Euclidean Sparse Recovery with Optimal Measurement

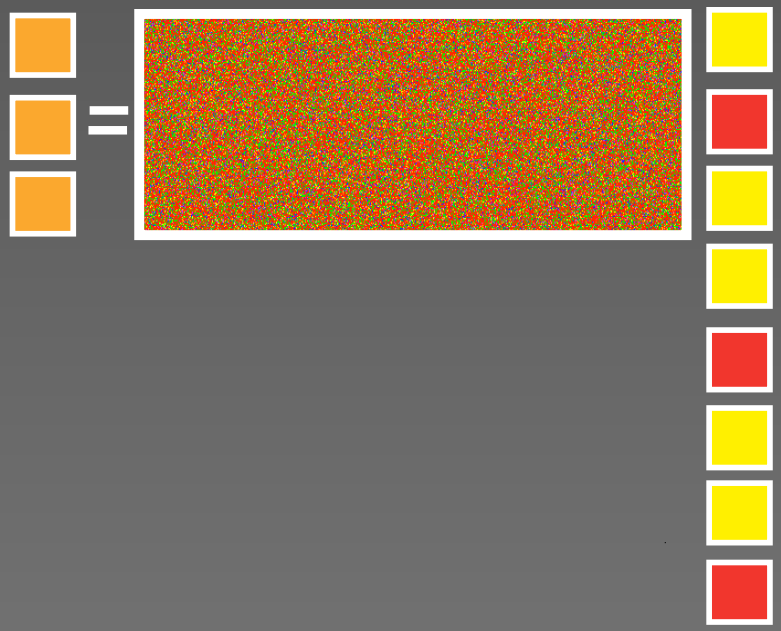
(in progress)

Martin J. Strauss

University of Michigan

Joint work with A. C. Gilbert (Mich.), Y. Li (Mich.), and E. Porat (Bar Ilan)

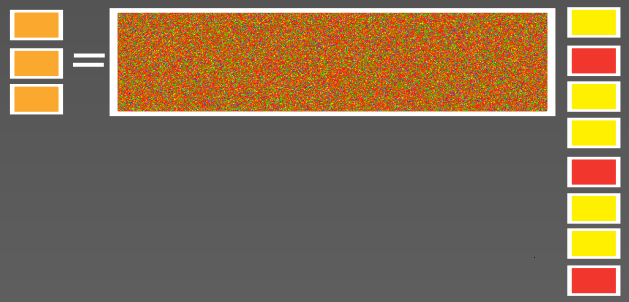
Sparse signal recovery



measurements
length $m = k \log(n)$

k-sparse signal
length n

Problem statement



m as small as possible

Assume x has low complexity: x is k-sparse (with noise)

Construct

- Matrix A: $\mathbb{R}^n \to \mathbb{R}^m$
- Decoding algorithm $\mathcal{D}$

Given Ax for any signal $x \in \mathbb{R}^n$, we can quickly recover $\widehat{x}$ with

$$\|x - \widehat{x}\|_2 \leq (1+\epsilon) \min_{y\ k-sparse} \|x - y\|_2$$

Parameters

- Number of measurements m
- Recovery time
- Approximation guarantee (norms, mixed)
- One matrix vs. distribution over matrices
- Explicit construction
- Universal matrix (for any basis, after measuring)
- Tolerance to measurement noise

	Excellent	Good	Fair	Poor

Paper	Rand/ Det.	Sketch length	Encode time	Update time	Recovery time	Approx.
[CCF'02][CM'06]	R	k log n	n log n	log n	n log n	l_2/l_2
[CM'04]	R	k log n	n log n	log n	n log n	l_1/l_1
[CRT'04] [RV'05]	D	k log(n/k)	nk log(n/k)	k log(n/k)	n^c	l_2/l_1
	D	k $\log^c$ n	n log n	k $\log^c$ n	n^c	l_2/l_1
[BGIKS'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n^c	l_1/l_1
[GLR'08]	D	k log $^{\text{loglog(n)}}$	$kn^{(1-a)}$	$n^{(1-a)}$	n^c	l_2/l_1
[NV'07],[DM'08] [NT'08]	D	k log(n/k)	nk log(n/k)	k log(n/k)	T nk log(n/k)	l_2/l_1
	D	k $\log^c$ n	n log n	k $\log^c$ n	T n log n	l_2/l_1
[IR'08]	D	k log(n/k)	n log(n/k)	log(n/k)	n log(n/k)	l_1/l_1

poly(n) time algorithms

[CM'04]	R	k $\log^c$ n	n $\log^c$ n	$\log^c$ n	k $\log^c$ n	l_1/l_1
[CM'06]	R	k $\log^c$ n	n $\log^c$ n	$\log^c$ n	k $\log^c$ n	l_2/l_2
[GSTV'06] [GSTV'07]	D	k $\log^c$ n	n $\log^c$ n	$\log^c$ n	k $\log^c$ n	l_1/l_1
	D	k $\log^c$ n	n $\log^c$ n	k $\log^c$ n	k^2 $\log^c$ n	l_2/l_1

sublinear time algorithms

THIS TALK	R	k log(n/k)	n log(n/k)loglog(k)	log(n/k)loglog(k)	k $\log^c$(n/k)	l_2/l_2

"For Each L^2 Noise"

Our result:

- There exists a distribution on A and decoding algorithm $\mathcal{D}$ such that
- For each x,
- For most A,

$$\|x - \widehat{x}\|_2 \leq (1+\epsilon) \min_{y\ k-sparse} \|x - y\|_2$$

Roughly:

- Recover "for each" noise vector in L^2 ball.
- Compare to Donoho, Candès-Romberg-Tao: Recover "for all" noise vectors in L^1 ball.

Approximation Schemes

Goal:

- Get within factor $(1+\epsilon)$ in fidelity
- Willing to pay factor $1/\epsilon^3$ in runtime

What about number of measurements? To optimize, or cost of fidelity?

Medical Imaging

Number of measurements $= \Theta$(time patient holds breath)
Extra factor of "$1000 \cdot \log$(something)" unacceptable.

Medical Imaging

Number of measurements $= \Theta$(time patient holds breath)
Extra factor of "$1000 \cdot \log$(something)" unacceptable.

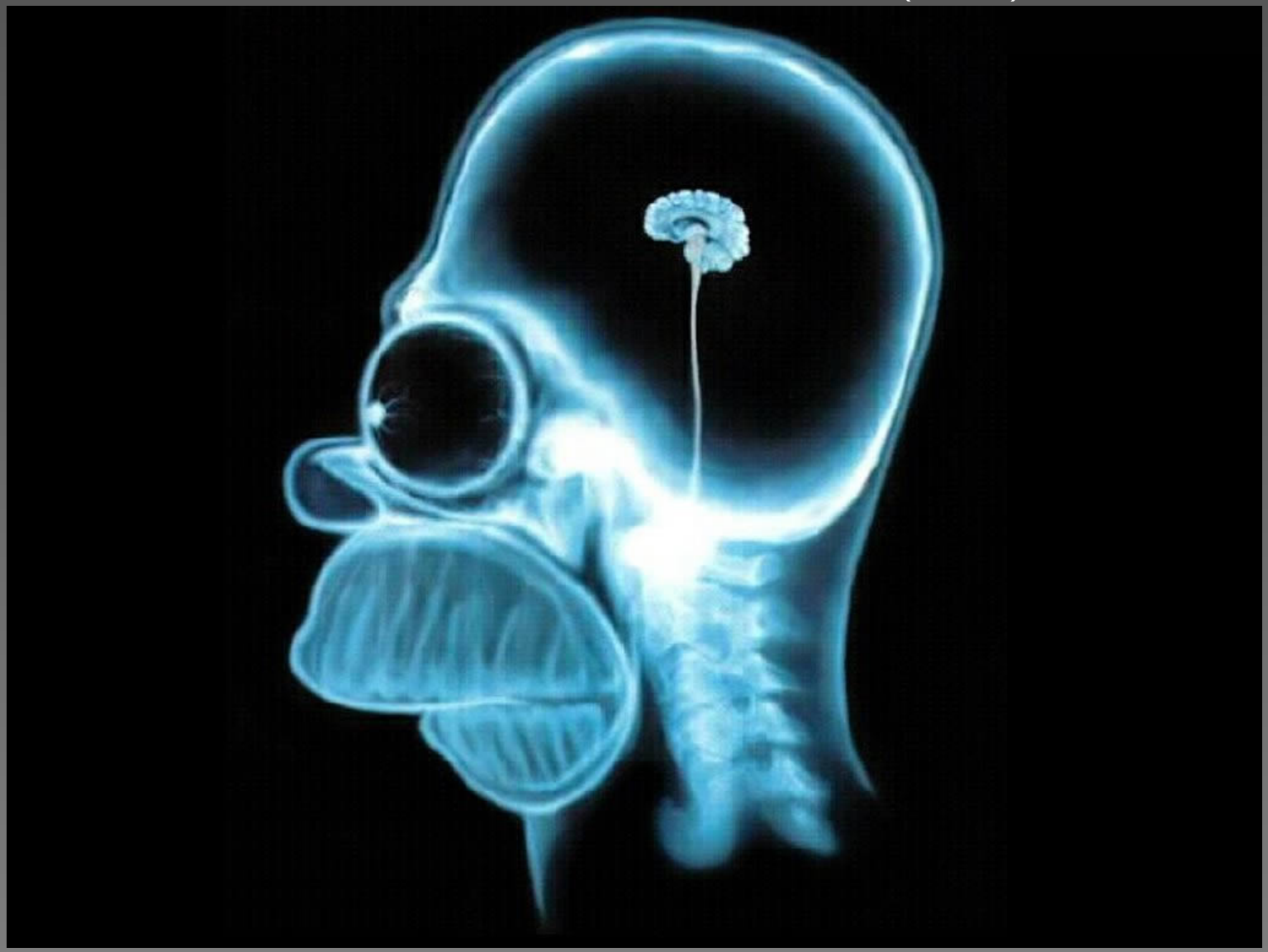
Disclaimers: This algo won't work with Fourier measurements or on sparse-gradient images.

Pixel-Sparse MRIs

Increasingly, MRI's are sparse in canonical (pixel) domain

Pixel-Sparse MRIs

Increasingly, MRI's are sparse in canonical (pixel) domain



Overview

Our algorithm:

- Find a set of positions containing at least $3k/4$ of top k positions (and others)
- Estimate their values
  - Get at most $k/4$ totally wrong
  - Other estimates increase noise slightly—key invariant
- Subtract off and repeat for $k/2$

Estimation is easy; we focus on other aspects.

Overview

Our algorithm:

- Find a set of positions containing at least $3k/4$ of top k positions (and others)
- Estimate their values
  - Get at most $k/4$ totally wrong
  - Other estimates increase noise slightly—key invariant
- Subtract off and repeat for $k/2$

Estimation is easy; we focus on other aspects.
Geometric decrease $k \to k/2$ pays for geometric tightening in noise increase:

$$1 \to (1 + 2 \cdot 2/3) \to (1 + 2 \cdot (2/3) + 2 \cdot (2/3)^2) \to \cdots \leq 5 (\to 1 + \epsilon)$$

and geometric tightening in failure probability

$$1/4 \to (1/4 + 1/16) \to (1/4 + 1/16 + 1/64) \to \cdots \leq 1/3.$$

Noise in L^2

Multiply signal slotwise by known random ± 1 before we start.

- Doesn't permanently effect heavy hitters

Noise in L^2

Multiply signal slotwise by known random ± 1 before we start.

- Doesn't permanently effect heavy hitters
- $\sigma = \pm 1$ random vector; ν is noise vector.

Noise in L^2

Multiply signal slotwise by known random ± 1 before we start.

- Doesn't permanently effect heavy hitters
- $\sigma = \pm 1$ random vector; ν is noise vector.
- What about $\sum_j \sigma_j \nu_j$?

Noise in L^2

Multiply signal slotwise by known random ± 1 before we start.

- Doesn't permanently effect heavy hitters
- $\sigma = \pm 1$ random vector; ν is noise vector.
- What about $\sum_j \sigma_j \nu_j$?
- $E\left[\sum_j \sigma_j \nu_j\right] = 0$

Noise in L^2

Multiply signal slotwise by known random ± 1 before we start.

- Doesn't permanently effect heavy hitters
- $\sigma = \pm 1$ random vector; ν is noise vector.
- What about $\sum_j \sigma_j \nu_j$?
- $E\left[\sum_j \sigma_j \nu_j\right] = 0$
- $E\left[\left|\sum_j \sigma_j \nu_j\right|^2\right] = \sum_{j,k} \nu_j \nu_k E[\sigma_j \sigma_k] = \sum_j \nu_j^2$.

Noise in L^2

Multiply signal slotwise by known random ± 1 before we start.

- Doesn't permanently effect heavy hitters
- $\sigma = \pm 1$ random vector; ν is noise vector.
- What about $\sum_j \sigma_j \nu_j$?
- $E\left[\sum_j \sigma_j \nu_j\right] = 0$
- $E\left[\left|\sum_j \sigma_j \nu_j\right|^2\right] = \sum_{j,k} \nu_j \nu_k E[\sigma_j \sigma_k] = \sum_j \nu_j^2$.

So $\left|\sum_j \sigma_j \nu_j\right| \approx \|\nu\|_2$.

Heavy Hitter Size v. Noise

Scale so $\|\text{noise}\|_2 = 1$.
Ignore heavy hitters with energy $\leq c\epsilon/k$; altogether, degrade noise from 1 to $1 + c\epsilon$.
(Set c later...)

Hashing

Send each position into one of $O(k)$ buckets, at random.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Hashing

Send each position into one of $O(k)$ buckets, at random.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Accomplished by a matrix

Hashing

Send each position into one of $O(k)$ buckets, at random.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

Accomplished by a matrix ...which is $1/k$ sparse

Hashing, results

Get around $3k/4$ buckets with:

- Exactly one fresh heavy hitter
- Around n/k total positions
- Noise contribution $\frac{1}{k}\|\nu\|_2^2$.
  - small compared with energy of heavy hitter

Reduced to 1-sparse signal of length $\approx n/k$.

1-Sparse Signal of length n/k

Ideally, recover bit-by-bit. E.g.,

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

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

But:

- Constant fraction of measurements may be wrong!

Solution: Use "good" ECC...

- Tolerate constant fraction of errors
- Blow up length by constant factor only
- Use appropriate blocksize; exhaustive algorithms are efficient.

Iterate...

First iteration:

- # heavy hitters: $k \to k/2$.
- Noise energy: 1 to $1 + 2 \cdot 2/3$
- Failure probability $1/4$.

Repeat for $(k/2)$-sparse signal.

Iterate...

Repeat for $(k/2)$-sparse signal.
j'th iteration:

- Sparsity: $k/2^j$ to $k/2^{j+1}$
- Noise: $5 - 2 \cdot (2/3)^j$ to $5 - 2 \cdot (2/3)^{j+1}$
- Failure prob: $1/3 - 1/4^j$ to $1/3 - 1/4^{j+1}$

Iterate...

Repeat for $(k/2)$-sparse signal.
j'th iteration:

- Sparsity: $k/2^j$ to $k/2^{j+1}$
- Noise: $5 - 2 \cdot (2/3)^j$ to $5 - 2 \cdot (2/3)^{j+1}$
- Failure prob: $1/3 - 1/4^j$ to $1/3 - 1/4^{j+1}$

Cost:

$$\left(\frac{k}{2^j}\right) \log\left(\frac{d \cdot 2^j}{k}\right) \left(\frac{3}{2}\right)^j \log\left(4^j\right) = k \log\left(\frac{d}{k}\right) \left(\frac{3}{4} + o(1)\right)^j.$$

Summable over j.

Summary

For each x, with high probability,

$$\|x - \widehat{x}\|_2 \leq (1+\epsilon) \min_{y\ k-\text{sparse}} \|x - y\|_2.$$

- $\widehat{x} = \mathcal{D}(Ax)$.
- A has $O(k \log(n/k))$ rows (and is $\approx 1/k$ sparse)
- $\mathcal{D}$ runs in time $k\,\text{poly}(\log(n))$.