Sparse Fourier Transform (lecture 3)

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:

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

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:

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

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 last lecture:

- 1-sparse noiseless case: two-point sampling

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

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

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 last lecture:

- 1-sparse noiseless case: two-point sampling
- 1-sparse noisy case: $O(\log n \log\log n)$ time and samples

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

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

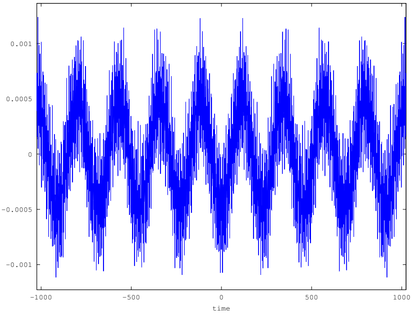
where $\omega = e^{2\pi i/n}$ is the n-th root of unity.

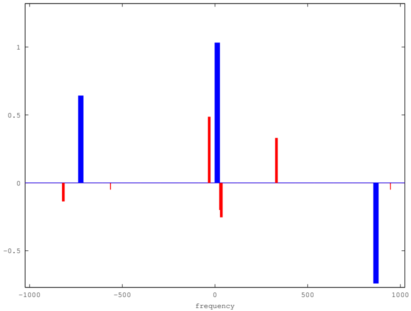
Goal: find the top k coefficients of $\widehat{x}$ approximately

In last lecture:

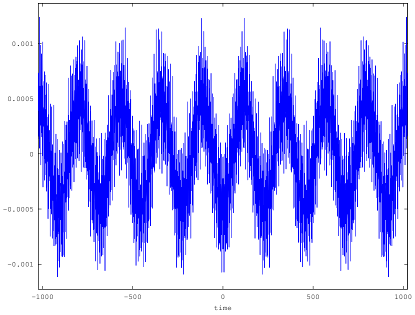
- 1-sparse noiseless case: two-point sampling
- 1-sparse noisy case: $O(\log n \log\log n)$ time and samples
- reduction from k-sparse to 1-sparse case, via filtering

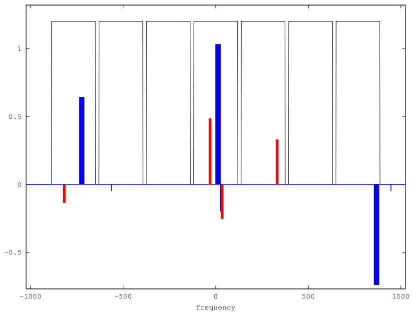
Partition frequency domain into $B \approx k$ buckets



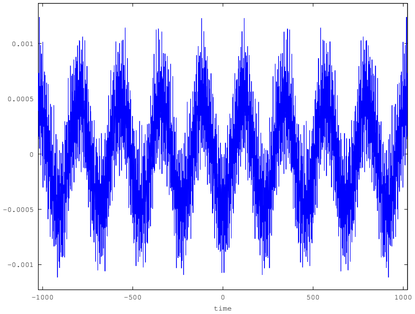


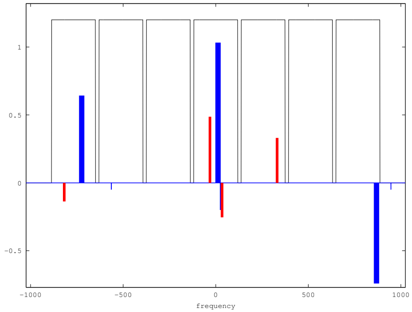
Partition frequency domain into $B \approx k$ buckets





Partition frequency domain into $B \approx k$ buckets



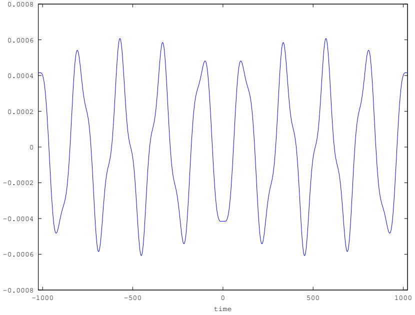


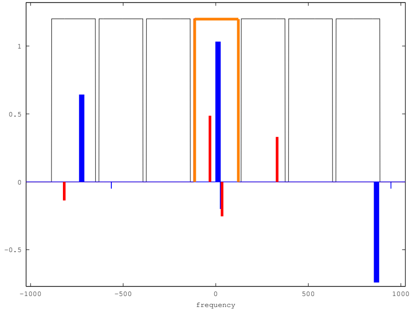
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



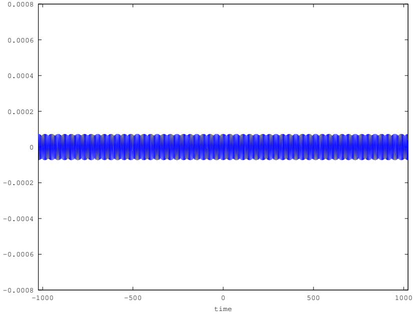


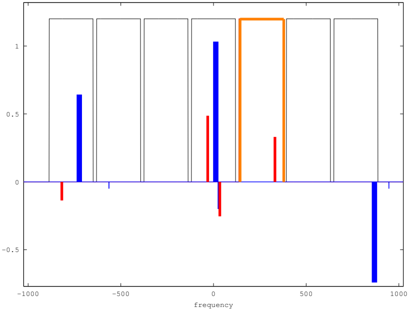
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



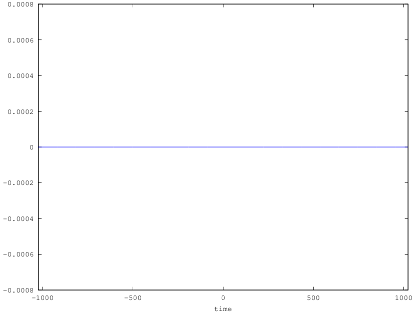


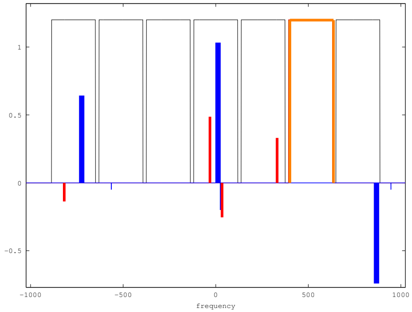
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



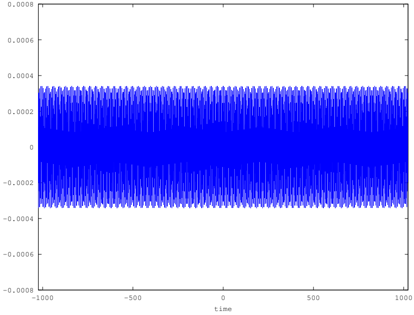


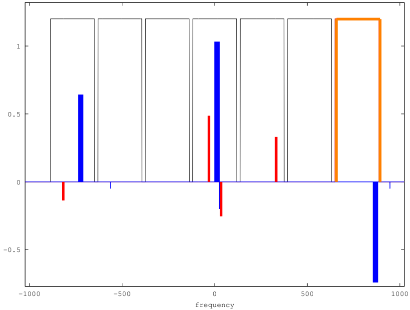
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



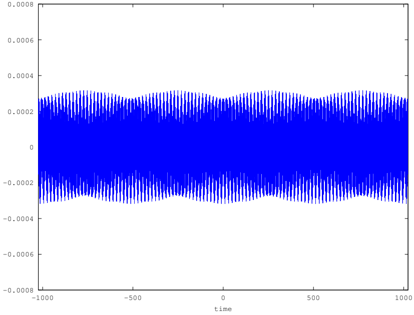


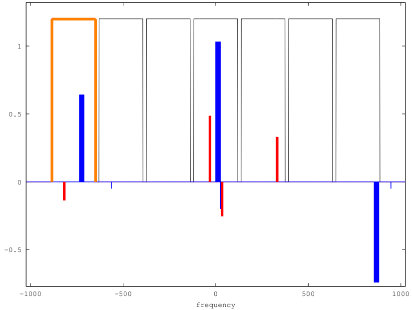
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



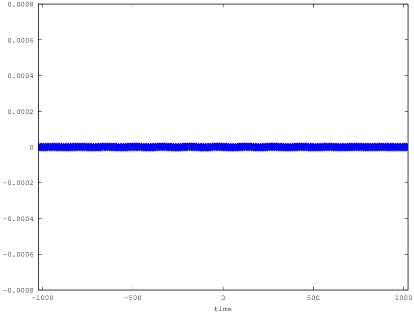


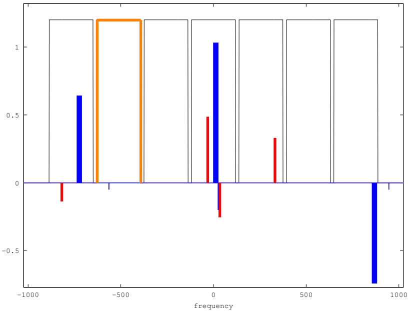
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets



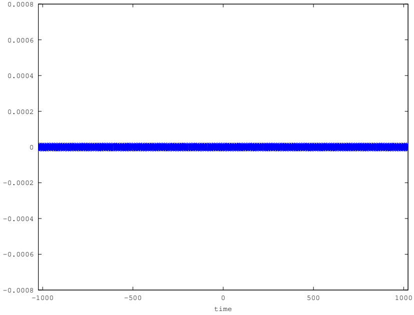


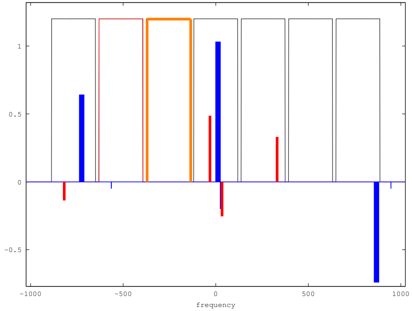
For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

Partition frequency domain into $B \approx k$ buckets





For each $j = 0, \ldots, B-1$ let

$$\widehat{u}^j_f = \begin{cases} \widehat{x}_f, & \text{if } f \in j\text{-th bucket} \\ 0 & \text{o.w.} \end{cases}$$

Restricted to a bucket, signal is likely approximately 1-sparse!

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_{-\frac{n}{2B} \le f \le \frac{n}{2B}} \widehat{x}_f \cdot \omega^{f \cdot a}.$$

Let

$$\widehat{G}_f = \begin{cases} 1, & \text{if } f \in \left[-\frac{n}{2B} : \frac{n}{2B}\right] \\ 0 & \text{o.w.} \end{cases}$$

Then

$$u^0_a = (\widehat{x_{\cdot + a}} * \widehat{G})(0)$$

We want time domain access to u^0: for any $a = 0, \ldots, n-1$, compute

$$u^0_a = \sum_{-\frac{n}{2B} \leq f \leq \frac{n}{2B}} \widehat{x}_f \cdot \omega^{f \cdot a}.$$

Let

$$\widehat{G}_f = \begin{cases} 1, & \text{if } f \in [-\frac{n}{2B} : \frac{n}{2B}] \\ 0 & \text{o.w.} \end{cases}$$

Then

$$u^0_a = (\widehat{x_{\cdot + a}} * \widehat{G})(0)$$

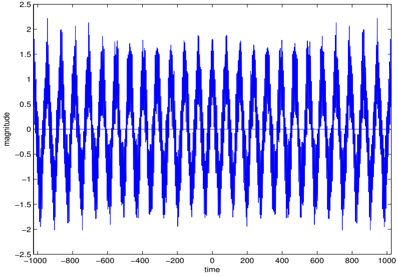
For any $j = 0, \ldots, B-1$

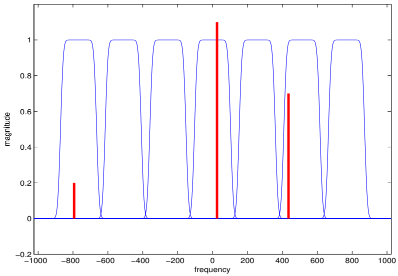
$$u^j_a = (\widehat{x_{\cdot + a}} * \widehat{G})(j \cdot \frac{n}{B})$$

Reducing k-sparse recovery to 1-sparse recovery

For any $j = 0, \ldots, B-1$

$$u_a^j = (\widehat{x_{\cdot+a}} * \widehat{G})(j \cdot \frac{n}{B})$$

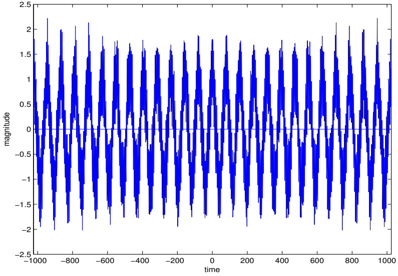


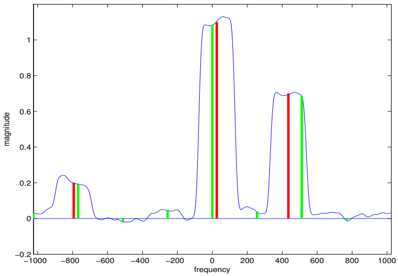


Reducing k-sparse recovery to 1-sparse recovery

For any $j = 0, \ldots, B-1$

$$u_a^j = (\widehat{x_{\cdot+a}} * \widehat{G})(j \cdot \frac{n}{B})$$

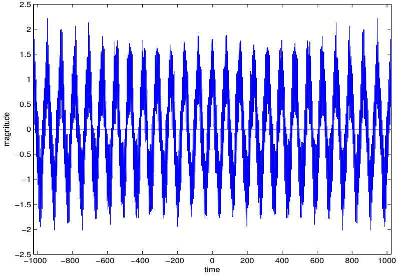


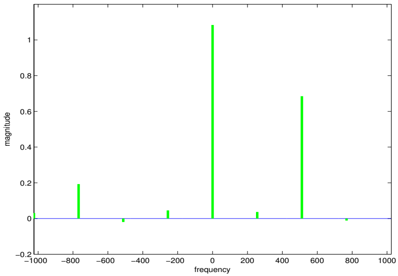


Reducing k-sparse recovery to 1-sparse recovery

For any $j = 0, \ldots, B-1$

$$u_a^j = (\widehat{x_{\cdot+a}} * \widehat{G})(j \cdot \frac{n}{B})$$





Need to evaluate

$$(\widehat{x}_{\cdot+a} * \widehat{G})\left(j \cdot \frac{n}{B}\right)$$

for $j = 0, \ldots, B-1$.

We have access to x, not $\widehat{x}$...

Need to evaluate

$$(\widehat{x}_{\cdot+a} * \widehat{G})\left(j \cdot \frac{n}{B}\right)$$

for $j = 0, \ldots, B-1$.

We have access to x, not $\widehat{x}$...

By the convolution identity

$$\widehat{x}_{\cdot+a} * \widehat{G} = \widehat{(x_{\cdot+a} \cdot G)}$$

Need to evaluate

$$(\widehat{x}_{\cdot+a} * \widehat{G})\left(j \cdot \frac{n}{B}\right)$$

for $j = 0, \ldots, B-1$.

We have access to x, not $\widehat{x}$...

By the convolution identity

$$\widehat{x}_{\cdot+a} * \widehat{G} = (\widehat{x_{\cdot+a} \cdot G})$$

Suffices to compute

$$\widehat{x_{\cdot+a} \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B-1$$

Suffices to compute

$$\widehat{x_{\cdot+a} \cdot G}_{j\cdot\frac{n}{B}}, j = 0, \ldots, B-1$$

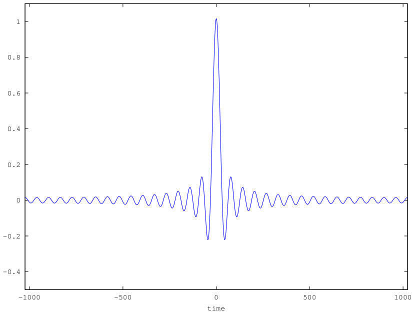
Suffices to compute

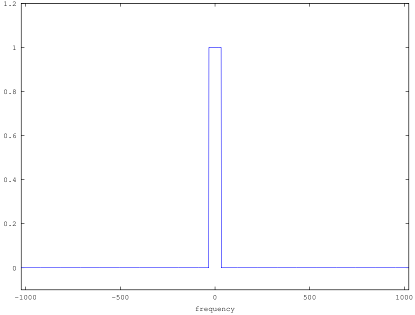
$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B-1$$

Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B-1$$

Sample complexity? Runtime?

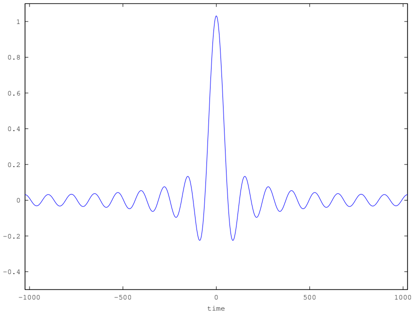


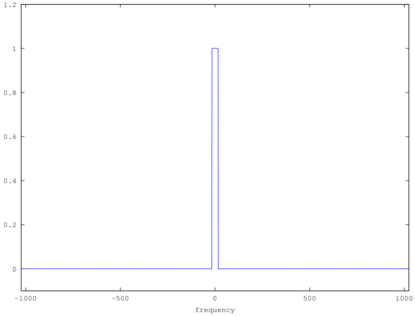


Suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B-1$$

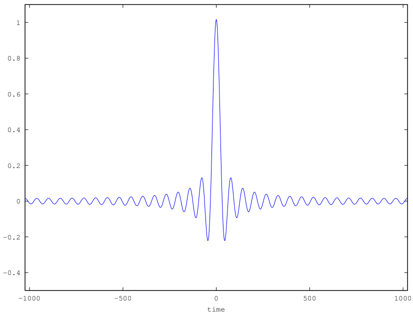
Sample complexity? Runtime?

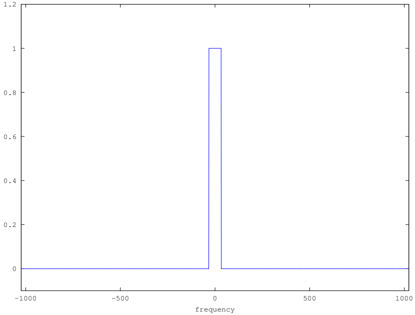




To sample all signals $u^j, j = 0, \ldots, B-1$ in time domain, it suffices to compute

$$\widehat{x \cdot G}_{j \cdot \frac{n}{B}}, j = 0, \ldots, B-1$$





Computing $x \cdot G$ takes $\text{supp}(G)$ samples.

Design G with $\text{supp}(G) \approx k$ that approximates rectangular filter?

Last lecture: designed G with $\text{supp}(G) = O(k \log N)$ that approximates rectangular filter

In this lecture:

- recovery algorithm (k-sparse noiseless case)
- recovery algorithm (k-sparse noisy case)

Hassanieh-Indyk-Katabi-Price'STOC12

1. **Basic block: partial recovery**
2. Full algorithm

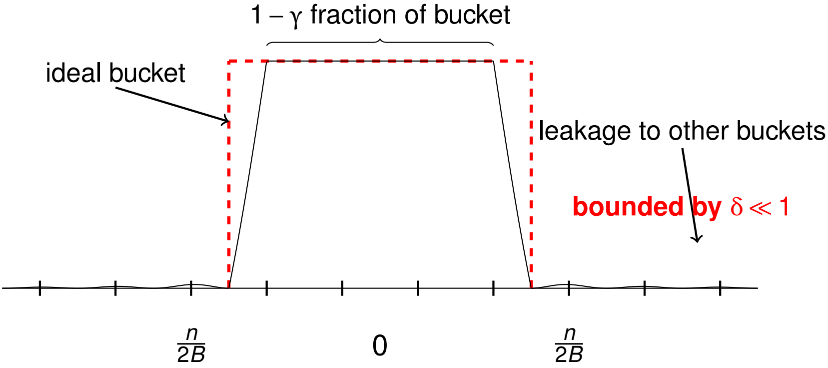
Basic block

Assume

- n is a power of 2
- $\widehat{x}$ contains at most k coefficients with polynomial precision (e.g. $\widehat{x}_f$ in $\{-n^{O(1)}, \ldots, n^{O(1)}\}$)

Then there exists an $O(k \log n)$ time algorithm that

- outputs at most k potential coefficients
- outputs each nonzero $\widehat{x}_f$ correctly with probability at least $1 - \beta$ for a constant $\beta > 0$



Let G be a $(B, \delta/n, \gamma)$-flat window function:

- B buckets
- flat region of width $1 - \gamma$
- leakage $\leq \delta/n = 1/n^{O(1)}$

Such G can be constructed with

$$\text{supp}(G) = O((k/\gamma)\log n)$$

PartialRecovery – algorithm

Main idea: filter, then run 1-sparse algorithm on each subproblem

PartialRecovery(x, B, γ, δ)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \dots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$

$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

Run 1-sparse decoding one $\widehat{c}', \widehat{c}''$

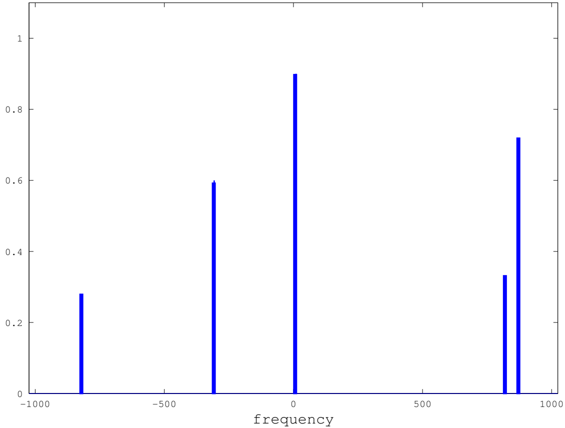
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

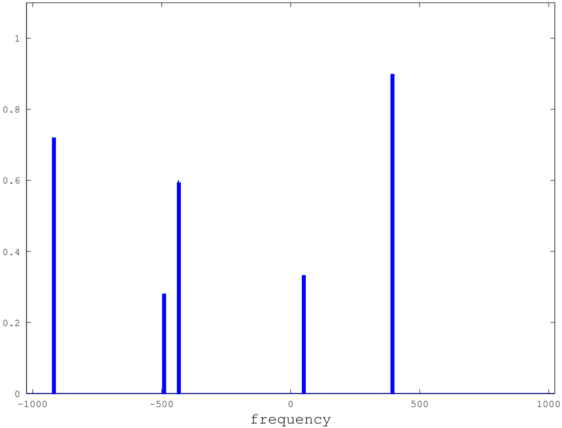
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

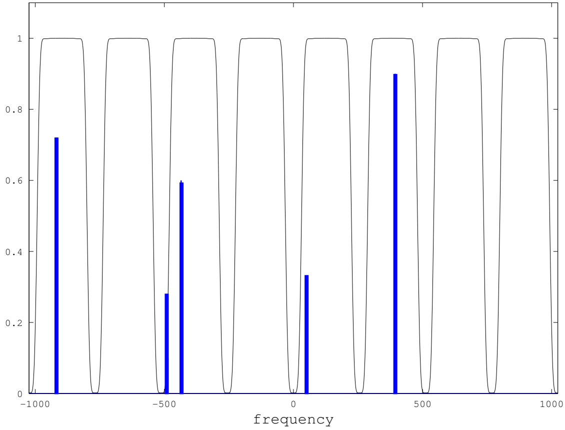
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

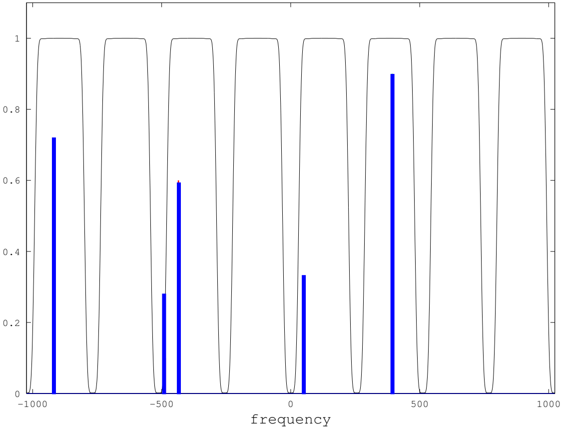
PARTIALRECOVERY – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

PartialRecovery – algorithm

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$
$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$
If $|\widehat{c}'_{j \cdot n/B}| > 1/2$
Decode from $\widehat{c}'_{j \cdot n/B}, \widehat{c}''_{j \cdot n/B}$
(Two-point sampling)
End
End

Basic block – analysis

Claim

For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B + \gamma)$.

Basic block – analysis

Claim

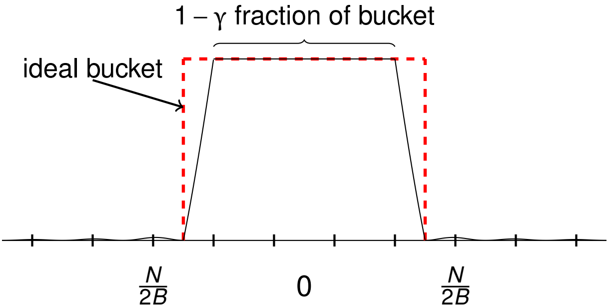
For each $u \in supp(\hat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

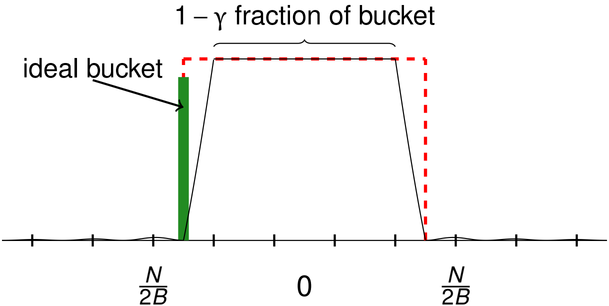
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

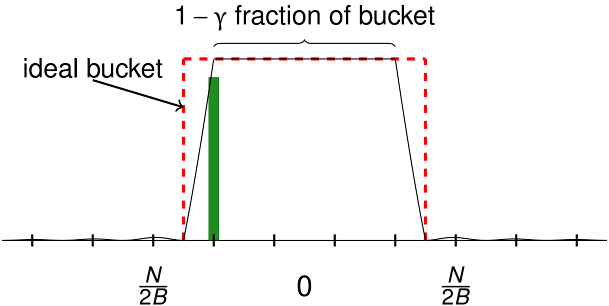
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

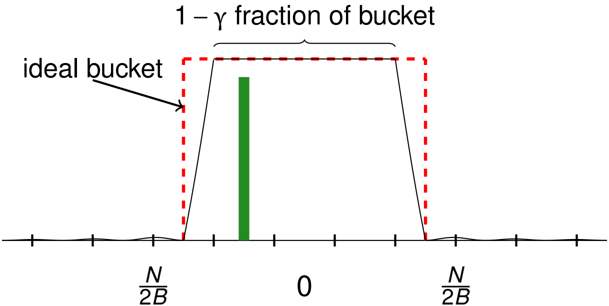
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

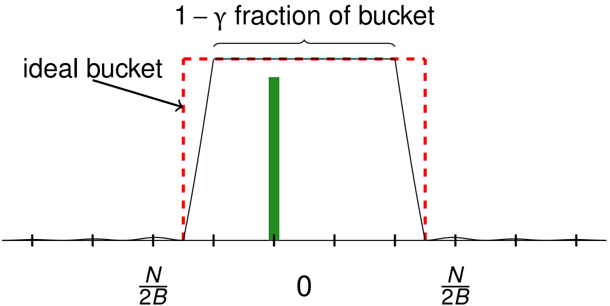
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B + \gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

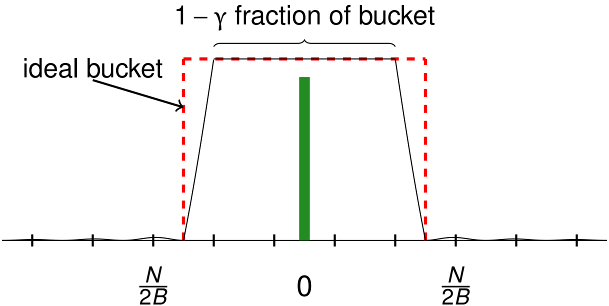
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B + \gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

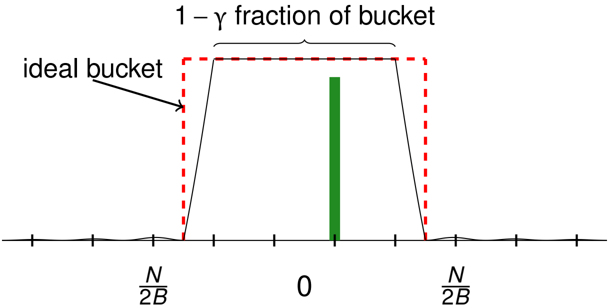
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

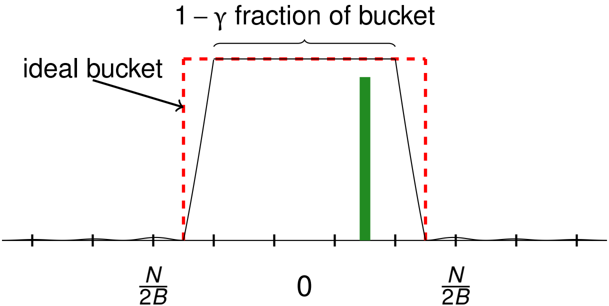
For each $u \in supp(\widehat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

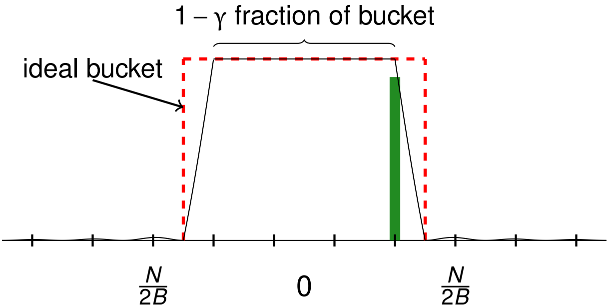
For each $u \in supp(\hat{x})$ the probability that u is not reported is bounded by $O(k/B + \gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Basic block – analysis

Claim

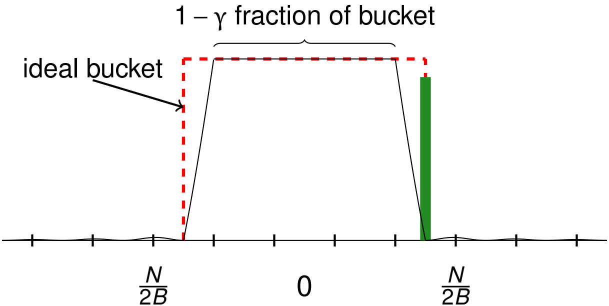
For each $u \in supp(\hat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Proof.

Probability of being mapped

- within n/B of another frequency is $O(k/B)$
- close to boundary of the bucket is $O(\gamma)$

□



Computing $\widehat{c}_{j \cdot n/B}$

Option 1 – directly compute FFT of $(x \cdot G)_{-T}, \ldots, (x \cdot G)_T$, $T = O((k/\gamma)\log n)$

- Can be done in time $O((k/\gamma)\log^2 n)$
- Computes too many samples of $\widehat{x} * \widehat{G}$

Computing $\hat{c}_{j \cdot n/B}$

Option 1 – directly compute FFT of $(x \cdot G)_{-T}, \dots, (x \cdot G)_T$, $T = O((k/\gamma)\log n)$

- Can be done in time $O((k/\gamma)\log^2 n)$
- Computes too many samples of $\hat{x} * \hat{G}$

Option 2 – alias $x \cdot G$ to B bins first

- Compute
$$b_i = \sum_{j \in [n/B]} x_{i+j \cdot B} G_{i+j \cdot B}$$
- Compute FFT of b in time
$$O(B \log B) = O((k/\gamma)\log n)$$

1. Basic block: partial recovery
2. **Full algorithm**

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k \quad, \frac{1}{16} \qquad, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k \quad, \frac{1}{16} \qquad, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n))$

…

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k \quad, \frac{1}{16} \qquad, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k \quad, \frac{1}{16} \qquad , 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\text{PartialRecovery}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\text{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\text{PartialRecovery}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

$\text{PartialRecovery}(x, 10 \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, 10 \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, 10 \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n))$

...

Full algorithm

Permute spectrum

Hash to 8 buckets

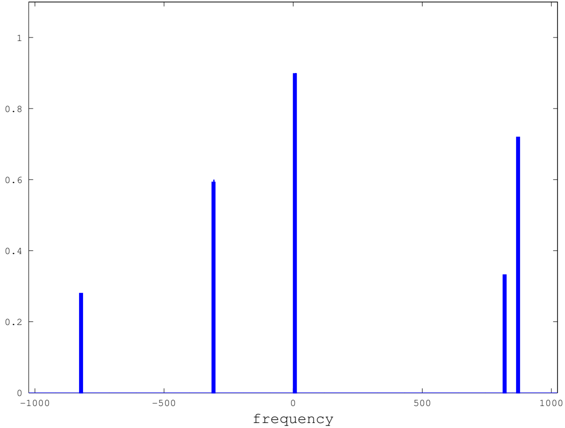
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

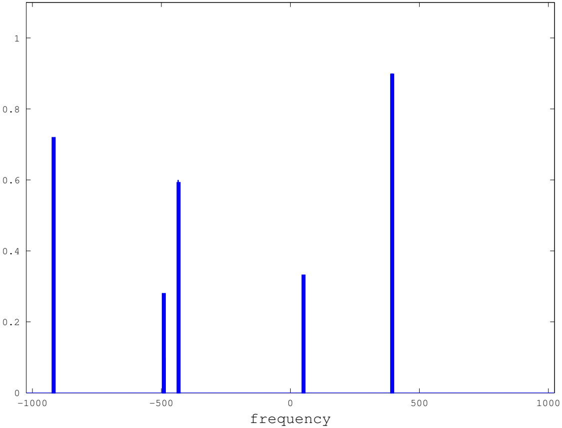
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

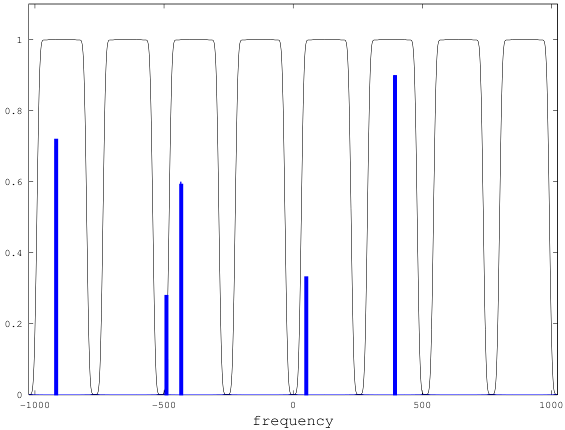
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

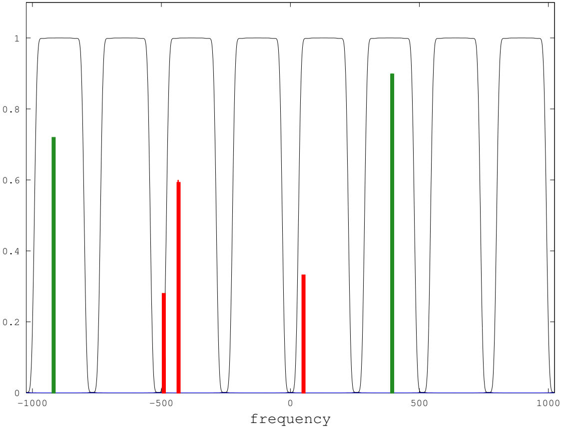
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

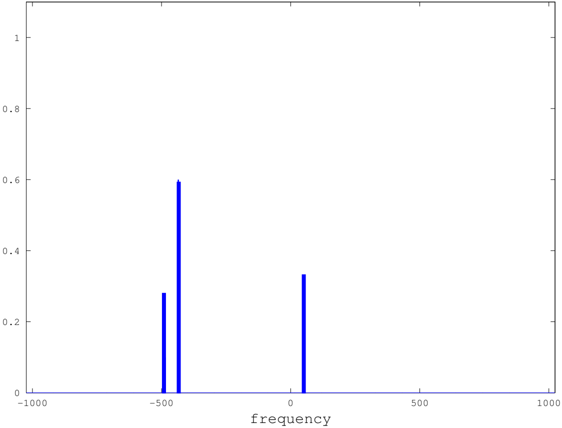
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

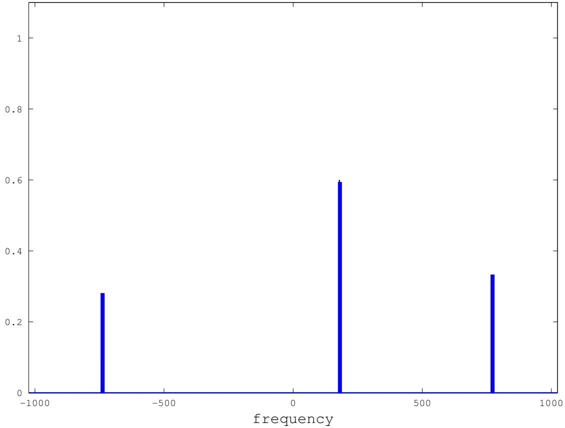
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

…



Full algorithm

Permute spectrum

Hash to 8 buckets

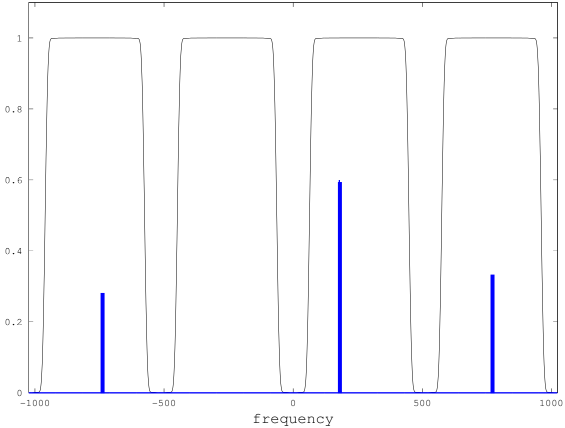
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

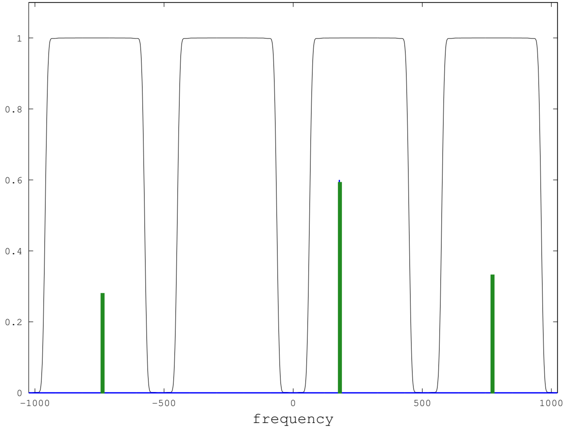
Recover isolated coeffs

Permute spectrum

Hash to 4 buckets

Recover isolated coeffs

…



Modified PartialRecovery

PartialRecovery($B, \alpha, List$)

Choose random b, odd σ

Define $x'_j = x_{\sigma j}\omega^{jb}$
$x''_j = x'_{j+1}$

Compute $\widehat{c}'_{j\cdot\frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j\cdot\frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$
  If $|\widehat{c}'_{j\cdot n/B}| > 1/2$
    Decode from $\widehat{c}'_{j\cdot n/B}, \widehat{c}''_{j\cdot n/B}$
    (Two-point sampling)
  End
End

PartialRecovery – updating the bins

Previously located elements are still in the signal...

Subtract recovered elements from the bins

For each $(pos, val) \in List$

$u \leftarrow \sigma \cdot pos - b$

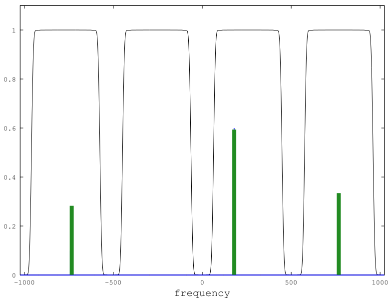
$j \leftarrow$ closest bin to u

$off \leftarrow u - jn/B$

$\widehat{c}'_{j \cdot n/B} \leftarrow \widehat{c}'_{j \cdot n/B} - val \cdot \widehat{G}_{off}$

$\widehat{c}''_{j \cdot n/B} \leftarrow \widehat{c}''_{j \cdot n/B} - val \cdot \omega^u \cdot \widehat{G}_{off}$

End



PartialRecovery – updating the bins

Previously located elements are still in the signal...

Subtract recovered elements from the bins

For each $(pos, val) \in List$

$u \leftarrow \sigma \cdot pos - b$

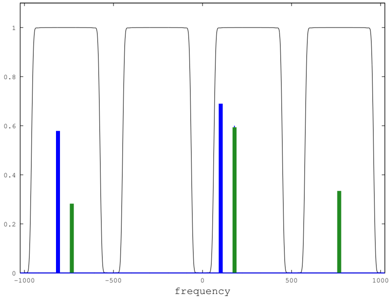
$j \leftarrow$ closest bin to u

$off \leftarrow u - jn/B$

$\widehat{c}'_{j \cdot n/B} \leftarrow \widehat{c}'_{j \cdot n/B} - val \cdot \widehat{G}_{off}$

$\widehat{c}''_{j \cdot n/B} \leftarrow \widehat{c}''_{j \cdot n/B} - val \cdot \omega^{u} \cdot \widehat{G}_{off}$

End



Full algorithm

List $\leftarrow \emptyset$
For $t = 0$ **to** $\log k$
$B_t \leftarrow Ck/4^t$ ▷ # of buckets to hash to
$\gamma_t \leftarrow 1/(C2^t)$ ▷ sharpness of filter
$List \leftarrow List + \textsc{PartialRecovery}(B_t, \gamma_t, List)$
End

Full algorithm – analysis

Let

$$\widehat{e}_t \leftarrow \text{contents of the list after stage } t.$$

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \le k/8^t \right\}$$

Conditional on $\mathscr{E}_{t-1}$, for every $f \in [n]$ the probability of failure to recover is at most the sum of

Full algorithm – analysis

Let

$$\hat{e}_t \leftarrow \text{contents of the list after stage } t.$$

Define 'good event' $\mathcal{E}_t$ as

$$\mathcal{E}_t := \left\{ ||\hat{x} - \hat{e}_t||_0 \leq k/8^t \right\}$$

Conditional on $\mathcal{E}_{t-1}$, for every $f \in [n]$ the probability of failure to recover is at most the sum of

- probability of collision with another element, which is no more than

$$\frac{k/8^t}{B_t} = \frac{k/8^t}{C \cdot k/4^t} \leq \frac{1}{C \cdot 2^t}$$

Full algorithm – analysis

Let

$$\widehat{e}_t \leftarrow \text{contents of the list after stage } t.$$

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \leq k/8^t \right\}$$

Conditional on $\mathscr{E}_{t-1}$, for every $f \in [n]$ the probability of failure to recover is at most the sum of

- probability of collision with another element, which is no more than
$$\frac{k/8^t}{B_t} = \frac{k/8^t}{C \cdot k/4^t} \leq \frac{1}{C \cdot 2^t}$$
- probability of being hashed to the non-flat region, which is no more than
$$O(\gamma_t) = O\left(\frac{1}{C2^t}\right)$$

Full algorithm – analysis

Define 'good event' $\mathscr{E}_t$ as

$$\mathscr{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \leq k/8^t \right\}$$

Then

$$\mathbf{Pr}[\mathscr{E}_t | \mathscr{E}_{t-1}] \leq \mathbf{Pr}[\text{fraction of failures is} \geq 1/8 | \mathscr{E}_{t-1}] \leq O\left(\frac{1}{C \cdot 2^t}\right)$$

Full algorithm – analysis

Define 'good event' $\mathcal{E}_t$ as

$$\mathcal{E}_t := \left\{ ||\widehat{x} - \widehat{e}_t||_0 \le k/8^t \right\}$$

Then

$$\mathbf{Pr}[\mathcal{E}_t | \mathcal{E}_{t-1}] \le \mathbf{Pr}[\text{fraction of failures is} \ge 1/8 | \mathcal{E}_{t-1}] \le O\left(\frac{1}{C \cdot 2^t}\right)$$

So for a sufficiently large $C > 0$

$$\mathbf{Pr}[\overline{\mathcal{E}}_1 \vee \ldots \vee \overline{\mathcal{E}}_{\log k}] \le O(1/C) \cdot (1/2 + 1/4 + \ldots) = O(1/C) < 1/10$$

Full algorithm – analysis

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

Time complexity

- DFT:
  $O(k \log n) + O((k/4) \log n) + \ldots = O(k \log n)$
- List update: $k \cdot \log n$

Sample complexity

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

Sample complexity $O(k \log n) + O((k/4) \log n) + \ldots = O(k \log n)$

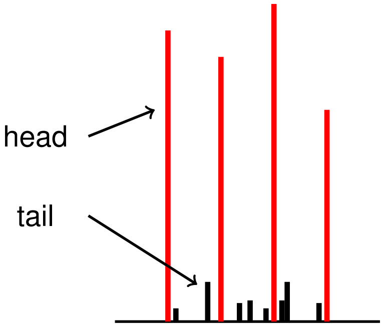
Suboptimal: sufficient to measure $x_0, x_1, \ldots, x_{2k}$ to reconstruct $\widehat{x}$ if $\mathrm{supp}(\widehat{x}) \leq k$ (exercise).

Next:

- recovery in the noisy setting

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$/\sqrt{k}$

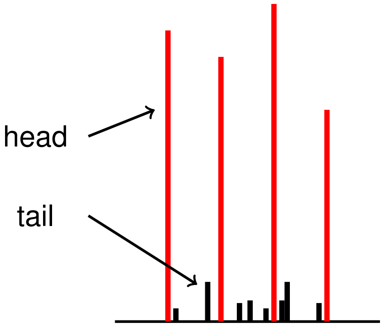
ℓ_2/ℓ_2 sparse recovery guarantees:

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

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

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

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



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

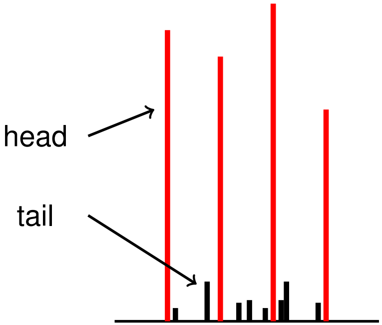
ℓ_2/ℓ_2 sparse recovery guarantees:

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

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

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

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



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

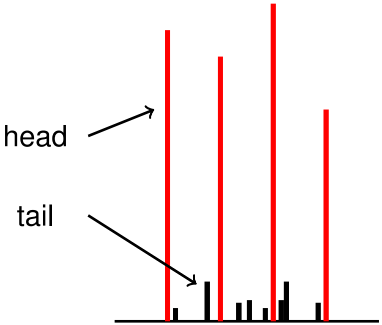
ℓ_2/ℓ_2 sparse recovery guarantees:

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

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

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

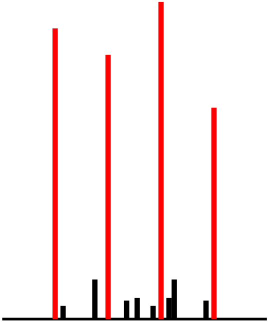
Residual error bounded by noise energy $\mathrm{Err}_k^2(\widehat{x})$



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

ℓ_2/ℓ_2 sparse recovery guarantees:

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



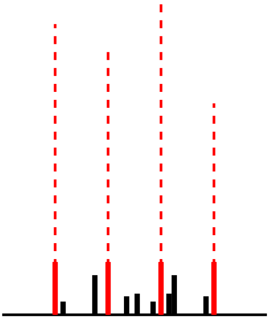
$\mu \approx$ tail noise$/\sqrt{k}$

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

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

ℓ_2/ℓ_2 sparse recovery guarantees:

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



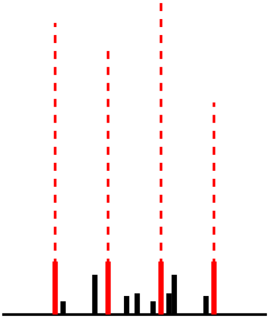
$\mu \approx \text{tail noise}/\sqrt{k}$

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

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

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$/\sqrt{k}$

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

$$|\widehat{x}_i - \widehat{y}_i| \leq c\mu$$

Next:

1. Full algorithm for noisy setting

Next:

1. **Full algorithm for noisy setting**

Basic block (noiseless setting)

Assume

- n is a power of 2
- $\widehat{x}$ contains at most k coefficients with polynomial precision (e.g. $\widehat{x}_f$ in $\{-n^{O(1)}, \ldots, n^{O(1)}\}$)

Then there exists an $O(k \log n)$ time algorithm that

- outputs at most k potential coefficients
- outputs each nonzero $\widehat{x}_f$ correctly with probability at least $1 - \beta$ for a constant $\beta > 0$

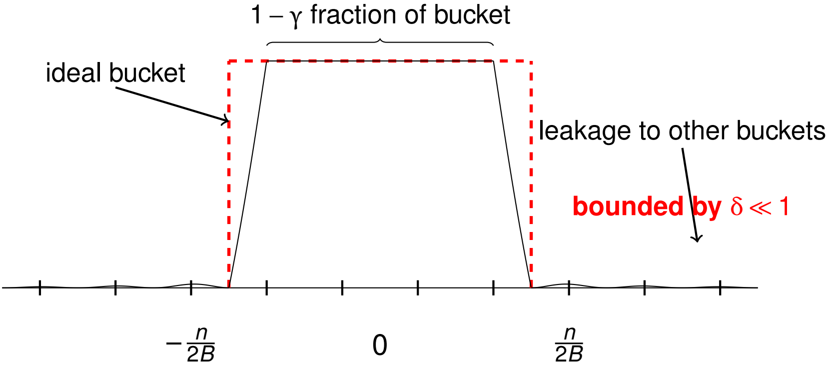
Basic block (noisy setting)

Assume

- n is a power of 2
- $\widehat{x}$ contains at most k coefficients with polynomial precision (e.g. $\widehat{x}_f$ in $\{-n^{O(1)}, \ldots, n^{O(1)}\}$), **plus noise**

Then there exists an $O(k \log n)$ time algorithm that

- outputs at most k potential coefficients
- outputs each nonzero $\widehat{x}_f$ **that is above noise level** correctly with probability at least $1 - \beta$ for a constant $\beta > 0$



Let G be a $(B, \delta/n, \gamma)$-flat window function:

- B buckets
- flat region of width $1-\gamma$
- leakage $\leq \delta/n = 1/n^{O(1)}$

Such G can be constructed with

$$\text{supp}(G) = O((k/\gamma)\log n)$$

PartialRecovery – algorithm

Main idea: filter, then run 1-sparse algorithm on each subproblem

PartialRecovery(x, B, γ, δ)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j}\omega^{jb}$
$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

Run 1-sparse decoding one $\widehat{c}', \widehat{c}''$

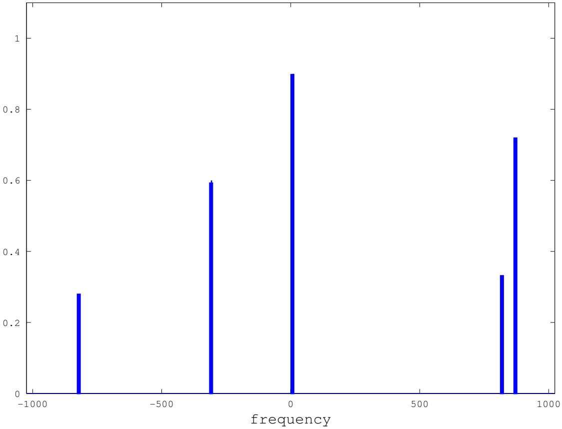
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

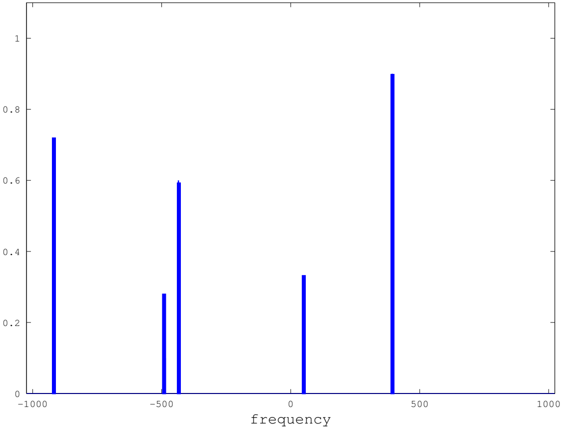
PartialRecovery – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

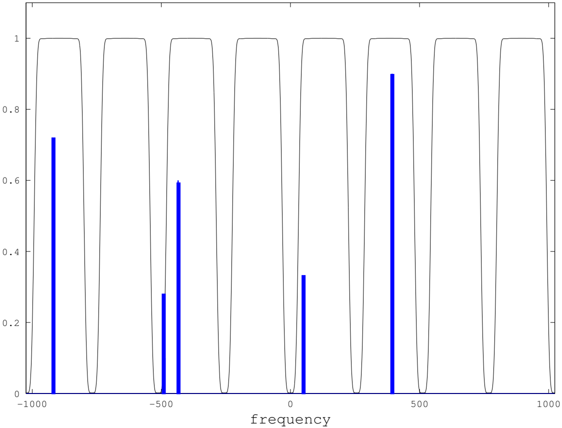
PARTIALRECOVERY – algorithm

Recovering 5-sparse signal $\widehat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

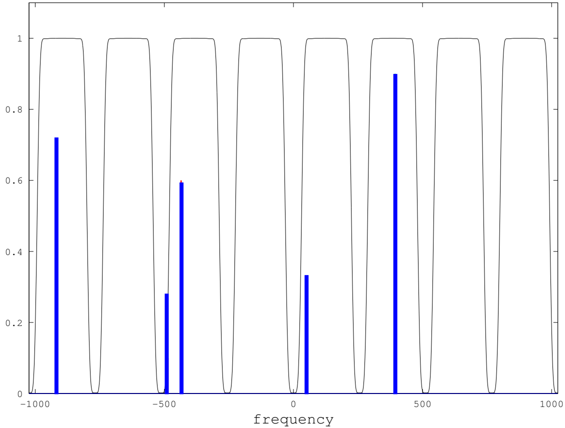
PartialRecovery – algorithm

Recovering 5-sparse signal $\hat{x}$ from measurements of x

Permute spectrum

Filter signal

1-sparse decoding



Isolated frequencies are decoded successfully

PartialRecovery (noiseless setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$
$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$

$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$
If $|\widehat{c}'_{j \cdot n/B}| > 1/2$
Decode from $\widehat{c}'_{j \cdot n/B}, \widehat{c}''_{j \cdot n/B}$
(Two-point sampling)
End
End

PartialRecovery (noisy setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

Define $x'_j \leftarrow x_{\sigma j} \omega^{jb}$
$x''_j \leftarrow x'_{j+1}$

Compute $\widehat{c}'_{j \cdot \frac{n}{B}}, j \in [B]$, where $c' = x' \cdot G$
$\widehat{c}''_{j \cdot \frac{n}{B}}, j \in [B]$, where $c'' = x'' \cdot G$

For $j \in [B]$
If $|\widehat{c}'_{j \cdot n/B}| > 1/2$
Decode from $\widehat{c}'_{j \cdot n/B}, \widehat{c}''_{j \cdot n/B}$
(Two-point sampling)
End
End

PartialRecovery (noisy setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

For $s = 0, \ldots, \log_2 n$

$r = 1, \ldots, O(\log\log n)$

Define $x_j^{\mathbf{s},\mathbf{0},\mathbf{r}} \leftarrow x_{\sigma(j+\mathbf{r})}\omega^{(j+\mathbf{r})b}$

$x_j^{\mathbf{s},\mathbf{1},\mathbf{r}} \leftarrow x_{j+\mathbf{n}/\mathbf{2}^{\mathbf{s}+\mathbf{1}}}^{s,1,r}$

Compute $(\widehat{x^{s,0,r} \cdot G})_{j \cdot n/B}$, for $j \in [B]$

$(\widehat{x^{s,1,r} \cdot G})_{j \cdot n/B}$, for $j \in [B]$

Initialize list $L \leftarrow \emptyset$

For $j \in [B]$

Decode from $\widehat{x}_{j \cdot n/B}^{s,*,r}$, add to list L (output B elements)

(As in lecture 1)

End

Estimate values of $i \in L$, output top $3k$

Noise-tolerant decoding from lecture 1

Suppose that x is approximately 1-sparse, i.e.

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\widehat{x}_{f^*}|^2 \qquad \text{(small noise)}$$

for some small constant ε.

Then

1. can find f^* using $O(\log n \cdot \log\log n)$ runtime
   $O(\log n \cdot \log\log n)$ samples
   with $\geq 1 - 1/4$ success probability

Noise-tolerant decoding from lecture 1

Suppose that x is approximately 1-sparse, i.e.

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\widehat{x}_{f^*}|^2 \qquad \text{(small noise)}$$

for some small constant ε.

Then

1. can find f^* using $O(\mathbf{t} \cdot \log n \cdot \log\log n)$ runtime
   $O(\mathbf{t} \cdot \log n \cdot \log\log n)$ samples
   with $\geq 1 - 4^{-\mathbf{t}}$ success probability

Noise-tolerant decoding from lecture 1

Suppose that x is approximately 1-sparse, i.e.

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\widehat{x}_{f^*}|^2 \qquad \text{(small noise)}$$

for some small constant ε.

Then

1. can find f^* using $O(\mathbf{t} \cdot \log n \cdot \log\log n)$ runtime
   $O(\mathbf{t} \cdot \log n \cdot \log\log n)$ samples
   with $\geq 1 - 4^{-\mathbf{t}}$ success probability

Need to ensure that noise is small in most subproblems

Noise-tolerant decoding from lecture 1

Suppose that x is approximately 1-sparse, i.e.

$$\sum_{f \neq f^*} |\widehat{x}_f|^2 \leq \varepsilon |\widehat{x}_{f^*}|^2 \qquad \text{(small noise)}$$

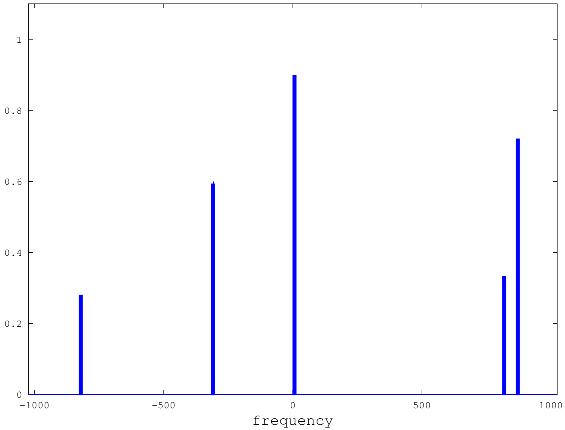
for some small constant ε.

Then

1. can find f^* using $O(\log(\mathbf{1}/\gamma) \cdot \log n \cdot \log\log n)$ runtime
   $O(\log(\mathbf{1}/\gamma) \cdot \log n \cdot \log\log n)$ samples
   with $\geq 1 - \gamma$ success probability

Need to ensure that noise is small in most subproblems

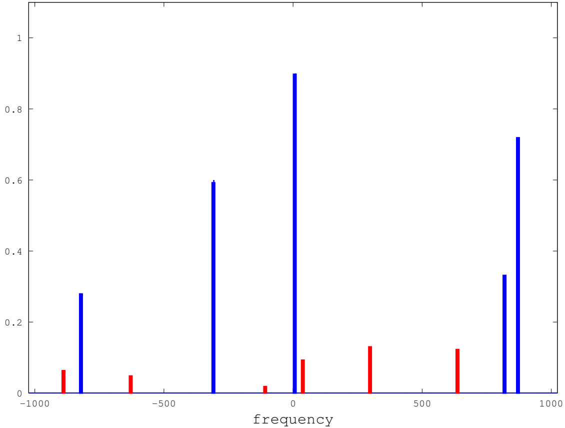
Let $\mu^2 := \frac{1}{k} \min_{k-\text{sparse } y} ||x-y||_2^2 = \frac{1}{k} \sum_{j=k+1}^{n} |x_j|^2$ (average noise level)



For every head element $i \in [k]$ and every tail element $j \in [n] \setminus [k]$

$$\textbf{Pr}[i \text{ and } j \text{ hash to the same bucket}] = O(1/B)$$

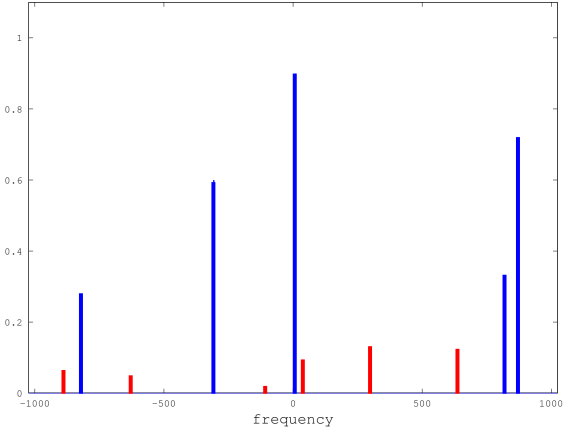
Let $\mu^2 := \frac{1}{k} \min_{k\text{-sparse } y} ||x - y||_2^2 = \frac{1}{k} \sum_{j=k+1}^{n} |x_j|^2$ (average noise level)



For every head element $i \in [k]$ and every tail element $j \in [n] \setminus [k]$

$$\mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)] = O(1/B)$$

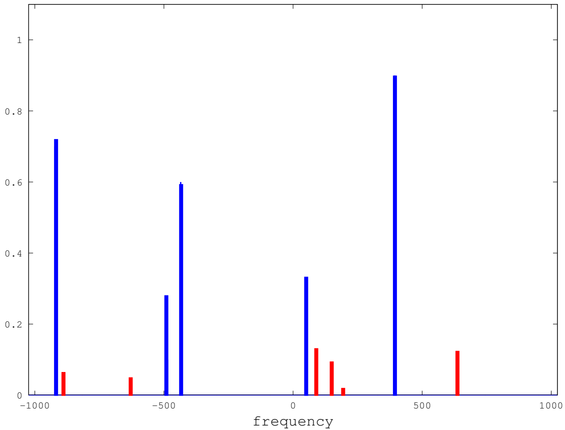
Let $\mu^2 := \frac{1}{k} \min_{k\text{-sparse } y} ||x - y||_2^2 = \frac{1}{k} \sum_{j=k+1}^{n} |x_j|^2$ (average noise level)



For every head element $i \in [k]$, expected noise in i's bucket is

$$\mathbf{E}\left[\sum_{j=k+1}^{n} |\widehat{x}_j|^2 \cdot \mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)]\right] = \mu^2 \cdot O(k/B)$$

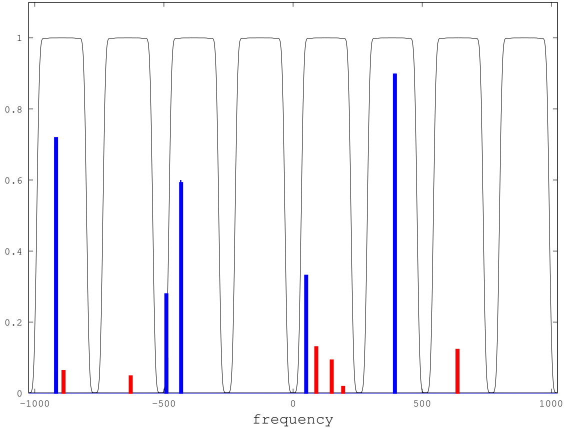
Let $\mu^2 := \frac{1}{k} \min_{k\text{-sparse } y} ||x-y||_2^2 = \frac{1}{k} \sum_{j=k+1}^{n} |x_j|^2$ (average noise level)



For every head element $i \in [k]$, expected noise in i's bucket is

$$\mathbf{E}\left[\sum_{j=k+1}^{n} |\widehat{x}_j|^2 \cdot \mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)]\right] = \mu^2 \cdot O(k/B)$$

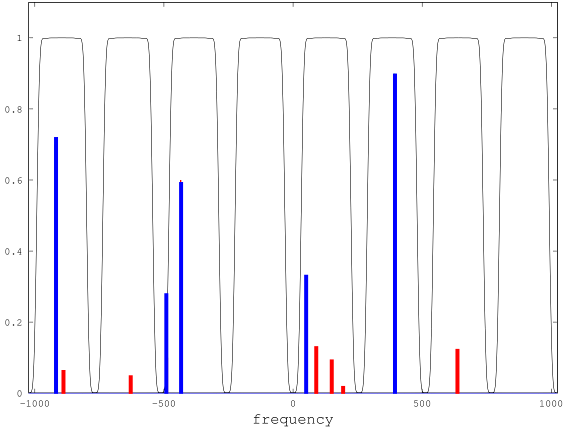
Let $\mu^2 := \frac{1}{k} \min_{k-\text{sparse } y} \|x - y\|_2^2 = \frac{1}{k} \sum_{j=k+1}^{n} |x_j|^2$ (average noise level)



For every head element $i \in [k]$, expected noise in i's bucket is

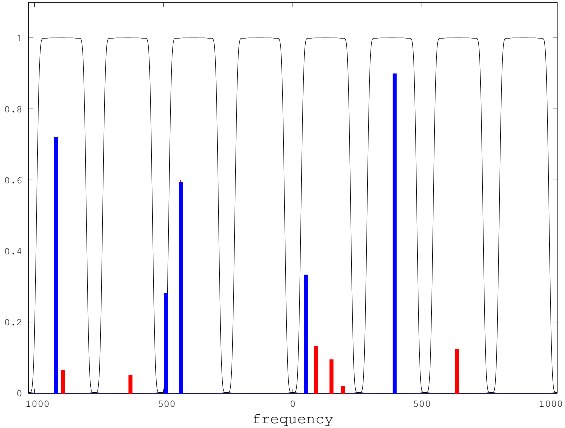
$$\mathbf{E}\left[\sum_{j=k+1}^{n} |\widehat{x}_j|^2 \cdot \mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)]\right] = \mu^2 \cdot O(k/B)$$

Let $\mu^2 := \frac{1}{k} \min_{k\text{-sparse } y} \|x - y\|_2^2 = \frac{1}{k} \sum_{j=k+1}^{n} |x_j|^2$ (average noise level)



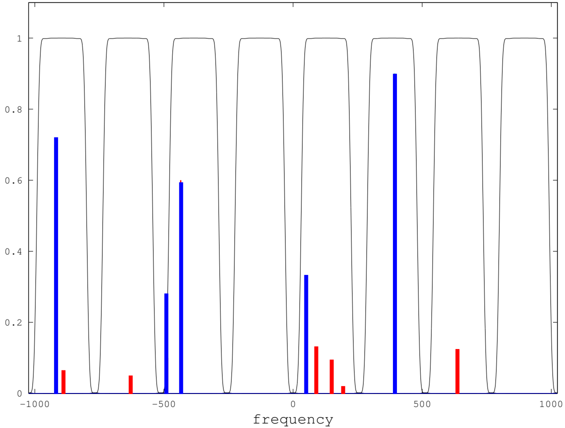
For every head element $i \in [k]$, expected noise in i's bucket is

$$\mathbf{E}\left[\sum_{j=k+1}^{n} |\widehat{x}_j|^2 \cdot \mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)]\right] = \mu^2 \cdot O(k/B)$$



For every head element $i \in [k]$, expected noise in i's bucket is

$$\mathbf{E}\left[\sum_{j=k+1}^{n} |\widehat{x}_j|^2 \cdot \mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)]\right] = \mu^2 \cdot O(k/B)$$



For every head element $i \in [k]$, expected noise in i's bucket is

$$\mathbf{E}\left[\sum_{j=k+1}^{n} |\widehat{x}_j|^2 \cdot \mathbf{Pr}[\mathbf{h}(i) = \mathbf{h}(j)]\right] = \mu^2 \cdot O(k/B)$$

So by Markov's inequality for every head element $i \in [k]$

$$\mathbf{Pr}\left[\sum_{j\in[k+1:n] \text{ s.t. } \mathbf{h}(i)=\mathbf{h}(j)} |\widehat{x}_j|^2\right] > \epsilon\mu^2] = O(k/(\epsilon B))$$

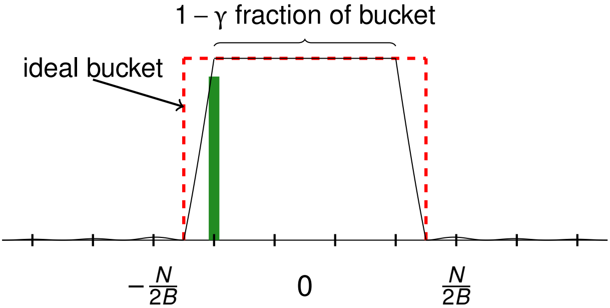
Basic block analysis (noiseless setting)

Claim

For each $u \in supp(\hat{x})$ the probability that u is not reported is bounded by $O(k/B+\gamma)$.

Probability of

- being mapped within n/B of another frequency is $O(k/B)$
- being mapped close to boundary of the bucket is $O(\gamma)$



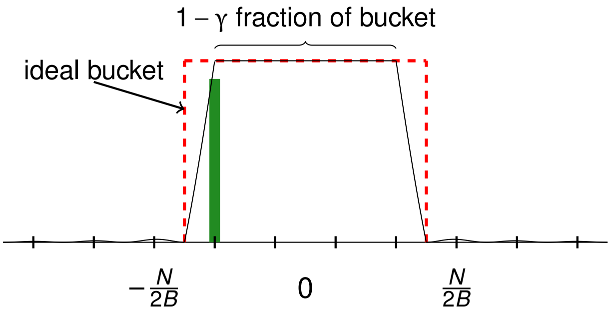
Basic block analysis (**noisy** setting)

Claim

For each $u \in supp(\widehat{x})$ **with** $|\widehat{x}_u|^2 \geq \mu^2$ *the probability that u is not reported is bounded by $O(k/B+\gamma)$.*

Probability of

- being mapped within n/B of another frequency is $O(k/B)$
- being mapped close to boundary of the bucket is $O(\gamma)$
- **colliding with too many tail elements is** $O(k/B)$



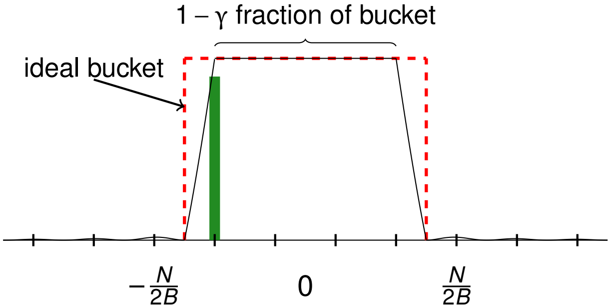
Basic block analysis (**noisy** setting)

Claim

For each $u \in supp(\widehat{x})$ **with** $|\widehat{x}_u|^2 \geq \mu^2$ *the probability that u is not reported is bounded by $O(k/B + \gamma)$.*

Probability of

- being mapped within n/B of another frequency is $O(k/B)$
- being mapped close to boundary of the bucket is $O(\gamma)$
- **colliding with too many tail elements is** $O(k/B)$
- decoding failure is $O(\gamma)$



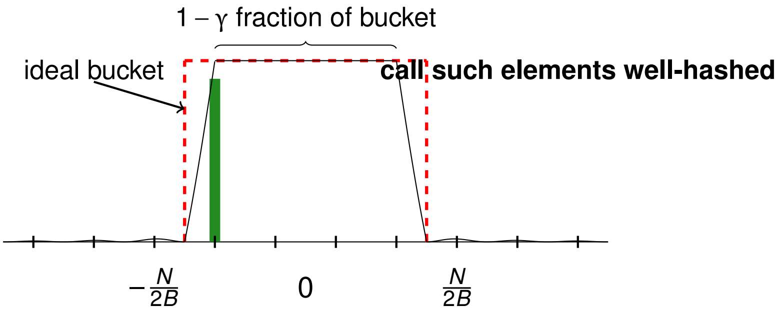
Basic block analysis (**noisy** setting)

Claim

For each $u \in supp(\hat{x})$ **with** $|\hat{x}_u|^2 \geq \mu^2$ *the probability that u is not reported is bounded by* $O(k/B + \gamma)$.

Probability of

- being mapped within n/B of another frequency is $O(k/B)$
- being mapped close to boundary of the bucket is $O(\gamma)$
- **colliding with too many tail elements is** $O(k/B)$
- decoding failure is $O(\gamma)$



PartialRecovery (noisy setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1,2,\ldots,n\}$

For $s = 0,\ldots,\log_2 n$

$r = 1,\ldots,O(\log\log n)$

Define $x_j^{\mathbf{s},\mathbf{0},\mathbf{r}} \leftarrow x_{\sigma(j+\mathbf{r})}\omega^{(j+\mathbf{r})b}$

$x_j^{\mathbf{s},\mathbf{1},\mathbf{r}} \leftarrow x^{s,1,r}_{j+\mathbf{n}/\mathbf{2}^{\mathbf{s}+\mathbf{1}}}$

Compute $(\widehat{x^{s,0,r}\cdot G})_{j\cdot n/B}$, for $j \in [B]$

$(\widehat{x^{s,1,r}\cdot G})_{j\cdot n/B}$, for $j \in [B]$

Initialize list $L \leftarrow \emptyset$

For $j \in [B]$

Decode from $\widehat{x}^{s,*,r}_{j\cdot n/B}$, add to list L (output B elements)

(As in lecture 1)

End

Estimate values of $i \in L$, output top $3k$

PARTIALRECOVERY (noisy setting)

Choose random $b \in [n]$ and odd $\sigma \in \{1, 2, \ldots, n\}$

For $s = 0, \ldots, \log_2 n$

$r = 1, \ldots, O(\log\log n)$

Define $x_j^{\mathbf{s},\mathbf{0},\mathbf{r}} \leftarrow x_{\sigma(j+\mathbf{r})}\omega^{(j+\mathbf{r})b}$

$x_j^{\mathbf{s},\mathbf{1},\mathbf{r}} \leftarrow x^{s,1,r}_{j+\mathbf{n}/\mathbf{2^{s+1}}}$

Compute $(\widehat{x^{s,0,r} \cdot G})_{j \cdot n/B}$, for $j \in [B]$

$(\widehat{x^{s,1,r} \cdot G})_{j \cdot n/B}$, for $j \in [B]$

Initialize list $L \leftarrow \emptyset$

For $j \in [B]$

Decode from $\widehat{x}^{s,*,r}_{j \cdot n/B}$, add to list L (output B elements)

(As in lecture 1)

End

Estimate values of $i \in L$, output top $3k$

Estimating value of a heavy hitter (lecture 1)

Given $f^* \in [n]$,

1. can find w_{f^*} (estimate for $\widehat{x}_{f^*}$) in $O(1)$ time and samples such that

$$|w_{f^*} - \widehat{x}_{f^*}|^2 \leq 3\varepsilon|\widehat{x}_{f^*}|^2$$

with probability $1 - 1/100$

Estimating value of a heavy hitter (lecture 1)

Given $f^* \in [n]$,

1. can find w_{f^*} (estimate for $\widehat{x}_{f^*}$) in $O(t)$ time and samples such that

$$|w_{f^*} - \widehat{x}_{f^*}|^2 \leq 3\varepsilon|\widehat{x}_{f^*}|^2$$

with probability $1 - \mathbf{2^{-t}}$

Estimating value of a heavy hitter (lecture 1)

Given $f^* \in [n]$,

1. can find w_{f^*} (estimate for $\widehat{x}_{f^*}$) in $O(\log n)$ time and samples such that

$$|w_{f^*} - \widehat{x}_{f^*}|^2 \leq 3\varepsilon |\widehat{x}_{f^*}|^2$$

   with probability $1 - \mathbf{1/n^2}$

Estimating value of a heavy hitter (lecture 1)

Given $f^* \in [n]$,

1. can find w_{f^*} (estimate for $\widehat{x}_{f^*}$) in $O(\log n)$ time and samples such that
$$|w_{f^*} - \widehat{x}_{f^*}|^2 \leq 3\varepsilon |\widehat{x}_{f^*}|^2$$
with probability $1 - \mathbf{1/n^2}$

Let L denote the list of located elements

Estimating value of a heavy hitter (lecture 1)

Given $f^* \in [n]$,

1. can find w_{f^*} (estimate for $\widehat{x}_{f^*}$) in $O(\log n)$ time and samples such that

$$|w_{f^*} - \widehat{x}_{f^*}|^2 \leq 3\varepsilon|\widehat{x}_{f^*}|^2$$

with probability $1 - \mathbf{1/n^2}$

Let L denote the list of located elements

Estimating value of a heavy hitter (lecture 1)

Given $f^* \in [n]$,

1. can find w_{f^*} (estimate for $\widehat{x}_{f^*}$) in $O(\log n)$ time and samples such that

$$|w_{f^*} - \widehat{x}_{f^*}|^2 \leq 3\varepsilon|\widehat{x}_{f^*}|^2$$

with probability $1 - \mathbf{1/n^2}$

Let L denote the list of located elements

Using $O(k \log^2 n)$ samples and runtime, can find w_L such that

$$|w_f - \widehat{x}_f|^2 \leq 3\varepsilon|\widehat{x}_f|^2$$

for all $f \in L$.

Let $L' \subseteq L$ denote list of top $3k$ values in L (in terms of magnitude)

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k \quad , \frac{1}{16} \qquad , 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\mathrm{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\mathrm{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n))$

Full algorithm

Let $C > 0$ be a sufficiently large constant.

$\textsc{PartialRecovery}(x, C \cdot k, \frac{1}{16}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/2, \frac{1}{16} \cdot 2^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/4, \frac{1}{16} \cdot 4^{-1}, 1/\text{poly}(n))$

$\textsc{PartialRecovery}(x, C \cdot k/8, \frac{1}{16} \cdot 8^{-1}, 1/\text{poly}(n))$

...

Full algorithm

Permute spectrum

Hash to 8 buckets

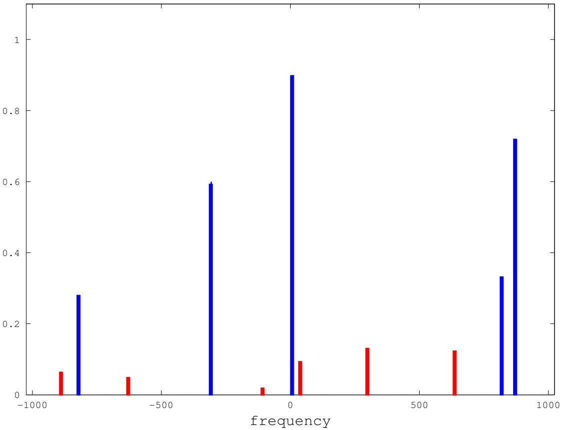
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

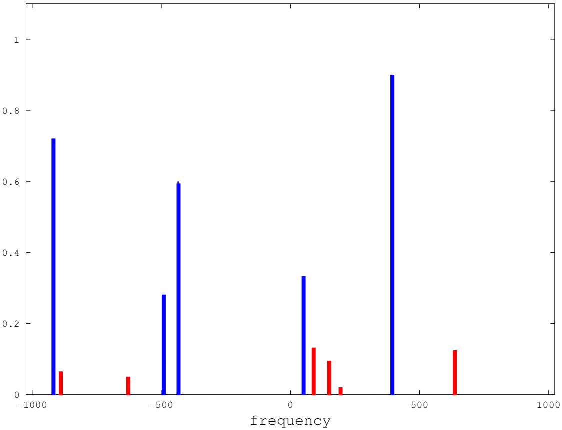
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

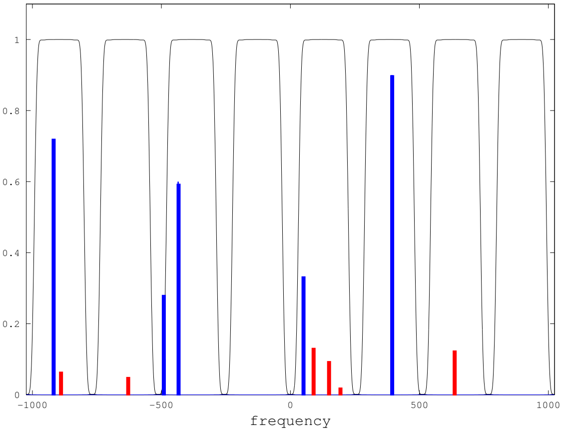
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

…



Full algorithm

Permute spectrum

Hash to 8 buckets

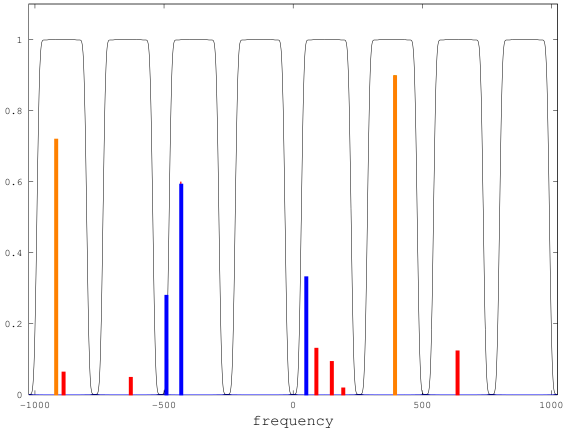
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

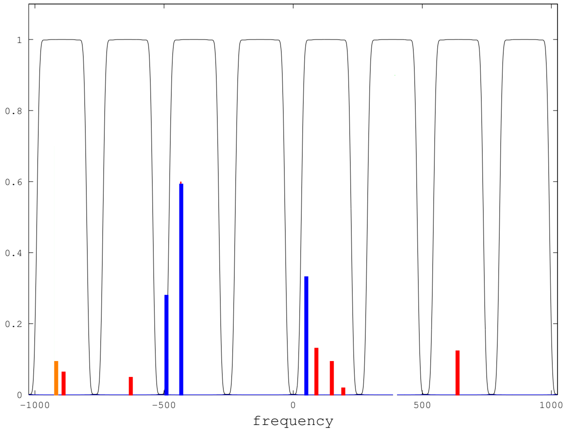
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

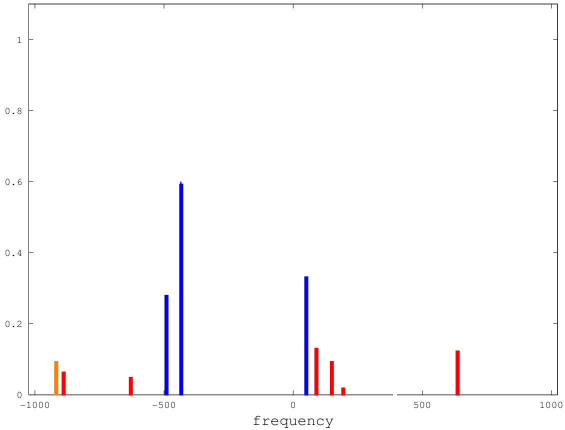
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

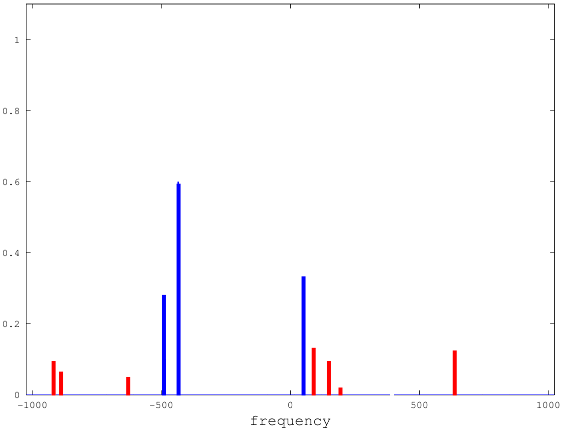
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

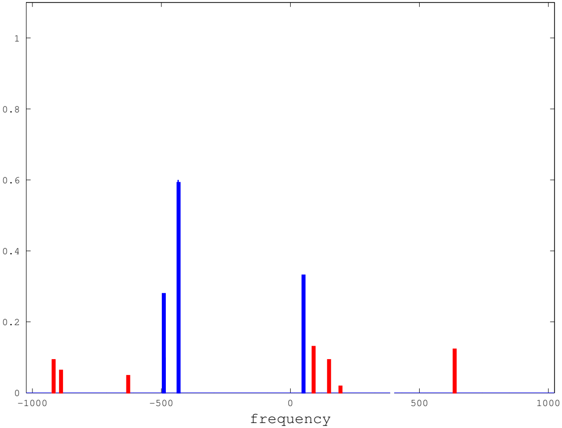
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

…



Full algorithm

Permute spectrum

Hash to 8 buckets

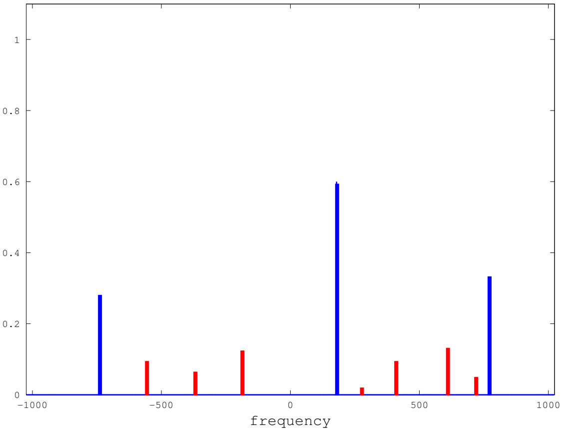
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

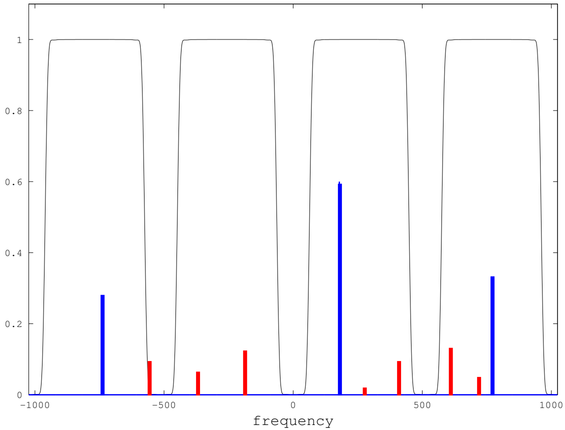
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

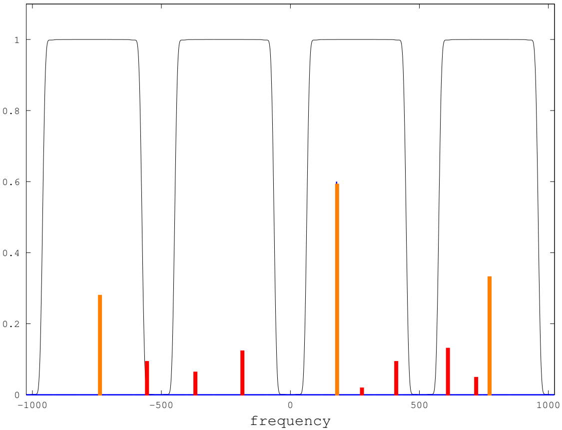
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

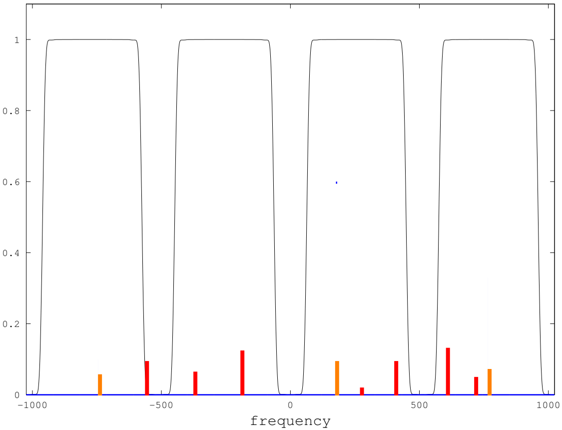
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

Permute spectrum

Hash to 8 buckets

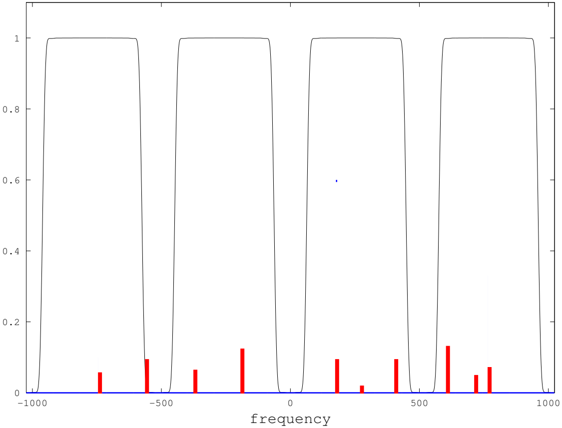
Recover well-hashed coeffs

Permute spectrum

Hash to 4 buckets

Recover well-hashed coeffs

...



Full algorithm

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

Time complexity:

- DFT: $O(k\log^2 n(\log\log n)) + O((k/4)\log^2 n\log\log n) + \ldots = O(k\log^2 n\log\log n)$
- List update: $k \cdot \log n$

Sample complexity

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

Sample complexity:
$O(k \log^2 n(\log\log n)) + O((k/4)\log^2 n(\log\log n)) + \ldots =$
$O(k \log^2 n \log\log n)$

Suboptimal (?): a lower bound of $\Omega(k \log(n/k))$ known

Runtime and sample complexity

Noisy: runtime $O(k \log^2 n)$, sample complexity $O(k \log^2 n \log \log n)$

$O(\log \log n)$ can be removed, see Hassanieh-Indyk-Katabi-Price'STOC12

Sample complexity lower bound: $\Omega(k \log(n/k))$ (Do Ba, Indyk, Price, Woodruff'SODA10)

Next lecture:

$O(k \log n (\log\log n)^{O(1)})$ samples, $O(k \log^2 n (\log\log n)^{O(1)})$ runtime
(Indyk-Kapralov-Price'SODA14)

and

$O(k \log n)$ samples and $O(n \log^3 n)$ runtime
(Indyk-Kapralov'FOCS14)