Streaming space complexity of nearly all functions of one variable

Vladimir Braverman, Stephen Chestnut, Lin F. Yang

April 28, 2015

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$4,\ \ 2,\ \ 3,\ \ 2,\ \ 4,\ \ 2,\ \ \ 2$$

$$f = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\sum f_i^2 = 0$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$4,\ 2,\ 3,\ 2,\ 4,\ 2,\ 2$$

$$f = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\sum f_i^2 = 1$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$2,\ 3,\ 2,\ 4,\ 2,\ 2$$

$$f = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\sum f_i^2 = 2$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$3,\ 2,\ 4,\ 2,\ 2$$

$$f = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\sum f_i^2 = 3$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$2,\ 4,\ 2,\ 2$$

$$f = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

$$\sum f_i^2 = 6$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$4,\ 2,\ 2$$

$$f = \begin{bmatrix} 0 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

$$\sum f_i^2 = 9$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$2, \quad 2$$

$$f = \begin{bmatrix} 0 \\ 3 \\ 1 \\ 2 \end{bmatrix}$$

$$\sum f_i^2 = 14$$

Streaming: Memory limited computing

A stream of $m = 7$ items from $[n] = [4]$

$$\begin{matrix} & 2 \\ f = & \begin{bmatrix} 0 \\ 4 \\ 1 \\ 2 \end{bmatrix} \\ \sum f_i^2 = & 21 \end{matrix}$$

Applications of streaming & sketching

Streaming data

- IP traffic analysis ($\approx 10^6$ packets/sec)
- rapid scientific data (CERN: 800Gb/s, SDSS: 200GB/night)

Faster algorithms from sketches

- graph spanners and sparsifiers
- approximate numerical linear algebra (mat. mul., low rank approx.)

Distributed computing

- (distributed) database monitoring and approximate queries
- processing distributed data & parallelization

The basic problem

Given a function $G : \mathbb{Z}^n \to \mathbb{R}$, how much storage is needed to approximate the value $G(f)$?

Examples:

- General f
  - frequency moments: $G(f) = \sum_i f_i^p$
  - entropy: $G(f) = -\sum_i \frac{f_i}{m} \log \frac{f_i}{m}$
  - order statistics: $G(f) = \max_i f_i$
- $f \in \{0,1\}^{\binom{V}{2}}$ is a graph
  - min-cut
  - max-matching
- $f \in \mathbb{R}^{r \times c}$ is a matrix
  - determinant
  - mat. mul.
  - low-rank approximation
  - approximate least-squares

Remainder of the talk

Given a function $g : \mathbb{Z}_{\geq 0} \to \mathbb{R}$ and $\epsilon > 0$ is there a $(1 \pm \epsilon)$-approximation algorithm for $\sum_i g(f_i)$ using only polylog(nm) bits?

Does the same algorithm work for all g?

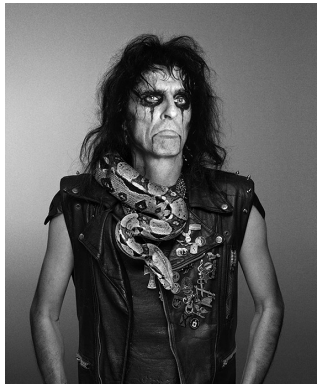
Outline

- Storage lower bound example
  - The maximum frequency
- Streaming algorithms
  - Recursive Subsampling
  - Count Sketch
- Our contribution
  - Answer the question for *almost* any g

$$\epsilon = \Omega(\tfrac{1}{\text{polylog}(n)})$$
$$m = \text{poly}(n)$$
$$g(0) = 0$$
$$g(x) > 0, \ \forall x > 0$$

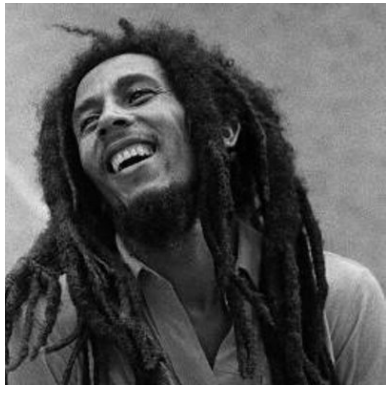
$\max_i f_i$ lower bound [Alon, Matias, & Szegedy '96]

1-way communication complexity of $index(n)$ is $\Omega(n)$



Alice:
$A = \{a_1, a_2, \ldots\} \subseteq [n]$

message

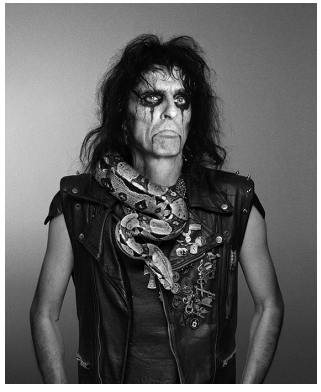


Bob:
$b \in [n]$

$\max_i f_i$ lower bound [Alon, Matias, & Szegedy '96]

1-way communication complexity of $index(n)$ is $\Omega(n)$

$(1 \pm \frac{1}{4})$-approximation algorithm $\mathcal{A}$ on stream S: $\mathcal{A}(S) \in (1 \pm \frac{1}{4}) \max_i f_i$

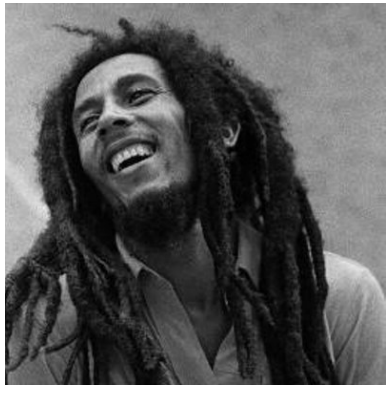


Alice:

$A = \{a_1, a_2, \ldots\} \subseteq [n]$

message

memory of $\mathcal{A}(a_1, a_2, \ldots)$



Bob:

$b \in [n]$

$$\mathcal{A}(a_1, a_2, \ldots, b) \text{ is } \begin{cases} \leq 1.25 & \text{if } b \notin A \\ \geq 1.5 & \text{if } b \in A \end{cases}$$

More lower bound examples

Bounds for the storage need to approximate $\sum_i g(f_i)$.

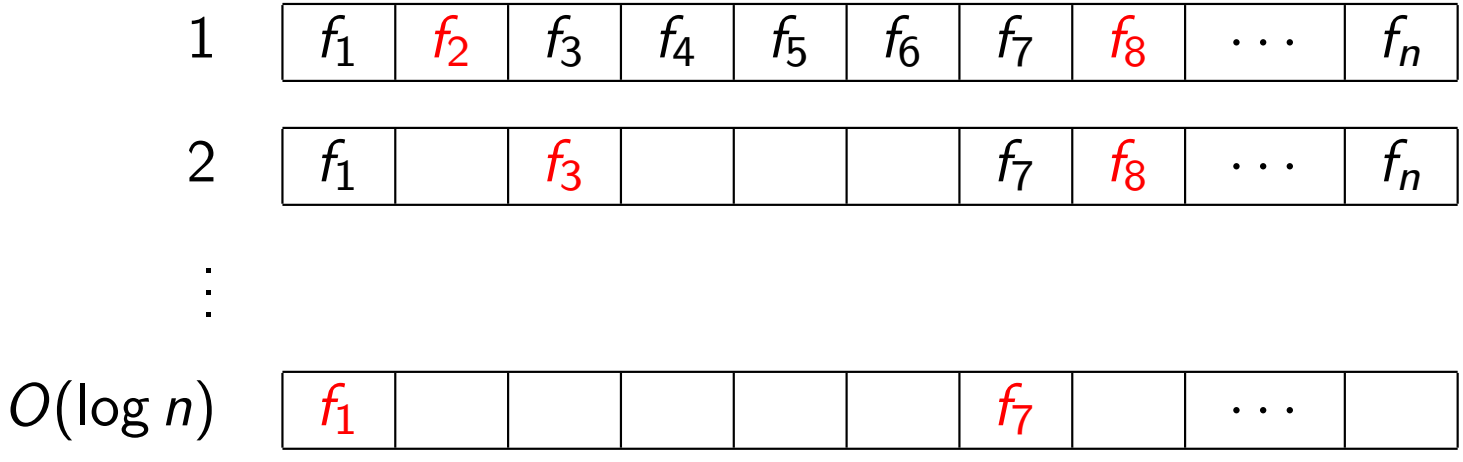
hard		easy	
$g(x)$	lower bound	$g(x)$	upper bound
$1/x$	$\Omega(n)$	$\frac{1}{1+\log x}$	$\text{poly}(\frac{1}{\epsilon}\log n)$
x^3	$\Omega(n^{1/3})$	x^2	$O(\epsilon^{-2}\log n)$
$(2+\sin\frac{\pi}{2}x)x^2$	$\Omega(n)$	$(2+\sin\frac{\pi}{2}\log x)x^2$	$\text{poly}(\frac{1}{\epsilon}\log n)$

Recursive Subsampling [Indyk & Woodruff '05]

An α-heavy hitter is any item i^* such that $g(f_{i^*}) \geq \alpha \sum_i g(f_i)$.

Theorem (Braverman & Ostrovsky 2010)

If there is a space s algorithm that finds all $\frac{\epsilon^2}{\log^3 n}$-heavy hitters i^* and for each returns a $(1 \pm \epsilon)$-approximation to $g(f_{i^*})$ then there is a space $O(s \log n)$ algorithm that outputs a $(1 \pm \epsilon)$-approximation to $\sum_i g(f_i)$.



Count Sketch [Charikar, Chen, & Farach-Colton '02]

Goal: α-heavy hitters for $\sum_i f_i^2 = \|f\|_2^2$

Count Sketch [Charikar, Chen, & Farach-Colton '02]

Goal: α-heavy hitters for $\sum_i f_i^2 = \|f\|_2^2$

- $B = 3\alpha^{-1}\epsilon^{-2}$
- $h : [n] \to [B]$ pairwise-indep.
- $\sigma : [n] \to \{-1, 1\}$ pairwise-indep.

$$C_b = \sum_{i:h(i)=b} \sigma_i f_i, \text{ for } b \in [B].$$

Example

5,6	$C_1 = -f_5 + f_6$
3,4,7	$C_2 = -f_3 + f_4 - f_7$
8,10	$C_3 = f_8 - f_{10}$
	$C_4 = 0$
1,2,9	$C_5 = -f_1 - f_2 + f_9$

Easy calculations for $\hat{f}_i = \sigma_i C_{h(i)}$:

$$E\hat{f}_i = f_i, \qquad Var(\hat{f}_i) = \frac{1}{B}\sum_{j \neq i} f_j^2 \leq \frac{\alpha}{3}\left(\epsilon\|f\|_2\right)^2.$$

Count Sketch [Charikar, Chen, & Farach-Colton '02]

Data structure $R = O(\log n)$ vectors in $\mathbb{Z}^B$ with independent randomness

$$C^{(1)}, C^{(2)}, \ldots, C^{(R)}.$$

Query Estimated frequency of i

$$\hat{f}_i = \operatorname*{median}_{r \in [R]} \sigma_i^{(r)} C^{(r)}_{h^{(r)}(i)}$$

Guarantee With high probability for all $i \in [n]$

$$\left| f_i - \hat{f}_i \right|^2 \leq \epsilon^2 \alpha \| f \|_2^2.$$

Space $O(\alpha^{-1} \epsilon^{-2} \log n \log m)$ bits.

Three properties are sufficient for $\widetilde{O}(1)$ bits

Three properties

There exists $h \in \widetilde{O}(1)$ such that for all integers $0 < x < y$

slow-dropping $\frac{1}{h(y)} \leq \frac{g(y)}{g(x)}$,

slow-jumping $\frac{g(y)}{g(x)} \leq h(y)\left(\frac{y}{x}\right)^2$, and

predictable whenever $x \leq \frac{y}{h(y)}$

$$\frac{g(y)}{g(x)} \leq h(x) \quad \text{or} \quad g(x+y) \in (1 \pm \epsilon)g(y).$$

$g(x)$	lower bound	fails
$1/x$	$\Omega(n)$	slow-dropping
x^3	$\Omega(n^{1/3})$	slow-jumping
$g(x) = (2 + \sin \frac{\pi}{2}x)x^2$	$\Omega(n)$	predictability

Why are they sufficient?

slow-jumping + slow-dropping : heavy hitters for $\sum g(f_i)$ are heavy hitters for $\sum f_i^2$.

\+ predictability : $\hat{f}_i \approx f_i$ yields $g(\hat{f}_i) \approx g(f_i)$.

Count Sketch & Recursive Subsampling does the job

Theorem

There exists a streaming algorithm that given $\epsilon = 1/\widetilde{O}(1)$, $m = \text{poly}(n)$, *and* g *which is slow-dropping, slow-jumping, and predictable*

1. *outputs a* $(1 \pm \epsilon)$*-approximation to* $\sum_i g(f_i)$ *with probability at least* $2/3$ *and*
2. *uses* $\text{poly}(\epsilon^{-1} \log n)$ *bits of storage.*

Why are they sufficient?

slow-jumping + slow-dropping : heavy hitters for $\sum g(f_i)$ are heavy hitters for $\sum f_i^2$.

\+ predictability : $\hat{f}_i \approx f_i$ yields $g(\hat{f}_i) \approx g(f_i)$.

Count Sketch & Recursive Subsampling does the job

Theorem

There exists a streaming algorithm that given $\epsilon = 1/\widetilde{O}(1)$, $m = \text{poly}(n)$, *and* g *which is slow-dropping, slow-jumping, and predictable*

1. *outputs a* $(1 \pm \epsilon)$*-approximation to* $\sum_i g(f_i)$ *with probability at least* $2/3$ *and*
2. *uses* $\text{poly}(\epsilon^{-1} \log n)$ *bits of storage.*

The converse is true for all but a small class of functions!