Lecture 14: Compressed sensing

Tuesday, October 13, 2015 9:30 AM

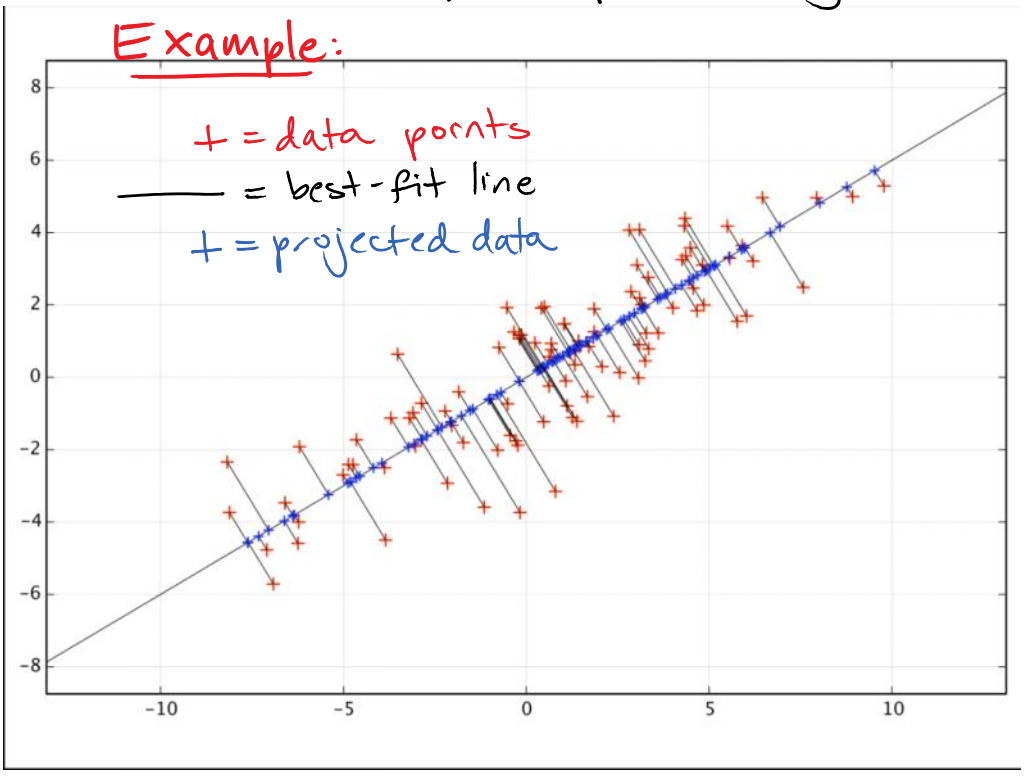
Outline: Haar wavelet basis & sparsity
Randomized dimension reduction
Compressed sensing:
Restricted isometry property (RIP)
ℓ_1 and ℓ_2 norms, and sparsity
Image reconstruction

DIMENSION REDUCTION

TRADITIONAL APPROACH:

Structured data can be projected onto a lower-dimensional subspace. "Principal component analysis" [Pearson 1901]

Example:



We will go over this later!

We will go over this later!

Problems: 1. Finding and projecting onto a good-fitting subspace is SLOW!
2. A good, low-dimensional subspace might not exist!
3. Even if the fit is decent, the data's local properties are not preserved!
(e.g., the projections of some pair of data points might be much closer than the original points)

MODERN APPROACH: Randomized dimension reduction

Ignore any structure — just apply a random linear function to the data

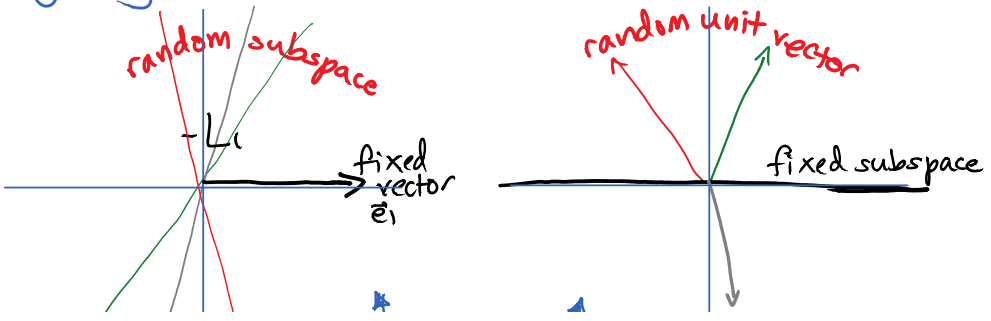
Example: Compute the projection of $e_1 = (1, 0, 0, \ldots)$ onto a random 50-dimensional subspace of $\mathbb{R}^{100}$.

```
n = 100;
d = 50;

v = zeros(n,1);
v(1) = 1;

A = randn(n, d);     % choose d random vectors in R^n
[Q,R] = qr(A,0);     % Gram-Schmidt => Q's columns are an
                     %  orthonormal basis for R(A)

projectedv = Q * (Q' * v);
```

What is $\mathbb{E}\left[\|\text{projectedv}\|^2\right]$?

Answer:

Projecting the fixed vector e_1 onto a random subspace is equivalent to projecting a random vector onto a fixed subspace.



the length of the projection has the same distribution

Squared length of random unit vector $\vec{v}$ $= \sum_{i=1}^{n} |v_i|^2 = 1$

$\Rightarrow 1 = \mathbb{E}[\|\vec{v}\|^2] = \sum_{i=1}^{n} \mathbb{E}[|v_i|^2]$

(expectation is linear)

$\Rightarrow$ for all i, $\mathbb{E}[|v_i|^2] = 1/n$ (by symmetry)

$\Rightarrow \mathbb{E}\left[\left\|P_{\text{first } d \text{ coords}}(\vec{v})\right\|^2\right] = \mathbb{E}\left[\sum_{i=1}^{d} |v_i|^2\right]$

$$= \frac{d}{n}$$

$\Rightarrow$ After scaling projection up by $\sqrt{\frac{n}{d}}$, length "should be" almost unchanged.

Lemma: [Johnson-Lindenstrauss 1984]

Start with N data points, in $\mathbb{R}^n$.

Let V be a random d-dim. subspace, where

$$d = \frac{8}{\varepsilon^2} \log N.$$

Then, with high probability, for any pair of points p, q,

$$(1-\varepsilon)\|p-q\| \leq \left\|\sqrt{\frac{n}{d}} P_V(p) - \sqrt{\frac{n}{d}} P_V(q)\right\| \leq (1+\varepsilon)\|p-q\|.$$

* Distances between all data points are nearly preserved.
* The dimension need only be $O(\log(\#\text{ of data points}))$ — independent of ambient dimension n. Logarithmic = huge reduction!!

Proof idea: "Concentration of measure"

We know $\mathbb{E}[\|p'-q'\|^2] = \|p-q\|^2$.

Show that the distribution is narrowly concentrated about its mean, and use a union bound over the $\binom{N}{2}$ pairs of data points...

about its mean, and use a union bound over the $\binom{n}{2}$ pairs of data points...

(Law of Large Numbers: Average of many independent, identically distributed random variables is tightly concentrated about the mean. But the $|v_i|^2$ are not independent...)

This is important for many, many applications....
⟶ very fast algorithms on large datasets.

Problem: Gram-Schmidt to find orthonormal basis for dimension-d subspace of $\mathbb{R}^n$ takes $O(d^2 n)$ time.

Faster, easier way: Don't bother projecting

$$A = \frac{1}{\sqrt{d}} \text{randn}(d, n)$$

scaling by $\sqrt{d}$ means each entry is $N(\text{mean } 0, \text{variance } \frac{1}{d})$
$\Rightarrow \mathbb{E}[\text{squared length of a column}]$
$= \text{total variance} = \frac{d}{d} = 1$ ✓

map data point $p \longmapsto Ap$

Even faster: Multiply by a random matrix $\{-1, 0, 1\}$ entries

$$A = \frac{1}{\sqrt{d}} \text{randi}([-1, 1], [d, n])$$ [Achlioptas 2003]

$p \longmapsto Ap$ still preserves distances!

Test this in Matlab....

HAAR WAVELET BASIS

Haar orthonormal basis for $\mathbb{R}^8$:

$$\frac{1}{\sqrt{8}}(1, 1, 1, 1, 1, 1, 1, 1)$$
$$\frac{1}{\sqrt{8}}(1, 1, 1, 1, -1, -1, -1, -1)$$
$$\frac{1}{2}(1, 1, -1, -1, 0, 0, 0, 0)$$
$$\frac{1}{\sqrt{2}}(1, -1, 0, 0, 0, 0, 0, 0)$$
$$\frac{1}{\sqrt{2}}(0, 0, 1, -1, 0, 0, 0, 0)$$
$$\frac{1}{\sqrt{2}}(0, 0, 0, 0, 1, -1, 0, 0)$$
$$\frac{1}{\sqrt{2}}(0, 0, 0, 0, 0, 0, 1, -1)$$

Intuition: It comes from recursive downsampling:

$(a, b, c, d, e, f, g, h) \in \mathbb{R}^8$

$(a,b,c,d,e,f,g,h) \in \mathbb{R}^8$

$\longrightarrow (a+b, c+d, e+f, g+h, a-b, c-d, e-f, g-h)$

first $\frac{8}{2}=4$ coordinates come from binning coords two at a time

downsample 1st 4 coords. $\longrightarrow$ $(a+b+c+d, e+f+g+h, \underset{-c-d}{a+b}, \underset{-g-h}{e+f}, \ldots)$

downsample again $\longrightarrow$ $\left(\underset{+e+f+g+h}{a+b+c+d}, \underset{-e-f-g-h}{a+b+c+d}, \ldots\right)$

DONE! Up to scaling, this gives the above basis vectors.

Haar wavelets for arrays $\mathbb{R}^{2^k \times 2^k}$

What if your vector gives the pixel values of a 2D image?
→ We should use a 2D version of the Haar basis.

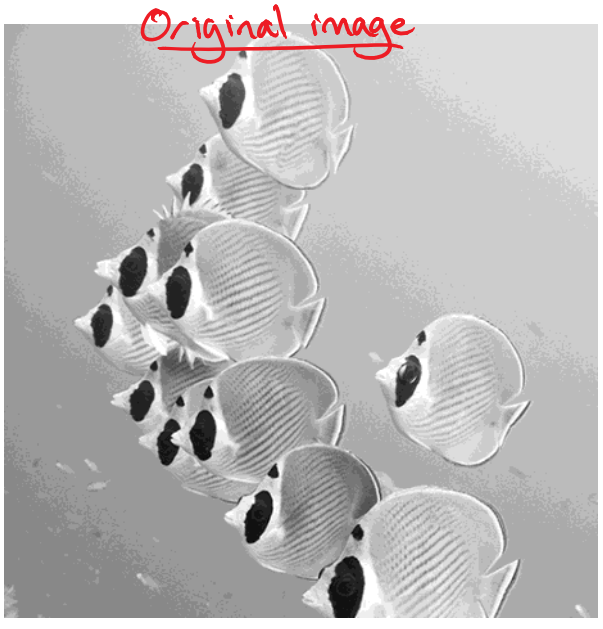
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \longrightarrow \begin{pmatrix} \text{average } a+b+c+d & \text{left-right diff. } a+c-b-d \\ \text{up-down difference } a+b-c-d & \text{cross difference } a-b-c+d \end{pmatrix}$$

& recurse in upper-left quadrant

Example:

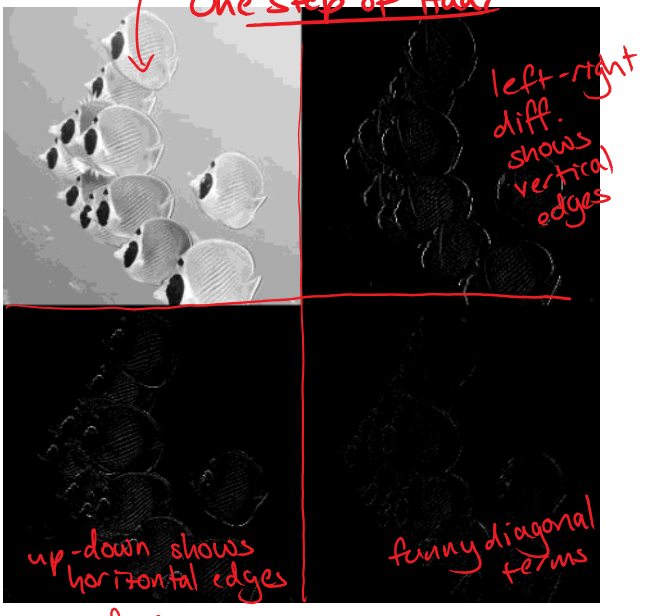
```
pkg load image;
P = double(imread('IMG_8474_g512.jpg'))/256;
PH1 = Haarstep(P, length(P));
imshow(PH1/2);
imwrite(PH1/2, 'IMG_8474_g512_Haar1.jpg');
```

Original image



downsampled image

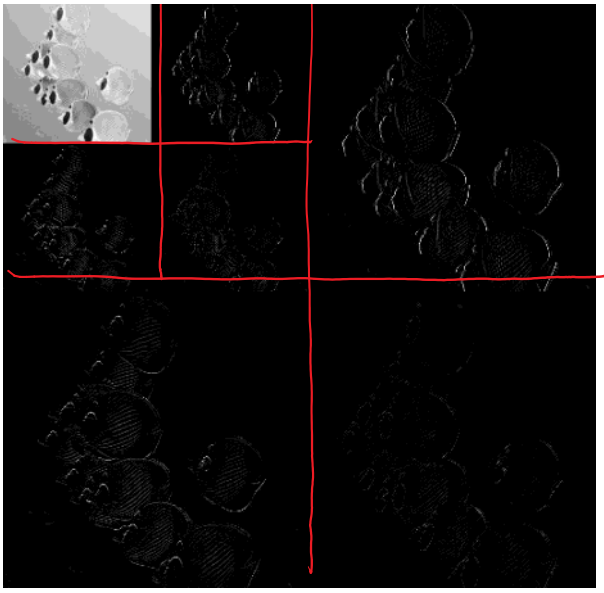
One step of Haar

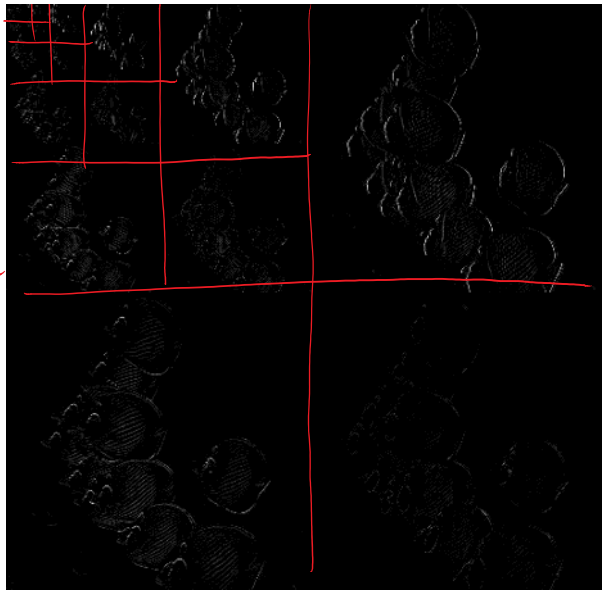


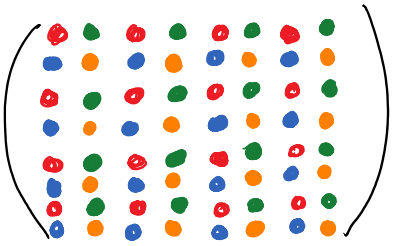
Two steps of Haar

average of everything!

Haar basis







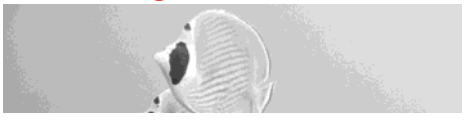
```
function p = Haarstep(p, n)
  oddodd   = p(1:2:n, 1:2:n);
  evenodd  = p(2:2:n, 1:2:n);
  oddeven  = p(1:2:n, 2:2:n);
  eveneven = p(2:2:n, 2:2:n);
  NW = (oddodd + evenodd + oddeven + eveneven)/2;
  NE = (oddodd + evenodd - oddeven - eveneven)/2;
  SW = (oddodd - evenodd + oddeven - eveneven)/2;
  SE = (oddodd - evenodd - oddeven + eveneven)/2;
  p(1:n/2,1:n/2) = NW;
  p(1:n/2,n/2+1:n) = NE;
  p(n/2+1:n,1:n/2) = SW;
  p(n/2+1:n,n/2+1:n) = SE;
end
```

```
function p = Haar(p)
  n = length(p);
  while (n > 1)
    p = Haarstep(p, n);
    n = n/2;
  end
end
```

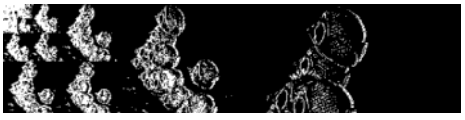
Nice property: Real-world images tend to have fairly sparse representations in the Haar basis (i.e., most Haar coordinates are close to 0).

Examples:

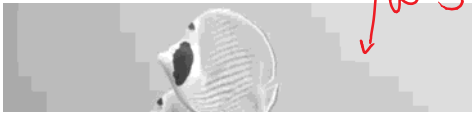
Original



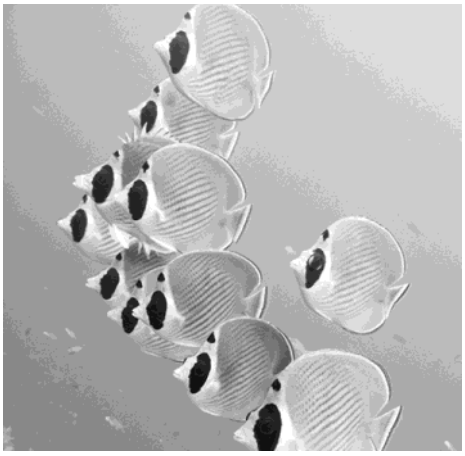
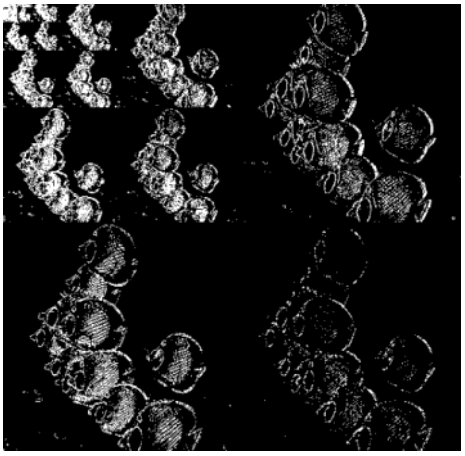
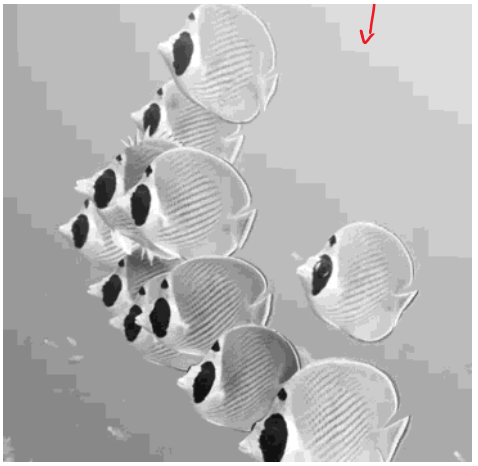
Keep 10% of coeffs



Result



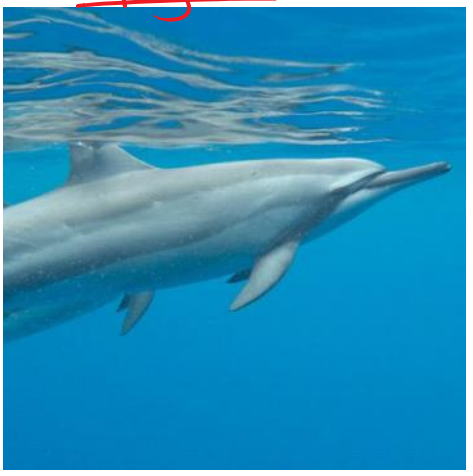
notice the blocky gradients. Why?

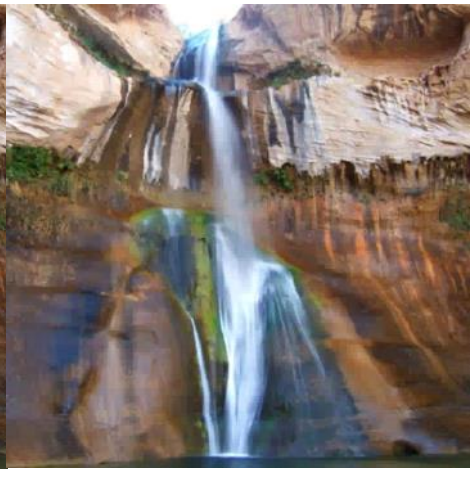




Original

Keeping 10%

Keeping 20%




Why are gradients so blocky?

- Other bases perform better...

Code to change back from the Haar basis:

```
function p = undoHaarstep(p, n)
  NW = p(1:n/2,1:n/2);
  NE = p(1:n/2,n/2+1:n);
  SW = p(n/2+1:n,1:n/2);
  SE = p(n/2+1:n,n/2+1:n);
  oddodd = (NW + NE + SW + SE)/2;
  evenodd = (NW + NE - SW - SE)/2;
  oddeven = (NW - NE + SW - SE)/2;
  eveneven = (NW - NE - SW + SE)/2;
  p(1:2:n, 1:2:n) = oddodd;
  p(2:2:n, 1:2:n) = evenodd;
  p(1:2:n, 2:2:n) = oddeven;
  p(2:2:n, 2:2:n) = eveneven;
end

function p = undoHaar(p)
  n = 2;
  while (n <= length(p))
    p = undoHaarstep(p, n);
    n = n*2;
  end
end
```

Moral: Real-world signals are often nearly sparse in some known basis.

Compressed Sensing

Model: True signal $\hat{x}$
Observed signal $b = A\hat{x}$
(for a known A)

Goal: Recover $\hat{x}$ by solving $Ax = b$.

Example: Camera sensor

$\hat{x}$ = image values at each pixel

$A = I$ identity matrix

$\Rightarrow$ you observe $\hat{x}$, done.

Different cases for solving $Ax = b$:

1. A $n \times n$ has rank n
   $\Rightarrow$ unique solution $x = A^{-1}b$
2. A $m \times n$ has rank n
   $\Rightarrow N(A) = \{0\}$ (since $\dim N(A) = n - \text{rank}(A)$)
   $\Rightarrow$ unique solution if $b \in R(A)$
3. $b \notin R(A)$
   $\Rightarrow$ no solution
   $\Rightarrow$ New goal: Find x to minimize $\|Ax - b\|$?
4. $b \in R(A)$ but $N(A) \neq \{0\}$
   $\Rightarrow \infty$ many solutions!
   e.g., if A $m \times n$ with $m < n$,
   $\text{rank}(A) \leq m \Rightarrow \dim N(A) = n - \text{rank}(A) \geq 1$
   $\Rightarrow$ New goal: Find "best" solution?

Compressed sensing:

A $m \times n$ with $m < n$

"#observations < #degrees of freedom"

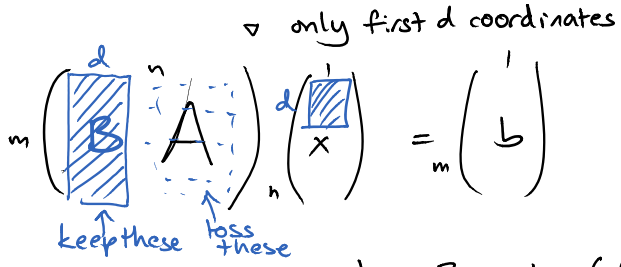
Goal: Find **sparsest** solution

$$\begin{cases} \text{minimize } (\#\text{ of nonzero coords of } x) \\ \text{subject to } Ax = b \end{cases}$$

Unfortunately, this is NP-hard.
Even if we know there is a solution x with $\leq d$ nonzero coordinates, best approach is to try all $\binom{n}{d} = O(n^d)$ possible subsets.

$$\underset{m}{\left(\quad \overset{n}{A} \quad\right)}\underset{n}{\begin{pmatrix} x \end{pmatrix}} = \underset{m}{\begin{pmatrix} b \end{pmatrix}}$$

↓ if we guess x uses only first d coordinates



solve $By = b$ (if $m \geq d$)

Key trick: ℓ_1 norm minimization

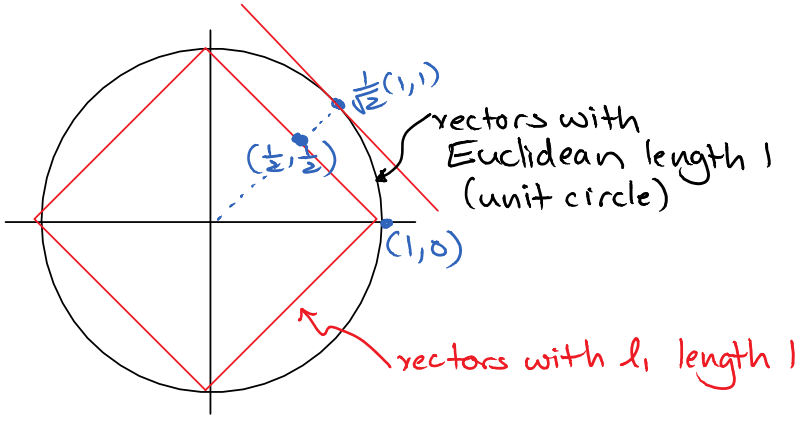
Definition:

- The ℓ_2, or Euclidean, norm is given by
$$\|x\|_2 = \|x\| = \sqrt{\sum_i |x_i|^2}$$
- The ℓ_1 norm is given by
$$\|x\|_1 = \sum_i |x_i| .$$

Example:

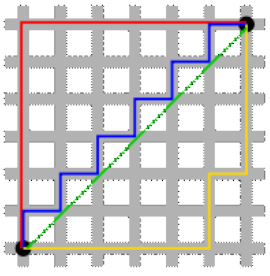
$$\|(1,0)\|_1 = 1 = \|(1,0)\|_2$$

$$\|(\tfrac{1}{2},\tfrac{1}{2})\|_1 = 1 > \|(\tfrac{1}{2},\tfrac{1}{2})\|_2 = \tfrac{1}{\sqrt{2}}$$

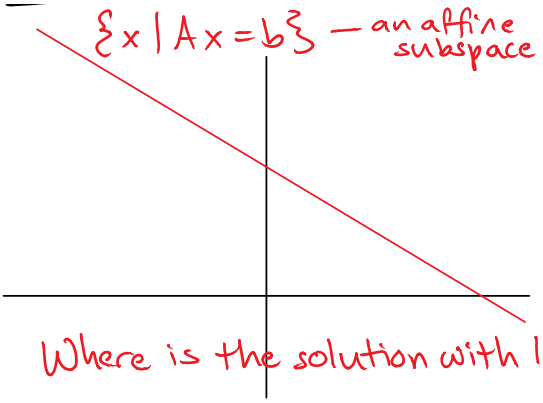


$$\|\tfrac{1}{\sqrt{2}}(1,1)\|_1 = \tfrac{1}{\sqrt{2}}\|(1,1)\|_1 = \sqrt{2}$$

"Taxicab geometry/Manhattan distance"

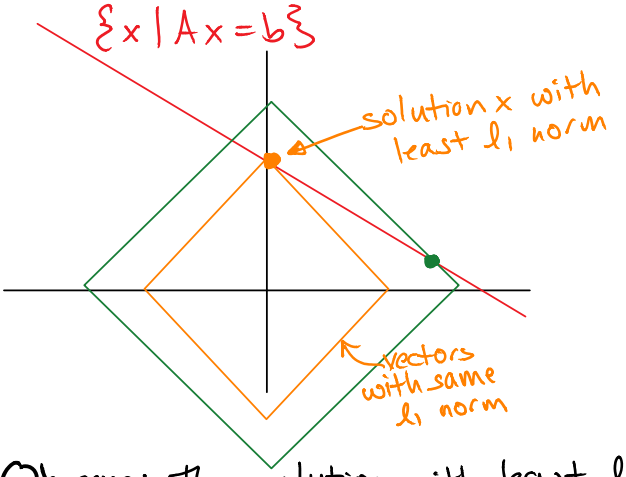


Question:



Where is the solution with least ℓ_1 norm?

Answer:



Observe: The solution with least ℓ_1 norm is also the sparsest solution!!
(This is because the line of solutions doesn't have slope ± 1.)

Intuition: The Euclidean norm treats

$\frac{1}{\sqrt{n}}(1, 1, 1, \ldots, 1)$, $\frac{1}{\sqrt{2}}(1, 1, 0, 0, \ldots, 0)$

and $(0, 1, 0, 0, \ldots, 0)$

all the same; they all have length one.

But the ℓ_1 norm favors sparsity

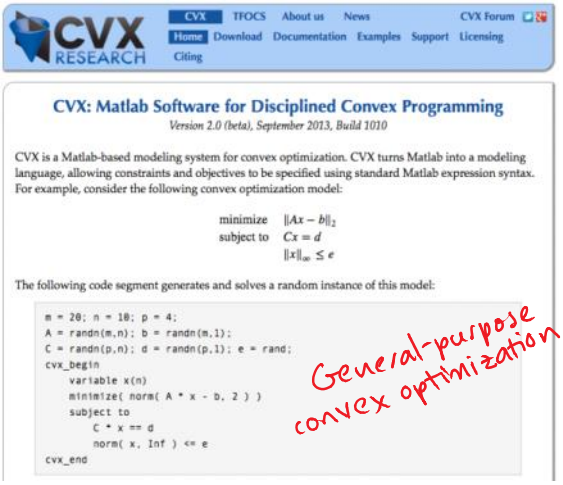
$\frac{1}{\sqrt{n}} \cdot n = \sqrt{n}$ $\qquad$ $\frac{1}{\sqrt{2}} \cdot 2 = \sqrt{2}$

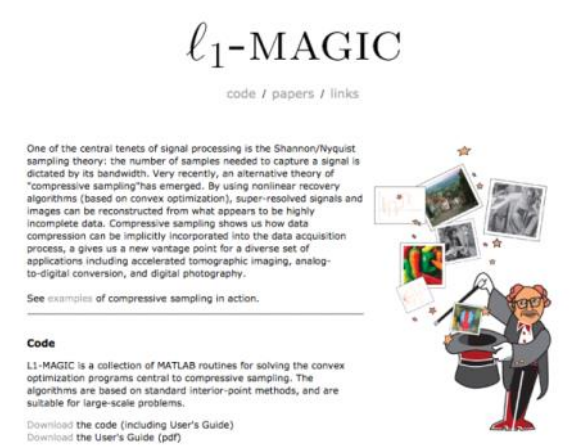
1 in ℓ_1 norm

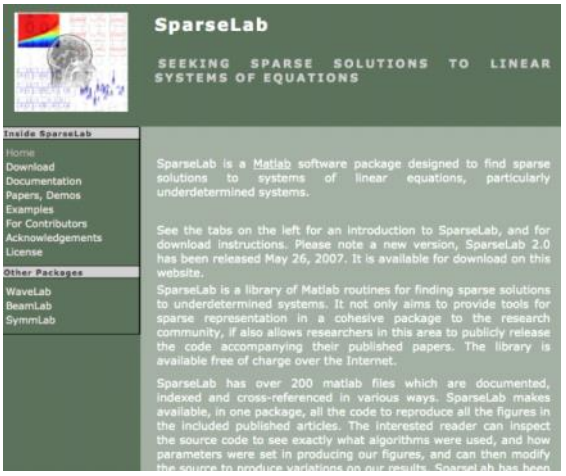
And there are **efficient algorithms** for solving

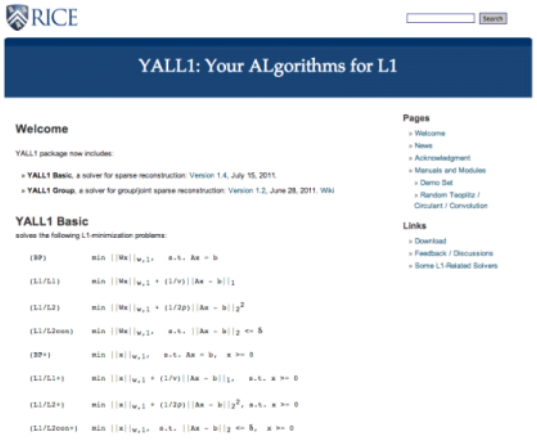
$$\begin{cases} \text{minimize} & \|x\|_1 \\ \text{subject to} & Ax = b \end{cases}$$

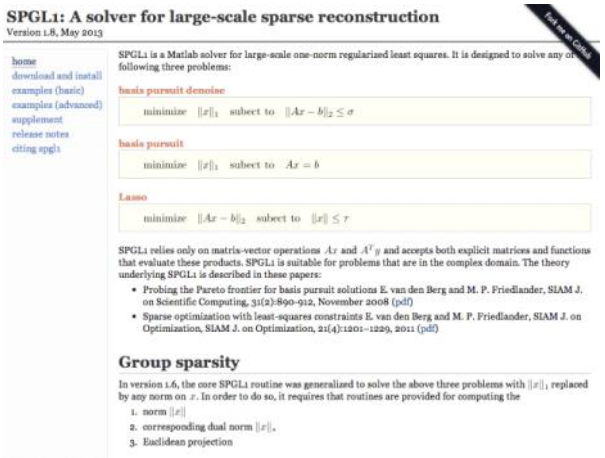
Examples: (all these are Matlab-only, not Octave)











and many more...

Other applications of the minimizing $\ell_1 \rightarrow$ sparsity trick include deep neural networks in machine learning...

EXAMPLE:

Original image:
"Shepp-Logan phantom"
(supposed to be a brain)
64 × 64 pixels

or P = phantom("Modified-Shepp-Logan",64)

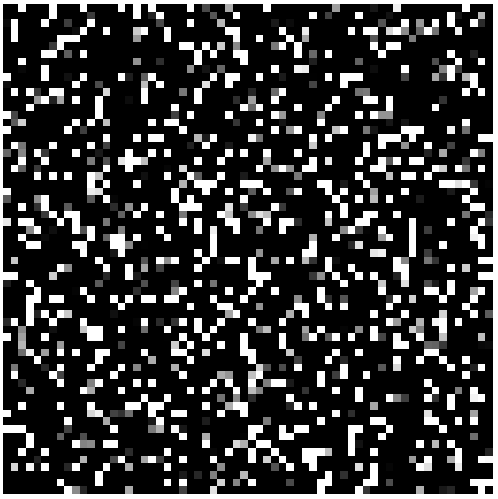
```
P = double(imread('Shepp-Logan64.png'))/256;
n = length(P);  =64
N = n * n;  =4096
```

Measurement model:

```
d = n * log2(n) * 5;   = 1920 = 0.47·N
A = 1/sqrt(d) * randn(d, N);
b = A * P(:);
```

inner product with d random vectors
(this is not realistic)

① Matlab's solve routine



```
x = A\b;
Pout = reshape(x, n, n);
imshow(Pout);
imwrite(Pout, 'Pout.png');
```

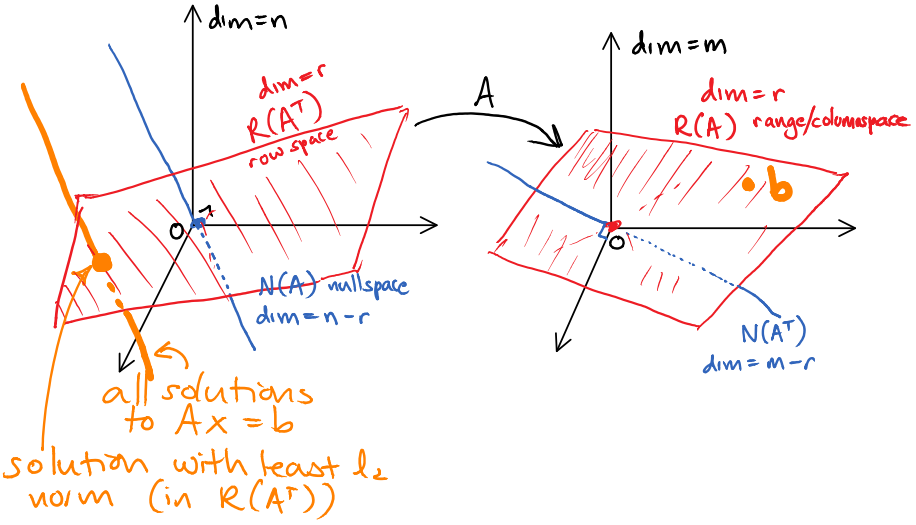
System is completely undetermined:
1920 equations in 4096 unknowns

② Solution with least l_2 norm

brain?

```
cvx_begin
    variable x(N);
    minimize( norm(x, 2) )
    subject to
        A*x == b
cvx_end

Plsq = reshape(x, n, n);
imshow(Plsq);
```

takes ~2 minutes using CVX
there are better ways!!



③ Look for a sparse solution

Move to the Haar basis:

```
% put rows of A into the Haar basis
AHaar = A;
for i = 1:d
  reshapedrow = reshape(AHaar(i,:), n, n);
  reshapedrow = Haar(reshapedrow);
  AHaar(i,:) = reshapedrow(:);
end
```

```
sparsity = numnonzeroentries(Haar(P)); 725 (this is cheating...)
subsets = combnk(1:N, sparsity);  % don't run this code!
for i = 1:size(subsets,1)
  % try to find a solution using those columns only
  % if successful, then stop
  AHaarrestricted = AHaar(:,subsets(i));
  x = AHaarrestricted\b;
  if (sum(sum(abs(AHaarrestricted*x-b))) < 10^-5)
    break;
  end;
end
```

Note: This takes $\binom{64^2}{725} = 2^{2750}$ time steps.

This is more than

(# atoms in observable universe) $\cdot$ (# of Planck time steps since Big Bang)

2^{270} $\qquad$ 2^{207}

$\cdot$ (# books in Library of Congress)^{94}.

2^{24}

④ Solution with least ℓ_1 norm

```
cvx_begin
    variable x(N);
    minimize( norm(x, 1) )
    subject to
        AHaar*x == b
cvx_end

Pl1 = reshape(x, n, n);
Pl1 = undoHaar(Pl1);
imshow(Pl1);
imwrite(Pl1, 'Pl1.png');
```

takes ~20 minutes

Summary of results:

Original image | Least ℓ_2 norm | Least ℓ_1 norm

Haar: 725 nonzero $= .17N$
278 $\ell_1 = .06N$

ℓ_1 error $= 0.1 \times N$

Haar: 3943 nonzero
342 ℓ_1

ℓ_1 error $0.007 \times N$

Haar: 2199 nonzero
275 ℓ_1 !

Running ℓ_1 magic

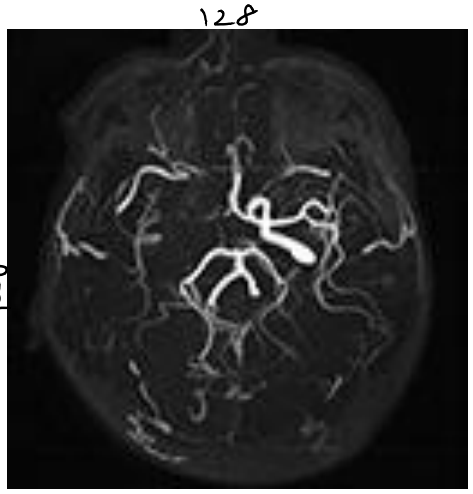
```
tic
xp = l1eq_pd(x0, AHaar, [], b, 1e-3);
toc
```

only 50 seconds

ℓ_1 error $= 0.8 = 0.0002\,N$

Haar: 734 nonzero
277 ℓ_1

EXAMPLE 2:



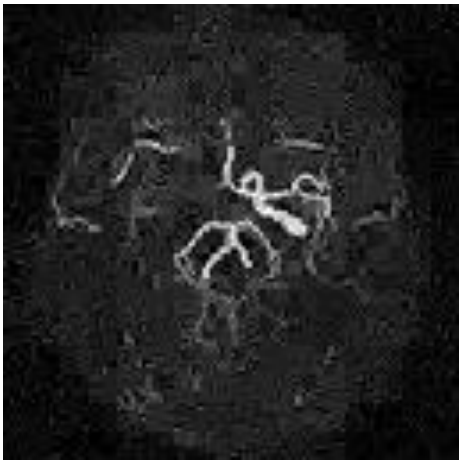
# of variables
$N = 128^2 = 16384$

# of observations
$d = 4480 = 0.27N$

Haar: ~0.87N nonzero
.05N ℓ_1

~15 minutes

15 minutes in ℓ_1 magic



ℓ_1 error = .046N
Haar: 0.4N nonzero
.03N ℓ_1

How to do better:

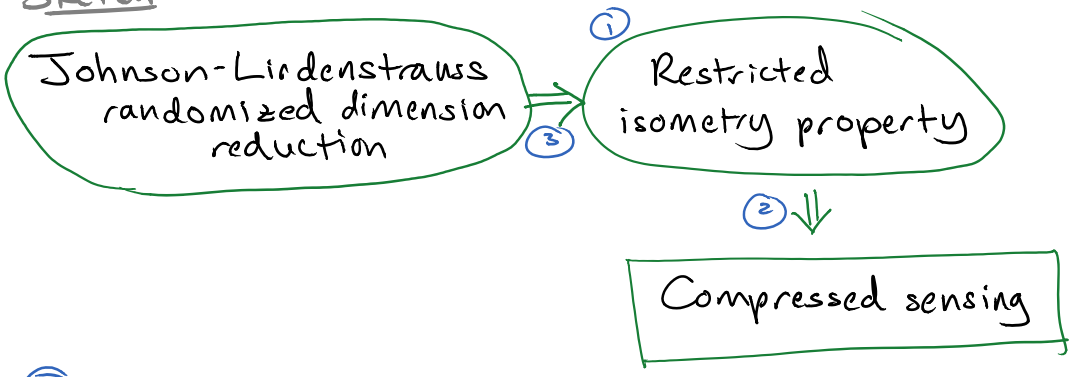
1. More equations
2. Better wavelet basis
3. Smarter optimization algorithms (see references)

Why does compressed sensing work?

1. Signals are **sparse**, or "nearly sparse" in a **known basis**.
2. Dimension reduction

Dimension-Reduction & Compressed Sensing

Sketch:



(1)

Definition: "Restricted Isometry Property" (RIP)

Roughly: A matrix is RIP if multiplying by that matrix preserves the lengths of all sparse vectors.

Precisely: The $d \times n$ matrix A is (s, ε)-RIP if
(sparsity, length error)

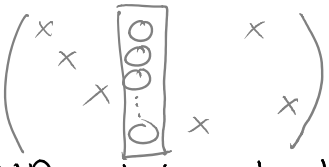
for all $\vec{x} \neq 0$ with $\leq s$ nonzero coordinates,

$$\left| \frac{\|A\vec{x}\|}{\|\vec{x}\|} - 1 \right| \leq \varepsilon.$$

Example: Obviously, the $n \times n$ identity I is RIP; it preserves all lengths exactly. But there are also many nontrivial examples, including most random projections, most samples of Fourier coefficients.

Example: If A, $d \times n$ with $d < n$, is very sparse — fewer than n nonzero entries — then it cannot be RIP.

because then one column of A would be 0:



$\Rightarrow$ **Intuition:** Rows of RIP matrices should be "diffuse," not sparse.

Note: Of course, RIP depends on the basis for sparsity. Sampling Fourier coefficients will not be RIP if the

source has a sparse Fourier representation.

(2) Restricted isometry property $\Rightarrow$ Compressed sensing

Theorem: Let A be a $(2s, \varepsilon)$-RIP matrix, with $\varepsilon < 1$.
Let x have $\leq s$ non-zero coeffs, and $b = Ax$.
Let

$\tilde{x}$ = the sparsest solution to $Ay = b$.

Then $\tilde{x} = x$ —— sparse recovery works.

Proof: Assume $\tilde{x} \neq x$. Then $x - \tilde{x}$ has $\leq 2s$ nonzero coeffs,

$$A(x - \tilde{x}) = 0.$$

$$\Rightarrow \left| \frac{\|A(x-\tilde{x})\|}{\|x-\tilde{x}\|} - 1 \right| = 1 > \varepsilon, \text{ a contradiction. } \square$$

(3) Johnson-Lindenstrauss randomized dimension reduction $\Rightarrow$ Restricted isometry property

J-L lemma: N^2 vectors $\longrightarrow$ $O(\log N)$ dimensions
w/o much changing any vectors' length.

For RIP we want all s-sparse vectors to have lengths preserved. There are

$$\binom{n}{s} = O(n^s)$$

coordinate subsets of size s, but infinitely many s-sparse vectors.

Solution:

(1) If A preserves the length of x, then it also preserves the length of $2x$, $10x$ —— any multiple of x.

$$\|A \cdot 2x\| = \|2Ax\| = |2| \cdot \|Ax\|$$

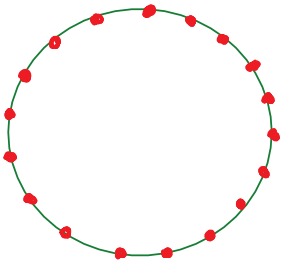
$\Rightarrow$ It is enough that A preserve lengths for all x with $\|x\| = 1$.

$\Rightarrow$ We only need to check that A works on the

$\binom{n}{s}$ spheres in $\mathbb{R}^s$.
~still ∞ many points

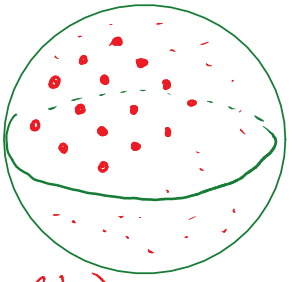
② Approximate the sphere by a finite set

$\mathbb{R}^2$:



all points on the circle are within ε of one of the $\frac{2\pi}{2\varepsilon}$ red points

$\mathbb{R}^3$:



$O(\frac{1}{\varepsilon^2})$ points in net

$\mathbb{R}^s$: $\left(\frac{3}{\varepsilon}\right)^{s-1}$ red points are enough

Fact: to be within ε of any point on the sphere

Claim: If the scaled projection A $(1\pm\varepsilon)$-preserves the lengths of all points in the net, then it $(1\pm 3\varepsilon)$-preserves lengths for all points in the sphere.

(Intuitively obvious.)

Proof: We are given $\left|\frac{\|Ax\|}{\|x\|} - 1\right| \le \varepsilon$ for all red points in the net.

Let y be some point in the sphere; it is within ε of some net point x. So $\|y\| = \|x\| = 1$ and $\|y - x\| \le \varepsilon$.

$$\Rightarrow \|Ay\| = \|Ax + A(y-x)\|$$
$$\le \|Ax\| + \|A(y-x)\|$$
$$\le 1+\varepsilon \qquad \le \sqrt{\frac{n}{d}} \cdot \|y-x\|$$

since $A = \sqrt{\frac{n}{d}}$ times a projection, and projections can't increase lengths

$$\le 1 + \left(\sqrt{\frac{n}{d}} + 1\right)\varepsilon.$$

This is no good! We want $d \ll n$, $\varepsilon \sim$ constant.

Better approach: For the worst y, let $\delta = \|Ay\| - 1$.

$$\Rightarrow \|Ay\| = 1+\delta \le (1+\varepsilon) + (1+\delta)\cdot\varepsilon$$
$$\Rightarrow \delta(1-\varepsilon) \le 2\varepsilon$$

$\Rightarrow \delta \le \frac{2\varepsilon}{1-\varepsilon} \le 3\varepsilon$ for $\varepsilon \le \frac{1}{3}$.

To lower-bound $\|Ay\|$, use

$$\|Ay\| \ge \|Ax\| - \|A(y-x)\|$$

$$\ge 1-\varepsilon-(1+\delta)\varepsilon$$

$$\ge 1-3\varepsilon \quad \text{for } \varepsilon \le \tfrac{1}{3}. \qquad \square$$

$\Rightarrow$ total # of vectors whose lengths we need to preserve

$$\le \binom{n}{s} \times \left(\frac{3}{\varepsilon}\right)^{s-1} = N$$

J-L $\Rightarrow$ $d = O\left(\frac{\log N}{\varepsilon^2}\right)$ dimensions are enough

$$= O\left(\tfrac{1}{\varepsilon^2} \log \left(\tfrac{n}{\varepsilon}\right)^s\right)$$

$$\boxed{d = O\left(s \cdot \log\left(\tfrac{n}{\varepsilon}\right) \cdot \tfrac{1}{\varepsilon^2}\right)}$$

- s: sparsity = # coeffs we must recover
- $\log\left(\frac{n}{\varepsilon}\right)$: log-dependence on dimension
- $\frac{1}{\varepsilon^2}$: fudge factor

Conclusion: If A is $\sqrt{\frac{n}{d}}$ times the projection onto a random d-dimensional subspace, with d as above, then with high probability A is $(s, 3\cdot\varepsilon)$-RIP.

Johnson-Lindenstrauss randomized dimension reduction $\Rightarrow$ Restricted isometry property $\Downarrow$ Compressed sensing ✓ DONE.