Sparse Fourier Transform (lecture 4)

Michael Kapralov[1]

[1] IBM Watson → EPFL

St. Petersburg CS Club
November 2015

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \omega^{-ij},$$

where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \omega^{-ij},$$

where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

Goal: find the top k coefficients of $\widehat{x}$ approximately

In previous lectures:

- exactly k-sparse: $O(k \log n)$ runtime and samples

Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform of x:

$$\widehat{x}_i = \frac{1}{n} \sum_{j \in [n]} x_j \omega^{-ij},$$

where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

Goal: find the top k coefficients of $\widehat{x}$ approximately

In previous lectures:

- exactly k-sparse: $O(k \log n)$ runtime and samples
- approximately k-sparse: $O(k \log^2 n(\log\log n))$ runtime and samples

This lecture, for approximately k-sparse case:

- $k \log n \log^{O(1)} \log n$ samples in $k \log^2 n \log^{O(1)} \log n$ time;
- $O(k \log n)$ samples (optimal).

Improvements?

List$\leftarrow \emptyset$

For $t = 1$ **to** $\log k$

$B_t \leftarrow Ck/4^t$

$\gamma_t \leftarrow 1/(C2^t)$

$List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$

End

Improvements?

List$\leftarrow \emptyset$
For $t = 1$ **to** $\log k$

$B_t \leftarrow Ck/4^t$

$\gamma_t \leftarrow 1/(C2^t)$

$List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$

End

Summary:

- independent invocations of PARTIALRECOVERY: use fresh samples in every iteration

Improvements?

List$\leftarrow \emptyset$
For $t = 1$ **to** $\log k$

$B_t \leftarrow Ck/4^t$

$\gamma_t \leftarrow 1/(C2^t)$

$List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$

End

Summary:

- independent invocations of PartialRecovery: use fresh samples in every iteration
- reduce (approximate) sparsity at geometric rate

Improvements?

List $\leftarrow \emptyset$
For $t = 1$ **to** $\log k$

$B_t \leftarrow Ck/4^t$

$\gamma_t \leftarrow 1/(C2^t)$

$List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$

End

Summary:

- independent invocations of PARTIALRECOVERY: use fresh samples in every iteration
- reduce (approximate) sparsity at geometric rate
- need sharp filters to reduce sparsity

Improvements?

List$\leftarrow \emptyset$
For $t = 1$ **to** $\log k$
$B_t \leftarrow Ck/4^t$

$\gamma_t \leftarrow 1/(C2^t)$

$List \leftarrow List + \text{PARTIALRECOVERY}(B_t, \gamma_t, List)$
End

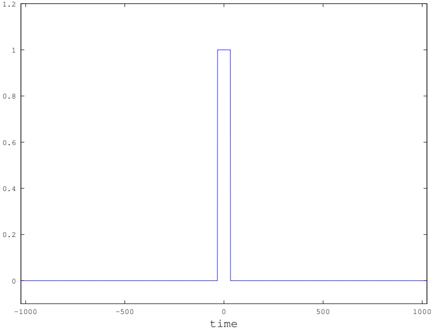
Summary:

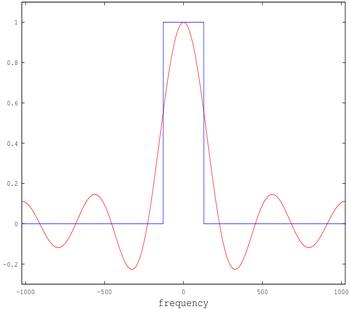
- independent invocations of PARTIALRECOVERY: use fresh samples in every iteration
- reduce (approximate) sparsity at geometric rate
- need sharp filters to reduce sparsity
- lose $\Omega(\log n)$ time and sparsity because of sharpness

Why not use simpler filters with smaller support?

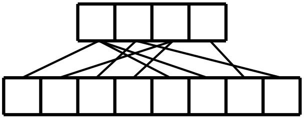
Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





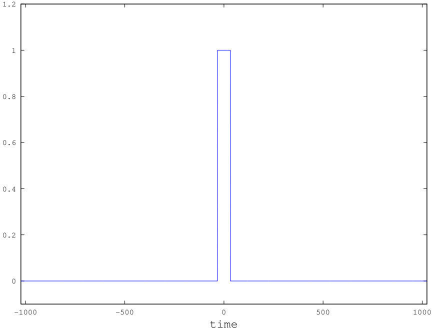
$\mathrm{supp}(G) = B \approx k$ as opposed to $B \approx k \log n$, but buckets leak

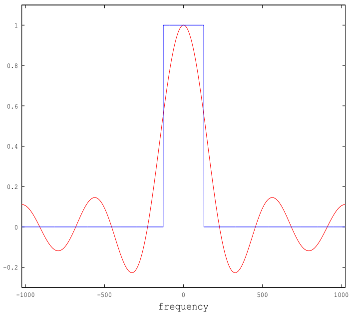


Why not use simpler filters with smaller support?

Let

$$G_j := \begin{cases} 1/(B+1) & \text{if } j \in [-B/2, B/2] \\ 0 & \text{o.w.} \end{cases}$$





$\text{supp}(G) = B \approx k$ as opposed to $B \approx k \log n$, but buckets leak

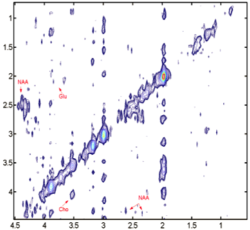
Can only identify and approximate elements of value at least $\approx \|\widehat{x}\|_2^2/k$, and estimate up to $\approx \|\widehat{x}\|_2^2/k$ additive error, so need to repeat $\Omega(\log n)$ times

Sample complexity

Sample complexity=number of samples accessed in time domain.

In some applications at least as important as runtime

Shi-Andronesi-Hassanieh-Ghazi-Katabi-Adalsteinsson'
ISMRM'13

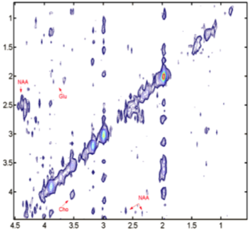


Sample complexity

Sample complexity=number of samples accessed in time domain.

In some applications at least as important as runtime

Shi-Andronesi-Hassanieh-Ghazi-Katabi-Adalsteinsson'
ISMRM'13



Given access to $x \in \mathbb{C}^n$, find $\widehat{y}$ such that

$$||\widehat{x} - \widehat{y}||^2 \leq C \cdot \min_{k-\text{sparse } \widehat{z}} ||\widehat{x} - \widehat{z}||^2$$

Use smallest possible number of samples?

Uniform bounds (for all):

Candes-Tao'06
Rudelson-Vershynin'08
Cheraghchi-Guruswami-Velingker'12
Bourgain'14
Haviv-Regev'15

Deterministic, $\Omega(n)$ runtime

$O(k \log^2 k \log n)$

Non-uniform bounds (for each):

Goldreich-Levin'89
Kushilevitz-Mansour'91, Mansour'92
Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02
Gilbert-Muthukrishnan-Strauss'05
Hassanieh-Indyk-Katabi-Price'12a
Hassanieh-Indyk-Katabi-Price'12b

Randomized, $O(k \cdot \mathrm{poly}(\log n))$ runtime

$O(k \log^2 n)$

Lower bound: $\Omega(k \log(n/k))$ for non-adaptive algorithms Do-Ba-Indyk-Price-Woodruff'10

Uniform bounds (for all):

Candes-Tao'06
Rudelson-Vershynin'08
Cheraghchi-Guruswami-Velingker'12
Bourgain'14
Haviv-Regev'15

Deterministic, $\Omega(n)$ runtime

$O(k \log^2 k \log n)$

Non-uniform bounds (for each):

Goldreich-Levin'89
Kushilevitz-Mansour'91, Mansour'92
Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02
Gilbert-Muthukrishnan-Strauss'05
Hassanieh-Indyk-Katabi-Price'12a
Hassanieh-Indyk-Katabi-Price'12b

Randomized, $O(k \cdot \text{poly}(\log n))$ runtime

$O(k \log^2 n)$

Lower bound: $\Omega(k \log(n/k))$ for non-adaptive algorithms Do-Ba-Indyk-Price-Woodruff'10

Theorem

There exists an algorithm for ℓ_2/ℓ_2 sparse recovery from Fourier measurements using $O(k \log n \cdot \log^{O(1)} \log n)$ samples and $O(k \log^2 n \cdot \log^{O(1)} \log n)$ runtime.

Optimal up to a poly($\log\log n$) factors for $k \leq n^{1-\delta}$.

ℓ_2/ℓ_2 sparse recovery guarantees:

$$||\widehat{x} - \widehat{y}||^2 \leq C \cdot \min_{k-\text{sparse } \widehat{z}} ||\widehat{x} - \widehat{z}||^2$$

ℓ_2/ℓ_2 sparse recovery guarantees:

$$||\widehat{x}-\widehat{y}||^2 \le C \cdot \mathrm{Err}_k^2(\widehat{x})$$

$$|\widehat{x}_1| \ge \ldots \ge |\widehat{x}_k| \ge$$
$$|\widehat{x}_{k+1}| \ge |\widehat{x}_{k+2}| \ge \ldots$$

$$\mathrm{Err}_k^2(\widehat{x}) = \sum_{j=k+1}^{n} |\widehat{x}_j|^2$$

Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$

ℓ_2/ℓ_2 sparse recovery guarantees:

Signal to noise ratio $R = ||\widehat{x} - \widehat{y}||^2 / \mathrm{Err}_k^2(\widehat{x}) \leq C$

$|\widehat{x}_1| \geq \ldots \geq |\widehat{x}_k| \geq$
$|\widehat{x}_{k+1}| \geq |\widehat{x}_{k+2}| \geq \ldots$

$\mathrm{Err}_k^2(\widehat{x}) = \sum_{j=k+1}^{n} |\widehat{x}_j|^2$

Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$

ℓ_2/ℓ_2 sparse recovery guarantees:

Signal to noise ratio $R = ||\widehat{x} - \widehat{y}||^2/\mathrm{Err}_k^2(\widehat{x}) \leq C$

$|\widehat{x}_1| \geq \ldots \geq |\widehat{x}_k| \geq$
$|\widehat{x}_{k+1}| \geq |\widehat{x}_{k+2}| \geq \ldots$

$\mathrm{Err}_k^2(\widehat{x}) = \sum_{j=k+1}^{n} |\widehat{x}_j|^2$

Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$

Sufficient to ensure that most elements are below average noise level:

$$|\widehat{x}_i - \widehat{y}_i|^2 \leq c \cdot \mathrm{Err}_k^2(\widehat{x})/k =: \mu^2$$

Iterative recovery

Many algorithms use the iterative recovery scheme:

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L
  $\widehat{z} \leftarrow \textsc{PartialRecovery}(x, \widehat{y}_{t-1})$ ▷ Takes random samples of $x - y$
  Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

$\textsc{PartialRecovery}(x, \widehat{y})$
return dominant Fourier coefficients $\widehat{z}$ of $x - y$ (approximately)

dominant coefficients $\approx |\widehat{x}_i - \widehat{y}_i|^2 \geq \mu^2$(above average noise level)

PARTIALRECOVERY$(x, \widehat{y})$

return dominant Fourier coefficients $\widehat{z}$ of $x - y$ (approximately)

dominant coefficients $\approx |\widehat{x}_i - \widehat{y}_i|^2 \geq \mu^2$ (above average noise level)

Main questions:

- How many samples per SNR reduction step?
- How many iterations?

$\textsc{PartialRecovery}(x, \widehat{y})$

return dominant Fourier coefficients $\widehat{z}$ of $x - y$ (approximately)

dominant coefficients $\approx |\widehat{x}_i - \widehat{y}_i|^2 \geq \mu^2$ (above average noise level)

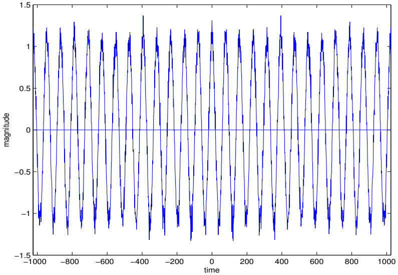
Main questions:

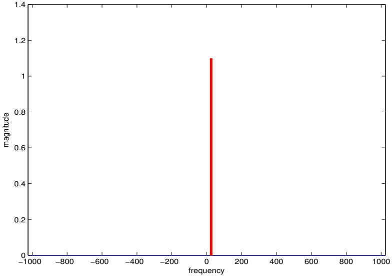
- How many samples per SNR reduction step?
- How many iterations?

Summary of techniques from

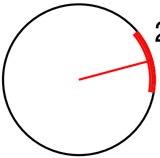
Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02, Akavia-Goldwasser-Safra'03, Gilbert-Muthukrishnan-Strauss'05, Iwen'10, Akavia'10, Hassanieh-Indyk-Katabi-Price'12a, Hassanieh-Indyk-Katabi-Price'12b

1-sparse recovery from Fourier measurements





$$\boxed{x_a = \omega^{a \cdot f} + \text{noise}}$$

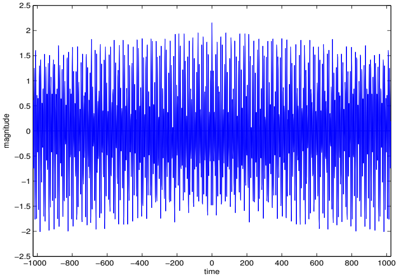


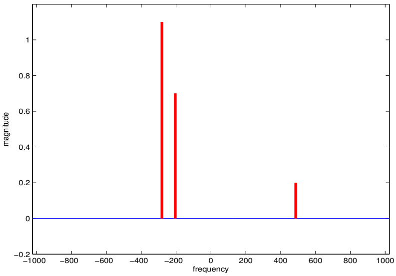
$O(\log_{SNR} n)$ measurements

for random a

Reducing *k*-sparse recovery to 1-sparse recovery

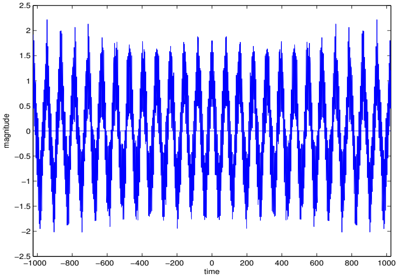
Permute with a random linear transformation and phase shift

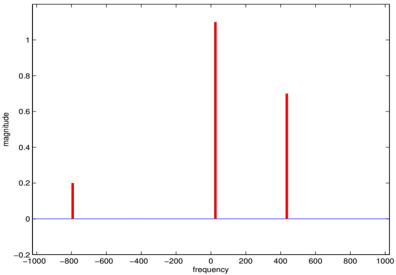




Reducing k-sparse recovery to 1-sparse recovery

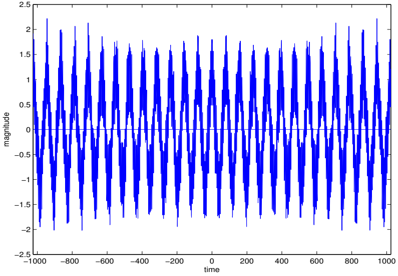
Permute with a random linear transformation and phase shift

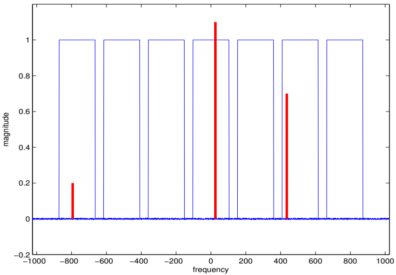




Reducing k-sparse recovery to 1-sparse recovery

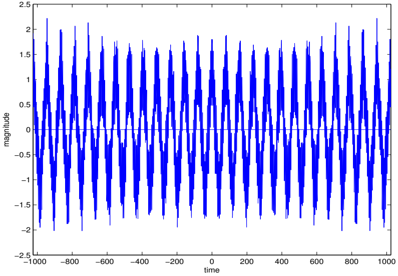
Permute with a random linear transformation and phase shift

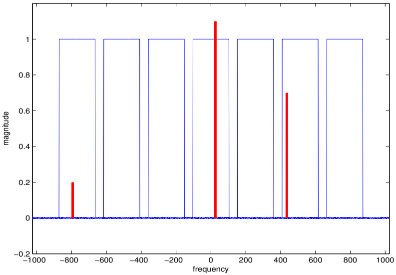




Reducing k-sparse recovery to 1-sparse recovery

Partition frequency space into $B = k/\alpha$ buckets for constant $\alpha \in (0,1)$





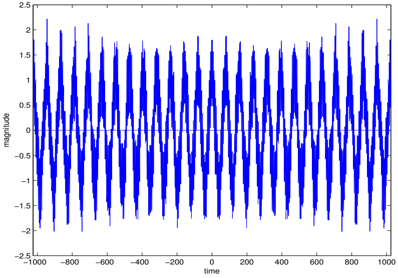
Choose a filter $G, \widehat{G}$ such that

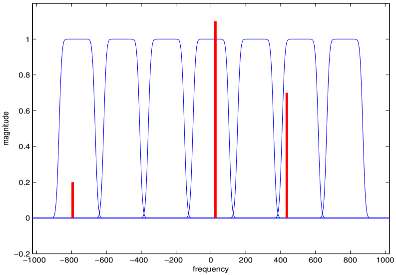
- $\widehat{G}$ approximates the buckets
- G has small support

Compute $\widehat{x} * \widehat{G} = \widehat{(x \cdot G)}$

Reducing k-sparse recovery to 1-sparse recovery

Partition frequency space into $B = k/\alpha$ buckets for constant $\alpha \in (0,1)$





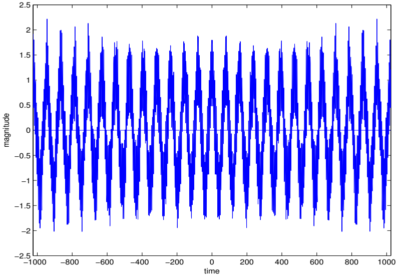
Choose a filter $G, \widehat{G}$ such that

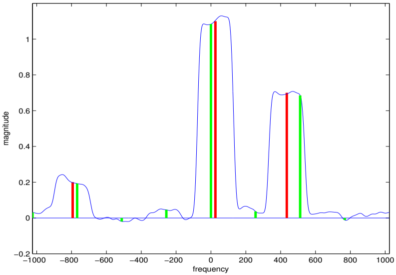
- $\widehat{G}$ approximates the buckets
- G has small support

Compute $\widehat{x} * \widehat{G} = \widehat{(x \cdot G)}$

Reducing k-sparse recovery to 1-sparse recovery

Partition frequency space into $B = k/\alpha$ buckets for constant $\alpha \in (0,1)$





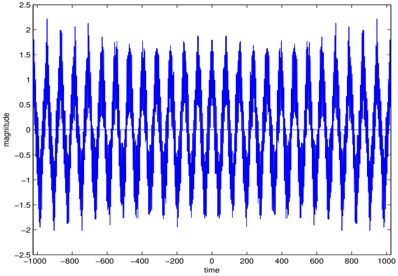
Choose a filter $G, \widehat{G}$ such that

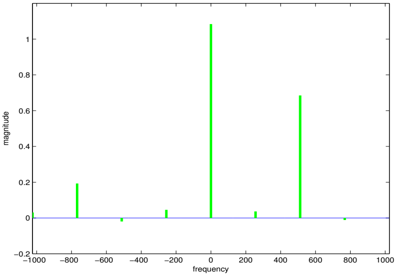
- $\widehat{G}$ approximates the buckets
- G has small support

Compute $\widehat{x} * \widehat{G} = \widehat{(x \cdot G)}$

Reducing k-sparse recovery to 1-sparse recovery

Partition frequency space into $B = k/\alpha$ buckets for constant $\alpha \in (0,1)$





Choose a filter $G, \widehat{G}$ such that

- $\widehat{G}$ approximates the buckets
- G has small support

Compute $\widehat{x} * \widehat{G} = \widehat{(x \cdot G)}$

Sample complexity=supp G!

PartialRecovery step

PartialRecovery(x,$\widehat{y}$)

- Make measurements (independent permutation+filtering)
- Locate and estimate large frequencies (1-sparse recovery)

return dominant Fourier coefficients $\widehat{z}$ of $x-y$ (approximately)

Sample complexity = support of G

PARTIALRECOVERY step

PARTIALRECOVERY$(x, \widehat{y})$

- Make measurements (independent permutation+filtering)
- Locate and estimate large frequencies (1-sparse recovery)

return dominant Fourier coefficients $\widehat{z}$ of $x - y$ (approximately)

Sample complexity = support of G

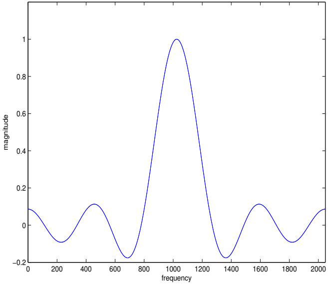
- How many measurements do we need?
- How effective is a refinement step?

Both determined by signal to noise ratio in each bucket – function of filter choice

Time domain:

support $O(k)$ [GMS'05]

Frequency domain:



SNR = O(1)

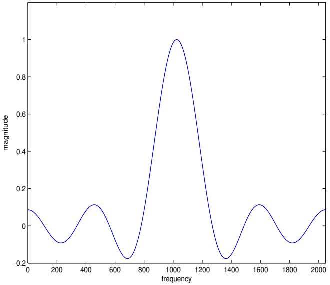
Reduce SNR by $O(1)$ factor

$\Omega(k \log^2 n)$ samples

Time domain:

support $O(k)$ [GMS'05]

Frequency domain:



SNR = O(1)

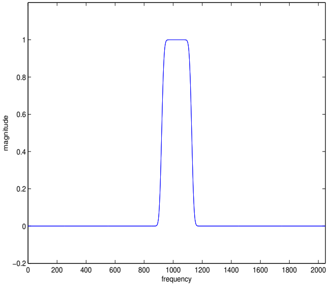
Reduce SNR by $O(1)$ factor

$\Omega(k \log^2 n)$ samples

Time domain:

support $\Theta(k \log n)$ [HIKP12]

Frequency domain:



SNR = can by poly(n)

Reduce sparsity by $O(1)$ factor

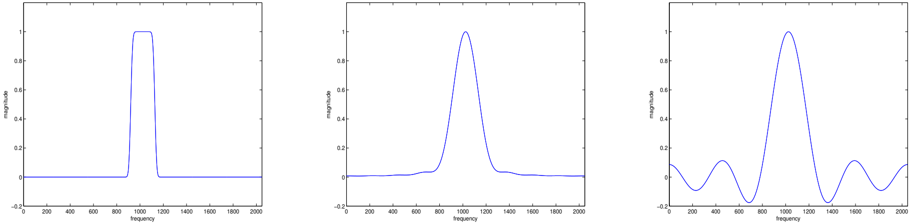
$\Omega(k \log^2 n)$ samples

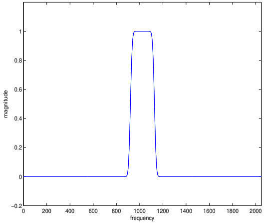
This paper: interpolate between the two extremes, get all benefits

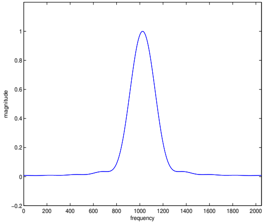
Main idea

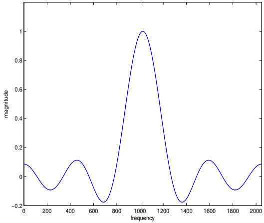
A new family of filters that adapt to current upper bound on SNR.

- Sharp filters initially, more blurred later



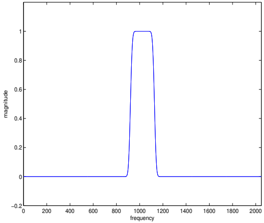


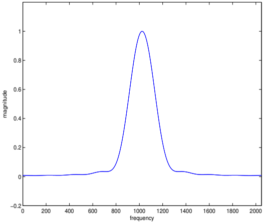


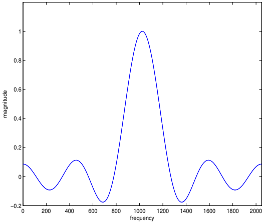


When SNR is bounded by R:

- filter support $O(k \log R)$ ($\approx$ convolve boxcar with itself $\log R$ times)







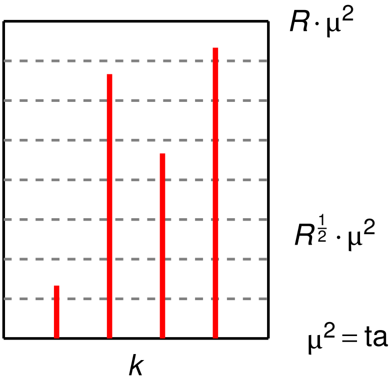
When SNR is bounded by R:

- filter support $O(k \log R)$ ($\approx$ convolve boxcar with itself $\log R$ times)
- (most) 1-sparse recovery subproblems for dominant frequencies have high SNR (about R) so $O^*(\log_R n)$ measurements!

$$O^*(k \log R \cdot \log_R n) = O^*(k \log n) \text{ samples per step!}$$

$$\underbrace{R \to R^{1/2} \to R^{1/4} \to \ldots \to C^2 \to C}_{O(\log\log n)\text{ iterations}}$$

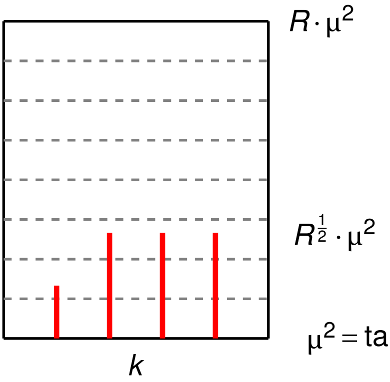
$\textsc{PartialRecovery}(x, \widehat{y}, R)$



$\mu^2 = \text{tail noise}/B$

$$\underbrace{R \to R^{1/2} \to R^{1/4} \to \ldots \to C^2 \to C}_{O(\log\log n) \text{ iterations}}$$

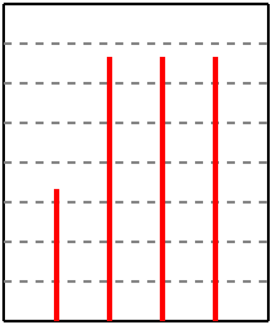
$\textsc{PartialRecovery}(x, \widehat{y}, R)$



$\mu^2 = \text{tail noise}/B$

$$\underbrace{R \to R^{1/2} \to R^{1/4} \to \ldots \to C^2 \to C}_{O(\log\log n)\text{ iterations}}$$

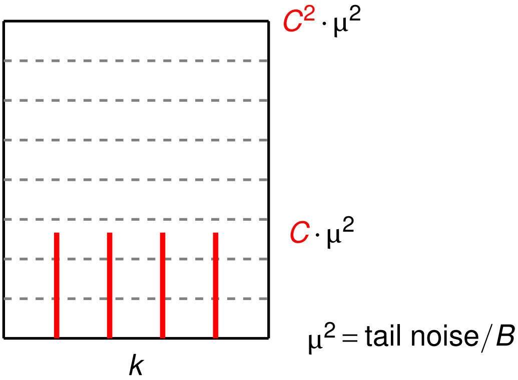
$\textsc{PartialRecovery}(x, \widehat{y}, R^{\frac{1}{2}})$



$\mu^2 = \text{tail noise}/B$

$$\underbrace{R \to R^{1/2} \to R^{1/4} \to \ldots \to C^2 \to C}_{O(\log\log n)\text{ iterations}}$$

PartialRecovery$(x, \widehat{y}, C^2)$



Algorithm

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
$R_0 \leftarrow \mathrm{poly}(n)$
For $t = 1$ to $O(\log\log n)$

$\widehat{z} \leftarrow \textsc{PartialRecovery}(x, \widehat{y}_{t-1}, R_{t-1})$ ▷Takes samples of $x - y$

Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

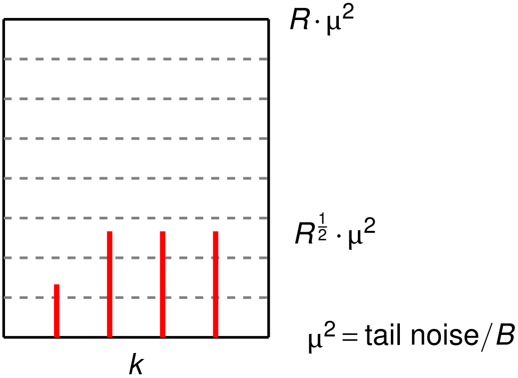
$R_t \leftarrow \sqrt{R_{t-1}}$

PartialRecovery step:

- Takes $O^*(k \log n)$ samples independent of R
- Is very effective: reduces $R \to R^{\frac{1}{2}}$, so $O(\log\log n)$ iterations suffice

Partial recovery analysis

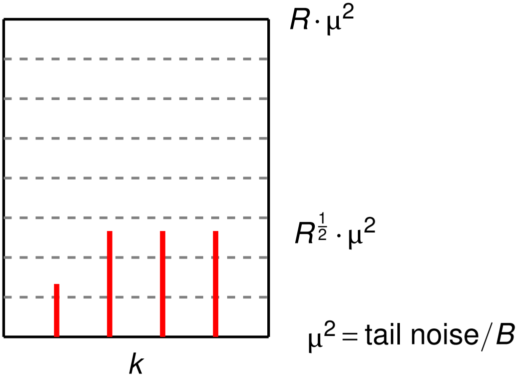
PARTIALRECOVERY($x, \widehat{y}, R$)



- Need to reduce most ‘large’ frequencies, i.e. $|\widehat{x}_i|^2 \geq \sqrt{R}\mu^2$

Partial recovery analysis

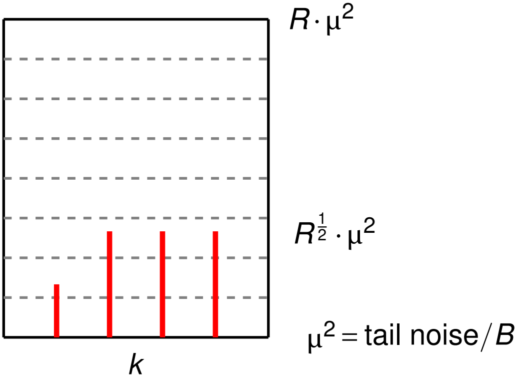
PartialRecovery$(x, \widehat{y}, R)$



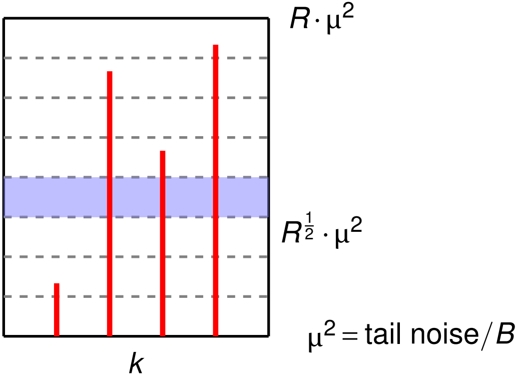
- Need to reduce most ‘large’ frequencies, i.e. $|\widehat{x}_i|^2 \geq \sqrt{R}\mu^2$
- Most=$1 - 1/\text{poly}(R)$ fraction

Partial recovery analysis

PARTIALRECOVERY($x, \widehat{y}, R$)



- Need to reduce most 'large' frequencies, i.e. $|\widehat{x}_i|^2 \geq \sqrt{R}\mu^2$
- Most=$1 - 1/\text{poly}(R)$ fraction
- Iterative process, $O(\log\log n)$ steps



- partition elements into geometric weight classes
- write down recursion that governs the dynamics
- top half classes are reduced at double exponentialy rate* if we use $\Omega(\log \log R)$ levels

Sample optimal algorithm (reusing measurements)

Uniform bounds (for all):

Candes-Tao'06
Rudelson-Vershynin'08
Cheraghchi-Guruswami-Velingker'12
Bourgain'14
Haviv-Regev'15

Deterministic, $\Omega(n)$ runtime

$O(k \log^2 k \log n)$

Non-uniform bounds (for each):

Goldreich-Levin'89
Kushilevitz-Mansour'91, Mansour'92
Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02
Gilbert-Muthukrishnan-Strauss'05
Hassanieh-Indyk-Katabi-Price'12a
Hassanieh-Indyk-Katabi-Price'12b
Indyk-K.-Price'14

Randomized, $O(k \cdot \mathrm{poly}(\log n))$ runtime

$O(k \log n \cdot (\log\log n)^C)$

Lower bound: $\Omega(k \log(n/k))$ for non-adaptive algorithms Do-Ba-Indyk-Price-Woodruff'10

Uniform bounds (for all):

Candes-Tao'06
Rudelson-Vershynin'08
Cheraghchi-Guruswami-Velingker'12
Bourgain'14
Haviv-Regev'15

Deterministic, $\Omega(n)$ runtime

$O(k \log^2 k \log n)$

Non-uniform bounds (for each):

Goldreich-Levin'89
Kushilevitz-Mansour'91, Mansour'92
Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02
Gilbert-Muthukrishnan-Strauss'05
Hassanieh-Indyk-Katabi-Price'12a
Hassanieh-Indyk-Katabi-Price'12b
Indyk-K.-Price'14

Randomized, $O(k \cdot \mathrm{poly}(\log n))$ runtime

$O(k \log n \cdot (\log \log n)^C)$

Lower bound: $\Omega(k \log(n/k))$ for non-adaptive algorithms Do-Ba-Indyk-Price-Woodruff'10

Theorem

There exists an algorithm for ℓ_2/ℓ_2 sparse recovery from Fourier measurements using $O(k \log n)$ samples and $O(n \log^3 n)$ runtime.

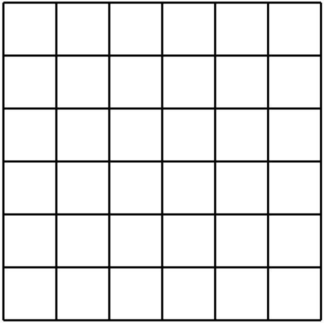
Optimal up to constant factors for $k \le n^{1-\delta}$.

Higher dimensional Fourier transform is needed in some applications

Given $x \in \mathbb{C}^{[n]^d}$, $N = n^d$, compute

$$\widehat{x}_j = \frac{1}{\sqrt{N}} \sum_{i \in [n]^d} \omega^{i^T j} x_i \quad \text{and} \quad x_j = \frac{1}{\sqrt{N}} \sum_{i \in [n]^d} \omega^{-i^T j} \widehat{x}_i$$

where ω is the n-th root of unity, and n is a power of 2.



Previous sample complexity bounds:

- $O(k \log^d N)$ in sublinear time algorithms
  - runtime $k \log^{O(d)} N$, for each
- $O(k \log^4 N)$ for any d
  - $\Omega(N)$ time, for all

This lecture:

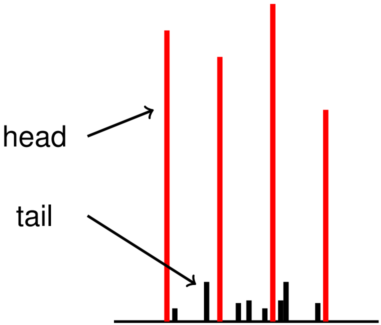
Theorem

There exists an algorithm for ℓ_2/ℓ_2 sparse recovery from Fourier measurements using $O_d(k \log N)$ samples and $O(N \log^3 N)$ runtime.

Sample-optimal up to constant factors for any constant d.

ℓ_2/ℓ_2 sparse recovery guarantees:

$$||\widehat{x}-\widehat{y}||^2 \le C \cdot \min_{k-\text{sparse } \widehat{z}} ||\widehat{x}-\widehat{z}||^2$$



$\mu \approx$ tail noise$/\sqrt{k}$

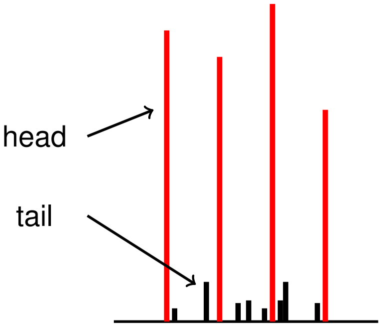
ℓ_2/ℓ_2 sparse recovery guarantees:

$$||\widehat{x}-\widehat{y}||^2 \leq C \cdot \min_{k-\text{sparse } \widehat{z}} ||\widehat{x}-\widehat{z}||^2$$

$|\widehat{x}_1| \geq \ldots \geq |\widehat{x}_k| \geq$
$|\widehat{x}_{k+1}| \geq |\widehat{x}_{k+2}| \geq \ldots$

$\mathrm{Err}_k^2(\widehat{x}) = \sum_{j=k+1}^{n} |\widehat{x}_j|^2$

Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$



$\mu \approx$ tail noise$/\sqrt{k}$

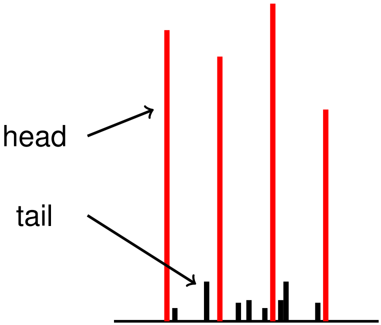
ℓ_2/ℓ_2 sparse recovery guarantees:

$$||\widehat{x} - \widehat{y}||^2 \leq C \cdot \mathrm{Err}_k^2(\widehat{x})$$

$|\widehat{x}_1| \geq \ldots \geq |\widehat{x}_k| \geq$
$|\widehat{x}_{k+1}| \geq |\widehat{x}_{k+2}| \geq \ldots$

$\mathrm{Err}_k^2(\widehat{x}) = \sum_{j=k+1}^n |\widehat{x}_j|^2$

Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$



$\mu \approx$ tail noise$/\sqrt{k}$

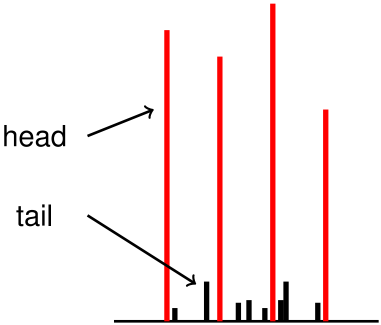
ℓ_2/ℓ_2 sparse recovery guarantees:

$$||\widehat{x}-\widehat{y}||^2 \le C \cdot \mathrm{Err}_k^2(\widehat{x})$$

$|\widehat{x}_1| \ge \ldots \ge |\widehat{x}_k| \ge$
$|\widehat{x}_{k+1}| \ge |\widehat{x}_{k+2}| \ge \ldots$

$\mathrm{Err}_k^2(\widehat{x}) = \sum_{j=k+1}^{n} |\widehat{x}_j|^2$

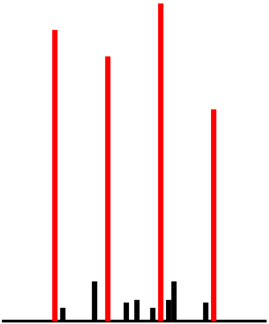
Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$



$\mu \approx$ tail noise$/\sqrt{k}$

ℓ_2/ℓ_2 sparse recovery guarantees:

$$\|\widehat{x}-\widehat{y}\|^2 \leq C \cdot \mathrm{Err}_k^2(\widehat{x})$$



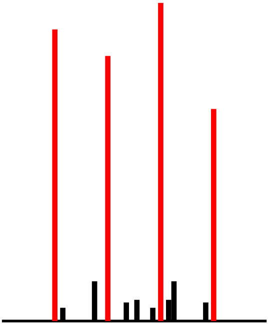
$\mu \approx$ tail noise$/\sqrt{k}$

Sufficient to ensure that most elements are below average noise level:

$$|\widehat{x}_i-\widehat{y}_i|^2 \leq c \cdot \mathrm{Err}_k^2(\widehat{x})/k =: \mu^2$$

ℓ_2/ℓ_2 sparse recovery guarantees:

$$\|\widehat{x} - \widehat{y}\|^2 \leq C \cdot \mathrm{Err}_k^2(\widehat{x})$$



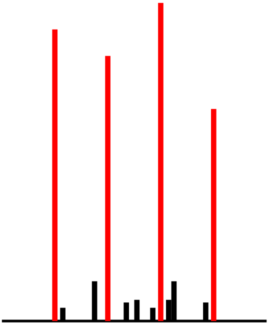
$\mu \approx$ tail noise$/\sqrt{k}$

Will ensure that **all elements** are below average noise level:

$$\|\widehat{x} - \widehat{y}\|_\infty^2 \leq c \cdot \mathrm{Err}_k^2(\widehat{x})/k =: \mu^2$$

ℓ_∞/ℓ_2 sparse recovery guarantees:

$$\|\widehat{x}-\widehat{y}\|_\infty^2 \le C \cdot \mathrm{Err}_k^2(\widehat{x})/k$$



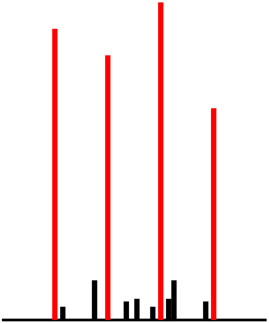
$\mu \approx$ tail noise$/\sqrt{k}$

Will ensure that **all elements** are below average noise level:

$$\|\widehat{x}-\widehat{y}\|_\infty^2 \le c \cdot \mathrm{Err}_k^2(\widehat{x})/k =: \mu^2$$

ℓ_∞/ℓ_2 sparse recovery guarantees:

$$||\hat{x}-\hat{y}||_\infty^2 \leq C \cdot \mathrm{Err}_k^2(\hat{x})/k$$



$\mu \approx$ tail noise$/\sqrt{k}$

Will ensure that **all elements** are below average noise level:

$$||\hat{x}-\hat{y}||_\infty^2 \leq \mu^2$$

Iterative recovery

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

- $\widehat{z} \leftarrow \textsc{PartialRecovery}(x - y_{t-1})$ $\triangleright$ Takes random samples of $x - y$
- Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

Iterative recovery

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

- $\widehat{z} \leftarrow \textsc{PartialRecovery}(x - y_{t-1})$ $\triangleright$ Takes random samples of $x - y$
- Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

In most prior works sampling complexity is

samples per $\textsc{PartialRecovery}$ step $\times$ number of iterations

Iterative recovery

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

- $\widehat{z} \leftarrow \textsc{PartialRecovery}(x - y_{t-1})$ $\triangleright$ Takes random samples of $x - y$
- Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

In most prior works sampling complexity is

samples per $\textsc{PartialRecovery}$ step $\times$ number of iterations

Lots of work on carefully choosing filters, reducing number of iterations:
Hassanieh-Indyk-Katabi-Price'12,
Ghazi-Hassanieh-Indyk-Katabi-Price-Shi'13, Indyk-K.-Price'14

- still lose $\Omega(\log\log n)$ in sample complexity (number of iterations)
- lose $\Omega((\log n)^{d-1} \log\log n)$ in higher dimensions

Iterative recovery

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

- $\widehat{z} \leftarrow \textsc{PartialRecovery}(x - y_{t-1})$ $\triangleright$ Takes random samples of $x - y$
- Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

Our sampling complexity is

samples per $\textsc{PartialRecovery}$ step $\times$ number of iterations

Iterative recovery

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

- $\widehat{z} \leftarrow \textsc{PartialRecovery}(x - y_{t-1})$ $\triangleright$ ~~Takes random samples~~ of $x - y$
- Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

Our sampling complexity is

samples per $\textsc{PartialRecovery}$ step ~~$\times$ number of iterations~~

Iterative recovery

Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

- $\widehat{z} \leftarrow \textsc{PartialRecovery}(x - y_{t-1})$ $\triangleright$ Takes random samples of $x - y$
- Update $\widehat{y}_t \leftarrow \widehat{y}_{t-1} + \widehat{z}$

Our sampling complexity is

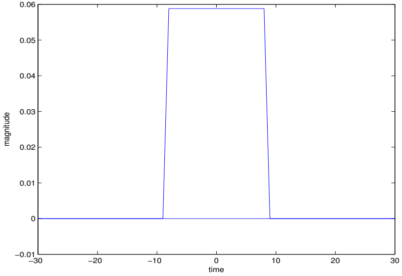
samples per PartialRecovery step ~~× number of iterations~~

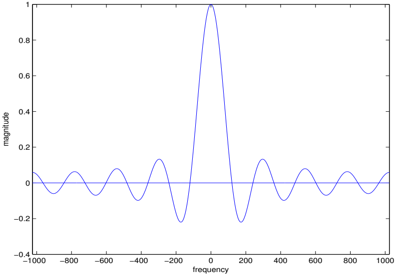
Can use very simple filters!

Our filter=boxcar convolved with itself $O(1)$ times

Filter support is $O(k)$ (=samples per measurement)

$O(k \log n)$ samples in PARTIALRECOVERY step



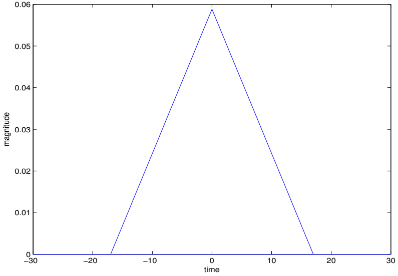


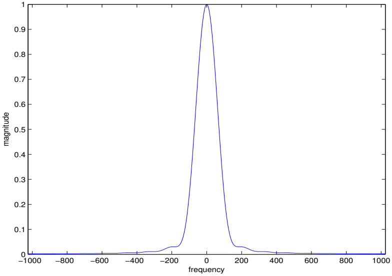
Can choose a rather weak filter, but do not need fresh randomness

Our filter=boxcar convolved with itself $O(1)$ times

Filter support is $O(k)$ (=samples per measurement)

$O(k \log n)$ samples in PARTIALRECOVERY step



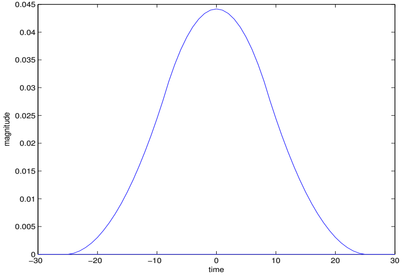


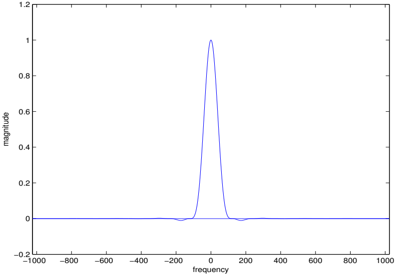
Can choose a rather weak filter, but do not need fresh randomness

Our filter=boxcar convolved with itself $O(1)$ times

Filter support is $O(k)$ (=samples per measurement)

$O(k \log n)$ samples in PARTIALRECOVERY step





Can choose a rather weak filter, but do not need fresh randomness

$G \leftarrow B * B * B$ ▷ Take samples of x

Let $y^m \leftarrow (P_m x) \cdot G$

$m = 0, \ldots, M = C \log n$

$\widehat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$: ▷ Loop over thresholds

For $f \in [n]$:

$\widehat{w}_f \leftarrow \text{median}\left\{\widetilde{y}_f^1, \ldots, \widetilde{y}_f^M\right\}$ ▷ Estimate, prune small elements

If $|\widehat{w}_f| < 2^{T-t}\mu/3$ **then**

$\widehat{w}_f \leftarrow 0$

End

$\widehat{z}_{t+1} = \widehat{z}_t + \widehat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$ ▷ Update samples

for $m = 1, \ldots, M$

End

$G \leftarrow B * B * B$

Let $y^m \leftarrow (P_m x) \cdot G$

$m = 0, \ldots, M = C \log n$

$\widehat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

For $f \in [n]$:

$\widehat{w}_f \leftarrow \text{median}\left\{\widetilde{y}_f^1, \ldots, \widetilde{y}_f^M\right\}$

If $|\widehat{w}_f| < 2^{T-t}\mu/3$ **then**

$\widehat{w}_f \leftarrow 0$

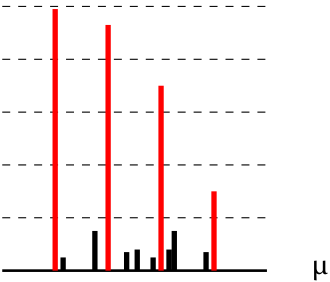
End

$\widehat{z}_{t+1} = \widehat{z}_t + \widehat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$

for $m = 1, \ldots, M$

End



$G \leftarrow B * B * B$

Let $y^m \leftarrow (P_m x) \cdot G$

$\quad m = 0, \ldots, M = C \log n$

$\hat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

For $f \in [n]$:

$\hat{w}_f \leftarrow \text{median}\left\{\tilde{y}_f^1, \ldots, \tilde{y}_f^M\right\}$

If $|\hat{w}_f| < 2^{T-t}\mu/3$ **then**

$\hat{w}_f \leftarrow 0$

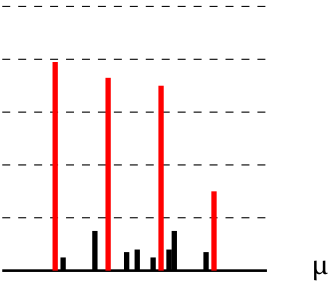
End

$\hat{z}_{t+1} = \hat{z}_t + \hat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$

for $m = 1, \ldots, M$

End



$G \leftarrow B * B * B$

Let $y^m \leftarrow (P_m x) \cdot G$

$m = 0, \ldots, M = C \log n$

$\widehat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

For $f \in [n]$:

$\widehat{w}_f \leftarrow \text{median}\left\{\widetilde{y}_f^1, \ldots, \widetilde{y}_f^M\right\}$

If $|\widehat{w}_f| < 2^{T-t}\mu/3$ **then**

$\widehat{w}_f \leftarrow 0$

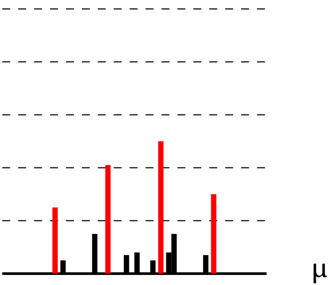
End

$\widehat{z}_{t+1} = \widehat{z}_t + \widehat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$

for $m = 1, \ldots, M$

End



$G \leftarrow B * B * B$

Let $y^m \leftarrow (P_m x) \cdot G$

$m = 0, \ldots, M = C \log n$

$\widehat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

For $f \in [n]$:

$\widehat{w}_f \leftarrow \text{median}\left\{\widetilde{y}_f^1, \ldots, \widetilde{y}_f^M\right\}$

If $|\widehat{w}_f| < 2^{T-t}\mu/3$ **then**

$\widehat{w}_f \leftarrow 0$

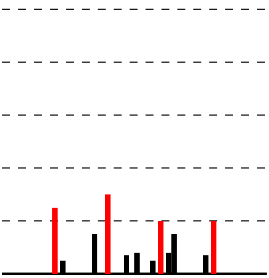
End

$\widehat{z}_{t+1} = \widehat{z}_t + \widehat{w}$

$y^m \leftarrow y^m - (P_m w) \cdot G$

for $m = 1, \ldots, M$

End



μ

$G \leftarrow B * B * B$

Let $y^m \leftarrow (P_m x) \cdot G$

$\quad m = 0, \ldots, M = C \log n$

$\widehat{z}_0 \leftarrow 0$

For $t = 1, \ldots, T = O(\log n)$:

    For $f \in [n]$:

        $\widehat{w}_f \leftarrow \text{median}\left\{\widetilde{y}_f^1, \ldots, \widetilde{y}_f^M\right\}$

        If $|\widehat{w}_f| < 2^{T-t}\mu/3$ **then**

                $\widehat{w}_f \leftarrow 0$

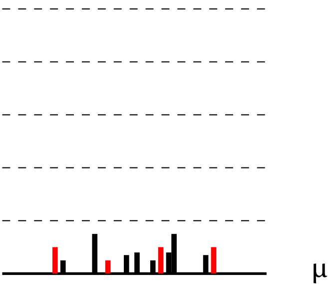
    End

    $\widehat{z}_{t+1} = \widehat{z}_t + \widehat{w}$

    $y^m \leftarrow y^m - (P_m w) \cdot G$

        for $m = 1, \ldots, M$

End

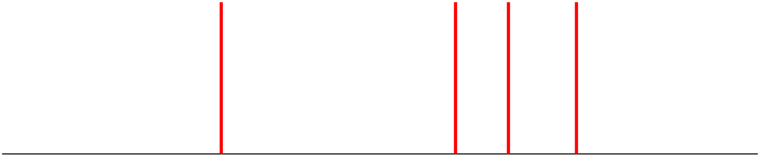


Lecture so far

- Optimal sample complexity by reusing randomness
- Very simple algorithm, can be implemented
- Extension to higher dimensions: algorithm is the same, permutations are different.
    - Choose random invertible linear transformation over $\mathbb{Z}_n^d$

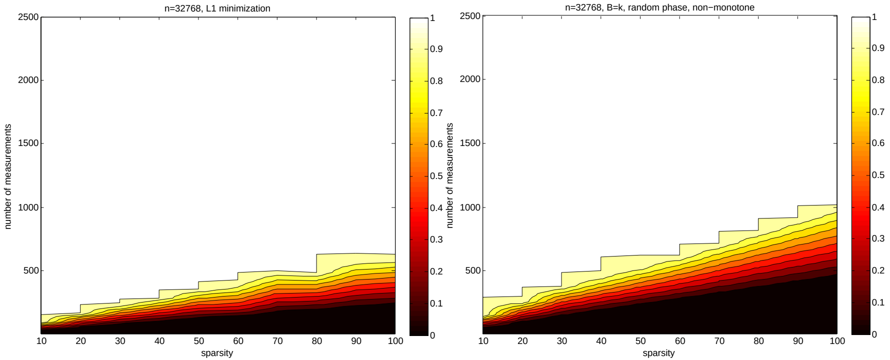
Experimental evaluation

Problem: recover support of a random k-sparse signal from Fourier measurements.
Parameters: $n = 2^{15}$, $k = 10, 20, \ldots, 100$
Filter: boxcar filter with support $k + 1$



Comparison to ℓ_1-minimization (SPGL1)

$O(k \log^3 k \log n)$ sample complexity, requires LP solve



Within a factor of 2 of ℓ_1 minimization

Open questions:

- $O(k \log n)$ in $O(k \log^2 n)$ time?
- $O(k \log n)$ runtime?
- remove dependence on dimension? Current approaches lose C^d in sample complexity, $(\log n)^d$ in runtime

Open questions:

- $O(k \log n)$ in $O(k \log^2 n)$ time?
- $O(k \log n)$ runtime?
- remove dependence on dimension? Current approaches lose C^d in sample complexity, $(\log n)^d$ in runtime

More on sparse FFT:
`http://groups.csail.mit.edu/netmit/sFFT/index.html`