Sublinear Time, Measurement-Optimal, Sparse Recovery For All

Ely Porat and Martin J. Strauss

BIU/UM
Outline

1. Preliminaries
   - Problem we are addressing

2. Algorithm and Result

3. Analysis

4. Conclusion
Sparse recovery

- $\hat{x} = R(\Phi x + \nu) \approx x$
- Approximate best $k$-term signal; length is $N$

![Diagram of sparse recovery process]
Some criteria of algorithms

Speed
Some criteria of algorithms

Speed

Accuracy
Some criteria of algorithms

Number of measurements

Speed

Accuracy
Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \left( \binom{N}{k} \right)$
Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \left( \frac{N}{k} \right)$
- Recovery runtime (speed):
Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \left( \frac{N}{k} \right)$
- Recovery runtime (speed):
  - Want $\text{poly}(k \log N)$.
Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \binom{N}{k}$
- Recovery runtime (speed):
  - Want $\text{poly}(k \log N)$.
  - Faster than previous *measurement-optimal* algorithms.

Other interesting points on this spectrum—[HIKP'12]

Accuracy—how much error, which norm, universality...our model: Recover all signals in (relatively smaller) $\ell_1$ ball, by one matrix.

Norm of error

- Want $\ell_2$: $\|x - \hat{x}\|_2 \leq \epsilon \sqrt{k} \|x - x_k\|_1$.

Here get only $\ell_1$ (strictly worse): $\|x - \hat{x}\|_1 \leq (1 + \epsilon) \|x - x_k\|_1$. 
Some criteria of algorithms

- **Number of measurements**: want $O(k \log N/k) \approx \log \binom{N}{k}$

- **Recovery runtime (speed)**:
  - Want $\text{poly}(k \log N)$.
  - Faster than previous *measurement-optimal* algorithms.
  - (“Sublinear time” measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \binom{N}{k}$
- Recovery runtime (speed):
  - Want $\text{poly}(k \log N)$.
  - Faster than previous *measurement-optimal* algorithms.
  - (“Sublinear time” measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
  - (Other interesting points on this spectrum—[HIKP’12])
Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \left( \frac{N}{k} \right)$

- Recovery runtime (speed):
  - Want $\text{poly}(k \log N)$.
  - Faster than previous measurement-optimal algorithms.
  - (“Sublinear time” measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
  - (Other interesting points on this spectrum—[HIKP’12])

- Accuracy—how much error, which norm, universality...our model:
Some criteria of algorithms

- **Number of measurements**: want $O(k \log N/k) \approx \log \binom{N}{k}$
- **Recovery runtime (speed)**:
  - Want $\text{poly}(k \log N)$.
  - Faster than previous *measurement-optimal* algorithms.
  - (“Sublinear time” measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
  - (Other interesting points on this spectrum—[HIKP’12])
- **Accuracy**—how much error, which norm, universality...our model:
  - Recover *all* signals in (relatively smaller) $\ell_1$ ball, by *one* matrix.
Some criteria of algorithms

- **Number of measurements:** want $O(k \log N/k) \approx \log \left( \frac{N}{k} \right)$

- **Recovery runtime (speed):**
  - Want $\text{poly}(k \log N)$.
  - Faster than previous *measurement-optimal* algorithms.
  - (“Sublinear time” measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
  - (Other interesting points on this spectrum—[HIKP’12])

- **Accuracy**—how much error, which norm, universality...our model:
  - Recover *all* signals in (relatively smaller) $\ell_1$ ball, by *one* matrix.

- **Norm of error**
Preliminaries

Some criteria of algorithms

- Number of measurements: want $O(k \log N/k) \approx \log \binom{N}{k}$

- Recovery runtime (speed):
  - Want poly($k \log N$).
  - Faster than previous measurement-optimal algorithms.
  - ("Sublinear time" measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
  - (Other interesting points on this spectrum—[HIKP’12])

- Accuracy—how much error, which norm, universality...our model:
  - Recover all signals in (relatively smaller) $\ell_1$ ball, by one matrix.

- Norm of error
  - Want $\ell_2$:

$$\|x - \hat{x}\|_2 \leq \frac{\epsilon}{\sqrt{k}} \|x - x_k\|_1.$$
Some criteria of algorithms

- **Number of measurements**: want $O(k \log N/k) \approx \log \left(\frac{N}{k}\right)$
- **Recovery runtime (speed)**:
  - Want $\text{poly}(k \log N)$.
  - Faster than previous *measurement-optimal* algorithms.
  - (“Sublinear time” measurement-optimal algos generally lose to efficient full-measurement algorithms, such as FFT.)
  - (Other interesting points on this spectrum—[HIKP’12])
- **Accuracy**—how much error, which norm, universality...our model:
  - Recover *all* signals in (relatively smaller) $\ell_1$ ball, by *one* matrix.
- **Norm of error**
  - Want $\ell_2$:
    $$\|x - \hat{x}\|_2 \leq \frac{\epsilon}{\sqrt{k}} \|x - x_k\|_1.$$  
  - Here get only $\ell_1$ (strictly worse):
    $$\|x - \hat{x}\|_1 \leq (1 + \epsilon)\|x - x_k\|_1.$$
## Some results

<table>
<thead>
<tr>
<th>Paper</th>
<th>No. meas.</th>
<th>time</th>
<th>norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GSTV07]</td>
<td>$k \ \text{polylog}$</td>
<td>$\text{poly}(k \ \log N)$</td>
<td>2</td>
</tr>
<tr>
<td>[Donoho04]</td>
<td>$k \ \log(N/k)$</td>
<td>$\text{poly}(N)$</td>
<td>2</td>
</tr>
<tr>
<td>[CRT04]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[RI08]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Red is optimal.
Many other results omitted.
Some results

<table>
<thead>
<tr>
<th>Paper</th>
<th>No. meas.</th>
<th>time</th>
<th>norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GSTV07]</td>
<td>$k \ \text{polylog}$</td>
<td>$\text{poly}(k \ \log N)$</td>
<td>2</td>
</tr>
<tr>
<td>[Donoho04]</td>
<td>$k \ \log(N/k)$</td>
<td>$\text{poly}(N)$</td>
<td>2</td>
</tr>
<tr>
<td>[CRT04]</td>
<td>$k \ \log(N/k)$</td>
<td>$N \ \log(N/k)$</td>
<td>1</td>
</tr>
<tr>
<td>[RI08]</td>
<td>$k \ \log(N/k)$</td>
<td>$\sqrt{kN}$</td>
<td></td>
</tr>
</tbody>
</table>

Red is optimal.

Many other results omitted.
## Some results

<table>
<thead>
<tr>
<th>Paper</th>
<th>No. meas.</th>
<th>time</th>
<th>norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GSTV07]</td>
<td>$k \ \text{polylog}$</td>
<td>$\text{poly}(k \ \log N)$</td>
<td>2</td>
</tr>
<tr>
<td>[Donoho04]</td>
<td>$k \ \log(N/k)$</td>
<td>$\text{poly}(N)$</td>
<td>2</td>
</tr>
<tr>
<td>[CRT04]</td>
<td>$k \ \log(N/k)$</td>
<td>$N \ \log(N/k)$</td>
<td>1</td>
</tr>
<tr>
<td>[RI08]</td>
<td>$k \ \log(N/k)$</td>
<td>$\sqrt{kN}$</td>
<td>1</td>
</tr>
<tr>
<td>Here</td>
<td>$k \ \log(N/k)$</td>
<td>$\text{poly}(\ell)k \ \log(N/k)$</td>
<td>1</td>
</tr>
<tr>
<td>Here</td>
<td>$\text{poly}(\ell)k \ \log(N/k)$</td>
<td>$\text{poly}(\ell)k(N/k)^{1/\ell}$</td>
<td>1</td>
</tr>
</tbody>
</table>

Red is optimal.

Many other results omitted.
Group testing 1-sparse signals

Group testing on 1-sparse signal. First half or second? Recover bit-by-bit:

$$\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}$$
Group testing 1-sparse signals

Group testing on 1-sparse signal. First half or second? Recover bit-by-bit:

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{pmatrix}
\cdot
\begin{pmatrix}
\text{noise} \\
\text{noise} \\
\text{noise} \\
\text{noise} \\
7 \\
\text{noise} \\
\text{noise} \\
\text{noise}
\end{pmatrix}
\]
Group testing 1-sparse signals

Group testing on 1-sparse signal. First half or second? Recover bit-by-bit:

\[
\begin{pmatrix}
7 \\
0 \\
7 \\
0 \\
\end{pmatrix}
\approx
\begin{pmatrix}
\text{Reference} \\
\text{Small} \\
\text{BIG} \\
\text{Small} \\
\end{pmatrix}
\approx
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{pmatrix}
\cdot
\begin{pmatrix}
\text{noise} \\
\text{noise} \\
\text{noise} \\
\text{noise} \\
\text{noise} \\
\text{noise} \\
\text{noise} \\
\text{noise} \\
\end{pmatrix}
\]
Techniques for some sublinear algorithms

- Hash into $k$ buckets (hope to isolate HH’s with low noise)

$$H = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Group testing on 1-sparse signal.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

- Typically lose log factor in meas. Top row of $H$ becomes:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$
Outline

1 Preliminaries
   - Problem we are addressing

2 Algorithm and Result

3 Analysis

4 Conclusion
Algorithm

- Hash into $B = \sqrt{kn}$ buckets;
Algorthm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate;

Hash

Aggregate

$\approx (k, B)$

$(k, N)$
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure

$$\text{Time} \approx (\text{no. buckets}) \cdot (\text{bucket size}) = k \left(\frac{N}{B}\right) = \sqrt{kN}$$
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure
- Repeat $\log(N/k)/\log(B/k) = 2$ times;
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure
- Repeat $\log(N/k)/\log(B/k) = 2$ times; collect measurements
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure
- Repeat $\log(N/k)/\log(B/k) = 2$ times; collect measurements
- Recursively find heavy buckets.

**Diagram**

```
Hash                                    (k, N)
Aggregate                                (∝ k, B)
Measure                                  k log(B/k)
Collect                                  k log(N/k)
```
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure
- Repeat $\log(N/k) / \log(B/k) = 2$ times; collect measurements
- Recursively find heavy buckets. Time $\approx$ length $= B = \sqrt{kN}$
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure
- Repeat $\log(N/k)/\log(B/k) = 2$ times; collect measurements
- Recursively find heavy buckets. Time $\approx$ length $= B = \sqrt{kN}$
- Lift solution (search heavy buckets).
Algorithm

- Hash into $B = \sqrt{kN}$ buckets; Aggregate; Measure
- Repeat $\log(N/k)/\log(B/k) = 2$ times; collect measurements
- Recursively find heavy buckets. Time $\approx$ length $= B = \sqrt{kN}$
- Lift solution (search heavy buckets).
  Time $\approx$ (no. buckets) $\cdot$ (bucket size) $= k(N/B) = \sqrt{kN}$
Theorem

Algorithm takes $\approx \sqrt{kN}$ time and uses $k \log(N/k)$ measurements.
Algorithm and Result

Result

Theorem

*Algorithm takes* $\approx \sqrt{kN}$ *time and uses* $k \log(N/k)$ *measurements.*

Next slides:

- Number of measurements and runtime
- Correctness of hashing procedure
- Correctness of recursive solution—easy
- Correctness of lifting—easy by (lazy) design
Outline

1. Preliminaries
   - Problem we are addressing

2. Algorithm and Result

3. Analysis

4. Conclusion
Replace one \((k, N)\) problem with two \((\approx k, B)\) recursive problems and two lifting problems, for \(B \approx \sqrt{kN}\).
Time and Measurements

- Replace one \((k, N)\) problem with two \(\approx (k, B)\) recursive problems and two lifting problems, for \(B \approx \sqrt{kn}\).
- Generate measurements only for recursive problems:
  \[2 \cdot k \log(B/k) = k \log(N/k).\]
  - Toplevel measurement matrix is 2 copies of matrix product of hashing \((k \text{ rows})\) and recovery \((\log \frac{B}{k} = \frac{1}{2} \log \frac{N}{k} \text{ rows})\) matrices.
Time and Measurements

- Replace one \((k, N)\) problem with two \((\approx k, B)\) recursive problems and two lifting problems, for \(B \approx \sqrt{kN}\).
- Generate measurements only for recursive problems:
  \[
  2 \cdot k \log(B/k) = k \log(N/k).
  \]
  - Toplevel measurement matrix is 2 copies of matrix product of hashing \((k \text{ rows})\) and recovery \((\log \frac{B}{k} = \frac{1}{2} \log \frac{N}{k} \text{ rows})\) matrices.
  - (Number of rows in measurement matrix is number of measurements.)
Replace one \((k, N)\) problem with two \((\approx k, B)\) recursive problems and two lifting problems, for \(B \approx \sqrt{kN}\).

Generate measurements only for recursive problems:
\[
2 \cdot k \log(B/k) = k \log(N/k).
\]

- Toplevel measurement matrix is 2 copies of matrix product of hashing \((k\) rows\) and recovery \((\log B/k = \frac{1}{2} \log \frac{N}{k}\) rows\) matrices.
- (Number of rows in measurement matrix is number of measurements.)

Recursive problems take time \(B\); lifting takes \(k(N/B)\).
Time and Measurements

- Replace one \((k, N)\) problem with two \((\approx k, B)\) recursive problems and two lifting problems, for \(B \approx \sqrt{kN}\).
- Generate measurements only for recursive problems:
  \[2 \cdot k \log(B/k) = k \log(N/k).\]
  - Toplevel measurement matrix is 2 copies of matrix product of hashing (\(k\) rows) and recovery (\(\log \frac{B}{k} = \frac{1}{2} \log \frac{N}{k}\) rows) matrices.
  - (Number of rows in measurement matrix is number of measurements.)
- Recursive problems take time \(B\); lifting takes \(k(N/B)\).
  - \(B = \sqrt{kN}\) is optimal; get time \(O(\sqrt{kN})\) up to log factors.
Correctness of Hashing

Lemma

*Intermediate signal is indeed \( \approx k\)-sparse and length \( B \).*
Correctness of Hashing

Lemma

*Intermediate signal is indeed $\approx k$-sparse and length $B$.*

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
Correctness of Hashing

Lemma

Intermedite signal is indeed $\approx k$-sparse and length $B$. 

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
- Each heavy hitter is isolated except with prob $k/B$. 

Failure probs drop to $(k/N)^k \leq (N/k)^{k-1}$.
Correctness of Hashing

Lemma

Intermediate signal is indeed $\approx k$-sparse and length $B$.

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
- Each heavy hitter is isolated except with prob $k/B$.
  - $\geq k/2$ fail with prob $(k/B)^{\Omega(k)} \approx 2^{-k \log(B/k)}$
Correctness of Hashing

Lemma

Intermediate signal is indeed $\approx k$-sparse and length $B$.

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
- Each heavy hitter is isolated except with prob $k/B$.
  - $\geq k/2$ fail with prob $(k/B)^{\Omega(k)} \approx 2^{-k \log(B/k)}$
- Heavy hitters land in set $S$ of about $k$ of $B$ buckets. Consider $t$ noise items of size $1/t$, for $t \geq k$:
Correctness of Hashing

**Lemma**

*Intermediate signal is indeed \( \approx k\)-sparse and length \( B \).*

- Hash \( N \) positions including \( k \) heavy hitters into \( B \) buckets.
- Each heavy hitter is isolated except with prob \( k/B \).
  - \( \geq k/2 \) fail with prob \( (k/B)^\Omega(k) \approx 2^{-k \log(B/k)} \)
- Heavy hitters land in set \( S \) of about \( k \) of \( B \) buckets. Consider \( t \) noise items of size \( 1/t \), for \( t \geq k \):
  - Each noise item lands in \( S \) with prob \( k/B \)
Correctness of Hashing

Lemma

*Intermediate signal is indeed $\approx k$-sparse and length $B$.*

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
- Each heavy hitter is isolated except with prob $k/B$.
  - $\geq k/2$ fail with prob $(k/B)^\Omega(k) \approx 2^{-k \log(B/k)}$
- Heavy hitters land in set $S$ of about $k$ of $B$ buckets. Consider $t$ noise items of size $1/t$, for $t \geq k$:
  - Each noise item lands in $S$ with prob $k/B$
  - $\geq t/2$ noise items land in $S$ with prob $(k/B)^\Omega(t) \approx 2^{-t \log(B/k)}$
Correctness of Hashing

Lemma

*Intermediate signal is indeed \( \approx k\text{-sparse and length } B\).*

- Hash \( N \) positions including \( k \) heavy hitters into \( B \) buckets.
- Each heavy hitter is isolated except with prob \( k/B \).
  - \( \geq k/2 \) fail with prob \( (k/B)^{\Omega(k)} \approx 2^{-k\log(B/k)} \)
- Heavy hitters land in set \( S \) of about \( k \) of \( B \) buckets. Consider \( t \) noise items of size \( 1/t \), for \( t \geq k \):
  - Each noise item lands in \( S \) with prob \( k/B \)
  - \( \geq t/2 \) noise items land in \( S \) with prob \( (k/B)^{\Omega(t)} \approx 2^{-t\log(B/k)} \)
    —handle mild dependence.

\[ \]
Correctness of Hashing

Lemma

Intermedidate signal is indeed $\approx k$-sparse and length $B$.

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
- Each heavy hitter is isolated except with prob $k/B$.
  - $\geq k/2$ fail with prob $(k/B)^{\Omega(k)} \approx 2^{-k \log(B/k)}$
- Heavy hitters land in set $S$ of about $k$ of $B$ buckets. Consider $t$ noise items of size $1/t$, for $t \geq k$:
  - Each noise item lands in $S$ with prob $k/B$
  - $\geq t/2$ noise items land in $S$ with prob $(k/B)^{\Omega(t)} \approx 2^{-t \log(B/k)}$
    —handle mild dependence. (Otherwise, enough of $S$ survives)
Correctness of Hashing

Lemma

*Intermediate signal is indeed \( \approx k \)-sparse and length \( B \).*

- Hash \( N \) positions including \( k \) heavy hitters into \( B \) buckets.
  - Each heavy hitter is isolated except with prob \( k/B \).
    - \( \geq k/2 \) fail with prob \( (k/B)^\Omega(k) \approx 2^{-k\log(B/k)} \)
- Heavy hitters land in set \( S \) of about \( k \) of \( B \) buckets. Consider \( t \) noise items of size \( 1/t \), for \( t \geq k \):
  - Each noise item lands in \( S \) with prob \( k/B \)
  - \( \geq t/2 \) noise items land in \( S \) with prob \( (k/B)^\Omega(t) \approx 2^{-t\log(B/k)} \)
    —handle mild dependence. (Otherwise, enough of \( S \) survives)
- Repeat \( \log(N/k)/\log(B/k) \) times
Correctness of Hashing

Lemma

Intermediate signal is indeed $\approx k$-sparse and length $B$.

- Hash $N$ positions including $k$ heavy hitters into $B$ buckets.
- Each heavy hitter is isolated except with prob $k/B$.
  - $\geq k/2$ fail with prob $(k/B)^\Omega(k) \approx 2^{-k \log(B/k)}$
- Heavy hitters land in set $S$ of about $k$ of $B$ buckets. Consider $t$ noise items of size $1/t$, for $t \geq k$:
  - Each noise item lands in $S$ with prob $k/B$
  - $\geq t/2$ noise items land in $S$ with prob $(k/B)^\Omega(t) \approx 2^{-t \log(B/k)}$ —handle mild dependence. (Otherwise, enough of $S$ survives)
- Repeat $\log(N/k)/\log(B/k)$ times
  - Failure probs drop to $(k/N)^k \leq \binom{N}{k}^{-1}$ and $(k/N)^t \leq \binom{N}{t}^{-1}$
  - Take union bound.
Correctness of recursive solution

- Use any algorithm with time polynomial in signal length, $B$
- For best results: use time around linear $B$
- Other algorithms can be dropped into this framework and/or
- This technique can be used for other algorithms, e.g., [HIKP12]
Correctness of lifting

- Generate \( \approx k \) recursive HH’s with \( N/B = \sqrt{N/k} \) preimages each; total \( \sqrt{kN} \) preimages.
Correctness of lifting

- Generate $\approx k$ recursive HH’s with $N/B = \sqrt{N/k}$ preimages each; total $\sqrt{kN}$ preimages.
- Look up preimages from table (hard to invert random hash, generally)
Correctness of lifting

- Generate $\approx k$ recursive HH’s with $N/B = \sqrt{N/k}$ preimages each; total $\sqrt{kN}$ preimages.
- Look up preimages from table (hard to invert random hash, generally)
- Testing all $\sqrt{kN}$ preimages in set $I \subseteq [N]$ takes time about $|I| = \sqrt{kN}$ (up to log factors).
Correctness of lifting

- Generate \( \approx k \) recursive HH’s with \( N/B = \sqrt{N/k} \) preimages each; total \( \sqrt{kN} \) preimages.
- Look up preimages from table (hard to invert random hash, generally)
- Testing all \( \sqrt{kN} \) preimages in set \( I \subseteq [N] \) takes time about \( |I| = \sqrt{kN} \) (up to log factors).
- Note: Must measure original length-\( N \) signal \textit{before} learning \( I \).
More generally...

- Cascade through any chosen number $\ell$ of levels.
- $\text{poly}(\ell)$ problems with parameters $(k, k(N/k)^{1/\ell})$
- Time around $\text{poly}(\ell)k(N/k)^{1/\ell}$
- Number of measurements is around $\text{poly}(\ell)k \log(N/k)$
Outline

1 Preliminaries
   • Problem we are addressing

2 Algorithm and Result

3 Analysis

4 Conclusion
Conclusion

- First sublinear-time algo with optimal measurements in forall model, with

\[ \| x - \hat{x} \|_1 \leq (1 + \epsilon)\| x - x_k \|_1. \]
Conclusion

- First sublinear-time algo with optimal measurements in forall model, with
  \[ \|x - \hat{x}\|_1 \leq (1 + \epsilon)\|x - x_k\|_1. \]

- Time \( \sqrt{kN \log^{O(1)}(N)} \).
  - To do: Improve to \((k \log N)^{O(1)}\) or \(k \log^{O(1)} N\). (Keep time near optimal as \(k \to 0\).)
  - To do: Improve to, e.g., \(k \log^{O(1)}(N/k)\). (Keep time sublinear as \(k \to N\).)
Conclusion

- First sublinear-time algo with optimal measurements in forall model, with
  
  \[ \| x - \hat{x} \|_1 \leq (1 + \epsilon) \| x - x_k \|_1. \]

- Time \( \sqrt{kN \log^O(1)(N)} \).
  - To do: Improve to \((k \log N)^O(1)\) or \(k \log^O(1) N\). (Keep time near optimal as \(k \to 0\).)
  - To do: Improve to, e.g., \(k \log^O(1)(N/k)\). (Keep time sublinear as \(k \to N\).)

- Lookup table of size \(Nk^{1/4}\).
  - To do: remove lookup table.
Conclusion

- First sublinear-time algo with optimal measurements in forall model, with
  \[ \|x - \hat{x}\|_1 \leq (1 + \epsilon)\|x - x_k\|_1. \]

- Time \( \sqrt{kN \log^O(1)(N)} \).
  - To do: Improve to \((k \log N)^O(1)\) or \(k \log^O(1) N\). (Keep time near optimal as \(k \to 0\).)
  - To do: Improve to, e.g., \(k \log^O(1)(N/k)\). (Keep time sublinear as \(k \to N\).)

- Lookup table of size \(Nk^{1/4}\).
  - To do: remove lookup table.

Finale is open: Improve to 2-norm:
\[ \|x - \hat{x}\|_2 \leq \frac{\epsilon}{\sqrt{k}}\|x - x_k\|_1. \]
Other topics

- Data privacy
Other topics

- Data privacy
- Geolocation and self localization—where am I?
Other topics

- Data privacy
- Geolocation and self localization—where am I?
- Energy-aware computing
Other topics

- Data privacy
- Geolocation and self localization—where am I?
- Energy-aware computing
- Outreach