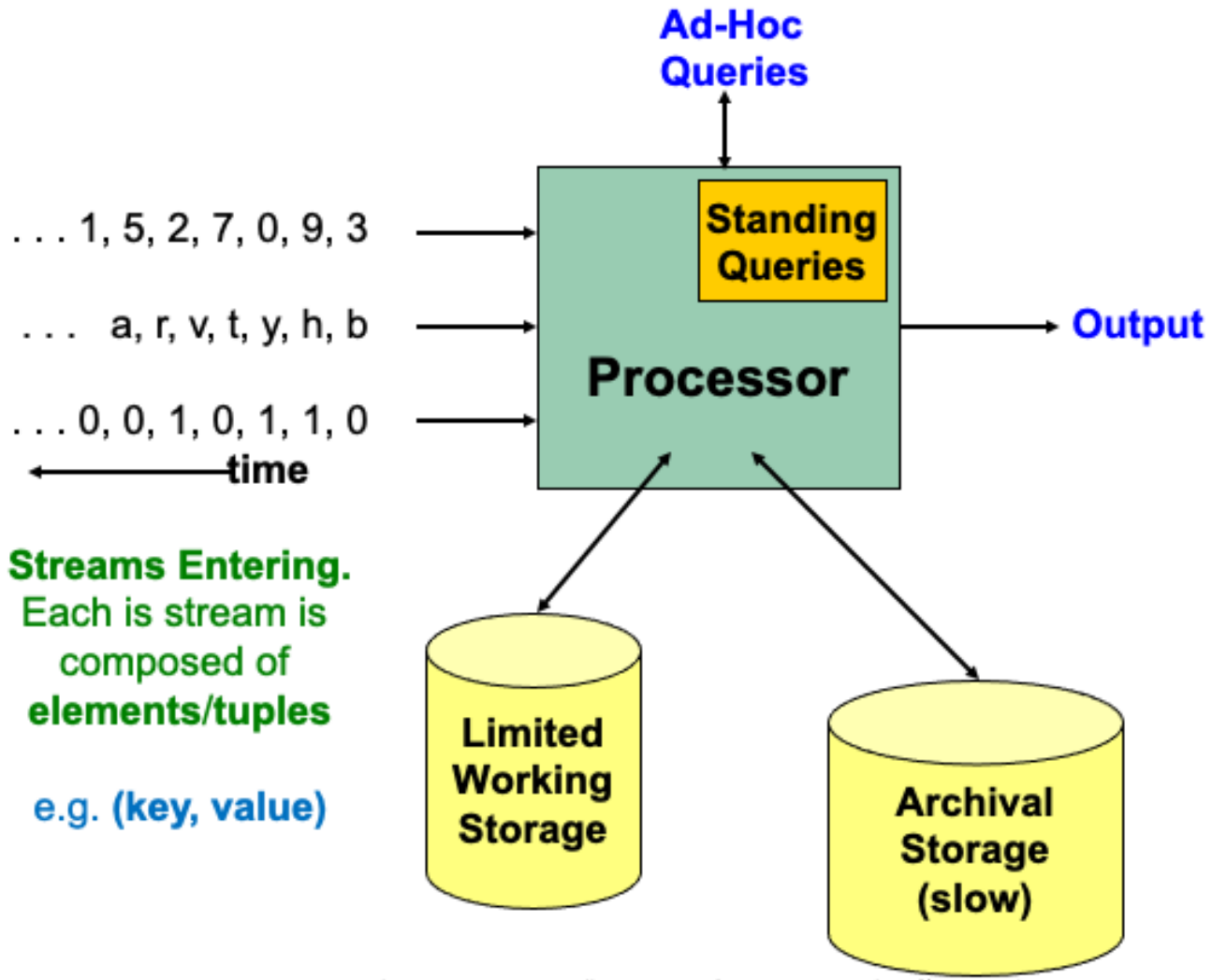
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 4

Side note: SGD is a Streaming Alg.

- **Stochastic Gradient Descent (SGD) is an example of a stream algorithm**
- **In Machine Learning we call this: Online Learning**
  - Allows for modeling problems where we have a continuous stream of data
  - We want an algorithm to learn from it and slowly adapt to the changes in data
- **Idea: Do slow updates to the model**
  - **SGD** (SVM, Perceptron) makes small updates
  - **So:** First train the classifier on training data.
  - **Then:** For every example from the stream, we slightly update the model (using small learning rate)

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 5

General Stream Processing Model



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 6

Problems on data streams

- Subsampling
- Maintaining a random sample: **Reservoir sampling**

- Counting over sliding windows (number of type x keys over last k items)
- Counting distinct elements **Flajolet-Martin**
- Filtering a stream **Bloom filter**
- Finding frequent elements
- Computing moments of count data **AMS method**

Applications (1)

- **Mining query streams**
  - Google wants to know what queries are more frequent today than yesterday
- **Mining click streams**
  - Yahoo wants to know which of its pages are getting an unusual number of hits in the past hour
- **Mining social network news feeds**
  - E.g., look for trending topics on Twitter, Facebook

Applications (2)

- **Sensor Networks**
  - Many sensors feeding into a central controller
- **Telephone call records**
  - Data feeds into customer bills as well as settlements between telephone companies
- **IP packets monitored at a switch**
  - Gather information for optimal routing
  - Detect denial-of-service attacks

Sampling from a data stream

- Sample a fixed proportion p of elements (e.g. $p = 1/10$)
  - Hashing!
- Maintain a sample of fixed size s (Reservoir sampling)

Sampling a Fixed Proportion

- **Problem 1: Sampling fixed proportion**
- **Scenario:** Search engine query stream
  - **Stream of tuples:** (user, query, time)
  - **Answer questions such as: How often did a user run the same query in a single days**
  - Have space to store **1/10th** of query stream
- **Naïve solution:**
  - Generate a random integer in **[0..9]** for each query
  - Store the query if the integer is **0**, otherwise discard

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 13

Problem with Naïve Approach

- **Simple question: What fraction of queries by an average search engine user are duplicates?**
  - Suppose each user issues ***x*** queries once and ***d*** queries twice (total of ***x+2d*** queries)
    - **Correct answer: *d*/(*x*+*d*)**
  - **Proposed solution: We keep 10% of the queries**
    - Sample will contain ***x*/10** of the singleton queries and ***2d*/10** of the duplicate queries at least once
    - But only ***d*/100** pairs of duplicates
      - **d/100 = 1/10 · 1/10 · d**
    - Of ***d*** "duplicates" ***18d/100*** appear exactly once
      - 2d/10 – 2d/100 = 18d/100
  - **So the sample-based answer is** $\frac{\frac{d}{100}}{\frac{x}{10}+\frac{18d}{100}+\frac{d}{100}} = \frac{d}{10x+19d}$

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 14

Solution: Sample Users

Solution:

- Pick **1/10th** of **users** and take all their searches in the sample
- Use a hash function that hashes the user name or user id uniformly into 10 buckets

Generalized Solution

- **Stream of tuples with keys:**
  - Key is some subset of each tuple's components
    - e.g., tuple is (user, search, time); key is **user**
  - Choice of key depends on application
- **To get a sample of *a/b* fraction of the stream:**
  - Hash each tuple's key uniformly into ***b*** buckets
  - Pick the tuple if its hash value is at most ***a***

Hash table with **b** buckets, pick the tuple if its hash value is at most **a.**
How to generate a 30% sample?
Hash into b=10 buckets, take the tuple if it hashes to one of the first 3 buckets

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 16

Maintaining a sample of fixed size

- Sample S of size $|S| = s$
- Fill with first s items, then randomly replace an existing item with the current item
- Problem: how to maintain a uniform sample?

Maintaining a sample of fixed size

- Sample S of size $|S| = s$
- Fill with first s items, then randomly replace an existing item with the current item
- Problem: how to maintain a uniform sample?

Uniform in streaming context

At time step $n \geq s$ for each $x \in S$, $Pr[x \in S] = \frac{s}{n}$.

Example $s = 2$, stream $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, \ldots$
at step $n = 5$, each of $x_{1:5}$ is included in S w.p. $2/5$

Reservoir sampling

Algorithm RESERVOIR SAMPLING

1. For $n = 1 : s$ fill S with $x_{1:s}$
2. For $n > s$, w.p. $\frac{s}{n}$
   keep item x_n
   select uniformly at random an item j in S and replace it with x_n

Reservoir sampling

Algorithm RESERVOIR SAMPLING

1. For $n = 1 : s$ fill S with $x_{1:s}$
2. For $n > s$, w.p. $\frac{s}{n}$
   keep item x_n
   select uniformly at random an item j in S and replace it with x_n

Proof by induction

- $n = s$, $x_{1:s} \in S$ w.p. 1
- Assume now Uniform sampling property true for step $n \geq s$
- Prove it for step $n + 1$

x_{n+1} is kept w.p. $\frac{s+1}{n+1}$ ✓

Reservoir sampling

Algorithm RESERVOIR SAMPLING

1. For $n = 1 : s$ fill S with $x_{1:s}$
2. For $n > s$, w.p. $\frac{s}{n}$
   keep item x_n
   select uniformly at random an item j in S and replace it with x_n

Proof by induction

- $n = s$, $x_{1:s} \in S$ w.p. 1
- Assume now Uniform sampling property true for step $n \geq s$
- Prove it for step $n + 1$

x_{n+1} is kept w.p. $\frac{s+1}{n+1}$ ✓

$x_j \in S$ replaced w.p. $1/s$, hence staying w.p. $d\,\frac{s}{s+1}$ if x_{n+1} kept, and w.p. 1 otherwise

- $P[x_j \in S \text{ at } n+1 \,|\, x_j \in S \text{ at } n] = \left(1 - \frac{s}{n+1}\right) + \frac{s}{n+1}\frac{s-1}{s} = \frac{n}{n+1}$
- But $x_j \in S$ w.p. $\frac{s}{n}$ at step n, hence

$$\begin{aligned} P[x_j \in S \text{ at } n+1] &= P[x_j \in S \text{ at } n] P[x_j \in S \text{ at } n+1 \,|\, x_j \in S \text{ at } n] \\ &= \frac{s}{n}\frac{n}{n+1} = \frac{s}{n+1} \checkmark \end{aligned}$$

Bloom Filter – Problem and first idea

- S is set of keys of interest
- $|S| = m$ –large

- Problem Filter from stream the items with keys in S
- Challenge Do it in time $< m$/item

- **First idea** hash the keys
- B is hash table of size $|B| = n$, with $n > m$
- Initially B filled with 0s
- $h : S \to 0 : n-1$ is hash function

Init for $s \in S$, $B[h(s)] \leftarrow 1$ (Preprocess)

Run for $a \in$ stream (Stream)

- if $B[h(a)] = 1$ output a

[1] For simplicity, we write $a \in S$ when we mean key$(a) \in S$.

Bloom Filter – Problem and first idea

- S is set of keys of interest
- $|S| = m$ –large

- Problem Filter from stream the items with keys in S
- Challenge Do it in time $< m$/item

- **First idea** hash the keys
- B is hash table of size $|B| = n$, with $n > m$
- Initially B filled with 0s
- $h : S \to 0 : n-1$ is hash function

Init for $s \in S$, $B[h(s)] \leftarrow 1$ (Preprocess)

Run for $a \in$ stream (Stream)
  - if $B[h(a)] = 1$ output a

- **Analysis**[1]
- False negatives (not outputting $a \in S$): never

[1] For simplicity, we write $a \in S$ when we mean key$(a) \in S$.

Bloom Filter – Problem and first idea

- S is set of keys of interest
- $|S| = m$ –large

- Problem Filter from stream the items with keys in S
- Challenge Do it in time $< m$/item

- **First idea** hash the keys
- B is hash table of size $|B| = n$, with $n > m$
- Initially B filled with 0s
- $h : S \to 0 : n-1$ is hash function

Init for $s \in S$, $B[h(s)] \leftarrow 1$ (Preprocess)

Run for $a \in$ stream (Stream)
  - if $B[h(a)] = 1$ output a

- **Analysis**[1]
- False negatives (not outputting $a \in S$): never
- False positives (outputting $a \notin S$): $h[a] = h[s]$ for some $s \in S$ given $a \notin S$

[1] For simplicity, we write $a \in S$ when we mean key$(a) \in S$.

Bloom Filter – Problem and first idea

- S is set of keys of interest
- $|S| = m$ –large

- Problem Filter from stream the items with keys in S
- Challenge Do it in time $< m$/item

- **First idea** hash the keys
- B is hash table of size $|B| = n$, with $n > m$
- Initially B filled with 0s
- $h : S \to 0 : n-1$ is hash function

Init for $s \in S$, $B[h(s)] \leftarrow 1$ (Preprocess)

Run for $a \in$ stream (Stream)
  - if $B[h(a)] = 1$ output a

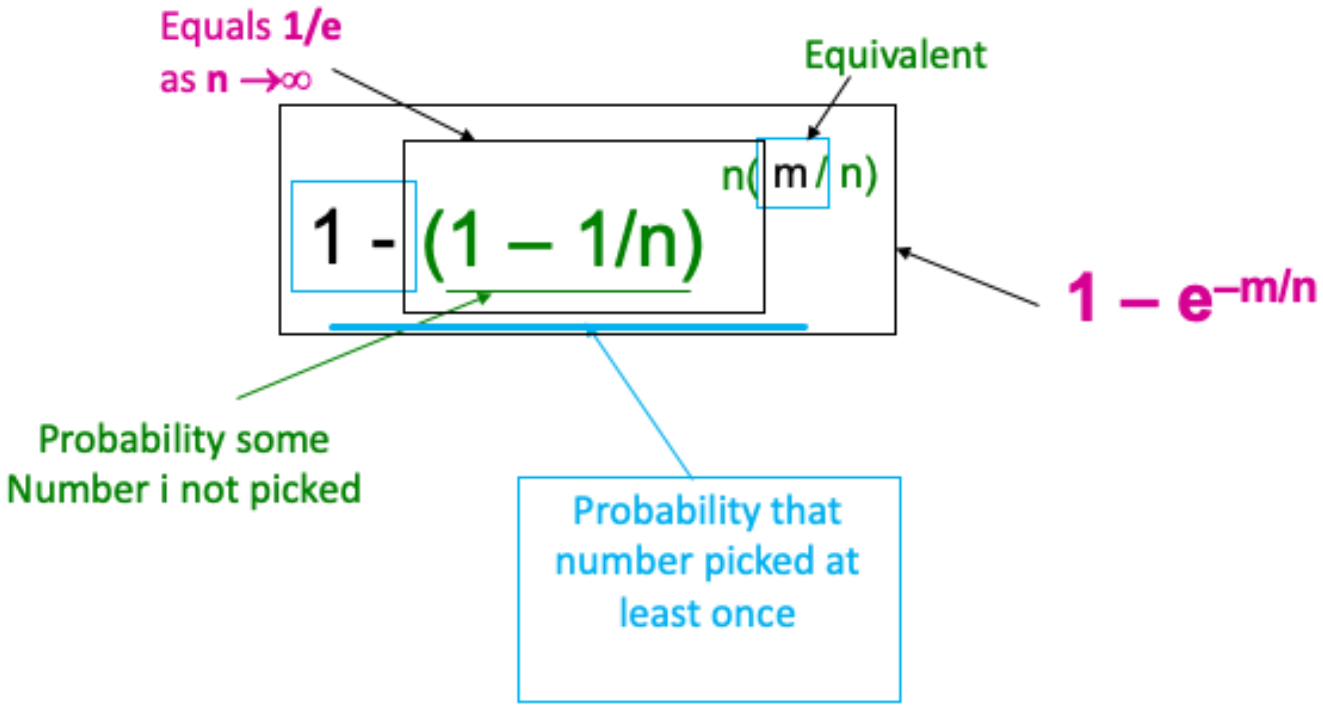
- **Analysis**[1]
- False negatives (not outputting $a \in S$): never
- False positives (outputting $a \notin S$): $h[a] = h[s]$ for some $s \in S$ given $a \notin S$
- What is the probability of this occurrence?

[1] For simplicity, we write $a \in S$ when we mean key$(a) \in S$.

Probability of a False Positive

- False positives (outputting $a \notin S$): $h[a] = h[s]$ for some $s \in S$ given $a \notin S$
- $h[a] \sim uniform(0 : n-1)$ therefore $p_{FP} = \#(B_j = 1)/n =$ fraction of occupied entries in table
-
- Remark **Hashing the keys is like picking m balls from an urn containing n balls numbered $0 : n-1$, with replacement** How many distinct numbers we expect?
- Also like throwing m darts into n targets
- To calculate: $Pr[B_i = 1]$ after all m keys are hashed

- We have ***m*** draws, ***n*** numbers = ***m*** darts, ***n*** targets
- What is the probability that a number gets picked at least once?



J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 10

Bloom Filter – Algorithm

- S is set of keys, B is hash table, $|S| = m$, $|B| = n$, with $n > m$
- Initially B filled with 0s
- $h_{1:k} : S \to 0 : n-1$ are random hash functions

Init for $s \in S$, and for $j = 1 : k$, $B[h_k(s)] \leftarrow 1$
(we set k bits to 1 for each s)

Run for $a \in$ stream (Stream)

- if $B[h_j(a)] = 1$ for all $j \in 1 : k$ output a

- Probability of a False Positive
  - Fraction of B entries set to 1: $1 - e^{-km/n}$ (km draws from n numbers)
  - Output a only if we find "1" in k table entries
  - Hence $Pr[a \text{ False Positive}] = (1 - e^{-km/n})^k$

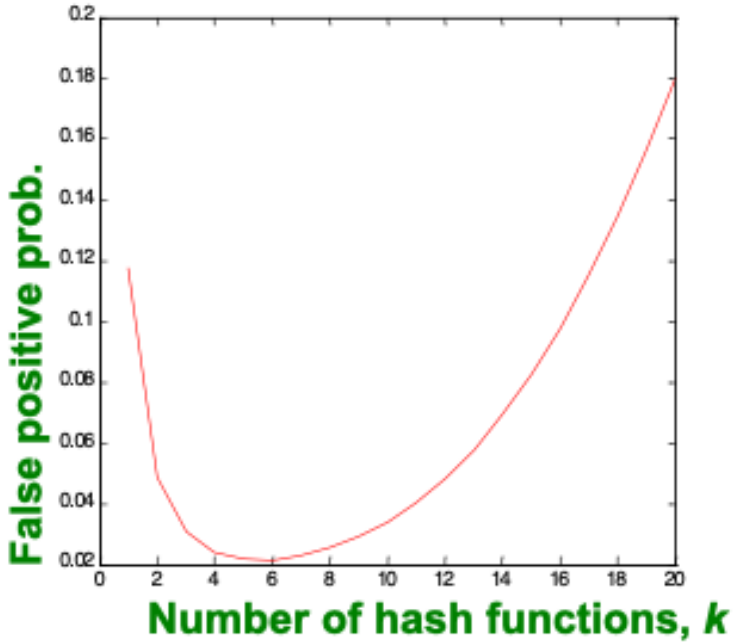
Bloom Filter -- Analysis

- **What fraction of the bit vector B are 1s?**
  - Throwing $k \cdot m$ darts at n targets
  - So fraction of **1**s is $(1 - e^{-km/n})$
- But we have k independent hash functions and we only let the element x through **if all** k hash element x to a bucket of value **1**
- So, false **positive probability** = $(1 - e^{-km/n})^k$

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 13

Bloom Filter – Analysis (2)

- m = 1 billion, n = 8 billion
  - **k = 1**: $(1 - e^{-1/8}) = 0.1175$
  - **k = 2**: $(1 - e^{-1/4})^2 = 0.0493$



- **What happens as we keep increasing k?**
- "Optimal" value of k: n/m ln(2)
  - **In our case:** Optimal **k = 8 ln(2) = 5.54 ≈ 6**
    - **Error at k = 6**: $(1 - e^{-1/6})^2 = 0.0235$

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 14

Bloom Filter: Wrap-up

- **Bloom filters guarantee no false negatives, and use limited memory**
  - Great for pre-processing before more expensive checks
- **Suitable for hardware implementation**
  - Hash function computations can be parallelized
- Is it better to have **1** big **B** or ***k*** small **Bs**?
  - **It is the same:** $(1 - e^{-km/n})^k$ vs. $(1 - e^{-m/(n/k)})^k$
  - But keeping **1 big B** is simpler

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 15

Applications

- **How many different words are found among the Web pages being crawled at a site?**
  - Unusually low or high numbers could indicate artificial pages (spam?)
- **How many different Web pages does each customer request in a week?**
- **How many distinct products have we sold in the last week?**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 18

Using Small Storage

- **Real problem: What if we do not have space to maintain the set of elements seen so far?**
- **Estimate the count in an unbiased way**
- **Accept that the count may have a little error, but limit the probability that the error is large**

Flajolet-Martin Approach

- Pick a hash function h that maps each of the N elements to at least $\log_2 N$ bits
- For each stream element a, let $r(a)$ be the number of trailing **0s** in $h(a)$
  - **r(a)** = position of first 1 counting from the right
    - E.g., say $h(a) = 12$, then 12 is 1100 in binary, so $r(a) = 2$
- Record R = the maximum $r(a)$ seen
  - $R = \max_a r(a)$, over all the items a seen so far
- **Estimated number of distinct elements = 2^R**

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 20

Flajolet-Martin: First intuition

- In base 10: 0,1,2,3,. . . 10,11,. . . ....
  - every 10-th number ends in 0
  - every 100-th number ends in 00
  - . . .
- In base 1: 0,1,10,11,100,101,110,111,1000,. . .
  - every 2-nd number ends in 0
  - every 4-th number ends in 00
  - every 2^r-th number ends in $\underbrace{0 \ldots 0}_{\times r}$ (these are the multiples of 2^r
- Pr[observe a multiple of 2^r]$= \frac{1}{2^r}$
- If multiple of 2^R observed: we must have taken about 2^R distinct samples

Why It Doesn't Work

- **$E[2^R]$ is actually infinite**
  - Probability halves when $R \rightarrow R+1$, but value doubles
- **Workaround involves using many hash functions h_i and getting many samples of R_i**
- **How are samples R_i combined?**
  - **Average?** What if one very large value 2^{R_i}?
  - **Median?** All estimates are a power of **2**
  - **Solution:**
    - Partition your samples into small groups
    - Take the median of groups
    - Then take the average of the medians

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org 25

Flajolet-Martin: Why not working