Experiments in Sparse Recovery with Random Matrices

Liam Thomas
Mathematics 123—Mathematical Aspects of Data Analysis

23 December 2021

Abstract

In this paper we implement a basic sparse recovery algorithm following an approach proposed in [1]. This approach uses a random, sparse, binary matrix A that is constructed to be an adjacency matrix to an expander graph. After testing this basic algorithm on synthetic data, we attempt some image compression with wavelets and outline possible next steps.

1 Relevant results in compressive sensing and sparse recovery

1.1 Outline of compressive sensing

Let $x \in \mathbb{R}^n$ and let $U = U_1, \dots, U_n$ be an orthonormal basis of $\mathbb{R}^n$. Then x can be written as $x = < x, U_1 > U_1 + \dots + < x, U_n > U_n$. We call x a *signal* for our compression scheme.

In general, compressive sensing is a two-step process. First, we select U such that x with respect to the basis U, in the representation given above, is sparse. Second, we select the first k coefficients of x with the largest magnitude.

A vector v is said to be *s-sparse* if $||v||_0 < s$. Given an s-sparse signal x, our goal is to compute $y = Ax$ and then recover x using some algorithmic approach. An important sparse recovery technique is l_1-minimization. Using this approach, x can be recovered from $y = Ax$ where A is generated as a random Bernoulli matrix (with values either 1 or -1 with equal probability), as a Gaussian (with matrix entries drawn from a normal distribution), or other matrices which are *sub-Gaussian,*[2] that is, matrices A that satisfy:

$$\mathbb{P}(|A_{ij}| \geq t) \leq \beta e^{-\kappa t^2} \tag{1}$$

$\forall\, t > 0, i \in [m], j \in [n],$ and for some constants $\beta, \kappa > 0$.

In the algorithm implemented for this paper, A was chosen to be *binary* (all values 0 or 1) and sparse. Following the definition above, our matrix A is also sub-gaussian.

1.2 Sparse recovery with random matrices

In sparse recovery, the matrix A can be deterministic or randomly generated. Our choice of matrix is related to some theorems on expander graphs.

For our purposes an (l, ϵ) *expander graph* is a simple bipartite graph $G = (A, B, E)$ of degree d such that $\forall\, X$ in A with $|X| \leq l$, the set of neighbors $N(X)$ of X satisfies $|N(X)| \geq (1-\epsilon)d|X|$. Now also define $\alpha(\epsilon) = 2\epsilon/(1-2\epsilon)$. We have the following result:

Theorem 1. *Let A be an m-by-n adjacency matrix of a $(2k, \epsilon)$-expander graph G. Let $y \in \mathbb{R}^n$ with $Ay = 0$ and let S be any collection of k coordinates of y. Then $||y_S||_1 \leq \alpha(\epsilon)||y||_1$.*

This statement provides some motivation for the unconventional choice (for sparse recovery) of a matrix A that is the adjacency matrix of an expander graph. The proof of this result can be found in [1].

2 Exact recovery algorithm

The sparse recovery algorithm takes as argument an n-dimensional randomly generated sparse signal x_0 with k nonzero entries equal to plus or minus one, at locations randomly generated. This is accomplished by means of a random index-vector. Then this initial signal x_0 is transformed by the random, binary, sparse matrix A to produce our initial approximation $y = Ax_0$. Each column of A is d-sparse and is randomly generated. After computing Ax we try to recover an approximation x^* of x using an l_1 minimization package cited in [4]. The algorithm iterates 11 times, and if the error $x_0 - x^*$ is less than a given threshold on each iteration then the algorithm ceases and returns the sparse recovery.

Here we provide pseudocode for the basic sparse recovery algorithm and some results on synthetic data. Some comments: the initial random signal x_0 remains unchanged for all 11 iterations, while the matrix A is randomly generated upon each iteration. Each column of the matrix A is randomly generated and d-sparse. A has m columns, corresponding to the number of measurements. The signal x_0 is k-sparse. The n-dimensional column vector x is entered as an argument for sparse recovery. The pseudocode for exactrecov is displayed on the next page. The algorithm avoids a lengthy run time and for loops through use of MATLAB's sub2ind function.

Algorithm 1 ExactRecov

```
procedure ExactRecov($x_0$,k,m,d)
Initial measurement:
    $n \leftarrow$ length of $x$
    $x_0 \leftarrow \mathbf{0} \in \mathbb{R}^n$
    $x_0 \leftarrow k$ entries equal to -1 or 1, placed and determined at random
    $A \leftarrow zeros(m, n)$
    $dmat \leftarrow randi([1 : m], d, n)$
    $colind \leftarrow repmat([1 : n], d, 1)$
    $in \leftarrow sub2ind([m, n], dmat(:), colind(:))$
    $A(in) \leftarrow 1$
    $y \leftarrow Ax_0$
    $x^* \leftarrow$ l1-min recovery of $y = Ax_0$
    $error \leftarrow \mathbf{0} \in \mathbb{R}^n$
    $error(1) \leftarrow ||x^* - x0||$
Ten more iterations:
    while $i < 11$
    $A \leftarrow zeros(m, n)$
    $dmat \leftarrow randi([1 : m], d, n)$
    $colind \leftarrow repmat([1 : n], d, 1)$
    $in \leftarrow sub2ind([m, n], dmat(:), colind(:))$
    $A(in) \leftarrow 1$
    $y \leftarrow Ax_0$
    $j \leftarrow patlen$
    $x^* \leftarrow$ l1-min recovery of $y = Ax_0$
    $error(i) \leftarrow ||x^* - x0||$
    $i \leftarrow i + 1$
    if $error(i) \leq 1e - 3$ then
        close;

```

2.1 Methodology

2.2 Results on synthetic data

We tested the basic exactrecov algorithm on randomly generated vectors of size $n = 10$, $n = 100$, $n = 1000$.

For size 10 we generated a random

$$x = [0, -1, 0, 0, -1, 1, 0, -1, 0, 1].$$

Setting $m, k, d = 3$ the algorithm returned the exact recovery of the original. For $n = 10$, $n = 100, k = d = 10, m = 15$ and $n = 1000, m = 50, k = d = 30$, the plots of an example original signal and the recovered sparse signal are displayed at the end of the paper.

3 Discussion

While my algorithm had to be thoroughly debugged, the results were a successful approximate recovery of the synthetic data. The result was not exact for values of n much larger than 10, which was a result of the l_1 minimization package ceasing iteration once the error (defined above) dropped below the threshold of $1e - 3$. Given more time, I would have tried to modify this package.

The significance of this approach is that sparse, binary, and random matrices A are ideal for sparse recovery. This has large advantages for computation speed for reasons that are hinted at in the lemma mentioned in section 1.2 above. My experiments could have been strengthened by moving beyond the types of synthetic data suggested in the paper, where I used initial signals as arguments that were already sparse.

4 Conclusion and next steps

Further approaches would feature different constructions of the matrix A (different probability densities, deterministic rather than random constructions). We could also use different norms to measure convergence, different numbers of iterations, different error tolerances, and compare the performance of these different approaches. Additionally, we could improve the exactrecov algorithm to ensure that the initial signal x_0 has exactly k nonzero values (our current algorithm does not ensure this because the vector of column indices might have duplicates) and to ensure that the columns of A are distinct (which is essential for ensuring that no sparse vectors x enter the nullspace of A during the computation Ax).

Image compression with wavelets would involve a function argument AW^Tx where x denotes a signal taken from an image, W is a wavelet matrix, A is a matrix (generated as above in exactrecov). The function would use linear programming for sparse recovery of x^* and return Wx^* which would be the compressed version of the image. An example of this approach can be found in [3].

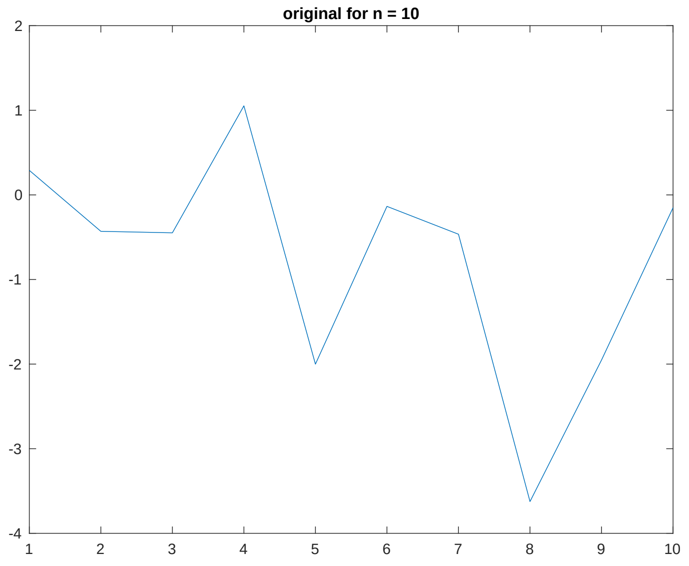
References

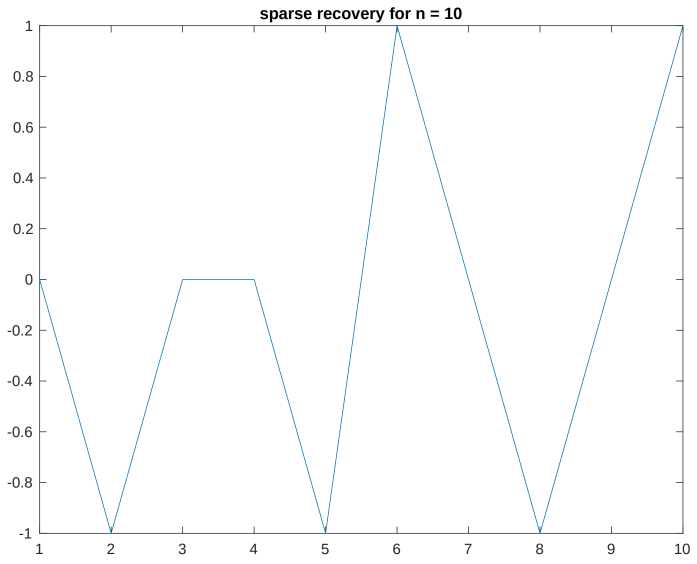
[1] Radu Berinde and Piotr Indyk. "Sparse recovery using sparse random matrices." ArXiv, 26 April 2008.

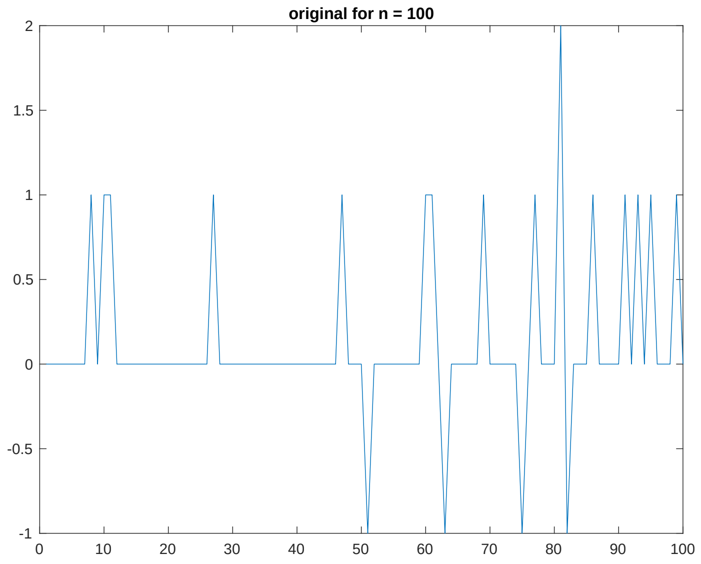
[2] Simon Foucart, Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Springer Monograph, 7 May 2012.

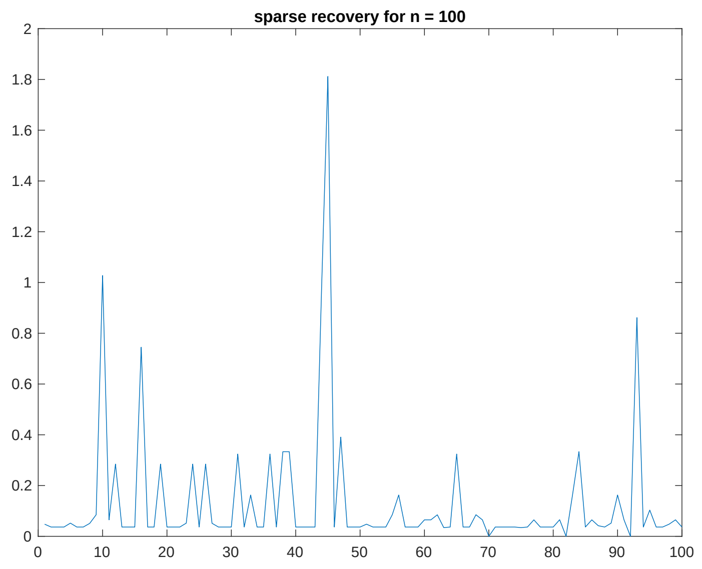
[3] E. J. Cand'es, J. Romberg, and T. Tao. "Stable signal recovery from incomplete and inaccurate measurements." *Comm. Pure Appl. Math.*, 59(8):1208–1223, 2006.

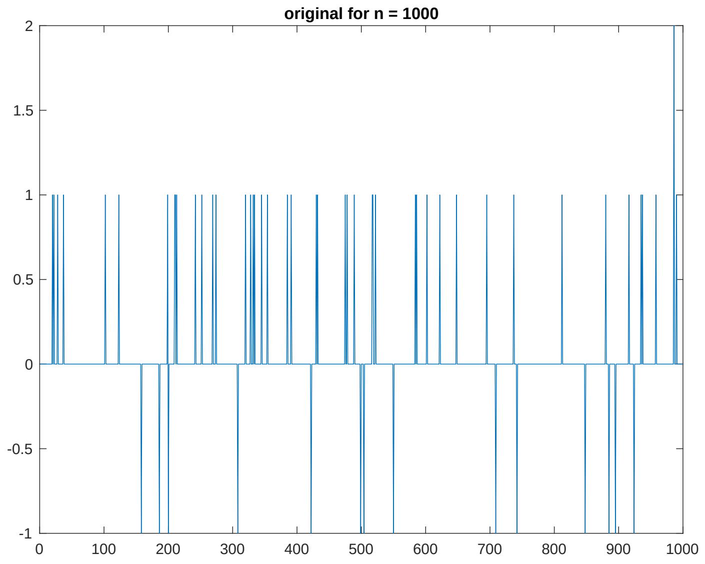
[4] E. J. Cand'es and J. Romberg. l1-MAGIC: Recovery of Sparse Signals via Convex Pro- gramming, 2005. Available at http://www.acm.caltech.edu/l1magic.

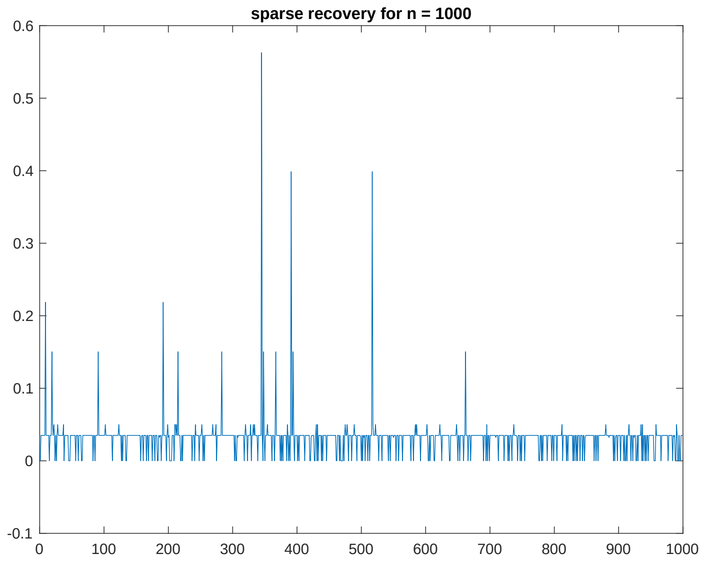

original for n = 10
2
1
0
-1
-2
-3
-4
1
2
3
4
5
6
7
8
9
10


sparse recovery for n = 10
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
1
2
3
4
5
6
7
8
9
10


original for n = 100
2
1.5
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70
80
90
100


sparse recovery for n = 100
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
10
20
30
40
50
60
70
80
90
100


original for n = 1000
2
1.5
1
0.5
0
-0.5
-1
0
100
200
300
400
500
600
700
800
900
1000



sparse recovery for n = 1000
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
0
100
200
300
400
500
600
700
800
900
1000