Sample Optimal Fourier Sampling in Any Constant Dimension

Piotr Indyk **Michael Kapralov**

MIT
IBM Watson

October 21, 2014

Discrete Fourier Transform

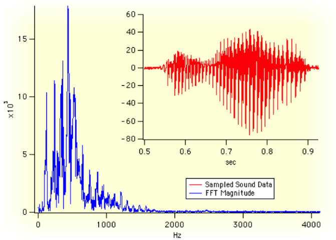
Given $x \in \mathbb{C}^n$, compute

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

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

Fundamental tool:

- Compression (image, audio, video)
- Signal processing
- Data analysis
- Medical imaging (MRI, NMR)



Sparse Fourier Transform

The fast algorithm for DFT is FFT, runs in $O(n \log n)$ time

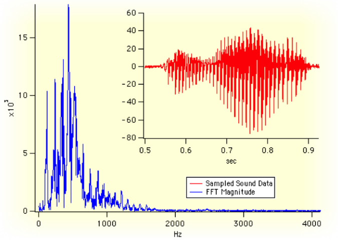
- improving on FFT runtime in full generality a major open problem

Sparse Fourier Transform

The fast algorithm for DFT is FFT, runs in $O(n \log n)$ time

- improving on FFT runtime in full generality a major open problem

Most interesting signals are sparse (have few nonzero entries) or approximately sparse in the Fourier domain.



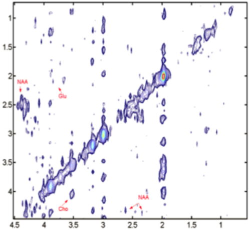
k-sparse=at most k non-zeros

Hassanieh-Indyk-Katabi-Price'12 compute approximate sparse FFT in $O(k \log n \log(n/k))$ time

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

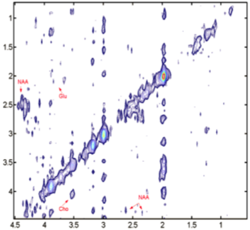


Sample complexity

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
Candes-Plan'10
Cheraghchi-Guruswami-Velingker'12

Deterministic, $\Omega(n)$ runtime

$O(k \log^3 k \log n)$

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
Candes-Plan'10
Cheraghchi-Guruswami-Velingker'12

Deterministic, $\Omega(n)$ runtime

$O(k \log^3 k \log n)$

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

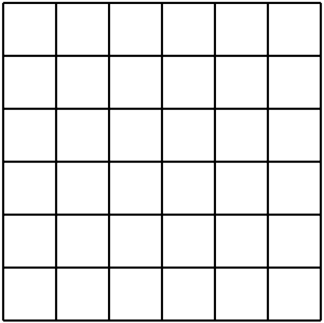
Optimal up to constant factors for $k \leq n^{1-\delta}$. First sample-optimal algorithm even if exponential runtime is allowed.

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

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

Our result:

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, first such algorithm.

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

Our result:

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, first such algorithm.

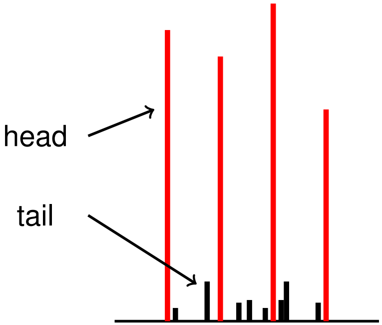
Also: sublinear time recovery, but with $\log \log^2 n$ loss in sample complexity
(extending Indyk-K.-Price'14 to higher dimensions)

Outline of talk:

1. ℓ_2/ℓ_2 sparse recovery guarantee
2. Summary of techniques for recovery from Fourier measurements
3. Sample-optimal algorithm in $O(N\log^3 N)$ time for $d = 1$

ℓ_2/ℓ_2 sparse recovery guarantees:

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



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

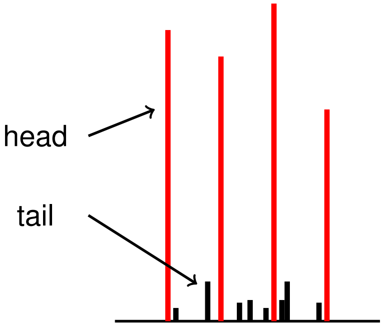
ℓ_2/ℓ_2 sparse recovery guarantees:

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

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

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

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



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

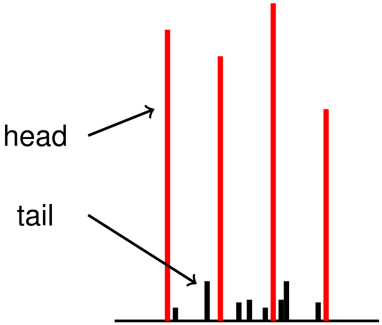
ℓ_2/ℓ_2 sparse recovery guarantees:

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

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

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

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



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

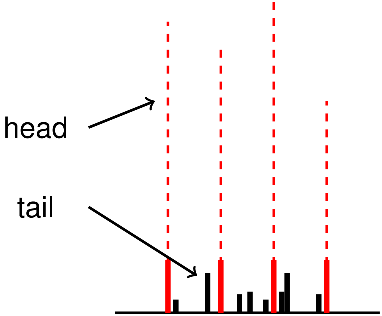
ℓ_2/ℓ_2 sparse recovery guarantees:

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

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

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

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



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

Iterative refinement

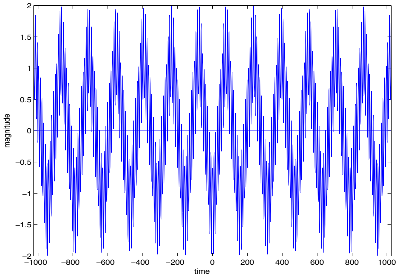
Many algorithms use the iterative refinement scheme:

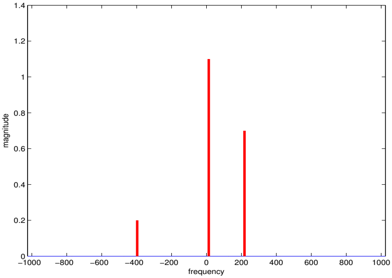
Input: $x \in \mathbb{C}^n$
$\widehat{y}_0 \leftarrow 0$
For $t = 1$ to L

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

$\textsc{Refinement}(x)$
return dominant Fourier coefficients $\widehat{z}$ of x (approximately)

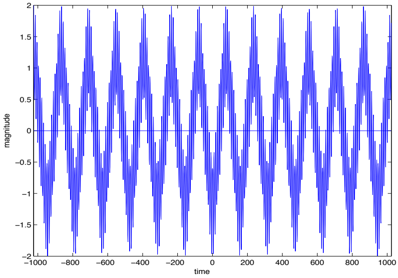
Gilbert-Guha-Indyk-Muthukrishnan-Strauss'02, Akavia-Goldwasser-Safra'03, Gilbert-Muthukrishnan-Strauss'05, Iwen'10, Akavia'10, Hassanieh-Indyk-Katabi-Price'12a, Hassanieh-Indyk-Katabi-Price'12b

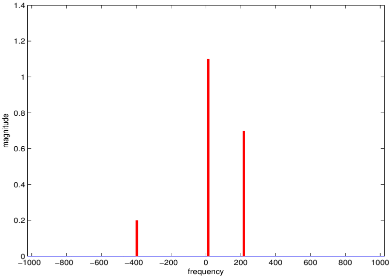

magnitude
time
2
1.5
1
0.5
0
–0.5
–1
–1.5
–2
–1000
–800
–600
–400
–200
0
200
400
600
800
1000


magnitude
frequency
1.4
1.2
1
0.8
0.6
0.4
0.2
0
–1000
–800
–600
–400
–200
0
200
400
600
800
1000

Task:approximate top k coeffs of $\widehat{x}$ using few samples

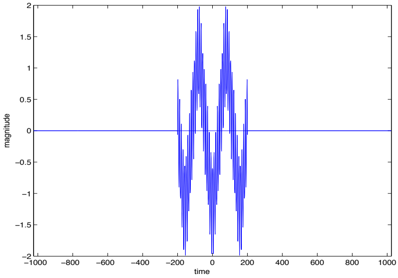
Natural idea: look at the value of the signal on the first $O(k)$ points

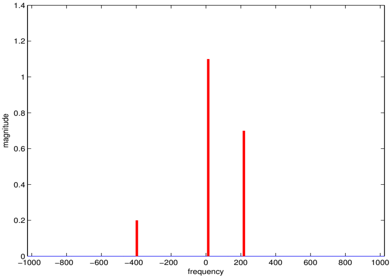




Task:approximate top k coeffs of $\widehat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

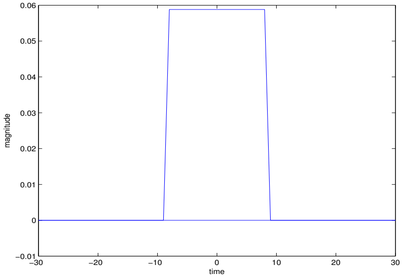


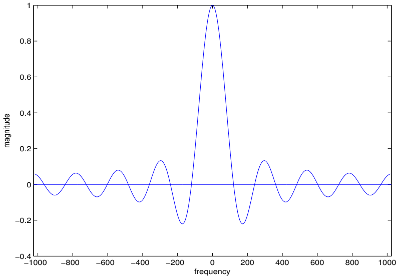


Task:approximate top k coeffs of $\widehat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc: $\widehat{(x \cdot G)} = \widehat{x} * \widehat{G}$

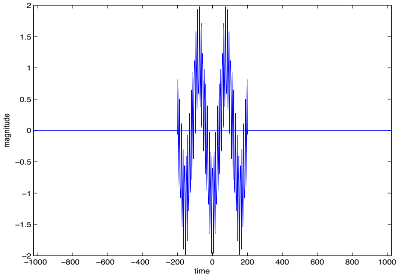


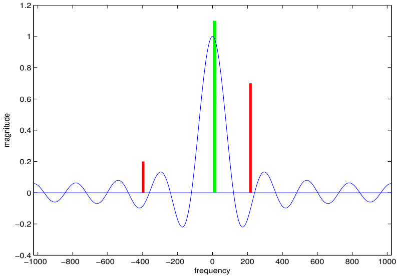


Task:approximate top k coeffs of $\widehat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc: $\widehat{(x \cdot G)} = \widehat{x} * \widehat{G}$



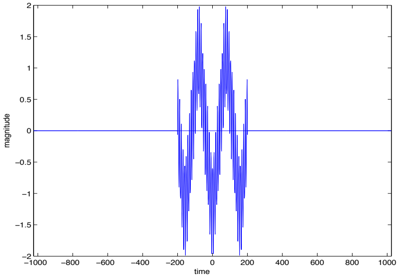


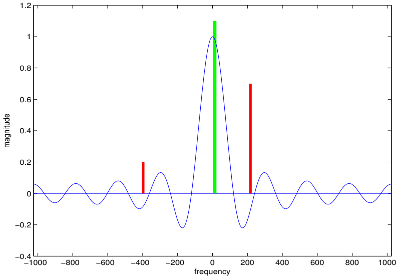
$$\widehat{(G \cdot x)}_f = \sum_{f' \in [n]} \widehat{x}_{f'} \widehat{G}_{f-f'}$$

Task:approximate top k coeffs of $\widehat{x}$ using few samples

Natural idea: look at the value of the signal on the first $O(k)$ points

This convolves spectrum with sinc: $\widehat{(x \cdot G)} = \widehat{x} * \widehat{G}$





$$\widehat{(G \cdot x)}_f = \color{red}{\widehat{x}_f} + \sum_{f' \in [n], f' \neq f} \widehat{x}_{f'} \widehat{G}_{f-f'}$$

REFINEMENT(x)

return dominant Fourier coefficients $\hat{z}$ of x (approximately)

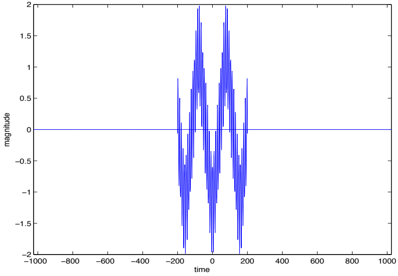
Take $M = C \log n$ independent measurements:

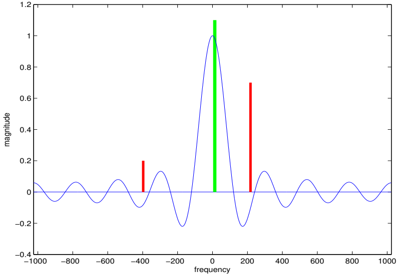
$$y^j \leftarrow (P_{\sigma_j, a_j, q_j} x) \cdot G$$

Estimate each $f \in [n]$ as

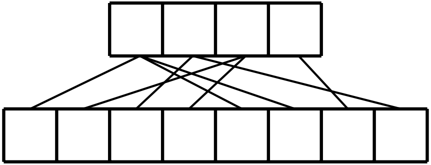
$$\hat{w}_f \leftarrow \text{median}\left\{ \tilde{y}_f^1, \ldots, \tilde{y}_f^M \right\}.$$

Sample complexity= filter support $\times \log n$





Like hashing heavy hitters into buckets (COUNTSKETCH, COUNTMIN), but buckets leak



Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

Iterative refinement

Many algorithms use the iterative refinement scheme:

Input: $x \in \mathbb{C}^n$

$\widehat{y}_0 \leftarrow 0$

For $t = 1$ to L

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

In most prior works sampling complexity is

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

Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

In most prior works sampling complexity is

$$\text{filter support} \times \log n \times \text{number of iterations}$$

Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

In most prior works sampling complexity is

$$\text{filter support} \times \log n \times \text{number of iterations}$$

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 refinement

Many algorithms use the iterative refinement scheme:

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

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

Our sampling complexity is

$$\text{filter support} \times \log n \times \text{number of iterations}$$

Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

Our sampling complexity is

$$\text{filter support} \times \log n \times \cancel{\times \text{ number of iterations}}$$

Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

Our sampling complexity is

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

Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

Our sampling complexity is

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

Do not use fresh randomness in each iteration! In general challenging: only one paper Bayati-Montanari'11 gives provable guarantees, with Gaussians

Iterative refinement

Many algorithms use the iterative refinement scheme:

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

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

Our sampling complexity is

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

Do not use fresh randomness in each iteration! In general challenging: only one paper Bayati-Montanari'11 gives provable guarantees, with Gaussians

Can use very simple filters! (Essentially) boxcar filter

$G \leftarrow B * B * B$ ▷ 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**

$\widehat{w}_f \leftarrow 0$

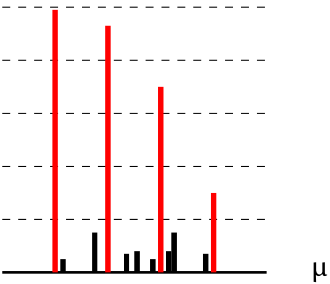
End

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

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

for $m = 1, \ldots, M$

End



$G \leftarrow B * B * B$

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

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

$\widehat{z}_0 \leftarrow 0$

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**

$\widehat{w}_f \leftarrow 0$

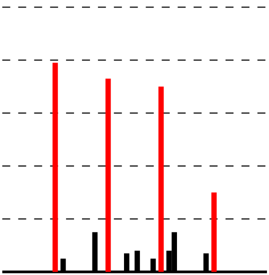
End

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

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

for $m = 1, \ldots, M$

End



μ

$G \leftarrow B * B * B$

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

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

$\widehat{z}_0 \leftarrow 0$

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**

$\widehat{w}_f \leftarrow 0$

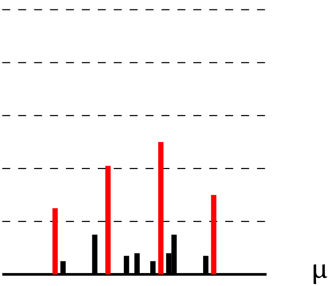
End

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

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

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**

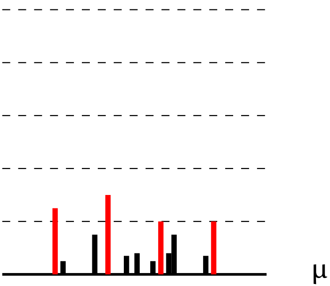
$\widehat{w}_f \leftarrow 0$

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



$G \leftarrow B * B * B$

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**

$\widehat{w}_f \leftarrow 0$

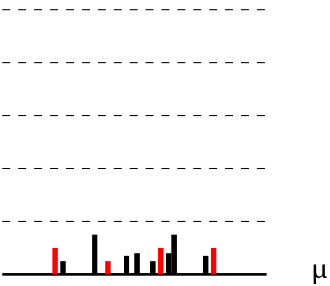
End

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

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

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**

$\widehat{w}_f \leftarrow 0$

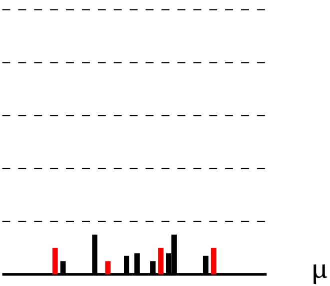
End

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

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

for $m = 1, \ldots, M$

End

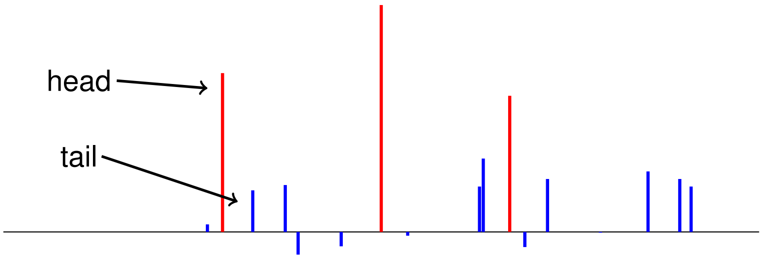


Main challenge: lack of fresh randomness. Why does median work?

Let S denote the set of heavy hitters:

$$S = \{i \in [n] : |\widehat{x}_i| > \mu\}.$$

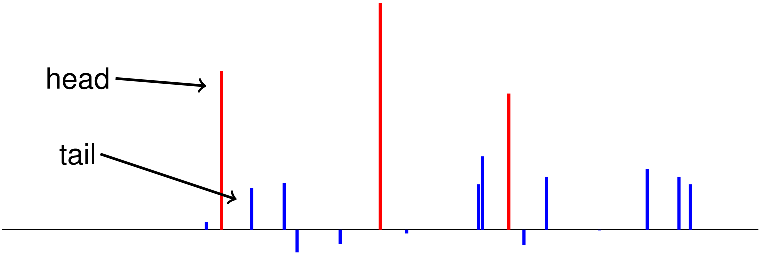
There cannot be too many of them: $|S| = O(k)$



Let S denote the set of heavy hitters:

$$S = \{i \in [n] : |\widehat{x}_i| > \mu\}.$$

There cannot be too many of them: $|S| = O(k)$



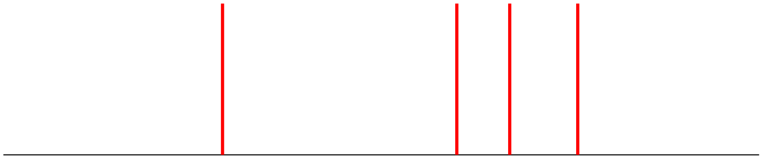
Main invariant: never modify $\widehat{x}$ outside of S

Prove: samples taken have error-correcting properties wrt set S of head elements

Set of head elements does not change, only their values

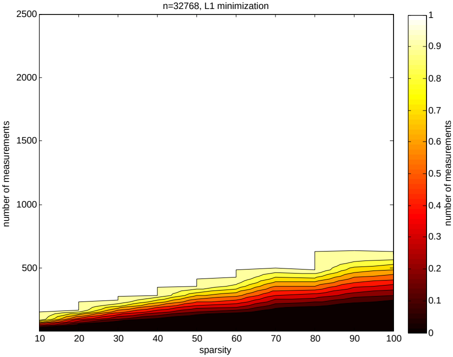
Experimental evaluation

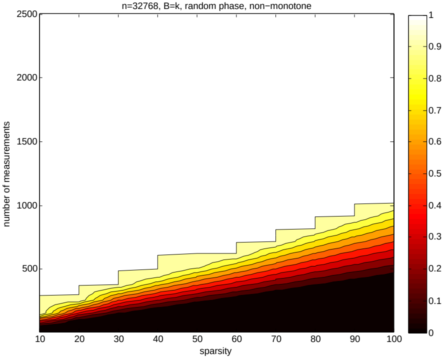
Problem: recover support of a random k-sparse signal from Fourier measurements.
Parameters: $n = 2^{15}$, $k = 10, 20, \ldots, 100$
Filter: boxcar filter with support $k + 1$



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

$O(k \log n)$ samples in $O(k \log^{O(1)} n)$ time?

Thank you!