Approximate Sparse Recovery: Optimizing Time and Measurements

Anna C. Gilbert*
Department of Mathematics
University of Michigan
annacg@umich.edu

Yi Li†
Department of Electrical Engineering and Computer Science
University of Michigan
leeyi@umich.edu

Ely Porat
Department of Computer Science
Bar-Ilan University
porately@cs.biu.ac.il

Martin J. Strauss‡
Department of Mathematics and the Department of Electrical Engineering and Computer Science
University of Michigan
martinjs@umich.edu

ABSTRACT

A Euclidean *approximate sparse recovery* system consists of parameters k, N, an m-by-N *measurement matrix*, $\mathbf{\Phi}$, and a decoding algorithm, $\mathcal{D}$. Given a vector, $\mathbf{x}$, the system approximates $\mathbf{x}$ by $\widehat{\mathbf{x}} = \mathcal{D}(\mathbf{\Phi}\mathbf{x})$, which must satisfy $\|\widehat{\mathbf{x}} - \mathbf{x}\|_2 \leq C \|\mathbf{x} - \mathbf{x}_k\|_2$, where $\mathbf{x}_k$ denotes the optimal k-term approximation to $\mathbf{x}$. (The output $\widehat{\mathbf{x}}$ may have more than k terms). For each vector $\mathbf{x}$, the system must succeed with probability at least 3/4. Among the goals in designing such systems are minimizing the number m of measurements and the runtime of the decoding algorithm, $\mathcal{D}$.

In this paper, we give a system with $m = O(k \log(N/k))$ measurements—matching a lower bound, up to a constant factor—and decoding time $k \log^{O(1)} N$, matching a lower bound up to $\log(N)$ factors. We also consider the encode time (*i.e.*, the time to multiply $\mathbf{\Phi}$ by x), the time to update measurements (*i.e.*, the time to multiply $\mathbf{\Phi}$ by a 1-sparse x), and the robustness and stability of the algorithm (adding noise before and after the measurements). Our encode and update times are optimal up to $\log(k)$ factors. The columns of $\mathbf{\Phi}$ have at most $O(\log^2(k) \log(N/k))$ non-zeros, each of which can be found in constant time. Our full result, an FPRAS, is as follows. If $\mathbf{x} = \mathbf{x}_k + \nu_1$, where ν_1 and ν_2 (below) are arbitrary vectors (regarded as noise), then, setting $\widehat{\mathbf{x}} = \mathcal{D}(\Phi\mathbf{x} + \nu_2)$, and for properly normalized $\mathbf{\Phi}$, we get

$$\|\widehat{\mathbf{x}} - \mathbf{x}\|_2^2 \leq (1+\epsilon) \|\nu_1\|_2^2 + \epsilon \|\nu_2\|_2^2,$$

using $O((k/\epsilon) \log(N/k))$ measurements and $(k/\epsilon) \log^{O(1)}(N)$ time for decoding.

Categories and Subject Descriptors

G.4 [**Mathematical Software**]: Algorithm design and analysis

General Terms

Algorithms, Theory

Keywords

Approximation, embedding, sketching, sparse approximation, sublinear algorithms

1. INTRODUCTION

Tracking heavy hitters in high-volume, high-speed data streams [4], monitoring changes in data streams [6], designing pooling schemes for biological tests [11] (e.g., high throughput sequencing, testing for genetic markers), localizing sources in sensor networks [17, 16], and combinatorial pattern matching [5] are all quite different technological challenges, yet they can all be expressed in the same mathematical formulation. We have a signal $\mathbf{x}$ of length N that is sparse or highly compressible; i.e., it consists of k significant entries ("heavy hitters") which we denote by $\mathbf{x}_k$ while the rest of the entries are essentially negligible. We wish to acquire a small amount of information (commensurate with the sparsity) about this signal in a linear, non-adaptive fashion and then use that information to quickly recover the significant entries. In a data stream setting, our signal is the distribution of items seen, while in biological group testing, the signal is proportional to the binding affinity of each drug compound (or the expression level of a gene in a particular organism). We want to recover the identities and values only

*Supported in part by DARPA/ONR N66001-08-1-2065.
†Supported in part by NSF CCF 0743372.
‡Supported in part by NSF CCF 0743372 and DARPA/ONR N66001-08-1-2065.

STOC'10, June 5–8, 2010, Cambridge, Massachusetts, USA

of the heavy hitters which we denote by $\mathbf{x}_k$, as the rest of the signal is not of interest. Mathematically, we have a signal $\mathbf{x}$ and an m-by-N measurement matrix $\mathbf{\Phi}$ with which we acquire measurements $\boldsymbol{y} = \mathbf{\Phi}\mathbf{x}$, and, from these measurements $\boldsymbol{y}$, we wish to recover $\widehat{\mathbf{x}}$, with $O(k)$ entries, such that

$$\|\mathbf{x} - \widehat{\mathbf{x}}\|_2 \leq C \|\mathbf{x} - \mathbf{x}_k\|_2 .$$

Our goal, which we achieve up to constant or log factors in the various criteria, is to design the measurement matrix $\mathbf{\Phi}$ and the decoding algorithm in an optimal fashion: (i) we minimize the number $m = O\big(k \log \frac{N}{k}\big)$ of measurements, (ii) the decoding algorithm runs in *sublinear* time $O\big(k \log \frac{N}{k}\big)$, and (iii) the encoding and update times are optimal $O\big(N \log \frac{N}{k}\big)$ and $O\big(k \log \frac{N}{k}\big)$, respectively. In order to achieve this, our algorithm is randomized; i.e., we specify a distribution on the measurement matrix $\mathbf{\Phi}$ and we guarantee that, for each signal, the algorithm recovers a good approximation with high probability over the choice of matrix.

In the above applications, it is important both to take as few measurements as possible and to recover the heavy hitters extremely efficiently. Measurements correspond to physical resources (e.g., memory in data stream monitoring devices, number of screens in biological applications) and reducing the number of necessary measurements is critical these problems. In addition, these applications require efficient recovery of the heavy hitters—we test many biological compounds at once, we want to quickly identify the positions of entities in a sensor network, and we cannot afford to spend computation time proportional to the size of the distribution in a data stream application.

Our result is robust to corruption of the measurements by an arbitrary noise vector ν_2, which is an important feature for such applications as high throughput screening and other physical measurement systems. (It is less critical for digital measurement systems that monitor data streams in which measurement corruption is less likely.) Our result is an FPRAS in both pre- and post- measurement noise.

1.1 Related Work

Do Ba et al. [2] give a lower bound of $\Omega(k \log(N/k))$ for the number of measurements for sparse recovery, in a model that is related to ours but different in some important respects. There are polynomial time algorithms [14, 3, 13] meeting this lower bound, both with high probability for each signal and the stronger setting, with high probability for all signals.[1] Previous sublinear time algorithms, whether in the "for each" model [4, 8] or in the "for all" model [12], however, used several additional factors of $\log(N)$ measurements. We summarize some previous algorithms in the "for each" signal model in Figure 1. The column sparsity denotes how many 1s there are per column of the measurement matrix and determines both the decoding and measurement update time and, for readability, we suppress $O(\cdot)$. The approximation error signifies the metric we use to evaluate the output; $\ell_j \leq C\ell_k(+\ell_m)$ is shorthand for $\|\widehat{\mathbf{x}} - \mathbf{x}\|_j \leq C \|\mathbf{x}_k - \mathbf{x}\|_k (+C \|\nu_2\|_m)$. (Some previous results that did not directly claim stability with respect to ν_2 can be modified easily to accomodate non-zero ν_2.)

[1] albeit with different error guarantees and different column sparsity depending on the error metric.

1.2 Our Result

We give a sublinear time recovery algorithm and a distribution over normalized measurement matrices that meet this lower bound (up to constant factors) in terms of the number of measurements and are within $\log(k)$ factors of optimal in the running time and the sparsity of the measurement matrix.

THEOREM 1. *There is an algorithm and distribution on matrices $\mathbf{\Phi}$ satisfying $\max_{\mathbf{x}} \mathbb{E}[\|\mathbf{\Phi}\mathbf{x}\|_2 / \|\mathbf{x}\|_2] = 1$ such that, given $\mathbf{\Phi}\mathbf{x}$, the parameters, and a concise description of $\mathbf{\Phi}$, the algorithm returns $\widehat{x}$ with approximation error $\|\mathbf{x} - \widehat{\mathbf{x}}\|_2^2 \leq (1+\epsilon)\|\nu_1\|_2^2 + \epsilon\|\nu_2\|_2^2$ with probability $\frac{3}{4}$. The algorithm runs in time $\frac{k}{\epsilon}\log^{O(1)} N$ and $\mathbf{\Phi}$ has $O\big(\frac{k}{\epsilon}\log\frac{N}{k}\big)$ rows. In expectation, there are $O(\log^2(k)\log(N/k))$ non-zeros in each column of $\mathbf{\Phi}$.*

The approximation $\widehat{\mathbf{x}}$ may have more than k terms. From previous work, it is known that, if

$$\|\widehat{\boldsymbol{z}} - \mathbf{x}\|_2^2 \leq (1+\epsilon^2)\|\mathbf{x}_k - \mathbf{x}\|_2^2 + \epsilon^2 \|\nu_2\|_2^2 ,$$

then the truncation $\widehat{\boldsymbol{z}}_k$ of $\widehat{\boldsymbol{z}}$ to k terms satisfies

$$\|\widehat{\boldsymbol{z}}_k - \mathbf{x}\|_2^2 \leq (1+\Theta(\epsilon))\|\mathbf{x}_k - \mathbf{x}\|_2^2 + \epsilon\|\nu_2\|_2^2 .$$

So an approximation with exactly k terms is possible, but with cost $1/\epsilon^2$ versus $1/\epsilon$ for the general case.

1.3 Our Technical Contributions

Previous sublinear algorithms begin with the observation that if a signal consists of a single heavy hitter, then the trivial encoding of the positions 1 through N with $\log(N)$ bits, referred to as a bit tester, can identify the position of the heavy hitter, as in the following.

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 7 \\ 0 \\ 7 \\ 0 \end{pmatrix} .$$

The second observation in previous work is that a number of hash or Bernoulli functions drawn at random from a hash family are sufficient to isolate enough of the heavy hitters, which can then be identified by the bit tester. Depending on the type of error metric desired, the hashing matrix is pre-multiplied by random ± 1 vectors (for the ℓ_2 metric) in order to estimate the signal values. In this case, the measurements are referred to as the COUNT SKETCH in the data stream literature [4] and, without the premultiplication, the measurements are referred to as COUNT MEDIAN [7, 8] and give $\ell_1 \leq C\ell_1$ error guarantees. In addition, the sublinear algorithms are typically greedy, iterative algorithms that recover portions of the heavy hitters with each iteration or that recover portions of the ℓ_2 (or ℓ_1) energy of the residual signal.

We build upon the COUNT SKETCH design but incorporate the following algorithmic innovations to ensure an optimal number of measurements:

- With a random assignment of N signal positions to $O(k)$ subsignals, we need to encode only $O(N/k)$ po-

Paper	No. Measurements	Column sparsity/ Update time	Decode time	Approx. error
[9, 3]	$k\log(N/k)$	$k\log(N/k)$	$\geq N$	$\ell_2 \leq (1/\sqrt{k})\ell_1 + \ell_2$
[4, 8]	$k\log^c N$	$\log^c N$	$k\log^c N$	$\ell_2 \leq C\ell_2$
[7]	$k\log^c N$	$\log^c N$	$k\log^c N$	$\ell_1 \leq C\ell_1$
This paper	$k\log(N/k)$	$\log^c N$	$k\log^c N$	$\ell_2 \leq C\ell_2 + \ell_2$

Table 1: Summary of the best previous results and the result obtained in this paper.

sitions, rather than N as in the previous approaches. Thus we can reduce the domain size which we encode.

- We use a good error-correcting code (rather than the trivial identity code of the bit tester).

- Our algorithm is an iterative algorithm but maintains a *compound* invariant: in our algorithm, the number of un-discovered heavy hitters decreases at each iteration while, simultaneously, the required error tolerance and failure probability become more stringent. Because there are fewer heavy hitters to find at each stage, we can use more measurements to meet more stringent guarantees.

We believe we are the first to consider a "for each" algorithm with post-measurement noise, ν_2. As we discuss below, we need to give a new definition of the appropriate metric under which to normalize $\mathbf{\Phi}$.

In Section 2 we detail the matrix algebra we use to describe the measurement matrix distribution which we cover in Section 3, along with the decoding algorithm. In Section 4, we analyze the foregoing recovery system.

2. PRELIMINARIES

2.1 Vectors

Let $\mathbf{x}$ denote a vector of length N. For each $k \leq N$, let $\mathbf{x}_k$ denote either the usual k'th component of $\mathbf{x}$ or the signal of length N consisting of the k largest-magnitude terms in $\mathbf{x}$; it will be clear from context. The signal $\mathbf{x}_k$ is the best k-term representation of $\mathbf{x}$. The energy of a signal $\mathbf{x}$ is $\|\mathbf{x}\|_2^2 = \sum_{i=1}^N |\mathbf{x}_i|^2$.

2.2 Matrices

In order to construct the overall measurement matrix, we form a number of different types of combinations of constituent matrices and to facilitate our description, we summarize our matrix operations in Table 2. The matrices that result from all of our matrix operations have N columns and, with the exception of the semi-direct product of two matrices $\ltimes_{\mathrm{r}}$, all operations are performed on matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with N columns.

Next we give a full description of the matrix algebra defined in Table 2.

- **Row direct sum.** The row direct sum $\boldsymbol{A} \oplus_{\mathrm{r}} \boldsymbol{B}$ is a matrix with N columns that is the vertical concatenation of $\boldsymbol{A}$ and $\boldsymbol{B}$.

- **Element-wise product.** If $\boldsymbol{A}$ and $\boldsymbol{B}$ are both $r \times N$ matrices, then $\boldsymbol{A} \odot \boldsymbol{B}$ is also an $r \times N$ matrix whose (i,j) entry is given by the product of the (i,j) entries in $\boldsymbol{A}$ and $\boldsymbol{B}$.

- **Semi-direct product.** Suppose $\boldsymbol{A}$ is a matrix of r_1 rows (and N columns) in which each row has exactly h non-zeros and $\boldsymbol{B}$ is a matrix of r_2 rows and h columns. Then $\boldsymbol{B} \ltimes_{\mathrm{r}} \boldsymbol{A}$ is the matrix with $r_1 r_2$ rows, in which each non-zero entry a of $\boldsymbol{A}$ is replaced by a times the j'th column of $\boldsymbol{B}$, where a is the j'th non-zero in its row.

  This matrix construction has the following interpretation. Consider $(\boldsymbol{B} \ltimes_{\mathrm{r}} \boldsymbol{A})\mathbf{x}$ where $\boldsymbol{A}$ consists of a single row, $\boldsymbol{\rho}$, with h non-zeros and $\mathbf{x}$ is a vector of length N. Let $\boldsymbol{y} = \boldsymbol{\rho} \odot \mathbf{x}$ be the element-wise product of ρ and $\mathbf{x}$. If $\boldsymbol{\rho}$ is 0/1-valued, $\boldsymbol{y}$ picks out a subset of $\mathbf{x}$. We then remove all the positions in $\boldsymbol{y}$ corresponding to zeros in $\boldsymbol{\rho}$, leaving a vector $\boldsymbol{y'}$ of length h. Finally, $(\boldsymbol{B} \ltimes_{\mathrm{r}} \boldsymbol{A})\mathbf{x}$ is simply the matrix-vector product $\boldsymbol{B}\boldsymbol{y'}$, which, in turn, can be interpreted as selecting subsets of $\boldsymbol{y}$, and summing them up. Note that we can modify this definition when $\boldsymbol{A}$ has fewer than h non-zeros per row in a straightforward fashion.

3. SPARSE RECOVERY SYSTEM

In this section, we specify the measurement matrix and detail the decoding algorithm.

3.1 Measurement matrix

The overall measurement matrix, $\mathbf{\Phi}$, is multi-layered. At the highest level, $\mathbf{\Phi}$ consists of a random permutation matrix $\boldsymbol{P}$ left-multiplying the row direct sum of $O(\log(k))$ summands, $\mathbf{\Phi}^{(j)}$, each of which is used in a separate iteration of the decoding algorithm. Each summand $\mathbf{\Phi}^{(j)}$ is the row direct sum of two separate matrices, an *identification* matrix, $\boldsymbol{D}^{(j)}$, and an *estimation* matrix, $\boldsymbol{E}^{(j)}$.

$$\mathbf{\Phi} = \boldsymbol{P} \begin{bmatrix} \mathbf{\Phi}^{(1)} \\ \hline \mathbf{\Phi}^{(2)} \\ \hline \vdots \\ \hline \mathbf{\Phi}^{(\log(k))} \end{bmatrix} \qquad \text{where } \mathbf{\Phi}^{(j)} = \boldsymbol{E}^{(j)} \oplus_{\mathrm{r}} \boldsymbol{D}^{(j)}.$$

In iteration j, the identification matrix $\boldsymbol{D}^{(j)}$ consists of the row direct sum of $O(j)$ matrices, all chosen independently from the same distribution. We construct that distribution, $\frac{2^{-\Theta(j)}}{\sqrt{\log N/k}}(\boldsymbol{C}^{(j)} \ltimes_{\mathrm{r}} \boldsymbol{H}^{(j)}) \odot \boldsymbol{S}^{(j)}$, as follows. (The values of some parameters will be given later.)

- For $j = 1, 2, \ldots, \log(k)$, the matrix $\boldsymbol{H}^{(j)}$ is a hashing matrix with dimensions kc^j-by-N, where c in the range $1/2 < c < 1$ will be specified later. Each entry is 1 with probability $\Theta\left(1/(kc^j)\right)$ and zero otherwise. Each row is a pairwise independent family and the set of row seeds is pairwise dependent.

- The matrix $\boldsymbol{C}^{(j)}$ is an encoding of positions by an error-correcting code with constant rate and relative

operator	name	input	output dimensions and construction
$\oplus_r$	row direct sum	$\boldsymbol{A}$: $r_1 \times N$ $\boldsymbol{B}$: $r_2 \times N$	$\boldsymbol{M}$: $(r_1 + r_2) \times N$ $\boldsymbol{M}_{i,j} = \begin{cases} \boldsymbol{A}_{i,j}, & 1 \le i \le r_1 \\ \boldsymbol{B}_{i-r_1,j}, & 1 + r_1 \le i \le r_2 \end{cases}$
$\odot$	element-wise product	$\boldsymbol{A}$: $r \times N$ $\boldsymbol{B}$: $r \times N$	$\boldsymbol{M}$: $r \times N$ $\boldsymbol{M}_{i,j} = \boldsymbol{A}_{i,j}\boldsymbol{B}_{i,j}$
$\ltimes_{\mathrm{r}}$	semi-direct product	$\boldsymbol{A}$: $r_1 \times N$ $\boldsymbol{B}$: $r_2 \times h$	$\boldsymbol{M}$: $(r_1 r_2) \times N$ $\boldsymbol{M}_{i+(k-1)r_2,\ell} = \begin{cases} 0, & \boldsymbol{A}_{k,\ell} = 0 \\ \boldsymbol{A}_{k,\ell}\boldsymbol{B}_{i,j}, & \boldsymbol{A}_{k,\ell} = \ j\text{th nonzero in row } \ell \end{cases}$

Table 2: Matrix algebra used in constructing an overall measurement matrix. The last column contains both the output dimensions of the matrix operation and its construction formula.

distance, together with several 1's. That is, fix an error-correcting code and encoding and decoding algorithms that encode messages of $\Theta(\log\log N)$ bits into longer codewords, also of length $\Theta(\log\log N)$, and can correct a constant fraction of errors. The i'th column of $\boldsymbol{C}^{(j)}$ is the direct sum of $\Theta(\log\log N)$ copies of 1 with the direct sum of $E(i_1)$, $E(i_2)$, ..., where $i_1, i_2, \ldots$ are blocks of $O(\log\log N)$ bits each, whose concatenation is the binary expansion of i; $E(\cdot)$ is the encoding function for the error-correcting code. The number of columns in $\boldsymbol{C}^{(j)}$ is the same as the maximum number of non-zeros in $\boldsymbol{H}^{(j)}$, which is approximately the expected number, $\Theta\left(c^j N/k\right)$, where $c < 1$. The number of rows in $\boldsymbol{C}^{(j)}$ is the logarithm of the number of columns, since the process of breaking the binary expansion of index i into blocks has rate 1 and encoding by $E(\cdot)$ has constant rate.

Note that error correcting encoding often is accomplished by a matrix-vector product, but we are *not* encoding a linear error-correcting code by the usual generator matrix process. Rather, our matrix explicitly lists all the codewords. The code may be non-linear.

- The matrix $\boldsymbol{S}^{(j)}$ is a pseudorandom sign-flip matrix. Each row is a pairwise independent family of uniform ± 1-valued random variables. The sequence of seeds for the rows is a fully independent family. The size of $\boldsymbol{S}^{(j)}$ matches the size of $\boldsymbol{C}^{(j)} \ltimes_{\mathrm{r}} \boldsymbol{H}^{(j)}$.

Below, to achieve our claimed runtime, we will construct $\boldsymbol{C}^{(j)}$ and $\boldsymbol{H}^{(j)}$ together. See Figure 1.

The identification matrix at iteration j is of the form

$$\boldsymbol{D}^{(j)} = \frac{2^{-\Theta(j)}}{\sqrt{\log N/k}} \begin{bmatrix} \left[2^{-\Theta(j)}(\boldsymbol{C}^{(j)} \ltimes_{\mathrm{r}} \boldsymbol{H}^{(j)}) \odot \boldsymbol{S}^{(j)}\right]_1 \\ \vdots \\ \left[(\boldsymbol{C}^{(j)} \ltimes_{\mathrm{r}} \boldsymbol{H}^{(j)}) \odot \boldsymbol{S}^{(j)}\right]_{O(j)} \end{bmatrix}.$$

In iteration j, the estimation matrix $\boldsymbol{E}^{(j)}$ consists of the direct sum of $O(j + \log(1/\epsilon))$ matrices, all chosen independently from the same distribution, $\frac{2^{-\Theta(j)}}{\sqrt{\log N/k}} \boldsymbol{H'}^{(j)} \odot \boldsymbol{S'}^{(j)}$, so that the estimation matrix at iteration j is of the form

$$\boldsymbol{E}^{(j)} = \frac{2^{-\Theta(j)}}{\sqrt{\log N/k}} \begin{bmatrix} \left[\boldsymbol{H'}^{(j)} \odot \boldsymbol{S'}^{(j)}\right]_1 \\ \vdots \\ \left[\boldsymbol{H'}^{(j)} \odot \boldsymbol{S'}^{(j)}\right]_{O(j)} \end{bmatrix}.$$

The construction of the distribution is similar to that of the identification matrix, but omits the error-correcting code and uses different constant factors for the number of rows, etc., compared with the analogs in the identification matrix.

- The matrix $\boldsymbol{H'}^{(j)}$ is a hashing with dimensions $O(kc^j)$-by-N, for appropriate c, $1/2 < c < 1$. Each entry is 1 with probability $\Theta\left(1/(kc^j)\right)$ and zero otherwise. Each row is a pairwise independent family and the set of seeds is fully independent.

- The matrix $\boldsymbol{S'}^{(j)}$ is a pseudorandom sign-flip matrix of the same dimension as $\boldsymbol{H'}^{(j)}$. Each row of $\boldsymbol{S'}^{(j)}$ is a pairwise independent family of uniform ± 1-valued random variables. The sequence of seeds for the rows is fully independent.

3.2 Measurements

The overall form of the measurements mirrors the structure of the measurement matrices. We do not, however, use all of the measurements in the same fashion. Upon receiving $\boldsymbol{\Phi}\mathbf{x} + \nu_2$, the algorithm first applies the permutation $\boldsymbol{P}^{-1}$. In iteration j of the algorithm, we use the measurements $\boldsymbol{y}^{(j)} = \boldsymbol{\Phi}^{(j)}\mathbf{x}$. As the matrix $\boldsymbol{\Phi}^{(j)} = \boldsymbol{E}^{(j)} \oplus_{\mathrm{r}} \boldsymbol{D}^{(j)}$, we have a portion of the measurements $\boldsymbol{w}^{(j)} = \boldsymbol{D}^{(j)}\mathbf{x}$ that we use for identification and a portion $\boldsymbol{z}^{(j)} = \boldsymbol{E}^{(j)}\mathbf{x}$ that we use for estimation. The $\boldsymbol{w}^{(j)}$ portion is further decomposed into measurements $[\boldsymbol{v}^{(j)}, \boldsymbol{u}^{(j)}]$ corresponding to the run of $O(\log\log N)$ 1's in $\boldsymbol{C}^{(j)}$ and measurements corresponding to each of the blocks in the error-correcting code. There are $O(j)$ i.i.d. repetitions in the identification part and $O(j + \log 1/\epsilon)$ repetitions in the estimation part.

3.3 Decoding

The decoding algorithm is shown in Figure 3.

4. ANALYSIS

In this section we analyze the decoding algorithm for correctness and efficiency.

4.1 Overview

The overall structure of our algorithm is greedy, similar to other algorithms in the literature. At each iteration, the algorithm recovers some of the signal, but introduce errors both through many coefficient estimates that are approximately but not perfectly correct, and also through a small number of terms that can be arbitrarily bad. The result is called a *residual signal*.

RECOVER($\boldsymbol{\Phi}, \boldsymbol{y}$)

Output: $\widehat{x}$ = approximate representation of x

$\boldsymbol{y} = \boldsymbol{P}^{-1}\boldsymbol{y}$
$\boldsymbol{a}^{(0)} = 0$
For $j = 0$ to $O(\log k)$ {
    $\boldsymbol{y} = \boldsymbol{y} - \boldsymbol{P}^{-1}\boldsymbol{\Phi}\boldsymbol{a}^{(j)}$
    split $\boldsymbol{y}^{(j)} = \boldsymbol{w}^{(j)} \oplus_{\mathrm{r}} \boldsymbol{z}^{(j)}$
    $\Lambda =$ IDENTIFY($\boldsymbol{D}^{(j)}, \boldsymbol{w}^{(j)}$)
    $\boldsymbol{b}^{(j)} =$ ESTIMATE($\boldsymbol{E}^{(j)}, \boldsymbol{z}^{(j)}, \Lambda$)
    $\boldsymbol{a}^{(j+1)} = \boldsymbol{a}^{(j)} + \boldsymbol{b}^{(j)}$
}
$\widehat{x} = \boldsymbol{a}^{(j)}$

IDENTIFY($\boldsymbol{D}^{(j)}, \boldsymbol{w}^{(j)}$)

Output: Λ = list of positions

$\Lambda = \emptyset$
Divide $\boldsymbol{w}^{(j)}$ into sections $[\boldsymbol{v}, \boldsymbol{u}]$ of size $O(\log(c^j(N/k)))$
For each section {
    $u = \mathrm{median}(|\boldsymbol{v}_\ell|)$
    For each ℓ // threshold measurements
        $\boldsymbol{u}_\ell = \Theta(\boldsymbol{u}_\ell - u/2)$ // $\Theta(u) = 1$ if $u > 0$, $\Theta(u) = 0$ otherwise
    Divide $\boldsymbol{u}$ into blocks b_i of size $O(\log \log N)$
    For each b_i
        β_i = DECODE(b_i) // using error-correcting code
    $\lambda =$INTEGER($\beta_1, \beta_2, \ldots$) // integer represented by bits $\beta_1, \beta_2, \ldots$
    $\lambda =$CONVERT(λ) // convert bucket index to signal index
    $\Lambda = \Lambda \cup \{\lambda\}$
}

ESTIMATE($\boldsymbol{E}^{(j)}, \boldsymbol{z}^{(j)}, \Lambda$)

Output: $\boldsymbol{b}$ = vector of positions and values

$\boldsymbol{b} = \emptyset$
For each $\lambda \in \Lambda$
    $\boldsymbol{b}_\lambda = \mathrm{median}_{\ell \text{ s.t.} \boldsymbol{H}^{(j)}_{\ell,\lambda}=1}(\boldsymbol{z}^{(j)}_\ell \boldsymbol{S}^{(j)}_{\ell,\lambda})$
For each $\lambda \in \Lambda$
    If $|\boldsymbol{b}_\lambda|$ is not among top $\Theta(k/2^j)$
        $\boldsymbol{b}_\lambda = 0$

Table 3: Pseudocode for the overall decoding algorithm.

The measurement and runtime costs of first iteration dominates the combined cost of all the others. In it, we reduce a bound on the number of heavy hitters to recover from k to $k/2$, while increasing the noise energy from 1 to $1+\epsilon/4$, using $O((k/\epsilon)\log(N/k))$ measurements. In subsequent iterations, the number of heavy hitters is reduced to $k/2^j$, which reduces the leading cost factor from k/ϵ to $2^{-j}k/\epsilon$. This gives the algorithm 2^j times more resources. In particular, the algorithm can tighten the approximation constant from $1+\epsilon/4$ to $1+(\epsilon/4)c^j$, for appropriate c in the range $1/2 < c < 1$, at cost factor $(1/c)^j < 2^j$, which is more than paid for by the $2^{-j} < 1$ savings in the leading factor. Similarly, the algorithm can simultaneously afford to have a smaller failure probability at iteration j. With the tightened approximation constant, the algorithm can tolerate additional ν_2 noise in later iterations, which, as we show below, saves resources.

To prove our result formally, we proceed as follows.

- We state a Loop Invariant maintained by our algorithm.
- We give a lemma (the Loop-Invariant Maintenance Lemma, or LIM lemma) that can be used to characterize a single iteration of the algorithm: how many measurements are used, how many non-zeros there are in each coluimn of the measurement matrix, the runtime, and the properties of the residual.

  To prove the LIM lemma, we proceed as follows.
- In Claim 1, we explain, structurally, how the conclusions of the lemma are met—what are the sources of errors, etc. We then examine the three subroutines in the algorithm—isolating heavy hitters, identifying them, and estimating coefficients.
- We discuss normalization of $\mathbf{\Phi}$ and show that it is, indeed, normalized.
- We conclude by analyzing the correctness and efficiency of the overall algorithm, using our results about each iteration.

4.2 Correctness

Formally, we maintain the following invariant.

INVARIANT 2 (LOOP INVARIANT). *At the beginning of iteration j, the residual signal has the form $\boldsymbol{r}^{(j)} = \sigma^{(j)} + \nu_1^{(j)}$ with*

$$\left\|\sigma^{(j)}\right\|_0 \leq \frac{k}{2^j} \text{ and } \left\|\nu_1^{(j)}\right\|_2^2 \leq \left(1+\epsilon\left(1-\left(\frac{3}{4}\right)^j\right)\right)\left\|\nu_1^{(j)}\right\|_2^2$$

except with probability $\frac{1}{4}(1-(\frac{1}{2})^j)$, where $\|\cdot\|_0$ is the number of non-zero entries. Furthermore, the algorithm has computed (the sparse partial representation) $\widehat{\mathbf{x}}^{(j)} = \mathbf{x} - \boldsymbol{r}^{(j)}$.

Clearly, the invariant holds at the start and maintaining the invariant is sufficient to prove the overall result. In order to show that the algorithm maintains the loop invariant, we demonstrate the following lemma, which, after proper instantiation of the lemma's variables, can be used to show the invariant is maintained.

4.2.1 Loop Invariant Maintenance

LEMMA 3 (LOOP INVARIANT MAINTENANCE). *Fix numerical parameters N, ℓ, δ, and η, with $\delta > 0$, and $\eta > 1/N$. Let $\boldsymbol{a}$ be a vector of length N that can be written as $\boldsymbol{a} = \sigma + \nu_1$, with $\|\sigma\|_0 \leq \ell$. Let $\mathbf{\Phi}$ be of the form of $O(\log 1/\delta)$ repetitions of $(\boldsymbol{C} \ltimes_r \boldsymbol{H}) \odot \boldsymbol{S}$ in row direct sum with $O(\log 1/(\delta\eta))$ repetitions of $\boldsymbol{H'} \odot \boldsymbol{S'}$ described in Section 3.1, where $\boldsymbol{H}$ and $\boldsymbol{H'}$ have $O(\ell/\eta)$ rows. Let ν_2 be a random noise vector, where each item has the same expected square magnitude.*

Then, except with probability δ, given $\mathbf{\Phi}, \boldsymbol{y} = \mathbf{\Phi}\boldsymbol{a} + \nu_2$, and appropriate parameters, the inner loop of the RECOVER *algorithm in Figure 3 recovers $\boldsymbol{b}$ that can be written as $\boldsymbol{b} = \sigma' + \nu_1'$, with $\|\sigma'\|_0 \leq \ell/2$ and $\|\nu_1'\|_2^2 \leq (1+\eta)\|\nu_1\|_2^2 + \frac{\eta}{\gamma}\|\nu_2\|_2^2$, where γ is the common expected number of non-zeros in each column of $\mathbf{\Phi}$. Furthermore,*

- *The number of rows in $\mathbf{\Phi}$ is $O(\ell/\eta)\log(N/\ell)\log(1/\delta)$.*
- *The computation time is*

$$(\ell/\eta)\log^{O(1)}(N/(\ell\delta\eta)).$$

- *The expected number γ of non-zeros in each column of $\mathbf{\Phi}$ is $O((\log N/\ell)\log(1/\delta))$.*

PROOF. Much of the algorithm and analysis is similar to previous work [4], so we sketch the proof, focusing on changes versus previous work. We first address the case $\nu_2 = 0$.

Recall that $\mathbf{\Phi}$ works by giving each element of the signal a random sign flip, hashing each item pairwise independently at random to each measurement, and encoding each index by an error-correcting code. We have:

CLAIM 1. *Except with probability $\delta/3$,*

- *The vector $\boldsymbol{b}$ contains all but at most $\ell/4$ terms of σ, with "good" estimates.*
- *The vector $\boldsymbol{b}$ contains at most $\ell/4$ terms with "bad" estimates, i.e., with square error greater than proportional to η/ℓ.*
- *The total sum square error over all "good" estimates is at most $\eta/4$.*

PROOF. To simplify notation, let T be the set of terms of $\boldsymbol{a}$ that are both among the top ℓ and have energy at least $\frac{\eta}{8\ell}\|\nu_1\|_2^2$. We know that $|T| \leq O(\ell)$. The proof proceeds in three steps.

Step 1. Isolate heavy hitters with little noise. Consider the action of a hashing and sign-flip matrix $\boldsymbol{H} \odot \boldsymbol{S}$ with $O(\ell/\eta)$ rows. From previous work [4, 1], it follows that, if constant factors parametrizing the matrices are chosen properly,

LEMMA 4. *For each $t \in T$, the following holds with probability $1 - O(\delta\eta)$:*

- *The term t is hashed by at least one row ρ in $\boldsymbol{H}$.*
- *There are $O(\eta N/\ell)$ total positions (out of N) hashed by ρ.*
- *The dot product $(\rho \odot \boldsymbol{S})\boldsymbol{a}$ is $\boldsymbol{S}_t\boldsymbol{a}_t \pm O\left(\sqrt{\frac{\eta}{\ell}}\|\nu_1\|_2\right)$.*

PROOF. (Sketch) For intuition, note that the estimator $\boldsymbol{S}_t(\rho \odot \boldsymbol{S})\boldsymbol{a}$ is a random variable with mean $\boldsymbol{a}_t$ and variance $\|\nu_1\|_2^2$. Then the claims in the Lemma assert that the expected behavior happens, up to constant factors, with probability $\Omega(1)$.

In the favorable case, each row of $\boldsymbol{H}$ is hashed one term of T that dominates the other $\eta N/\ell$ terms hashed to that bucket. □

Step 2. Identify heavy hitters with little noise. Next, we show how to identify t. Since there are $\eta N/\ell$ positions hashed by $\boldsymbol{H}$, we need to learn the $O(\log(\eta N/k))$ bits describing t in this context. Previous sublinear algorithms [8, 12] used a trivial error correcting code, in which the t'th column was simply the binary expansion of t in direct sum with a single 1, for the matrix $\boldsymbol{C}$ in semi-direct product with $\boldsymbol{H}$. Thus, if the signal $\mathbf{x}$ consists of $\mathbf{x}_t$ in the t'th position and zeros elsewhere, the vector $(\boldsymbol{C} \ltimes_r \boldsymbol{H})\mathbf{x}$ would include $\mathbf{x}_t$ and $\mathbf{x}_t$ times the binary expansion of t (the latter interpreted as a string of 0's and 1's as real numbers). These algorithms require strict control on the failure probability of each measurement in order to use such a trivial encoding. In our case, each measurement succeeds only with probability $\Omega(1)$ and, generally, fails with probability $\Omega(1)$. So we need to use a more powerful error correcting code and a more reliable estimate of $|\mathbf{x}_t|$.

To get an estimate of $|\mathbf{x}_t|$ that is reliable, we use the $b = \Theta(\log \log N)$-parallel repetition code of all 1s. That is, we get b independent measurements of $|\mathbf{x}_t|$ and we decode by taking the median. Let p denote the success probability of each individual measurement. Then we expect the fraction p to be approximately correct estimates of $|\mathbf{x}_t|$, we achieve close to the expectation, and we can arrange that $p > 1/2$. It follows that the median is approximately correct. We use this value to threshold the subsequent measurements (i.e., the bits in the encoding) to 0/1 values.

Now, let us consider these bit estimates. In a single error-correcting code block of $b = \Theta(\log \log N)$ measurements, we will get close to the expected number, bp, of successful measurements, except with probability $1/\log(N)$, using the Chernoff bound. In the favorable case, we get a number of failures less than the (properly chosen) distance of the error-correcting code and we can recover the block using standard nearest-neighbor decoding. The number of error-correcting code blocks associated with t is $O(\log(\eta N/k)/\log \log N) \leq O(\log N)$, so we can take a union bound over all blocks and conclude that we recover t with probability $\Omega(1)$. Because the algorithm takes $O(\log 1/\delta)$ parallel independent repetitions, we guarantee that the failure probability is δ for each $t \in T$ and we expect $\delta|T| = O(\delta\ell)$ failures, overall. The probability of getting more than $\ell/4$ failures is at most $O(\delta)$.

Step 3. Estimate heavy hitters.

Many of the details in this step are similar to those in Lemma 4 (as well as to previous work as the function ESTIMATE is essentially the same as COUNT SKETCH), so we give only a brief summary.

The error-correcting code is not necessary. As above, random sign flips and hashing into $O(\ell/\eta)$ buckets suffices to isolate a term t so that the remaining terms hashed to t's bucket have expected energy $O(\eta/\ell)$ and realize energy $O(\eta/\ell)$ with constant probability. Another factor $\log 1/(\delta\eta)$ repetitions suffices to make the failure probability $\delta\eta$, so that, except with probability δ, we have $O(\eta|\Lambda|) = O(\ell)$ failures overall among the $|\Lambda| = \Theta(\ell/\eta)$ candidates whose coefficients we estimate.

This concludes the proof of the Claim.

Number of measurements.

We now consider the number of measurements in the matrix. The hashing matrix $\boldsymbol{H}$ contributes $O(\ell/\eta)$ rows. The constant-rate error-correcting code matrix $\boldsymbol{C}$ contributes an additional factor of $O(\log \eta N/\ell)$, to identify one index out of $\eta N/\ell$. The $O(\log 1/\delta)$ repetitions contribute that additional factor, to drive down the overall failure probability of identification from $1 - \Omega(1)$ to δ. The $\boldsymbol{S}$ matrix does not contribute to the number of rows. This gives a product of

$$O((\ell/\eta)(\log(\eta N/\ell) \log 1/\delta))$$

for identification. Similarly, for estimation, we have $O(\ell/\eta)$ rows for hashing. Since we are estimating coefficients for $O(\ell/\eta)$ candidates and can only afford $O(\ell)$ errors except with probability δ, the Markov inequality requires that each estimate fail with probability at most $\eta\delta$, which contributes the factor $O(\log 1/(\eta\delta))$. Thus we get $O((\ell/\eta) \log 1/(\eta\delta))$ for estimation, and

$$O((\ell/\eta)(\log(\eta N/\ell) \log 1/\delta + \log 1/(\eta\delta)),$$

overall. Note that we may assume $\eta > \ell/N$, since, otherwise, we may use $\ell/\eta > N$ measurements to recover trivially. Thus the overall number of measurements is

$$O((\ell/\eta)(\log(N/\ell) \log 1/\delta)).$$

Number of non-zeros.

The expected number of non-zeros in each column of the identification part of $\boldsymbol{\Phi}$ is $O(1)$ from hashing, times the factor $O(\log \eta N/\ell)$ from the (generally, dense) error-correcting code, times $O(\log 1/\delta)$ for repetition. Analysis of the estimation part is similar. We get $O(\log(N/\ell) \log 1/\delta)$ non-zeros altogether.

Post-measurement noise.

Finally, consider the effect of ν_2. Suppose there are m rows in $\boldsymbol{\Phi}$. As in previous work, the sign-flip matrix $\boldsymbol{S}$ applied to $\boldsymbol{a}$ but not to ν_2 insures that, effectively, $\mathbb{E}[\nu_2] = 0$; our assumption about ν_2 then implies that the energy of ν_2 in each component has expectation $(1/m)\|\nu_2\|_2^2$. A careful inspection of the above proof indicates that ν_1 enters only through the expected energy in each bucket, which is $\Theta((\eta/\ell)\|\nu_1\|_2^2)$; the error-correcting code and parallel repetitions result in energy $\Theta((\eta/\ell)\|\nu_1\|_2^2) = \Theta((\gamma/m)\|\nu_1\|_2^2)$ in each component of $\boldsymbol{\Phi a}$. The error $(1+\eta)\|\nu_1\|_2^2$ represents the "inevitable" error $\|\nu_1\|_2^2$ due to terms outside the top $O(\ell)$ that are not recovered by the algorithm, plus "excess" error $\eta\|\nu_1\|_2^2$, due to missing large terms and introducing additional error through many small coefficient approximation errors and a few arbitrarily bad approximation errors, where the threshold for "good" depends on ν_1. Since ν_2 does not affect the inevitible error, we can replace $(\gamma/m)\|\nu_1\|_2^2$ with $(\gamma/m)\|\nu_1\|_2^2 + (1/m)\|\nu_2\|_2^2$ when figuring the excess error, giving the claimed result. (Below we will see that γ can be viewed as a normalization factor for $\boldsymbol{\Phi}$, that makes $\boldsymbol{\Phi}\nu_1$ and ν_2 comparable.) Full details will be given in the journal version of this paper.

This concludes the proof of the LIM lemma, Lemma 3.

4.2.2 Normalization of the Measurement Matrix

Next we consider the normalization of the overall matrix $\mathbf{\Phi}$ from Section 3.1.

As has been observed [2], $\mathbf{\Phi}$ should be normalized in the setting of $\nu_2 \neq 0$. Otherwise, the matrix $\mathbf{\Phi}$ can be scaled up by an arbitrary constant factor $c > 1$ which can be undone by the decoding algorithm: Let $\mathcal{D}'$ be a new decoding algorithm that calls the old decoding algorithm $\mathcal{D}$ as $\mathcal{D}'(\boldsymbol{y}) = \mathcal{D}(\frac{1}{c}\boldsymbol{y})$, so that $\mathcal{D}'(c\mathbf{\Phi}\mathbf{x} + \nu_2) = \mathcal{D}(\mathbf{\Phi}\mathbf{x} + \frac{1}{c}\nu_2)$. Thus we would be able to *reduce* the effect of ν_2 by an arbitrary factor $c > 1$. In our "for each, $\ell_2 \leq C\ell_2$" model, an appropriate way to normalize $\mathbf{\Phi}$ is as follows.

DEFINITION 5. *The $\|\mathbf{\Phi}\|_{2\rightsquigarrow 2}$ norm of a randomly constructed matrix $\mathbf{\Phi}$ is*

$$\max_{\boldsymbol{x}\neq 0} \mathbb{E}\left[\frac{\|\mathbf{\Phi}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}\right].$$

Note that the usual 2-operator norm,

$$\|\mathbf{\Phi}\|_2 = \max_{\boldsymbol{x}\neq 0}\left[\frac{\|\mathbf{\Phi}\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2}\right]$$

is typically much larger than $\|\mathbf{\Phi}\|_{2\rightsquigarrow 2}$, which would lead to a much weaker result. But it corresponds to an adversary choosing $\mathbf{x}$ and ν_2 knowing the outcome $\mathbf{\Phi}$, which is counter to the spirit of the "for each" model in previous work. Here we assume the adversary knows the distribution on $\mathbf{\Phi}$, but not the outcome, when choosing $\mathbf{x}$ and ν_2.

Now we bound $\|\mathbf{\Phi}\|_{2\rightsquigarrow 2}$ for our $\mathbf{\Phi}$. It is straightforward to see that this is the maximum column ℓ_2 norm. In the j'th iteration, there are at most $j \log N/k$ non-zero entries, each of magnitude $\sqrt{\frac{c^j}{\log N/k}}$ for some c in the range $1/2 < c < 1$. It follows that

$$\|\mathbf{\Phi}\|_{2\rightsquigarrow 2}^2 \leq \left(\sum_j jc^j\right) \leq O(1),$$

if constants are chosen properly.

4.2.3 Invariant

Now we show that the invariant is satisfied, using the LIM lemma. That is all that remains to prove our main theorem:

Theorem 1 *There is an algorithm and distribution on matrices $\mathbf{\Phi}$ satisfying $\max_{\mathbf{x}} \mathbb{E}[\|\mathbf{\Phi}\mathbf{x}\|_2 / \|\mathbf{x}\|_2] = 1$ such that, given $\mathbf{\Phi}\mathbf{x}$, the parameters, and a concise description of $\mathbf{\Phi}$, the algorithm returns $\widehat{x}$ with approximation error $\|\mathbf{x} - \widehat{\mathbf{x}}\|_2^2 \leq (1+\epsilon)\|\nu_1\|_2^2 + \epsilon\|\nu_2\|_2^2$ with probability $\frac{3}{4}$. The algorithm runs in time $\frac{k}{\epsilon}\log^{O(1)} N$ and $\mathbf{\Phi}$ has $O\left(\frac{k}{\epsilon}\log\frac{N}{k}\right)$ rows. In expectation, there are $O(\log^2(k)\log(N/k))$ non-zeros in each column of $\mathbf{\Phi}$.*

PROOF. First, note that the matrices described in Section 3.1 have two additional features, compared with the matrices in the LIM lemma. First, there is a single random permutation matrix $\boldsymbol{P}$ that multiplies all the error-correcting code, hashing, and sign-flip matrices, and, second, the matrices in iteration j are multiplied by $c^{j/2}$, where c is an appropriate constant in the range $1/2 < c < 1$. Also note that ν_2 is not *a priori* guaranteed to be symmetric in expectation, as stipulated by the LIM lemma.

Consider the effect of ν_2. We would like to argue that the noise vector ν_2 is "distributed" at random and each measurement is corrupted by $\frac{\|\nu_2\|_2^2}{m}$, where m is the number of measurements, approximately its fair share of $\|\nu_2\|_2^2$. Unfortunately, the contributions of ν_2 to the various measurements are not independent as ν_2 is permuted, so we cannot use such a simple analysis. Nevertheless, they are negatively correlated and we can achieve the result we want using [10]. Henceforth, we assume ν_2 does corrupt each measurement in $\mathbf{\Phi}\mathbf{x}$ with the same expectation, just as in the LIM lemma.[2]

In iteration j, we make ℓ of the LIM lemma equal to $k/2^j$, η of the LIM lemma is $\Theta(\epsilon\beta^j)$, and $\delta = 2^{-j}$, for $\beta < 1$ to be specified below. Without loss of generality, assume $\|\nu_1\|_2 = \|\nu_2\|_2 = 1$, since our analysis can scale the signal (the algorithm does not need to know the scaling) and, if ν_1 and ν_2 have different energies, we can increase the weaker of the two. Then $\left\|\nu_1^{(0)}\right\|_2 = 1$.

It is straightforward to confirm that $\left\|\sigma^{(j+1)}\right\|_0 \leq k/2^{j+1}$, provided the invariant held at the previous iteration. We now turn to $\left\|\nu_1^{(j+1)}\right\|_2$.

At the beginning of the j'th iteration, $\left\|\nu_1^{(j)}\right\|_2^2 \leq 1 + \epsilon\left(1 - \left(\frac{3}{4}\right)^j\right)$. This means that $1 \leq \left\|\nu_1^{(j)}\right\|_2^2 \leq 2$ remains unchanged up to the factor 2. By the LIM lemma and the discussion at the beginning of 4.2.2, each repetition gives, with high probability, an estimate with $\left\|\nu_1^{(j+1)}\right\|_2^2 - \left\|\nu_1^{(j)}\right\|_2^2$ at most $\epsilon\beta^j\left(\left\|\nu_1^{(j)}\right\|_2^2 + c^{-j}\|\nu_2\|_2^2\right)$, where $\sqrt{c^j / \log N/k}$ is the magnitude of the $O(\log N/k)$ non-zeros in each repetition at iteration j. It follows that the median over repetitions of this estimate satisfies the same bound with high probability. Since $1 \ll c^{-j}$, it follows that $1 + c^{-j} \approx c^{-j}$. If we put $\beta \approx 5/8$ and $c \approx 5/6$, the invariant is satisfied. □

4.3 Efficiency

4.3.1 Number of Measurements

The number of measurements in iteration j is computed as follows. There are $\log(\delta) = O(j)$ parallel repetitions in iteration j. They each consist of $O(k/(\epsilon\beta^j 2^j)\log(N/k))$ measurements, where $\beta = 5/8$. That is, the number of measurements is

$$\Theta\left(jk\left(\frac{4}{5}\right)^j \log\frac{N}{k}\right) = (k/\epsilon)\log\frac{N}{k}\left(\frac{4}{5} + o(1)\right)^j.$$

Thus we have a sequence bounded by a geometric sequence with ratio less than 1. The sum, over j, is bounded by $O((k/\epsilon)\log(N/k))$.

4.3.2 Encoding, Decoding, and Update Time

The encoding time is bounded by N times the number of non-zeros in each column of the measurement matrix. This was analyzed above in Section 4.2; there are $\log(j)\log(N/k)$ non-zeros per column in iteration j, for $j \leq O(\log(k))$, so the total is $\log^2(k)\log(N/k)$ non-zeros per column. This is

[2]Alternatively, note that one could plausibly model ν_2 as acting independently and identically on all measurements, leading to a weaker, but simpler and self-contained result.

suboptimal by the factor $\log^2(k)$. By comparison, however, some proposed methods use dense matrices, which are suboptimal by the exponentially-larger factor k.

When constructing the matrix for measuring the original signal or some intermediate representation, our algorithm will need to find, quickly, the bucket to which an index i is hashed and a codeword for i, where i is in the range $1 \leq i \leq N$. Note that it is crucial that we use $O(\log(N/B))$ bits for the codeword, where B is the number of buckets, and not $\log(N)$ bits. This means we need to find codewords for just the i's hashed to a particular bucket. Upon decoding, we need to be able to find i from its codeword, quickly.

We can use a pseudorandom number generator that hashes i to a bucket j if $jN/B \leq ai + b \bmod N < (j+1)N/B$ for random a and b. Then encode i by $E(ai+b)$, assuming quick encoding for numbers in the contiguous range 1 to N/B. To decode, we first recover $ai + b$, whence we easily recover i. An example is given in Figure 1.

Another issue is the time to find and to encode and to decode the error-correcting code. Observe that the length of the code is $O(\log \log N)$. We can afford time exponential in the length, *i.e.*, time $\log^{O(1)} N$, for finding, encoding, and decoding the code. These tasks are straightforward in that much time.

5. CONCLUSION AND OPEN PROBLEMS

In this paper, we construct an approximate sparse recovery system that is essentially optimal: the recovery algorithm is a sublinear algorithm (with near optimal running time), the number of measurements meets a lower bound, and the update time, encode time, and column sparsity are each within log factors of the lower bounds. We leave the following problems open, and make conjectures.

- We conjecture that with a few modifications to the distribution on measurement matrices, we can extend this result to the $\ell_1 \leq C\ell_1$ and $\ell_2 \leq (C/\sqrt{k})\ell_1$ error metric guarantees, still under the "for each" model. We do not, however, think that this approach can be extended to the "for all" signal model (all current sublinear algorithms use at least one factor $O(\log N)$ additional measurements) and leave open the problem of designing a sublinear time recovery algorithm and a measurement matrix with an optimal number of rows for this setting, under any suitable error metric.

- It remains to improve the number of nonzeros in the columns of our measurement matrix from

  $$O(\log^2(k) \log(N/k))$$

  to $O(\log(N/k))$.

- A straightforward way to encode a signal takes time proportional to the signal support size times the number of non-zeros per column of the measurement matrix. This is the case with our algorithm and the relevant related work. If the measurement matrix is properly structured, however—for example, if it consists of rows of a Fourier matrix in arithmetic progression—it is possible to multiply the measurement matrix by a vector faster than the trivial algorithm. Thus it would be interesting to improve the encode time of our algorithm (assuming a dense signal), even if one cannot improve the number of non-zeros per column.

- As discussed earlier, Do Ba et al. [2] give a lower bound of $\Omega(k \log(N/k))$ for the number of measurements, for constant ϵ, and Wainwright [15] gives a lower bound of $\Omega((k/\epsilon) \log(N/k))$, under certain restrictions, for problems and in models that are not exactly the same as ours. These lower bounds can be regarded as characterizing obstacles in our setting. It remains to give a lower bound for the dependence on ϵ, and to meet it.

  Note that the number of measurements used by our algorithm is

  $$O((k/\epsilon)(\log(\epsilon N/k) + \log(1/\epsilon))).$$

  Above we quoted the larger expression

  $$O((k/\epsilon) \log(N/k)).$$

  Note that we may assume that $\epsilon > k/N$ since, otherwise, we may use $k/\epsilon > N$ measurements and recover trivially. Also, if $\epsilon^{1.1} > k/N$ then

  $$\log(1/\epsilon) \leq O(\log(\epsilon N/k)).$$

  Thus the number of measurements used by our algorithm is $O((k/\epsilon) \log(\epsilon N/k))$ except for the (rare?) cases of $k/N < \epsilon < (k/N)^{.9}$. Nevertheless, we conjecture that the lower bound is $\Omega((k/\epsilon) \log(N/k))$, which quantitatively matches the result in [15], and that this holds even if one of ν_1 and ν_2 is zero.

  The dependence on ϵ seems to increase from $1/\epsilon$ to $1/\epsilon^2$ for the recovery of an exactly-k-sparse representation. Characterizing precisely the lower bound here would be nice, too.

6. REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. System Sci.*, 58(1):137–147, 1999.

[2] K. Do Ba, P. Indyk, E. Price, and D. Woodruff. Lower bounds for sparse recovery. In *ACM SODA*, page to appear, 2010.

[3] E. J. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Comm. Pure Appl. Math.*, 59(8):1208–1223, 2006.

[4] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *ICALP*, 2002.

[5] Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. k-mismatch with don't cares. In *ESA*, pages 151–162, 2007.

[6] G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proc. ACM Principles of Database Systems*, pages 296–306, 2003.

[7] G. Cormode and S. Muthukrishnan. Improved data stream summaries: The count-min sketch and its applications. *FSTTCS*, 2004.

[8] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for Compressed Sensing. In *Proc. 40th Ann. Conf. Information Sciences and Systems*, Princeton, Mar. 2006.

[9] D. L. Donoho. Compressed Sensing. *IEEE Trans. Info. Theory*, 52(4):1289–1306, Apr. 2006.

Figure 1: Example measurement matrix for identification. Here $N = 11$, $k = 3$, and, in the hashing $i \mapsto a + bi$, we have $a = 1$ and $b = 4$, so that the sequence $i = \langle 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\rangle$ is mapped to $\langle 1, 5, 9, 2, 6, 10, 3, 7, 0, 4, 8\rangle$. The three buckets are $\{i : 0 \le a + bi < 4\}, \{i : 4 \le a + bi < 8\}$, and $\{i : 8 \le a + bi < 11\}$. We number starting from 0, so $0 \le i < 11$. In this example, we use two rows of ones and the double repetition code instead of a good code.

$$\boldsymbol{H} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

If ρ is the top row of $\boldsymbol{H}$, which maps $\langle 8, 0, 3, 6\rangle$ to $\langle 0, 1, 2, 3\rangle$, and with $\boldsymbol{S}$ arbitrary, then

$$(\rho \ltimes_{\mathrm{r}} \boldsymbol{C}) \odot \boldsymbol{S} = \begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

[10] Devdatt Dubhashi, Volker Priebe, and Desh Ranjan. Negative dependence through the FKG inequality. In *Research Report MPI-I-96-1-020, Max-Planck-Institut für Informatik, Saarbrucken*, 1996.

[11] Yaniv Erlich, Kenneth Chang, Assaf Gordon, Roy Ronen, Oron Navon, Michelle Rooks, and Gregory J. Hannon. Dna sudoku—harnessing high-throughput sequencing for multiplexed specimen analysis. *Genome Research*, 19:1243—1253, 2009.

[12] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. One sketch for all: fast algorithms for compressed sensing. In *ACM STOC 2007*, pages 237–246, 2007.

[13] P. Indyk and M. Ruzic. Near-optimal sparse recovery in the l_1 norm. *FOCS*, 2008.

[14] D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harmonic Anal.*, 2008. To appear.

[15] M. Wainwright. Information-theoretic bounds on sparsity recovery in the high-dimensional and noisy setting. *IEEE Transactions on Information Theory*, 55(12):5728–5741, 2009.

[16] Y. H. Zheng, N. P. Pitsianis, and D. J. Brady. Nonadaptive group testing based fiber sensor deployment for multiperson tracking. *IEEE Sensors Journal*, 6(2):490–494, 2006.

[17] Y.H. Zheng, D. J. Brady, M. E. Sullivan, and B. D. Guenther. Fiber-optic localization by geometric space coding with a two-dimensional gray code. *Applied Optics*, 44(20):4306–4314, 2005.