One Sketch for All

Joel A. Tropp

Department of Mathematics
The University of Michigan
jtropp@umich.edu

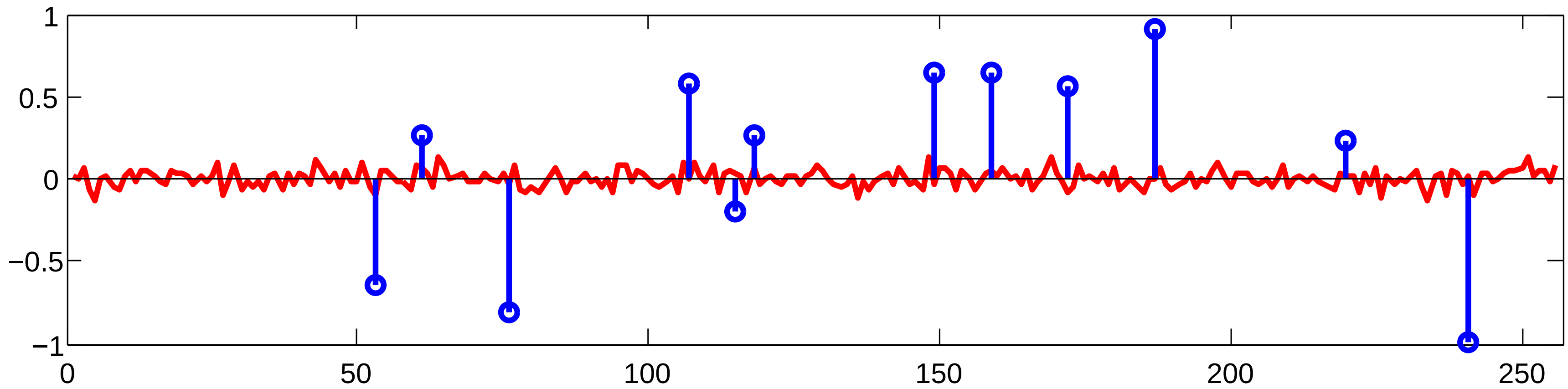
Joint work with
Anna C. Gilbert
Martin J. Strauss
Roman Vershynin

Research supported in part by NSF and DARPA

or, Heavy Hitters on Steroids*

*Allegedly

The Heavy Hitters Problem



Data: A signal $\boldsymbol{s}$ with d real entries

Query: Find locations and magnitudes of m largest entries

- Interesting case: d is massive and m is big
- Easy if signal is explicit (aggregate / one pass model)
- Challenging in streaming data model

Streaming Data Model

- Think of components of $\boldsymbol{s}$ as items in WalMart inventory

- Cash register records a sequence of additive updates, e.g.,

  . . . **Beer** $+3$ **Diapers** -1 **Ammo** $+50$ **Beer** $+2$ **. . .**

- Total sales are implicitly determined by the sum of updates

- **Query:** What items were sold or returned most?

Reference: Muthukrishnan 2003

Consequences of Streaming Model

- Must be able to process updates quickly
- Linear processing useful for signed additive updates

$$\mathbf{\Phi}(\boldsymbol{s} + \boldsymbol{u}) = \mathbf{\Phi}\boldsymbol{s} + \mathbf{\Phi}\boldsymbol{u}$$

- The signal evolves, so the heavy hitters evolve
- Must respond correctly to a query at any time

Sublinearity in Dimension

- Since d is massive, want to limit resource usage to $\mathrm{polylog}(d)$
  - Storage
  - Computation time
  - Randomness
- Locations and magnitudes of m heavy hitters take about

  $m \log(d/m)$ bits of storage

Moral: Heavy Hitters is possible with sublinear resources

Sketching

- A *synopsis data structure* maintains a small sketch of the data
- In many cases, sketch is a random linear projection
- Sketch supports two operations:
  - **Update** revises the sketch to reflect a change in the data
  - **Query** returns an estimate of a data statistic
- For Heavy Hitters,
  - Update supports signed additive changes to one signal component
  - Query returns m signal positions and approximate values

Reference: Gibbons–Matias 1998

One Sketch for All

- Many randomized sketches offer guarantees of the form

  On each signal, with high probability, the query succeeds

- May be too weak if
  - Many queries are made or
  - Updates are adaptive, adversarial, worst-case, etc.

- Better to have a guarantee of the form

  With high probability, on all signals, the query succeeds

- This criterion has not appeared in data stream literature, but see Candès et al. 2004 and Donoho 2004

Desiderata for Heavy Hitters

Want a synopsis data structure with these properties:

1. **Uniformity:** Sketch works for *all signals simultaneously*
2. **Optimal Size:** Sketch uses $m \operatorname{polylog}(d)$ storage
3. **Optimal Speed:** Update and query times are $m \operatorname{polylog}(d)$
4. **High Quality:** Answer to query has *near-optimal error*

Algorithm 1: Chaining Pursuit

- **Uniform:** $\textsc{Yes}$
- **Storage:** $O(m \log^2 d)$
- **Update time:** Amortized $m^{o(1)} \operatorname{polylog}(d)$
- **Query time:** $m^{1+o(1)} \operatorname{polylog}(d)$
- **Error bounds:**

$$\|\boldsymbol{s} - \widehat{\boldsymbol{s}}\|_1 \leq \mathrm{C} \log m \, \|\boldsymbol{s} - \boldsymbol{s}_m\|_1$$

$$\|\boldsymbol{s} - \widehat{\boldsymbol{s}}\|_{\text{weak-1}} \leq \mathrm{C} \, \|\boldsymbol{s} - \boldsymbol{s}_m\|_1$$

Algorithm 2: HHS Pursuit

- **Uniform:** Yes
- **Storage:** $m \operatorname{polylog}(d)/\varepsilon^2$
- **Update time:** $m \operatorname{polylog}(d)/\varepsilon^2$
- **Query time:** $m^2 \operatorname{polylog}(d)/\varepsilon^4$
- **Error bounds:**

$$\|\boldsymbol{s} - \widehat{\boldsymbol{s}}\|_1 \leq (1+\varepsilon)\,\|\boldsymbol{s} - \boldsymbol{s}_m\|_1$$

$$\|\boldsymbol{s} - \widehat{\boldsymbol{s}}\|_2 \leq \|\boldsymbol{s} - \boldsymbol{s}_m\|_2 + \frac{\varepsilon}{\sqrt{m}}\,\|\boldsymbol{s} - \boldsymbol{s}_m\|_1$$

Compressible Signals

- Results nontrivial for *compressible signals*:

$$\left|s_{(k)}\right| \leq \mathrm{C}k^{-\alpha} \qquad \text{for } \alpha \geq 1$$

- Tail behavior for $\alpha < 1$:

$$\left\|\boldsymbol{s} - \boldsymbol{s}_m\right\|_1 \asymp m^{1-\alpha}$$

$$\left\|\boldsymbol{s} - \boldsymbol{s}_m\right\|_2 \asymp m^{1/2-\alpha}$$

- Compressible signals are extremely common

Related Work

Reference	Uniform	Opt. Storage	Sublin. Query
GMS	X	✓	✓
CM	✓	X	✓
CRT, Don	✓	✓	X
Chaining	✓	✓	✓
HHS	✓	✓	✓

Remark: The numerous contributions in this area are not strictly comparable.

References: Gilbert et al. 2002, 2005; Cormode–Muthukrishnan 2005; Candès–Romberg–Tao 2004, Donoho 2004, . . .

Dimension Reduction for Sparse Vectors

- Let $X \subset \ell_1^d$ be the set of all m-sparse signals
- The Chaining sketch embeds X in ℓ_1 with dimension $O(m \log^2(d))$
- The embedding is bi-Lipshitz with polylogarithmic distortion
- Chaining algorithm allows sublinear-time reconstruction of sparse signals from their sketches
- Tolerant to noise in signal and in sketch
- Log error may be connected with lower bounds [Charikar–Sahai 2002]

Contributions

- Ask new questions:
  1. Is a uniform guarantee possible?
  2. What is the best error bound?

- New technical ideas:
  1. Restricted isometries
  2. Operator norm bounds

- Careful analysis:
  1. Detailed results on random matrices
  2. Understanding and controlling noise propagation

Overall Structure of Algorithms

1. **Identify** candidate heavy hitters

2. **Estimate** their magnitudes

3. **Cull** the herd

4. **Update** the sketch

5. **Iterate** the procedure

Different Intuitions

Chaining Algorithm

- Finds a constant proportion of the *heavy hitters* at each iteration
- Requires careful culling of candidate heavy hitters
- Careful analysis of "internal noise"

HHS Algorithm

- Finds a constant proportion of the *signal energy* at each iteration
- Must identify heavy hitters near noise level to find signal energy
- Careful analysis of batch estimation procedure

Locating a Heavy Hitter

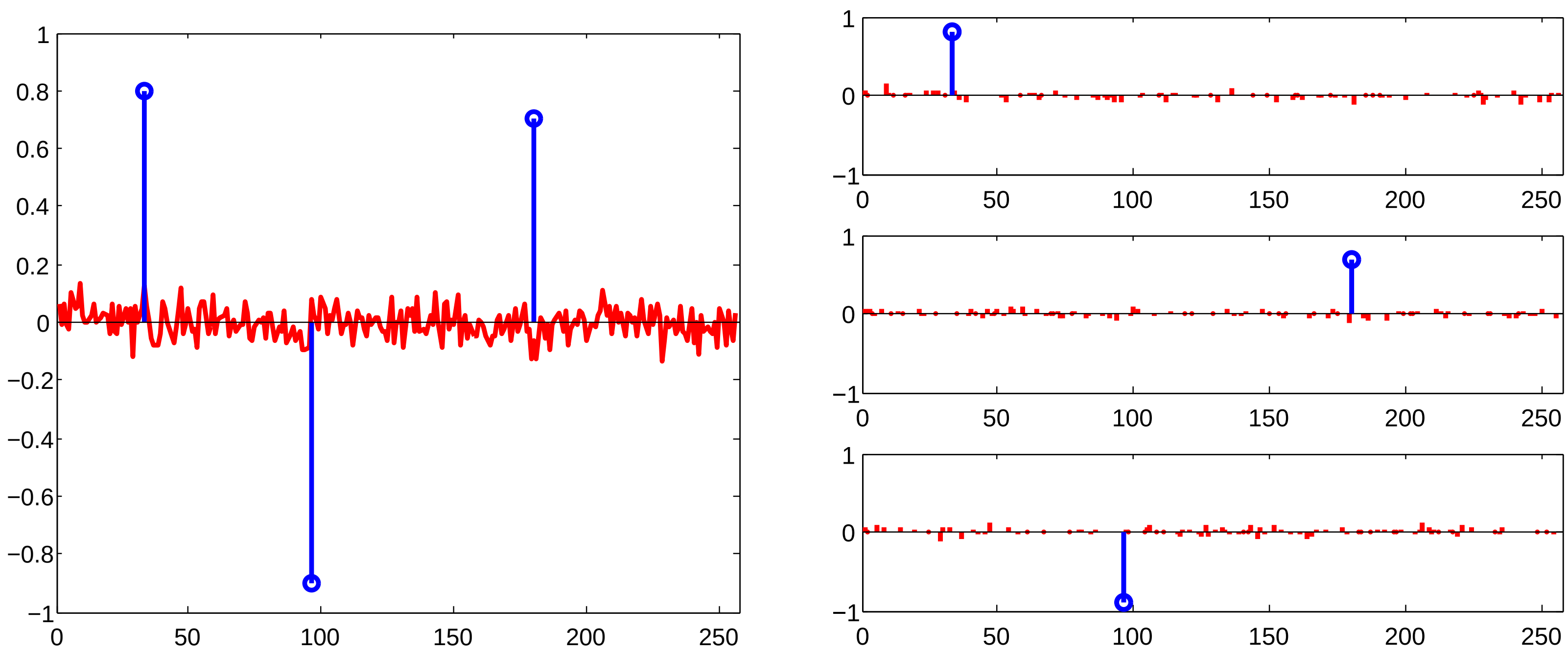
- Suppose the signal contains one "spike" and no noise
- $\log_2 d$ bit tests will identify its location, e.g.,

$$\boldsymbol{B}_1 \boldsymbol{s} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{array}{l} \text{MSB} \\ \\ \text{LSB} \end{array}$$

bit-test matrix $\cdot$ signal $=$ location in binary

Isolating Heavy Hitters

- To use bit tests, the measurements need to isolate many spikes
- Assign each of d signal positions at random to one of $O(m)$ different subsets
- Repeat to drive down failure probability



The Sketches

Chaining:

- Multiple trials of isolation + bit tests

HHS:

- Multiple trials of isolation + noise reduction + bit tests
- Separate sketch for estimation

Estimation for HHS

- Maintain separate sketch $\boldsymbol{v}$ to estimate size of candidates:

$$\boldsymbol{v} = \boldsymbol{P}\mathscr{F}\boldsymbol{s}$$

  where $\boldsymbol{P}$ is a random projection to $m \operatorname{polylog}(d)/\varepsilon^2$ coordinates, and $\mathscr{F}$ is the DFT

- Given list L of candidates, estimate magnitudes with LS:

$$\widehat{\boldsymbol{s}}_L = (\boldsymbol{P}\mathscr{F}_L)^\dagger \boldsymbol{v}$$

- Error estimate via new norm bound for restricted isometries

$$\|\boldsymbol{P}\mathscr{F}\boldsymbol{x}\|_2 \leq c\left(\|\boldsymbol{x}\|_2 + \frac{1}{\sqrt{m}}\|\boldsymbol{x}\|_1\right)$$

Chaining Algorithm

```
Inputs:  Number of spikes m, sketches, random projectors
Output:  A list of m spike locations and values

For each of O(log m) passes:
    For each trial:
        For each measurement:
            Use bit tests to identify the spike position
            Use a bit test to estimate the spike magnitude

        Retain m/2^k distinct spikes with largest values

    Retain spike positions that appear in most trials
    Estimate final spike magnitudes using medians

    Encode the spikes using the projection operator
    Subtract the encoded spikes from the sketch

Prune output to largest m spikes
```

HHS Algorithm

```
Inputs:  Number of spikes m, sketches, random projectors
Output:  A list of m spike locations and values

Run Chaining Pursuit to get first signal estimate

For each of O(log m) passes:
     For each measurement:
         Use bit tests to identify a spike position

     Retain spikes that appear frequently

     Use LS to estimate magnitudes of new candidate spikes

     Retain largest O(m) spikes identified to date
     Encode the spikes using the projection operators
     Subtract the encoded spikes from the original sketch

Prune output to largest m spikes
```

To learn more...

Web: `http://www.umich.edu/~jtropp`

E-mail: `jtropp@umich.edu`

- Matlab code for Chaining Pursuit* is freely available!
- GSTV, "Sublinear approximation of compressible signals," SPIE IIM, April 2006
- —, "Algorithmic dimension reduction in the ℓ_1 norm for sparse vectors," submitted April 2006
- HHS Pursuit still in preparation...