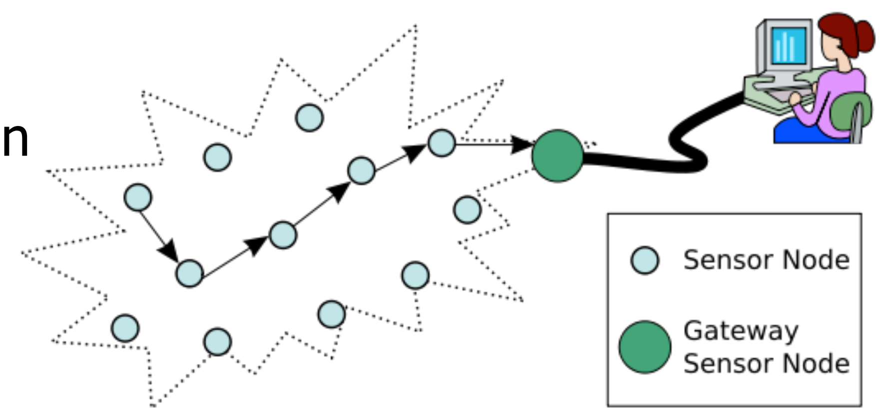
Sketching, streaming, and sub-linear space algorithms

Piotr Indyk
MIT
(currently at Rice U)

Data Streams

- A data stream is a sequence of data that is too large to be stored in available memory
- Examples:
  - Network traffic
  - Sensor networks
  - Approximate query optimization and answering in large databases
  - Scientific data streams
  - …and this talk



Outline

- Data stream model, motivations, connections
- Streaming problems and techniques
  - Estimating number of distinct elements in a stream
  - Other quantities: Norms, moments, heavy hitters…
  - What else ? Geometry, graphs, text,...
- Streaming and sparse approximations
  - Connections (compressive sensing, coding theory)
  - New developments
- 1-2 proofs, ~~3~~ 2 open problems ?

Basic Data Stream Model

- Single* pass over the input data: $i_1, i_2, \ldots, i_N$

8 2 1 9 1 9 2 4 6 3 9 4 2 3 4 2 3 8 5 2 5 6 ...

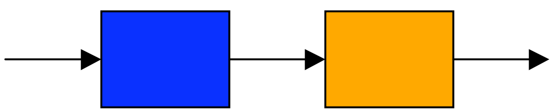
- Bounded storage (typically N^{α} or $\log^{c} N$)
  - Units of storage: bits, numbers or ”elements”
- Fast processing time

*Small number of passes interesting as well

Comments

- Almost all algorithms are approximate
- We assume worst-case input stream
  - Adversaries do exist

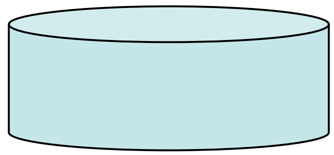
  - General algorithms
  - Modular composition



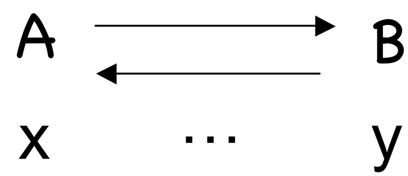
- Randomized algorithms OK (often necessary)
  - Randomness in the algorithm, not the input

Connections

8 2 1 9 1 9 2 4 6 3 9 4 2 3 4 2 3 8 5 2 5 6 ...



- External memory algorithms
  - Linear scan works best
- Communication complexity
  - Low-space algorithms yield low-communication protocols



- Metric embeddings, sub-linear time algorithms, pseudo-randomness, compressive sensing, error-correcting codes…

Part 1:

Streaming problems and techniques

Counting Distinct Elements

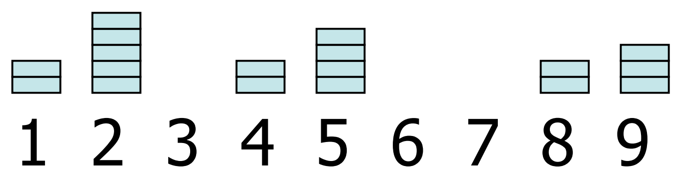
[Flajolet-Martin'85] (a version)

- Stream elements: numbers from $\{1...n\}$
- Goal: estimate the number of distinct elements DE in the stream
  - Up to $1\pm\varepsilon$
  - With probability $1-P$
- Simpler (gap) problem: for a given $T>0$, provide an algorithm which, with probability $1-P$:
  - Answers YES, if $DE > (1+\varepsilon)T$
  - Answers NO, if $DE < (1-\varepsilon)T$
- Reduction to gap problem problem:
  - Run $\log_{1+\varepsilon} n$ copies of the algorithm in parallel, with
    $T=1, 1+\varepsilon, (1+\varepsilon)^2,..., n$
  - Total space multiplied by $\log_{1+\varepsilon} n \approx \log(n)/\varepsilon$
  - Probability of failure multiplied by the same factor

Vector Interpretation

Stream: 8 2 1 9 1 9 2 4 4 9 4 2 5 4 2 5 8 5 2 5

Vector x:



- Initially, $x=0$
- Insertion of i is interpreted as

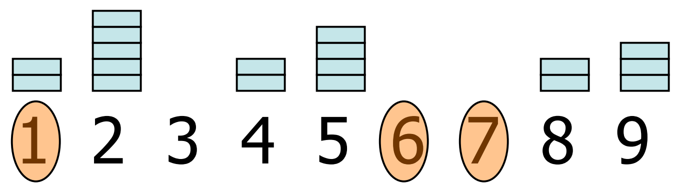
$$x_i = x_i + 1$$

- Want to estimate the number of non-zero entries in x (a.k.a. $||x||_0$)
- …without storing all coordinates

Is $DE>(1+\varepsilon)T$?

Vector x:

Set S:



- First attempt:
  - Choose a random set S of coordinates
    - For each i, we have $\Pr[i \in S]=1/T$
  - Maintain $Sum_S(x) = \Sigma_{i \in S}\, x_i$
    - YES, if $Sum_S(x)>0$
    - NO, if $Sum_S(x)=0$
- Analysis:
  - $\Pr[Sum_S(x)=0] = (1-1/T)^{DE}$
  - For T "large enough": $(1-1/T)^{DE} \approx e^{-DE/T}$
  - Using calculus, for ε small enough:
    - If $DE> (1+\varepsilon)T$, then $\Pr < e^{-(1+\varepsilon)} < 1/e - \varepsilon/3$
    - if $DE< (1-\varepsilon)T$, then $\Pr > e^{-(1-\varepsilon)} > 1/e + \varepsilon/3$

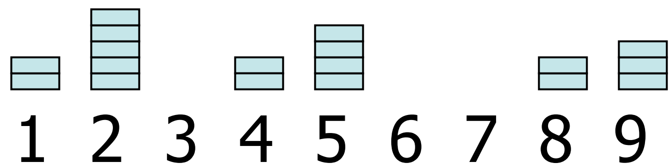
Is $DE>(1+\varepsilon)T$? (ctd)

- From our first attempt:
  - If $DE> (1+\varepsilon)T$, then $Pr[Sum_S(x)=0] < 1/e-\varepsilon/3$
  - if $DE< (1-\varepsilon)T$, then $Pr[Sum_S(x)=0] > 1/e+\varepsilon/3$
- Second attempt:
  - Select sets $S_1 \ldots S_k$, $k=O(\log(1/P)/\varepsilon^2)$
  - Let Z = number of $Sum_{Sj}(x)$ that are equal to 0
  - By Chernoff bound, with probability $>1-P$
    - If $DE> (1+\varepsilon)T$, then $Z<k/e$
    - If $DE< (1-\varepsilon)T$, then $Z>k/e$

- Total space: $O(\log(n)/\varepsilon \;\; \log (1/P)/\varepsilon^2)$ numbers in range $0\ldots N$
  - Can reduce to $O(\log (1/P)/\varepsilon^2)$ numbers
  - The $1/\varepsilon^2$ term essentially tight for single pass [Woodruff'04, Nelson-Woodruff'08]
  - …~~but not known~~ for two or more passes the bound holds as well [Brody-Chakrabarti'09, Patrascu'09]

Comments

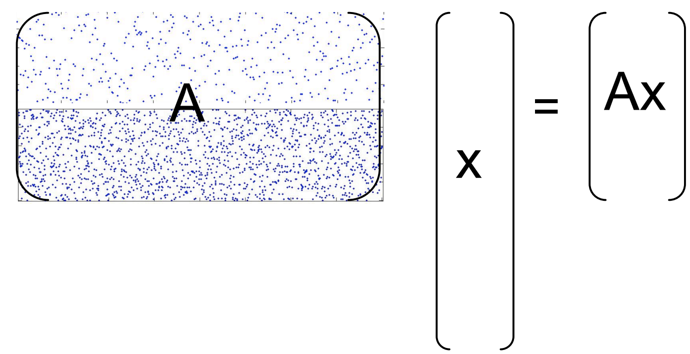
Vector x:



- Linearity:
  - The algorithm uses linear sketches

    $$Sum_{Sj}(x)=\Sigma_{i\in Sj}\, x_i$$

  - I.e., the algorithm maintains a vector Ax where A is a sparse 0-1 matrix of varying density
  - Can implement decrements $x_i=x_i-1$
    - I.e., the stream can contain deletions of elements (as long as $x\geq 0$)
  - Dynamic/turnstile model
  - In fact, can estimate $||x||_0$ for general x
- Pseudorandomness
  - Can use pseudorandom generators instead of storing A explicitly



Generalizations

- What other functions of a vector x can we maintain in small space ?
- L_p norms:

$$||x||_p = (\sum_i |x_i|^p)^{1/p}$$

  ($||x||_p^p$ also referred to as the "p-th frequency moment")
- How much space do you need to estimate $||x||_p$ (for const. ε) ?
- Theorem:
  - For $p \in [0,2]$: polylog n space suffices
  - For $p>2$: $n^{1-2/p}$ polylog n space suffices and is necessary

  [Alon-Matias-Szegedy'96, Feigenbaum-Kannan-Strauss-Viswanathan'99, Indyk'00, Coppersmith-Kumar'04, Ganguly'04, Bar-Yossef-Jayram-Kumar-Sivakumar'02'03, Saks-Sun'03, Indyk-Woodruff'05]

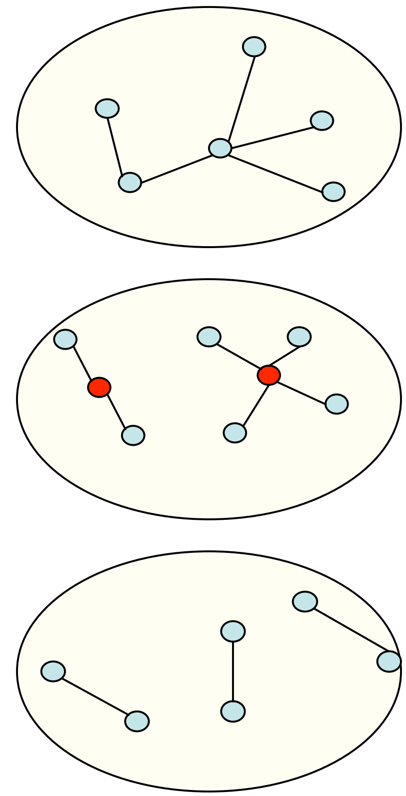
- Matrices:
  - $p=0$: sparse binary
  - $p=2$: Bernoulli, Gaussian
  - $p \in (0,2]$: p-stable distributions
  - $p>2$: sparse Bernoulli

What else?

- Mixed norms, e.g., L_2 of L_0 [Cormode-Muthukrishnan'05]
- Heavy hitters (a.k.a. elephants) [Misra-Gries'82, …, Charikar-Chen-FarachColton'02, Estan-Varghese'03, Cormode-Muthukrishnan'04,'05, Cormode-Hadjieleftheriou'07,…]
  – Coordinates i such that $|x_i|$ is "large"
  – Estimates $x^*_i = x_i \pm Err(x)$
- Entropy [DoBa-Chakrabarti-Muthukrishnan'05, Guha-McGregor-Venkatasubramanian'05, Chakrabarti-Cormode-McGregor'06, Bhuvanagiri-Ganguly'06, Harvey-Nelson-Onak'08]
- Independence testing [Indyk-McGregor'08]
- Median, quantiles, histograms [Munro-Paterson'80, Manku-Rajagopalan-Lindsay'98,'99, Greenwald-Khanna'02, Gilbert-Guha-Indyk-Kotidis-Muthukrishnan-Strauss'02,…]
- …

What else ?

- Geometric problems:
  - Stream of points
  - Minimum spanning tree (cost) [Indyk'04, Frahling-Indyk-Sohler'05]
    - polylog N space, constant approx
  - Partitioning into k clusters [HarPeled'04, Indyk'04, Frahling-Sohler'05, …]
    - poly(k+log N) space, constant approx
  - Matching (cost) [Indyk'04]
    - polylog N space, log N approx
- Metric problems
- Graph problems
- Text problems



Part 2: Connections

(compressive sensing, coding theory)

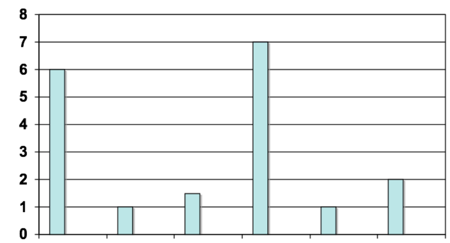
Heavy Hitters: Sparse Approximation View

- Heavy hitters / estimation:
  - Given: a sketch Ax where A is an $m \times n$ matrix, $m << n$
  - Want: estimate $x^*_i = x_i \pm Err(x)$
- Sparse recovery, compressive sensing [Candes-Romberg-Tao'04, Donoho'04,…]
  - Given: a "measurement vector" Ax
  - Want: an "approximation" x^* of x s.t.

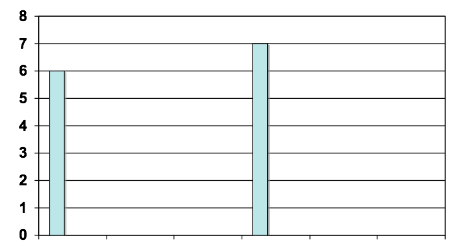
    $$||x^*-x||_p \leq C(k)\ ||x'-x||_q \quad (\ l_p/l_q \text{ guarantee })$$

    over all x' that are k-sparse (at most k non-zero entries)
- Connection:
  - The best k-sparse approximation x^* contains k coordinates of x with the largest abs value
  - Some of the heavy-hitter algorithms can be interpreted in sparse recovery framework (and vice versa)
- Differences: focus (physical devices vs. computer systems), algorithms (linear programming vs. local estimation), results (deterministic vs. randomized), matrices (dense vs. sparse)

x



x*



k=2

Scale: Excellent | Very Good | Good | Fair

Result Table

Paper	Rand. / Det.	Sketch length	Encode time	Col. sparsity/ Update time	Recovery time	Apprx
[CCF'02], [CM'06]	R	k log n	n log n	log n	n log n	l2 / l2
	R	k $\log^c$ n	n $\log^c$ n	$\log^c$ n	k $\log^c$ n	l2 / l2
[CM'04]	R	k log n	n log n	log n	n log n	l1 / l1
	R	k $\log^c$ n	n $\log^c$ n	$\log^c$ n	k $\log^c$ n	l1 / l1
[CRT'04] [RV'05]	D	k log(n/k)	nk log(n/k)	k log(n/k)	n^c	l2 / l1
	D	k $\log^c$ n	n log n	k $\log^c$ n	n^c	l2 / l1
[GSTV'06] [GSTV'07]	D	k $\log^c$ n	n $\log^c$ n	$\log^c$ n	k $\log^c$ n	l1 / l1
	D	k $\log^c$ n	n $\log^c$ n	k $\log^c$ n	$k^2 \log^c$ n	l2 / l1
[BGIKS'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n^c	l1 / l1
[GLR'08]	D	k $\log n^{\log\log\log n}$	kn^{1-a}	n^{1-a}	n^c	l2 / l1
[NV'07], [DM'08], [NT'08,BD'08]	D	k log(n/k)	nk log(n/k)	k log(n/k)	nk log(n/k) * T	l2 / l1
	D	k $\log^c$ n	n log n	k $\log^c$ n	n log n * T	l2 / l1
[IR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k)	l1 / l1
[BIR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k) *T	l1 / l1
[CDD'07]	D	Ω(n)				l2 / l2

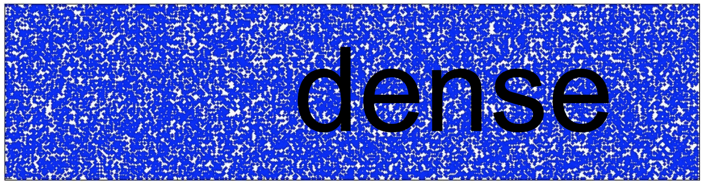
Legend:

- n=dimension of x
- m=dimension of Ax
- k=sparsity of x*
- T = #iterations

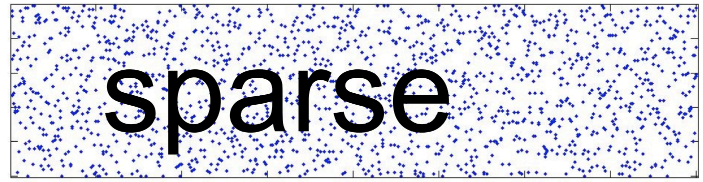
Approx guarantee:

- l2/l2: $\|x-x^*\|_2 \le C\|x-x'\|_2$
- l2/l1: $\|x-x^*\|_2 \le C\|x-x'\|_1/k^{1/2}$
- l1/l1: $\|x-x^*\|_1 \le C\|x-x'\|_1$

Caveats: (1) only results for general vectors x are shown; (2) all bounds up to O() factors; (3) specific matrix type often matters (Fourier, sparse, etc); (4) ignore universality, explicitness, etc (5) most "dominated" algorithms not shown;



vs.



- Restricted Isometry Property (RIP) - key property of a dense matrix A:

$$x \text{ is } k\text{-sparse} \Rightarrow \|x\|_2 \leq \|Ax\|_2 \leq C\,\|x\|_2$$

- Holds w.h.p. for:
  - Random Gaussian/Bernoulli: $m=O(k \log(n/k))$
  - Random Fourier: $m=O(k \log^{O(1)} n)$
- Consider random $m \times n$ 0-1 matrices with d ones per column
- Do they satisfy RIP ?
  - No, unless $m=\Omega(k^2)$ [Chandar'07]
- However, they can satisfy the following RIP-1 property [Berinde-Gilbert-Indyk-Karloff-Strauss'08]:

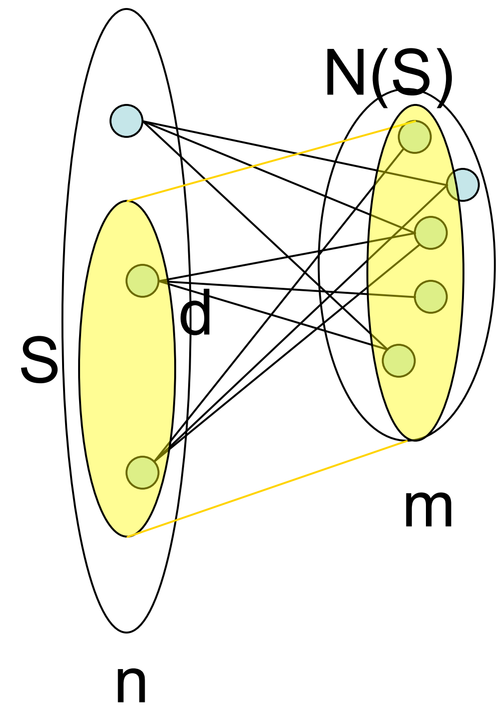
$$x \text{ is } k\text{-sparse} \Rightarrow d(1-\varepsilon)\,\|x\|_1 \leq \|Ax\|_1 \leq d\|x\|_1$$

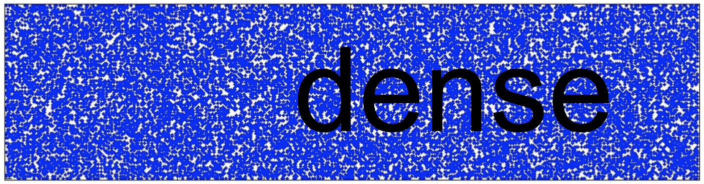
- Sufficient (and necessary) condition: the underlying graph is a $(k, d(1-\varepsilon/2))$-expander

Expanders

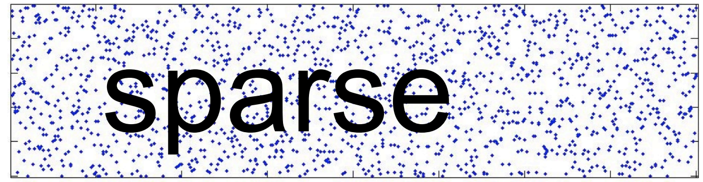
- A bipartite graph is a $(k,d(1-\varepsilon))$-expander if for any left set S, $|S|\le k$, we have $|N(S)|\ge(1-\varepsilon)d\,|S|$
- Constructions:
  - Randomized: $m=O(k \log (n/k))$
  - Explicit: $m=k$ quasipolylog n
- Plenty of applications in computer science, coding theory etc.
- In particular, LDPC-like techniques yield good algorithms for exactly k-sparse vectors x

[Xu-Hassibi'07, Indyk'08, Jafarpour-Xu-Hassibi-Calderbank'08]





vs.



- Instead of RIP in the L_2 norm, we have RIP in the L_1 norm
- Suffices for these results:

Paper	Rand. / Det.	Sketch length	Encode time	Sparsity/ Update time	Recovery time	Apprx
[BGIKS'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n^c	l1 / l1
[IR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k)	l1 / l1
[BIR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k) *T	l1 / l1

- Main drawback: l1/l1 guarantee
- Better guarantees with same time and sketch length ?

Other sparse matrix schemes, for (almost) k-sparse vectors:

- LDPC-like: [Xu-Hassibi'07, Indyk'08, Jafarpour-Xu-Hassibi-Calderbank'08]
- L1 minimization: [Wang-Wainwright-Ramchandran'08]
- Message passing: [Sarvotham-Baron-Baraniuk'06,'08, Lu-Montanari-Prabhakar'08]

Proof: $d(1-\varepsilon/2)$-expansion $\Rightarrow$ RIP-1

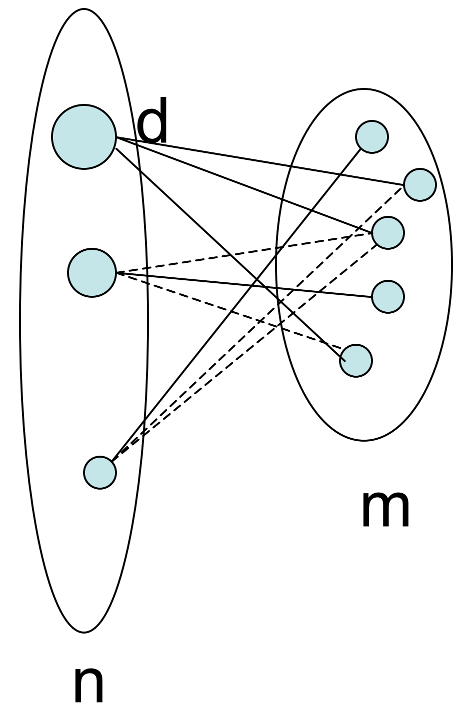
- Want to show that for any k-sparse x we have

$$d(1-\varepsilon)\,\|x\|_1 \le \|Ax\|_1 \le d\|x\|_1$$

- RHS inequality holds for ***any*** x
- LHS inequality:
  - W.l.o.g. assume

$$|x_1| \ge \ldots \ge |x_k| \ge |x_{k+1}| = \ldots = |x_n| = 0$$

  - Consider the edges $e=(i,j)$ in a lexicographic order
  - For each edge $e=(i,j)$ define $r(e)$ s.t.
    - $r(e)=-1$ if there exists an edge $(i',j)<(i,j)$
    - $r(e)=1$ if there is no such edge
- Claim: $\|Ax\|_1 \ge \sum_{e=(i,j)} |x_i| r_e$



Proof: $d(1-\varepsilon/2)$-expansion $\Rightarrow$ RIP-1 (ctd)

- Need to lower-bound

$$\Sigma_e z_e r_e$$

where $z_{(i,j)}=|x_i|$

- Let R_b= the sequence of the first bd r_e's
- From graph expansion, R_b contains at most $\varepsilon/2$ bd -1's

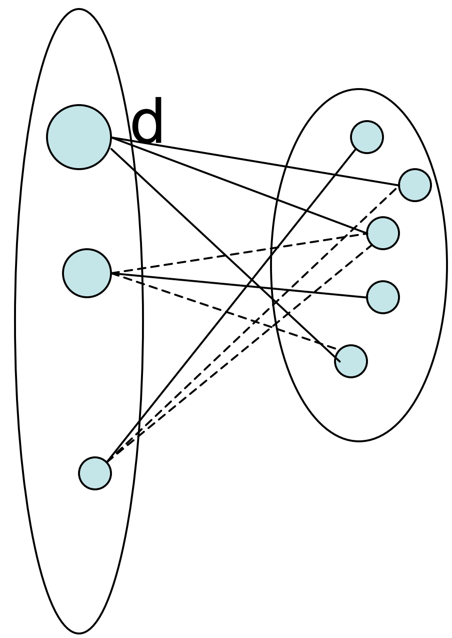
(for $b=1$, it contains no -1's)

- The sequence of r_e's that minimizes $\Sigma_e z_e r_e$ is

$$\underbrace{1,1,\ldots,1}_{d},\underbrace{-1,\ldots,-1}_{\varepsilon/2\ d},\underbrace{1,\ldots,1,}_{(1-\varepsilon/2)d}\ \ldots$$

- Thus

$$\Sigma_e z_e r_e \geq (1-\varepsilon)\Sigma_e z_e = (1-\varepsilon)\ d\|x\|_1$$



Conclusions

- Streaming algorithms in 80 minutes
  - Model, problems, techniques, open problems
- For more:
  - "Streaming…" course notes (my web site)
  - Survey by S. Muthukrishnan
- Also:
  - Streaming session Thu 10:45 am
  - Other talks

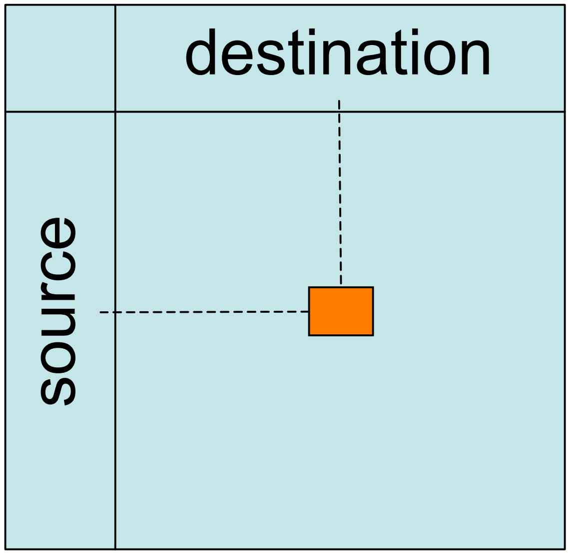
Thanks

… to Anna Gilbert and Volkan Cevher for many useful comments on this tutorial

Appendix

Example application: Monitoring Network Traffic

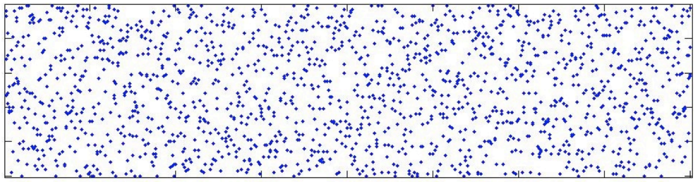
- Router routs packets (many packets)
  - Where do they come from ?
  - Where do they go to ?
- Ideally, would like to maintain a traffic matrix x[.,.]
  - For each (src,dst) packet, increment $x_{src,dst}$
  - Requires way too much space! (2^{32} x 2^{32} entries)
  - Need to maintain a compressed version of the matrix

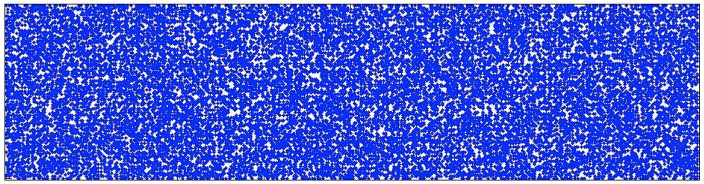


x

General approach

- Choose encoding matrix A at random
  - Sparse matrices:
    - Data stream algorithms
    - Coding theory (LDPCs)
  - Dense matrices:
    - Compressed sensing
    - Complexity theory (Fourier)





- Tradeoffs:
  - Sparse: computationally more efficient, explicit
  - Dense: shorter sketches
- Best of both worlds ?

A satisfies RIP-1 $\Rightarrow$ Sparse Matching Pursuit works

[Berinde-Indyk-Ruzic'08]

- Algorithm:
  - $x^*=0$
  - Repeat T times
    - Compute $c=Ax-Ax^* = A(x-x^*)$
    - Compute Δ such that Δ_i is the median of its neighbors in c
    - Sparsify Δ

      (set all but $2k$ largest entries of Δ to 0)
    - $x^*=x^*+\Delta$
    - Sparsify x^*

      (set all but k largest entries of x^* to 0)
- After $T=\log()$ steps we have

$$\|x-x^*\|_1 \leq c\ \mathrm{Err}^k_1$$