Advances in Sparse Signal Recovery Methods

by

Radu Berinde

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2009

Author .....
Department of Electrical Engineering and Computer Science
August 21, 2009

Certified by .....
Piotr Indyk
Associate Professor
Thesis Supervisor

Accepted by .....
Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Advances in Sparse Signal Recovery Methods

by

Radu Berinde

Submitted to the Department of Electrical Engineering and Computer Science
on August 21, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The general problem of obtaining a useful succinct representation (*sketch*) of some piece of data is ubiquitous; it has applications in signal acquisition, data compression, sub-linear space algorithms, etc. In this thesis we focus on *sparse recovery*, where the goal is to recover sparse vectors exactly, and to approximately recover nearly-sparse vectors. More precisely, from the short representation of a vector x, we want to recover a vector x^* such that the approximation erorr $\|x - x^*\|$ is comparable to the "tail" $\min_{x'} \|x - x'\|$ where x' ranges over all vectors with at most k terms. The sparse recovery problem has been subject to extensive research over the last few years, notably in areas such as *data stream computing* and *compressed sensing*.

We consider two types of sketches: linear and non-linear. For the linear sketching case, where the compressed representation of x is Ax for a *measurement matrix* A, we introduce a class of binary sparse matrices as valid measurement matrices. We show that they can be used with the popular geometric "ℓ_1 minimization" recovery procedure. We also present two iterative recovery algorithms, Sparse Matching Pursuit and Sequential Sparse Matching Pursuit, that can be used with the same matrices. Thanks to the sparsity of the matrices, the resulting algorithms are much more efficient than the ones previously known, while maintaining high quality of recovery. We also show experiments which establish the practicality of these algorithms.

For the non-linear case, we present a better analysis of a class of *counter* algorithms which process large streams of items and maintain enough data to approximately recover the item frequencies. The class includes the popular FREQUENT and SPACESAVING algorithms. We show that the errors in the approximations generated by these algorithms do not grow with the frequencies of the most frequent elements, but only depend on the remaining "tail" of the frequency vector. Therefore, they provide a non-linear sparse recovery scheme, achieving compression rates that are an order of magnitude better than their linear counterparts.

Thesis Supervisor: Piotr Indyk
Title: Associate Professor

Acknowledgments

First and foremost, I want to thank my advisor, Piotr Indyk, to whom I owe all my knowledge in this field; thank you for your effort, for your support, and for your lenience. I am also grateful for the useful feedback on this thesis.

Many thanks to my coauthors, whose work contributed to the results presented in this thesis: Graham Cormode, Anna Gilbert, Piotr Indyk, Howard Karloff, and Martin Strauss.

I am grateful to the many people who contributed to my early technical education and without whose help I would not be an MIT student today: Rodica Pintea, Adrian Atanasiu, Andrei Marius, Emanuela Cerchez, and others.

Finally, I want to thank Corina Tarniţă, as well as my family and all my friends, for their support over the years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data compression is the process of encoding information using a smaller number of bits than a regular representation requires. Compression is a fundamental problem in information technology, especially given the enormous amounts of data generated and transmitted today. From high-resolution sensors to massive distributed systems, from bioinformatics to large-scale networking, it is becoming increasingly clear that there is truth behind the popular saying "Data expands to fill the space available for storage."

While the general problem of data compression has been very well studied, there are specific situations which require new frameworks. In this thesis we focus on novel approaches for performing *lossy* data compression. Such compression is achieved by throwing away some - hopefully unimportant - information, and creating an acceptable approximation of the original data. Lossy compression allows us to obtain highly efficient encodings of the data that are often orders of magnitude smaller than the original data.

Our compression frameworks utilize the concept of *sparse approximation* ([Don04, CRT06, GSTV07, Gro06]). We define this concept in the following section[1].

1.1 Sparse approximations

We can denote any discrete signal or data as a real vector x in a high dimensional space. A k-sparse signal is a vector which has at most k non-zero components (see A.1.2). The simplest guarantee for a sparse recovery framework entails that x can be recovered from

[1]See appendix A for an overview of the basic mathematical notations we use in this thesis.

original	10	1	0	−5	2	23	1	1	−9	7	0	15	0	2
k=6	10	0	0	−5	0	23	0	0	−9	7	0	15	0	0
k=4	10	0	0	0	0	23	0	0	−9	0	0	15	0	0
k=2	0	0	0	0	0	23	0	0	0	0	0	15	0	0

Figure 1-1: Example of a signal and its optimal 6-sparse, 4-sparse, 2-sparse approximations

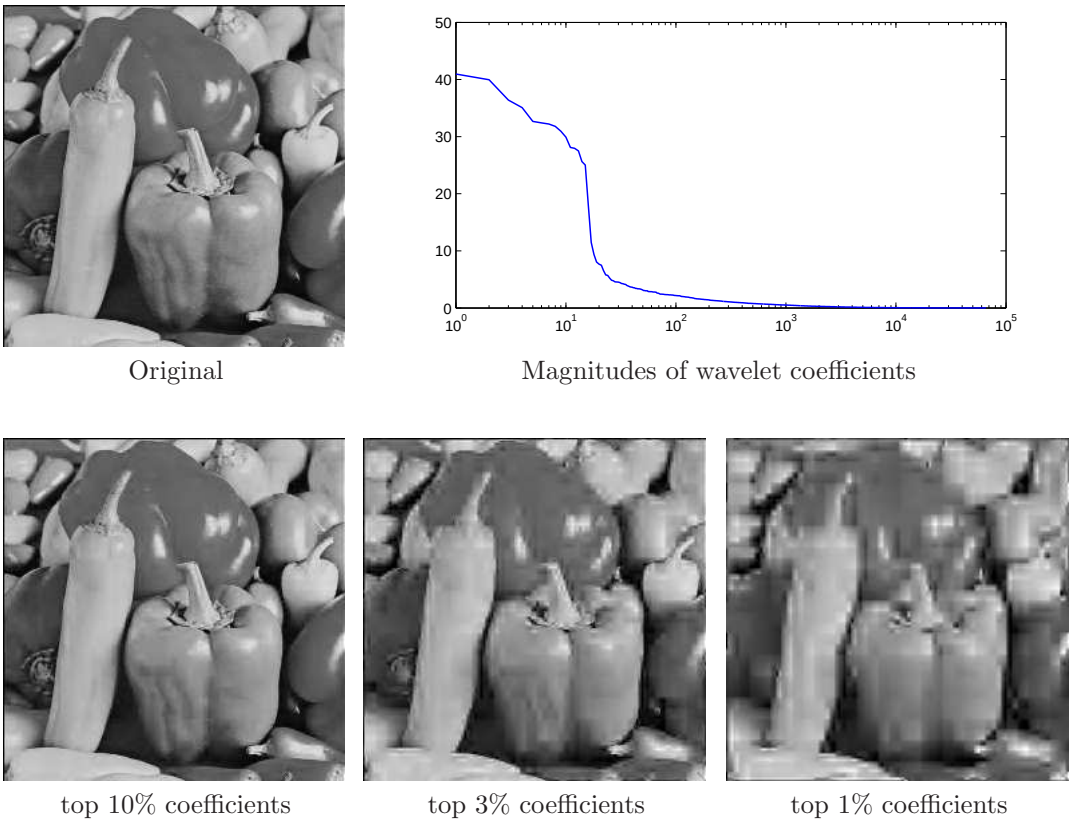
its succinct representation if it is k-sparse, for a certain value of k. The value of k depends on the size of the signal as well as on the size of the sketch. This type of guarantee allows us to easily test the performance of algorithms in practice: one can compress signals with different sparsities and check for correct recovery.

In practice signals are rarely sparse; however, they are often "nearly" sparse in that most of their weight is distributed on a small number of components. To extend the guarantee for these vectors, we must allow some error in the recovery; the allowed error must be at least as much as the small components - from the "tail" of the vector - amount to. To quantify this notion, we let $x^{(k)}$ be the k-sparse vector which is closest to x, i.e.

$$x^{(k)} = \underset{k-\text{sparse } x'}{\operatorname{argmin}} \|x - x'\|_p$$

Regardless of the norm p, the optimum is always achieved when $x^{(k)}$ retains the largest (in absolute value) k components of x. An example is shown in figure 1-1. Note that $x^{(k)}$ is not necessarily unique, however we are not interested in $x^{(k)}$ itself but in the error $\|x - x^{(k)}\|_p$.

In spite of its simplicity, the concept of sparse approximations is very powerful. The key to making use of this concept lies in the manner in which we choose to represent our data as a signal vector x. In many cases the trivial representation of a signal is not useful; however, an adequate *transform* associates nearly-sparse vectors to real-world data. For example, figure 1-2 shows the result of a *wavelet* transform on an image. Other examples include the JPEG image format, where only the largest values in the Discrete Cosine Transform representation are retained; and the MP3 music format which discards the less important Fourier frequencies in sounds. In all these examples, sparse approximations (as



Original | Magnitudes of wavelet coefficients

top 10% coefficients | top 3% coefficients | top 1% coefficients

Figure 1-2: Example of sparse approximations in a wavelet basis. On top: original 256×256 image along with a plot of (sorted) magnitudes of the `db2` wavelet coefficients. On bottom: resulting images when only the largest 10%, 3%, 1% coefficients are retained

defined by the above guarantees) yield useful results; the sparse approximation framework formally captures the general concept of lossy data compression.

1.2 Sparse recovery

The goal of sparse recovery is to obtain, from the compressed representation of a vector x, a "good" sparse approximation to x, i.e. a vector x^* such that the recovery error $\|x - x^*\|$ is "close" to the optimal sparse approximation error $\|x - x^{(k)}\|$. This can be formalized in several ways. In the simplest case of the ℓ_p/ℓ_p *guarantee*, we require that, for some constant C

$$\|x - x^*\|_p \leq C\|x - x^{(k)}\|_p$$

Sometimes, for technical reasons, we aim to achieve a *mixed* ℓ_p/ℓ_1 *guarantee*, where we require that

$$\|x - x^*\|_p \leq \frac{C}{k^{1-1/p}} \|x - x^{(k)}\|_1$$

Notice that the term $k^{1-1/p}$ is similar to the term in the ℓ_1/ℓ_p norm inequality (equation A.1). In general, the above guarantees do not imply that the recovered vector x^* must be sparse.

The guarantees we usually see in practice are ℓ_1/ℓ_1, ℓ_2/ℓ_2, and ℓ_1/ℓ_2. They are not in general directly comparable to each other (see appendix A.1.4 for a discussion). However, all sparse approximation guarantees imply the exact recovery guarantee when x is k-sparse: in this case the minimum error $\|x - x^{(k)}\|$ is 0 and any of the above guarantees implies $x^* = x$.

1.3 Types of representations and our results

The manner in which a succinct representation of the signal is acquired or computed depends strongly on the particular application. The method of *linear measurements* - in which the compressed representation is a linear function of the vector - is useful for most applications and will be the main focus of the thesis; in addition, we also present a class of results for a specialized problem in which non-linear measurements can be used with better results.

The results presented in this thesis were published as [BI08, BGI+08, BIR08, BI09] and [BCIS09].

1.3.1 Linear compression

When the measurements are linear, the representation, or *sketch* of x is simply $b = \boldsymbol{A}x$, where $\boldsymbol{A}$ is the $m \times n$ measurement matrix. The number of measurements, and thus the size of the sketch, is m. A solution to the problem of recovering a sparse approximation from linear sketches entails describing an algorithm to construct the matrix $\boldsymbol{A}$ and a corresponding recovery algorithm that given $\boldsymbol{A}$ and $b = \boldsymbol{A}x$ can recover either x or a sparse approximation of x - i.e. a vector x^* that satisfies one of the sparse approximation guarantees above.

In this thesis we introduce measurement matrices which are: *binary* - they contain only values of 1 and 0 - and *sparse* - the overwhelming majority of elements are 0. The main advantage of sparse matrices is that they require considerably less space and allow for faster algorithms. We show that these matrices allow sparse recovery using the popular "geometric" recovery process of finding x^* such that $Ax^* = Ax$ and $\|x^*\|_1$ is minimal. We also present two iterative recovery algorithms - Sparse Matching Pursuit (SMP) and Sequential Sparse Matching Pursuit (SSMP) - that can be used with the same matrices. These algorithms are important because they - along with the EMP algorithm [IR08] - are the first to require an asymptotically optimal number of measurements as well as near-linear decoding time. In addition, we present experimental results which establish the practicality of these methods. Our results in linear sketching are discussed in chapter 2.

Linear sketching finds use in many varied applications, in fields from compressed sensing to data stream computations. Some relevant examples are described in section 1.4.

1.3.2 Non-linear compression

We also present a result outside of the linear sketching realm, which applies to a class of *counter* algorithms; such algorithms process large streams of items and maintain enough data to approximately recover the item frequencies. The class includes the popular Frequent and SpaceSaving algorithms; we show that the errors in the approximations generated by these algorithms do not grow with the frequencies of the most frequent elements, but only depend on the remaining "tail" of the frequency vector. This implies that these heavy-hitter algorithms can be used to solve the more general sparse recovery problem. This result is presented in chapter 3.

1.4 Sample applications

To illustrate how the problem of recovering sparse approximations can be useful in practice we briefly discuss some specific applications from streaming algorithms, compressed sensing, and group testing.

1.4.1 Streaming algorithms: the network router problem

The ability to represent signals in sublinear space is very useful in streaming problems, in which algorithms with limited memory space process massive streams of data (see the surveys [Mut03, Ind07] on streaming and sublinear algorithms for a broad overview of the area). We discuss the simple example of a network router that needs to maintain some useful statistics about past packets, such as the most frequent source or destination addresses, or perhaps the most frequent source-destination pairs. The total number of packets routed, as well as the number of distinct source-destination pairs, grows quickly to an unmanageable size, so a succinct way of representing the desired statistics is needed. Of course, in order to achieve significantly smaller space requirements we must settle for non-exact results (approximation) and/or some probability of failure (randomization).

This problem fits our framework if we are to represent the statistic by a high-dimensional vector x; for example x_u can be the number of packets from source address u. A packet that arrives from source i corresponds to the simple linear operation $x \leftarrow x + e_i$, where e_i is the i-th row of the $n \times n$ identity matrix. If we are interested in the most traffic-heavy sources, our aim is to obtain (approximately) the heaviest elements of x. When a small number of sources are responsible for a large part of the total traffic - which might be the useful case in practice - this is achieved by recovering a sparse approximation to vector x.

Because of the linearity of the update operation, the method of linear sketches is a great match for this problem. If the sketch of x is $\boldsymbol{A}x$, the sketch of $x+e_i$ is $\boldsymbol{A}(x+e_i) = \boldsymbol{A}x+\boldsymbol{A}e_i$. Thus we can directly update the sketch with each operation; note that $\boldsymbol{A}e_i$ is simply the i-th column of $\boldsymbol{A}$. The sparsity of the matrix $\boldsymbol{A}$ is critical to the performance of the algorithm: if only a small fraction of the values of $\boldsymbol{A}e_i$ are non-zero, the update step can be performed very quickly. Note that in this setting, there is the additional requirement that we must be able to represent or generate $\boldsymbol{A}$'s columns using limited space: $\boldsymbol{A}$ has n columns, so storing $\boldsymbol{A}$ explicitly would defeat the purpose of using less than $O(n)$ space).

While this problem is a good witness to the versatility of linear sketching, there exist better *counter* algorithms that solve this specific problem. These algorithms maintain a small set of potentially frequent elements along with counters which estimate their frequencies. Such algorithms and our new analysis of their approximation quality are described

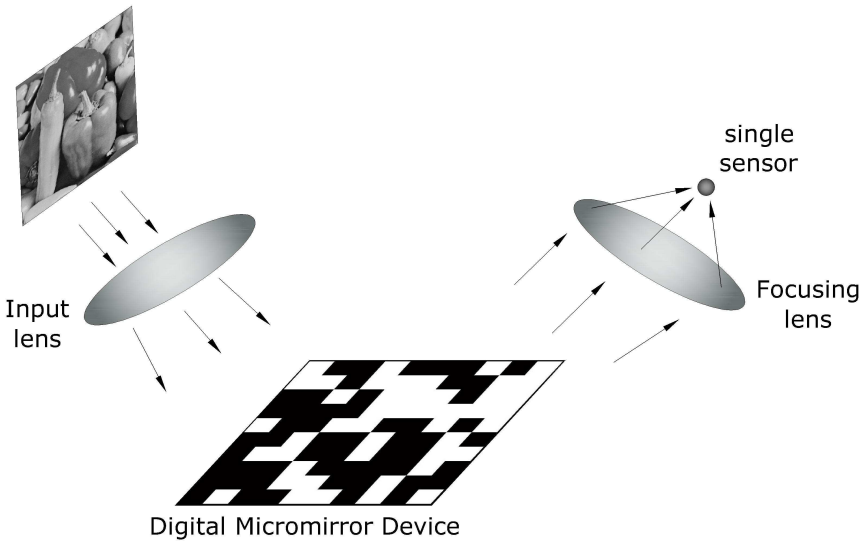


Figure 1-3: The single pixel camera concept

in chapter 3.

1.4.2 Compressed sensing: the single-pixel camera

The problem of sparse recovery is central to the field of compressed sensing, which challenges the traditional way of acquiring and storing a signal (e.g. a picture) - which involves sensing a high-resolution signal and then compressing it - inherently throwing away part of the sensed data in the process. Instead, compressed sensing attempts to develop methods to sense signals *directly* into compressed form. Depending on the application, this can lead to more efficient or cost-effective devices.

One such application is the single-pixel-camera developed at Rice University [TLW+06, DDT+08, Gro08], shown in figure 1-3. The camera uses a Digital Micromirror Device, which is a small array of microscopic mirrors; each mirror corresponds to a pixel of the image, and can be quickly rotated to either reflect the light towards a lens ("on" state) or away from it ("off" state). Such devices are currently used to form images in digital projectors; in our case, the mirrors are used to direct only a subset of pixels towards a *single sensor*, which reports the cumulated light level. Note that the micromirrors can turn on and off very quickly, and thus one pixel can be partially reflected as determined by the ratio between the on and off time (pulse-width-modulation) - much in the same way a projector is able to generate many shades of gray. In effect, the described process

results in a linear measurement $x \cdot u$ of the (flattened) image vector x, where u represents the setting of the mirrors; by repeating this process a number of times, we can sense a number of measurements that constitute a linear sketch $\boldsymbol{A}x$ of the signal x. Note that for each measurement, the setting of the micromirrors u is directed by the corresponding line of the measurement matrix $\boldsymbol{A}$.

In practice, building such a device can be advantageous if arrays of high-resolution sensors (as used in regular digital cameras) are not available for the given application; this can be the case with some ranges of the non-visible light spectrum, such as terahertz radiation.

1.4.3 Group testing

Another application of sparse recovery is group testing [GIS08, Che09, DH93]; the problem is to devise tests that efficiently identify members of a group with a certain rare property. The earliest example of group testing was used in World War II to identify men who carry a certain disease [Dor43]: the idea was to avoid individual testing of all candidates by pooling up blood from multiple individuals and testing the group sample first.

Recovering the sparse vector identifying special members is thus exactly what group testing aims. Depending on the exact setting, the tests can yield more than a binary result - e.g. the amount of disease in a sample, or the fraction of defect items; this allows us to take linear measurements. The density of the measurement matrix is again relevant as a sparse matrix ensures that only a relatively small number of items are grouped for a test (which might be critical for practical reasons).

Chapter 2

Linear Measurements: Sparse Matrices

2.1 Introduction

In this chapter we present results that fit the linear measurements framework, in which the aim is to design measurement matrices $\boldsymbol{A}$ and algorithms to recover sparse approximations to vectors x from the linear sketch $\boldsymbol{A}x$. Linear sketching exhibits a number of useful properties:

- One can maintain the sketch $\boldsymbol{A}x$ under coordinate increases: after incrementing the i-th coordinate x_i, the sketch becomes $\boldsymbol{A}(x + e_i) = \boldsymbol{A}x + \boldsymbol{A}e_i$
- Given the sketches of two signals x and y, one can compute the sketch of the sum $x + y$ directly, since $\boldsymbol{A}(x + y) = \boldsymbol{A}x + \boldsymbol{A}y$
- One can easily make use of sparsity in any linear basis: if $\boldsymbol{B}$ is a change of base matrix such that sparse approximations to $\boldsymbol{B}x$ are useful, one can combine $\boldsymbol{B}$ with a measurement matrix $\boldsymbol{A}$ and use the matrix $\boldsymbol{AB}$ for sketching. Sparse recovery algorithms can then be used to recover sparse approximations y' to $y = \boldsymbol{B}x$ from the sketch $\boldsymbol{AB}x$, and finally useful approximations $x' = \boldsymbol{B}^{-1}y'$ for the initial signal. In practice $\boldsymbol{B}$ can represent for example a Fourier or wavelet basis. Figure 1-2 shows an example of how sparse approximations in a wavelet basis can be very useful for images.

- In some applications, linearity of measurements is inherent to the construction of a device, and thus linear sketching is the only usable method.

These and other properties enable linear sketching to be of interest in several areas, like computing over data streams [AMS99, Mut03, Ind07], network measurement [EV03], query optimization and answering in databases [AMS99], database privacy [DMT07], and compressed sensing [CRT06, Don06]; some particular applications are discussed in section 1.4.

2.2 Background

The early work on linear sketching includes the *algebraic* approach of [Man92] (cf. [GGI+02a]). Most of the later algorithms, however, can be classified as either *combinatorial* or *geometric*. The geometric approach utilizes geometric properties of the measurement matrix $\boldsymbol{A}$, which is traditionally a dense, possibly random matrix. On the other hand, the combinatorial approach utilizes sparse matrices, interpreted as adjacency matrices of sparse (possibly random) graphs, and uses combinatorial techniques to recover an approximation to the signal. The results presented in this thesis constitute a unification of these two approaches - each of which normally has its own advantages and disadvantages. This is achieved by demonstrating geometric properties for sparse matrices, enabling geometric as well as combinatorial algorithms to be used for recovery. We obtain new measurement matrix constructions and algorithms for signal recovery, which compared to previous algorithms, are superior in either the number of measurements or computational efficiency of decoders. Interestingly, the recovery methods presented use the same class of measurement matrices, and thus one can use any or all algorithms to recover the signal *from the same sketch*.

Geometric approach

This approach was first proposed in [CRT06, Don06] and has been extensively investigated since then (see [Gro06] for a bibliography). In this setting, the matrix A is dense, with at least a constant fraction of non-zero entries. Typically, each row of the matrix is independently selected from an n-dimensional distribution such as Gaussian or Bernoulli.

The key property of the matrix $\boldsymbol{A}$ that allows signal recovery is the *Restricted Isometry Property* [CRT06]:

Definition 1 (RIP)**.** *A matrix $\boldsymbol{A}$ satisfies the Restricted Isometry Property property with parameters k and δ if for any k-sparse vector x*

$$(1-\delta)\|x\|_2 \leq \|\boldsymbol{A}x\|_2 \leq \|x\|_2$$

Intuitively, the RIP property states that the matrix approximately preserves lengths for sparse signals. The main result in geometric algorithms is the following: if the matrix $\boldsymbol{A}$ satisfies the RIP property, a sparse approximation for the signal can be computed from the sketch $b = \boldsymbol{A}x$ by solving the following convex program:

$$\min \ \|x^*\|_1 \text{ subject to } \boldsymbol{A}x^* = b. \tag{P1}$$

This result is somewhat surprising, as we are finding the vector with smallest ℓ_1 norm among all vectors in the subspace. The convex program (P1) can be recast as a linear program (see [CDS99]). The method can be extended to allow *measurement errors* or noise using the convex program

$$\min \ \|x^*\|_1 \text{ subject to } \|\boldsymbol{A}x^* - b\|_2 \leq \gamma \tag{P1-noise}$$

where γ is the allowed measurement error.

The advantages of the geometric approach include a small number of necessary measurements $O(k\log(n/k))$ for Gaussian matrices and $O(k\log^{O(1)} n)$ for Fourier matrices, and resiliency to measurement errors; in addition the geometric approach resulted in the first *deterministic* or *uniform* recovery algorithms, where a fixed matrix $\boldsymbol{A}$ was guaranteed to work for *all* signals x[1]. In contrast, the early combinatorial sketching algorithms only guaranteed $1-1/n$ probability of correctness for *each* signal x.[2]. The main disadvantage is the running time of the recovery procedure, which involves solving a linear program with n variables and $n+m$ constraints. The computation of a sketch $\boldsymbol{A}x$ can be performed efficiently only for some matrices (e.g. Fourier), and an efficient sketch update is not possible

[1]Note that a uniform guarantee does not prohibit the matrix $\boldsymbol{A}$ to be chosen randomly; the uniform guarantee can be of the form "a random matrix $\boldsymbol{A}$, with good probability, will uniformly be able to recover an approximation for any vector". Contrast this with "given a vector, a random matrix $\boldsymbol{A}$ will, with good probability, recover an approximation to the vector."

[2]Note however that the papers [GSTV06, GSTV07] showed that combinatorial algorithms can achieve deterministic or uniform guarantees as well.

(as it is with sparse matrices). In addition, the problem of finding an *explicit* construction of efficient matrices satisfying the RIP property is open [Tao07]; the best known explicit construction [DeV07] yields $\Omega(k^2)$ measurements.

Combinatorial approach

In the combinatorial approach, the measurement matrix $\boldsymbol{A}$ is sparse and often binary. Typically, it is obtained from the adjacency matrix of a sparse bipartite random graph. The recovery algorithms iteratively identifies and eliminates "large" coefficients of the vector. Examples of combinatorial sketching and recovery algorithms include [GGI+02b, CCFC02, CM04, GKMS03, DWB05, SBB06b, SBB06a, CM06, GSTV06, GSTV07, Ind08, XH07] and others.

The typical advantages of the combinatorial approach include fast recovery (often sublinear in the signal length n if $k \ll m$), as well as fast and incremental (under coordinate updates) computation of the sketch vector $\boldsymbol{A}x$. In addition, it is possible to construct efficient - albeit suboptimal - measurement matrices explicitly, at least for simple type of signals. For example, it is known [Ind08, XH07] how to explicitly construct matrices with $k2^{(\log \log n)^{O(1)}}$ measurements, for signals x that are exactly k-sparse. The main disadvantage of the approach is the suboptimal sketch length.

Connections

Recently, progress was made towards obtaining the advantages of both approaches by decoupling the algorithmic and combinatorial aspects of the problem. Specifically, the papers [NV09, DM08, NT08] show that one can use *greedy* methods for data compressed using *dense* matrices satisfying the RIP property. Similarly [GLR08], using the results of [KT07], show that sketches from (somewhat) sparse matrices can be recovered using linear programming.

The state-of-the art results (up to $O(\cdot)$ constants) are shown in table 2.1[3]. We present only the algorithms that work for arbitrary vectors x, while many other results are known

[3]Some of the papers, notably [CM04], are focused on a somewhat different formulation of the problem. However, it is known that the guarantees presented in the table hold for those algorithms as well. See Lecture 4 in [Ind07] for a more detailed discussion.

for the case where the vector x itself is always exactly k-sparse; e.g., see [TG05, DWB05, SBB06b, Don06, XH07]. The columns describe:

- citation,
- whether the recovery is **D**eterministic (uniform) or **R**andomized,
- sketch length,
- time to compute $\boldsymbol{A}x$ given x,
- time to update $\boldsymbol{A}x$ after incrementing one of the coordinates of x,
- time[4] to recover an approximation of x given $\boldsymbol{A}x$,
- approximation guarantee, and
- whether the algorithm is robust to noisy measurements.

The approximation error column shows the type of guarantee: $\ell_p \leq A\ell_q$ means that recovered vector x^* satisfies $\|x - x^*\|_p \leq A\|x^{(k)} - x\|_q$. The parameters $C > 1$, $c \geq 2$ and $a > 0$ denote absolute constants, possibly different in each row. The parameter ϵ denotes any positive constant. We assume that $k < n/2$. Some of the running times of the algorithms depend on the "precision parameter" R, which if is always upper-bounded by the norm of the vector x if its coordinates are integers.

2.3 Expanders, sparse matrices, and RIP-1

An essential tool for our constructions are *unbalanced expander graphs*. Consider a bipartite graph $G = (U, V, E)$. We refer to U as the "left" part, and refer to V as the "right" part; a vertex belonging to the left (respectively right) part is called a left (respectively right) vertex. In our constructions the left part will correspond to the set $\{1, 2, \ldots, n\}$ of coordinate indexes of vector x, and the right part will correspond to the set of row indexes of the measurement matrix. A bipartite graph is called *left-d-regular* if every vertex in the left part has exactly d neighbors in the right part.

For a set S of vertices of a graph G, the set of its neighbors in G is denoted by $\Gamma_G(S)$. The subscript G will be omitted when it is clear from the context, and we write $\Gamma(u)$ as a shorthand for $\Gamma(\{u\})$.

[4]In the decoding time column LP=LP(n, m, T) denotes the time needed to solve a linear program defined by an $m \times n$ matrix $\boldsymbol{A}$ which supports matrix-vector multiplication in time T. Heuristic arguments indicate that LP$(n, m, T) \approx \sqrt{n}T$ if the interior-point method is employed.

Paper	R/D	Sketch length	Encoding time	Sparsity/ Update time	Decoding time	Approximation error	Noise
[CCFC02, CM06]	R	$k \log^d n$	$n \log^d n$	$\log^d n$	$k \log^d n$	$\ell_2 \leq C\ell_2$	
	R	$k \log n$	$n \log n$	$\log n$	$n \log n$	$\ell_2 \leq C\ell_2$	
[CM04]	R	$k \log^d n$	$n \log^d n$	$\log^d n$	$k \log^d n$	$\ell_1 \leq C\ell_1$	
	R	$k \log n$	$n \log n$	$\log n$	$n \log n$	$\ell_1 \leq C\ell_1$	
[CRT06, RV06]	D	$k \log \frac{n}{k}$	$nk \log \frac{n}{k}$	$k \log \frac{n}{k}$	LP	$\ell_2 \leq \frac{C}{k^{1/2}}\ell_1$	Y
	D	$k \log^d n$	$n \log n$	$k \log^d n$	LP	$\ell_2 \leq \frac{C}{k^{1/2}}\ell_1$	Y
[GSTV06]	D	$k \log^d n$	$n \log^d n$	$\log^d n$	$k \log^d n$	$\ell_1 \leq C \log n \ell_1$	Y
[GSTV07]	D	$k \log^d n$	$n \log^d n$	$\log^d n$	$k^2 \log^d n$	$\ell_2 \leq \frac{\epsilon}{k^{1/2}}\ell_1$	
[GLR08] (k "large")	D	$k(\log n)^{d \log\log\log n}$	kn^{1-a}	n^{1-a}	LP	$\ell_2 \leq \frac{C}{k^{1/2}}\ell_1$	
[DM08]	D	$k \log \frac{n}{k}$	$nk \log \frac{n}{k}$	$k \log \frac{n}{k}$	$nk \log \frac{n}{k} \log R$	$\ell_2 \leq \frac{C}{k^{1/2}}\ell_1$	Y
[NT08]	D	$k \log \frac{n}{k}$	$nk \log \frac{n}{k}$	$k \log \frac{n}{k}$	$nk \log \frac{n}{k} \log R$	$\ell_2 \leq \frac{C}{k^{1/2}}\ell_1$	Y
	D	$k \log^d n$	$n \log n$	$k \log^d n$	$n \log n \log R$	$\ell_2 \leq \frac{C}{k^{1/2}}\ell_1$	Y
[IR08]	D	$k \log \frac{n}{k}$	$n \log \frac{n}{k}$	$\log \frac{n}{k}$	$n \log \frac{n}{k}$	$\ell_1 \leq (1+\epsilon)\ell_1$	Y
Sec. 2.4 [BGI+08]	D	$k \log \frac{n}{k}$	$n \log \frac{n}{k}$	$\log \frac{n}{k}$	LP	$\ell_1 \leq C\ell_1$	Y
Sec. 2.5 [BIR08]	D	$k \log \frac{n}{k}$	$n \log \frac{n}{k}$	$\log \frac{n}{k}$	$n \log \frac{n}{k} \log R$	$\ell_1 \leq C\ell_1$	Y
Sec. 2.6 [BI09]	D	$k \log \frac{n}{k}$	$n \log \frac{n}{k}$	$\log \frac{n}{k}$	$n \log \frac{n}{k} \log n \log R$	$\ell_1 \leq C\ell_1$	Y

Table 2.1: Summary of sparse recovery results. Last entries correspond to results presented in this thesis (section numbers are indicated).

Definition 2. *A bipartite, left-d-regular graph $G = (U, V, E)$ is a (k, d, ϵ)-*unbalanced expander *if any set $S \subset U$ of at most k left vertices has at least $(1-\epsilon)k|S|$ neighbors.*

Intuitively, the graph must "expand well" in that any small-enough set of left vertices has almost as many distinct neighbors as it is theoretically possible. Since expander graphs are meaningful only when $|V| < d|U|$, some vertices must share neighbors, and hence the parameter ϵ cannot be smaller than $1/d$. Using the probabilistic method (see A.1.5) one can show that there exist (k, d, ϵ)-expanders with $d = O(\log(|U|/k)/\epsilon)$. and $|V| = O(k \log(|U|/k)/\epsilon^2)$.

In practice, randomly generated graphs with the above parameters will be, with good probability, expanders. For many applications one usually prefers an *explicit* expander, i.e., an expander that can be generated in polynomial time by a deterministic algo-

rithm. No explicit constructions with the aforementioned (optimal) parameters are known. However, it is known [GUV07] how to explicitly construct expanders with left degree $d = O\left(((\log |U|)(\log s)/\epsilon)^{1+1/\alpha}\right)$ and right set size $(d^2 s^{1+\alpha})$, for any fixed $\alpha > 0$. For the results presented, we will assume expanders with the optimal parameters.

Consider the $m \times n$ adjacency matrix $\boldsymbol{A}$ of an unbalanced expander. Notice that $\boldsymbol{A}$ is binary and sparse, as its only nonzero values are d values of 1 on each column. Our methods use such matrices as measurement matrices for linear sketches. Traditionally, the geometric methods use matrices that exhibit the Restricted Isometry Property (definition 1). The sparse matrices described do not exhibit this property for the ℓ_2 metric; however, if we generalize the definition of RIP to other metrics, we can show that these matrices do exhibit the similar property for the ℓ_1 metric.

Definition 3 ($\text{RIP}_{p,k,\delta}$)**.** *A matrix $\boldsymbol{A}$ satisfies the $\text{RIP}_{p,k,\delta}$ property if for any k-sparse vector x*

$$(1-\delta)\|x\|_p \leq \|\boldsymbol{A}x\|_p \leq \|x\|_p$$

We loosely denote $\text{RIP}_{p,k,\delta}$ by RIP-p. Note that technically most matrices must be scaled with a proper factor to satisfy the above inequality. While dense Gaussian or Fourier matrices satisfy RIP-2, the following theorem from [BGI+08] establishes that sparse expander matrices satisfy RIP-1:

Theorem 1 (expansion $\implies$ RIP-1)**.** *Consider any $m \times n$ matrix $\boldsymbol{A}$ that is the adjacency matrix of a (k, d, ϵ)-unbalanced expander $G = (U, V, E)$, $|U| = n$, $|V| = m$ such that $1/\epsilon$, d are smaller than n. Then the scaled matrix $\boldsymbol{A}/d$ satisfies the $\text{RIP}_{1,k,2\epsilon}$ property.*

Proof. Let $x \in \mathbb{R}^n$ be a k-sparse vector. Without loss of generality, we assume that the coordinates of x are ordered such that $|x_1| \geq \ldots \geq |x_n|$.

We order the edges $e_t = (i_t, j_t)$, $t = 1 \ldots dn$ of G in a lexicographic manner. It is helpful to imagine that the edges $e_1, e_2 \ldots$ are being added to the (initially empty) graph. An edge $e_t = (i_t, j_t)$ causes a *collision* if there exists an earlier edge $e_s = (i_s, j_s), s < t$, such that $j_t = j_s$. We define E' to be the set of edges which do *not* cause collisions, and $E" = E - E'$. Figure 2-1 shows an example.

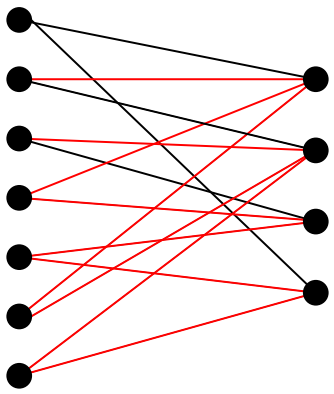


Figure 2-1: Example graph for proof of theorem 1: no-collision edges in E' are (dark) black, collision edges in E'' are (lighter) red

Lemma 1. *We have*

$$\sum_{(i,j)\in E"} |x_i| \leq \epsilon d \|x\|_1$$

Proof. For each $t = 1 \ldots dn$, we use an indicator variable $r_t \in \{0, 1\}$, such that $r_t = 1$ iff $e_t \in E''$. Define a vector $z \in \mathbb{R}^{dn}$ such that $z_t = |x_{i_t}|$. Observe that

$$\sum_{(i,j)\in E"} |x_i| = \sum_{e_t=(i_t,j_t)\in E} r_t |x_{i_t}| = r \cdot z$$

To upper bound the latter quantity, observe that the vectors satisfy the following constraints:

- The vector z is non-negative.
- The coordinates of z are monotonically non-increasing.
- For each *prefix set* $P_i = \{1 \ldots di\}$, $i \leq k$, we have $\|r_{|P_i}\|_1 \leq \epsilon di$ - this follows from the expansion properties of the graph G.
- $r_{|P_1} = 0$, since the graph is simple.

It is now immediate that for any r, z satisfying the above constraints, we have $r \cdot z \leq \|z\|_1 \epsilon$. Since $\|z\|_1 = d\|x\|_1$, the lemma follows. □

Lemma 1 immediately implies that $\|\boldsymbol{A}x\|_1 \geq d\|x\|_1(1 - 2\epsilon)$. Since for any x we have $\|\boldsymbol{A}x\|_1 \leq d\|x\|_1$, the theorem follows. □

This proof can be extended to show that the matrix not only satisfies $\mathrm{RIP}_{1,k,2\epsilon}$ but in fact satisfies $\mathrm{RIP}_{p,k,O(\epsilon)}$ for all $1 \leq p \leq 1 + 1/\log n$ (see [BGI+08]).

Interestingly, there is a very tight connection between the RIP-1 property and expansion of the underlying graph. The same paper [BGI+08] shows that a binary matrix $\boldsymbol{A}$ with d ones on each column[5] which satisfies RIP-1 must be the adjacency matrix of a good expander. This is important because without significantly improved explicit constructions of unbalanced expanders with parameters that match the probabilistic bounds (a long-standing open problem), we do not expect significant improvements in the explicit constructions of RIP-1 matrices.

2.4 ℓ_1-minimization with sparse matrices

The geometric approach involves solving (P1) via linear programming (LP). The main result presented in this section is that sparse matrices can be used within this solution: a matrix that satisfies RIP-1 allows recovery of sparse approximations via the ℓ_1-minimization program (P1). We thus show that sparse matrices are in this respect comparable with dense matrices (like those with elements chosen from the Gaussian distribution). In addition, the experimental section will show that in practice their performance is virtually indistinguishable from that of dense matrices.

We reproduce the proof, which appeared in [BGI+08] and [BI08]. The first part of the proof establishes that any vector in the nullspace of a RIP-1 matrix is "smooth", i.e. a large part of its ℓ_1 mass cannot be concentrated on a small subset of its coordinates. An analogous result for RIP-2 matrices and with respect to the ℓ_2 norm has been used before (e.g., in [KT07]) to show guarantees for LP-based recovery procedures. The second part of the proof establishes decodability via ℓ_1 minimization.

L1 Uncertainty Principle

Let $\boldsymbol{A}$ be the $m \times n$ adjacency matrix of a $(2k, d, \epsilon)$-unbalanced expander G. Let $\alpha(\epsilon) = (2\epsilon)/(1 - 2\epsilon)$. For any n-dimensional vector y, and $S \subset \{1 \ldots n\}$, we use y_S to denote an

[5]Note that for any binary matrix to have the RIP-1 property, it must have roughly the same number of ones on each column.

$|S|$-dimensional projection of y on coordinates in S. S^c is the complement of S (e.g. so that $y = y_S + y_{S^c}$).

Lemma 2. *Consider any $y \in \mathbb{R}^n$ such that $\boldsymbol{A}y = 0$, and let S be any set of k coordinates of y. Then we have*

$$\|y_S\|_1 \leq \alpha(\epsilon)\|y\|_1$$

Proof. Without loss of generality, we can assume that S consists of the largest (in magnitude) coefficients of y. We partition coordinates into sets $S_0, S_1, S_2, \ldots S_t$, such that (i) the coordinates in the set S_l are not-larger (in magnitude) than the coordinates in the set S_{l-1}, $l \geq 1$, and (ii) all sets but S_t have size k. Therefore, $S_0 = S$. Let $\boldsymbol{A'}$ be a submatrix of $\boldsymbol{A}$ containing rows from $\Gamma(S)$, the neighbors of S in the graph G.

By the $\text{RIP}_{1,2k,2\epsilon}$ property of the matrix $\boldsymbol{A}/d$ (as per theorem 1) we know that $\|\boldsymbol{A'}y_S\|_1 = \|\boldsymbol{A}y_S\|_1 \geq d(1-2\epsilon)\|y_S\|_1$. At the same time, we know that $\|\boldsymbol{A'}y\|_1 = 0$. Therefore

$$\begin{aligned}
0 = \|\boldsymbol{A'}y\|_1 &\geq \|\boldsymbol{A'}y_S\|_1 - \sum_{l\geq 1} \sum_{(i,j)\in E, i\in S_l, j\in\Gamma(S)} |y_i| \\
&\geq d(1-2\epsilon)\|y_S\|_1 - \sum_{l\geq 1} |E(S_l : \Gamma(S))| \min_{i\in S_{l-1}} |y_i| \\
&\geq d(1-2\epsilon)\|y_S\|_1 - \sum_{l\geq 1} |E(S_l : \Gamma(S))| \cdot \|y_{S_{l-1}}\|_1/k
\end{aligned}$$

From the expansion properties of G it follows that, for $l \geq 1$, we have $|\Gamma(S \cup S_l)| \geq d(1-\epsilon)|S \cup S_l|$. It follows that at most $d\epsilon 2k$ edges can cross from S_l to $\Gamma(S)$, and therefore

$$\begin{aligned}
0 &\geq d(1-2\epsilon)\|y_S\|_1 - \sum_{l\geq 1} |E(S_l : \Gamma(S))| \cdot \|y_{S_{l-1}}\|_1/k \\
&\geq d(1-2\epsilon)\|y_S\|_1 - d\epsilon 2k \sum_{l\geq 1} \|y_{S_{l-1}}\|_1/k \\
&\geq d(1-2\epsilon)\|y_S\|_1 - 2d\epsilon\|y\|_1
\end{aligned}$$

It follows that $d(1-2\epsilon)\|y_S\|_1 \leq 2d\epsilon\|y\|_1$, and thus $\|y_S\|_1 \leq (2\epsilon)/(1-2\epsilon)\|y\|_1$. □

LP recovery

The following theorem establishes the result if we apply it with $u = x$ and $v = x^*$, the solution of the ℓ_1-minimization program (P1); notice that $\boldsymbol{A}v = \boldsymbol{A}u$, $\|v\|_1 \leq \|u\|_1$, and

$\|u_{S^c}\|_1 = \|x - x^{(k)}\|_1$.

Theorem 2. *Consider any two vectors u, v such that for $y = v - u$ we have $\boldsymbol{A}y = 0$, and $\|v\|_1 \leq \|u\|_1$. Let S be the set of k largest (in magnitude) coefficients of u. Then*

$$\|v - u\|_1 \leq 2/(1 - 2\alpha(\epsilon)) \cdot \|u_{S^c}\|_1$$

Proof. We have

$$\begin{aligned} \|u\|_1 \geq \|v\|_1 &= \|(u+y)_S\|_1 + \|(u+y)_{S^c}\|_1 \\ &\geq \|u_S\|_1 - \|y_S\|_1 + \|y_{S^c}\|_1 - \|u_{S^c}\|_1 \\ &= \|u\|_1 - 2\|u_{S^c}\|_1 + \|y\|_1 - 2\|y_S\|_1 \\ &\geq \|u\|_1 - 2\|u_{S^c}\|_1 + (1 - 2\alpha(\epsilon))\|y\|_1 \end{aligned}$$

where we used Lemma 2 in the last line. It follows that

$$2\|u_{S^c}\|_1 \geq (1 - 2\alpha(\epsilon))\|y\|_1$$

□

We can generalize the result to show resilience to measurement errors: we allow a certain ℓ_1 error γ in the measurements so that $\|\boldsymbol{A}x - b\|_1 \leq \gamma$, and use the following linear program:

$$\min \ \|x^*\|_1 \text{ subject to } \|\boldsymbol{A}x^* - b\|_1 \leq \gamma \qquad \text{(P1")}$$

Note that 2γ is an upper bound for the resulting sketch difference $\beta = \|\boldsymbol{A}x - \boldsymbol{A}x^*\|_1$. As before, the following theorem is applied with $u = x$ and $v = x^*$, the solution to (P1") above.

Theorem 3. *Consider any two vectors u, v such that for $y = v - u$ we have $\|\boldsymbol{A}y\|_1 = \beta \geq 0$, and $\|v\|_1 \leq \|u\|_1$. Let S be the set of k largest (in magnitude) coefficients of u. Then*

$$\|v - u\|_1 \leq 2/(1 - 2\alpha(\epsilon)) \cdot \|u_{S^c}\|_1 + \frac{2\beta}{d(1 - 2\epsilon)(1 - 2\alpha(\epsilon))}$$

Proof. We generalize Lemma 2 to the case when $\|\boldsymbol{A}y\|_1 = \beta$ yielding

$$\|y_S\|_1 \leq \frac{\beta}{d(1 - 2\epsilon)} + \alpha(\epsilon)\|y\|_1$$

The proof is identical, noticing that $\|\boldsymbol{A}'y\|_1 \leq \beta$.

The proof of the theorem is then similar to that of Theorem 2. The extra term appears when we apply the lemma:

$$\begin{aligned}\|u\|_1 &\geq \|u\|_1 - 2\|u_{S^c}\|_1 + \|y\|_1 - 2\|y_S\|_1 \\ &\geq \|u\|_1 - 2\|u_{S^c}\|_1 + (1 - 2\alpha(\epsilon))\|y\|_1 - \frac{2\beta}{d(1-2\epsilon)}\end{aligned}$$

which implies

$$\|y\|_1 \leq \frac{2}{1-2\alpha(\epsilon)}\|u_{S^c}\|_1 + \frac{2\beta}{d(1-2\epsilon)(1-2\alpha(\epsilon))}$$

□

The factor $\frac{1}{d}$ suggests that increasing d improves the error bound; this is misleading as β is the absolute sketch error, and we expect sketch values to increase proportionally with d[6]. If we instead consider that the ℓ_1 measurement error is at most some fraction ρ of the total ℓ_1 norm of the sketch, i.e. $\|\boldsymbol{A}x - b\|_1 \leq \rho\|\boldsymbol{A}x\|_1$ then $\beta \leq 2\rho\|\boldsymbol{A}x\|_1 \leq d\|x\|_1$ and the above guarantee becomes

$$\|x^* - x\|_1 \leq \frac{2}{1-2\alpha(\epsilon)}\|x - x^{(k)}\|_1 + \frac{4\rho}{(1-2\epsilon)(1-2\alpha(\epsilon))}\|x\|_1$$

Thus the resulting fraction of induced ℓ_1 noise in the recovered vector is within a constant factor of the fraction of ℓ_1 noise in the sketch (regardless of d).

We have established that expander matrices can be used with linear programming-based recovery. The experimental results in section 2.7.2 show that the method is of practical use; in our experiments, sparse matrices behave very similarly to dense matrices in terms of sketch length and approximation error.

[6]For example, this is exactly true for positive vectors x^+ as $\|\boldsymbol{A}x^+\|_1 = d\|x^+\|_1$

Algorithm 1: SMP

$x^0 \leftarrow 0$;
for $j = 1 \dots T$ **do**
  $c \leftarrow b - \boldsymbol{A}x^{j-1}$; `/* Note:` $c = \boldsymbol{A}(x - x^{j-1}) + \mu$ `*/`
  foreach $i \in \{1 \dots n\}$ **do**
    $u_i^* \leftarrow \text{median}(c_{\Gamma(i)})$;
  $u^j \leftarrow H_{2k}[u^*]$; `/* From Lemma 3 we have` $\|u^j - (x - x^{j-1})\|_1 \le \|x - x^{j-1}\|/4 + C\eta$ `*/`
  $x^j \leftarrow x^{j-1} + u^j$; `/* Note:` $\|x - x^j\|_1 \le \|x - x^{j-1}\|/4 + C\eta$ `*/`
  $x^j \leftarrow H_k[x^j]$; `/* From Lemma 6 we have` $\|x - x^j\|_1 \le \|x - x^{j-1}\|/2 + 2C\eta$ `*/`

Figure 2-2: The Sparse Matching Pursuit algorithm

2.5 Sparse Matching Pursuit

The *Sparse Matching Pursuit* algorithm [BIR08] is presented in figure 2-2; it provides an ℓ_1/ℓ_1 guarantee. At each iteration it improves upon an estimate, using a subroutine similar to the Count-Min sketch [CM04]. Each iteration runs in time $O(nd)$, linear in the number of non-zeros in the matrix. The number of iterations is a logarithmic factor.

Consider any n-dimensional vector x that is k-sparse. The sketching matrix $\boldsymbol{A}$ is a $m \times n$ matrix induced by a (s, d, ϵ)-expander $G = (\{1 \dots n\}, \{1 \dots m\}, E)$, for $s = O(k)$ and a sufficiently small ϵ. Let μ be the m-dimensional "noise" vector, and let $b = \boldsymbol{A}x + \mu$ be the "noisy measurement" vector. Also, denote $\eta = \|\mu\|_1/d$.

Theorem 4. *For any k-sparse signal x and noise vector μ and given $b = \boldsymbol{A}x + \mu$, SMP recovers x^* such that $\|x - x^*\|_1 = O(\|\mu\|_1/d)$. The algorithm runs in time $O(nd \log \frac{d\|x\|_1}{\|\mu\|_1})$.*

For general vectors x, notice that $\boldsymbol{A}x + \mu = \boldsymbol{A}x^{(k)} + [\mu + \boldsymbol{A}(x - x^{(k)})]$, and $\|\boldsymbol{A}(x - x^{(k)})\|_1 \le d\|x - x^{(k)}\|_1$; then Theorem 4 immediately implies the following more general statement:

Corollary 1. *For any parameter k, any vector x and noise vector μ, given $b = \boldsymbol{A}x + \mu$, SMP recovers x^* such that $\|x - x^*\|_1 = O(\|\mu\|_1/d + \|x - x^{(k)}\|_1)$.*

2.5.1 Main Lemmas

Lemma 3. *For any $2k$-sparse u, let $c = \boldsymbol{A}u + \mu$ and $\eta = \|\mu\|_1/d$. Then*

$$\|H_{2k}[u^*(c)] - u\|_1 \le O(\epsilon)\|u\|_1 + O(\eta)$$

The above lemma is in fact a simplification of Theorem 5 in [IR08]. However, [BIR08] presents the following self-contained proof, which is very different from the combinatorial argument used in [IR08].

Proof. For the purpose of analysis, we assume that $|u_1| \geq \ldots \geq |u_n|$. Note that since u is $2k$-sparse, we have $u_{2k+1} = \ldots = u_n = 0$. Let $S = \{1 \ldots 2k\}$. The proof of Lemma 3 relies on the following two lemmas, whose proof is deferred to the next section.

Lemma 4.

$$\|(u^* - u)_S\|_1 = O(\epsilon)\|u\|_1 + O(\eta)$$

Lemma 5. *Let $B \subset \overline{S}$ be a set of coordinates of size at most $2k$. Then*

$$\|u^*_B - u_B\|_1 = \|u^*_B\|_1 = O(\epsilon)\|u\|_1 + O(\eta)$$

Let T be the coordinates of the $2k$ largest in magnitude coefficients of u^*. Then:

$$\begin{aligned}
\|H_{2k}[u^*] - u\|_1 &= \|u^*_T - u\|_1 \\
&= \|(u^* - u)_{S\cap T}\|_1 + \|u^*_{T-S}\|_1 + \|u_{S-T}\|_1 \\
&\leq \|(u^* - u)_{S\cap T}\|_1 + \|u^*_{T-S}\|_1 + \|u^*_{S-T}\|_1 + \|(u^* - u)_{S-T}\|_1 \\
&= \|(u^* - u)_S\|_1 + \|u^*_{T-S}\|_1 + \|u^*_{S-T}\|_1
\end{aligned}$$

To bound $\|u^*_{S-T}\|_1$, observe that for any $i \in S - T$ and $i' \in T - S$, we have $|u^*_i| \leq |u^*_{i'}|$. Since $|T - S| = |S - T|$, it follows that $\|u^*_{S-T}\|_1 \leq \|u^*_{T-S}\|_1$. Hence, for $B = T - S$, it follows from Lemmas 4 and 5 that

$$\|H_{2k}[u^*] - u\|_1 \leq \|(u^* - u)_S\|_1 + 2\|u^*_B\|_1 = O(\epsilon)\|u\|_1 + O(\eta)$$

□

Lemma 6. *For any k-sparse vector x, and any vector x' we have*

$$\|H_k[x'] - x\|_1 \leq 2\|x' - x\|_1$$

Proof. Observe that $H_k[x']$ is the closest k-sparse vector to x. Thus, we have $\|H_k[x'] - x'\|_1 \leq \|x - x'\|_1$. The lemma follows from triangle inequality. □

2.5.2 Proofs of Lemmas 4 and 5

We start from some basic observations.

Decomposition. For the analysis, it is convenient to decompose the vector $\boldsymbol{A}u$ into a sum $v + v''$. The vector v is such that $v_j = u_{i(j)}$ where $i(j)$ is the index i' from $\Gamma(j)$ with the largest value of $|u_{i'}|$.

Fact 1. *Let $v'' = \boldsymbol{A}u - v$. Then $\|v''\|_1 \leq 2\epsilon d\|u\|_1$.*

The proof of this fact is virtually identical to the proof of theorem 1.

It follows that the vector $c = \boldsymbol{A}u + \mu$ can be represented as $c = v + v'$, where $\|v'\|_1 \leq O(\epsilon)d\|u\|_1 + \|\mu\|_1$.

Composing quantiles. In the proof we will compute bounds on the medians of vectors $|c_B|$, where B are subsets of coordinates. In light of the above decomposition, it would make sense to compute separate bounds for medians of $|v_B|$ and $|v'_B|$, and then combine them. Unfortunately, it is in general not true that $\text{median}(u+v) \leq \text{median}(u)+\text{median}(v)$, even for positive vectors u, v. Fortunately, we can overcome this problem by using lower quantiles. Specifically, for any non-negative n-dimensional vector u, and any $\alpha \in (0, 1)$ such that αn is an integer, we define $\text{quant}_\alpha(u)$ to be the αn-th largest element of u. Then we have the following fact.

Fact 2. *For any vectors $u, v \geq 0$ of dimension n divisible by 4, we have*

$$\text{quant}_{1/2}(u + v) \leq \text{quant}_{1/4}(u) + \text{quant}_{1/4}(v)$$

Proof. Let $U = \text{quant}_{1/4}(u)$ and $V = \text{quant}_{1/4}(v)$. There are at most $n/4-1$ elements of u (v, resp.) that are greater than U(V, resp.). Therefore, there are at least $n-2(n/4-1) = n/2+2$ elements of $u+v$ that are not greater than $U+V$, which concludes the proof. □

Telescoping trick. Consider two sequences: $a = a_1, \ldots, a_s$ and $b = b_1, \ldots b_t$ of positive numbers, such that $\{a_1, \ldots, a_s\} \subset \{b_1, \ldots b_t\}$. In addition, we assume $b_1 \geq b_2 \geq \ldots \geq b_t = 0$. The two sequences are related in the following way. For each b_i, define $c(i)$ to be the cardinality of the set $C(i) = \{j : a_j \geq b_i\}$, i.e., the number of times an element from $\{b_1 \ldots b_i\}$ appears in the sequence a; we assume $c(0) = 0$. The following claim states that

if this number is bounded, then the sum of the elements of a is only a fraction of the sum of elements in b.

Claim 1. *Assume that $c(i) \leq \alpha i$ for some $\alpha > 0$. Then $\|a\|_1 \leq \alpha \|b\|_1$.*

Proof. For simplicity of exposition, we assume that all terms in b are distinct. First, observe that for each i, the number of times the value b_i occurs in a is equal to $c(i) - c(i-1)$. Thus

$$\begin{aligned}
\sum_j a_j &= \sum_{i=1}^{t} [c(i) - c(i-1)] b_i \\
&\leq \sum_{i=1}^{t} c(i) b_i - \sum_{i=2}^{t} c(i-1) b_{i-1} + \sum_{i=2}^{t} c(i-1)(b_{i-1} - b_i) \\
&\leq \alpha t b_t + \sum_{i=2}^{t} \alpha (i-1)(b_{i-1} - b_i) \\
&\leq \sum_{i=2}^{t} \alpha (b_{i-1} - b_t) \\
&\leq \alpha \|b\|_1
\end{aligned}$$

□

Proof of Lemma 4

Recall that $u_i^* = \text{median}(v_{\Gamma(i)} + v'_{\Gamma(i)})$. Therefore, we need to bound

$$\begin{aligned}
\|(u^* - u)_S\|_1 &= \sum_{i \in S} |\,\text{median}(v_{\Gamma(i)} + v'_{\Gamma(i)}) - u_i| \\
&= \sum_{i \in S} |\,\text{median}(v_{\Gamma(i)} - u_i^d + v'_{\Gamma(i)})| \\
&\leq \sum_{i \in S} \text{median}(|v_{\Gamma(i)} - u_i^d| + |v'_{\Gamma(i)}|)
\end{aligned}$$

where u_i^d is a vector of dimension d containing as coordinates d copies of u_i. For any $i \in S$, let $w^i = v_{\Gamma(i)} - u_i^d$. Then, it suffices to bound

$$\begin{aligned}
\sum_{i \in S} \text{median}(|w^i| + |v'_{\Gamma(i)}|) \;\leq\; & \sum_{i \in S} \text{quant}_{1/4}(|w^i|) + \\
& \sum_{i \in S} \text{quant}_{1/4}(|v'_{\Gamma(i)}|)
\end{aligned}$$

We bound the second term using the following claim.

Claim 2. *Let P be any set of at most s coordinates. Then*

$$\sum_{i \in P} \text{quant}_{1/4}(|v'_{\Gamma(i)}|) = O(\|v'\|_1/d)$$

Proof. For each v'_j, let $c(j)$ be the number of $i \in P$ having at least $d/4$ neighbors in the set $\{1 \dots j\}$. From the expansion properties of the graph G it follows that $c(j)(d/4 - \epsilon d) \leq j$. Thus $c(j) \leq 8j/d$. Applying Claim 1 finishes the proof. □

Therefore, the contribution of the second term is at most $O(\|v'\|_1/d) = O(\epsilon\|u\|_1)+O(\eta)$.

To bound the first term, we proceed as follows. Recall that we assumed that the entries $|u_1|, |u_2|, \dots$ appear in the non-increasing order. Partition S into the union of S^+ and $S^- = S - S^+$, where $S^+ = \{i \in S : |\Gamma(i) \cap \Gamma(\{1 \dots i-1\})| < d/4\}$. We will bound the first term separately for elements in S^+ and S^-.

Observe that for each $i \in S^+$ the number of non-zero elements in w^i is smaller than $d/4$. Therefore, $\sum_{i \in S^+} \text{quant}_{1/4}(|w^i|) = 0$.

To take care of the elements in S^- we need to bound

$$\sum_{i \in S^-} \text{quant}_{1/4}(|v_{\Gamma(i)} - u^d_i|) \leq \sum_{i \in S^-} \text{quant}_{1/4}(|v_{\Gamma(i)}|) \tag{2.1}$$

For any $r \in S$, consider the set S_r containing indices $i \in S^-$ such that $|\Gamma(i) \cap \Gamma(\{1 \dots r\})| \geq d/4$. Our goal is to show that $|S_r|$ is relatively small, and therefore few elements of the sum in equation 2.1 can be large.

We partition S_r into $S^<_r = S_r \cap \{1 \dots r\}$ and $S^>_r = S_r - S^<_r$. From the expansion properties of G it follows that

$$d(1-\epsilon)(r + |S^>_r|) \leq dr + 3/4 \cdot d|S^>_r|$$

Therefore we have $|S^>_r| \leq \frac{\epsilon}{1/4-\epsilon} r \leq 8\epsilon r$.

To bound $S^<_r$, we observe that from the definition of S^- and the expansion of G it follows that

$$d(1-\epsilon)r \leq d(r - |S^<_r|) + 3/4 \cdot d|S^<_r|$$

Therefore we have $|S^>_r| \leq 4\epsilon r$ and $|S_r| = |S^>_r| + |S^<_r| \leq 12\epsilon r$. That is, for any r, at most $12\epsilon r$ terms in the sum in equation 2.1 can be greater than $|u_r|$. From Claim 1 it follows

that the total value of the sum is at most $O(\epsilon)\|u\|_1$.

Putting all terms together concludes the proof of Lemma 4. □

Proof of Lemma 5

We need to bound

$$\begin{aligned}\|(u^*)_B\|_1 &= \sum_{i\in B} |\operatorname{median}(v_{\Gamma(i)} + v'_{\Gamma(i)})| \\ &\leq \sum_{i\in B} |\operatorname{quant}_{1/4}(v_{\Gamma(i)})| + \sum_{i\in B} |\operatorname{quant}_{1/4}(v'_{\Gamma(i)})|\end{aligned}$$

Using Claim 2 we can bound the second term by $O(\epsilon)\|u\|_1 + O(\eta)$. To bound the first term we proceed as follows. For each $r \in S$ we define $B_r = \{i \in B : |\Gamma(i) \cap \Gamma(\{1 \dots r\})| \geq d/4\}$. From expansion of the graph G and the fact that $B \cap S = \emptyset$ it follows that

$$(1-\epsilon)d(r + |B_r|) \leq dr + 3/4 \cdot d|B_r|$$

It follows that $|B_r| \leq 8\epsilon r$. From Claim 1 we conclude that the first term is bounded by $O(\epsilon)\|u\|_1$. □

We have presented the SMP algorithm and proved that it recovers sparse approximations. Section 2.7.3 shows how the algorithm behaves in practice; while the quality of the approximations is not as good as that obtained with ℓ_1-minimization, SMP is a very fast algorithm, orders of magnitude faster than LP-based methods.

Algorithm 2: SSMP

$x^0 \leftarrow 0$;
for $j = 1 \ldots (T = O(\log(d\|x\|_1/\|\mu\|_1)))$ **do**
  $x^j \leftarrow x^{j-1}$;
  for $step = 1 \ldots (S = (c-1)k)$ **do**
    find a coordinate i and increment z that minimizes $\|\boldsymbol{A}(x^j + ze_i) - b\|_1$;
    $x^j \leftarrow x^j + ze_i$;
  /* From Corollary 3 we have $\|x - x^j\|_1 \leq \|x - x^{j-1}\|_1/4 + C\|\mu\|_1/d$ */
  $x^j \leftarrow H_k[x^j]$; /* From Lemma 6 we have $\|x - x^j\|_1 \leq \|x - x^{j-1}\|_1/2 + 2C\|\mu\|_1/d$ */

Figure 2-3: The Sequential Sparse Matching Pursuit algorithm

2.6 Sequential Sparse Matching Pursuit

The *Sequential Sparse Matching Pursuit* algorithm is presented in figure 2-3. It is similar to SMP, except that the estimate is improved one coordinate at a time. As with SMP, consider a k-sparse vector x and a noise vector μ. The $m \times n$ measurement matrix $\boldsymbol{A}$ is the adjacency matrix of an $((c+1)k, d, \epsilon/2)$-unbalanced expander G; note that this implies that $\boldsymbol{A}/d$ has the $\mathrm{RIP}_{1,(c+1)k,\epsilon}$ property. The algorithm consists of two nested iterations: the inner one and the outer one. The goal of the inner iteration is to reduce the residual error $\|\boldsymbol{A}x^j - b\|_1$ by a constant factor, unless the error is already smaller than $c_l\|\mu\|_1$. This is done in $S = (c-1)k$ update steps, where each step reduces the residual error by a factor $(1 - c_u/k)$ or better. In each update step, the algorithm finds a coordinate i and an increment z such that $\|\boldsymbol{A}(x^j + ze_i) - b\|_1$ is minimized. For a given i, the value of z that minimizes the expression is equal to the median of the vector $(\boldsymbol{A}x^j - b)_N$, where $N = \Gamma(i)$ is the set of neighbors of i in G.

In each step of the outer loop, the inner loop is executed, and then the vector x^j is re-sparsified by keeping the largest (in the absolute value) k coordinates of x^j. As in SMP, this step approximately preserves the error of x^j.

The algorithm can be efficiently implemented in the following way. For each i, we maintain the optimum increment $z = z_i$, together with the resulting change D_i to the L1 error norm. The D_i's are stored in a priority queue (e.g. in a heap), which enables finding the largest value of D_i, as well as update the value of each D_i in time $O(\log n)$. When a value of some (say the i-th) coordinate of x^j is modified, this affects the values of d entries l of the vector $Ax^j - b$. In turn, each entry l can affect the values $D_{i'}$ of all $O(dn/m)$

neighbors i' of l. For each such i' we will need to recompute the median in $O(d)$ time, and update the heap. Thus, each coordinate update takes $O(d^2n/m(d+\log n))$ time. Therefore, the total running time of the algorithm is at most $O(\log(d\|x\|_1/\|\mu\|_1) \cdot d^2nk/m(d+\log n))$, which simplifies to $O(\log(d\|x\|_1/\|\mu\|_1) \cdot dn(d+\log n))$ since $m = \Theta(kd)$.

A formal statement of the result follows.

Theorem 5. *For any k-sparse signal x and noise vector μ and given $b = \boldsymbol{A}x + \mu$, SSMP recovers x^* such that $\|x - x^*\|_1 = O(\|\mu\|_1/d)$.*

As with SMP, we can generalize this theorem for all vectors x: since $\boldsymbol{A}x+\mu = \boldsymbol{A}x^{(k)} + [\mu + \boldsymbol{A}(x - x^{(k)})]$, and $\|\boldsymbol{A}(x - x^{(k)})\|_1 \leq d\|x - x^{(k)}\|_1$, we have the following corollary:

Corollary 2. *For any parameter k, any vector x and noise vector μ, given $b = Ax + \mu$, the SSMP algorithm recovers x^* such that $\|x - x^*\|_1 = O(\|\mu\|_1/d + \|x - x^{(k)}\|_1)$.*

The correctness proof is outlined in the remarks in the algorithm description (figure 2-3). The key part of the argument is to show that if the residual error is at least $c_l\|\mu\|_1$, then each update step reduces it by a factor $(1 - c_u/k)$ or better. Specifically, we show the following lemma.

Lemma 7. *If x' is ck-sparse and $\|\boldsymbol{A}x' - b\|_1 \geq c_l\|\mu\|_1$, then there exists an index i such that*

$$\|\boldsymbol{A}(x' - e_ix'_i + e_ix_i) - b\|_1 \leq (1 - c_u/k)\|\boldsymbol{A}x' - b\|_1$$

To relate the improvement in the residual error to the reduction in approximation error, we use the following claim:

Claim 3.

$$d(1-\epsilon)\|x' - x\|_1 \leq \|\boldsymbol{A}x' - b\|_1 + \|\mu\|_1 \leq d\|x - x'\|_1 + 2\|mu\|_1$$

Proof. From the $\text{RIP}_{1,(c+1)k,\epsilon}$ property of matrix $\boldsymbol{A}/d$:

$$\begin{aligned} d(1-\epsilon)\|x' - x\|_1 &\leq \|\boldsymbol{A}x' - \boldsymbol{A}x\|_1 \leq \|\boldsymbol{A}x' - (\boldsymbol{A}x + \mu) + \mu\|_1 \\ &\leq \|\boldsymbol{A}x' - b\|_1 + \|\mu\|_1 = \|\boldsymbol{A}(x' - x) - \mu\|_1 + \|\mu_1\| \\ &\leq d\|x - x'\|_1 + 2\|\mu\|_1 \end{aligned}$$

□

Using this claim, we obtain the following corollary to Lemma 7:

Corollary 3. *There exist constants c and C such that, after the j-th inner loop, we have*

$$\|x - x^j\|_1 \leq \|x - x^{j-1}\|_1/4 + C\|\mu\|_1/d$$

The sparsification step is handled by Lemma 6 which states that for any k-sparse vector x, and any vector x' we have

$$\|H_k[x'] - x\|_1 \leq 2\|x' - x\|_1$$

Theorem 5 now follows from the above two lemmas.

2.6.1 Proof of Lemma 7

Let $\Delta = x' - x$. Since $b = \boldsymbol{A}x + \mu$, the thesis of the theorem can be rewritten as

$$\|\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu\|_1 \leq (1 - c_u/k)\|\boldsymbol{A}\Delta - \mu\|_1$$

First, observe that, by triangle inequality, the assumption $\|\boldsymbol{A}x' - b\|_1 \geq c_l\|\mu\|_1$ implies that $\|\boldsymbol{A}x' - \boldsymbol{A}x\|_1 \geq (c_l - 1)\|\mu\|_1$. By the $\mathrm{RIP}_{1,(c+1)k,\epsilon}$ property of $\boldsymbol{A}/d$ this implies

$$\|\Delta\|_1 = \|x' - x\|_1 \quad \geq \quad (1 - \epsilon)(c_l - 1)\|\mu\|_1/d \tag{2.2}$$

Let $T = \mathrm{supp}(\Delta)$. Clearly, $|T| \leq (c+1)k$. Consider any $i \in T$, and let $N_i = \Gamma(i)$ be the set of neighbors of i in the graph G. The key idea in the proof is to split the neighborhood set N_i into a union of N_i^+ and N_i^-. This is done by proceeding as in the proof of theorem 1. First, w.l.o.g., we reorder the coordinates so that $|\Delta_1| \geq |\Delta_2| \geq \ldots \geq |\Delta_n|$. Then, we enumerate the edges (i, j) of the graph G in lexicographic order. If (i, j) is the first edge from any vertex to the vertex j, then j is included in N_i^+, otherwise it is included in N_i^-. Note that the sets N_i^+ are pairwise disjoint.

From the expansion property of G, it follows that for any prefix of p first vertices i, we have $\sum_{i=1}^{p} |N_i^-| \leq \epsilon dp$. This in turn implies several other properties. In particular, for a constant $c_p > 1$, define $T^+ \subset T$ to contain all indices i such that $|N_i^-| \leq c_p d\epsilon$. The following claim states that the coordinates in T^+ contain most of the "L1 mass" of Δ.

Claim 4.

$$\sum_{i \in T^+} |\Delta_i| \geq (1 - 1/c_p)\|\Delta\|_1$$

Proof. Let $T^- = T \setminus T^+$. Consider all the indices $u_1, u_2, \ldots$ in T^- so that $u_1 < u_2 < \ldots$; by definition $|N^-_{u_i}| > c_p d\epsilon$ for all i. For any $k \geq 1$ consider the k-th index u_k. Then

$$kc_p d\epsilon < \sum_{j=1}^{k} |N^-_{u_j}| \leq \sum_{i=1}^{u_k} |N^-_i| \leq u_k d\epsilon$$

It follows that $u_k > kc_p$. Thus there are at least $k(c_p - 1)$ indices in $T^+ \cap \{1 \ldots u_k\}$ for all k. This allows us to partition T^+ in the following way: for any k let S_k be the set containing the smallest $c_p - 1$ elements of $(T^+ \cap \{1 \ldots u_k\}) \setminus \left(\bigcup_{i=1}^{k-1} S_i\right)$.

Notice that sets S_k are by construction disjoint. For any index u_k in T^-, we have a set of $c_p - 1$ unique indices in T^+, all of which are smaller than u_k; hence for any $v \in S_k$, $|\Delta_v| \geq |\Delta_{u_k}|$. Since

$$\begin{aligned} \|\Delta\|_1 &\geq \sum_{u_k \in T^-} \left(|\Delta_{u_k}| + \sum_{v \in S_k} |\Delta_v| \right) \\ &\geq \sum_{u_k \in T^-} c_p |\Delta_{u_k}| \end{aligned}$$

it follows that $\sum_{i \in T^+} |\Delta_i| \geq (1 - 1/c_p)\|\Delta\|_1$. □

The above claim implies

$$\sum_{i \in T^+} \|(\boldsymbol{A}\Delta_i e_i)_{N^+_i}\|_1 \;\geq\; d \sum_{i \in T^+} \Delta_i |N^+_i|/d \geq (1 - 1/c_p)(1 - c_p\epsilon)\|\Delta\|_1 \tag{2.3}$$

The next claim concerns the amount of the L1 mass contributed to a coordinate j of the vector $\boldsymbol{A}\Delta$ by the coordinates $i \in T$ such that $j \in N^-_i$. We can think about this mass as "noise" contributed by the coordinates of Δ itself (as opposed to μ). Again, from the edge enumeration process, it follows that this contribution is low overall. Specifically:

Claim 5.

$$\sum_{i \in T^+} \|(\boldsymbol{A}(\Delta - \Delta_i e_i))_{N^+_i}\|_1 \leq d\epsilon \|\Delta\|_1$$

Proof. Since sets N^+_i are disjoint

$$\begin{aligned} \sum_{i \in T^+} \|(\boldsymbol{A}(\Delta - \Delta_i e_i))_{N^+_i}\|_1 &\leq \sum_{i \in T^+} \sum_{j \in N^+_i} \sum_{i' | j \in N^-_{i'}} |\Delta_{i'}| \\ &\leq \sum_{j} \sum_{i | j \in N^-_i} |\Delta_i| \;\leq\; \sum_{i=1}^{n} |\Delta_i| \cdot |N^-_i| \end{aligned}$$

Define the value $s_p = pd\epsilon - \sum_{i=1}^{p} |N_i^-|$. We know that all $s_p \geq 0$.

$$\begin{aligned}
\sum_{i=1}^{n} |\Delta_i| \cdot |N_i^-| &= d\epsilon|\Delta_1| - s_1|\Delta_1| + \sum_{i=2}^{n} |\Delta_i| \cdot |N_i^-| \\
&\leq d\epsilon|\Delta_1| - s_1|\Delta_2| + \sum_{i=2}^{n} |\Delta_i| \cdot |N_i^-| \\
&\leq d\epsilon(|\Delta_1| + |\Delta_2|) - s_2|\Delta_2| + \sum_{i=3}^{n} |\Delta_i| \cdot |N_i^-| \\
&\leq d\epsilon(|\Delta_1| + |\Delta_2|) - s_2|\Delta_3| + \sum_{i=3}^{n} |\Delta_i| \cdot |N_i^-| \\
&\leq \cdots \leq d\epsilon \sum_{i=1}^{p} |\Delta_i| - s_p|\Delta_p| + \sum_{i=p+1}^{n} |\Delta_i| \cdot |N_i^-| \leq \cdots \\
&\leq d\epsilon \sum_{i=1}^{n} |\Delta_i| - s_n|\Delta_n| \leq d\epsilon\|\Delta\|_1
\end{aligned}$$

□

Now we proceed with the main part of the proof. Define

$$\text{gain}_i = \|\boldsymbol{A}\Delta - \mu\|_1 - \|\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu\|_1$$

Observe that, equivalently, we have

$$\text{gain}_i = \|(\boldsymbol{A}\Delta - \mu)_{N_i}\|_1 - \|(\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu)_{N_i}\|_1$$

Therefore

$$\begin{aligned}
\text{gain}_i &= \|(\boldsymbol{A}\Delta - \mu)_{N_i}\|_1 - \|(\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu)_{N_i}\|_1 \\
&= \|(\boldsymbol{A}\Delta - \mu)_{N_i^+}\|_1 - \|(\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu)_{N_i^+}\|_1 + \\
&\quad + \left[\|(\boldsymbol{A}\Delta - \mu)_{N_i^-}\|_1 - \|(\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu)_{N_i^-}\|_1\right] \\
&\geq \|(\boldsymbol{A}\Delta - \mu)_{N_i^+}\|_1 - \|(\boldsymbol{A}(\Delta - e_i\Delta_i) - \mu)_{N_i^+}\|_1 - \|(\boldsymbol{A}e_i\Delta_i)_{N_i^-}\|_1 \\
&\geq \|(\boldsymbol{A}\Delta)_{N_i^+}\|_1 - \|(\boldsymbol{A}(\Delta - e_i\Delta_i))_{N_i^+}\|_1 - 2\|\mu_{N_i^+}\|_1 - \|(\boldsymbol{A}e_i\Delta_i)_{N_i^-}\|_1 \\
&\geq \|(\boldsymbol{A}e_i\Delta_i)_{N_i^+}\|_1 - 2\|(\boldsymbol{A}(\Delta - e_i\Delta_i))_{N_i^+}\|_1 - 2\|\mu_{N_i^+}\|_1 - \|(\boldsymbol{A}e_i\Delta_i)_{N_i^-}\|_1 \\
&\geq \left(1 - \frac{c_p\epsilon}{1 - c_p\epsilon}\right)\|(\boldsymbol{A}e_i\Delta_i)_{N_i^+}\|_1 - 2\|(\boldsymbol{A}(\Delta - e_i\Delta_i))_{N_i^+}\|_1 - 2\|\mu_{N_i^+}\|_1
\end{aligned}$$

Aggregating over all $i \in T^+$, we have

$$\sum_{i \in T^+} \text{gain}_i \geq \sum_{i \in T^+} \left(1 - \frac{c_p \epsilon}{1 - c_p \epsilon}\right) \|(\boldsymbol{A} e_i \Delta_i)_{N_i^+}\|_1 - 2\|(\boldsymbol{A}(\Delta - e_i \Delta_i))_{N_i^+}\|_1 - 2\|\mu_{N_i^+}\|_1$$

From Claims 4 and 5, and the fact that the sets N_i^+ are pairwise disjoint, we have

$$\begin{aligned}
\sum_{i \in T^+} \text{gain}_i &\geq d\left(1 - \frac{c_p \epsilon}{1 - c_p \epsilon}\right)(1 - 1/c_p)(1 - c_p \epsilon)\|\Delta\|_1 - 2d\epsilon\|\Delta\|_1 - 2\|\mu\|_1 \\
&= d\left[\left(1 - \frac{c_p \epsilon}{1 - c_p \epsilon}\right)(1 - 1/c_p)(1 - c_p \epsilon) - 2\epsilon - 2/((1 - \epsilon)(c_l - 1))\right]\|\Delta\|_1 \\
&= dC\|\Delta\|_1
\end{aligned}$$

where C is a positive constant as long as ϵ is small enough; we have used Equation (2.2) to relate $\|\Delta\|_1$ and $\|\mu\|_1$. It follows that there exists a coordinate $i \in T^+$ such that

$$\text{gain}_i \geq \frac{dC}{|T^+|}\|\Delta\|_1 \geq \frac{dC}{(c+1)k}\|\Delta\|_1$$

At the same time

$$d\|\Delta\|_1 \geq \|\boldsymbol{A}(x' - x)\|_1 \geq \|\boldsymbol{A}x' - \boldsymbol{A}x - \mu\|_1 - \|\mu\|_1 \geq \|\boldsymbol{A}x' - b\|_1 - \frac{d\|\Delta\|_1}{(1 - \epsilon)(c_l - 1)}$$

Therefore

$$\text{gain}_i \geq \frac{C}{(c+1)k}\|\boldsymbol{A}x' - b\|_1 / \left(1 + \frac{1}{(1 - \epsilon)(c_l - 1)}\right)$$

and we are done. □

We have presented the SSMP algorithm and proved that it recovers sparse approximations. The experiments in section 2.7.4 show that in practice, SSMP results in considerably better approximations than SMP at the cost of increased recovery time.

2.7 Experiments

In this section we show how the presented algorithms behave in practice. Section 2.7.1 describes the testing method; section 2.7.2 shows that in practice sparse matrices behave very similar to dense Gaussian matrices when used with the ℓ_1-minimization program. Sections 2.7.3 and 2.7.4 present the behavior of the SMP and SSMP algorithms. Finally, section 2.7.5 shows comparisons between the three methods.

2.7.1 Methodology

We generate random sparse matrices one column at a time, randomly choosing k positions from the set $\{1 \dots m\}$. If, for a column, the d values contain duplicates, we repeat the process for that column.

Our experiments fall into two categories: exact recovery of sparse vectors, and approximate recovery of non-sparse vectors.

Sparse signals

For sparse signals we use synthetic data; the presented experiments use signals generated by randomly choosing k coordinates and setting each to either -1 or $+1$. For many experiments we also tried signals with different values for the k peaks (e.g. Gaussians) but the results were very similar.

We seek perfect recovery of sparse signals; thus an experiment is either a success if the vector was recovered correctly or a failure otherwise. We can characterise the behavior of an algorithm using a sparse recovery plot: a signal length n is fixed, and the sparsity of the signal k and the number of measurements m are swept over sensible ranges. For each set of parameters m and k, the probability of correct recovery is estimated by randomly choosing a measurement matrix and repeatedly attempting recovery of different sparse vectors; the fraction of successful recoveries is noted. We can generate similar plots by keeping the sparsity k fixed and varying m and n. For all plots, we show the resolution (number of different values for the two variable parameters) as well as the number of trials per data point.

Approximately sparse signals - images

We also present experiments with recovery of natural images in the spirit of [CRT06, BI08, APT09, BI09]. For a given image, we perform measurements on the vector containing the image's Daubechies-2 wavelet coefficients and use our algorithms on the sketch to reconstruct the image.

We use two 256×256 grayscale images: the boat image in [CRT06, BI08] and the peppers image in [APT09]. We use the Daubechies-2 wavelet basis (as in [APT09]). See figure 1-2 for the peppers image and the usefulness of sparse approximations to the `db2` wavelet coefficients.

We evaluate the quality of each recovery by computing the ℓ_1 norm of the approximation error in the wavelet basis as well as the peak signal-to-noise ratio (PSNR). For two $u \times v$ monochrome images $I(x, y)$ and $J(x, y)$, with $I(x, y), J(x, y) \in [0, 1]$, the peak signal-to-noise ratio is

$$PSNR = -10 \log_{10} \left(\frac{1}{uv} \sum_{x=1}^{u} \sum_{y=1}^{v} (I(x, y) - J(x, y))^2 \right)$$

In our case, if w is the wavelet coefficient vector and w^* is the recovered vector, the PSNR is equivalent to

$$PSNR = -10 \log_{10} \left(\frac{1}{uv} (\|w - w^*\|_2)^2 \right)$$

This is because the ℓ_2 approximation error is the same in both the wavelet and the original (image) bases. We use the PSNR in comparisons because it is the generally preferred metric to compare image compression quality; however, since our algorithms provide ℓ_1/ℓ_1 sparse recovery guarantees, we also show the ℓ_1 approximation error $\|w - w^*\|_1$ for many of the image experiments.

2.7.2 ℓ_1-minimization and sparse matrices

The result in section 2.4 was that sparse matrices can be used with the LP decoder (P1). It is thus natural to attempt to compare them experimentally with dense random matrices. This section shows that in practice, sparse matrices perform very similarly to dense random matrices such as Gaussians, both in terms of number of measurements required and in terms of the approximation errors. Note that additional sparse and Gaussian matrix experiments can be found in [BI08].

All the experiments in this section are performed with LP reconstruction via (P1), using the ℓ_1-Magic package for Matlab (see [CR05]). The basic operations performed by this interior point method are vector multiplications with the measurement matrix $\boldsymbol{A}$ and its transpose $\boldsymbol{A}^T$; since multiplication time is in general proportional to the sparsity of the matrix, recovery is an order of magnitude faster when sparse matrices are used.

Sparse signals

For exact recovery of sparse signals, we show sparse recovery plots for Gaussian matrices and sparse matrices. Figure 2-4 shows how Gaussian and sparse matrices with $d = 8$ behave in practice when the input signal is sparse (has k peaks of values ± 1). We can see that the number of measurements required by the two types of matrices is very similar for all n, k pairs.

In figure 2-5 (plot originally in [BGI+08]), we overlay our experimental results for sparse matrices with the analytic results for Gaussian matrices presented in [DT06]. The figure shows through level curves the probability of correct signal recovery with signals $x \in \{-1, 0, 1\}^n$ and positive signals $x \in \{0, 1\}^n$. The probabilities were estimated by partitioning the domain into 40×40 data points and performing 50 independent trials for each data point, using random sparse matrices with $d = 8$. The thick curve shows the threshold for correct recovery with Gaussian matrices showed in [DT06]. The empirical behavior for binary sparse matrices is thus consistent with the analytic behavior for Gaussian random matrices.

For the purpose of comparing LP decoding with other sparse matrix algorithms like SMP and SSMP, figure 2-6 shows a plot similar to the top-left plot in figure 2-4 but with

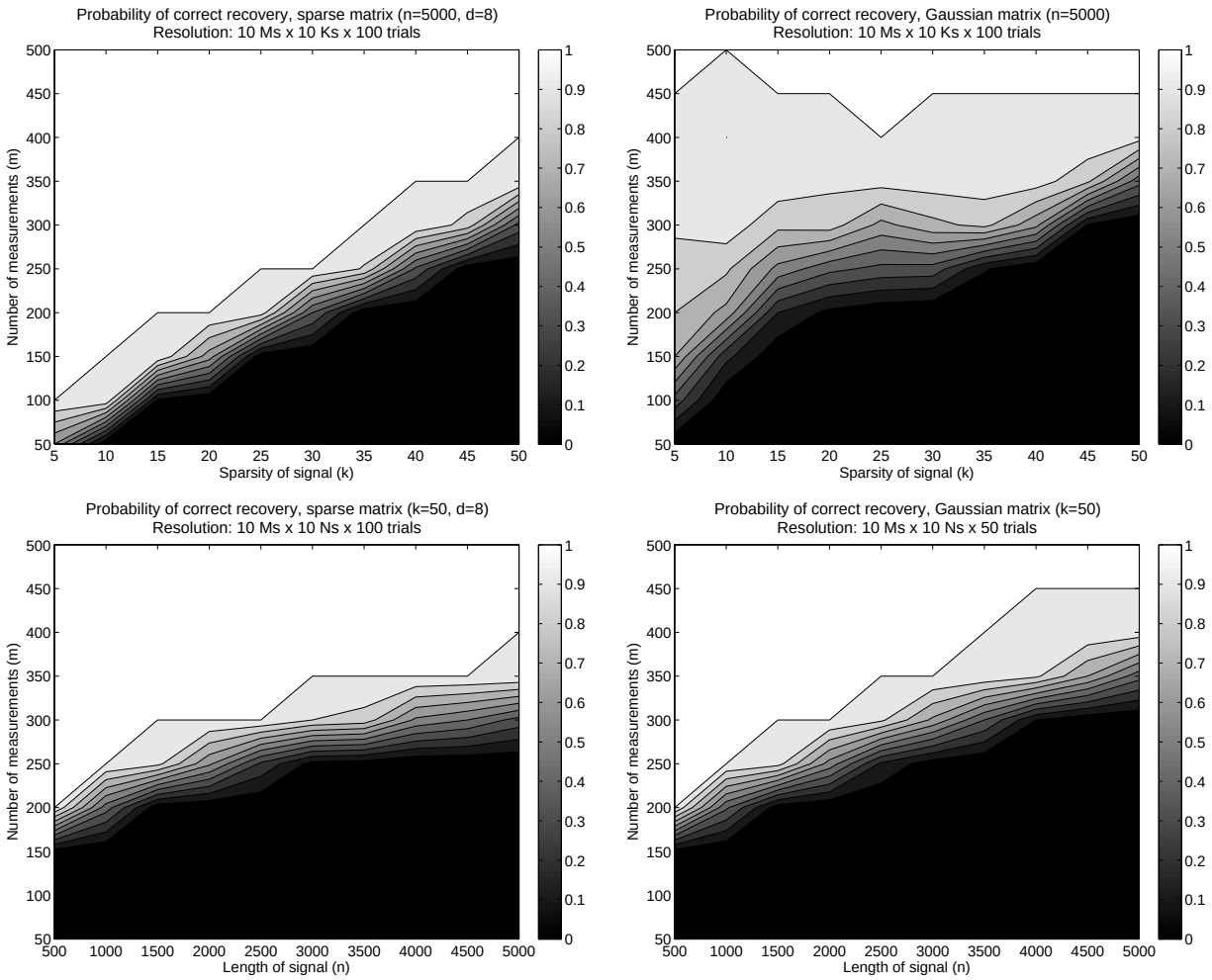


Figure 2-4: Sparse vector recovery experiments with sparse (left) and Gaussian (right) matrices. The top plots are for constant signal length n and varying sparsity k; the bottom plots are for constant sparsity k and varying length n.

larger signal size n. A similar plot for Gaussians was not generated because the recovery using dense matrices becomes too slow for larger signal sizes.

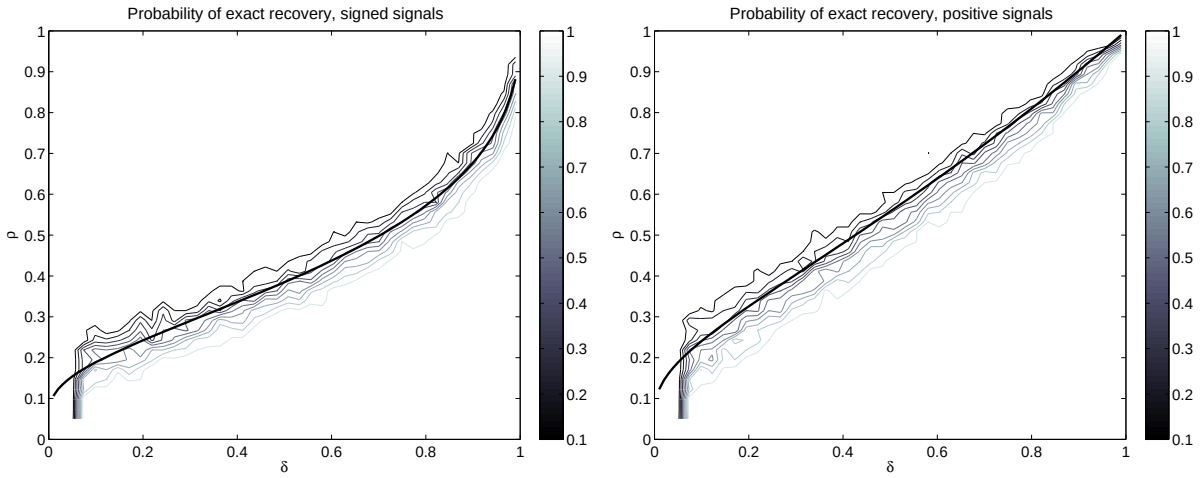


Figure 2-5: Sparse matrix recovery experiments: probability of correct signal recovery of a random k-sparse signal $x \in \{-1, 0, 1\}^n$ (left) and $x \in \{0, 1\}^n$ (right) as a function of $k = \rho m$ and $m = \delta n$, for $n = 200$. The thick curve is the threshold for correct recovery with Gaussian matrices (see [DT06])

.

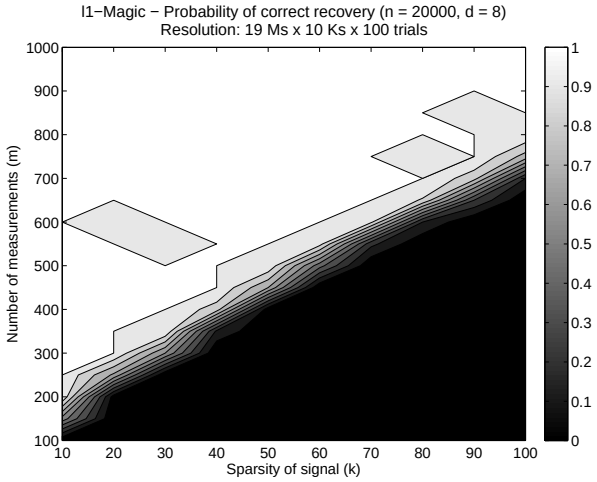


Figure 2-6: Sparse experiments with LP decoding using sparse matrices ($d = 8$) with constant signal length $n = 20000$

Image recovery

Gaussian matrices are impractical for image recovery experiments because of the large space and multiplication time requirements. Instead, we use a heuristic construction - the *real-valued scrambled Fourier ensemble*; it is obtained from the Fourier transform matrix by randomly permuting the columns, selecting a random subset of rows, and separating the real and complex values of each element into two real values (see [CRT06]). In practice, with good probability for a given signal, the scrambled sine and cosine functions in the ensemble rows are effectively similar to Gaussian white noise; the ensemble requires only $O(n)$ space and $O(n \log n)$ multiply time. Experiments with smaller signals in [BI08] show that the ensemble behaves almost as good as real Gaussian matrices.

We compare the approximation error of sparse matrices and scrambled Fourier ensembles on the boat and peppers images. Figure 2-7 shows the resulting ℓ_1 error in the wavelet basis (left side, logarithmic plot) as well as the peak signal-to-noise ratio (right side) with varying number of measurements. The sparsity used was $d = 8$, but other sparsities like $d = 4$ or $d = 16$ yield very similar results.

While in theory the method of decoding via ℓ_1-minimization has no parameters, in practice linear programming algorithms do. In the case of ℓ_1-Magic, the program iteratively improves the solution until either it detects that the optimum is found or until the maximum number of iterations is reached. For non-sparse signals like images, the latter case is usual in practice; thus the maximum number of iterations is a relevant parameter. Figure 2-8 shows how changing the maximum number of iterations affects the quality of the recovery. The decoding time is of course proportional to the number of iterations. Note that each run is with a different random matrix which causes small random variations in the results. Other LP experiments shown in this section were performed with the default setting of 50 iterations.

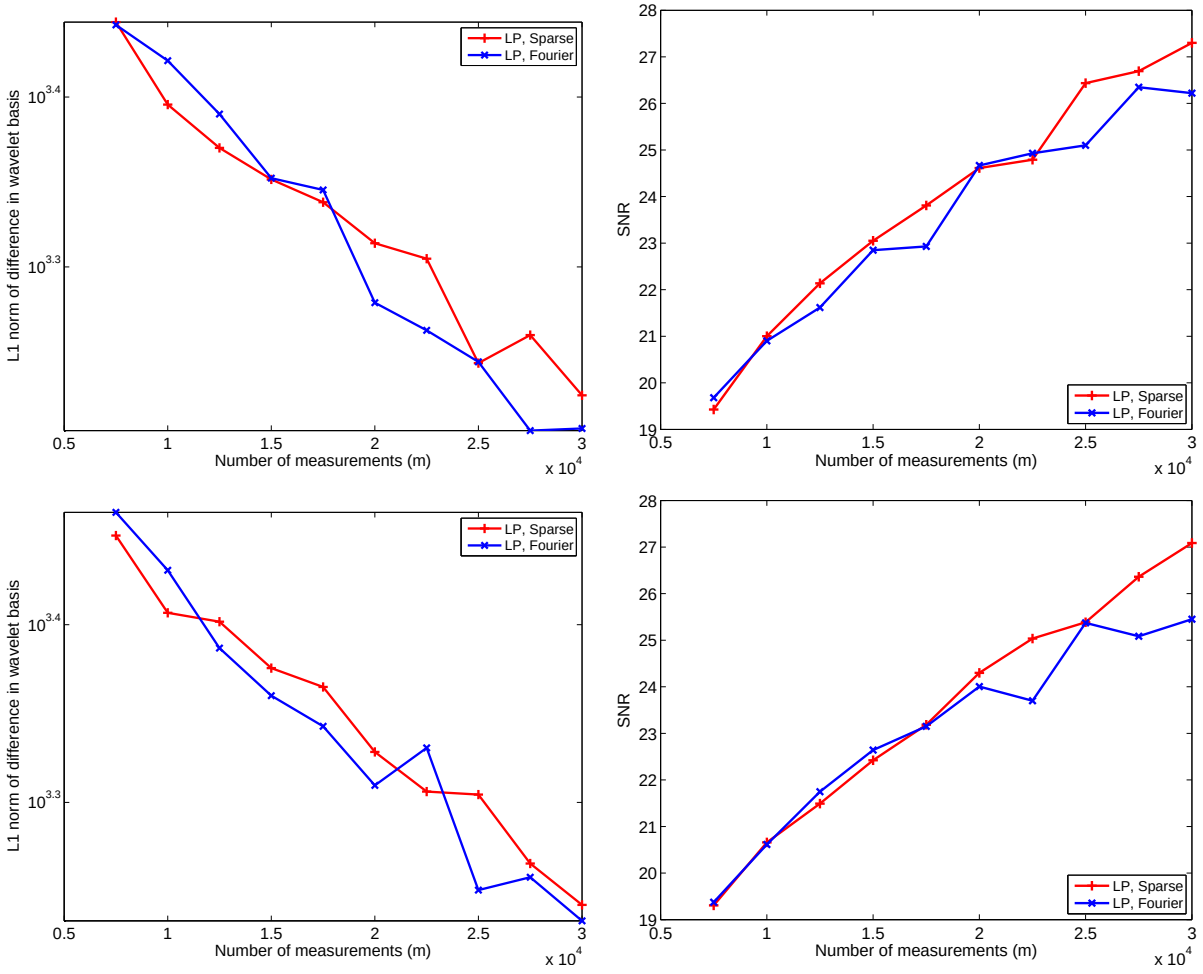


Figure 2-7: Recovery quality with LP on peppers (top) and boat (bottom) images ($d = 8$, $\xi = 0.6$)

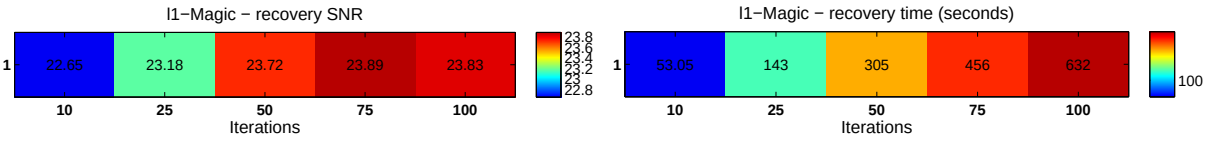


Figure 2-8: Changing the number of iterations in ℓ_1-Magic (peppers image, $m = 17000$, $d = 8$)

2.7.3 SMP

Sparse signals

Figure 2-9 shows sparse recovery experiments with SMP. The left-side plot is obtained by keeping the signal length n fixed and varying the sparsity k; the right-side plot keeps k fixed and varies n. SMP is run with $T = 10$ iterations. Increasing T beyond this value didn't result in a significant improvement. An interesting thing to note is that SMP works as well even when the recovery sparsity is higher than the actual sparsity of the signal, i.e. the k in SMP (figure 2-2) can be higher than the actual k by some factor, with very similar results.

Image recovery

In practice, the SMP algorithm diverges for non-sparse signals if the parameters (most notably the sparsity k and number of measurements m) fall outside the theoretically guaranteed region. The algorithm can be forced to converge by limiting the size of the update vector u^j. The modified algorithm, shown in figure 2-10, performs much better in practice for non-sparse signals like images.

Figure 2-11 shows the recovery quality of SMP on the two images with varying sketch length m. The top plots are for the peppers image, the bottom plots are for the boat image. The left-side plots show the ℓ_1 error on a logarithmic axis. The right-side plots

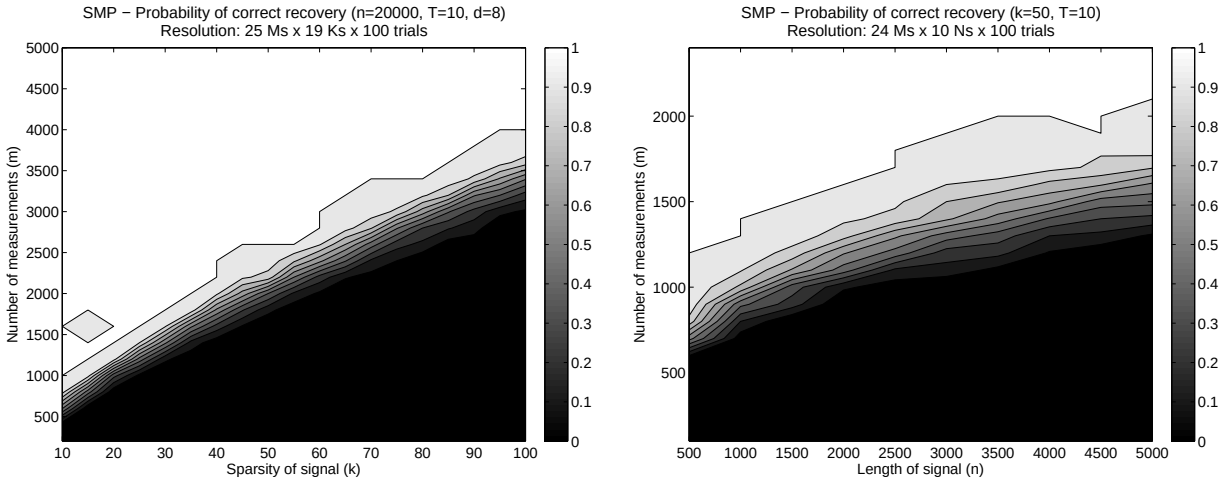


Figure 2-9: Sparse vector recovery experiments with SMP ($d = 8$)

Algorithm 3: SMP-controlled

$x^0 \leftarrow 0$;
for $j = 1 \dots T$ **do**
  $c \leftarrow b - \boldsymbol{A}x^{j-1}$;
  foreach $i \in \{1 \dots n\}$ **do**
    $u_i^* \leftarrow \text{median}(c_{\Gamma(i)})$;
  $u^j \leftarrow H_{2k}[u^*]$;
  if $j > 1$ *and* $\xi\|x^{j-1}\|_1/\|u^j\|_1 < 1$ **then**
    $u^j \leftarrow u^j \cdot \xi\|x^{j-1}\|_1/\|u^j\|_1$; `/* Convergence control */`
  $x^j \leftarrow x^{j-1} + u^j$;
  $x^j \leftarrow H_k[x^j]$;

Figure 2-10: The Sparse Matching Pursuit algorithm, with convergence control (parameter ξ)

show the peak signal to noise ratio. The recovery sparsities k were chosen as a fraction of m. The two best choices - as determined through experimentation - are shown: one choice equal to 2.5% of m is better for lower m values, the other choice equal to 7.5% of m is better for higher values of m. We also show two choices of T since the speed of recovery depends heavily on T (see figures 2-16 and 2-17 for recovery time plots).

Figure 2-12 attempts to show how the three parameters (number of iterations T, recovery sparsity k, and convergence control ξ) affect the quality of the recovered image[7]. For each plot, one parameter is fixed to a sensible value while the other two parameters are varied over a range. The tests are run on the peppers image, with $m = 17000$ measurements (chosen so that the SNRs can be compared to those for the same image in [APT09]). As expected, the recovery quality grows with more iterations (up to a certain point). Convergence control parameter ξ values in the range $0.5 - 0.6$ are suitable for all instances. The best choice for the recovery sparsity k is harder to predict: the recovery quality depends heavily on it, and the correct choice varies with the number of of measurements m.

The last plot of figure 2-12 shows the running time of the recovery algorithm, which depends entirely on the total number of iterations T. The algorithm is very fast, requiring a few seconds or even fractions of a second for one image recovery.

Note that we have used $d = 8$ in all experiments but other choices such as $d = 16$ yield similar results in terms of quality.

[7]Note that each instance is with a different random matrix so small variations exist

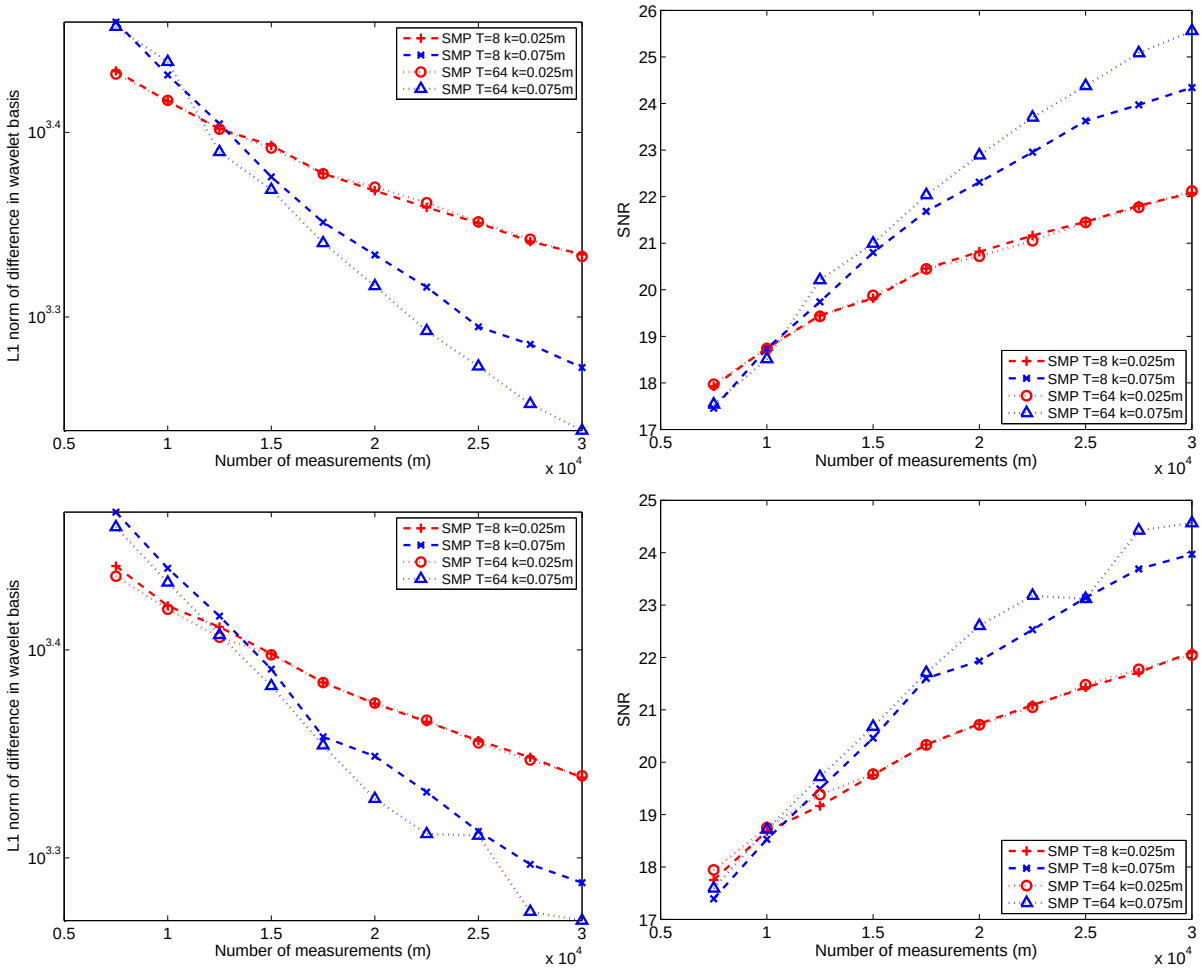


Figure 2-11: Recovery quality of SMP on peppers (top) and boat (bottom) images ($d = 8$, $\xi = 0.6$)

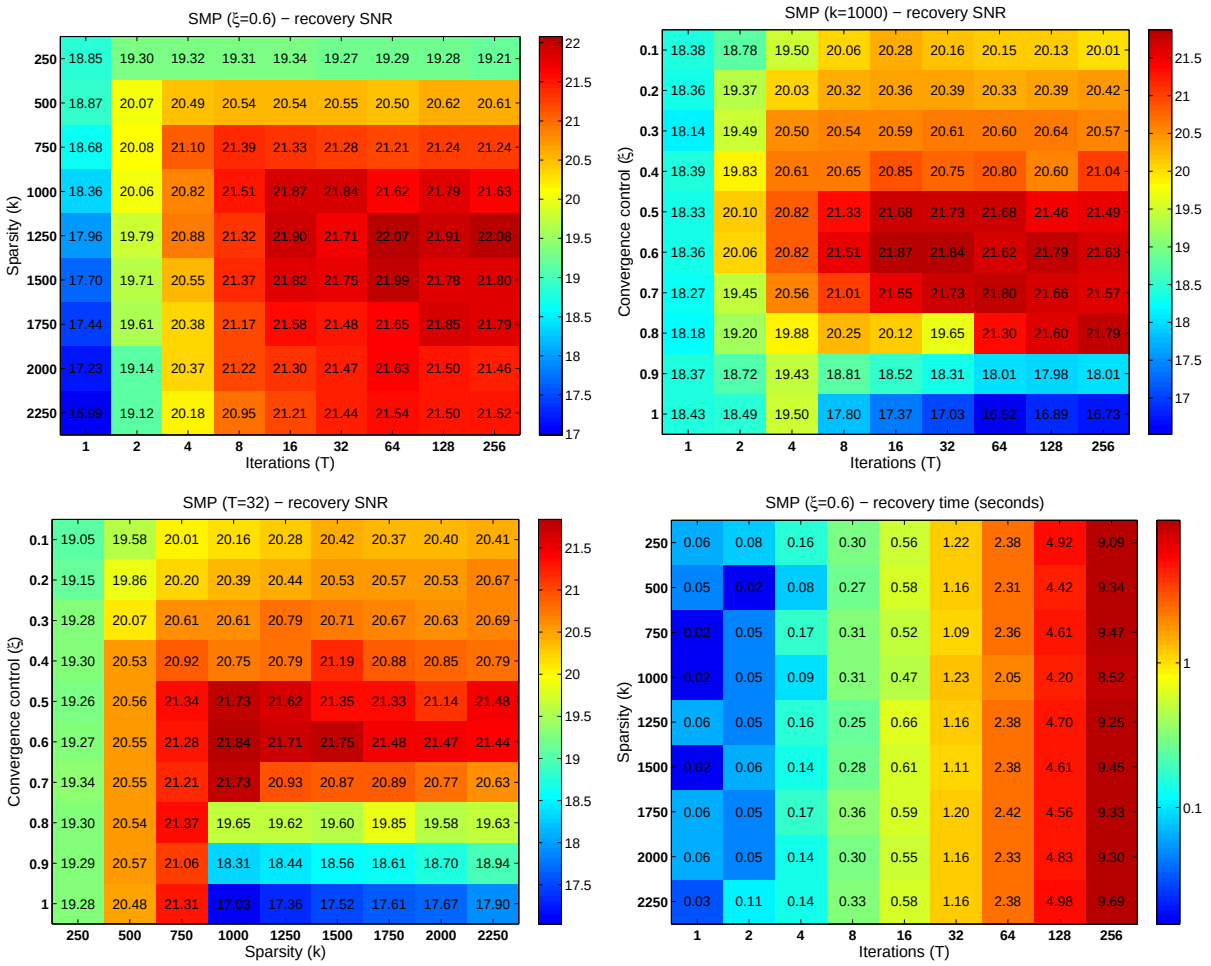


Figure 2-12: Experimental characterisation of SMP (peppers image, $m = 17000$, $d = 8$)

2.7.4 SSMP

Sparse signals

Figure 2-13 shows sparse recovery experiments with SSMP. The left-side plot is obtained by keeping the signal length n fixed and varying the sparsity k; the right-side plot keeps k fixed and varies n. SSMP is run with $S = 4k$ inner steps and only $T = 1$ iteration. For sparse signals more iterations are not needed; even the sparsification step $x^j \leftarrow H_k[x^j]$ (see figure 2-3) - and thus parameter k - can be removed with similar results.

Image recovery

Figure 2-14 shows the recovery quality of SSMP on the two images with varying sketch length m. The top plots are for the peppers image, the bottom plots are for the boat image. The left-side plots show the ℓ_1 error on a logarithmic axis. The right-side plots show the peak signal to noise ratio. The recovery sparsities k were chosen as a fraction of m. Three choices are shown for k: 5% of m, 10% of m, and 15% of m. Each choice is optimal for a certain range of m values. For each choice of k, two settings for S, T are used, a faster instance with $S = 16000$, $T = 16$ and a slower, better quality instance with $S = 32000$, $T = 64$. Note that figures 2-16 and 2-17 show recovery times for SSMP as well as the other methods presented.

Similarly to the SMP plot, figure 2-15 shows how varying two parameters affects the

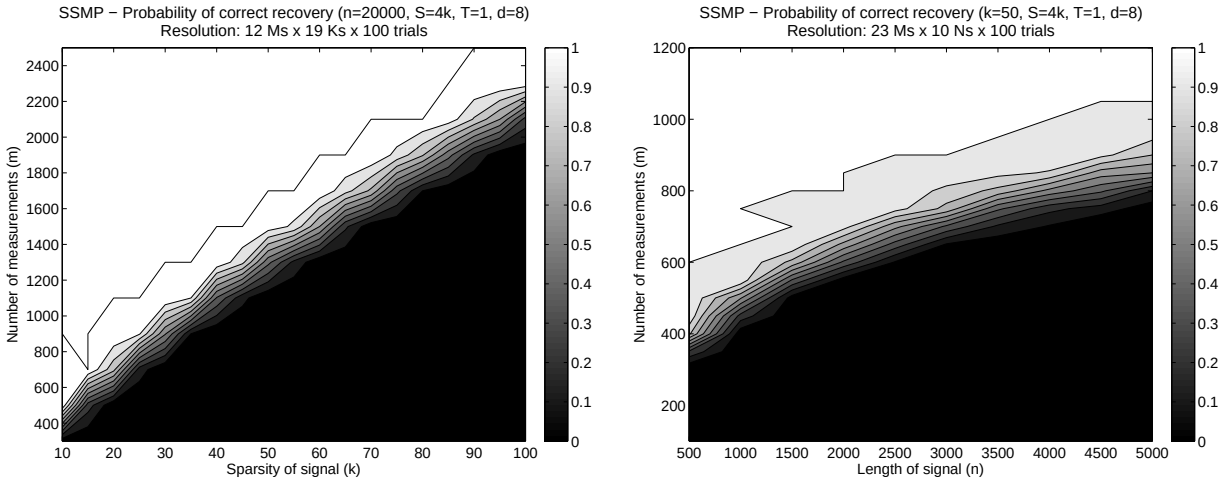


Figure 2-13: Sparse vector recovery experiments with SSMP ($d = 8$)

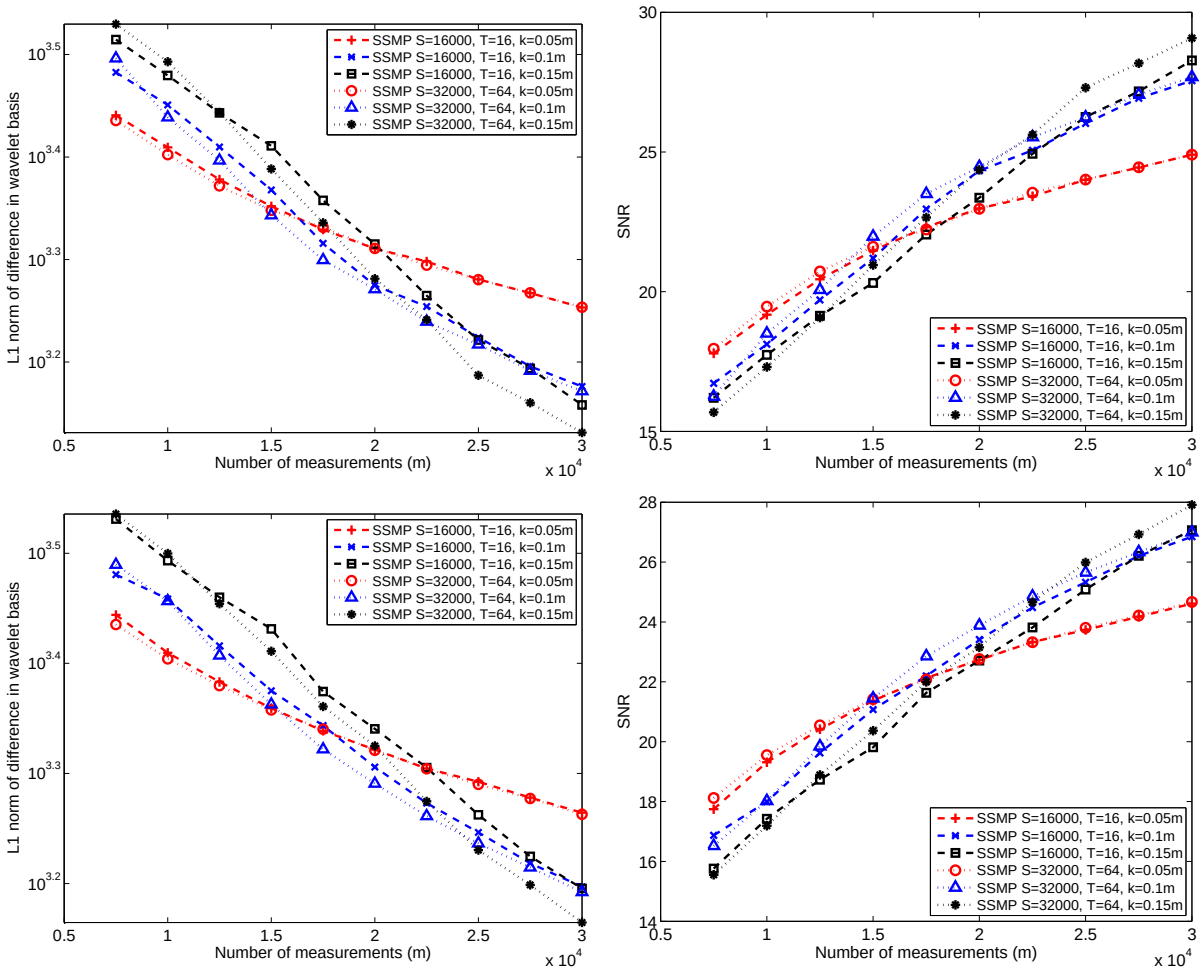


Figure 2-14: Recovery quality of SSMP on peppers (top) and boat (bottom) images ($d = 8$)

recovery quality; note that small random variations exist as each experiment uses a different random matrix. As with SMP, the choice of k is important in retrieving the best approximation. Increasing the number of inner steps S beyond a small multiple of k does not lead to significant improvement; on the other hand, increasing the number of iterations T generally increases the quality of the recovery. Comparing the SNR values with those in figures 2-8, 2-12, we see that SSMP can achieve significantly higher recovery quality than SMP, in many instances achieving SNRs very close to LP. The running times are, as expected, proportional to the total number of steps $S \cdot T$ (see bottom-right plot of figure 2-15). Varying k does not affect the running time significantly.

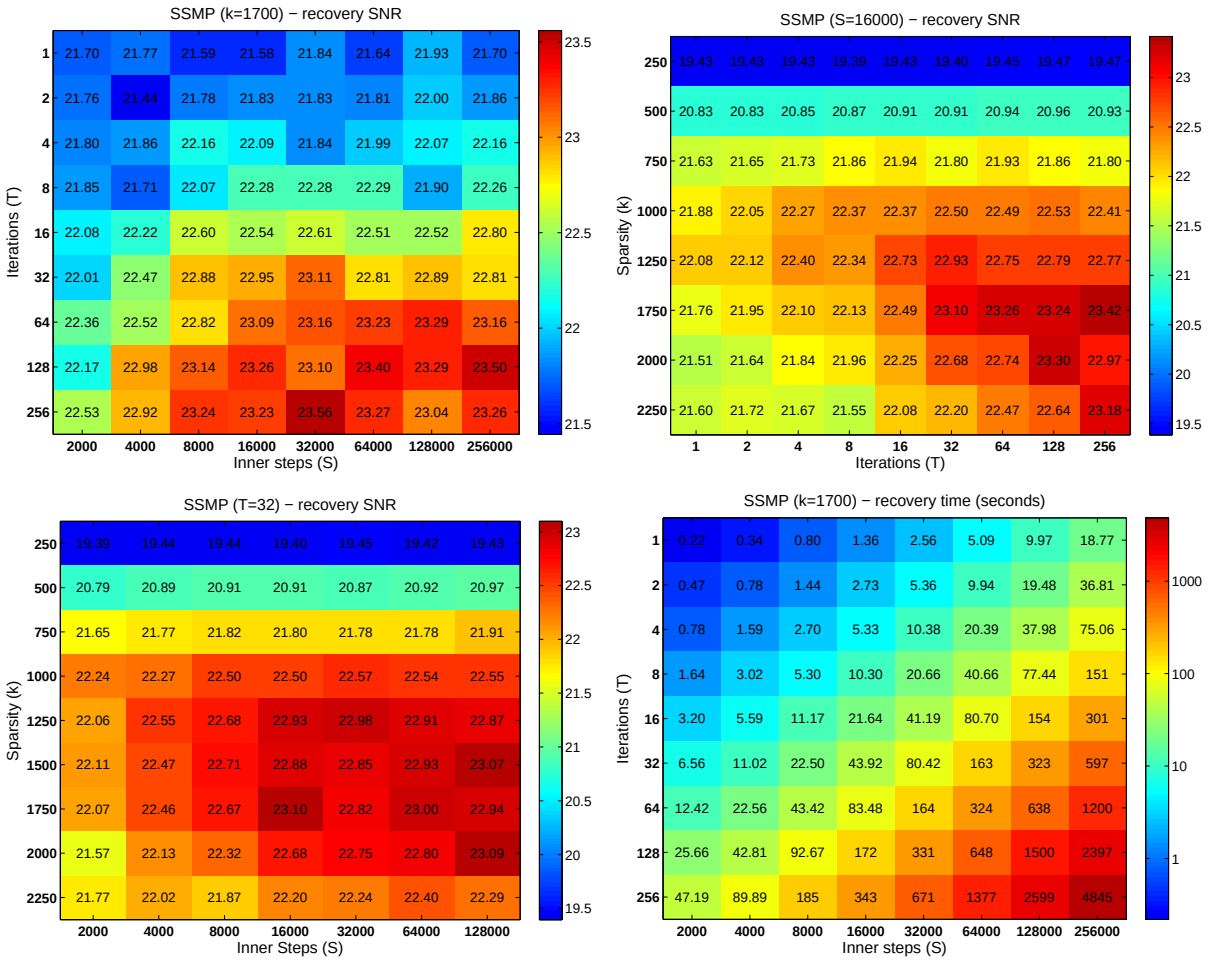


Figure 2-15: Experimental characterisation of SSMP (peppers image, $m = 17000$, $d = 8$)

2.7.5 Comparisons

The presented sparse recovery plots for LP (figure 2-6 and 2-4), SMP (figure 2-9) and SSMP (figure 2-13) are designed to be directly comparable. They show the necessary number of measurements to recover sparse vectors, with the same signal lengths and sparsities. SMP requires about 5 times more measurements than LP decoding, but the decoding process is very fast. SSMP falls between the two, requiring about 2-3 times more measurements than LP. SSMP is generally slower than SMP and faster than LP; for the particular case of strictly sparse signals however, SSMP is at least as fast as SMP as only one iteration is sufficient ($T = 1$).

For image recovery, we show plots comparing the ℓ_1 recovery error, the signal-to-noise ratio, and the decoding time for LP, SMP, and SSMP. We run SMP with $\xi = 0.6$ and two possibilities for k: $k = 0.25m$ and $k = 0.75m$, exactly as in plot 2-11; each data point corresponds to the better of the two recoveries (in terms of SNR). Similarly, we run SSMP with $k = 0.05m$, $k = 0.1m$ and $k = 0.15m$ as in plot 2-14, and we show the data point corresponding to the best recovery. The results are shown in figure 2-16 for the peppers image and in figure 2-17 for the boat image. SMP results in the lowest recovery quality, but it is the fastest algorithm. SSMP is slower, but the quality is very good, exceeding that of LP decoding for high values of m.

In addition, the values shown in 2-8, 2-12, and 2-15 are also directly comparable. Appendix B.1 contains a collection of actual recovered images corresponding to different instances pictured in these plots. They all use the same number of measurements $m = 17000$, and are meant to be comparable with each other as well as with the images in [APT09], where other algorithms are compared. Notice in particular that when SSMP is run with a smaller number of inner steps S and iterations T, it achieves decoding times and recovery errors similar to SMP. SSMP is very versatile: one is able to choose the trade-off between decoding time and recovery error over a large range, effectively "filling the gap" between the fast SMP algorithm and the high-quality LP method.

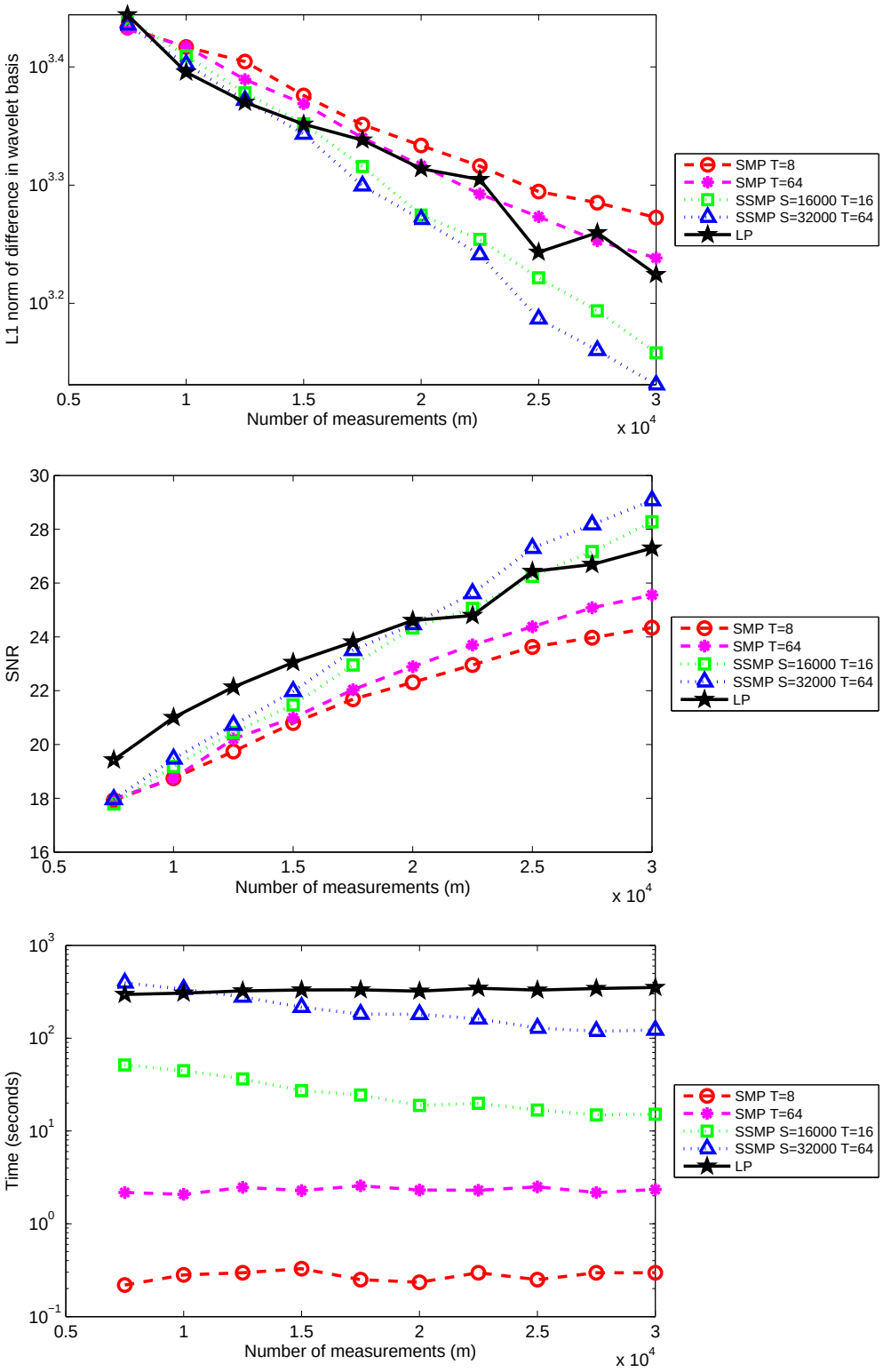


Figure 2-16: Comparison between SMP, SSMP, and LP recovery on the peppers image ($d = 8$)

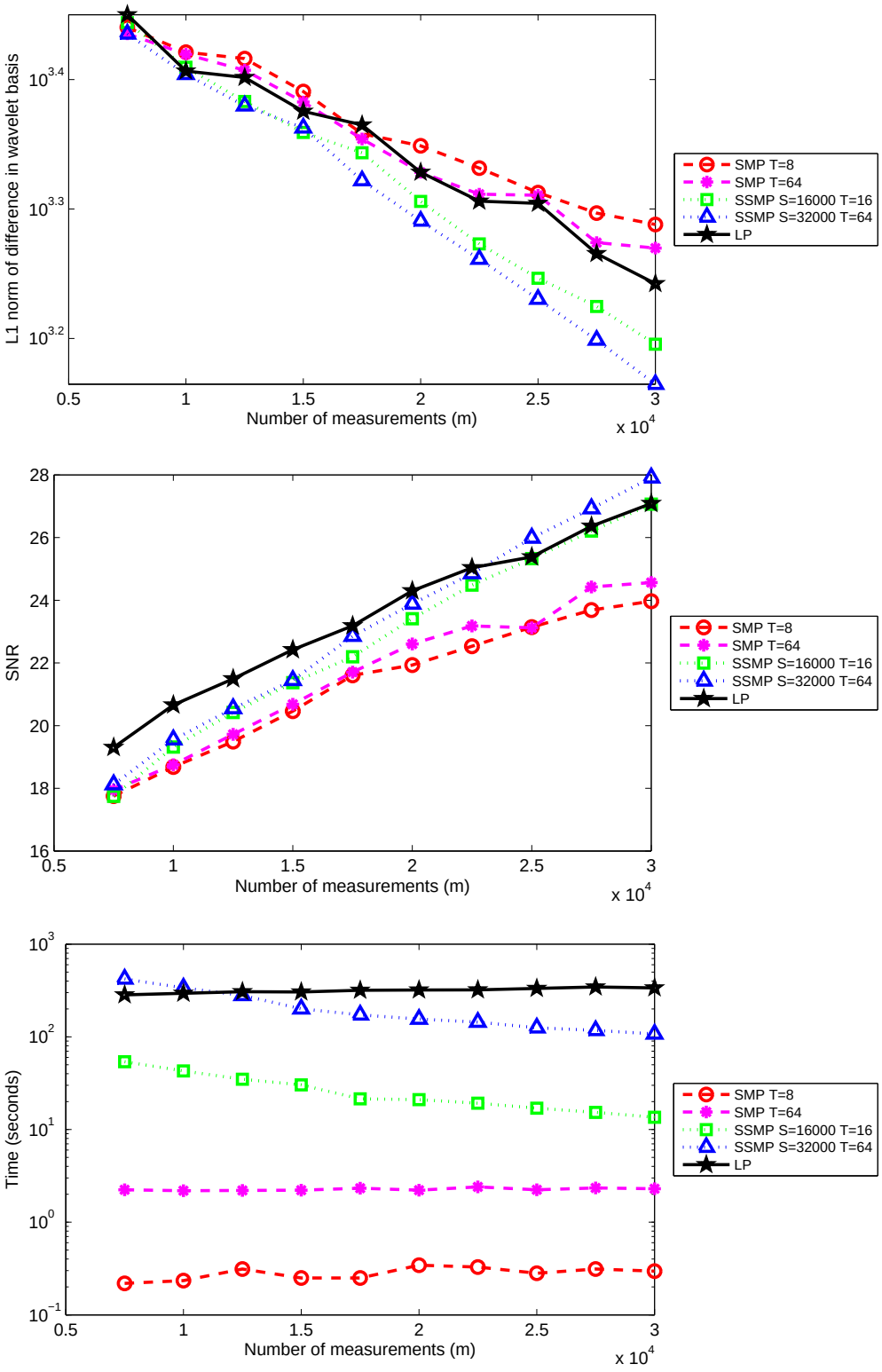


Figure 2-17: Comparison between SMP, SSMP, and LP recovery on the boat image ($d = 8$)

Chapter 3

Non-linear Measurements: Counters

3.1 Introduction

The results presented in this chapter are relevant for data stream algorithms [Ind07, Mut05]; such algorithms aim to process huge streams of updates in a single pass and store a compact summary from which properties of the input can be discovered, with strong guarantees on the quality of the result. This approach has found many applications in large scale data processing and data warehousing [HSST05, BR99, FSGM+98, HPDW01], as well as in other areas, such as network measurements [ABW03, CKMS03, DOM02, EV01], sensor networks [BGS01, SBAS04] and compressed sensing [GSTV07, CRT06].

Finding the "heavy hitters" is one of the quintessential problems in data stream algorithms. Given a stream of items (possibly with weights attached), find those items with the greatest total weight. This is an intuitive problem, that applies to many natural questions: given a stream of search engine queries, which are the most frequently occurring terms? Given a stream of supermarket transactions and prices, which items have the highest total dollar sales? Further, this simple question turns out to be a core subproblem of many more complex computations over data streams, such as estimating the entropy [CCM07], and clustering geometric data [Ind04]. Therefore, it is of high importance to design efficient algorithms for this problem, and understand the performance of existing ones.

The problem can be formalized into one of estimating item frequencies. In this problem we are given a stream of N elements from some universe; the goal is to compute, for each universe element i, an estimator $\hat{f}_i$ that approximates f_i, the number of times the element

Algorithm	Type	Space	Error bound
FREQUENT [DOM02, MG82, KSP03]	Counter	$O(1/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1$
FREQUENT [BKMT03]	Counter	$O(1/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1^{res(1)}$
LOSSYCOUNTING [MM02]	Counter	$O(1/\epsilon \log(\epsilon F_1))$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1$
SPACESAVING [MAA05]	Counter	$O(1/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon F_1$
Count-Min [CM04]	Sketch	$O((k/\epsilon) \cdot \log n)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon/k \cdot F_1^{res(k)}$
Count-Sketch [CCFC02]	Sketch	$O((k/\epsilon) \cdot \log n)$	$(f_i - \hat{f}_i)^2 \leq \epsilon/k \cdot F_2^{res(k)}$
Presented results [BCIS09]	Counter	$O(k/\epsilon)$	$\lvert f_i - \hat{f}_i \rvert \leq \epsilon/k \cdot F_1^{res(k)}$

Table 3.1: Previously known bounds of frequency estimation algorithms.
F_1 is the sum of all frequencies; $F_1^{res(k)}$ is the sum of all but the top k frequencies; $F_2^{res(k)}$ is the sum of the squares of all but the top k frequencies; n is the size of the domain from which the stream elements are drawn.

i occurs in the data stream (or the sum of associated weights in a weighted version). Such estimators provide a succinct representation of the data stream, with a controllable trade-off between description size and approximation error.

An algorithm for frequency estimation is characterized by two related parameters: the space[1] and the bounds on the error in estimating the f_is. The error bounds are typically of the "additive" form, namely we have $|f_i - \hat{f}_i| \leq \epsilon B$, for a B (as in "bound") that is a function of the stream. The bound B is equal either to the size of the whole stream (equivalently, to the quantity F_1, where $F_p = \sum_i (f_i)^p$ for $p \geq 1$), or to the size of the *residual* tail of the stream, given by $F_1^{res(k)}$, the sum of the frequencies of all elements other than the k most frequent ones (heavy hitters) [2]. The residual guarantee is more desirable, since it is always at least as good as the F_1 bound. More strongly, since streams from real applications often obey a very *skewed* frequency distribution, with the heavy hitters constituting the bulk of the stream, a residual guarantee is *asymptotically* better. In particular, in the extreme case when there are only k distinct elements present in the stream, the residual error bound is zero, i.e. the frequency estimation is exact.

Algorithms for this problem have fallen into two main classes: (deterministic) "counter" algorithms and (randomized) "sketch" algorithms (like those presented in chapter 2). Table 3.1 summarizes the space and error bounds of some of the main examples of such algorithms. As is evident from the table, the bounds for the counter and sketching algorithms are incomparable: counter algorithms use less space, but have worse error guarantees than

[1] We measure space in memory words, each consisting of a logarithmic number of bits.

[2] In this chapter we use a different notation which is more common with streaming algorithms; relating this to the notation of chapter 2, we have $F_1^{res(k)} = \|f - f^{(k)}\|_1$ and $F_p = \|f\|_p^p$, in particular $F_1 = \|f\|_1$.

sketching algorithms. In practice, however, the *actual* performance of counter-based algorithms has been observed to be appreciably better than of the sketch-based ones, given the same amount of space [CH08]. The reason for this disparity has not previously been well understood or explained. This has led users to apply very conservative bounds in order to provide the desired guarantees; it has also pushed users towards sketch algorithms in favor of counter algorithms since the latter are not perceived to offer the same types of guarantee as the former.

Presented results We present the results published in [BCIS09], in which we show that the good empirical performance of counter-based algorithms is not an accident: they actually *do* satisfy a much stronger error bound than previously thought. Specifically:

- We identify a general class of *Heavy-Tolerant Counter algorithms* (HTC), that contains the most popular FREQUENT and SPACESAVING algorithms. The class captures the essential properties of the algorithms and abstracts away from the specific mechanics of the procedures.
- We show that any HTC algorithm that has an ϵF_1 error guarantee in fact satisfies the stronger residual guarantee.

We conclude that FREQUENT and SPACESAVING offer the residual bound on error, while using less space than sketching algorithms. Moreover, counter algorithms have small constants of proportionality hidden in their asymptotic cost compared to the much larger logarithmic factors of sketch algorithms, making these space savings very considerable in practice. We also establish through a lower bound that the space usage of these algorithms is within a small constant factor of the space required by any counter algorithm that offers the residual bound on error.

The new bounds have several consequences beyond the immediate practical ramifications. First, we show that they provide better bounds for the *sparse approximation* problem, as defined in chapter 1. This problem is to find the best representation f^* of the frequency distribution, so that f^* has only k non-zero entries. Such a representation captures exact stream statistics for all but $\|f - f^*\|_1$ stream elements. We show that using a counter algorithm to produce the k largest estimated frequencies $\hat{f}_i$ yields a good

solution to this problem. Formally, let S be the set of the k largest entries in $\hat{f}$, generated by a counter algorithm with $O(k/\epsilon)$ counters. Let f^* be an n-dimensional vector such that f_i^* is equal to $\hat{f}_i$ if $i \in S$ and $f_i^* = 0$ otherwise. Then we show that under the L_p norm, for any $p \geq 1$, we have

$$\|f - f^*\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p}$$

This is the best known result for this problem in a streaming setting; note that the error is always at least $(F_p^{res(k)})^{1/p}$. The best known sketching algorithms achieve this bound using $\Omega(k \log \frac{n}{k})$ space (see [BGI+08, BIR08, IR08]); in contrast, our approach yields a space bound of $O(k)$. By extracting all m approximated values from a counter algorithm (as opposed to just top k), we are able to show another result. Specifically, by modifying the algorithms to ensure that they always provide an *underestimate* of the frequencies, we show that the resulting reconstruction has L_p error $(1+\epsilon)(\epsilon/k)^{1-1/p} F_1^{res(k)}$ for any $p \geq 1$.

As noted above, many common frequency distributions are naturally skewed. We show that if the frequencies follow a Zipfian distribution with parameter $\alpha > 1$, then the same tail guarantee follows using only $O(\epsilon^{-1/\alpha})$ space. Lastly, we also discuss extensions to the cases when streams can include arbitrary weights for each occurrence of an item; and when multiple streams are summarized and need to be merged together into a single summary. We show how the algorithms considered can be generalized to handle both of these situations.

3.1.1 Related Work

There is a large body of algorithms proposed in the literature for heavy hitters problems and their variants; see [CH08] for a survey. Most of them can be classified as either *counter-based* or *sketch-based*. The first counter algorithm is due to Misra and Gries [MG82], which we refer to as FREQUENT. Several subsequent works discussed efficient implementation and improved guarantees for this algorithm [DOM02, BKMT03]. In particular, Bose *et al.* showed that it offers an $F_1^{res(1)}$ guarantee [BKMT03]. Our main result is to improve this to $F_1^{res(k)}$, for a broader class of algorithms.

A second counter algorithm is the LOSSYCOUNTING algorithm of Manku and Motwani. This has been shown to require $O(1/\epsilon)$ counters over randomly ordered streams

to give an ϵF_1 guarantee, but there are adversarial order streams for which it requires $O(1/\epsilon \log \epsilon n)$ [MM02]. Our results hold over all possible stream orderings.

The most recent counter solution is the SpaceSaving algorithm due to Metwally *et al.* [MAA05]. The algorithm is shown to offer an F_1 guarantee, and also analyzed in the presence of data with Zipfian frequency distribution. Here, we show an $F_1^{res(k)}$ bound, and demonstrate similar bounds for Zipfian data for a larger class of counter algorithms.

Sketch algorithms are based on linear projections of the frequency vector onto a smaller sketch vector, using compact hash functions to define the projection. Guarantees in terms of $F_1^{res(k)}$ or $F_2^{res(k)}$ follow by arguing that the items with the k largest frequencies are unlikely to (always) collide under the random choice of the hash functions, and so these items can effectively be "removed" from consideration. Because of this random element, sketches are analyzed probabilistically, and have a probability of failure that is bounded by $1/n^c$ for a constant c (n is the size of the domain from which the stream elements are drawn). The Count-Sketch requires $O((k/\epsilon) \log n)$ counters to give guarantees on the sum of squared errors in terms of $F_2^{res(k)}$ [CCFC02]; the Count-Min sketch uses $O((k/\epsilon) \log n)$ counters to give guarantees on the absolute error in terms of $F_1^{res(k)}$ [CM04]. These two guarantees are incomparable in general, varying based on the distribution of frequencies. A key distinction of sketch algorithms is that they allow both positive and negative updates (where negative updates can correspond to deletions, in a transactional setting, or simply arbitrary signal values, in a signal processing environment). This, along with the fact that they are linear transforms, means that they can be used to solve problems such as designing measurements for compressed sensing systems [GSTV07, CRT06]. So, although our results show that counter algorithms are strictly preferable to sketches when both are applicable, there are problems that are solved by sketches that cannot be solved using counter algorithms.

We summarize the main properties of these algorithms, along with the correspond results based on our analysis, in Table 3.1.

Algorithm 4: FREQUENT(m)

```
T ← ∅;
foreach i do
    if i ∈ T then
        c_i ← c_i + 1;
    else if |T| < m then
        T ← T ∪ {i};
        c_i ← 1;
    else forall j ∈ T do
        c_j ← c_j − 1;
        if c_j = 0 then
            T ← T\{j};
```

Algorithm 5: SPACESAVING(m)

```
T ← ∅;
foreach i do
    if i ∈ T then
        c_i ← c_i + 1;
    else if |T| < m then
        T ← T ∪ {i};
        c_i ← 1;
    else
        j ← arg min_{j∈T} c_j;
        c_i ← c_j + 1;
        T ← T ∪ {i}\{j};
```

Figure 3-1: Pseudocode for FREQUENT and SPACESAVING algorithms

3.2 Preliminaries

We introduce the notation used throughout this chapter. The algorithms maintain at most m counters which correspond to a "frequent" set of elements occurring in the input stream. The input stream contains elements, which we assume to be integers between 1 and n. We denote a stream of size N by $u_1, u_2, \ldots u_N$. We use $u_{x\ldots y}$ as a shorthand for the partial stream $u_x, u_{x+1}, \ldots, u_y$.

We denote frequencies of elements by an n-dimensional vector f. For ease of notation, we assume without loss of generality that elements are indexed in order of decreasing frequency, so that $f_1 \geq f_2 \geq \ldots \geq f_n$. When the stream is not understood from context, we specify it explicitly, e.g. $f(u_{x\ldots y})$ is the frequency vector for the partial stream $u_{x\ldots y}$. We denote the sum of the frequencies by F_1; we denote the sum of frequencies except the largest ones by $F_1^{res(k)}$, and we generalize the definition to sums of powers of the frequencies:

$$F_p^{res(k)} = \sum_{i=k+1}^{n} f_i^p, \qquad F_p = F_p^{res(0)}$$

The algorithms considered in this chapter can be thought of as adhering to the following form. The state of an algorithm is represented by an n-dimensional vector of counters c. The vector c has at most m non-zero elements. We denote the "frequent" set by $T = \{i \mid c_i \neq 0\}$, since only this set needs to be explicitly stored. The counter value of an element is an approximation for its frequency; the error vector of the approximation is denoted by δ, with $\delta_i = |f_i - c_i|$.

We demonstrate our results with reference to two known counter algorithms: FREQUENT and SPACESAVING. Although similar, the two algorithms differ in the analysis and their behavior in practice. Both maintain their frequent set T, and process a stream of updates. Given a new item i in the stream which is stored in T, both simply increase the corresponding counter c_i; or, if $i \notin T$ and $|T| < m$, then i is stored with a count of 1. The algorithms differ when an unstored item is seen and $|T| = m$: FREQUENT decrements all stored counters by 1, and (implicitly) throws out any counters with zero count; SPACESAVING finds an item j with smallest non-zero count c_j and assigns $c_i \leftarrow c_j + 1$, followed by $c_j \leftarrow 0$, so in effect i replaces j in T. Pseudocode for these algorithms is presented in Figure 3-1

These algorithms are known to provide a "heavy hitter" guarantee on the approximation errors of the counters:

Definition 4. *An m-counter algorithm provides a* heavy hitter guarantee with constant $A > 0$ *if, for any stream,*

$$\delta_i \leq \left\lfloor A\frac{F_1}{m} \right\rfloor \qquad \forall i$$

More precisely, they both provide this guarantee with constant $A = 1$. Our result is that they also satisfy the following stronger guarantee:

Definition 5. *An m-counter algorithm provides a k-*tail guarantee with constants (A, B), with $A, B > 0$ *if for any stream*

$$\delta_i \leq \left\lfloor A\frac{F_1^{res(k)}}{m - Bk} \right\rfloor \qquad \forall i$$

Note that the heavy hitter guarantee is equivalent to the 0-tail guarantee. Our general proof (which can be applied to a broad class of algorithms) yields a k-tail guarantee with constants $A = 1$, $B = 2$ for both algorithms (for any $k \leq m/2$). However, by considering particular features of FREQUENT and SPACESAVING, we prove a k-tail guarantee with constants $A = B = 1$ for any $k < m$ following appropriate analysis (see 3.3.1, 3.3.2).

The lower bound proved in 3.8 establishes that any counter algorithm that provides an error bound of $\frac{F_1^{res(k)}}{m-k}$ must use at least $(m - k)/2$ counters; thus the number of counters FREQUENT and SPACESAVING use is within a small factor (3 for $k \leq m/3$) of the optimal.

3.3 Specific proofs

We begin with specific proofs for the FREQUENT and SPACESAVING algorithms (see 3-1).

3.3.1 Tail guarantee with constants $A = B = 1$ for FREQUENT

We can interpret the FREQUENT algorithm in the following way: each element in the stream results in incrementing one counter; in addition, some number of elements (call this number d) also result in decrementing $m+1$ counters (we can think of the d elements incrementing and later decrementing their own counter). The sum of the counters at the end of the algorithm is $\|c\|_1$. We have

$$\|c\|_1 = \|f\|_1 - d(m+1)$$

Since there were d decrement operations, and each operation decreases any given counter by at most one, it holds that the final counter value for any element is at least $f_i - d$. We restrict our attention to the k most frequent elements. Then

$$\begin{aligned}
\|c\|_1 = \|f\|_1 - d(m+1) &\geq \sum_{i=1}^{k}(f_i - d) \\
\|f\|_1 - d(m+1) &\geq -dk + \sum_{i=1}^{k} f_i \\
\sum_{i=k+1}^{n} f_i &\geq d(m+1-k) \\
d &\leq \frac{F_1^{res(k)}}{m+1-k}
\end{aligned}$$

Since the error in any counter is at most d, this implies the k-tail guarantee with $A = B = 1$.

3.3.2 Tail guarantee with constants $A = B = 1$ for SPACESAVING

The tail guarantee follows almost immediately from the following claims proven in [MAA05]:

LEMMA 3 IN [MAA05]: *If the minimum non-zero counter value is Δ, then $\delta_i \leq \Delta$ for all i.*

THEOREM 2 IN [MAA05]: *Whether or not element i (i.e. i-th most frequent element) corresponds to the i-th largest counter, the value of this counter is at least f_i, the frequency of i.*

If we restrict our attention to the k largest counters, the sum of their values is at least $\sum_{i=1}^{k} f_i$. Since in this algorithm the sum of the counters is always equal to the length of the stream, it follows that:

$$\Delta \leq \frac{\|f\|_1 - \sum_{i=1}^{k} f_i}{m-k}$$

thus by Lemma 3

$$\delta_i \leq \frac{F_1^{res(k)}}{m-k} \qquad \forall i$$

which is the k-tail guarantee with constants $A = B = 1$.

3.4 Residual Error Bound

In this section we state and prove our main result on the error bound for a class of heavy-tolerant counter algorithms. We begin by formally defining this class.

Definition 6. *A value i is x-*prefix guaranteed *for the stream $u_{1\ldots s}$ if after the first $x < s$ elements of the stream have been processed, i will stay in T even if some elements are removed from the remaining stream (including occurrences of i). Formally, the value i is x-*prefix guaranteed *if $0 \leq x < s$ and $c_i(u_{1\ldots x}v_{1\ldots t}) > 0$ for all subsequences $v_{1\ldots t}$ of $u_{(x+1)\ldots s}$, $0 \leq t \leq s - x$.*

Note that if i is x-prefix guaranteed, then i is also y-prefix guaranteed for all $y > x$.

Definition 7. *A counter algorithm is* heavy-tolerant *if extra occurrences of guaranteed elements do not increase the estimation error. Formally, an algorithm is* heavy-tolerant *if for any stream $u_{1\ldots s}$, given any x, $1 \leq x < s$, for which element $i = u_x$ is $(x-1)$-prefix guaranteed, it holds that*

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots(x-1)}u_{(x+1)\ldots s}) \qquad \forall j$$

Theorem 6. *Algorithms* FREQUENT *and* SPACESAVING *are heavy-tolerant.*

Theorem 7. *If a heavy-tolerant algorithm provides a heavy hitter guarantee with constant A, it also provides a k-tail guarantee with constants $(A, 2A)$, for any k, $1 \leq k < m/2A$.*

3.4.1 Proof of Heavy Tolerance

Intuitively, this is true because occurrences of an element already in the frequent set only affect the counter value of that element; and, as long as the element never leaves the frequent set, the value of its counter does not affect the algorithm's other choices.

of Theorem 6. Denote $v_{1\ldots t} = u_{(x+1)\ldots(x+t)}$, with $t \leq s - x$. We prove by induction on t that for both algorithms

$$c(u_{1\ldots x}v_{1\ldots t}) = c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i$$

where e_i is the i-th row of I_n, the $n \times n$ identity matrix; this implies that

$$\delta(u_{1\ldots x}v_{1\ldots t}) = \delta(u_{1\ldots(x-1)}v_{1\ldots t})$$

Base case at $t = 0$: By the hypothesis: $c_i(u_{1\ldots(x-1)}) \neq 0$, hence when element $u_x = i$ arrives after processing $u_{1\ldots x}$, both FREQUENT and SPACESAVING just increase i's counter:

$$c(u_{1\ldots x}) = c(u_{1\ldots(x-1)}) + e_i$$

Induction step for $t > 0$: We are given that

$$c(u_{1\ldots x}v_{1\ldots(x-1)}) = c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_i$$

Note that since i is $(x-1)$-prefix guaranteed, these vectors have the same support.

Case 1: $c_{v_t}(u_{1\ldots x}v_{1\ldots(t-1)}) > 0$. Hence $c_{v_t}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) > 0$. For both streams, v_t's counter just gets incremented and thus

$$\begin{aligned} c(u_{1\ldots x}v_{1\ldots t}) &= c(u_{1\ldots x}v_{1\ldots(t-1)}) + e_{v_t} \\ &= c(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) + e_{v_t} + e_i \\ &= c(u_{1\ldots(x-1)}v_{1\ldots t}) + e_i \end{aligned}$$

Case 2: $c_{v_t}(u_{1\ldots x}v_{1\ldots(t-1)}) = 0$. Note that $v_t \neq i$ since i is x-prefix guaranteed and $c_{v_t}(u_{1\ldots(x-1)}v_{1\ldots(t-1)}) = 0$. By the induction hypothesis, both counter vectors have the

same support (set of non-zero entries). If the support is less than m, then the algorithm adds e_{v_k} to the counters, and the analysis follows Case 1 above. Otherwise, the two algorithms differ:

- FREQUENT algorithm: In this case all non-zero counters will be decremented. Since both counter vectors have the same support, they will be decremented by the same m-sparse binary vector $\gamma = \chi(T) = \sum_{j:c_j \neq 0} e_j$.

- SPACESAVING algorithm: The minimum non-zero counter is set to zero. To avoid ambiguity, we specify that SPACESAVING will pick the counter c_j with the smallest identifier j if there are multiple counters with equal smallest non-zero value. Let

$$j = \operatorname*{argmin}_{j \in T(u_{1\ldots x} v_{1\ldots(t-1)})} c_j(u_{1\ldots x} v_{1\ldots(t-1)})$$

and

$$j' = \operatorname*{argmin}_{j' \in T(u_{1\ldots(x-1)} v_{1\ldots(t-1)})} c_{j'}(u_{1\ldots(x-1)} v_{1\ldots(t-1)})$$

Since i is x-prefix guaranteed, its counter can never become zero, hence $j \neq i, j' \neq i$. Since

$$c_{i'}(u_{1\ldots x} v_{1\ldots(t-1)}) = c_{i'}(u_{1\ldots(x-1)} v_{1\ldots(t-1)})$$

for all $i' \neq i$, it follows that $j = j'$ and

$$c_j(u_{1\ldots x} v_{1\ldots(t-1)}) = c_{j'}(u_{1\ldots(x-1)} v_{1\ldots(t-1)}) = M.$$

Hence both streams result in updating the counters by subtracting the same difference vector $\gamma = Me_j - (M+1)e_{v_t}$

So each algorithm computes some difference vector γ irrespective of which stream it is applied to, and updates the counters:

$$\begin{aligned} c(u_{1\ldots x} v_{1\ldots t}) &= c(u_{1\ldots x} v_{1\ldots(t-1)}) - \gamma \\ &= c(u_{1\ldots(x-1)} v_{1\ldots(t-1)}) + e_i - \gamma \\ &= c(u_{1\ldots(x-1)} v_{1\ldots t}) + e_i \end{aligned}$$

□

3.4.2 Proof of k-tail guarantee

Let $\texttt{Remove}(u_{1\ldots s}, i)$ be the subsequence of $u_{1\ldots s}$ with all occurrences of value i removed, i.e.

$$\texttt{Remove}(u_{1\ldots s}, i) = \begin{cases} \text{empty sequence} & \text{if } s = 0 \\ (u_1, \texttt{Remove}(u_{2\ldots s}, i)) & \text{if } u_1 \neq i \\ \texttt{Remove}(u_{2\ldots s}, i) & \text{if } u_1 = i \end{cases}$$

Lemma 8. *If i is x-prefix guaranteed and the algorithm is heavy-tolerant, then*

$$\delta_j(u_{1\ldots s}) \leq \delta_j(u_{1\ldots x} v_{1\ldots t}) \qquad \forall j$$

where $v_{1\ldots t} = \texttt{Remove}(u_{(x+1)\ldots s}, i)$, with $0 \leq t \leq s - x$.

Proof. Let $x_1, x_2, \ldots, x_q$ be the positions of occurrences of i in $u_{(x+1)\ldots s}$, with $x < x_1 < x_2 < \ldots < x_q$. We apply the heavy-tolerant definition for each occurrence; for all j:

$$\begin{aligned} \delta_j(u_{1\ldots s}) &\leq \delta_j(u_{1\ldots(x_1-1)} u_{(x_1+1)\ldots s}) \\ &\leq \delta_j(u_{1\ldots(x_1-1)} u_{(x_1+1)\ldots(x_2-1)} u_{(x_2+1)\ldots s}) \\ &\leq \ldots \\ &\leq \delta_j(u_{1\ldots x} v_{1\ldots t}) \end{aligned}$$

Note in particular that $\delta_i(u_{1\ldots p})$, the error in estimating the frequency of i in the original stream, is identical to $\delta_i(u_{1\ldots x} v_{1\ldots q})$, the error of i on the derived stream, since i is x-prefix guaranteed. □

Definition 8. *An* error bound *for an algorithm is a function $\Delta : \mathbb{N}^n \to \mathbb{R}^+$ such that for any stream $u_{1\ldots s}$*

$$\delta_i(u_{1\ldots s}) \leq \lfloor \Delta(f(u_{1\ldots s})) \rfloor \qquad \forall i$$

In addition, Δ must be "increasing" in the sense that for any two frequency vectors f' and f'' such that $f'_i \leq f''_i$ for all i, it holds that $\Delta(f') \leq \Delta(f'')$.

Lemma 9. *Let Δ be an error bound for a heavy-tolerant algorithm that provides a heavy hitter guarantee with constant A. Then the following function is also an error bound for*

the algorithm, for any k, $1 \le k < m/A$:

$$\Delta'(f) = A\frac{k\Delta(f) + k + F_1^{res(k)}}{m}$$

Proof. Let $u_{1\ldots s}$ be any stream. Let $D = 1 + \lfloor \Delta(f(u_{1\ldots s})) \rfloor$. We assume without loss of generality that the elements are indexed in order of increasing frequency.

Let $k' = \max\{i \mid 1 \le i \le k \text{ and } f_i(u_{1\ldots s}) > D\}$.

For each $i \le k'$ let x_i be the position of the D-th occurrence of i in the stream. We claim that any $i \le k'$ is x_i-prefix guaranteed: let $v_{1\ldots t}$ be any subsequence of $u_{(x_i+1)\ldots s}$; it holds for all j that

$$\delta_j(u_{1\ldots x_i}v_{1\ldots t}) \le \lfloor \Delta(f(u_{1\ldots x_i}v_{1\ldots t})) \rfloor < D$$

$$\text{and so } c_j(u_{1\ldots x_i}v_{1\ldots t}) \ge f_j(u_{1\ldots x_i}v_{1\ldots t}) - \delta_j(u_{1\ldots x_i}v_{1\ldots t}) > D - D = 0.$$

Let $i_1, i_2, \ldots i_{k'}$ be the permutation of $1 \ldots k'$ so that $x_{i_1} > x_{i_2} > \ldots > x_{i_{k'}}$. We can apply Lemma 8 for i_1 which is x_{i_1}-prefix guaranteed; for all j

$$\delta_j(u_{1\ldots s}) \le \delta_j(u_{1\ldots x_{i_1}}v_{1\ldots s_v})$$

where $v_{1\ldots s_v} = \texttt{Remove}(u_{(x_{i_1}+1)\ldots s}, i_1)$.

Since $x_2 < x_1$, i_2 is x_2-prefix guaranteed for the new stream $u_{1\ldots x_{i_1}}v_{1\ldots s_v}$ and we apply Lemma 8 again:

$$\delta_j(u_{1\ldots s}) \le \delta_j(u_{1\ldots x_{i_1}}v_{1\ldots s_v}) \le \delta_j(u_{1\ldots x_{i_2}}w_{1\ldots s_w}) \qquad \forall j$$

where $w_{1\ldots s_w} = \texttt{Remove}(u_{(x_{i_2}+1)\ldots x_{i_1}}v_{1\ldots s_v}, i_2)$. Since the x_{i_j} values are decreasing, we can continue this argument for $i = 3, 4, \ldots, k'$. We obtain the following inequality for the final stream $z_{1\ldots s_z}$

$$\delta_j(u_{1\ldots s}) \le \delta_j(z_{1\ldots s_z}) \qquad \forall j$$

where $z_{1\ldots s_z}$ is the stream $u_{1\ldots s}$ with all "extra" occurrences of elements 1 to k' removed ("extra" means after the first D occurrences). Thus

$$\|f(z_{1\ldots s_z})\|_1 = k'D + \sum_{i=k'+1}^{n} f_i(u_{1\ldots s})$$

Either $k' = k$, or $k' < k$ and $f_i(u_{1\ldots s}) \le D$ for all $k' < i \le k$; in both cases we can replace

k' with k:

$$\|f(z_{1\ldots s_z})\|_1 \leq kD + \sum_{i=k+1}^{n} f_i(u_{1\ldots s})$$

We now apply the heavy hitter guarantee for this stream; for all j:

$$\begin{aligned}
\delta_j(u_{1\ldots s}) &\leq \delta_j(z_{1\ldots s_z}) \\
&\leq \left\lfloor A\frac{kD + \sum_{i=k+1}^{n} f_i(u_{1\ldots s})}{m} \right\rfloor \\
&\leq \left\lfloor A\frac{k\Delta(u_{1\ldots s}) + k + F_1^{res(k)}}{m} \right\rfloor
\end{aligned}$$

□

We can now prove theorem 7.

of Theorem 7. We start with the initial error bound given by the heavy hitter guarantee $\Delta(f) = A\frac{\|f\|_1}{m}$ and apply Lemma 9 to obtain another error bound Δ'. We can continue iteratively applying Lemma 9 in this way. Either we will eventually obtain a new bound which is worse than the previous one, in which case this process halts with the previous error bound; or else we can analyze the error bound obtained in the limit (in the spirit of [BKMT03]). In both cases, the following holds for the best error bound Δ:

$$\Delta(f) \leq A\frac{k\Delta(f) + k + F_1^{res(k)}}{m}$$

$$\text{and so } \Delta(f) \leq A\frac{k + F_1^{res(k)}}{m - Ak} .$$

We have shown that for any stream $u_{1\ldots p}$,

$$\delta_i(u_{1\ldots p}) \leq \left\lfloor A\frac{k + F_1^{res(k)}}{m - Ak} \right\rfloor \qquad \forall i$$

We show that this implies the guarantee

$$\delta_i(u_{1\ldots p}) \leq \left\lfloor A\frac{F_1^{res(k)}}{m - 2Ak} \right\rfloor \qquad \forall i$$

Case 1: $AF_1^{res(k)} < m - 2Ak$. In this case both guarantees are identical: all errors are 0.

Case 2: $AF_1^{res(k)} \geq m - 2Ak$:

$$\begin{aligned} A^2 k F_1^{res(k)} &\geq Ak(m - 2Ak) \\ A(m - Ak)F_1^{res(k)} &\geq A(m - 2Ak)\left(k + F_1^{res(k)}\right) \\ A\frac{F_1^{res(k)}}{m - 2Ak} &\geq A\frac{k + F_1^{res(k)}}{m - Ak} \end{aligned}$$

□

3.5 Sparse Recoveries

The k-sparse recovery problem is to find a representation f' so that f' has only k non-zero entries ("k-sparse"), and the L_p norm $\|f - f'\|_p = (\sum_{i=1}^{n} |f_i - f'_i|^p)^{1/p}$ is minimized. A natural approach is to build f' from the heavy hitters of f, and indeed we show that this method gives strong guarantees for frequencies from heavy tolerant counter algorithms.

3.5.1 k-sparse recovery

To get a k-sparse recovery, we run counter algorithm that provides a k-tail guarantee with m counters and create f' using the k largest counters. These are not necessarily the k most frequent elements (with indices 1 to k in our notation), but we show that they must be "close enough".

Theorem 8. *If we run a counter algorithm which provides a k-tail guarantee with constants (A, B) using $m = k(\frac{3A}{\varepsilon} + B)$ counters and retain the top k counter values into the k-sparse vector f', then for any $p \geq 1$:*

$$\|f - f'\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p}$$

Proof. Let $K = \{1, \ldots, k\}$ be the set of the k most frequent elements. Let S be the set of elements with the k largest counters. Let $R = \{1, \ldots, n\} \setminus (S \cup K)$ be the set of all other remaining elements. Let $k' = |K \setminus S| = |S \setminus K|$.

Let $x_1 \ldots x_{k'}$ be the k' elements in $S \setminus K$, with $c_{x_1} \geq c_{x_2} \geq \ldots \geq c_{x_{k'}}$. Let $y_1 \ldots y_{k'}$ be the k' elements in $K \setminus S$, with $c_{y_1} \geq c_{y_2} \geq \ldots \geq c_{y_{k'}}$. Notice that $c_{x_i} \geq c_{y_i}$ for any i: c_{y_i} is the i^{th} largest counter in $K \setminus S$, whereas c_{x_i} is the i^{th} largest counter in $(K \cup S) \setminus (S \cap K)$,

a superset of $K \setminus S$. Let Δ be an upper bound on the counter errors δ. Then for any i

$$f_{y_i} - \Delta \leq c_{y_i} \leq c_{x_i} \leq f_{x_i} + \Delta \tag{3.1}$$

Hence $f_{y_i} \leq f_{x_i} + 2\Delta$. Let f' be the recovered frequency vector ($f'_{x_i} = c_{x_i}$ and zero everywhere else). For any $p \geq 1$, and using the triangle inequality $\|a+b\|_p \leq \|a\|_p + \|b\|_p$ on the vector f_i restricted to $i \in R \cup S$ and the vector equal to the constant 2Δ restricted to $i \in S \setminus K$:

$$\begin{aligned}
\|f - f'\|_p &= \left(\sum_{i \in S} (c_i - f_i)^p + \sum_{i \in R \cup K \setminus S} (f_i)^p \right)^{1/p} \\
&\leq \left(\sum_{i=1}^{k} \Delta^p + \sum_{i \in K \setminus S} (f_i)^p + \sum_{i \in R} (f_i)^p \right)^{1/p} \\
&\leq k^{1/p}\Delta + \left(\sum_{i=1}^{k'} (f_{y_i})^p + \sum_{i \in R} (f_i)^p \right)^{1/p} \\
&\leq k^{1/p}\Delta + \left(\sum_{i=1}^{k'} (f_{x_i} + 2\Delta)^p + \sum_{i \in R} (f_i)^p \right)^{1/p} \\
&\leq 3k^{1/p}\Delta + \left(\sum_{i \in R \cup S \setminus K} (f_i)^p \right)^{1/p} \\
&\leq 3k^{1/p}\Delta + (F_p^{res(k)})^{1/p}
\end{aligned}$$

If an algorithm has the tail guarantee with constants (A, B), by using $m = k(\frac{3A}{\varepsilon} + B)$ counters we get

$$\|f - f'\|_p \leq \frac{\varepsilon F_1^{res(k)}}{k^{1-1/p}} + (F_p^{res(k)})^{1/p} \tag{3.2}$$

□

Note that $(F_p^{res(k)})^{1/p}$ is the smallest possible L_p error of any k-sparse recovery of f. Also, if the algorithm provides one-sided error on the estimated frequencies (as is the case for FREQUENT and SPACESAVING), it is sufficient to use $m = k(\frac{2A}{\varepsilon} + B)$ counters, since now $f_{y_i} \leq f_{x_i} + \Delta$.

Estimating $F_1^{res(k)}$. Since our algorithms give guarantees in terms of $F_1^{res(k)}$, a natural

question is to estimate the value of this quantity.

Theorem 9. *If we run a counter algorithm which provides a k-tail guarantee with constants (A, B) using $(Bk + \frac{Ak}{\varepsilon})$ counters and retain the largest k counter values as the k-sparse vector f', then:*

$$F_1^{res(k)}(1-\varepsilon) \leq F_1 - \|f'\|_1 \leq F_1^{res(k)}(1+\varepsilon)$$

Proof. To show this result, we rely on the definitions and properties of sets S and K from the proof of Theorem 8. By construction of sets S and K, $f_{x_i} \leq f_{y_i}$ for any i. Using equation (3.1) it follows that

$$f_{y_i} - \Delta \leq c_{x_i} \leq f_{y_i} + \Delta$$

So the norm of f' must be close to the norm of the best k-sparse representative of f, i.e. $(F_1 - F_1^{res(k)})$. Summing over each of the k counters yields

$$\begin{aligned} F_1 - F_1^{res(k)} - k\Delta \leq \quad & \|f'\|_1 \quad && \leq F_1 - F_1^{res(k)} + k\Delta \\ F_1^{res(k)} - k\Delta \leq \quad & F_1 - \|f'\|_1 && \leq F_1^{res(k)} + k\Delta \end{aligned}$$

The result follows when setting $m = k(\frac{Ak}{\varepsilon} + B)$ so the upper bound ensures $\Delta \leq \frac{\varepsilon}{k} F_1^{res(k)}$.

□

3.5.2 m-sparse recovery

When the counter algorithm uses m counters, it stores approximate values for m elements. It seems intuitive that by using all m of these counter values, the recovery should be even better. This turns out not to be true in general. Instead, we show that it is possible to derive a better result given an algorithm which always *underestimates* the frequencies $(c_i \leq f_i)$. For example, this is true in the case of FREQUENT.

As described so far, SPACESAVING always overestimates, but can be modified to underestimate the frequencies. In particular, the algorithm has the property that error is bounded by the smallest counter value, i.e. $\Delta = \min\{c_j | c_j \neq 0\}$. So setting $c'_i = \max\{0, c_i - \Delta\}$ ensures that $c'_i \leq f_i$. Because $f_i + \Delta \geq c_i \geq f_i$, $f_i - c'_i \leq \Delta$ and thus c' satisfies the same k-tail bounds with $A = B = 1$ (as per 3.3.2). Note that in practice, slightly improved per-item guarantees follow by storing ϵ_i for each non-zero counter c_i as

the value of Δ when i last entered the frequent set, and using $c_i - \epsilon_i$ as the estimated value (as described in [MAA05]).

Theorem 10. *If we run an underestimating counter algorithm which provides a k-tail guarantee with constants (A, B) using $(Bk + \frac{Ak}{\varepsilon})$ counters and retain the counter values into the m-sparse vector f', then for any $p \geq 1$:*

$$\|f - f'\|_p \leq (1 + \varepsilon) \left(\frac{\varepsilon}{k}\right)^{1-1/p} F_1^{res(k)}$$

Proof. Set $m = k(\frac{A}{\varepsilon} + B)$ in Definition 5 to obtain

$$\begin{aligned}
\|f - f'\|_p &= \left(\sum_{i=1}^{k} (f_i - c_i)^p + \sum_{i=k+1}^{n} (f_i - c_i)^p \right)^{1/p} \\
&\leq \left(k \frac{\varepsilon^p}{k^p} (F_1^{res(k)})^p + \sum_{i=k+1}^{n} (f_i - c_i) \frac{\varepsilon^{p-1}}{k^{p-1}} (F_1^{res(k)})^{p-1} \right)^{1/p} \\
&\leq \left(\frac{\varepsilon^p}{k^{p-1}} (F_1^{res(k)})^p + \frac{\varepsilon^{p-1}}{k^{p-1}} (F_1^{res(k)})^p \right)^{1/p} \\
&\leq (1 + \varepsilon) \left(\frac{\varepsilon}{k}\right)^{1-1/p} F_1^{res(k)}
\end{aligned}$$

□

3.6 Zipfian Distributions

Realistic data can often be approximated with a Zipfian [Zip49] distribution; a stream of length $F_1 = N$, with n distinct elements, distributed (exactly) according to the Zipfian distribution with parameter α has frequencies

$$f_i = N \frac{1}{i^\alpha \zeta(\alpha)} \quad \text{where} \quad \zeta(\alpha) = \sum_{i=1}^{n} \frac{1}{i^\alpha}$$

The value $\zeta(\alpha)$ converges to a small constant when $\alpha > 1$. Although data rarely obeys this distribution exactly, our first result requires only that the "tail" of the distribution can be bounded by a (small constant multiple of) a Zipfian distribution. Note that this requires that the frequencies follow this distribution, but the order of items in the stream can be arbitrary.

Theorem 11. *Given Zipfian data with parameter $\alpha \geq 1$, if a counter algorithm that provides a k-tail guarantee with constants (A, B) for $k = \left(\frac{1}{\varepsilon}\right)^{1/\alpha}$ is used with $m = (A + B)\left(\frac{1}{\varepsilon}\right)^{1/\alpha}$ counters, the counter errors are at most εF_1.*

Proof. The k-tail guarantee with constants (A, B) means

$$\Delta = A\frac{F_1^{res(k)}}{m - Bk} \leq A\frac{N}{\zeta(\alpha)}\frac{\sum_{i=k+1}^{n} i^{-\alpha}}{m - Bk}$$

Then

$$\sum_{i=k+1}^{n} \frac{1}{i^\alpha} \leq \int_k^n \frac{1}{x^\alpha}dx = \frac{1}{k^{\alpha-1}}\int_1^{n/k} \frac{1}{x^\alpha}dx \leq \frac{\zeta(\alpha)}{k^{\alpha-1}}$$

$$\Delta \leq A\frac{\zeta(\alpha)}{k^{\alpha-1}}\frac{N}{\zeta(\alpha)(m - Bk)} = \frac{N}{k^\alpha}A\frac{k}{m - Bk}$$

by setting $k = \left(\frac{1}{\varepsilon}\right)^{1/\alpha}$, $m = (A + B)k$,

$$\Delta \leq \frac{N}{k^\alpha} = \varepsilon N$$

□

A similar result is proved for SpaceSaving in [MAA05] under the stronger assumption that the frequencies are exactly as defined by the Zipfian distribution.

3.6.1 Top-k

In this section we analyze the algorithms in the context of the problem of finding top k elements, when the input is Zipf distributed.

Theorem 12. *Assuming Zipfian data with parameter $\alpha > 1$, a counter algorithm that provides a k'-tail guarantee for $k' = \Theta\left(k\left(\frac{k}{\alpha}\right)^{1/\alpha}\right)$ can retrieve the top k elements in correct order using $O\left(k\left(\frac{k}{\alpha}\right)^{1/\alpha}\right)$ counters. For Zipfian data with parameter $\alpha = 1$, an algorithm with k'-tail guarantee for $k' = \Theta(k^2 \ln n)$ can retrieve the top k elements in correct order using $O(k^2 \ln n)$ counters.*

Proof. To get the top k elements in the correct order we need

$$\Delta < \frac{f_k - f_{k+1}}{2}$$

$$\begin{aligned} f_k - f_{k+1} &= \frac{N}{\zeta(\alpha)}\left(\frac{1}{k^\alpha} - \frac{1}{(k+1)^\alpha}\right) \\ &= \frac{N}{\zeta(\alpha)}\frac{(k+1)^\alpha - k^\alpha}{(k+1)^\alpha k^\alpha} \\ &< \frac{N}{\zeta(\alpha)}\frac{\alpha k^{\alpha-1}}{(k+1)^\alpha k^\alpha} = \frac{N}{\zeta(\alpha)}\frac{\alpha}{(k+1)^\alpha k} \end{aligned}$$

Thus we need error rate

$$\varepsilon = \frac{\alpha}{2\zeta(\alpha)(k+1)^\alpha k} = \begin{cases} \Theta(\alpha/k^{1+\alpha}) & \text{for } \alpha > 1 \\ \Theta(1/(k^2 \ln n)) & \text{for } \alpha = 1 \end{cases}$$

The result then follows from Theorem 11. □

3.7 Extensions

3.7.1 Real-Valued Update Streams

So far, we have considered a model of streams where each stream token indicates an arrival of an item with (implicit) unit weight. More generally, streams often include a weight for each arrival: a size in bytes or round-trip time in seconds for Internet packets; a unit price for transactional data, and so on. When these weights are large, or not necessarily integral, it is still desirable to solve heavy hitters and related problems on such streams.

In this section, we make the observation that the two counter algorithms Frequent and SpaceSaving naturally extend to streams in which each update includes a positive real valued weight to apply to the given item. That is, the stream consists of tuples u_i, Each u_i is a tuple (a_i, b_i) representing b_i occurrences of element a_i where $b_i \in \mathbb{R}^+$ is a positive real value.

We outline how to extend the two algorithms to correctly process such streams. For SpaceSaving, observe that when processing each new item a_i, the algorithm identifies a counter corresponding to a_i and increments it by 1. We simply change this to incrementing the appropriate counter by b_i to generate an algorithm we denote SpaceSaving$\mathbb{R}$. It is straightforward to modify the analysis of [MAA05] to demonstrate that SpaceSaving$\mathbb{R}$ achieves the basic Heavy Hitters guarantee (Definition 4). This generalizes SpaceSaving, since when every b_i is 1, then the two algorithms behave identically.

Defining FREQUENTℝ is a little more complex. If the new item $a_i \in T$, then we can simply increases a_i's counter by b_i; and if there are fewer than $m-1$ counters then one can be allocated to a_i and set to b_i. But if a_i is not stored, then the next step depends on the size of $c_{\min}$, the smallest counter value stored in T. If $b_i \leq c_{\min}$, then all stored counters are reduced by b_i. Otherwise, all counters are reduced by $c_{\min}$, and some counter with zero count (there must be at least one now) is assigned to a_i and given count $b_i - c_{\min}$. Following this, items with zero count are removed from T. Then FREQUENTℝ achieves the basic Heavy Hitter guarantee by observing that every subtraction of counter values for a given item coincides with the same subtraction to $m-1$ others, and all counter increments correspond to some b_i of a particular item. Therefore, the error in the count of any item is at most F_1/m.

We comment that a similar analysis to that provided in Section 3.4 applies, to demonstrate that these new counter algorithms give a tail guarantee. The main technical challenge is generalizing the definitions of x-prefix guaranteed and heavy tolerant algorithms in the presence of arbitrary real updates. We omit the detailed analysis from this presentation, and instead we state in summary:

Theorem 13. FREQUENTℝ *and* SPACESAVINGℝ *both provide k-tail guarantees with $A = B = 1$ over real-valued non-negative update streams.*

3.7.2 Merging Multiple Summaries

A consequence of sparse recovery is the fact that multiple summaries of separate streams can be merged together to create a summary of the union of the streams. More formally, consider ℓ streams, defining frequency distributions $f^{(1)} \dots f^{(\ell)}$ respectively. Given a summary of each stream produced by (the same) algorithm with m counters, the aim is to construct an accurate summary of $f = \sum_{j=1}^{\ell} f^{(j)}$.

Theorem 14. *Given summaries of each $f^{(j)}$ produced by a counter algorithm that provides a k-tail guarantee with constants (A, B), a summary of f can be obtained with a k-tail guarantee with constants $(3A, B + A)$.*

Proof. We construct a summary by first building a k-sparse vector $f'^{(j)}$ from the summary of $f^{(j)}$, with the guarantee of equation (3.2). By generating a stream corresponding to this

vector for each stream, and feeding this into the counter algorithm, we obtain a summary of the distribution $f' = \sum_{j=1}^{\ell} f'^{(j)}$. Now observe that from this we have an estimated frequency for any item i as c_i so that

$$|c_i - f_i| \leq \Delta = \Delta_{f'} + \sum_{j=1}^{\ell} \Delta_j$$

where each Δ_j is the error from summarizing $f^{(j)}$ by $f'^{(j)}$, while $\Delta_{f'}$ is the error from summarizing f'. For the analysis, we require the following bound:

Lemma 10. *For any n-dimensional vectors x and y,*

$$|F_1^{res(k)}(x) - F_1^{res(k)}(y)| \leq \|x - y\|_1$$

Proof. Let X denote the set of k largest entries of x, and Y the set of k largest entries of y. Let $\pi(i)$ determine any bijection from $i \in Y \backslash X$ to $\pi(i) \in X \backslash Y$. Then

$$\begin{aligned} F_1^{res(k)}(x) - F_1^{res(k)}(y) &= \sum_{i \notin X} x_i - \sum_{i \notin Y} y_i \\ &\leq \sum_{i \in Y \backslash X} x_{\pi(i)} - \sum_{i \in X \backslash Y} y_i + \sum_{i \notin (X \cup Y)} |x_i - y_i| \\ &= \sum_{i \notin Y} |x_i - y_i| \leq \sum_i |x_i - y_i| \leq \|x - y\|_1 \end{aligned}$$

Interchanging the roles of x and y gives the final result. □

This lets us place an upper bound on the first component of the error:

$$\Delta_{f'} \leq \frac{A}{m - Bk} F_1^{res(k)}(f') \leq \frac{A}{m - Bk}(F_1^{res(k)}(f) + \|f - f'\|_1)$$

where, by the triangle inequality and the proof of Theorem 8,

$$\|f - f'\|_1 \leq \sum_{j=1}^{\ell} \|f^{(j)} - f'^{(j)}\|_1 \leq \sum_{j=1}^{\ell} (3k\Delta_j + F_1^{res(k)}(f^{(j)}))$$

Since $\Delta_j \leq A F_1^{res(k)}(f^{(j)})/(m - Bk)$, the total error obeys

$$\Delta \leq \frac{A}{m - Bk} \left(F_1^{res(k)}(f) + \sum_{j=1}^{\ell} (3k\Delta_j + 2F_1^{res(k)}(f^{(j)})) \right)$$

We observe that

$$\sum_{j=1}^{\ell} F_1^{res(k)}(f^{(j)}) \leq F_1^{res(k)}\left(\sum_{j=1}^{\ell} f^{(j)}\right) = F_1^{res(k)}(f)$$

since $\sum_{j=1}^{\ell} F_1^{res(k)}(f^{(j)}) \leq \sum_{j=1}^{\ell} \sum_{i \notin T} f^{(j)}$ for any T such that $|T| = k$. So

$$\begin{aligned}\Delta &\leq \frac{A}{m - Bk}\left(3F_1^{res(k)}(f) + 3k\frac{A}{m - Bk}\left(F_1^{res(k)}(f)\right)\right) \\ &= \frac{3A}{m - Bk}\left(1 + \frac{Ak}{m - Bk}\right)F_1^{res(k)}(f)\end{aligned}$$

This can be analyzed as follows:

$$\begin{aligned}(m - Bk)^2 - (Ak)^2 &\leq (m - Bk)^2 \\ (m - Bk + Ak)(m - Bk - Ak) &\leq (m - Bk)^2 \\ 1 + \frac{Ak}{m - Bk} &\leq \frac{(m - Bk)}{m - (A + B)k} \\ \frac{3A}{m - Bk}\left(1 + \frac{Ak}{m - Bk}\right) &\leq \frac{3A}{m - (A + B)k}\end{aligned}$$

Hence, we have a $(3A, A + B)$ guarantee for the k-tail estimation. □ □

In particular, since the two counter algorithms analyzed have k tail guarantees with constants $(1, 1)$, their summaries can be merged in this way to obtain k tail summaries with constants $(3, 2)$. Equivalently, this means to obtain a desired error Δ, we need to pick the number of counters m to be at most a constant factor (three) times larger to give the same bound on merging multiple summaries as for a single summary.

3.8 Lower bound

The following theorem establishes a lower bound for the estimation error of any counter algorithm.

Theorem 15. *For any deterministic counter algorithm with m counters, for any k, $1 \leq k \leq m$, there exists some stream in which the estimation error of an element is at least $\frac{F_1^{res(k)}}{2m}$*

Proof. The proof is similar to that of Theorem 2 in [BKMT03]. For some integer X, consider two streams A and B. The streams share the same prefix of size $X(m+k)$, where

elements $a_1 \ldots a_{m+k}$ occur X times each. After the counter algorithm runs on this first part of each stream, only m elements can have non-zero counters. Assume without loss of generality that the other k elements are $a_1 \ldots a_k$.

Then stream A continues with elements $a_1 \ldots a_k$, while stream B continues with k other elements $z_1 \ldots z_k$ distinct from $a_1 \ldots a_{m+k}$. Both streams thus have total size $X(m+k)+k$.

For both streams, after processing the prefix of size $X(m+k)$, the algorithm has no record of any of the elements in the remaining parts of either of the streams. So the two remaining parts look identical to the algorithm and will yield the same estimates. Thus, for $1 \leq i \leq k$, $c_{a_i}(A) = c_{z_i}(B)$. But $f_{a_i}(A) = X + 1$ while $f_{z_i}(B) = 1$. The counter error for one of the two streams must be at least $X/2$. Note that $F_1^{res(k)}(A) = Xm$ and $F_1^{res(k)}(B) = Xm + k$; then the error is at least

$$\frac{X}{2} \geq \frac{F_1^{res(k)}}{2m + 2k/X}$$

As $X \to \infty$, this approaches our desired bound. □

Thus an algorithm that provides an error bound of $\frac{F_1^{res(k)}}{m-k}$ must use at least $(m-k)/2$ counters.

Chapter 4

Conclusions and open problems

In chapter 2 we introduced binary sparse matrices as valid measurement matrices for linear sketching. We showed that they work with the ℓ_1-minimization method; in addition, we introduced two faster iterative algorithms that use the same matrices. Finally, we presented experiments showing that these methods have practical value.

In chapter 3 we showed strong error bounds for counter algorithms. While they are not as versatile as linear sketching - finding application in a number of specific problems - these algorithm are efficient and space-optimal, using only $O(k)$ space to recover k-sparse approximations.

We discuss open problems and possible directions for future research. First, a faster way of implementing SSMP would be of importance; the current implementation yields good recoveries, but at the cost of increased running time compared to SMP. Second, the algorithms would be cleaner and more practical if they would not need an explicit sparsity parameter k, the best choice of which many times involves guesswork.

An interesting fact to notice is that any one of the presented methods can be used to recover a signal from *the same* linear sketch; the measurement matrix can be chosen without a priori knowledge of which algorithm will be used. A possible research direction is to find ways of combining two distinct methods (perhaps using the results of one as a starting point for the other) in order to obtain better recovery quality.

An important open problem is, of course, that of finding an explicit representation for expanders with optimal parameters. However, this has been a very studied problem; a solution would be a breakthrough with consequences in many other fields, like error

correcting codes or design of computer networks.

A related problem is that of finding an implicit expander representation which requires sublinear space; more precisely, one needs to be able to compute the neighbors of a vertex without explicitly maintaining this data for all vertices. This is imperative if linear sketching is to be used in data stream algorithms (e.g. the problem described in section 1.4.1) where by definition any solution must use sublinear space. An example of a possible construction one might use in practice is the following: generate d 2-independent hash functions $h_i : \{1 \dots n\} \to \{1 \dots \frac{m}{d}\}$ with $1 \leq i \leq d$ and let the i-th neighbor of a left vertex v be $\frac{m}{d}(i-1) + h_i(v)$. In practice, this construction appears to work as well as fully random matrices; however there is no theoretic basis for it.

Finally, while the linear programming method yields very good approximations, how much it can be improved even further is an interesting open problem. For example, the reweighting method of [CWB09] achieves better results by running the linear program multiple times. Message passing algorithms inspired from error correcting codes (see for example [APT09]) also have the potential to achieve better approximations than the linear programming method, even with decreased recovery times.

Appendix A

Mathematical facts and notations

In this appendix we introduce some of the mathematical notation used throughout this document.

A.1 Mathematical facts

A.1.1 Vector Norms

Let x be a vector in n-dimensional real space, $x \in \mathbb{R}^n$. The ℓ_p norm of vector x is defined as:

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

The ℓ_2 norm is thus the usual Euclidean distance. The ℓ_1 norm is simply the sum of the absolute values of the elements of x. Two related definitions are those of the ℓ_0 and ℓ_∞ norms:

$$\begin{aligned} \|x\|_0 &:= \lim_{p \to 0} \|x\|_p^p = \sum_{i=1}^{n} x_i^0 \qquad \text{(under the definition } 0^0 = 0) \\ \|x\|_\infty &:= \lim_{p \to \infty} \|x\|_p = \max(|x_1|, \ldots, |x_n|) \end{aligned}$$

The ℓ_0 norm of x is the number of non-zero elements of x, while the ℓ_∞ norm is the maximum absolute value of a coordinate of x.

A.1.2 Sparsity

In general, sparse vectors are vectors for which most of their components are zero. To quantitatively describe this, for an integer k we call a vector *k-sparse* if at most k of its components are non-zero. Formally,

Definition. *A vector $x \in \mathbb{R}^n$ is k-sparse iff $\|x\|_0 \leq k$.*

A.1.3 Norm inequalities

Theorem. *For any $0 < p < q \leq \infty$ it holds that $\|x\|_p \geq \|x\|_q$.*

A useful inequality is an upper-bound on higher norms relative to the ℓ_1 norm:

Theorem. *For any vector x and any norm p it holds that*

$$\|x\|_1 \leq \|x\|_p \cdot \|x\|_0^{1-1/p} \tag{A.1}$$

Proof. We use Hölder's inequality: for $1 \leq p, q \leq \infty$ such that $\frac{1}{q} + \frac{1}{p} = 1$, for any vectors f, g it holds that $\|fg\|_1 \leq \|f\|_p \cdot \|g\|_q$. We use this inequality with vectors $g = x$ and vector f where f_i is 1 if $x_i \neq 0$ or 0 otherwise, and the norm inequality follows directly. □

A.1.4 Sparse approximation guarantees

The ℓ_p/ℓ_p guarantee for a sparse approximation x^* to a vector x is

$$\|x - x^*\|_p \leq C\|x - x^{(k)}\|_p$$

The mixed ℓ_p/ℓ_1 guarantee is

$$\|x - x^*\|_p \leq \frac{C}{k^{1-1/p}}\|x - x^{(k)}\|_1$$

We discuss a few facts about the relationship between these guarantees.

Fact 3. *The ℓ_p/ℓ_1 and ℓ_p/ℓ_p guarantees are not directly comparable.*

This fact was pointed out in [CDD06]; we reproduce the proof here:

Proof. Note that in both guarantees the same term $\|x - x^*\|_p$ is bounded.

Let $a \in (0,1)$ be a constant; consider two vectors u and v as follows. Vector u has the first k coordinates equal to 1, coordinate $k+1$ equal to a, and the rest of coordinates 0. Vector v has the first k coordinates 1, and the rest of coordinates all equal to a.

For vector u, $\|u-u^{(k)}\|_p = \|u-u^{(k)}\|_1 = a$ and the bound given by the ℓ_p/ℓ_1 guarantee is smaller than the bound given by the ℓ_p/ℓ_p guarantee by a factor of $k^{1-1/p}$.

On the other hand, for vector v, $\|v-v^{(k)}\|_1 = (n-k)a$ and $\|v-v^{(k)}\|_p = (n-k)^{1/p}a$. In this case, the ℓ_p/ℓ_1 bound is larger than the ℓ_p/ℓ_p bound by a factor of $\left(\frac{n-k}{k}\right)^{1-1/p}$. □

Fact 4. *The ℓ_1/ℓ_1 and ℓ_p/ℓ_p guarantees with $p>1$ are not directly comparable.*

Proof. Consider the same u and v vectors as in the above proof.

For u, $\|u-u^{(k)}\|_1 = \|u-u^{(k)}\|_p = a$ so for this vector the ℓ_1/ℓ_1 guarantee implies the ℓ_p/ℓ_p guarantee with the same constant since the right-hand sides of the guarantees are identical and $\|u-u^*\|_p \le \|u-u^*\|_1$ for any vector $(u-u^*)$.

For v, $\|v-v^{(k)}\|_1 = (n-k)a$ and $\|v-v^{(k)}\|_p = (n-k)^{1/p}a$. The norm inequality (A.1) states that $\|v-v^*\|_1 \le \|v-v^*\|_1 \cdot \|v-v^*\|_0^{1-1/p}$. Thus for recovered vectors v^* such that $(v-v^*)$ is at most $(n-k)$-sparse, then the ℓ_p/ℓ_p guarantee implies the ℓ_1/ℓ_1 guarantee with the same constant. For general vectors v^*, the ℓ_p/ℓ_p guarantee with constant C implies the ℓ_1/ℓ_1 guarantee with constant $C\left(\frac{n}{n-k}\right)^{1-1/p} \approx C$ when $k \ll n$. □

Fact 5. *If the recovered signal x^* is $O(k)$ sparse, then the ℓ_p/ℓ_1 guarantee with constant C implies the ℓ_1/ℓ_1 guarantee with constant $O(C)$.*

Proof. Assume that x^* is Ak-sparse. Let S be the set which includes the nonzero coordinates of x^* as well as the top k (in absolute value) coordinates of x, with $k \le |S| \le (A+1)k$. We use S^c for the complement of set S and x_S for the vector obtained from x by keeping only the coordinates in S. Let $e = x - x^*$. The ℓ_p/ℓ_1 guarantee states that

$$\|e\|_p \le \frac{C}{k^{1-1/p}}\|x - x^{(k)}\|_1$$

From this and the norm inequality (A.1)

$$\|e_S\|_1 \le \|e_S\|_p((A+1)k)^{1-1/p} \le \|e\|_p((A+1)k)^{1-1/p} \le C(A+1)^{1-1/p}\|x-x^{(k)}\|_1$$

Notice that since S includes the top k coordinates of x

$$\|e_{S^c}\|_1 = \|x_{S^c}\|_1 \leq \|x - x^{(k)}\|_1$$

We then have

$$\|e\|_1 = \|e_S\|_1 + \|e_{S^c}\|_1 \leq (1 + C(A+1)^{1-1/p})\|x - x^{(k)}\|_1$$

which is the ℓ_1/ℓ_1 guarantee with constant $(1 + C(A+1)^{1-1/p})$. □

Fact 6. *The ℓ_2/ℓ_2 guarantee cannot be obtained deterministically (for all signals x simultaneously) unless the number of measurements is linear, i.e. $m = \Omega(n)$.*

This was proved in [CDD06].

A.1.5 Proof that random graphs are expanders

Theorem 16. *There exist graphs $G = (U, V, E)$ that are (k, d, ϵ)-expanders with $d = O(\log(|U|/k)/\epsilon)$ and $|V| = O(k \log(|U|/k)/\epsilon^2)$.*

Proof. Consider graphs $G = (U, V, E)$ with $|U| = n$ and $|V| = m$. Let $d = \ln(ne^2/k)/\epsilon$ and $m = e^2 k \ln(ne^2/k)/\epsilon^2$. We show that a random graph G is with constant probability a (k, d, ϵ)-unbalanced expander[1]. A random graph is generated by randomly and independently choosing d neighbors for each left vertex.

Consider any of the $\binom{n}{s}$ left-vertex sets of size $s \leq k$. The s vertices have d neighbors each; consider a sequence containing the ds vertex indices. For G to fail to be an expander on this set, at least ϵds of these values must be "repeats", i.e. identical to some earlier value in the sequence. The probability that a given neighbor is a repeat is at most $\frac{ds}{m}$. By the union bound, the probability that G fails to expand at least one of these sets is at most

$$\begin{aligned}
\binom{n}{s}\binom{ds}{\epsilon ds}\left(\frac{ds}{m}\right)^{\epsilon ds} &\leq \left(\frac{ne}{s}\right)^s \left(\frac{e}{\epsilon}\right)^{\epsilon ds} \left(\frac{s\epsilon}{ke^2}\right)^{\epsilon ds} \\
&\leq \left(\frac{ne}{s}\right)^s \left(\frac{s}{ek}\right)^{\epsilon ds} \leq \left(\frac{ne}{s} e^{-\epsilon d} \left(\frac{s}{k}\right)^{\epsilon d}\right)^s \\
&\leq \left(\frac{ne}{s}\frac{k}{ne^2}\left(\frac{s}{k}\right)^{\epsilon d}\right)^s \leq \left(\frac{1}{e}\left(\frac{s}{k}\right)^{\epsilon d - 1}\right)^s \leq e^{-s}
\end{aligned}$$

[1]Note that this analysis is not tight in terms of constants or success probability.

Thus the probability that G is not a (k, d, ϵ)-expander is at most

$$\sum_{s=1}^{k} e^{-s} < \frac{1}{e} \sum_{s=0}^{\infty} e^{-s} = \frac{1}{e} \frac{1}{1 - 1/e} = \frac{1}{e - 1} < 0.59$$

□

A.2 Common notation

We reproduce some of the notation frequently used throughout this paper:

e_i : the vector with all components zero except the i-th component which is 1; equivalently, the i-th line of an identity matrix. The size of the vector is usually understood from context.

$H_k[x]$: thresholding operator, the result of which is a k-sparse vector which retains only the k largest (in absolute value) components of x.

$x^{(k)} = \text{argmin}_{k-\text{sparse } x'} \|x - x'\|_p$: the best k-sparse approximation of x. While the solution might not be unique, $x^{(k)} = H_k[x]$ is always one optimal solution (regardless of the norm p).

x_S : the $|S|$-dimensional projection of x on coordinates in $S \subset \{1 \dots n\}$, i.e. the vector obtained from x by zeroing out all coordinates except those in set S.

S^c : the complement of set S, so that $x = x_S + x_{S^c}$.

$\Gamma_G(S)$: the set of neighbors in graph G of nodes in set S. For a single vertex u, we use $\Gamma(u)$ as a shorthand for $\Gamma(\{u\})$.

We use the following notations in chapter 3:

$F_1 = \sum f_i = \|f\|$: sum of all frequencies, i.e. total number of elements in the stream.

$F_p = \sum_i f_i^p = \|f\|_p^p$: sum of frequencies to the p-th power.

$F_1^{res(k)} = \|f - H_k[f]\|_1 = \|f - f^{(k)}\|_1$: sum of all but top k frequencies.

Appendix B

Sample recovered images

B.1 Peppers image, $m = 17000$

Original

LP

SNR: 23.22 time: 284s

SMP

$T=4,\ k=1000,\ \xi=0.6$

SNR: 20.82 time: 0.09s

SMP

$T=8,\ k=1000,\ \xi=0.6$

SNR: 21.51 time: 0.31s

SMP

$T=16,\ k=1250,\ \xi=0.6$

SNR: 21.90 time: 0.66s

SMP

$T=64,\ k=1250,\ \xi=0.6$

SNR: 22.07 time: 2.38s

SSMP

$S=4000,\ T=4,\ k=1700$

SNR: 21.86 time: 1.59s

SSMP

$S=8000,\ T=4,\ k=1700$

SNR: 22.16 time: 2.70s

SSMP

$S=8000,\ T=16,\ k=1700$

SNR: 22.60 time: 11.17s

SSMP

$S=16000,\ T=32,\ k=1750$

SNR: 23.10 time: 48.14s

SSMP

$S=16000,\ T=64,\ k=1750$

SNR: 23.26 time: 102s

SSMP

$S=32000,\ T=256,\ k=1700$

SNR: 23.56 time: 671s

B.2 Boat image, $m = 10000$

Original

LP

SNR: 20.66 time: 295s

SMP

$T=4,\ k=250,\ \xi=0.6$

SNR: 18.56 time: 0.13s

SMP

$T=8,\ k=250,\ \xi=0.6$

SNR: 18.68 time: 0.23s

SMP

$T=16,\ k=250,\ \xi=0.6$

SNR: 18.63 time: 0.53s

SMP

$T=64,\ k=500,\ \xi=0.6$

SNR: 19.09 time: 2.36s

SSMP

$S=4000,\ T=4,\ k=500$

SNR: 19.00 time: 2.98s

SSMP

$S=8000,\ T=4,\ k=500$

SNR: 18.97 time: 5.47s

SSMP

$S=8000,\ T=16,\ k=500$

SNR: 19.43 time: 21.53s

SSMP

$S=16000,\ T=32,\ k=500$

SNR: 19.22 time: 83.92s

SSMP

$S=16000,\ T=64,\ k=500$

SNR: 19.43 time: 172s

SSMP

$S=32000,\ T=256,\ k=500$

SNR: 19.48 time: 1314s

B.3 Boat image, $m = 25000$

Original

LP

SNR: 25.38 time: 333s

SMP

$T=4,\ k=1875,\ \xi=0.6$

SNR: 22.14 time: 0.13s

SMP

$T=8,\ k=1875,\ \xi=0.6$

SNR: 23.05 time: 0.19s

SMP

$T=16,\ k=1875,\ \xi=0.6$

SNR: 23.67 time: 0.50s

SMP

$T=64,\ k=1875,\ \xi=0.6$

SNR: 23.13 time: 2.27s

SSMP

$S=4000,\ T=4,\ k=1875$

SNR: 24.11 time: 1.36s

SSMP

$S=8000,\ T=4,\ k=2500$

SNR: 24.36 time: 2.48s

SSMP

$S=8000,\ T=16,\ k=2500$

SNR: 25.25 time: 10.02s

SSMP

$S=16000,\ T=32,\ k=2500$

SNR: 25.63 time: 37.45s

SSMP

$S=16000,\ T=64,\ k=3750$

SNR: 25.98 time: 72.36s

SSMP

$S=32000,\ T=256,\ k=3750$

SNR: 26.37 time: 552s

Bibliography

[ABW03] A. Arasu, S. Babu, and J. Widom. Cql: A language for continuous queries over streams and relations. *Proceedings of the 9th DBPL International Confenrence on Data Base and Programming Languages*, pages 1–11, 2003.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. System Sci.*, 58(1):137–147, 1999.

[APT09] M. Akçakaya, J. Park, and V. Tarokh. Compressive sensing using low density frames. *Submitted to IEEE Trans. on Signal Processing*, 2009.

[BCIS09] R. Berinde, G. Cormode, P. Indyk, and M. Strauss. Space-optimal heavy hitters with strong error bounds. *Proceedings of the ACM Symposium on Principles of Database Systems*, 2009.

[BGI+08] R. Berinde, A. Gilbert, P. Indyk, H. Karloff, and M. Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. *Allerton*, 2008.

[BGS01] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. *Proceedings of the 2nd IEEE MDM International Conference on Mobile Data Management*, pages 3–14, 2001.

[BI08] R. Berinde and P. Indyk. Sparse recovery using sparse random matrices. *MIT-CSAIL Technical Report*, 2008.

[BI09] R. Berinde and P. Indyk. Sequential Sparse Matching Pursuit. *Allerton*, 2009.

[BIR08] R. Berinde, P. Indyk, and M. Ružić. Practical near-optimal sparse recovery in the L1 norm. *Allerton*, 2008.

[BKMT03] P. Bose, E. Kranakis, P. Morin, and Y. Tang. Bounds for frequency estimation of packet streams. *Proceedings of the 10th International Colloquium on Structural Information and Communication Complexity*, pages 33–42, 2003.

[BR99] K. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. *Proceedings of 1999 ACM SIGMOD*, pages 359–370, 1999.

[CCFC02] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *ICALP*, 2002.

[CCM07] A. Chakrabarti, G. Cormode, and A. McGregor. A near-optimal algorithm for computing the entropy of a stream. In *SODA*, 2007.

[CDD06] A. Cohen, W. Dahmen, and R. DeVore. Compressed sensing and best k-term approximation. *Preprint*, 2006.

[CDS99] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by Basis Pursuit. *SIAM J. Sci. Comput.*, 20(1):33–61, 1999.

[CH08] G. Cormode and M. Hadjieleftheriou. Finding frequent items in data streams. *PVLDB*, 1(2):1530–1541, 2008.

[Che09] M. Cheraghchi. Noise-resilient group testing: Limitations and constructions. *To appear in 17th International Symposium on Fundamentals of Computation Theory*, 2009.

[CKMS03] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Finding hierarchical heavy hitters in data streams. *Proceedings of the 29th ACM VLDB International Conference on Very Large Data Bases*, pages 464–475, 2003.

[CM04] G. Cormode and S. Muthukrishnan. Improved data stream summaries: The count-min sketch and its applications. *FSTTCS*, 2004.

[CM06] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for Compressed Sensing. In *Proc. 40th Ann. Conf. Information Sciences and Systems*, Princeton, Mar. 2006.

[CR05] E. J. Candès and J. Romberg. *ℓ_1-MAGIC: Recovery of Sparse Signals via Convex Programming*, 2005. Available at: `http://www.acm.caltech.edu/l1magic`.

[CRT06] 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.

[CWB09] E. J. Candès, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by rewieghted $\ell(1)$ minimization. *Journal of Fourier Analysis and Applications*, 2009.

[DDT+08] M. Duarte, M. Davenport, D. Takhar, J. Laska, T. Sun, K. Kelly, and R. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 2008.

[DeV07] R. DeVore. Deterministic constructions of compressed sensing matrices. *preprint*, 2007.

[DH93] Ding-Zhu Du and Frank K. Hwang. *Combinatorial group testing and its applications*. World Scientific, 1993.

[DM08] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing: Closing the gap between performance and complexity. *Arxiv:0803.0811*, 2008.

[DMT07] C. Dwork, F. McSherry, and K. Talwar. The price of privacy and the limits of LP decoding. *STOC*, 2007.

[DOM02] E. Demaine, A. López Ortiz, and J. Munro. Frequency estimation of internet packet streams with limited space. *Proceedings of the 10th ESA Annual European Symposium on Algorithms*, pages 348–360, 2002.

[Don04] D. L. Donoho. Compressed sensing. Unpublished manuscript, Oct. 2004.

[Don06] D. L. Donoho. Compressed Sensing. *IEEE Trans. Info. Theory*, 52(4):1289–1306, Apr. 2006.

[Dor43] R. Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 1943.

[DT06] D. L. Donoho and J. Tanner. Thresholds for the recovery of sparse solutions via l1 minimization. *Proc. of the 40th Annual Conference on Information Sciences and Systems (CISS)*, 2006.

[DWB05] M. F. Duarte, M. B. Wakin, and R. G. Baraniuk. Fast reconstruction of piecewise smooth signals from random projections. In *Proc. SPARS05*, 2005.

[EV01] C. Estan and G. Verghese. New directions in traffic measurement and accounting. *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[EV03] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems*, 2003.

[FSGM+98] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. Ullman. Computing iceberg queries efficiently. *Proceedings of the 24th ACM VLDB International Conference on Very Large Data Bases*, pages 299–310, 1998.

[GGI+02a] A. Gilbert, S. Guha, P. Indyk, M. Muthukrishnan, and M. Strauss. Near-optimal sparse Fourier representations via sampling. *STOC*, 2002.

[GGI+02b] A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *ACM Symposium on Theoretical Computer Science*, 2002.

[GIS08] A. C. Gilbert, M. A. Iwen, and M. J. Strauss. Group testing and sparse signal recovery. *42nd Asilomar Conference on Signals, Systems, and Computers, Monterey, CA*, 2008.

[GKMS03] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. One-Pass Wavelet Decompositions of Data Streams. *IEEE Trans. Knowl. Data Eng.*, 15(3):541–554, 2003.

[GLR08] V. Guruswami, J. Lee, and A. Razborov. Almost Euclidean subspaces of l_1 via expander codes. *SODA*, 2008.

[Gro06] Rice DSP Group. Compressed sensing resources. *Available at:* `http://www.dsp.ece.rice.edu/cs/`, 2006.

[Gro08] Rice DSP Group. Rice single-pixel camera project. *Available at:* `http://dsp.rice.edu/cscamera/`, 2008.

[GSTV06] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin. Algorithmic linear dimension reduction in the ℓ_1 norm for sparse vectors. Submitted for publication, 2006.

[GSTV07] 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.

[GUV07] V. Guruswami, C. Umans, and S. P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. In *IEEE Conference on Computational Complexity (CCC 2007)*, pages 96–108, 2007.

[HPDW01] J. Han, J. Pei, G. Dong, and K. Wang. Efficient computation of iceberg cubes with complex measures. *Proceedings of 2001 ACM SIGMOD*, pages 1–12, 2001.

[HSST05] J. Hershberger, N. Shrivastava, S. Suri, and C. D. Tóth. Space complexity of hierarchical heavy hitters in multi-dimensional streams. *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 338–347, 2005.

[Ind04] P. Indyk. Algorithms for dynamic geometric problems over data streams. In *STOC*, 2004.

[Ind07] P. Indyk. Sketching, streaming and sublinear-space algorithms. *Graduate course notes, available at:* *`http://stellar.mit.edu/S/course/6/fa07/6.895`*, 2007.

[Ind08] P. Indyk. Explicit constructions for compressed sensing of sparse signals. *SODA*, 2008.

[IR08] P. Indyk and M. Ružić. Near-optimal sparse recovery in the L1 norm. *FOCS*, 2008.

[KSP03] R. M. Karp, S. Shenker, and C. H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Transactions on Database Systems (TODS)*, 28(1):51–55, 2003.

[KT07] B. S. Kashin and V. N. Temlyakov. A remark on compressed sensing. *Preprint*, 2007.

[MAA05] A. Metwally, D. Agrawal, and A.E. Abbabi. Efficient computation of frequent and top-k elements in data streams. *International Conference on Database Theory*, pages 398–412, 2005.

[Man92] Y. Mansour. Randomized interpolation and approximation of sparse polynomials. *ICALP*, 1992.

[MG82] J. Misra and D. Gries. Finding repeated elements. *Science of Computer Programming*, 2:142–152, 1982.

[MM02] G.S. Manku and R. Motwani. Approximate frequency counts over data streams. In *VLDB*, pages 346–357, 2002.

[Mut03] S. Muthukrishnan. Data streams: Algorithms and applications. *Invited talk at SODA 2003; available at:* `http://athos.rutgers.edu/~muthu/stream-1-1.ps`, 2003.

[Mut05] S.M. Muthukrishnan. *Data Streams: Algorithms and Applications*. Foundations and Trends in Theoretical Computer Science, 2005.

[NT08] D. Needell and J. A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harmonic Anal.*, 2008. To appear.

[NV09] D. Needell and R. Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, 9(3):317–334, 2009.

[RV06] M. Rudelson and R. Veshynin. Sparse reconstruction by convex relaxation: Fourier and Gaussian measurements. In *Proc. 40th Ann. Conf. Information Sciences and Systems*, Princeton, Mar. 2006.

[SBAS04] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. *Proceedings of the 2nd International Conference on Embedded Network Sensor Systems*, pages 239–249, 2004.

[SBB06a] S. Sarvotham, D. Baron, and R. G. Baraniuk. Compressed sensing reconstruction via belief propagation. *Technical Report ECE-0601, Electrical and Computer Engineering Department, Rice University*, 2006.

[SBB06b] S. Sarvotham, D. Baron, and R. G. Baraniuk. Sudocodes - fast measurement and reconstruction of sparse signals. *IEEE International Symposium on Information Theory*, 2006.

[Tao07] T. Tao. Open question: deterministic UUP matrices. *Weblog at:* `http://terrytao.wordpress.com`, 2007.

[TG05] J. A. Tropp and A. C. Gilbert. Signal recovery from partial information via Orthogonal Matching Pursuit. Submitted to *IEEE Trans. Inform. Theory*, April 2005.

[TLW+06] Dharmpal Takhar, Jason Laska, Michael B. Wakin, Marco F. Duarte, Dror Baron, Shriram Sarvotham, Kevin Kelly, and Richard G. Baraniuk. A new compressive imaging camera architecture using optical-domain compression. In *Proc. IS&T/SPIE Symposium on Electronic Imaging*, 2006.

[XH07] W. Xu and B. Hassibi. Efficient compressive sensing with determinstic guarantees using expander graphs. *IEEE Information Theory Workshop*, 2007.

[Zip49] G. Zipf. *Human Behavior and The Principle of Least Effort*. Addison-Wesley, 1949.