Sparse Fourier Transform
(lecture 4)

Michael Kapralov

\(^1\)IBM Watson → EPFL

St. Petersburg CS Club
November 2015
Given $x \in \mathbb{C}^n$, compute the Discrete Fourier Transform of $x$:

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

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 $x$:

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

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

**Goal:** find the top $k$ coefficients of $\hat{x}$ 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 $x$:

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

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

**Goal:** find the top $k$ coefficients of $\hat{x}$ 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 ← φ

For \( t = 1 \) to \( \log k \)

\( B_t ← Ck/4^t \)

\( γ_t ← 1/(C2^t) \)

\[ List ← List + \text{PARTIALRECOVERY}(B_t, γ_t, List) \]

End

Summary:

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

List ← φ

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

$B_t ← Ck/4^t$

$\gamma_t ← 1/(C2^t)$

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

End

Summary:

- independent invocations of $\text{PARTIALRECOVERY}$: use fresh samples in every iteration
Improvements?

List ← \emptyset 

\textbf{For} \ t = 1 \ \textbf{to} \ \log k \\
\quad B_t ← Ck / 4^t \\
\quad γ_t ← 1/(C2^t) \\
\quad List ← List + \textsc{PartialRecovery}(B_t, γ_t, List) \\
\textbf{End}

Summary:

\begin{itemize}
  \item independent invocations of \textsc{PartialRecovery}: use fresh samples in every iteration
  \item reduce (approximate) sparsity at geometric rate
\end{itemize}
Improvements?

\[ \text{List} \leftarrow \emptyset \]
\[ \text{For } t = 1 \text{ to } \log k \]
\[ B_t \leftarrow Ck/4^t \]
\[ \gamma_t \leftarrow 1/(C2^t) \]
\[ \text{List} \leftarrow \text{List} + \text{PARTIALRECOVERY}(B_t, \gamma_t, \text{List}) \]
\[ \text{End} \]

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

\[
\begin{align*}
\text{List} & \leftarrow \emptyset \\
\text{For } t = 1 \text{ to } \log k & \\
B_t & \leftarrow Ck/4^t \\
\gamma_t & \leftarrow 1/(C2^t) \\
\text{List} & \leftarrow \text{List} + \text{PARTIALRECOVERY}(B_t, \gamma_t, \text{List})
\end{align*}
\]

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

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

\(\text{supp}(G) = B \approx k\) as opposed to \(B \approx k \log n\), but **buckets leak**.
Why not use simpler filters with smaller support?

Let

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

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 
\( \approx \| \hat{x} \|_2^2 / k \), and estimate up to \( \approx \| \hat{x} \|_2^2 / k \) 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’
ISMRRM’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 $\hat{y}$ such that

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

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

### 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

### Deterministic, $\Omega(n)$ runtime

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

### 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
Uniform bounds (for all):

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

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

Deterministic, $\Omega(n)$ runtime

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

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 $\ell_2/\ell_2$ 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 $k \leq n^{1-\delta}$. 
\( \ell_2/\ell_2 \) sparse recovery guarantees:

\[
\|\hat{x} - \hat{y}\|^2 \leq C \cdot \min_{k\text{-sparse } \hat{z}} \|\hat{x} - \hat{z}\|^2
\]
\( \ell_2/\ell_2 \) sparse recovery guarantees:

\[
||\hat{x} - \hat{y}||^2 \leq C \cdot \text{Err}_k^2(\hat{x})
\]

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

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

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

\[\text{Residual error bounded by noise energy } \text{Err}_k^2(\hat{x})\]
\ell_2/\ell_2 \text{ sparse recovery guarantees:}

\[
\text{Signal to noise ratio } R = \frac{||\hat{x} - \hat{y}||^2}{\text{Err}^2_k(\hat{x})} \leq C
\]

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

\[
\text{Residual error bounded by noise energy } \text{Err}^2_k(\hat{x})
\]

\[
\text{Err}^2_k(\hat{x}) = \sum_{j=k+1}^{n} |\hat{x}_j|^2
\]
\( \ell_2/\ell_2 \) sparse recovery guarantees:

Signal to noise ratio \( R = \|\hat{x} - \hat{y}\|^2/\text{Err}_{k}^2(\hat{x}) \leq C \)

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

Residual error bounded by noise energy \( \text{Err}_{k}^2(\hat{x}) \)

\[
\text{Err}_{k}^2(\hat{x}) = \sum_{j=k+1}^{n} |\hat{x}_j|^2
\]

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

\[
|\hat{x}_i - \hat{y}_i|^2 \leq c \cdot \text{Err}_{k}^2(\hat{x})/k =: \mu^2
\]
Iterative recovery

Many algorithms use the iterative recovery scheme:

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

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

$\hat{z} \leftarrow \text{PARTIALRECOVERY}(x, \hat{y}_{t-1}) \quad \triangleright \text{Takes random samples of } x - y$

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

**PARTIALRECOVERY**($x, \hat{y}$)

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

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

\textbf{return} dominant Fourier coefficients \hat{Z} of \( x - y \) (approximately)

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

Main questions:

- How many samples per SNR reduction step?
- How many iterations?
**PARTIAL\textsc{RECOVERY}(x, \hat{y})**

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

\begin{align*}
\text{dominant coefficients} & \approx |\hat{x}_i - \hat{y}_i|^2 \geq \mu^2 (\text{above average noise level})
\end{align*}

\textbf{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

\[ x_a = \omega^{a \cdot f} + \text{noise} \]

\[ O(\log_{SNR} n) \text{ measurements} \]

for random \( a \)
Reducing $k$-sparse recovery to 1-sparse recovery

Permute with a random linear transformation and phase shift

Choose a filter $G$, $\hat{G}$ such that $\hat{G}$ approximates the buckets $G$ has small support

Sample complexity $= \supp G$
Reducing $k$-sparse recovery to 1-sparse recovery

Permute with a random linear transformation and phase shift

Choose a filter $G$, $\hat{G}$ such that $\hat{G}$ approximates the buckets $G$ has small support

Sample complexity $= \supp G$!
Reducing $k$-sparse recovery to 1-sparse recovery

Permute with a random linear transformation and phase shift

Choose a filter $G$, $\hat{G}$ such that $\hat{G}$ approximates the buckets $G$ has small support.
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 $G, \hat{G}$ such that

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

Compute $\hat{x} \ast \hat{G} = \overline{\langle x \cdot G \rangle}$
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 $G, \hat{G}$ such that

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

Compute $\hat{x} \ast \hat{G} = (\overline{x \cdot G})$
Reducing $k$-sparse recovery to 1-sparse recovery

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

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

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

Compute $\hat{x} \ast \hat{G} = (x \cdot \hat{G})$
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 $G, \hat{G}$ such that

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

Compute $\hat{x} \ast \hat{G} = (x \cdot \hat{G})$

Sample complexity $= \text{supp } G$!
**Partial Recovery step**

**Partial Recovery**($x, \hat{y}$)

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

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

Sample complexity = support of $G$
**PARTIAL RECOVERY** step

**PARTIAL RECOVERY**($x, \hat{y}$)

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

**return** dominant Fourier coefficients $\hat{Z}$ 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

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 $O(k \log R) \approx \text{convolve boxcar with itself log } R \text{ times}$
When SNR is bounded by $R$:

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

- (most) $1$-sparse recovery subproblems for dominant frequencies have high SNR (about $R$) so $\mathcal{O}^*(\log_R n)$ measurements!

$$\mathcal{O}^*(k \log R \cdot \log_R n) = \mathcal{O}^*(k \log n)$$ samples per step!
\[
\begin{align*}
R &\rightarrow R^{1/2} \rightarrow R^{1/4} \rightarrow \ldots \rightarrow C^2 \rightarrow C \\
&\underbrace{\text{O(loglog} \ n) \text{ iterations}}
\end{align*}
\]

**PARTIAL \text{RECOVERY}(x, \hat{y}, R)**

\[
\begin{align*}
R \cdot \mu^2 \\
R^{1/2} \cdot \mu^2 \\
\mu^2 = \text{tail noise}/B
\end{align*}
\]
\[
R \rightarrow R^{1/2} \rightarrow R^{1/4} \rightarrow \cdots \rightarrow C^2 \rightarrow C
\]

\[O(\log \log n) \text{ iterations}\]

\textsc{PartialRecovery}(x, \hat{y}, R)

\[R \cdot \mu^2\]

\[R^{1/2} \cdot \mu^2\]

\[\mu^2 = \text{tail noise}/B\]
\[ R \to R^{1/2} \to R^{1/4} \to \ldots \to C^2 \to C \]

\[ O(\log \log n) \text{ iterations} \]

**PARTIAL\text{RECOVERY}(x, \hat{y}, R^{1/2})**

\[ \mu^2 = \text{tail noise} / B \]
$R \rightarrow R^{1/2} \rightarrow R^{1/4} \rightarrow \ldots \rightarrow C^2 \rightarrow C$

$O(\log \log n)$ iterations

**Partial Recovery** $(x, \hat{y}, C^2)$

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

Input: $x \in \mathbb{C}^n$
\begin{align*}
\hat{y}_0 &\leftarrow 0 \\
R_0 &\leftarrow \text{poly}(n)
\end{align*}

For $t = 1$ to $O(\log \log n)$
\begin{align*}
\hat{z} &\leftarrow \text{PARTIAL\_RECOVERY}(x, \hat{y}_{t-1}, R_{t-1}) \quad \triangleright \text{Takes samples of } x - y \\
\text{Update } \hat{y}_t &\leftarrow \hat{y}_{t-1} + \hat{z} \\
R_t &\leftarrow \sqrt{R_{t-1}}
\end{align*}

\text{PARTIAL\_RECOVERY step:}
\begin{itemize}
  \item Takes $O^*(k \log n)$ samples independent of $R$
  \item Is very effective: reduces $R \rightarrow R^\frac{1}{2}$, so $O(\log \log n)$ iterations suffice
\end{itemize}
Need to reduce most ‘large’ frequencies, i.e. $|\hat{x}_i|^2 \geq \sqrt{R}\mu^2$
Partial recovery analysis

\textsc{PartialRecovery}(x, \hat{y}, R)

\[ R \cdot \mu^2 \]

\[ R^{1/2} \cdot \mu^2 \]

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

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

\textsc{PartialRecovery}(x, \hat{y}, R)

\[ R \cdot \mu^2 \]

\[ R^{\frac{1}{2}} \cdot \mu^2 \]

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

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

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

Deterministic, $\Omega(n)$ runtime
$O(k \log^2 k \log n)$

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
Uniform bounds (for all):
Candes-Tao’06
Rudelson-Vershynin’08
Cheraghchi-Guruswami-Velingker’12
Bourgain’14
Haviv-Regev’15

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

Deterministic, $\Omega(n)$ runtime
$O(k \log^2 k \log n)$

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 $\ell_2/\ell_2$ sparse recovery from Fourier measurements using $O(k \log n)$ samples and $O(n \log^3 n)$ runtime.

Optimal up to constant factors for $k \leq n^{1-\delta}$. 
Higher dimensional Fourier transform is needed in some applications.

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

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

where $\omega$ 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 $\ell_2/\ell_2$ 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 $d$. 
\( \ell_2 / \ell_2 \) sparse recovery guarantees:

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

\( \mu \approx \text{tail noise} / \sqrt{k} \)
\( \ell_2/\ell_2 \) sparse recovery guarantees:

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

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

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

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

\( \mu \approx \text{tail noise} / \sqrt{k} \)
$\ell_2/\ell_2$ sparse recovery guarantees:

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

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

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

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

$\mu \approx \text{tail noise}/\sqrt{k}$
\( \ell_2/\ell_2 \) sparse recovery guarantees:

\[
||\hat{x} - \hat{y}||^2 \leq C \cdot \text{Err}_k^2(\hat{x})
\]

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

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

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

\[\mu \approx \text{tail noise}/\sqrt{k}\]
\[ \ell_2/\ell_2 \text{ sparse recovery guarantees:} \]

\[ \| \hat{x} - \hat{y} \|^2 \leq C \cdot \text{Err}^2_k(\hat{x}) \]

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

\[ |\hat{x}_i - \hat{y}_i|^2 \leq c \cdot \text{Err}^2_k(\hat{x})/k =: \mu^2 \]
\( \ell_2/\ell_2 \) sparse recovery guarantees:

\[
\| \hat{x} - \hat{y} \|_2^2 \leq C \cdot \text{Err}_k^2(\hat{x})
\]

\( \hat{\mu} \approx \text{tail noise}/\sqrt{k} \)

Will ensure that all elements are below average noise level:

\[
\| \hat{x} - \hat{y} \|_\infty^2 \leq c \cdot \text{Err}_k^2(\hat{x})/k =: \mu^2
\]
$\ell_\infty/\ell_2$ sparse recovery guarantees:

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

Will ensure that all elements are below average noise level:

$$\|\hat{x} - \hat{y}\|_\infty^2 \leq c \cdot \text{Err}^2_k(\hat{x})/k =: \mu^2$$
\( \ell_\infty / \ell_2 \) sparse recovery guarantees:

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

\( \mu \approx \text{tail noise} / \sqrt{k} \)

Will ensure that all elements are below average noise level:

\[
\| \hat{x} - \hat{y} \|_\infty^2 \leq \mu^2
\]
Iterative recovery

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

\[ \hat{y}_0 \leftarrow 0 \]

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{PARTIAL\_RECOVERY}(x - y_{t-1})$  \(\triangleright \) Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

In most prior works, sampling complexity is $\text{samples per PARTIAL\_RECOVERY step} \times \text{number of iterations}$.

Lots of work on carefully choosing filters, reducing the 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 $\Omega((\log n \cdot d - 1 \log \log n))$ in higher dimensions.
Iterative recovery

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

$\hat{y}_0 \leftarrow 0$

For $t = 1$ to $L$

- $\hat{z} \leftarrow \text{PARTIALRECOVERY}(x - y_{t-1})$ ▶ Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

In most prior works sampling complexity is

samples per PARTIALRECOVERY step $\times$ number of iterations
Iterative recovery

**Input:** \( x \in \mathbb{C}^n \)

\( \hat{y}_0 \leftarrow 0 \)

**For** \( t = 1 \) to \( L \)

- \( \hat{z} \leftarrow \text{PARTIALRECOVERY}(x - y_{t-1}) \) \( \triangleright \) Takes random samples of \( x - y \)
- Update \( \hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z} \)

In most prior works sampling complexity is

\[
\text{samples per PARTIALRECOVERY step} \times \text{number of iterations}
\]

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 recovery

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

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

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

Our sampling complexity is

samples per PARTIALRECOVERY step $\times$ number of iterations
Iterative recovery

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

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

- $\hat{z} \leftarrow \text{PARTIAL\_RECOVERY}(x - y_{t-1})$ △ Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{z}$

Our sampling complexity is

samples per PARTIAL\_RECOVERY step $\times$ number of iterations
Iterative recovery

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

$\hat{y}_0 \leftarrow 0$

**For** $t = 1$ to $L$

- $\hat{Z} \leftarrow \text{PARTIALRECOVERY}(x - y_{t-1})$ ▶ Takes random samples of $x - y$
- Update $\hat{y}_t \leftarrow \hat{y}_{t-1} + \hat{Z}$

Our sampling complexity is

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

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
\[ G \leftarrow B \ast B \ast B \]

Let \( y^m \leftarrow (P_m x) \cdot G \)
\[ m = 0, \ldots, M = C \log n \]

\( \hat{z}_0 \leftarrow 0 \)

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

**For** \( f \in [n] \):

\( \hat{w}_f \leftarrow \text{median}\{\tilde{y}_f^1, \ldots, \tilde{y}_f^M\} \)

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

\( \hat{w}_f \leftarrow 0 \)

**End**

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

\( y^m \leftarrow y^m - (P_m w) \cdot G \)
\[ \text{for } m = 1, \ldots, M \]

**End**

▷ Take samples of \( x \)

▷ Loop over thresholds

▷ Estimate, prune small elements

▷ Update samples
Let $y^m \leftarrow (P_m x) \cdot G$
$m = 0, \ldots, M = C \log n$

$\hat{z}_0 \leftarrow 0$

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

For $f \in [n]$: 

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

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

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

$\hat{z}_0 \leftarrow 0$

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

For $f \in [n]$:

$\hat{w}_f \leftarrow \text{median}\{\tilde{y}_1^f, \ldots, \tilde{y}_M^f\}$

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

$\hat{w}_f \leftarrow 0$

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

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

for $m = 1, \ldots, M$

End

Main challenge: lack of fresh randomness. Why does median work?
\[ G \leftarrow B \ast B \ast B \]

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

\[ m = 0, \ldots, M = C \log n \]

\[ \hat{z}_0 \leftarrow 0 \]

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

For \( f \in [n] \):

\[ \hat{w}_f \leftarrow \text{median}\{\tilde{y}^1_f, \ldots, \tilde{y}^M_f\} \]

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
$G \leftarrow B \ast B \ast B$

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

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

$
\hat{z}_0 \leftarrow 0$

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

For $f \in [n]$

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

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
\[ G \leftarrow B \ast B \ast B \]

Let \( y^m \leftarrow (P_m x) \cdot G \)
\[ m = 0, \ldots, M = C \log n \]

\[ \hat{z}_0 \leftarrow 0 \]

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

\[ \text{For } f \in [n]: \]
\[ \hat{w}_f \leftarrow \text{median} \{ \tilde{y}_f^1, \ldots, \tilde{y}_f^M \} \]
\[ \text{If } |\hat{w}_f| < 2^{T-t} \mu / 3 \text{ then} \]
\[ \hat{w}_f \leftarrow 0 \]

End

\[ \hat{z}_{t+1} = \hat{z}_t + \hat{w} \]
\[ y^m \leftarrow y^m - (P_m w) \cdot G \]
\[ \text{for } m = 1, \ldots, M \]

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 $\ell_1$-minimization (SPGL1)

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

Within a factor of 2 of $\ell_1$ 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