One-bit Compressed Sensing: Provable Support and Vector Recovery

Sivakant Gopi GOPISIVAKANTH@GMAIL.COM
IIT Bombay, Mumbai, India

Praneeth Netrapalli PRANEETHN@UTEXAS.EDU
The University of Texas at Austin, Austin, TX, 78705 USA

Prateek Jain PRAJAIN@MICROSOFT.COM
Microsoft Research India, Bangalore, India

Aditya Nori ADITYAN@MICROSOFT.COM
Microsoft Research India, Bangalore, India

Abstract

In this paper, we study the problem of one-bit compressed sensing (1-bit CS), where the goal is to design a measurement matrix A and a recovery algorithm such that a k-sparse unit vector $\boldsymbol{x}^*$ can be efficiently recovered from the sign of its linear measurements, i.e., $\boldsymbol{b} = \text{sign}(A\boldsymbol{x}^*)$. This is an important problem for signal acquisition and has several learning applications as well, e.g., multi-label classification (Hsu et al., 2009). We study this problem in two settings: a) support recovery: recover the support of $\boldsymbol{x}^*$, b) approximate vector recovery: recover a unit vector $\hat{\boldsymbol{x}}$ such that $\|\hat{\boldsymbol{x}} - \boldsymbol{x}^*\|_2 \leq \epsilon$. For support recovery, we propose two novel and efficient solutions based on two combinatorial structures: union free families of sets and expanders. In contrast to existing methods for support recovery, our methods are universal i.e. a single measurement matrix A can recover all the signals. For approximate recovery, we propose the first method to recover a sparse vector using a near optimal number of measurements. We also empirically validate our algorithms and demonstrate that our algorithms recover the true signal using fewer measurements than the existing methods.

Proceedings of the 30^{th} International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

1. Introduction

Several machine learning tasks require estimating a large number of parameters using a small number of training samples. In general, this problem is degenerate as many parameter vectors can be consistent with the same training data. However, recent works in the area of compressed sensing as well as high-dimensional statistics have shown that if the true parameter vector has certain structure (for example, sparsity, low-rank), then the estimation problem can be solved efficiently (Candès & Tao, 2005; Candès & Recht, 2009; Negahban et al., 2009).

The above problem can be studied in a number of different settings such as compressed sensing, statistical learning etc. In this paper, we mostly focus on the popular compressed sensing setting, where the goal is to design measurement matrices and recovery algorithms to estimate sparse vectors using a few linear measurements (Baraniuk et al., 2010). While the key application of this problem has been in the area of signal acquisition, it has also found applications in several learning related problems (Hsu et al., 2009; Duarte et al., 2008; Wright et al., 2010).

In compressive sensing, a k-sparse signal $\boldsymbol{x}^* \in \mathbb{R}^n$ is encoded as

$$\boldsymbol{b} = A\boldsymbol{x}^*, \quad A \in \mathbb{R}^{m \times n},$$

so that given $\boldsymbol{b}$ and A, the *sparse* signal $\boldsymbol{x}^*$ can be recovered exactly. In this paper, we mostly focus on recovering sparse signals; we briefly discuss extensions to other compressible signals in Section 4.1.

Several results in compressive sensing (Candès & Tao, 2005; Garg & Khandekar, 2009) have shown that $\boldsymbol{x}^*$

Table 1. Support Recovery: Comparison of different algorithms for 1-bit CS support recovery in terms of number of measurements required, running time and universality. Note that our methods require slightly more measurements but are universal which is critical for compressed sensing algorithms as sampling a new matrix A for each signal is practically infeasible.

Algorithm	No. of measurements (m)	Running Time	Universal	Negative $\boldsymbol{x}^*$ allowed
HB (Haupt & Baraniuk, 2011)	$\mathcal{O}(k \log n)$	$\mathcal{O}(n \log n)$	No	Yes
UFF (Algorithm 2)	$\mathcal{O}\left(k^2 \log n\right)$	$\mathcal{O}(nk \log n)$	Yes	No
Expanders (Algorithm 3)	$\mathcal{O}\left(k^3 \log n\right)$	$\mathcal{O}(nk \log n)$	Yes	Yes

Table 2. Approximate Vector Recovery: Comparison of different algorithms for 1-bit CS approximate vector recovery. Note that both our methods have a dependence on error (ϵ) that is near optimal while existing methods require a significantly larger number of measurements and in typical settings, higher running time complexity as well.

Algorithm	No. of measurements (m)	Running Time	Universal
Plan and Vershynin (Plan & Vershynin, 2011)	$\mathcal{O}\left(\frac{1}{\epsilon^5} k \log^2 \frac{n}{k}\right)$	$\mathcal{O}\left(\frac{1}{\epsilon^5} kn^4 \log^2 \frac{n}{k}\right)$	Yes
Plan and Vershynin (Plan & Vershynin, 2012)	$\mathcal{O}\left(\frac{1}{\epsilon^6} k \log \frac{n}{k}\right)$	$\mathcal{O}\left(\frac{1}{\epsilon^6} nk \log \frac{n}{k}\right)$	Yes
Two-stage Algorithm (Algorithm 6)	$\widetilde{\mathcal{O}}\left(\frac{1}{\epsilon} k \log \frac{n}{k}\right)$	$\widetilde{\mathcal{O}}\left(nk \log \frac{n}{k} + \frac{1}{\epsilon^5}(k \log \frac{n}{k})^5\right)$	Yes
S-Approx (Algorithm 7)	$\widetilde{\mathcal{O}}\left(k^3 \log \frac{n}{k} + \frac{k}{\epsilon}\right)$	$\widetilde{\mathcal{O}}\left(nk \log \frac{n}{k} + \frac{k^5}{\epsilon^5}\right)$	Yes

can be recovered using only $m = \mathcal{O}(k \log n)$ linear measurements. However, all the above approaches require the measurements $\boldsymbol{b}$ to be known *exactly* (up to infinite precision). Naturally, this requirement is not practical, e.g., image sensors cannot store measurements up to arbitrary accuracy. Furthermore, arbitrarily quantized measurements might lead to large errors in recovery.

To address this issue and to simplify the signal acquisition process, (Boufounos & Baraniuk, 2008) introduced the problem of *one-bit compressed sensing* (or 1-bit CS) where only one bit of the linear measurements, specifically their signs are observed. In particular, given A and

$$\boldsymbol{b} = \text{sign}(A\boldsymbol{x}^*), \tag{1}$$

we need to recover the k-sparse signal $\boldsymbol{x}^*$. Apart from ease of their implementation using comparators, the above measurements are also known to be more robust to noise and non-linearity, and in certain situations perform better than standard compressive sensing (Laska & Baraniuk, 2012).

Note that using 1-bit measurements (1), we cannot recover the norm of $\boldsymbol{x}^*$ from $\boldsymbol{b}$ because scaling $\boldsymbol{x}^*$ does not change the measurements. Similarly, a small perturbation in $\boldsymbol{x}^*$ may not change $\boldsymbol{b}$. Therefore, exact recovery of $\boldsymbol{x}^*$ is in general not possible, even when $\boldsymbol{x}^*$ is a unit vector.

Instead, 1-bit CS is typically studied in these two settings:

Support recovery: recover the support of $\boldsymbol{x}^*$,

Approximate vector recovery: recover $\hat{\boldsymbol{x}}$ that is close to $\boldsymbol{x}^*$ (up to normalization), i.e., $\left\| \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|_2} - \frac{\boldsymbol{x}^*}{\|\boldsymbol{x}^*\|_2} \right\|_2 \leq \epsilon$, where $\epsilon > 0$ is a given approximation factor.

For both of the above problems, a solution is evaluated on the following three critical parameters: 1) Number of measurements (m), 2) Running time of the recovery algorithm, and 3) Universality of the measurement matrix. A 1-bit CS method is universal if a *fixed* design matrix A can be used to recover *all* sparse signals. Note that universality is a crucial requirement, as it is practically infeasible in several 1-bit CS applications (for instance, a single-pixel camera) to construct a new A for each signal.

In this paper, we study one-bit compressive sensing in both the above mentioned settings and improve upon the state-of-the-art results in those settings.

1.1. Support Recovery

Existing work: The best known solution for support recovery is by (Haupt & Baraniuk, 2011) that uses $O(k \log n)$ measurements. However, their solution is not universal, which is crucial for several real-world applications.

Our Results: We propose the first universal measurement matrices for support recovery in the 1-bit compressed sensing problem. Our solutions are based on two combinatorial structures, called union free sets and expander graphs. Compared to existing work, our measurement schemes however require a factor of $O(k)$ and $O(k^2)$ more measurements respectively. See Table 1 for a comparison of our methods with the method by (Haupt & Baraniuk, 2011) with respect to the above mentioned critical problem parameters. We would like to note that while expanders have previously been used in compressed sensing (Jafarpour et al., 2009), to the best of our knowledge, union free sets have so far not been used in this domain and might have applications to other related tasks as well.

1.2. Approximate Recovery

Existing work: (Plan & Vershynin, 2011) and (Plan & Vershynin, 2012) provide provable and efficient recovery methods for this problem. In particular, (Plan & Vershynin, 2012) provides a powerful framework for recovering a large class of compressible signals using only one-bit measurements. However, the number of measurements required by both (Plan & Vershynin, 2011) and (Plan & Vershynin, 2012) are sub-optimal in the dependence on ϵ $\left(\mathcal{O}\left(\epsilon^{-5}\right)\right.$ and $\left.\mathcal{O}\left(\epsilon^{-6}\right)\right.$ respectively$\left.\right)$.
Our Results: We propose a novel solution that exploits well-known results for the standard compressed sensing problem to guarantee recovery using an optimal number of measurements, i.e., $\mathcal{O}\left(\epsilon^{-1}\right)$ – see supplementary material for a lower bound. See Table 2 for a comparison of our proposed method with the existing methods.

Finally, our experimental results show that our methods are also empirically competitive with existing methods. Since the focus of this paper is on practical and provable methods for 1-bit CS, we draw a comparison only against known state-of-the-art provable methods.

Notation: We denote vectors using bold-faced letters (e.g., $\boldsymbol{x}$) and matrices using capital letters (e.g., A). x_i denotes the i-th element of $\boldsymbol{x}$, and $\mathbf{a}^{(i)}$ denotes i-th row of A. $\boldsymbol{x}(S)$ denotes elements of $\boldsymbol{x}$ restricted to set S, and $A(S)$ denotes columns of A from set S. $A \in \mathbb{R}^{m\times n}$ denotes a design matrix, and $\boldsymbol{x}^* \in \mathbb{R}^n$ denotes the true signal. $\|\boldsymbol{x}\|_p$ denotes the ℓ_p norm of $\boldsymbol{x}$, and $\|\boldsymbol{x}\|_0$ denotes the number of non-zeros in $\boldsymbol{x}$. supp$(\boldsymbol{x})$ denotes the set of non-zero elements of $\boldsymbol{x}$. We use $\widetilde{\mathcal{O}}\left(\cdot\right)$ to ignore $poly(\log k + \log\log n)$ factors. Wlog stands for with out loss of generality, and s.t. stands for such that.

2. Related Work

Compressive sensing (CS) using precise linear measurements is a well-studied problem and several methods (Candès & Tao, 2005; Tropp & Gilbert, 2007; Jafarpour et al., 2009) are known to achieve efficient recovery using a near optimal number of measurements. In comparison, the problem of 1-bit CS is relatively new and the state-of-the-art algorithms still lack in certain regards. For support recovery, existing algorithms are not universal while for approximate recovery they do not have information theoretic optimal measurement complexity bounds (See previous section for more detailed discussion).

Apart from provable recovery methods, several heuristics have also been proposed for this problem (Boufounos, 2009; Laska et al., 2011; Jacques et al., 2011); these methods have good empirical performance but lack theoretical guarantees. Apart from the standard 1-bit CS problem, several variants/extensions have also been studied. For instance, (Davenport et al., 2012) recently studied a similar problem called 1-bit matrix completion . (Ai et al., 2012) recently extended recovery results to measurement matrices A sampled from more general sub-Gaussian distributions.

3. Support Recovery

Problem statement: Design a measurement matrix $A \in \mathbb{R}^{m\times n}$, and a recovery algorithm for the following problem: given $\boldsymbol{b} = \text{sign}(A\boldsymbol{x}^*)$ with $\boldsymbol{x}^* \in \mathbb{R}^n, \|\boldsymbol{x}^*\|_0 \leq k$, find supp$(\boldsymbol{x}^*)$.
For this problem, we propose two different approaches based on: a) union free family (UFF) of sets, b) expander graphs. For both these approaches, we provide the design matrix as well as the corresponding recovery algorithm.

3.1. Support Recovery using Union Free Family (UFF)

In this section, we describe an efficient algorithm that recovers the support of any *non-negative* vector using $\mathcal{O}\left(k^2\log n\right)$ measurements.

3.1.1. UFF Background

Let U be a fixed set, and let $B_i \subseteq U, 1 \leq i \leq n$. Then, the family of sets $\mathcal{F} = \{B_1, \cdots, B_n\}$ is said to be k-union free if no B_i lies in the union of *any* other k sets from $\mathcal{F}$.

Definition 1. *A family of sets $\mathcal{F}$: $= \{B_1, \cdots, B_n\}$ with underlying set $U = \cup_{i=1}^n B_i$ is called a k-union-free family (k-UFF) iff: $B_{i_0} \nsubseteq B_{i_1} \cup \cdots \cup B_{i_k}$, for all distinct $B_{i_0}, B_{i_1}, \cdots, B_{i_k} \in \mathcal{F}$.*

Definition 2. *A k-UFF is called d-regular-k-union-free (d,k)-UFF if: $\forall B_i \in \mathcal{F}$, $|B_i| = d$.*

The following theorem from (Erdös et al., 1982) guarantees existence of a large (d,k)-UFF.

Theorem 1. *(Erdös et al., 1982) Let $n(m,k,d)$ denote the maximum cardinality of a (d,k)-UFF over an m-element underlying set ($|U| = m$). Let $h = \lceil \frac{d}{k} \rceil$. Then,*

$$n(m,k,d) \geq \binom{m}{h} / \binom{kh}{h}^2$$

It is well known that such a family can be constructed using a randomized method: form subsets $B_i, 1 \leq i \leq n$ by selecting d elements of the underlying set U uniformly at random. The algorithm is formally given in Algorithm 1.

Algorithm 1 Probabilistic construction of a size n, (d,k)-UFF from an m-element underlying set.

input m, n, d
$U \leftarrow \{1,2,\ldots,m\}$, $\quad \mathcal{F} \leftarrow \emptyset$
for $i = 1, \cdots, n$ **do**
  Obtain B_i by randomly sampling d distinct elements of U with replacement
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{B_i\}$
end for
output $\mathcal{F}$

The following theorem shows that with high probability, for appropriate choice of the parameters m and d, Algorithm 1 outputs a (d,k)-UFF with high probability.

Theorem 2. *For* $m = 10k^2 \log\left(\frac{3n}{\delta}\right)$ *and* $d = k\log\left(\frac{3n}{\delta}\right)$*, Algorithm 1 outputs a* (d,k)*-UFF with probability greater than* $1-\delta$*.*

We provide a proof of Theorem 2 in the supplementary material.

3.1.2. UFF based sensing matrix

We now provide a novel method of constructing measurement matrix A using a given (d,k)-UFF.

Using the randomized construction mentioned in Algorithm 1, construct a $\left(k\log\left(\frac{3n}{\delta}\right), k\right)$-UFF $\mathcal{F}$, where $|\mathcal{F}| = n$ with underlying set $U = \{1,2,\ldots,m\}$. Note that, from Theorem 2, we can choose $m = 10k^2\log\left(\frac{3n}{\delta}\right)$.

Next, the sensing matrix A is defined as follows:

$$A_{ij} = \mathbb{1}_{\{i \in B_j\}}. \tag{2}$$

That is, the j-th column of A is the incidence vector of B_j. Also, if $\boldsymbol{x}^* \in \mathbb{R}^n$ is a non-negative vector, then $A\boldsymbol{x}^* \geq 0$. Hence, the i^{th} measurement $b_i = \text{sign}(\mathbf{a}^{(i)}\boldsymbol{x}^*)$ is given by:

$$b_i = \mathbb{1}_{\{\sum_{j:i\in B_j} x_j^* > 0\}}. \tag{3}$$

Note that $A \in \{0,1\}^{m\times n}$, $b \in \{0,1\}^m$ where $m = O(k^2 \log n)$.

Support Recovery Algorithm: We now present our support recovery algorithm that estimates the support set $\widehat{S}$ of $\boldsymbol{x}^*$ using measurements $\boldsymbol{b}$ constructed by the above described (d,k)-UFF based design matrix A.

Our algorithm proceeds in n steps: at the j-th step $(1 \leq j \leq n)$, we add element j to the support set $\widehat{S}$ if $\min_{i\in B_j} b_i$ is positive. See Algorithm 2 for a detailed pseudo code.

The following theorem proved in the supplementary material establishes the correctness of Algorithm 2.

Algorithm 2 UFF based Support Recovery (UFF)

input A : measurement matrix, $\boldsymbol{b}$: measurement vector ($\boldsymbol{b} = \text{sign}(A\boldsymbol{x}^*)$)
$\widehat{S} \leftarrow \emptyset$
for $j = 1, \cdots, n$ **do**
  if $\min_{i\in B_j} b_i > 0$ **then**
    $\widehat{S} \leftarrow \widehat{S} \cup \{j\}$
  end if
end for
output $\widehat{S}$

Theorem 3. *Suppose* $\boldsymbol{x}^* \in \mathbb{R}^n$ *is a non-negative vector s.t.* $\|\boldsymbol{x}^*\|_0 \leq k$*, A is a sensing matrix constructed according to* (2) *and* $\boldsymbol{b}$ *is computed using* (3)*. Then, the set* $\widehat{S}$ *returned by Algorithm 2 satisfies:* $\widehat{S} = supp(\boldsymbol{x}^*)$*.*

3.1.3. Discussion

Above, we described our UFF based algorithm for the support recovery problem. Note that, the algorithm is universal, i.e., one design matrix A can be used to recover the support of every k-sparse non-negative vector. Furthermore, the algorithm is efficient, with time complexity $O(nk\log n)$, and is easy to implement.

Also, note that the measurements given by (3) are binary measurements (i.e., in $\{0,1\}^m$) rather than signed measurements (i.e., in $\{-1,1\}^m$), but they are essentially of the same nature.

Robustness to noise: Note that Algorithm 2 requires *exact* measurements without any noise. However, we can easily extend our method for handling arbitrary adversarial noise in measurements $\boldsymbol{b}$. That is, for the case where value of a small number of b_i's can be flipped arbitrarily. To this end, we use the following robust version of UFFs:

Definition 3. *A family of sets* $\mathcal{F} = \{B_1, B_2 \cdots, B_n\}$ *is called a* (d,k,ϵ)*-UFF if* $|B_{i_0} \cap (B_{i_1} \cup B_{i_2} \cup \cdots \cup B_{i_k})| < \epsilon|B_{i_0}|$ *holds for all distinct* $B_{i_0}, B_{i_1}, \cdots B_{i_k} \in \mathcal{F}$ *and each set in* $\mathcal{F}$ *has size* d*.*

Theorem 4. *(de Wolf, 2012) There exists a* (d,k,ϵ)*-UFF* $\mathcal{F}$ *over an* m*-element underlying set such that* $|\mathcal{F}| = n, m = \mathcal{O}(\frac{k^2\log n}{\epsilon^2}), d = \mathcal{O}(\frac{k\log n}{\epsilon})$*.*

Using a (d,k,ϵ) UFF as in Theorem 4, Algorithm 2 can be modified to make it robust up to $\left(\frac{1}{2}-\epsilon\right)d$ adversarial errors, i.e., $\left(\frac{1}{2}-\epsilon\right)d$ arbitrarily flipped measurements. See Algorithm 8 and Theorem 8 in the supplementary material for further details.

Handling Vectors with Negative Elements: One drawback of our UFF based algorithm is that it cannot handle cases where the underlying signal $\boldsymbol{x}^*$ has negative entries. A solution to this problem is to select each non-zero A_{ij} uniformly at random from $[0,1]$ (instead

of fixing it to be 1). This will ensure the following with probability 1:

- if $\ell \in S^*$ then $\sum_{j:i\in B_j} x^*_j \neq 0 \; \forall i \in B_\ell$, and
- if $\ell \notin S^*$ then $\exists j^* \in B_l$ such that $\sum_{i:j^*\in B_i} x^*_j = 0$

The above observations along with proof of Theorem 3 shows that we can recover support of $\boldsymbol{x}^*$ even if it has negative entries. However, a drawback of this solution is that the resulting algorithm is not *universal* as the values of A need to be sampled afresh for each $\boldsymbol{x}^*$. In the next section, we present a solution based on expanders that can handle vectors with negative elements and is universal as well (although with higher measurement complexity).

3.2. Support Recovery using Expanders

We now describe an expanders based algorithm that recovers the support using $\mathcal{O}\left(k^3 \log n\right)$ measurements.

3.2.1. Expanders Background

A left regular bipartite graph is called an expander if every small enough subset of the nodes on the left have a large enough neighborhood set on the right.

Definition 4. *A d-left-regular bipartite graph (U, V), s.t. $|U| = n, |V| = m$, is an (n, m, d, k, ϵ)-expander if*
$$\forall S \subset U,\ |S| \leq k \Rightarrow |N(S)| > (1-\epsilon)d|S|,$$
where $N(S)$ is the neighborhood of nodes in set S.

Expanders are closely related to UFF and hence can be constructed in the same way as in Algorithm 1. For the sake of completeness, we recall the following result that establishes the existence of good expanders.

Lemma 1. *(Claim 1, (Berinde & Indyk, 2008)) For any $n/2 \geq k \geq 1, \epsilon > 0$ there exists an (n, m, d, k, ϵ) expander with $d = \mathcal{O}\left(\frac{\log \frac{n}{k}}{\epsilon}\right)$ and $m = \mathcal{O}\left(\frac{k \log \frac{n}{k}}{\epsilon^2}\right)$*

3.2.2. Expander based Sensing Matrix

In this section, we present a method to construct a sensing matrix A using a given expander. We first construct a $(n, m, d, k+1, \epsilon)$-expander with $\epsilon = \frac{1}{16k}$. Using Theorem 1, we can choose $m = \mathcal{O}(k^3 \log \frac{n}{k})$ and $d = \mathcal{O}(k \log \frac{n}{k})$. Let $A \in \mathbb{R}^{m\times n}$ be the adjacency matrix of the expander:

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \text{ is an edge of the expander} \\ 0 & \text{otherwise.} \end{cases}$$

Then, we use A as the sensing matrix and observe $\boldsymbol{b} = \text{sign}(A\boldsymbol{x}^*)$, with $\boldsymbol{x}^* \in \mathbb{R}^n,\ \|\boldsymbol{x}^*\|_0 \leq k$.

Support Recovery Algorithm: We now present our support recovery algorithm that estimates support set $\widehat{S}$ of $\boldsymbol{x}^*$ using measurements $\boldsymbol{b}$ constructed using the above described design matrix A. Our algorithm proceeds in n steps: at the j-th step ($1 \leq j \leq n$), we add element j to $\widehat{S}$ if the measurement corresponding to at least half of the neighbors of j (i.e. $N(j)$) are non-zero i.e.,$|N(j) \cap \text{supp}(\boldsymbol{b})| > \frac{d}{2}$. See Algorithm 3 for a detailed pseudo code.

Algorithm 3 Support recovery algorithm when A is constructed from a (n, m, d, k, ϵ)-expander.

input A : measurement matrix, $\boldsymbol{b}$: measurement vector ($\boldsymbol{b} = \text{sign}(Ax^*)$)
$\widehat{S} \leftarrow \emptyset$
for $j = 1, \cdots, n$ **do**
  if $|N(j) \cap \text{supp}(\boldsymbol{b})| > \frac{d}{2}$ **then**
    $\widehat{S} \leftarrow \widehat{S} \cup \{j\}$
  end if
end for
output $\widehat{S}$

The following theorem proved in the supplementary material shows correctness of Algorithm 3.

Theorem 5. *Let $\boldsymbol{b} = sign(A\boldsymbol{x}^*)$ with k-sparse $\boldsymbol{x}^* \in \mathbb{R}^n$ and A as constructed in Section 3.2.2, then Algorithm 3 correctly identifies $S^* = supp(\boldsymbol{x}^*)$, i.e., $\widehat{S} = S^*$.*

Discussion: Note that Algorithm 3 can exactly recover $\boldsymbol{x}^* \in \{-1, 0, 1\}^n$: first recover $\text{supp}(\boldsymbol{x}^*)$ and then set sign of each element $\widehat{x}_j$ ($j \in \widehat{S}$) to be the sign of the majority of elements in $N(j) \cap \text{supp}(\boldsymbol{b})$.

Robustness: for our choice of parameters, the algorithm can tolerate up to $\frac{d}{4}$ adversarial bit flips in $\boldsymbol{b}$. Robustness up to d adversarial errors can be obtained by choosing graphs with better expansion property.

Finally, observe that the computational complexity of Algorithm 3 is $O\left(nk \log \frac{n}{k}\right)$.

3.2.3. Divide and Conquer

In this section, we present a "Divide and Conquer" approach that in conjunction with our support recovery algorithms can achieve even lower measurement complexity than our support recovery algorithms. However, the obtained approach is no longer universal.

The key idea is to first partition the n coordinates of x^* into k disjoint random sets of equal size; Wlog we can assume that k divides n. Since the sparsity of x^* is k, on an average, each of the random partitions has sparsity 1. Using standard concentration bounds, with high probability, each of the partitions has at most $O(\log k)$ non-zeros. We can then use our algorithms from Sections 3.1 or 3.2 to recover the support of each of the k subsets.

Algorithm 4 Measurements for Algorithm 5

input m, n, d, k

1: $P \leftarrow$ random permutation matrix
2: Generate $A'_1, A'_2, \ldots, A'_k$ using Algorithm 1 with input $(\frac{m}{k}, \frac{n}{k}, d)$
3: B : generate block diagonal matrix using $A'_1, \ldots, A'_k$ similar to (4)

output $B \cdot P$

Algorithm 5 Support Recovery for Divide and Conquer Approach

input A'_ℓ : ℓ-th block UFF-based sensing matrix, B: block matrix (4), P : permutation matrix, $\boldsymbol{b}$: measurement vector ($\boldsymbol{b} = \text{sign}(B \cdot Px^*)$)

1: $\widehat{S} \leftarrow \emptyset$
2: **for** $l = 1, \cdots, k$ **do**
3: $b_\ell \leftarrow \boldsymbol{b}((\ell-1)\frac{m}{k}, \cdots, \ell\frac{m}{k} - 1)$ i.e. ℓ-th block of $\boldsymbol{b}$
4: Run Algorithm 2 on b_ℓ and A'_ℓ to recover $\widehat{S}_\ell$
5: $\widehat{S} \leftarrow \widehat{S} \cup P^{-1}(\widehat{S}_\ell)$
6: **end for**

output $\widehat{S}$

Similar to the result by (Haupt & Baraniuk, 2011), we can show that the number of measurements needed by this approach is optimal up to poly $(\log k)$, although the obtained approach is not universal. Algorithm 4 provides a pseudo-code for generating sensing matrix and Algorithm 5 provides the recovery algorithm.

Construction of the measurement matrix: The measurement matrix A is given by $A = B \cdot P$ where P is a random permutation matrix and B is a block diagonal matrix with k equal blocks, each block being a UFF-based matrix A'_ℓ, constructed using Algorithm 1 with parameters $(\frac{m}{k}, \frac{n}{k}, d)$.

$$B = \begin{bmatrix} A'_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A'_k \end{bmatrix}, \tag{4}$$

where A'_ℓ =Algorithm 1$(\frac{m}{k}, \frac{n}{k}, d)$

Theorem 6. *Suppose $\boldsymbol{x}^* \in \mathbb{R}^n_+$ s.t. $||\boldsymbol{x}^*||_0 \leq k$, $A = B \cdot P$ is a sensing matrix as in Algorithm 4 with $m = \widetilde{\mathcal{O}}\left(k \log \frac{n}{k}\right)$ and $d = \mathcal{O}\left(\log k \log \frac{n}{k}\right)$. Then, Algorithm 5 returns supp$(\boldsymbol{x}^*)$ in time $\widetilde{\mathcal{O}}\left(n \log \frac{n}{k}\right)$ with probability at least $1 - e^{-\widetilde{\Omega}(\log k)}$.*

See the supplementary material for a detailed proof.

4. Approximate Vector Recovery

Problem Statement: Design matrix $A \in \mathbb{R}^{m \times n}$ and an algorithm to solve: given $\boldsymbol{b} = \text{sign}(A\boldsymbol{x}^*)$ (where $\boldsymbol{x}^* \in \mathbb{R}^n$, $\|\boldsymbol{x}^*\|_0 \leq k$ and $\|\boldsymbol{x}^*\|_2 = 1$), output $\hat{\boldsymbol{x}}$ such that:

$$\left\| \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|_2} - \boldsymbol{x}^* \right\|_2 \leq \epsilon,$$

where $\epsilon > 0$ is a given tolerance parameter. Note that assuming $\boldsymbol{x}^*$ to be of unit norm entails no loss of generality since scaling $\boldsymbol{x}^*$ doesn't change $\boldsymbol{b}$. In particular, we can never recover $\|\boldsymbol{x}^*\|_2$ from b.

For this problem, we propose two novel solutions which are both universal, provide their measurement complexity and also provide their time complexity. Our first solution is based on combining standard compressed sensing techniques with Gaussian measurements (see Section 4.1). Our second method first recovers the true support using methods of Section 3 and then uses Gaussian measurements to approximately recover elements of $\boldsymbol{x}^*$ (see Section 4.1).

4.1. Two-stage Approximate Recovery

In this section, we present our first approach for two-stage approximate recovery that exploits existing compressed sensing methods. Broadly, we design the measurement matrix A as a product of two matrices $A_1 \in \mathbb{R}^{m' \times n}$ and $A_2 \in \mathbb{R}^{m \times m'}$ (i.e., $A = A_2 A_1$). We select A_1 to be a standard compressed sensing matrix and A_2 to be an iid Gaussian matrix. So the measurements are:

$$\boldsymbol{b} = \text{sign}(A_2 A_1 \boldsymbol{x}^*).$$

Now, let $\boldsymbol{z}^* = A_1\boldsymbol{x}^*$ and $\boldsymbol{b} = \text{sign}(A_2\boldsymbol{z}^*)$. The main idea is that given $\boldsymbol{b}$ and A_2, we can find a vector $\widehat{\boldsymbol{z}}$ that satisfies each of the measurement, i.e., $\text{sign}(A_2\widehat{\boldsymbol{z}}) = \boldsymbol{b}$. Furthermore, using Theorem 10 (Theorem 2, Jacques et al. (2011)), $\widehat{\boldsymbol{z}}$ should closely approximate $\boldsymbol{z}^*$. Next, given $\widehat{\boldsymbol{z}}$ and A_1, using standard compressed sensing algorithms we estimate $\hat{\boldsymbol{x}}$ which should be a close approximation to $\boldsymbol{x}^*$.

Construction of the measurement matrix: Let $A = A_2 \cdot A_1$ where $A_1 \in \mathbb{R}^{m' \times n}$ and $A_2 \in \mathbb{R}^{m \times m'}$. A_1 is a matrix that satisfies the restricted isometry property (RIP) (Candès & Tao, 2005) with $\delta_{2k} < \frac{1}{6}$. A_1 is said to satisfy k-RIP with constant δ_k if, $\forall \boldsymbol{x} \in \mathbb{R}^n, \|\boldsymbol{x}\|_0 \leq k$:

$$(1 - \delta_k)\|\boldsymbol{x}\|_2^2 \leq \|A_1\boldsymbol{x}\|_2^2 \leq (1 + \delta_k)\|\boldsymbol{x}\|_2^2.$$

Also, if $m' = O(k \log \frac{n}{k})$ and each entry of A_1 is sampled from a centered sub-Gaussian then A_1 satisfies $2k$-RIP with constant $\delta_{2k} < \frac{1}{6}$ (Candès & Tao, 2005).

Next, select $m = \mathcal{O}\left(\frac{1}{\epsilon} m' \log \frac{m'}{\epsilon}\right) = \widetilde{\mathcal{O}}\left(\frac{1}{\epsilon} k \log \frac{n}{k}\right)$ and sample each entry of A_2 independently from $\mathcal{N}(0, 1)$. Using Theorem 10 (supplementary material) by (Jacques et al., 2011), with high probability such a

Algorithm 6 Two-stage Approximate Recovery

input A_1, A_2: measurement matrices (see Section 4.1), $\boldsymbol{b}$: measurement vector ($\boldsymbol{b} = \text{sign}(A_2 A_1 \boldsymbol{x}^*)$)

1: *Stage 1*: Run an LP solver for the following LP: find $\widehat{z}$ s.t. $b_i \mathbf{a}_2^{(i)} \widehat{z} > 0, \forall i$.
2: *Stage 2*: Run GradeS algorithm (Garg & Khandekar, 2009) (supplementary material) with inputs $\widehat{\boldsymbol{z}}$ and A_1 to obtain $\widehat{\boldsymbol{x}}$

output $\widehat{\boldsymbol{x}}$

measurement matrix ensures that:

$$\forall \boldsymbol{x}, \boldsymbol{y},\ \text{sign}(A_2\boldsymbol{x}) = \text{sign}(A_2\boldsymbol{y}) \Rightarrow \left\| \frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2} - \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|_2} \right\|_2 \leq \epsilon.$$

Algorithm for approximate recovery: In this section, we present our two-stage algorithm for approximate recovery. As mentioned earlier, the algorithm first uses a half space learning algorithm to obtain an estimate $\widehat{\boldsymbol{z}}$ of $A_1\boldsymbol{x}^*$ and then uses the GradeS algorithm, a robust compressed sensing algorithm by (Garg & Khandekar, 2009), on $\widehat{\boldsymbol{z}}$ to obtain an estimate $\widehat{\boldsymbol{x}}$ of $\boldsymbol{x}^*$. See Algorithm 6 for a pseudo-code of our approach. For completeness, we provide the GradeS algorithm in the supplementary material.

Below, we provide proof of correctness of Algorithm 6.

Theorem 7. *Let $\boldsymbol{x}^* \in \mathbb{R}^n$ be a k-sparse vector with $\|\boldsymbol{x}^*\|_2 = 1$ and let A_1 and A_2 be chosen as described in the previous section. Also, let $\boldsymbol{b} = \text{sign}(A_2 A_1 \boldsymbol{x}^*)$. Then for $\widehat{\boldsymbol{x}}$ returned by Algorithm 6, we have, $\left\| \frac{\widehat{\boldsymbol{x}}}{\|\widehat{\boldsymbol{x}}\|_2} - \boldsymbol{x}^* \right\|_2 \leq 20\epsilon$, where $0 < \epsilon < \frac{1}{4}$.*

See the supplementary material for a detailed proof.

Note that the computational complexity of solving the LP in Stage 1 of our algorithm can be bounded by $\mathcal{O}\left(\left(\frac{k \log n}{\epsilon}\right)^5\right)$.

Remarks: The above algorithm can be made robust to classification noise by repeating each measurement a fixed number of times and taking a majority vote. For instance, suppose each measurement is correct with probability $\frac{1}{2}+p$ and is flipped with probability $\frac{1}{2}-p$. Then repeating each measurement $\mathcal{O}\left(\frac{\log m}{p}\right)$ times, we can argue that a majority vote of measurements will give us the true measurements (with high probability). We can then use Algorithm 6 to recover $\boldsymbol{x}^*$.

Extension to Other Compressible Signals: Note that, the second stage of Algorithm 6 is essentially just a "standard" compressed sensing module, whose goal is used to recover $\boldsymbol{x}^*$ from "standard" (noisy) linear measurement of $\boldsymbol{x}^*$, i.e., $\widehat{\boldsymbol{z}} = A_1\boldsymbol{x}^* + \boldsymbol{\eta}$. Hence, we can modify our second stage to recover other compressible signals as well, by directly using the corresponding recovery method. Examples of such compressible signals include low-rank matrices, low-rank + sparse matrices, wavelet based sparse vectors etc.

Algorithm 7 Support Recovery based Approximate Recovery (S-Approx)

input A_1 and A_2, $\boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} = \text{sign}\begin{pmatrix} A_1\boldsymbol{x}^* \\ A_2\boldsymbol{x}^* \end{pmatrix}$

1: *Stage 1*: Run Expanders algorithm (Algorithm 3) with inputs $\boldsymbol{b}_1 = \text{sign}(A_1\boldsymbol{x}^*)$ and A_1 to output $\widehat{S}$.
2: $\widehat{\boldsymbol{x}} \leftarrow 0_{n\times 1}$
3: *Stage 2*: Run an LP solver for the following LP: find $\widehat{x}$ s.t. $b_2(i)\mathbf{a}_2^{(i)}(\widehat{S})\widehat{\boldsymbol{x}}(\widehat{S}) > 0$, $\forall 1 \leq i \leq m'$.

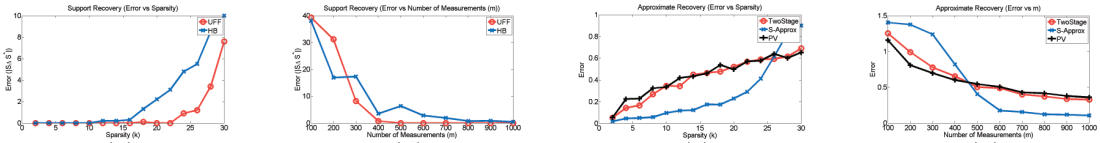
output $\widehat{\boldsymbol{x}}$

The framework by (Plan & Vershynin, 2012) can also recover a large class of compressible signals. However, as in the case of sparse vectors, their dependence on the error ϵ is ϵ^{-6} while ours is only ϵ^{-1}. Furthermore, (Plan & Vershynin, 2012) needs to compute "Gaussian width" for each of these class of functions; in contrast, we can directly use the existing results for these class of signals to provide measurement complexity.

Support Recovery based Approximate Recovery: In this section, we present another approach for approximate vector recovery that first recovers the support of $\boldsymbol{x}^*$ using our Expanders algorithm (Algorithm 3) and then solves the resulting low dimensional problem. That is, we choose the design matrix to be: $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ where A_1 is a design matrix based on expanders (as in Section 3.2) and A_2 is an iid standard Gaussian matrix. Using the measurements corresponding to A_1, we can first recover the support using Algorithm 3.

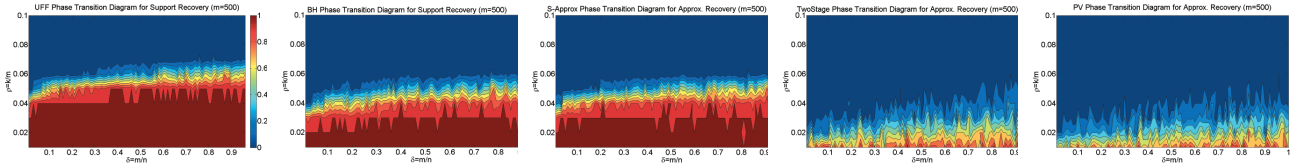
Once we have the support, we can solve an LP restricted to the support, to obtain $\widehat{x}$ that is consistent with the measurements. That is, $\text{sign}(A_2\widehat{\boldsymbol{x}}) = \text{sign}(A_2\boldsymbol{x}^*)$. Again using Theorem 10 (supplementary material) by (Jacques et al., 2011), we can conclude that $\left\| \frac{\widehat{\boldsymbol{x}}}{\|\widehat{\boldsymbol{x}}\|_2} - \boldsymbol{x}^* \right\|_2 < \epsilon$. See Algorithm 7 for a pseudo-code of our approach.

Now, the first step of support recovery requires $\mathcal{O}\left(k^3 \log \frac{n}{k}\right)$ measurements (Theorem 5) and $\mathcal{O}\left(nk \log \frac{n}{k}\right)$ time. The second step needs $\widetilde{\mathcal{O}}\left(\frac{k}{\epsilon}\right)$ measurements and $\widetilde{\mathcal{O}}\left(\frac{k^5}{\epsilon^5}\right)$ time for recovery. So overall, the algorithm needs $\widetilde{\mathcal{O}}\left(k^3 \log \frac{n}{k} + \frac{k}{\epsilon}\right)$ measurements and $\widetilde{\mathcal{O}}\left(nk \log \frac{n}{k} + \frac{k^5}{\epsilon^5}\right)$ time.



(a) (b) (c) (d)

Figure 1. **(a), (b)**: Error ($|S^* \Delta \widehat{S}|$) incurred by various support-recovery methods ($n = 3000$, varying k, m). **(c), (d)**: Error ($\left|\left| \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|_2} - \frac{\boldsymbol{x}^*}{\|\boldsymbol{x}^*\|_2} \right|\right|_2$) incurred by various approximate recovery methods with fixed $n = 3000$ but varying k, m. TwoStage (Algorithm 6) and PV incurs comparable error while S-approx (Algorithm 7) is significantly more accurate.



(a) (b) (c) (d) (e)

Figure 2. Phase transition diagrams for different methods when applied to support recovery (a, b) and approximate recovery (c, d, e). Each figure plots probability of success in 100 trials for different values of n and k. Red represents high probability of success (see plot (a) for color coding). Clearly, UFF recovers the support in a larger regime as compared to BH. For approximate recovery, S-approx performs better in a larger regime of the parameters as compared to both TwoStage and PV, while TwoStage slightly outperforms PV.

5. Experiments

In this section, we present empirical results for our algorithms for support recovery as well as approximate vector recovery. For support recovery, we evaluate our UFF algorithm (Algorithm 2) against the sketch based algorithm by (Haupt & Baraniuk, 2011) (**HB**). For support recovery, we evaluate our TwoStage algorithm and S-Approx algorithm against the algorithm by (Plan & Vershynin, 2012) (**PV**).

Support Recovery: For these experiments, we generate a k-sparse signal $\boldsymbol{x}^* \in \{0,1\}^n$ randomly and estimate its support using linear measurements proposed by each of the algorithms. We report the L_1 error in support estimation: $Error_{Support}(\widehat{S}, S^*) = |S^* \Delta \widehat{S}|$.

We first compare recovery properties of different methods using phase transition diagrams that are commonly used in the compressive sensing literature (see Figure 2 (a), (b)). For this, we fix the number of measurements ($m = 500$) while varying n and k. For each problem size (k, n, m) we generate 20 synthetic problems and plot probability of *exact* support recovery; probability values in Figure 2 are color coded with red representing high probability of recovery while blue represents low probability of recovery. Figure 2 (a), (b) show the phase transition diagrams of our UFF method (Algorithm 2) and the HB method, respectively. Note that UFF is able to recover the support for a significantly larger fraction of problems than HB.

Next, we study error incurred by different methods when the number of measurements are not enough for recovery. First, we fix $n = 3000, m = 500$ and vary k. Figure 1(a) compares the error incurred by our UFF algorithm against the HB algorithm. Clearly, our UFF based algorithm incurs smaller error than HB for large k. For example, for $k = 20$, UFF is able to recover the support exactly, while HB incurs around 20% error. Next, we fix $n = 3000, k = 20$ while varying m. Figure 1(b) shows that UFF is able to achieve exact recovery with around 400 measurements while HB requires around 800 measurements.

Approximate Recovery: Here, we generate k-sparse signals $\boldsymbol{x}^* \in \mathbb{R}^n$ where non-zeros are sampled using the standard k-variate Gaussian. We report error in recovery, i.e., $Error_{Approx} = \left|\left| \frac{\hat{\boldsymbol{x}}}{\|\hat{\boldsymbol{x}}\|_2} - \frac{\boldsymbol{x}^*}{\|\boldsymbol{x}^*\|_2} \right|\right|_2$.

Here again, we first plot phase transition diagrams for different methods. We fix $m = 500$ and vary n, k; for each problem size (m, n, k) we measure probability of success (out of 20 runs) where a method is considered to be successful for an instance if the error incurred is less than 0.3. Figures 2 (c), (d), (e) clearly show that S-Approx is significantly better than both TwoStage and PV; TwoStage is also marginally better than PV.

Next, we fix $n = 3000$ and $m = 500$, while varying k. Figure 1(c) compares TwoStage and S-approx algorithms with the PV algorithm. Here again, TwoStage and PV are comparable while S-approx incurs significantly less error for $k < 24$. For larger k, TwoStage and PV are significantly better than S-approx. Finally, we fix $n = 3000$ and $k = 20$, while varying m. Here again, for small number of measurements, S-approx incurs more error compared to TwoStage and PV. But, for larger number of measurements, it is significantly more accurate.

References

Ai, A., Lapanowski, A., Plan, Y., and Vershynin, R. One-bit compressed sensing with non-gaussian measurements. *arXiv preprint arXiv:1208.6279*, 2012.

Baraniuk, Richard G., Cevher, Volkan, Duarte, Marco F., and Hegde, Chinmay. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

Berinde, Radu and Indyk, Piotr. Sparse recovery using sparse random matrices, 2008.

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

Boufounos, Petros T. Greedy sparse signal reconstruction from sign measurements. In *Proceedings of the 43rd Asilomar conference on Signals, systems and computers*, pp. 1305–1309, 2009.

Candès, Emmanuel J. and Recht, Benjamin. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, December 2009.

Candès, Emmanuel J. and Tao, Terence. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

Davenport, M.A., Plan, Y., Berg, E., and Wootters, M. 1-bit matrix completion. *arXiv preprint arXiv:1209.3672*, 2012.

de Wolf, Ronald. Efficient data structures from union-free families of sets. `http://homepages.cwi.nl/~rdewolf/unionfree_datastruc.pdf`, 2012.

Duarte, M.F., Davenport, M.A., Takhar, D., Laska, J.N., Sun, Ting, Kelly, K.F., and Baraniuk, R.G. Single-pixel imaging via compressive sampling. *Signal Processing Magazine, IEEE*, 25(2):83 –91, march 2008. ISSN 1053-5888. doi: 10.1109/MSP.2007.914730.

Erdös, Péter L., Frankl, Peter, and Füredi, Zoltán. Families of finite sets in which no set is covered by the union of two others. *J. Comb. Theory, Ser. A*, 33(2):158–166, 1982.

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

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

Hsu, D., Kakade, S. M., Langford, J., and Zhang, T. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems*, 2009.

Jacques, Laurent, Laska, Jason N., Boufounos, Petros, and Baraniuk, Richard G. Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors. *CoRR*, abs/1104.3160, 2011.

Jafarpour, Sina, Xu, Weiyu, Hassibi, Babak, and Calderbank, A. Robert. Efficient and robust compressed sensing using optimized expander graphs. *IEEE Transactions on Information Theory*, 55(9): 4299–4308, 2009.

Laska, Jason N. and Baraniuk, Richard G. Regime change: Bit-depth versus measurement-rate in compressive sensing. *IEEE Transactions on Signal Processing*, 60(7):3496–3505, 2012.

Laska, Jason N., Wen, Zaiwen, Yin, Wotao, and Baraniuk, Richard G. Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements. *IEEE Transactions on Signal Processing*, 59 (11):5289–5301, 2011.

Negahban, Sahand, Ravikumar, Pradeep D., Wainwright, Martin J., and Yu, Bin. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. In *NIPS*, pp. 1348–1356, 2009.

Plan, Y. and Vershynin, R. One-bit compressed sensing by linear programming. *arXiv preprint arXiv:1109.4299*, 2011.

Plan, Yaniv and Vershynin, Roman. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *CoRR*, abs/1202.1212, 2012.

Tropp, Joel A. and Gilbert, Anna C. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

Wright, J., Ma, Yi, Mairal, J., Sapiro, G., Huang, T.S., and Yan, Shuicheng. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031 –1044, june 2010. ISSN 0018-9219. doi: 10.1109/JPROC.2010.2044470.