Lecture 12

Heavy Hitters in Data Streams

Course: Algorithms for Big Data

Instructor: Hossein Jowhari

Department of Computer Science and Statistics
Faculty of Mathematics
K. N. Toosi University of Technology

Spring 2021

A General Framework for Data Stream Problems

We have an initially-zero vector $\boldsymbol{x} \in \mathbb{R}^n$ where n is large.

$$\boldsymbol{x} = (0, 0, \ldots, 0)$$

Every stream item is an update of some coordinate in $\boldsymbol{x}$

$$(i_1, +2.8), \ldots, \overbrace{(i_k}^{\text{coordinate index}}, \overbrace{+2.3)}^{\text{update value}}, \ldots, (i_m, +10)$$

The stream item (i, u) means the value u is added to the i-th coordinate.

$$(i, u) \quad \rightarrow \quad \text{add } u \text{ to } x_i$$

At the end, we like to have an estimate of $f(\boldsymbol{x})$ for some function f

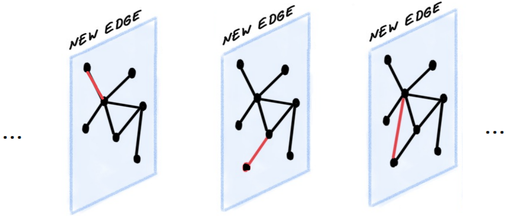
The General Framework: Special Cases

Insert-only Model (for Graph Streams)

Every update $u = 1$ and each coordinate i is updated at most one time $\quad (i_1, 1), (i_2, 1), \ldots, (i_m, 1)$

Specific problems: Maximum Matching, Number of Connected Components, ...

The vector $\boldsymbol{x} \in \{0,1\}^{\binom{n}{2}}$ represents a graph on n vertices. An stream item is the insertion of an edge e_i.

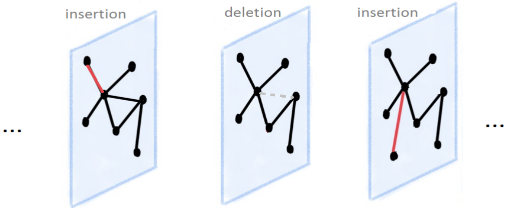


Insertion/Deletion Model (for Graph Streams)

Every update $u \in \{+1, -1\}$ and each coordinate i may be updated multiple times but every deletion $(u = -1)$ is preceded by an insertion $(u = +1)$.

Specific problems: Dynamic Maximum Matching, Dynamic Spanning Tree , ...

The vector $\boldsymbol{x} \in \{0,1\}^{\binom{n}{2}}$ represents a graph on n vertices. An stream item is the insertion/deletion of an edge e_i.



Cash Register Model

Every update u is a positive number. Each coordinate i may be updated multiple times. $(i_1, +4), (i_2, +1), \ldots, (i_m, +6)$

Specific problems:

- Frequency moments $F_k = \sum_{i=1}^{n} x_i^k$.

  (Here the vector $\boldsymbol{x} \in \mathbb{Z}^{+n}$ represents a frequency vector. Each stream item increments a coordinate of $\boldsymbol{x}$.)

- Finding the most frequent element: $\arg\max_{i=1}^{n} x_i$

- Empirical entropy $H = \sum_{i=1}^{n} -\frac{x_i}{F_1} \log \frac{x_i}{F_1}$

- Weighted graph problems

Turnstile Model

In the **turnstile model**, every update $u \in \mathbb{R}$ and each coordinate i may be updated multiple times.
$(i_1, +4.2), (i_2, -1.9), \ldots, (i_m, +6.5)$

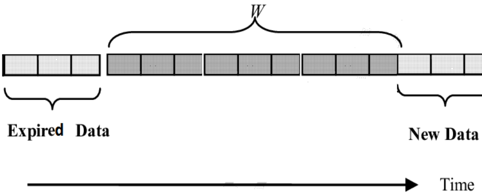
The **strict turnstile model** is like the turnstile model except that no x_i never goes below zero. At all times $x_i \geq 0$.

Specific problems:

- Estimating the ℓ_p norm $\|\boldsymbol{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$.
- Various matrix norms (Frobenius norm, Spectral norm, ...)
- Weighted graph problems (strict turnstile)

Sliding Window Model

In this model, we are interested in computing the $f(x)$ when the input is restricted to the last W data items.



The space of the algorithm is NOT enough to store the entire window.

Heavy Hitters

Finding the most frequent element (in the cash-register model) requires $\Omega(n)$ space in general.

We study a relaxed version of the problem:

Definition: Given a frequency vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, the coordinate i is a ϵ-HH (Heavy Hitter) iff

$$x_i \geq \epsilon \sum_{i=1}^{n} x_i = \epsilon \|\boldsymbol{x}\|_1 = \epsilon F_1$$

When $\epsilon > \frac{1}{2}$, an ϵ-HH is called a majority element.

The number of coordinates that are ϵ-HH is at most $\frac{1}{\epsilon}$.

Streaming Algorithms for Finding Heavy Hitters

Counter-based algorithms: these algorithm find the most frequent items by storing a subset of the elements along with a counter (an estimate) for this occurrences. A few examples:

Majority-based algorithm (Misra-Gries), **Space Saving**, **Lossy Count**

Sketch-based algorithms: these algorithm keep a summary of data which often consists of the inner products of the frequency vector and some random vector. A few examples:

CountMin, CountSketch

The majority-based algorithm

This algorithm is rediscovered many times by various people. (Boyer-Moore, Karp-Papadimitriou, Misra-Gries)

Let H_ϵ denote the set of coordinates that are ϵ-HH.

Description of the result: The majority-based algorithm outputs the subset $S \subseteq [n]$ where $H_\epsilon \subseteq S$ and $|S| \leq \frac{1}{\epsilon}$. The algorithm works in $O(\frac{1}{\epsilon})$ words of space.

Given an additional pass over the stream, the algorithm can eliminate all elements in S that are not in H_ϵ.

Finding the majority element

Lets consider a special case: $\epsilon \in (\frac{1}{2}, 1]$. In this case $|H_\epsilon| \leq 1$.

$\text{Stream} : a, b, a, a, a, f, a, h, a, j, k, t, a, b, a, a, a, a, c, a$

length of stream = 20, $x_a = 12$ (a is the majority element)

Algorithm: Keep an (element, counter) pair (v, c). In the beginning, $v = \varnothing$ and $c = 0$.

For item x in the stream do the following:

- If $v = \varnothing$, set $v \leftarrow x$ and $c \leftarrow 1$.
- Otherwise if $v \neq x$, $c \leftarrow c - 1$. If $c = 0$ then $v \leftarrow \varnothing$.
- Otherwise if $v = x$, $c \leftarrow c + 1$

Stream = a, b, a, a, a, f, a, h, a, j, k, t, a, b, a, a, a, a, c, a

element	counter	next item
	0	a
a	1	b
	0	a
a	1	a
a	2	a
a	3	f
a	2	a
a	3	h
a	2	a
a	3	j
a	2	k
a	1	t
	0	a
a	1	b
	0	a
a	1	a
a	2	a
a	3	a
a	4	c
a	3	a
a	4	

In case the stream does not have a majority the algorithm might return a non-majority element.

Generalization of the idea: Suppose we keep k element-counter pairs.

$$(v_1, c_1,), (v_2, c_2), \ldots, (v_k, c_k)$$

In the beginning, each $v_i = \varnothing$ and $c_i = 0$.

For item x in the stream do the following:

- If there is $v_i = \varnothing$, set $v_i \leftarrow x$ and $c_i \leftarrow 1$.
- Otherwise if there is $v_i = x$, set $c_i \leftarrow c_i + 1$.
- Otherwise, for all i, $c_i \leftarrow c_i - 1$. If there is $c_i = 0$ set $v_i \leftarrow \varnothing$.

At the end, let S be the set of elements where their corresponding counters is non-zero. The algorithm outputs S as the candidates for heavy hitters.

Example

stream length $= 32$ number of counters $k = 3$

element	counter	element	counter	element	counter	next item
	0		0		0	f
f	1		0		0	g
f	1	g	1		0	h
f	1	g	1	h	1	d
	0		0		0	c
c	1		0		0	c
c	2		0		0	d
c	2	d	1		0	a
c	2	d	1	a	1	b
c	1		0		0	t
c	1	t	1		0	a
c	1	t	1	a	1	w
	0		0		0	a
a	1		0		0	s
a	1	s	1		0	a
a	2	s	1		0	b
a	2	s	1	b	1	a
a	3	s	1	b	1	b
a	3	s	1	b	2	c
a	2		0	b	1	n
a	2	n	1	b	1	a
a	3	n	1	b	1	c
a	2		0		0	c
a	2	c	1		0	a
a	3	c	1		0	a
a	4	c	1		0	b
a	4	c	1	b	1	f
a	3		0		0	c
a	3	c	1		0	a
a	4	c	1		0	c
a	4	c	2		0	c
a	4	c	3		0	c
a	4	c	4		0	

Claim: The candidate set S contains all elements in H_ϵ where $\epsilon = \frac{1}{k}$.

Proof: Consider $a \in H_\epsilon$. We have $x_a \geq \frac{m}{k}$. Recall $m = F_1$ is the length of the stream. We claim the element a should be in S at the end. Note that every time the algorithm decreases the values of the k counters upon seeing a new element x, it is as if it throws away $k+1$ different elements from the stream. This can be done at most $\frac{m}{k+1}$ times. Since $x_a \geq \frac{m}{k} > \frac{m}{k+1}$, some occurrences of a remain at the end. Therefore the element a should be in candidate set S.

Claim: For $a \in S$, let x'_a be the value of the corresponding counter. We have

$$x_a - \frac{m}{k+1} \leq x'_a \leq x_a$$