One-bit Compressed Sensing: Provable Support and Vector Recovery

Praneeth Netrapalli

The University of Texas at Austin

Joint work with Sivakant Gopi, Prateek Jain and Aditya Nori

Jun 17, 2013
Goal: Reconstruct a sparse signal using very few linear measurements

Tremendous amount of work in the last decade

$O(k \log n)$ measurements to reconstruct $k$-sparse signals in $\mathbb{R}^n$

\(^1\)http://lions.epfl.ch/research
Quantization

- Measurements up to infinite precision - not practical
  - $y_j = -1.010001011110$
- Arbitrary quantization does not work well [BB08]
  - $y_j = -1.01$
- Extreme quantization - single bit measurements
  - $y_j = -1$
One-bit Compressed Sensing [BB08]

Goal: Reconstruct a sparse signal using \textit{signs} of very few linear measurements

Formally, given $y$ and $A$, recover $x$

Motivation:
- Captures extreme quantization
- Easy to implement
- Robust to noise
Exact recovery not possible:

For example, \( y = \text{Sign}(Ax) = \text{Sign}(A(cx)), \forall c > 0 \)

Frameworks

- Support recovery: recover \( \hat{x} \) such that
  \[
  \text{Supp}(x) = \text{Supp}(\hat{x})
  \]

- \( \epsilon \)-approximate recovery\(^a\): recover \( k \)-sparse \( \hat{x} \) such that
  \[
  \| \hat{x} - \frac{x}{\|x\|} \| < \epsilon
  \]

\(^a\| \cdot \| \) refers to two norm
Metrics

- Measurement (or sample) complexity: dimension of $y$
- Computational complexity: time taken by the recovery algorithm
- Universality: use the same measurement matrix $A$ for all sparse vectors $x$
  - critical for many applications e.g., single pixel camera
Our Results – Support Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[HB11]</th>
<th>UFF</th>
<th>Expanders</th>
</tr>
</thead>
</table>

First universal measurement schemes

Open problem: O(k log n) universal measurement scheme not known
## Our Results – Support Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[HB11]</th>
<th>UFF</th>
<th>Expanders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Our Results – Support Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[HB11]</th>
<th>UFF</th>
<th>Expanders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>-ve entries allowed?</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

-ve entries allowed? allows for the inclusion of negative entries in the support recovery process.
## Our Results – Support Recovery

<table>
<thead>
<tr>
<th></th>
<th>[HB11]</th>
<th>UFF</th>
<th>Expanders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>-ve entries allowed?</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td># measurements</td>
<td>$O(k \log n)$</td>
<td>$O(k^2 \log n)$</td>
<td>$O(k^3 \log n)$</td>
</tr>
</tbody>
</table>
## Our Results – Support Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[HB11]</th>
<th>UFF</th>
<th>Expanders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>-ve entries allowed?</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td># measurements</td>
<td>$O(k \log n)$</td>
<td>$O(k^2 \log n)$</td>
<td>$O(k^3 \log n)$</td>
</tr>
<tr>
<td>Running Time</td>
<td>$O(n \log n)$</td>
<td>$O(nk \log n)$</td>
<td>$O(nk \log n)$</td>
</tr>
</tbody>
</table>

- *First universal* measurement schemes
- Open problem: $O(k \log n)$ universal measurement scheme not known
Union Free Family

Definition (k-Union Free Family)

Sets $\mathcal{F} := \{B_1, \cdots, B_n\}$:

Well studied combinatorial objects
Use their structure in the measurement scheme and recovery algorithm
Works for $x \geq 0$
Another algorithm based on expanders, which works for any $x$
Definition (k-Union Free Family)

Sets $\mathcal{F} := \{B_1, \cdots, B_n\}$:

$$B_{i_0} \not\subset B_{i_1} \cup \cdots \cup B_{i_k},$$

for all distinct $B_{i_0}, B_{i_1}, \cdots, B_{i_k} \in \mathcal{F}$. 
Union Free Family

Definition (k-Union Free Family)

Sets $\mathcal{F} := \{ B_1, \cdots, B_n \}$:

$$B_{i_0} \not\subseteq B_{i_1} \cup \cdots \cup B_{i_k},$$

for all distinct $B_{i_0}, B_{i_1}, \cdots, B_{i_k} \in \mathcal{F}$. 
Definition (k-Union Free Family)

Sets $\mathcal{F} := \{B_1, \cdots, B_n\}$:

$$B_{i_0} \not\subseteq B_{i_1} \cup \cdots \cup B_{i_k},$$

for all distinct $B_{i_0}, B_{i_1}, \cdots, B_{i_k} \in \mathcal{F}$.
Union Free Family

Definition (k-Union Free Family)

Sets $\mathcal{F} := \{B_1, \cdots, B_n\}$:

$$B_{i_0} \not\subseteq B_{i_1} \cup \cdots \cup B_{i_k},$$

for all distinct $B_{i_0}, B_{i_1}, \cdots, B_{i_k} \in \mathcal{F}$. 

Element
\begin{itemize}
\item Green
\item Red
\item Black
\end{itemize}

Set
\begin{itemize}
\item Green
\item Red
\item Black
\end{itemize}
Union Free Family

Definition (k-Union Free Family)

Sets $\mathcal{F} := \{B_1, \cdots, B_n\}$:

$$B_{i_0} \not\subseteq B_{i_1} \cup \cdots \cup B_{i_k},$$

for all distinct $B_{i_0}, B_{i_1}, \cdots, B_{i_k} \in \mathcal{F}$.

- Well studied combinatorial objects
- Use their structure in the measurement scheme and recovery algorithm
- Works for $x \geq 0$
  - Another algorithm based on expanders, which works for any $x$
Experiments

Support Recovery (Error vs Sparsity)

Support Recovery (Error vs Number of Measurements (m))
- **Recall**: Need to recover \( \hat{x} \) such that
  \[ \left\| \hat{x} - \frac{x}{\|x\|} \right\| < \varepsilon \]

- Plan and Vershynin [PV11, PV12]: \( O(\varepsilon^{-5}) \) measurements
  - hard thresholding and soft thresholding based approaches

- **Our result**: \( \widetilde{O}(\varepsilon^{-1}) \) measurements

- **Key idea**: Combine optimal results from compressed sensing and learning halfspaces
### Our Results – Approximate Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[PV11, PV12]</th>
<th>Two-stage</th>
<th>S-Approx</th>
</tr>
</thead>
</table>

Praneeth Netrapalli
### Our Results – Approximate Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[PV11, PV12]</th>
<th>Two-stage</th>
<th>S-Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- **Algorithm**: [PV11, PV12]
- **Universal**: Yes
- **Two-stage**: Yes
- **S-Approx**: Yes
## Our Results – Approximate Recovery

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[PV11, PV12]</th>
<th>Two-stage</th>
<th>S-Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td># measurements ($m$)</td>
<td>$\frac{k}{\epsilon^5} \log^2 \frac{n}{k}$</td>
<td>$\frac{k}{\epsilon} \log \frac{n}{k}$</td>
<td>$k^3 \log \frac{n}{k} + \frac{k}{\epsilon}$</td>
</tr>
</tbody>
</table>
**Our Results – Approximate Recovery**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>[PV11, PV12]</th>
<th>Two-stage</th>
<th>S-Approx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td># measurements ((m))</td>
<td>(\frac{k}{\epsilon^5} \log^2 \frac{n}{k})</td>
<td>(\frac{k}{\epsilon} \log \frac{n}{k})</td>
<td>(k^3 \log \frac{n}{k} + \frac{k}{\epsilon})</td>
</tr>
<tr>
<td>Running Time</td>
<td>(\frac{kn}{\epsilon^6} \log \frac{n}{k})</td>
<td>(\frac{kn}{\epsilon^5} \log \frac{n}{k})</td>
<td>(\frac{k^5 n}{\epsilon^5} \log \frac{n}{k})</td>
</tr>
</tbody>
</table>

- **Two-stage**: Near optimal measurement complexity
1-bit CS Support Approx. Summary

Measurement Scheme and Algorithm

\[ y = \text{Sign} \left( A_2 A_1 x \right) \]

Random Gaussian Matrix \quad Compressed Sensing Matrix

Praneeth Netrapalli
Measurement Scheme and Algorithm

Algorithm

1. Linear programming: Obtain $\hat{z}$ such that $y = \text{Sign}(A_2\hat{z})$
Measurement Scheme and Algorithm

Algorithm

1. Linear programming: Obtain $\hat{z}$ such that $y = \text{Sign}(A_2\hat{z})$

2. Solve CS problem: $\hat{z} = A_1 x + e$ (GraDeS [GK09])
Proof Outline

Robust CS: sufficient to obtain $\hat{z}$ such that $\|\hat{z} - z\| < C\epsilon$.
- requires $O(k \log n)$ measurements

Obtain $\hat{z}$: requires $O\left(\frac{k \log n}{\epsilon} \log \left(\frac{k \log n}{\epsilon}\right)\right)$ random Gaussian measurements.

$$y = \text{Sign}(A_2 z)$$
$$\| A_1 x \|$$
Experiments

Approximate Recovery (Error vs Sparsity)

Approximate Recovery (Error vs m)
One-bit Compressed Sensing
- captures extreme quantization
- several other motivations

Support recovery
- *First universal* measurement schemes using UFF and expanders
  - UFF: $O\left(k^2 \log n\right)$ measurements but for $x \geq 0$
    - Can be used in other settings?
  - Expanders: $O\left(k^3 \log n\right)$ measurements for any $x$
- Open question: achieving $O\left(k \log n\right)$ sample complexity

$\epsilon$-approximate recovery
- *Near optimal* sample complexity with $\epsilon$
- Support recovery based algorithm - works well empirically
References

Petros Boufounos and Richard G. Baraniuk.
1-bit compressive sensing.
In CISS, pages 16–21, 2008.

Rahul Garg and Rohit Khandekar.
Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property.
In ICML, 2009.

Jarvis Haupt and Richard G. Baraniuk.
Robust support recovery using sparse compressive sensing matrices.
In CISS, pages 1–6, 2011.

Y. Plan and R. Vershynin.
One-bit compressed sensing by linear programming.

Yaniv Plan and Roman Vershynin.
Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach.