Sparse Recovery *

Eric Price

September 22, 2020

Abstract

Many real-world signals are approximately *sparse*, meaning that a small fraction of the coordinates contain almost all the signal mass; examples include images, audio, and any signals drawn from Zipfian, power-law, or log-normal distributions. If a signal $x \in \mathbb{R}^n$ is approximately k-sparse, then ideally the complexity of estimating or manipulating x should scale primarily with k rather than n.

Such *sparse recovery* algorithms are possible for a variety of different problem variants, corresponding to different modalities of measuring x and different guarantees on the estimation error. In this chapter we will consider streaming algorithms, compressed sensing, and sparse Fourier transforms, as well as extensions to low-rank matrix recovery.

1 Sparse Recovery

Imagine that you are tallying the results of an election by hand, and would like to find the top few candidates. You might maintain a sheet of paper recording the count for each candidate, while you pass through the giant stack of ballots. But write-in candidates make this challenging in a large election: the recording sheet would run out of space from votes for "candidates" like Batman or Bart Simpson. You would be fine ignoring such joke candidates with very few votes, but you don't want to miss a significant write-in candidate—and you don't want to miss them even if all their votes happened late in the day, at the bottom of the stack of ballots, after your tally sheet has run out of space. An algorithm due to Misra and Gries (1982), to be covered in the next section, offers a solution that only uses a small amount of space, at the cost of giving an *approximate* answer. If there are only k "real" candidates, and all the other candidates are rare, the approximation error will be small.

This property, that a small fraction of coordinates contains a large fraction of the mass, has been empirically observed for signals in many different domains. It follows from popular rules of thumb such as Zipf's law and the 80/20 rule, as well as popular generative models that yield power law or lognormal distributions. In Figure 1 we show a few examples of this phenomenon: in music (the Fourier transform of a short snippet), images (represented in a basis such as the Haar wavelet), and networks (number of inlinks per page). These different

*Excerpt from *Beyond the Worst-Case Analysis of Algorithms*, edited by Tim Roughgarden.

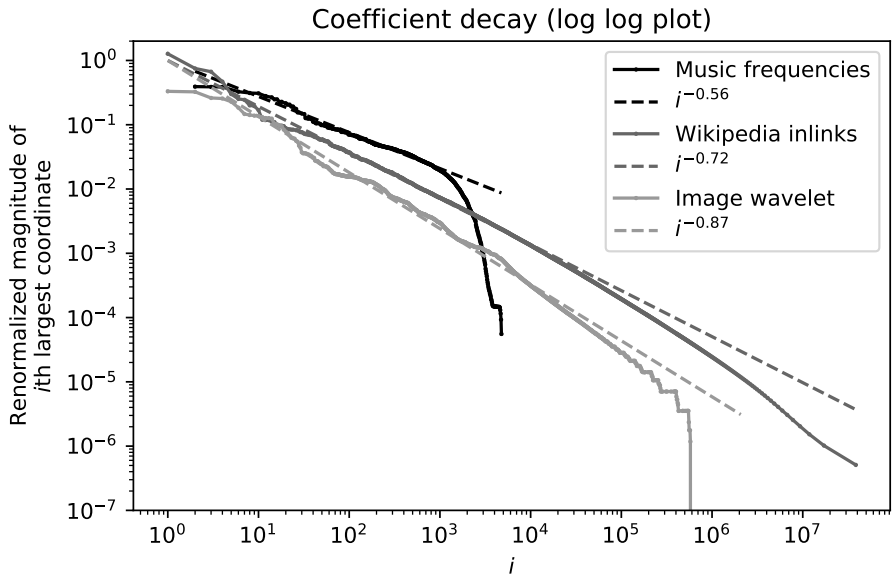


Figure 1: Coefficient decay in three example signals of different domains. In each example, the ith largest coordinate has magnitude decaying as $i^{-\alpha}$ for some $\alpha \in (0.5, 1)$. The audio data contains the frequencies in a 1/10 second clip of a popular music video; the image data is the Haar wavelet representation of one frame of this video; the graph data is the number of inlinks per page on English Wikipedia.

domains vary in how quickly the coefficients decay, but they all have the same qualitative behavior: the ith largest coordinate has magnitude roughly proportional to $i^{-\alpha}$ for some $\alpha \in (0.5, 1)$ for small i, followed by even faster decay for large i.

None of the results presented in this chapter rely on any distributional assumptions on signals, and require only that the signal to be recovered or manipulated is (approximately) sparse. This assumption is analogous to the stability definitions of Chapters 5 and 6, except with "meaningful solutions" now identified with "(approximate) sparsity." Most of the algorithms in this chapter provide input-by-input guarantees, parameterized by how close the unknown signal is to being k-sparse. As with the parameterized guarantees in Chapters 1 and 2, these will be non-trivial only when the parameter is small (i.e., when the signal is approximately k-sparse).

Outline of the chapter. In Section 2 we give a streaming algorithm for sparse recovery. In Section 3 we present two *linear sketching* algorithms for the problem. Linear sketching algorithms have several advantages over other streaming algorithms, and the second algorithm also achieves a stronger "ℓ_2" approximation guarantee for sparse recovery. In Section 4 we turn to *compressed sensing* algorithms. Compressed sensing is essentially the same problem as sparse recovery with linear sketching, but studied by a different community for a different purpose, leading to significant differences in techniques and subtler differences in goals. Section 5 contains a lower bound that matches the algorithms of Sections 3 and 4. Section 6 presents some more involved results in the area. Finally, Section 7 shows how sparse recovery

techniques extend to low-rank matrix estimation.

Notation. For any $x \in \mathbb{R}^n$ and $k \in [n]$, we use $H_k(x)$ to denote the k-sparse vector in $\mathbb{R}^n$ that sets all but the largest k entries (in magnitude) of x to zero.

2 A Simple Insertion-Only Streaming Algorithm

Algorithm 1 FrequentElements heavy hitters algorithm

```
1: function FrequentElements(Stream, k)
2:     d ← Dictionary()
3:     for u in Stream do
4:         if u in d then
5:             d[u] += 1
6:         else if d has less than k keys then
7:             d[u] ← 1
8:         else
9:             d[u'] −= 1    ∀u' ∈ d.
10:            Remove keys of d that now map to zero
11:        end if
12:    end for
13:    return d
14: end function
```

The election counting example is one example of a *data stream*. An (insertion-only) data stream consists of a long series of items

$$u_1, u_2, u_3, \ldots, u_N \in [n]$$

This stream represents the count vector $x \in \mathbb{R}^n$ given by

$$x_i = |\{j : u_j = i\}|.$$

The goal of sparse recovery (also known as heavy hitters) in this context is to approximate x while scanning through u, while storing much less than n or $N = \|x\|_1$ space (where $\|x\|_p := (\sum_i x_i^p)^{1/p}$ is the ℓ_p-norm).

The straightforward method for estimating x is to store it in a dictionary (a.k.a. associative array or map). We would start with an empty dictionary d, and for every element u that appears in the stream we increment $d[u]$ (with newly added elements set to 1). The problem with this method is that the space used is the total number of distinct elements in the stream, which could be as large as n.

The FrequentElements algorithm, presented in Algorithm 1 and due to Misra and Gries (1982), is a simple twist on the straightforward approach. The only difference is that we pick a parameter k (think, perhaps, $k = \sqrt{n}$) and, if incrementing $d[u]$ would make d have more than k keys, we instead *subtract* 1 from the counter of every key in the dictionary—and

if that brings a counter to zero, the corresponding key is removed. The space usage is then $\Theta(k)$ words and the error in the final count of every element is, at most, the total number of times this subtraction occurs. Since each subtraction removes k from the sum of the values in d, while each addition adds only 1 and the sum of values remains nonnegative, the subtraction step can happen at most a $1/(k+1)$ fraction of the stream steps. Thus:

Lemma 2.1. *The estimates $\widehat{x}_u = d[u]$ given by the* FrequentElements *algorithm satisfy*

$$x_u - \frac{1}{k+1}\|x\|_1 \leq \widehat{x}_u \leq x_u$$

for every element u.

One can also get a more refined bound that is significantly stronger in a sparse setting. If a few elements really do dominate the stream, those elements will end up with large values, which further constrains the number of deletions. One way to bound this is to consider as a potential function the sum of the entries of d that do not correspond to the $k/2$ largest entries of x. This potential is nonnegative at all times, only increases by 1 at a time, and does so at most $\|x - H_{k/2}(x)\|_1 \leq \|x\|_1$ times; on the other hand, each subtraction removes at least $k/2$ from this potential, so the total number of subtractions is at most $\|x - H_{k/2}(x)\|_1 \cdot \frac{2}{k}$. Thus:

Lemma 2.2. *The estimates $\widehat{x}_u = d[u]$ given by the* FrequentElements *algorithm satisfy*

$$x_u - \frac{2}{k}\|x - H_{k/2}(x)\|_1 \leq \widehat{x}_u \leq x_u$$

for every element u.

Which of Lemmas 2.1 and 2.2 more accurately characterizes the performance of FrequentElements depends on the sparsity of x. The sparsity-aware bound of Lemma 2.2 gives a better asymptotic bound on the error in terms of k when the frequencies decay faster than Zipf's law (in which the ith most common element having frequency proportional to $1/i$). When the frequencies decay slower, however, $\|x - H_{k/2}(x)\|_1 \approx \|x\|_1$ for $k \ll n$ so Lemma 2.1's better constant factors give a better bound.

3 Handling Deletions: Linear Sketching Algorithms

The FrequentElements algorithm is designed for *insertion-only* streams, where items arrive in sequence and never leave. A more general, and more challenging, setting is that of *turnstile* streams, where items can be both inserted and deleted. The name evokes an amusement park: you want to study the people who are currently inside the park, while only tracking people as they enter and leave through turnstiles. An important subclass is the *strict* turnstile stream, wherein the final vector x has nonnegative values (e.g., people cannot leave without arriving).

In Algorithm 2 we present two algorithms for solving sparse recovery in turnstile streams: CountMinSketch (Cormode and Muthukrishnan, 2005) and CountSketch (Charikar

Algorithm 2 CountMinSketch in black / CountSketch in black and gray

function CountMinSketch/CountSketch(Stream, B, R)
  Pick $h_1, \ldots, h_R : [n] \to [B]$ pairwise independent hash functions.
  Pick $s_1, \ldots, s_R : [n] \to \{-1, 1\}$ pairwise independent hash functions.
  $y_i^{(r)} \leftarrow 0 \quad \forall i \in [B], r \in [R]$
  for (u, a) in Stream **do** ▷ Corresponding to stream update $x_u \leftarrow x_u + a$
    for $r \in [R]$ **do**
      $y_{h_r(u)}^{(r)}$ += $a \cdot s_r(u)$.
    end for
  end for
  for $u \in [n]$ **do**
    $\widehat{x}_u \leftarrow \min_{r \in [R]} y_{h_r(u)}$. ▷ (CountMinSketch only)
    $\widehat{x}_u \leftarrow \text{median}_{r \in [R]} y_{h_r(u)} \cdot s_r(u)$. ▷ (CountSketch only)
  end for
  return $\widehat{x}$
end function

et al., 2002). The two algorithms are almost identical, with CountSketch having a few more pieces; these extra parts are displayed in gray, and should be ignored to read the CountMinSketch algorithm.

It turns out that almost every turnstile streaming algorithm can be implemented as a *linear sketch*. In a linear sketch, you store $y = Ax$ for some (possibly randomized) matrix $A \in \mathbb{R}^{m \times n}$. This can easily be maintained under streaming updates: when an element is inserted or deleted, you simply add or subtract the corresponding column of A from the sketch y. The space used by the linear sketch is m words to store y, plus the size of the random seed to produce A; and for the algorithms we will consider, the seed is small so this is essentially m. Another benefit of linear sketching algorithms over insertion-only streaming is *mergability*: you can split the stream into pieces (say, multiple routers), sketch the pieces individually, then add up the results to get the sketch for the full stream. One can observe that CountMinSketch/CountSketch are linear sketches. In particular, the final value stored in each coordinate $y_j^{(r)}$ is

$$y_j^{(r)} = \sum_{u=1}^{n} 1_{h_r(u)=j} s_r(u) \cdot x_u \tag{1}$$

which is a linear function of x.

3.1 The Count-Min Sketch: an ℓ_1 Guarantee

The idea behind CountMinSketch is that if we had unlimited space, we'd just store a single hash table with the counts for all items in the stream. If we instead store a hash table of much smaller size $B = O(k)$, there will be collisions. Standard methods for resolving those collisions, like linked lists, would again need linear space in the number of distinct items. But what happens if we don't resolve collisions at all, and just store in each hash cell the total number of elements that hash there?

Given such a "hash table", we can estimate the count for an item by the value in the cell it hashes to. For strict turnstile streams, this is an overestimate of the true answer: it contains the true count plus the counts of colliding elements. But any other element has only a $1/B$ chance of colliding in the hash table, so the expected error is at most $\|x\|_1/B$. This would be a decent bound comparable to Lemma 2.1, except that it is only in expectation for each element. Almost surely *some* element will have much higher error—indeed, there is no way to distinguish between the heavy hitters and the (small fraction, but still numerous) other elements that happen to collide with them.

To fix this, CountMinSketch repeats the process with $R = O(\log n)$ different hash tables. Since each hash table gives an overestimate, the final estimate of an element is the minimum estimate from any repetition. This achieves the following:

Theorem 3.1. *If x has nonnegative entries, then* CountMinSketch*, when run with $B \geq 4k$ and $R \geq 2\log_2 n$, returns an $\widehat{x}$ that satisfies*

$$x_u \leq \widehat{x}_u \leq x_u + \frac{1}{k}\|x - H_k(x)\|_1$$

for all u with $1 - 1/n$ probability.

The statement is very similar to the FrequentElements bound in Lemma 2.2. It is an overestimate rather than an underestimate, but otherwise the error bound is identical up to scaling k by 2. Unlike FrequentElements, CountMinSketch can handle deletions, but this comes at a cost: CountMinSketch uses $O(k \log n)$ words of space rather than $O(k)$, it is randomized, and the time required for computing $\widehat{x}$ at the end of the stream is $O(n \log n)$ rather than $O(k)$ because every coordinate x_u must be estimated to find the largest k. The first two issues cannot be avoided for "typical" values of $k \in (n^{0.01}, n^{0.99})$. In Section 5 we will show that $\Omega(k \log n)$ words of space are necessary to handle deletions, and randomization is needed to achieve $o(\min(k^2, n))$ words (Ganguly, 2008). The recovery time, however, can be improved; see the bibliographic notes for details.

Proof of Theorem 3.1. Define $\widehat{x}^{(r)}$ by $\widehat{x}^{(r)}_u = y_{h_r(u)}$ for each u, so that $\widehat{x}_u = \min_r \widehat{x}^{(r)}_u$. Let $H \subseteq [n]$ contain the largest k coordinates of x, known as the "heavy hitters". Then

$$0 \leq \widehat{x}^{(r)}_u - x_u = \sum_{\substack{h_r(v)=h_r(u)\\ v\neq u}} x_v = \underbrace{\sum_{\substack{v\in H\\ h_r(v)=h_r(u)\\ v\neq u}} x_v}_{E_H} + \underbrace{\sum_{\substack{v\notin H\\ h_r(v)=h_r(u)\\ v\neq u}} x_v}_{E_L}. \tag{2}$$

For u to be estimated badly, either E_H or E_L must be large. E_H represents the error u receives from colliding with heavy hitters. This is usually zero, because there aren't too many heavy hitters. E_L is the error from non–heavy-hitters. This is likely nonzero, but is small in expectation. Formally, we have that:

$$\Pr[E_H > 0] \leq \Pr[\exists v \in H \setminus \{u\} : h_r(v) = h_r(u)] \leq \frac{k}{B} \leq \frac{1}{4} \tag{3}$$

by our choice of $B \geq 4k$. We also have that

$$\mathbb{E}[E_L] = \sum_{\substack{v \in [n] \setminus H \\ v \neq u}} x_v \cdot \Pr[h(v) = h(u)] \leq \sum_{v \in [n] \setminus H} x_v \cdot \frac{1}{B} = \|x - H_k(x)\|_1 / B. \tag{4}$$

Hence by Markov's inequality,

$$\Pr[E_L > \|x - H_k(x)\|_1 / k] \leq \frac{k}{B} \leq \frac{1}{4}$$

so by a union bound, independently for each r

$$\Pr[\widehat{x}_u^{(r)} - x_u > \|x - H_k(x)\|_1 / k] \leq \frac{1}{2}. \tag{5}$$

Therefore because $R \geq 2 \log_2 n$,

$$\Pr[\widehat{x}_u - x_u > \|x - H_k(x)\|_1 / k] \leq \frac{1}{2^R} \leq \frac{1}{n^2}.$$

Taking a union bound over u gives the result. □

Negative entries and CountMedianSketch. The CountMinSketch algorithm relies on the strict turnstile assumption that the final vector x has only nonnegative coordinates. If the entries of x may be negative, one can simply replace the min on line 11 with a median and increase B and R by constant factors. By increasing B, the failure event (5) will have failure probability $2k/B < 1/2$. Then a Chernoff bound can show that with high probability *most* iterations r will not fail, and hence the median estimate is good. This algorithm is known as the CountMedianSketch, and achieves the same $\frac{1}{k}\|x - H_k(x)\|_1$ error guarantee as Theorem 3.1 but with two-sided error.

3.2 Count-Sketch: the ℓ_2 Guarantee

The gray lines in Algorithm 2 describe the modifications required to produce the CountSketch algorithm, which is like CountMedianSketch but with random signs introduced. This changes the error for a single r from (2) to

$$\widehat{x}_u^{(r)} - x_u = \sum_{\substack{h_r(v) = h_r(u) \\ v \neq u}} x_v s_r(v) s_r(u).$$

For fixed h_r, this is now a random variable in s_r, and because s_r is pairwise independent and mean zero, all the cross terms in $\mathbb{E}_{s_r}[(\widehat{x}_u^{(r)} - x_u)^2]$ disappear. In particular, (4) becomes

$$\mathop{\mathbb{E}}_{h_r, s_r}[E_L^2] = \sum_{\substack{v \in [n] \setminus H \\ v \neq u}} x_v^2 \cdot \Pr[h(v) = h(u)] \leq \|x - H_k(x)\|_2^2 / B.$$

If $B \geq 16k$, applying Markov's inequality and a union bound with the k/B chance that $E_H > 0$ shows that in each repetition

$$\Pr\left[(\widehat{x}_u^{(r)} - x_u)^2 > \|x - H_k(x)\|_2^2/k\right] < 1/8.$$

The chance this happens in at least $R/2$ of the R repetitions is then at most

$$\binom{R}{R/2} \cdot (1/8)^{R/2} < 2^R/8^{R/2} = 1/2^{R/2} \leq 1/n^2$$

for $R \geq 4\log_2 n$. If that failure event doesn't happen, the median is a good estimate, giving the following theorem:

Theorem 3.2. CountSketch*, when run with* $B \geq 16k$ *and* $R \geq 4\log_2 n$*, returns an* $\widehat{x}$ *that satisfies*

$$(\widehat{x}_u - x_u)^2 \leq \frac{1}{k}\|x - H_k(x)\|_2^2$$

for all u *with* $1 - 1/n$ *probability.*

At first glance, the bounds on $|\widehat{x}_u - x_u|$ given for CountMinSketch (Theorem 3.1) and CountSketch (Theorem 3.2) may seem incomparable—while $\|x - H_k(x)\|_2 \leq \|x - H_k(x)\|_1$, the denominator is only $\sqrt{k}$ for CountSketch rather than k for CountMinSketch. However, as shown in Exercise 1, this is misleading: up to constant factors, the ℓ_2 bound of Theorem 3.2 is stronger than the ℓ_1 bound of Theorem 3.1 for every vector x. For many natural vectors x, this difference is quite significant; we now examine it in detail.

3.3 Discussion of Recovery Guarantees

To better understand how much better ℓ_2 recovery guarantees are than ℓ_1 ones, we consider power-law (or "Zipfian") distributions where the ith largest element has frequency proportional to $i^{-\alpha}$. We also suppose the stream is distributed over a number of elements $n \gg k$ (which is finite so the sum of frequencies is still finite for $\alpha < 1.0$). For sharply decaying distributions of $\alpha > 1.0$, the ℓ_1 guarantee is

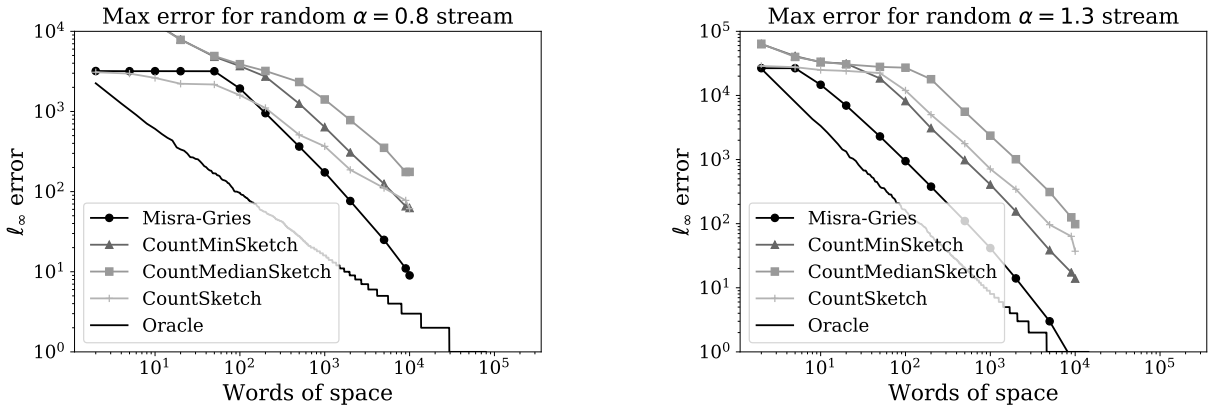
$$\|\widehat{x} - x\|_\infty \leq \frac{1}{k}\sum_{i=k+1}^{n} x_i \approx \frac{1}{k}x_1 \cdot \sum_{i=k+1}^{n} i^{-\alpha} \approx \frac{1}{\alpha - 1}x_k$$

while the ℓ_2 guarantee for $\alpha > 0.5$ is

$$\|\widehat{x} - x\|_\infty \leq \sqrt{\frac{1}{k}\sum_{i=k+1}^{n} x_i^2} \approx \sqrt{\frac{1}{k}x_1^2 \cdot \sum_{i=k+1}^{n} i^{-2\alpha}} \approx \frac{1}{\sqrt{2\alpha - 1}}x_k.$$

When $\alpha > 1.0$, these two guarantees are identical up to constant factors. But for intermediate decay of $0.5 < \alpha < 1.0$, the ℓ_1 guarantee is much worse:

$$\|\widehat{x} - x\|_\infty \leq \frac{1}{k}\sum_{i=k+1}^{n} x_i \approx \frac{1}{k}x_1 \cdot \sum_{i=k+1}^{n} i^{-\alpha} \approx \frac{1}{k}x_1\frac{1}{1-\alpha}n^{1-\alpha}$$



(a) $\alpha = 0.8$ (b) $\alpha = 1.3$

Figure 2: Comparison of error as a function of space for sparse recovery algorithms on random power law distribution streams, where the ith most common element has frequency proportional to $i^{-\alpha}$, with 10^5 items drawn over a 10^4 size domain. FREQUENTELEMENTS is assumed to use two words per entry of its table (one for the key, one for the value). ORACLE stores exactly the largest entries of the stream (with 2 words per entry). For $\alpha < 1$, the ℓ_2 bound of COUNTSKETCH gives significant benefit; for $\alpha > 1$, it performs worse than COUNTMINSKETCH due to constant factor inefficiency. In both cases, FREQUENTELEMENTS uses roughly an order of magnitude less space than COUNTMINSKETCH due to avoiding the $O(\log n)$ factor.

That is, unless $k > n^{1-\alpha}$, the ℓ_1 guarantee gives *no* nontrivial estimates (indeed, the all-zeros vector would satisfy it). Even above that threshold, the ℓ_1 guarantee remains a $(n/k)^{1-\alpha}$ factor worse than the ℓ_2 guarantee. For slow decay of $\alpha < 0.5$, the ℓ_2 guarantee becomes

$$\|\widehat{x} - x\|_\infty \leq \sqrt{\frac{1}{k}\sum_{i=k+1}^{n} x_i^2} \approx \sqrt{\frac{1}{k}x_1^2 \cdot \sum_{i=k+1}^{n} i^{-2\alpha}} \approx x_1 \frac{1}{\sqrt{1-2\alpha}}\sqrt{\frac{n}{k}}n^{-\alpha}$$

which is trivial until $k > n^{1-2\alpha}$, and remains a $\sqrt{n/k}$ factor better than the ℓ_1 bound for larger k.

The intermediate regime of $\alpha \in (0.5, 1.0)$ is the most relevant one in practice, as observed in the examples illustrated in Figure 1 as well as more generally (see, for example, Clauset et al. (2009)). Therefore the ℓ_2 guarantee is significantly more desirable than the ℓ_1 one.

In Figure 2 we illustrate these calculations with the empirical performance of the algorithms we have seen so far on such power-law distributions. The results closely match what one would expect from the theoretical bounds we have shown. For $\alpha = 0.8$, but not $\alpha = 1.3$, COUNTSKETCH's ℓ_2 bound is more important than COUNTMINSKETCH's constant factors, and in certain parameter regimes even enough to beat the $\Theta(\log n)$ factor savings in FREQUENTELEMENTS.

4 Uniform Algorithms

The sparse recovery algorithms described in the preceding sections originated in the computer science community in the context of streaming algorithms. Another body of work designed to solve very similar problems comes from the statistics and signal processing communities, where it is known as *compressed sensing* or *compressive sampling* (Donoho *et al.*, 2006; Candes et al., 2006). The motivation for compressed sensing is situations where one has a physical process that can cheaply observe linear measurements of a signal of interest—for example, MRI machines inherently sample Fourier measurements of the desired image; the single-pixel camera architecture takes pictures by applying brief masks during exposure; genetic testing companies can mix blood samples before testing; and radio telescopes sample from the Fourier spectrum based on their geometry. Without any assumption on the signal structure, learning an arbitrary $x \in \mathbb{R}^n$ would require n linear measurements, but a structural assumption such as sparsity can allow for fewer measurements—ideally leading to faster MRIs, higher-resolution photos, and cheaper genetic testing.

The high-level goal in compressed sensing is thus essentially identical to that of turnstile streaming sparse recovery: estimating approximately k-sparse vectors x from a small number of linear measurements $y = Ax$. (We say that x is k-sparse if it has at most k nonzero coordinates, and approximately k-sparse if it is "close" to a k-sparse vector.) But the emphasis is somewhat different, leading to different solutions.

Most notably, compressed sensing algorithms are designed to work even if the observation matrix A is not fully under the control of the algorithm designer. The observation matrix may have to satisfy a number of complicated constraints coming from how the physical sensing apparatus works; but as long as A is "good enough" in a formal sense, the recovery algorithms will work. Moreover, this allows for a degree of modularity: one can mix and match different algorithms and matrices, since essentially any "good enough" matrix construction will work with any algorithm. This modularity is in sharp contrast to most methods from the streaming community: it makes no sense to try and use (say) the CountSketch measurement matrix with the faster (Larsen et al., 2016) recovery algorithm, because the algorithms are intimately tied to their matrices.

4.1 The Restricted Isometry Property

The simplest approach to determining if A is "good enough" is that of *incoherence*:

Definition 4.1. Let $A \in \mathbb{R}^{m \times n}$ have columns $a_1, \ldots, a_n$ of ℓ_2 norm 1. The *coherence* μ of A is

$$\mu := \max_{i \neq j} |\langle a_i, a_j \rangle|.$$

If $\mu = 0$, then A has orthonormal columns so it is invertible and recovery is certainly possible. But our goal is to have $m \ll n$, so A cannot have orthonormal columns. The interesting thing is that even if μ is somewhat larger—up to $\Theta(1/k)$—then sparse recovery is possible with a variety of algorithms. Unfortunately, every matrix with $m < n/2$ has coherence $\mu > \sqrt{\frac{1}{2m}}$, so achieving "good enough" incoherence would require $\Omega(k^2)$ linear measurements. For the polynomially-large values of k typically considered, this is rather

more than the $O(k \log n)$ measurements we saw with streaming algorithms, suggesting the need for a different definition of "good enough". One popular definition is the Restricted Isometry Property:

Definition 4.2. For any k, the restricted isometry constant $\delta_k = \delta_k(A)$ of a matrix $A \in \mathbb{R}^{m \times n}$ is the smallest $\delta \geq 0$ such that

$$(1-\delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1+\delta)\|x\|_2^2 \qquad \text{for all } k\text{-sparse } x.$$

An equivalent formulation is that

$$\|(A^\top A - I)_{S \times S}\| \leq \delta \qquad \text{for all } S \subset [n],\ |S| \leq k \tag{6}$$

where $\|\cdot\|$ denotes the spectral norm.

We (informally) say that A satisfies the Restricted Isometry Property (RIP) if $\delta_{Ck} < c$ for some sufficiently large constant $C \geq 1$ and sufficiently small $c < 1$. The algorithmic results that follow show that the RIP (with sufficiently good constants C, c) implies that approximate k-sparse recovery is possible. One can show that $\delta_k \leq (k-1)\mu$, so this subsumes the incoherence-based results that require $\mu < \Theta(1/k)$, but the RIP bound is possible with only $m = O(k \log(n/k))$.

The Gaussian ensemble. A simple way to construct an RIP matrix with good parameters is by taking i.i.d. Gaussian entries of the appropriate variance.

Theorem 4.3. *Let $0 < \varepsilon < 1$ and $k > 1$ be parameters. If $A \in \mathbb{R}^{m \times n}$ has i.i.d. Gaussian entries of variance $1/m$, and $m > C\frac{1}{\varepsilon^2} k \log \frac{n}{k}$ for a sufficiently large constant C, then A has RIP constant $\delta_k < \varepsilon$ with $1 - e^{-\Omega(\varepsilon^2 m)}$ probability.*

The proof is based on applying a union bound to a net. We start with a lemma that shows how to bound an operator norm—which is a supremum over a continuous set—by the maximum over a finite set:

Lemma 4.4. *There exists a set $T \subset \mathbb{R}^n$ of 3^n unit vectors such that, for any symmetric matrix $M \in \mathbb{R}^{n \times n}$,*

$$\|M\| \leq 4 \max_{x \in T} x^\top M x.$$

Since $\|M\| = \sup_{\|x\|_2 = 1} x^\top M x$, this lemma loses at most a factor of 4. The proof is given as Exercise 5.

The other key lemma we need is the distributional Johnson-Lindenstrauss Lemma, which shows for any specific x that $\|Ax\|_2 \approx \|x\|_2$ with high probability:

Lemma 4.5 (Johnson-Lindenstrauss)**.** *For any $x \in \mathbb{R}^n$ and $\varepsilon \in (0,1)$, if $A \in \mathbb{R}^{m \times n}$ has i.i.d. Gaussian entries of variance $1/m$, then*

$$\Pr[|\|Ax\|_2^2 - \|x\|_2^2| > \varepsilon \|x\|_2^2] < 2e^{-\Omega(\varepsilon^2 m)}.$$

Proof of Theorem 4.3. Let $T \subset \mathbb{R}^k$ be the set of size 3^k given by Lemma 4.4 such that, for every set $S \subseteq [n]$ of size k,

$$\|(A^\top A - I)_{S\times S}\| \leq 4 \max_{x\in T} x^\top (A^\top A - I)_{S\times S} x.$$

By Lemma 4.5 applied with $\varepsilon' = \varepsilon/4$ and $n' = k$, we have for each S and $x \in T$ that

$$x^\top (A^\top A - I)_{S\times S} x \leq \frac{\varepsilon}{4}\|x\|_2^2 \leq \frac{\varepsilon}{4}$$

with probability at least $1 - 2e^{-\Omega(\varepsilon^2 m)}$. Taking a union bound over all S and $x \in T$, we have that

$$\delta_k \leq 4 \max_S \max_{x\in T} x^\top (A^\top A - I)_{S\times S} x$$

is bounded by ε with probability at least $1 - 2\binom{n}{k}3^k e^{-\Omega(\varepsilon^2 m)}$. If $m \geq O(\frac{1}{\varepsilon^2} k \log \frac{n}{k})$, this is $1 - e^{-\Omega(\varepsilon^2 m)}$. □

Gaussian matrices are just one way of constructing RIP matrices. Another example, with an essentially identical proof to the above, is a matrix with i.i.d. $\{\pm 1\}$ entries. We discuss more involved examples in Section 6 and the bibliographic notes.

4.2 Post-measurement vs Pre-measurement Noise

In streaming algorithms, it makes sense to suppose that $y = Ax$ is stored exactly: we see all of x eventually, and have complete control of the observations. But for the motivating applications for compressed sensing, where y represents a physical observation of some signal, one expects noise in the observation. Therefore we will aim for algorithmic guarantees in the presence of *post-measurement* noise: if

$$y = Ax^* + e$$

for an *exactly* k-sparse x^* and arbitrary noise vector e, the recovered $\widehat{x}$ will satisfy

$$\|\widehat{x} - x^*\|_2 \leq C\|e\|_2 \tag{7}$$

for some constant C.

Of course, signals such as images are unlikely to be exactly sparse, so a more realistic setting would have both post-measurement noise e and pre-measurement noise $x - H_k(x)$. For RIP matrices, however, such a result is actually implied by the post-measurement guarantee (7) by treating the pre-measurement noise $x - H_k(x)$ as post-measurement noise $A(x - H_k(x))$; see Exercise 2.

4.3 Iterative Methods

We now turn to algorithms that perform sparse recovery with RIP matrices. This can be done with either iterative methods or convex programming. The iterative methods are generally simpler and faster, but often require more measurements (by a constant factor). We will present a simple recovery algorithm, known as IterativeHardThresholding.

For intuition, suppose that there is no noise, so $y = Ax^*$. Recall that, since A satisfies the RIP, $A^\top A$ approximates the identity on any $O(k) \times O(k)$ submatrix. Therefore

$$A^\top y = A^\top A x^* \approx x^*,$$

where the approximation is good over $O(k)$-sized subsets. In particular, we will show

$$\|H_k(A^\top y) - x^*\|_2 \le O(\delta_{2k})\|x^*\|_2 \ll \|x^*\|_2.$$

This means that $x^{(1)} = H_k(A^\top y)$ is a good first step in recovering x^*: it is most of the way there. (The operation H_k, which thresholds to the largest k entries, is known as "hard" thresholding because of the discontinuity in treatment between elements just above and just below the threshold.) But $x^{(1)}$ still has some residual error $x^* - x^{(1)}$. To reduce this, we can compute $y - Ax^{(1)} = A(x^* - x^{(1)})$, which is effectively a measurement of this residual. We then repeat the procedure of multiplying by $A^\top$ and thresholding, getting a new estimate of x^*:

$$x^{(2)} = H_k(x^{(1)} + A^\top(y - Ax^{(1)})).$$

Algorithm 3 Iterative Hard Thresholding (IHT)

function IterativeHardThresholding(y, A, k)
  $x^{(0)} \leftarrow 0$
  for $r \leftarrow 0, 1, 2, \ldots, R-1$ **do**
    $x^{(r+1)} \leftarrow H_k\left(x^{(r)} + A^\top(y - Ax^{(r)})\right)$
  end for
  return $x^{(R)}$
end function

In Lemma 4.7 we show that this IterativeHardThresholding procedure works even with noise: if $y = Ax^* + e$ for an exactly k-sparse vector x^*, the residual error geometrically converges to the noise level $O(\|e\|_2)$. To establish this, we first show that the thresholding step does not increase the ℓ_2 distance by more than a constant factor:

Lemma 4.6. *Let $x, z \in \mathbb{R}^n$ so that x is k-sparse with support S and $T \subseteq [n]$ consists of the largest k terms of z. Then*

$$\|x - z_T\|_2^2 \le 3\|(x - z)_{S \cup T}\|_2^2.$$

Proof. For every $i \in S \setminus T$ we can assign a unique $j \in T \setminus S$ such that $|z_j| \ge |z_i|$. Therefore

$$x_i^2 \le (|x_i - z_i| + |z_i|)^2 \le (|x_i - z_i| + |z_j|)^2 \le 2(x_i - z_i)^2 + 2z_j^2.$$

Adding in the terms for $i \in T$ gives the result. □

Lemma 4.7. *In each iteration of* IterativeHardThresholding,

$$\|x^{(r+1)} - x^*\|_2 \le \sqrt{3}\delta_{3k}\|x^{(r)} - x^*\|_2 + \sqrt{6}\|e\|_2.$$

Proof. Define

$$x' := x^{(r)} + A^\top(y - Ax^{(r)}) = x^* + (A^\top A - I)(x^* - x^{(r)}) + A^\top e.$$

Let $S = \operatorname{supp}(x^{(r+1)}) \cup \operatorname{supp}(x^{(r)}) \cup \operatorname{supp}(x^*)$, so $|S| \le 3k$. Note that the RIP implies that

$$\|A_S^\top\|^2 = \|(A^\top A)_{S\times S}\| \le 1 + \delta_{3k}.$$

Therefore we have

$$\begin{aligned}\|(x' - x^*)_S\|_2 &\le \|((A^\top A - I)(x^* - x^{(r)}))_S\|_2 + \|(A^\top e)_S\|_2 \\ &\le \|(A^\top A - I)_{S\times S}\|\|x^* - x^{(r)}\|_2 + \|A_S^\top\| \cdot \|e\|_2 \\ &\le \delta_{3k}\|x^* - x^{(r)}\|_2 + \sqrt{1+\delta_{3k}} \cdot \|e\|_2.\end{aligned}$$

Finally, since $\delta_{3k} < 1$ and $x^{(r+1)} = H_k(x')$ we have by Lemma 4.6 that

$$\|x^{(r+1)} - x^*\|_2 \le \sqrt{3}\|(x^* - x')_S\|_2 \le \sqrt{3}\delta_{3k}\|x^{(r)} - x^*\|_2 + \sqrt{6}\|e\|_2.$$

□

If $\delta_{3k} < 1/\sqrt{3}$, this iteration will eventually converge to $O(\|e\|_2)$. If $\delta_{3k} < \frac{1}{4\sqrt{3}} \approx 0.144$, we will have

$$\|x^{(r+1)} - x^*\|_2 \le \max\left(\frac{1}{2}\|x^{(r)} - x^*\|_2, \sqrt{24}\|e\|_2\right)$$

and hence the residual error $\|x^{(r+1)} - x^*\|_2$ will converge geometrically to at most $\sqrt{24}\|e\|_2$:

Theorem 4.8. *If $\delta_{3k} < 0.14$, the output $x^{(R)}$ of* IterativeHardThresholding *will have*

$$\|x^{(R)} - x^*\|_2 \le \sqrt{24}\|e\|_2$$

after $R = \log_2 \frac{\|x^*\|_2}{\|e\|_2}$ iterations.

Uniformity vs nonuniformity. The above argument relies on the fact that RIP works uniformly for *all* sparse vectors, even ones that depend on the matrix A (as the residuals $x^* - x^{(r)}$ do). As a result, the theorem also applies to $y = Ax^* + e$ for every x^* and e. This stands in contrast to the non-uniform randomized guarantee of CountMinSketch: for each matrix A, there are many vectors x that will cause CountMinSketch to violate its ℓ_1 guarantee. For RIP-based algorithms, while the matrix A is typically randomized and as such might fail to satisfy the RIP, as long as A satisfies the RIP the recovery guarantee will hold on every input.

Uniformity is very convenient in proofs because it allows us to ignore any possible dependencies between the error and the measurement matrix. However, some properties cannot be achieved uniformly: the ℓ_2 bound achieved by CountSketch is one (Cohen et al., 2009).

4.4 L1 Minimization

Another method for performing compressed sensing from an RIP matrix is L1 minimization, also known as basis pursuit or, in its Lagrangian form, the LASSO. The intuition is that, since the true x^* is k-sparse, one would like to find the *sparsest* vector $\widehat{x}$ that approximately matches the measurements; here we say $\widehat{x}$ "matches" the measurements if $\|y - A\widehat{x}\|_2 \leq R$ for some external estimate R on the noise $\|e\|_2$. However finding the sparsest $\widehat{x}$ is a hard non-convex optimization problem, so we settle for minimizing its convex relaxation $\|\widehat{x}\|_1$. Remarkably, and in contrast to minimizing $\|\widehat{x}\|_p$ for $p > 1$, this tends to yield sparse solutions.

Algorithm 4 L1 minimization

```
function L1Minimization(y, A, R)
    x̂ ← arg min_{‖y − Ax′‖₂ ≤ R} ‖x′‖₁
    return x̂
end function
```

Theorem 4.9. *There exists a constant $C > 0$ such that the following holds. Let $A \in \mathbb{R}^{m \times n}$ have RIP constant $\delta_{2k} < 0.62$. Then for any k-sparse $x \in \mathbb{R}^n$ and any $e \in \mathbb{R}^m$, and any $R \geq \|e\|_2$, the L1 minimization result $\widehat{x} = \text{L1Minimization}(Ax + e, A, R)$ satisfies*

$$\|\widehat{x} - x^*\|_2 \leq CR.$$

See Candes et al. (2006), or the presentation in Foucart and Rauhut (2013), for a proof. Up to constant factors, this is essentially the same result as Iterative Hard Thresholding.

5 Lower Bound

A linear sparse recovery algorithm consists of a distribution on random matrices $A \in \mathbb{R}^{m \times n}$ and an algorithm for recovering $\widehat{x}$ from A and $y = Ax$. In the preceding sections we have given various such algorithms that achieve various guarantees, the weakest of which is the ℓ_1/ℓ_1 guarantee:

$$\|\widehat{x} - x\|_1 \leq O(1) \cdot \|x - H_k(x)\|_1.$$

Both CountMinSketch and CountSketch achieved this with $O(k \log n)$ linear measurements, and IterativeHardThresholding and L1Minimization achieve this with $O(k \log \frac{n}{k})$ Gaussian linear measurements; for $k < n^{0.99}$, the two bounds are equivalent. We now show that this many measurements are necessary for any linear sketching algorithm.

Theorem 5.1 (Do Ba et al. (2010))**.** *Any ℓ_1/ℓ_1 linear sparse recovery algorithm with constant approximation factor and constant success probability requires $\Omega(k \log \frac{n}{k})$ linear measurements.*

Proof sketch. The proof is based on communication complexity. Roughly speaking, we will produce a distribution on x that contains a lot of information, then show how to extract that information from Ax using the ℓ_1/ℓ_1 sparse recovery algorithm. This implies Ax also contains a lot of information, so m must be fairly large.

We pick a large "codebook" $T \subseteq \{0,1\}^n$ of k-sparse binary vectors of minimum Hamming distance $k/2$. One can construct such a T of size $2^{\Omega(k \log \frac{n}{k})}$ using a greedy construction (see Exercise 6).

Now, suppose we have an algorithm that can perform ℓ_1/ℓ_1 sparse recovery with approximation factor C. Set $R = \Theta(\log n)$, and for any $x_1, x_2, \ldots, x_R \in T$ take

$$x = x_1 + \varepsilon x_2 + \varepsilon^2 x_3 + \ldots + \varepsilon^R x_R$$

for $\varepsilon = \frac{1}{4C+6}$ a small constant. The idea of the proof is the following: given $y = Ax$, we can recover $\widehat{x}$ such that

$$\|\widehat{x} - x_1\|_1 \leq \|x - x_1\|_1 + \|\widehat{x} - x\|_1 \leq (C+1)\|x - x_1\|_1 \leq (C+1)k\frac{\varepsilon}{1-\varepsilon} < k/4$$

and so, because T has minimum distance $k/2$, we can exactly recover x_1 by rounding $\widehat{x}$ to the nearest element of T. But then we can repeat the process on $\frac{1}{\varepsilon}(Ax - Ax_1)$ to find x_2, then x_3, up to x_R, for $R \lg |T| = \Omega(Rk \log(n/k))$ bits total. Thus Ax must contain this many bits; but if the entries of A are rational numbers with $\text{poly}(n)$ bounded numerators and denominators, then each entry of Ax can be described in $O(R + \log n)$ bits, so

$$m \cdot O(R + \log n) \geq \Omega(Rk \log(n/k))$$

or $m \geq \Omega(k \log(n/k))$.

There are two issues that make the above outline not totally satisfactory, which we only briefly address how to resolve here. First, the theorem statement makes no supposition on the entries of A being polynomially bounded. To resolve this, we perturb x with a tiny (polynomially small) amount of additive Gaussian noise, after which discretizing Ax at an even tinier (but still polynomial) precision has negligible effect on the failure probability. The second issue is that the above outline requires the algorithm to recover all R vectors, so it only applies if the algorithm succeeds with $1 - 1/\log n$ probability rather than constant probability. This is resolved by using a reduction from the communication complexity of the *augmented indexing* problem. □

6 Different Measurement Models

6.1 A Hybrid Result: the RIP-1 and Sparse Matrices

Sparse matrices are much more convenient to store and manipulate than dense ones. Unfortunately, sparse matrices cannot satisfy the standard RIP (see Exercise 3). However, they can satisfy an ℓ_1 version of it:

Definition 6.1. For any k, the RIP-1 constant $\delta_k^{(1)}$ of a matrix $A \in \mathbb{R}^{m \times n}$ is the smallest $\delta \geq 0$ such that

$$(1-\delta)\|x\|_1 \leq \frac{1}{d}\|Ax\|_1 \leq \|x\|_1 \qquad \text{for all } k\text{-sparse } x$$

for some scale factor d.

We (informally) say that A satisfies the RIP-1 if $\delta^{(1)}_{Ck} < c$ for some sufficiently good constants $C \geq 1$, $c < 1$. The definition of the RIP-1 differs from the standard RIP in that it uses the ℓ_1 norm and that it includes a scale factor d. The scale factor is convenient, because the prototypical RIP-1 matrix is the adjacency matrix of an *unbalanced bipartite expander graph*:

Definition 6.2. A (k, ε) unbalanced bipartite expander is a bipartite graph $G = (A, B, E)$ with left degree d such that, for any set $S \subseteq A$ of vertices on the left with size $|S| \leq k$, the neighborhood $N(S) \subseteq B$ has size $|N(S)| \geq (1 - \varepsilon)d|S|$.

A random bipartite graph of left degree $d = \Theta(\log n)$, n right vertices, and $m = \Theta(\frac{1}{\varepsilon^2} k \log n)$ left vertices is an expander with high probability. There also exist explicit constructions, albeit with slightly worse parameters. Bipartite expansion is closely connected to the RIP-1:

Lemma 6.3 (Berinde et al. (2008a))**.** *A binary matrix $A \in \{0, 1\}^{m \times n}$ with d ones per column has RIP-1 constant $\delta^{(1)}_k < \varepsilon$ if and only if it is the adjacency matrix of a $(k, \Theta(\varepsilon))$-bipartite expander.*

Just like with the standard RIP, sparse recovery from RIP-1 matrices is possible through either linear programming or iterative methods. One such iterative method is SparseMatchingPursuit (Berinde et al., 2008b), shown in Algorithm 5.

Algorithm 5 Sparse Matching Pursuit (SMP)

function SparseMatchingPursuit(y, A, k)
  $x^{(0)} \leftarrow 0$
  for $r \leftarrow 0, 1, 2, \ldots, R - 1$ **do**
    $u_i \leftarrow \text{median}_{A_{ji}=1}(y - Ax^{(r)})_j \quad \forall i \in [n]$
    $x^{(r+1)} \leftarrow H_k\left(x^{(r)} + H_{2k}(u)\right)$
  end for
  return $x^{(R)}$
end function

Theorem 6.4. *Let $A \in \mathbb{R}^{m \times n}$ be a binary matrix with RIP-1 constant $\delta^{(1)}_{Ck} < c$ for sufficiently large constant C and small constant c. Then for any $x \in \mathbb{R}^n$, the result $\widehat{x}$ of either SMP or L1 minimization has*

$$\|\widehat{x} - x\| \leq O(1) \cdot \|x - H_k(x)\|_1.$$

The SparseMatchingPursuit algorithm is very similar to IterativeHardThresholding. In fact, if the H_{2k} threshold were removed and the median replaced by a mean, the algorithm would be identical to IterativeHardThresholding on $A/\sqrt{d}$ for d-regular graphs A. It seems plausible that IterativeHardThresholding also works in this setting, but we are not aware of such a result.

Alternatively, one can view SparseMatchingPursuit as an iterative version of CountMedianSketch. If the random hash functions used in CountMedianSketch were fully

independent, not just pairwise independent, then the associated matrix A would be a near-optimal RIP-1 matrix with high probability. Furthermore, the first iterate $x^{(1)}$ of SparseMatchingPursuit is identical to the (thresholded to top k) result of CountMedianSketch, which achieves the ℓ_1/ℓ_1 result with high probability for each x. By iteratively refining the estimates, SparseMatchingPursuit can achieve the ℓ_1/ℓ_1 result uniformly for all x.

Relative to algorithms previously considered in this chapter, the RIP-1 algorithm combines the uniform guarantees of RIP-based algorithms with the sparse matrices and fast algorithms of CountMin and CountSketch. The downside is that the Theorem 6.4 recovery guarantee is weaker than all the others: it depends on the ℓ_1 not the ℓ_2 norm of the tail, and only bounds the ℓ_1 not the ℓ_2 or ℓ_∞ error of the result.

6.2 Fourier Measurements

An important subclass of linear measurements is that of *Fourier measurements*, where A consists of rows of a Fourier matrix. In this section we will focus on the unidimensional discrete Fourier matrix $F \in \mathbb{C}^{n\times n}$ given by

$$F_{ij} = \frac{1}{\sqrt{n}} e^{2\pi \mathbf{i} ij/n},$$

although similar results exist for other Fourier-related matrices such as Hadamard or multi-dimensional discrete Fourier matrices. In this context, we consider the measurement matrix $A = F_\Omega$ that consists of a subset $\Omega \subset [n]$ of rows of the discrete Fourier matrix. The goal is to find conditions on Ω and algorithms under which sparse recovery is possible and efficient.

This problem is of interest to both the streaming and compressed sensing communities, but as with previous sections of this chapter there are differences in emphasis.

Compressed sensing. The main compressed sensing motivation for Fourier measurements is that physical processes such as MRIs, radio astronomy, and wireless communication naturally yield Fourier measurements of the signal. The secondary motivation is that subsampled Fourier matrices make compressed sensing algorithms more efficient: they can be stored in $O(m)$ words rather than the $O(mn)$ required by i.i.d. Gaussian matrices, and the running time for recovery algorithms—being dominated by the cost of multiplying a vector by A or $A^\top$—becomes $\widetilde{O}(n)$ rather than $\widetilde{O}(mn)$ by using the Fast Fourier Transform (FFT).

Fortunately, subsampled Fourier matrices satisfy the RIP with relatively few rows:

Theorem 6.5 (Haviv and Regev (2017))**.** *Let $0 < \varepsilon < 1$ and $k > 1$ be parameters. Let $\Omega \subset [n]$ by a random subset of size m. If $m > C\frac{1}{\varepsilon^2} k \log n \log^2 k$ for a sufficiently large constant C, then $\sqrt{\frac{n}{m}} F_\Omega$ satisfies $\delta_k < \varepsilon$ with high probability.*

Therefore, with an extra $O(\log^2 k)$ factor in measurements, standard recovery algorithms such as IterativeHardThresholding and L1Minimization give sparse recovery from Fourier measurements. We do not know if the extra $\log^2 k$ factor relative to Gaussian matrices' $O(k \log(n/k))$ is necessary. For the case of *Hadamard* matrices, the same theorem applies but we do know at least one extra $\log k$ is necessary (Błasiok et al., 2019).

Sublinear algorithms. The streaming and sublinear algorithms community became interested in sparse recovery with Fourier measurements for a different reason: it gives the prospect of a faster Fourier transform than the FFT, one which can approximate the Fourier transform of a signal in *sublinear* time.

Theorem 6.6 (Hassanieh et al. (2012))**.** *There exists an algorithm to compute $\widehat{x}$ using $O(k \log(n/k) \log(n/\delta))$ time and queries to Fx such that*

$$\|\widehat{x} - x\|_2 \leq 2\|x - H_k(x)\|_2 + \delta\|x\|_2$$

with 9/10 probability.

One can also optimize the number of queries at the expense of time, down to $O(k \log(n/\delta))$ queries with $O(k \log^{O(1)} n)$ time (Kapralov, 2017).

The basic approach for these results is to try to simulate streaming algorithms like CountSketch using Fourier measurements. We pick a "filter" $g \in \mathbb{C}^n$ that is sparse in both Fourier ("frequency") domain and regular ("time") domain: Fg is $B = O(k)$-sparse, while g is approximately n/B-sparse. We can use our queries to Fx to compute the sparse result of pointwise multiplication $Fx \cdot Fg$. By the Fourier convolution theorem,

$$F^{-1}(Fx \cdot Fg) = x * g.$$

We can use a B-dimensional inverse FFT on $Fx \cdot Fg$ to quickly compute $(x*g)_j$ at B different positions j. If g is chosen carefully, the result can be shown to behave similarly to the linear observations (1) in CountSketch: we can approximately "hash" the coordinates down to B cells, and observe the sum within each cell.

7 Matrix Recovery

A natural extension of sparse recovery is that of *low-rank matrix recovery*. Rather than estimating a k-sparse vector $x \in \mathbb{R}^n$, we consider estimating a rank-k matrix X. For this brief overview, we only consider positive semidefinite matrices $X \in \mathbb{R}^{n \times n}$. Let the eigenspectrum of X be $\lambda = \lambda(X) \in \mathbb{R}^n$, sorted in decreasing order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Then X having rank k is equivalent to λ being k-sparse. Low-rank matrix recovery shares much motivation with sparse recovery, since the matrix spectra often do empirically decay. Moreover, the techniques used in sparse recovery often extend to the matrix case. Such techniques include:

Insertion-only. Suppose that the matrix X is received as a series of rank one updates, $X = \sum u_i u_i^\top$ for a stream of vectors $u_i \in \mathbb{R}^n$. This setting is much like insertion-only streaming algorithms, and a simple extension of FrequentElements due to Liberty (2013), known as FrequentDirections, achieves a result analogous to Lemma 2.1. The idea is to keep track of a rank-k approximation $\widehat{X}$ to X (which can be stored in kn space). On any update $u_i u_i^\top$, first the update is added to $\widehat{X}$, then this updated matrix—which could have rank up to $k+1$—is "shrunk" back down to rank k by subtracting $s_i := \lambda_{k+1}(\widehat{X})$ from every

Algorithm 6 FREQUENTDIRECTIONS matrix heavy hitters algorithm

function FREQUENTDIRECTIONS(STREAM, k)
  $\widehat{X} \leftarrow 0 \in \mathbb{R}^{n \times n}$
  for u in STREAM **do**
    $\widehat{X} \mathrel{+}= uu^\top$
    if $\widehat{X}$ has rank $k+1$ **then**
      Compute the eigendecomposition $\widehat{X} = \sum_{i=1}^{k+1} \lambda_i v_i v_i^\top$
      Set $\widehat{X} \leftarrow \sum_{i=1}^{k} (\lambda_i - \lambda_{k+1}) v_i v_i^\top$
    end if
  end for
  return $\widehat{X}$
end function

eigenvalue. As shown in Exercise 4, one can prove bounds for this algorithm analogous to the FREQUENTELEMENTS bounds of Lemmas 2.1 and 2.2: both

$$X - \frac{1}{k+1}\|\lambda\|_1 \mathbf{I} \preceq \widehat{X} \preceq X \tag{8}$$

and a sparsity-aware bound

$$X - \frac{2}{k}\|\lambda - H_{k/2}(\lambda)\|_1 \mathbf{I} \preceq \widehat{X} \preceq X. \tag{9}$$

L1 minimization. The above algorithm relies on "insertion-only"–like updates to X. With more general updates, one would like an algorithm that can reconstruct an estimate of X from linear measurements $\mathcal{A} : \mathbb{R}^{n \times n} \to \mathbb{R}^m$.

The natural analog of L1 minimization is to minimize the *nuclear norm*, which for positive semidefinite matrices equals the trace:

$$\|\widehat{X}\|_* := \|\lambda(\widehat{X})\|_1 = \mathrm{Tr}(\widehat{X}) = \sum_{i=1}^{n} \lambda_i.$$

This nuclear norm minimization problem

$$\min_{\mathcal{A}(\widehat{X})=y} \|\widehat{X}\|_*$$

is a semidefinite program, and it turns out that this leads to an ℓ_1/ℓ_1 bound for recovery:

$$\|X - \widehat{X}\|_* \leq O(1)\|\lambda - H_k(\lambda)\|_1$$

if $\mathcal{A}$ is a "good" set of observations, as Gaussian linear measurements are w.h.p. once $m \geq O(kn)$.

Note that just as in the vector case, this ℓ_1/ℓ_1 bound from $L1$ minimization is weaker than the ℓ_∞/ℓ_1 achieved by FREQUENTELEMENTS/FREQUENTDIRECTIONS. Unlike the vector case, however, here L1 minimization does not lose an additional $\log(n/k)$ factor in the sample/space complexity.

Streaming algorithms. Nuclear norm minimization requires solving a semidefinite program, which is polynomial time but still not that efficient. It also uses a dense linear sketch $\mathcal{A}(X)$, which takes $m = O(kn)$ time to update whenever a single entry of X is updated.

One alternative is to store

$$Y = X\Omega \qquad \text{and} \qquad W = \Psi X$$

for random Gaussian matrices $\Omega \in \mathbb{R}^{n\times 2k+1}, \Psi \in \mathbb{R}^{4k+3\times n}$. These can be updated in $O(k)$ time under single-entry updates to X. Moreover, there is a relatively fast algorithm to compute a good approximation $\widehat{X}$ to X from Y and W: if Y has SVD $Q\Sigma R^\top$ for $Q \in \mathbb{R}^{n\times 2k+1}$,

$$\widehat{X} := Q(\Psi Q)^+ W$$

satisfies

$$\mathbb{E}[\|X - \widehat{X}\|_F^2] \le 4\|\lambda - H_k(\lambda)\|_2^2.$$

This is an ℓ_2/ℓ_2 bound on the eigenvalues of the approximation, which is stronger than the ℓ_1/ℓ_1 bound from L1 minimization (although the latter is a uniform bound).

8 Notes

For much more detail on compressed sensing, in both the vector and matrix case, we recommend the book by Foucart and Rauhut (2013). For a survey on sparse recovery from the perspective of streaming algorithms, see Gilbert and Indyk (2010). An empirical study of power-law distributions can be found in Clauset et al. (2009).

Algorithms similar to CountMinSketch or CountSketch but with sublinear recovery time can be found in Cormode and Hadjieleftheriou (2008), Gilbert et al. (2012), and Larsen et al. (2016).

Alternative RIP matrices. The sample complexity m required for subsampled Fourier matrices to satisfy the RIP has been the focus of a long line of improvements (Candes et al., 2006; Rudelson and Vershynin, 2008; Cheraghchi et al., 2012; Bourgain, 2014; Haviv and Regev, 2017). Partial circulant matrices are another construction of RIP matrices with similar benefits to subsampled Fourier matrices: they use $O(n)$ bits of randomness, they can be multiplied with a vector in $O(n \log n)$ time, and they satisfy the RIP with $O(k \log^c n)$ measurements (Krahmer et al., 2014). The best deterministic construction of RIP matrices uses $m = k^{2-\varepsilon}$ rows for a very small constant $\varepsilon > 0$ Bourgain et al. (2011). The lower bound on sparsity of RIP matrices given in Exercise 3 is due to Chandar (2010).

Matrix recovery. Nuclear norm minimization for low-rank matrix recovery was first shown for exactly low-rank matrices by Recht et al. (2010), and extended to the robust case by Candes and Plan (2011). The streaming algorithm we present is from Tropp et al. (2017), based on Upadhyay (2018) and Clarkson and Woodruff (2009).

References

Berinde, Radu, Gilbert, Anna C, Indyk, Piotr, Karloff, Howard, and Strauss, Martin J. 2008a. Combining geometry and combinatorics: A unified approach to sparse signal recovery. Pages 798–805 of: *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. IEEE.

Berinde, Radu, Indyk, Piotr, and Ruzic, Milan. 2008b. Practical near-optimal sparse recovery in the l1 norm. Pages 198–205 of: *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. IEEE.

Błasiok, Jarosław, Lopatto, Patrick, Luh, Kyle, and Marcinek, Jake. 2019. An Improved Lower Bound For Sparse Reconstruction From Subsampled Hadamard Matrices. *FOCS*.

Bourgain, Jean. 2014. An improved estimate in the restricted isometry problem. Pages 65–70 of: *Geometric aspects of functional analysis*. Springer.

Bourgain, Jean, Dilworth, Stephen, Ford, Kevin, Konyagin, Sergei, Kutzarova, Denka, et al. 2011. Explicit constructions of RIP matrices and related problems. *Duke Mathematical Journal*, **159**(1), 145–185.

Candes, Emmanuel J, and Plan, Yaniv. 2011. Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *IEEE Transactions on Information Theory*, **57**(4), 2342–2359.

Candes, Emmanuel J, Romberg, Justin K, and Tao, Terence. 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, **59**(8), 1207–1223.

Chandar, Venkat Bala. 2010. *Sparse graph codes for compression, sensing, and secrecy*. Ph.D. thesis, Massachusetts Institute of Technology.

Charikar, Moses, Chen, Kevin, and Farach-Colton, Martin. 2002. Finding frequent items in data streams. Pages 693–703 of: *International Colloquium on Automata, Languages, and Programming*. Springer.

Cheraghchi, Mahdi, Guruswami, Venkatesan, and Velingker, Ameya. 2012. Restricted isometry of Fourier matrices and list decodability of random linear codes. *"SODA"*.

Clarkson, Kenneth L, and Woodruff, David P. 2009. Numerical linear algebra in the streaming model. Pages 205–214 of: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. ACM.

Clauset, Aaron, Shalizi, Cosma Rohilla, and Newman, Mark EJ. 2009. Power-law distributions in empirical data. *SIAM review*, **51**(4), 661–703.

Cohen, A., Dahmen, W., and DeVore, R. 2009. Compressed sensing and best k-term approximation. *J. Amer. Math. Soc*, **22**(1), 211–231.

Cormode, Graham, and Hadjieleftheriou, Marios. 2008. Finding frequent items in data streams. *Proceedings of the VLDB Endowment*, **1**(2), 1530–1541.

Cormode, Graham, and Muthukrishnan, Shan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, **55**(1), 58–75.

Do Ba, Khanh, Indyk, Piotr, Price, Eric, and Woodruff, David P. 2010. Lower bounds for sparse recovery. Pages 1190–1197 of: *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM.

Donoho, David L, et al. 2006. Compressed sensing. *IEEE Transactions on information theory*, **52**(4), 1289–1306.

Foucart, Simon, and Rauhut, Holger. 2013. *A mathematical introduction to compressive sensing*. Springer.

Ganguly, Sumit. 2008. Lower bounds on frequency estimation of data streams. Pages 204–215 of: *International Computer Science Symposium in Russia*. Springer.

Gilbert, Anna, and Indyk, Piotr. 2010. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, **98**(6), 937–947.

Gilbert, Anna C, Li, Yi, Porat, Ely, and Strauss, Martin J. 2012. Approximate sparse recovery: optimizing time and measurements. *SIAM Journal on Computing*, **41**(2), 436–453.

Hassanieh, H., Indyk, P., Katabi, D., and Price, E. 2012. Nearly Optimal Sparse Fourier Transform. *STOC*.

Haviv, Ishay, and Regev, Oded. 2017. The restricted isometry property of subsampled Fourier matrices. Pages 163–179 of: *Geometric Aspects of Functional Analysis*. Springer.

Kapralov, Michael. 2017. Sample efficient estimation and recovery in sparse FFT via isolation on average. Pages 651–662 of: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. Ieee.

Krahmer, Felix, Mendelson, Shahar, and Rauhut, Holger. 2014. Suprema of chaos processes and the restricted isometry property. *Communications on Pure and Applied Mathematics*, **67**(11), 1877–1904.

Larsen, Kasper Green, Nelson, Jelani, Nguyên, Huy L, and Thorup, Mikkel. 2016. Heavy hitters via cluster-preserving clustering. Pages 61–70 of: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE.

Liberty, Edo. 2013. Simple and deterministic matrix sketching. Pages 581–588 of: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Misra, Jayadev, and Gries, David. 1982. Finding repeated elements. *Science of computer programming*, **2**(2), 143–152.

Recht, Benjamin, Fazel, Maryam, and Parrilo, Pablo A. 2010. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, **52**(3), 471–501.

Rudelson, M., and Vershynin, R. 2008. On sparse reconstruction from Fourier and Gaussian measurements. *CPAM*, **61**(8), 1025–1171.

Tropp, Joel A, Yurtsever, Alp, Udell, Madeleine, and Cevher, Volkan. 2017. Practical sketching algorithms for low-rank matrix approximation. *SIAM Journal on Matrix Analysis and Applications*, **38**(4), 1454–1485.

Upadhyay, Jalaj. 2018. The price of privacy for low-rank factorization. Pages 4176–4187 of: *Advances in Neural Information Processing Systems*.

9 Exercises

1. Comparison of CountSketch and CountMinSketch guarantees.

   (a) For any vector $x \in \mathbb{R}^n$, show that

   $$\|x - H_k(x)\|_2 \leq \frac{1}{\sqrt{k}}\|x\|_1.$$

   (b) Show that if $\widehat{x}$ is the result of CountSketch for $k' = 2k$, then

   $$\|\widehat{x} - x\|_\infty \leq \frac{1}{k}\|x - H_k(x)\|_1.$$

   Compare this bound to the Theorem 3.1 bound for CountMinSketch.

2. Pre-measurement noise and RIP-based methods.

   (a) Show that, if A has RIP constant δ_k,

   $$\|A(x - H_k(x))\|_2 \leq \frac{(1+\delta_k)}{\sqrt{k}}\|x\|_1$$

   for any vector $x \in \mathbb{R}^n$.

   (b) Show that the result $\widehat{x}$ of L1Minimization or IterativeHardThresholding from $y = Ax$ satisfies

   $$\|\widehat{x} - x\|_2 \leq \frac{O(1)}{\sqrt{k}}\|x - H_k(x)\|_1$$

   if A satisfies a sufficiently strong RIP.

(c) Use the Johnson-Lindenstrauss Lemma to show that, if A has i.i.d. Gaussian entries of variance $1/m$, the result $\widehat{x}$ of L1Minimization or IterativeHardThresholding from $y = Ax$ will satisfy

$$\|\widehat{x} - x\|_2 \leq O(1)\|x - H_k(x)\|_2$$

with $1 - e^{-\Omega(m)}$ probability. Note that this is a *nonuniform* bound. How does it compare to the bound in (b)?

3. In this problem we show that matrices that satisfy the RIP cannot be very sparse. Let $A \in \mathbb{R}^{m \times n}$ have $\delta_k < 1/2$ for $m < n$. Suppose that the average column sparsity of A is d, i.e., A has nd nonzero entries. Furthermore, suppose that $A \in \{0, \pm\alpha\}^{m \times n}$ for some parameter α.

   (a) By looking at the sparsest column, give a bound for α in terms of d.

   (b) By looking at the densest row, give a bound for α in terms of n, m, d and k.

   (c) Conclude that either $d \geq k/C$ or $m \geq n/C$ for a universal constant C.

   (d) [Optional] Extend the result to general settings of the non-zero $A_{i,j}$.

4. Consider the matrix FrequentElements-like algorithm FrequentDirections described in Algorithm 6.

   (a) Use the potential function $\mathrm{Tr}(\widehat{X})$ to show that $\sum_i s_i \leq \frac{1}{k+1}\|\lambda\|_1$, where s_i is the eigenvalue shrinkage after the ith update. Conclude that FrequentElements achieves (8).

   (b) Now let Π be the orthogonal projection matrix onto the span of all but the top $k/2$ eigenvectors of X. Using $\mathrm{Tr}(\Pi\widehat{X})$ as a potential function, prove that FrequentDirections also satisfies the bound (9).

5. Prove Lemma 4.4. Choose T to be a 1/2-cover of the unit ℓ_2 ball $\mathcal{B}$, meaning that $T \subset \mathcal{B}$ and, for every $x \in \mathcal{B}$, there exists an x' in T such that $\|x' - x\|_2 \leq \frac{1}{2}$. (This will give a set T satisfying the lemma of *at most* unit norm elements, but scaling them up to unit norm only makes the result more true.)

6. Construct the codebook T used in the proof of Theorem 5.1. First construct a code of Hamming distance $k/4$ over $[n/k]^k$, then embed this into $\{0,1\}^n$.