Tutorial on Compressed Sensing (or Compressive Sampling, or Linear Sketching)

Piotr Indyk
MIT

Linear Compression

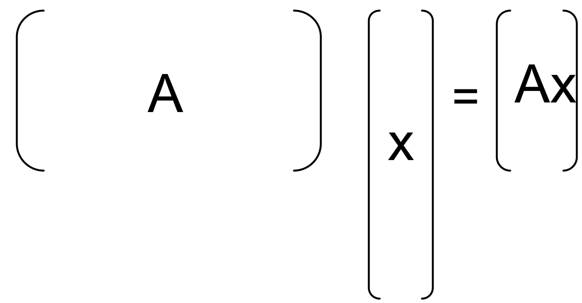
- Setup:
  - Data/signal in n-dimensional space : x

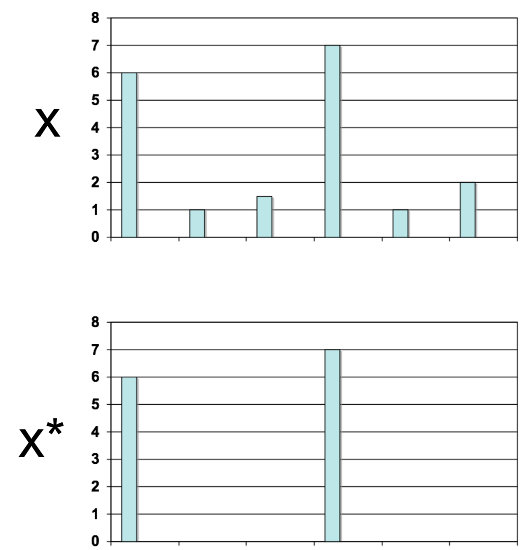
    E.g., x is an 1000x1000 image ⇒ n=1000,000
  - Goal: compress x into a “sketch” Ax ,

    where A is a carefully designed m x n matrix, m << n
- Requirements:
  - Plan A: want to recover x from Ax
    - Impossible: undetermined system of equations
  - Plan B: want to recover an “approximation” x* of x
    - Sparsity parameter k
    - Want x* such that $||x^*-x||_p \le C(k) \min_{x'} ||x'-x||_q$ (l_p/l_q)

      over all x’ that are k-sparse (at most k non-zero entries)
    - The best x* contains k coordinates of x with the largest abs value

      ⇒ if x itself is k-sparse, we have exact recovery: x=x*
- Want:
  - Good compression (small m)
  - Efficient algorithms for encoding and recovery
- Why linear compression ?

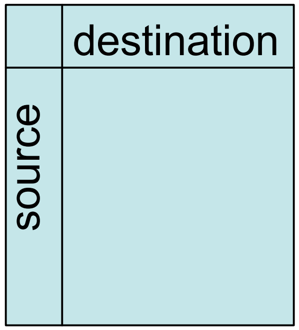




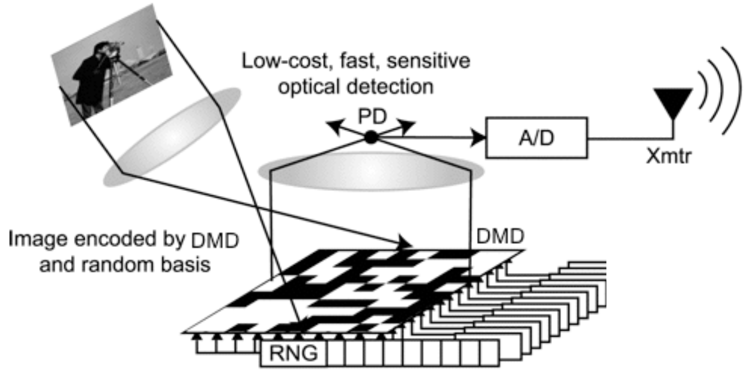
k=2

Applications of Linear Compression

- Streaming algorithms, e.g., for network monitoring
  - Would like to maintain a traffic matrix x[.,.]
    - Given a (src,dst) packet, increment $x_{src,dst}$
  - We can maintain sketch Ax under increments to x, since $A(x+\Delta) = Ax + A\Delta$



- Single pixel camera [Wakin, Laska, Duarte, Baron, Sarvotham, Takhar, Kelly, Baraniuk'06]

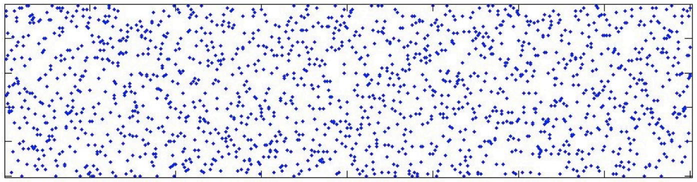


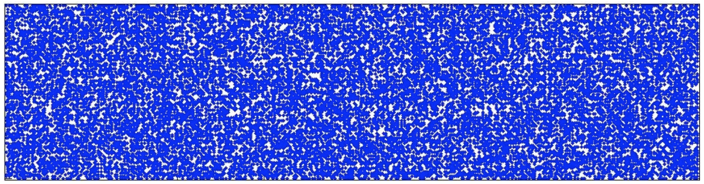
- Pooling microarray experiments

(talk by Anna Gilbert)

Types of matrices A

- 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

Parameters

- Given: dimension n, sparsity k
- Parameters:
  - Sketch length m
  - Time to compute/update Ax
  - Time to recover x^* from Ax
  - Matrix type:
    - Deterministic (one A that works for all x)
    - Randomized (random A that works for a fixed x w.h.p.)
  - Measurement noise, universality, …

Result Table

Paper	Rand. / Det.	Sketch length	Encode time	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,BM'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, BIR'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

Legend:

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

Approx guarantee:

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

Scale: Excellent | Very Good | Good | Fair

Result Table

Paper	Rand. / Det.	Sketch length	Encode time	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,BM'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, BIR'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: $||x-x^*||_2 \leq C||x-x'||_2$
- l1/l1: $||x-x^*||_1 \leq C||x-x'||_1$
- l2/l1: $||x-x^*||_2 \leq C||x-x'||_1/k^{1/2}$

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

Plan

- Classification+intuition:
  - Matrices: sparse / dense
  - Matrix properties that guarantee recovery
  - Recovery algorithms
- Result table (again)
- Sparse Matching Pursuit
- Conclusions

Matrix Properties

- Restricted Isometry Property(RIP) [Candes-Tao]:for all k-sparse vectors x

  $$||x||_2 \leq ||Ax||_2 \leq C\, ||x||_2$$

  – Random Gaussian/Bernoulli: $m=O(k \log (n/k))$
  – Random Fourier: $m=O(k \log^{O(1)} n)$
- k-neighborly polytopes [Donoho-Tanner]: only for exact recovery
- Euclidean sections of l_1 / width property [Kashin,…,Donoho,Kashin-Temlakov]: for all vectors x such that $Ax=0$, we have

  $$||x||_2 \leq C' / m^{1/2}\, ||x||_1$$

  – Random Gaussian/Bernoulli: $C'=C \ln(en/m)^{1/2}$
- RIP-1 property [Berinde-Gilbert-Indyk-Karlof-Strauss]: for all k-sparse vectors x

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

  Holds if (and only if*) A is an adjacency matrix of a $(k, d(1-\varepsilon/2))$-expander with left degree d

  – Randomized: $m=O(k \log (n/k))$; Explicit: $m=k$ quasipolylog n
- Expansion/randomness extraction property of the graph defined by A [Xu-Hassibi, Indyk]: originally for exact recovery

* for binary matrices and ε small enough

Recovery algorithms

- L1 minimization, a.k.a. Basis Pursuit [Donoho],[Candes-Romberg-Tao]:

  minimize $||x^*||_1$
  subject to $Ax^*=Ax$

  – Solvable in polynomial time using using linear programming

- Matching pursuit: OMP, ROMP, StOMP, CoSaMP, EMP, SMP,…
  – Basic outline:
    - Start from $x^*=0$
    - In each iteration
      – Compute an approximation Δ to $x-x^*$ from $A(x-x^*)=Ax-Ax^*$
      – Sparsify Δ , i.e., set all but t largest (in magnitude) coordinates to 0 (t = parameter)
      – $x^*=x^*+\Delta$
  – Many variations

Result Table (with techniques)

Paper	Rand. / Det.	Sketch length	Encode time	Sparsity	Recovery time	Apprx	Matrix property	Algo
[CCF'02], [CM'06]	R	k log n	n log n	log n	n log n	l2 / l2	sparse +1/-1	"one shot MP" *
	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	sparse binary	"one shot MP" *
	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	RIP2	BP
	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	augmented RIP1/RIP2*	MP
	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	RIP1	BP
[GLR'08]	D	$k \log n^{\log\log\log n}$	kn^{1-a}	n^{1-a}	n^c	l2 / l1	l2 sections of l1	BP
[NV'07], [DM'08], [NT'08,BM'08]	D	k log(n/k)	nk log(n/k)	k log(n/k)	nk log(n/k) * T	l2 / l1	RIP2	MP
	D	$k \log^c n$	n log n	$k \log^c n$	n log n * T	l2 / l1		
[IR'08, BIR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k)	l1 / l1	RIP1/ expansion	MP
[BIR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k) *T	l1 / l1		

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

* In retrospective

Sparse Matching Pursuit

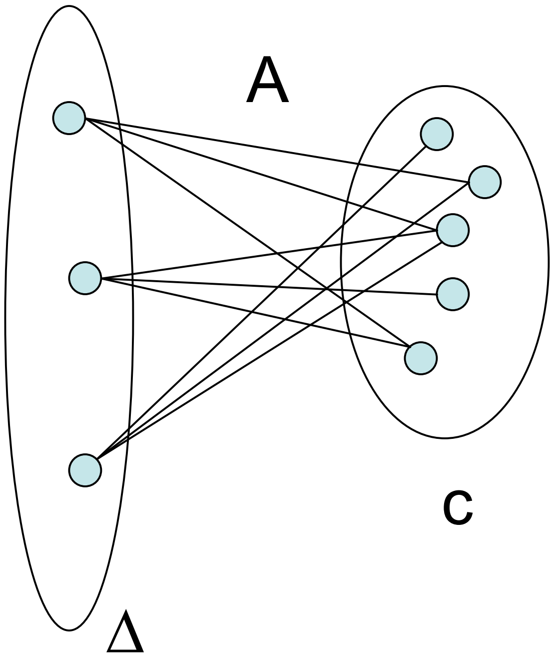
[Berinde-Indyk-Ruzic'08]

- 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

$$||x-x^*||_1 \leq C \min_{k\text{-sparse } x'} ||x-x'||_1$$



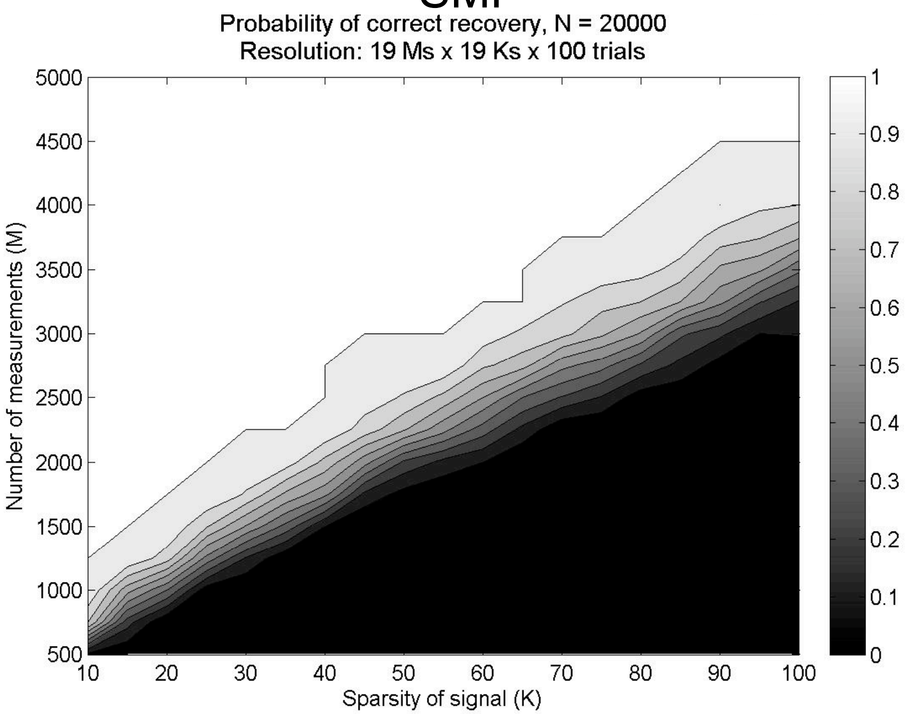
Conclusions

- Sparse approximation using sparse matrices
- State of the art: can do 2 out of 3:
  - Near-linear encoding/decoding
  - O(k log (n/k)) measurements
  - Approximation guarantee with respect to L2/L1 norm
- Open problems:
  - 3 out of 3 ?
  - Explicit constructions ?
    - RIP1: via expanders, quasipolylog m extra factor
    - l2 section of l1: quasipolylog m extra factor [GLR]
    - RIP2: extra factor of k [DeVore]

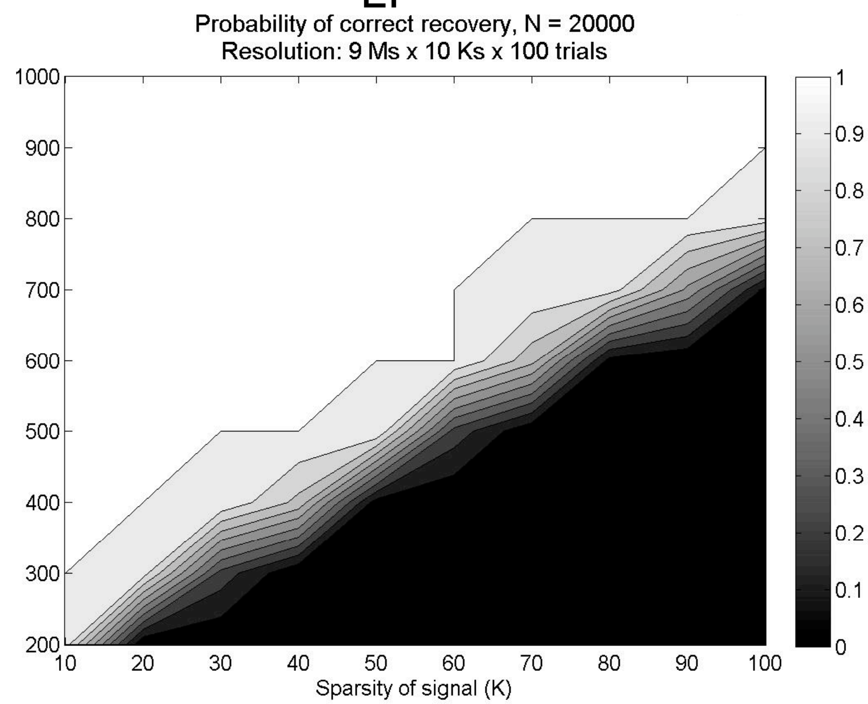
Experiments

- Probability of recovery of random k-sparse +1/-1 signals from m measurements
  - –Sparse matrices with d=10 1s per column
  - –Signal length n=20,000

SMP



LP



Running times