PRINCETON COS 521: ADVANCED ALGORITHM DESIGN

Compressive Sensing

This lecture is an introduction to compressed sensing. Also known as sparse recovery, compressed sensing is a research area that spans signal processing, imaging, statistics, algorithms, and many other research areas. It is closely connected to *every* previous section of this course.

General Setup

The basic set up for compressed setting is as follows:

- There is hidden vector $x \in \mathbb{R}^n$ which is k-sparse[1]. I.e. $\|x\|_0 = k$.
- We can access *m linear measurements* of x of the form $a_1^T x, \ldots, a_m^T x$.
- We want to recover x from these measurements, i.e. from $y = Ax$ where $A = [a_1, \ldots, a_m]^T$.

[1] In reality, we often care about the scenario when x is *close* to k sparse. The algorithms discussed in this lecture extend to that setting.

This may sound like an artificial problem, but it arises very frequently in applications. We will discuss a few examples shortly.

There are two main questions to ask about the compressed sensing problem. First, when is recovery of x possible? We won't assume anything about x besides its sparsity, so this question becomes, for what matrices A is recovery possible for any sparse vector x? Ideally we want A to have few rows – i.e. to recover x from a small number of linear measurements. The second important question is if x can be recovered *efficiently* from Ax.

The idea is to leverage the sparsity of x. A general n dimensional vector x can only be recovered from $y = Ax$ if A is left invertible: we just compute $A^{-1}y = A^{-1}Ax = x$. This requires A to have at least $m = n$ rows. The hope is to achieve $m \ll n$ rows by exploiting the fact that x is sparse.

Applications

Below are a few example applications of the compressed sensing problem. In these applications, we would typically be interested in slightly more general version of the problem posed above – e.g., the case when x is *close to* but not quite k sparse. We might assume that $\|x - \tilde{x}\|_2 \leq \epsilon$ for some $\tilde{x}$ with $\|\tilde{x}\|_0 = k$. When this is the case, we would hope to recover $\tilde{x}$, or a vector close to it. We can think about this as recovering the k largest entries in x.

We consider the simpler problem in this lecture, but similar techniques generalize to more realistic "noisy" settings.

Example 1 (Streaming heavy-hitters / frequency estimation)**.** *We are given a stream of updates to a vector $x \in \mathbb{R}^n$ of the form $x \leftarrow x \pm e_i$,* *where e_i is a standard basis vector. We want to maintain a compressed representation of x (i.e. in $\ll n$ space) so that, after our stream of updates, we can estimate the k values of x with largest magnitude – these are often call the "heavy-hitters" in x.*

A common way to solve this problem is to maintain some vector $y \in \mathbb{R}^m$ which is initialized with $y = 0$ and, at each update, updated to $y \pm Ae_i$ for some specifically designed matrix $A \in \mathbb{R}^{m\times n}$. The value of this approach is that, at the end of the stream, no matter how many updates were received, or in what order, we will have $y = Ax$. If we can extract x's heavy entries from y, which requires solving the compressed sensing problem, we have solved our streaming heavy-hitters problem.

The heavy hitters problem is very closely related to our homework problem on frequency estimation, where our goal was to output an estimate for each x_i. However, these estimates were only accurate for the top few largest entries in x_i: for small entries it would suffice to output an estimate of 0. So, algorithms for finding heavy-hitters can be used for frequency estimation.

Here is a very cartoonish picture of another application of compressed sensing to imaging technology. This is the "killer app" that most people think of when they think about compressed sensing.

Example 2 (Subsampled Fourier transform)**.** *Consider the discrete Fourier transform matrix $D \in \mathbb{R}^{n\times n}$ where:*

$$D_{hl} = e^{\frac{-2\pi i}{n} h\cdot l}$$

Dx is equal to the discrete Fourier transform of the signal x. Think of each row of D as a complex sinusoid with frequency $\frac{h}{n}$ evaluated at $\ell = 1, 2, \ldots, n$. $[Dx]_k$ is the inner product of x with this sinusoid and thus represents how much of frequency $\frac{h}{n}$ is contained in x.

Think of x as representing the pixels in an image. Many advanced image acquisition techniques, like MRI or ultrasound, can be modeled as obtaining entries of Dx and reconstructing x from those entries. If we collect all entries of Dx, we can find x by simply applying an inverse DFT to Dx. But this can be very expensive in practice. Each entry from the vector Dx is collected by some physical process – i.e. by blasting the object your imaging with a sound or light wave of a particular frequency. So, the more samples we collect, the more time the MRI takes, or the slower the ultrasound.

Compressed sensing asks whether its possible to recover x if we only collect a subset of Dx. We might hope that this is possible if the image data x is sparse, which is the case in many imaging applications. One common approach for choosing a subset of Dx is to sample entries uniformly at random. I.e. to compute SDx for a sampling matrix $S \in \mathbb{R}^{m\times n}$ where each row of S contains a single 1 in a uniformly random location, and all 0s elsewhere. In this case, $A = SD$.

There is a major difference between Examples 1 and 2. In the first, we have complete of freedom in how A is designed. In Example 2 we do not – it must be some subset of rows rows D.

Restricted Isometry Property

In this lecture we will try to understand basic conditions on A that allow us to solve the sparse recovery problem with $< n$ measurements. Ideally these are conditions that can be easily "designed into" A, or hold for certain measurement matrices that arise in practice (like subsampled DFT matrices).

We focus on the following property that has been especially important in understanding compressed sensing:

Definition 1 ((q, ϵ)-Restricted Isometry Property)**.** *A matrix A satisfies (q, ϵ)-RIP if, for all x with $\|x\|_0 \leq q$,*

$$(1-\epsilon)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1+\epsilon)\|x\|_2^2.$$

The RIP property is a Johnson-Lindenstrauss type condition. It requires that A preserves the norm of all q sparse vectors, instead of the norms of a fixed discrete set of vectors. What matrices satisfy this property?

- Random Gaussian matrices with $O(\frac{q\log(n/q)}{\epsilon^2})$ rows. You proved a very slightly weaker version of this result in Howework 3.
- Uniformly subsampled Fourier matrices with $O(q\log^4 n/\epsilon^2)$ rows [2] (or see [3]). You have seen some of the tools to prove this. In Lecture 11 we proved that a subsampled Hadamard matrix, which is a type of Fourier matrix, can be used to give the JL gaurantee. Your Homework 3 argument would thus apply to these matrices as well. Proving RIP for actual Fourier matrices isn't too much of a leap.
- Explicit deterministic constructions with $O(q^2)$ rows (and slightly better [4]). Whether or not $O(q\log n)$ is possible with a deterministic construction is a major open question.

[2]

[3]

[4]

Ultimately, we will show that any matrix A which satisfies $(O(k), O(1))$-RIP can be used as a measurement matrix to reconstruct any k sparse vector x. With these parameters, every construction above allows for a number of measurements $m \ll n$. The first two constructions (random Gaussian and random Fourier) reconstruct x from a compression $y = Ax$ of near optimal size. Roughly it takes $k\log n$ bits just to specify x: we at least need to record where all of its k non-zero entries are and each location takes $\log n$ bits to specify.

A first result on using RIP matrices for compressed sensing is as follows:

Theorem 1 (ℓ_0-minimization)**.** *If A is $(2k, \epsilon)$-RIP for any $\epsilon < 1$ and $\|x\|_0 = k$ then $z^\star = x$ is the unique minimizer of:*

$$\min \|z\|_0 \qquad \textit{subject to} \qquad Az = Ax.$$

In other words, if we measure x with linear measurements from a $(2k, \epsilon)$-RIP matrix A, we can recover the vector from $Ax = y$ by solving the simple minimization problem above. The sparsest z satisfying $Az = y$ will be equal to x. This minimization problem cannot be solved in polynomial time, but there is an $O(n^k)$ time solution: brute force over all subsets of $\leq k$ entries and check (via linear regression) if there exists some z supported on those entries that satisfies $y = Az$.

Proof. Suppose there is some optimal solution $z \neq x$. So $\|x - z\| \neq 0$. Since it is optimal for the ℓ_0 minimization problem, it must be that $\|z\|_0 \leq \|x\|_0 = k$ So $x - z$ is at most $2k$-sparse. It follows from the $(2k, \epsilon)$-RIP property of A that:

$$(1 - \epsilon)\|x - z\|_2 \leq \|Ax - Az\|_2.$$

However, if $Az = Ax$, $\|Ax - Az\|_2 = 0$, so we have a contradiction if $\epsilon < 1$: the right hand side is positive but the left side is 0. So it must be that any optimum z is equal to x. $\square$

Theorem 1 established that, information theoretically, it is possible to recover x from the linear measurements Ax when A satisfies $(O(k), O(1))$-RIP. This is already interesting from a practical perspective, but the result would be much more useful if ℓ_0-minimization was not so slow. Ideally, we recover x from Ax in polynomial time.

Basis Pursuit

It's possible to obtain a polynomial time algorithm by *relaxing* the ℓ_0 minimization problem:

Problem 1 (Basis Pursuit, i.e. ℓ_1 minimization.)**.** *Solve the following linear program:*

$$\min \|z\|_1 \qquad \textit{subject to} \qquad Az = Ax.$$

Why is this a linear program? Let $y = Ax$. By introducing n additional variables $[w_1, \ldots, w_n]$, the above minimization problem can be written as:

$$\min \mathbf{1}^T w \text{ s.t. } Az = y,\ z_i \leq w_i \,\forall i,\ -z_i \leq w_i \,\forall i.$$

This is a linear objective function with linear constraints, so Problem 1 can be solved in polynomial time.

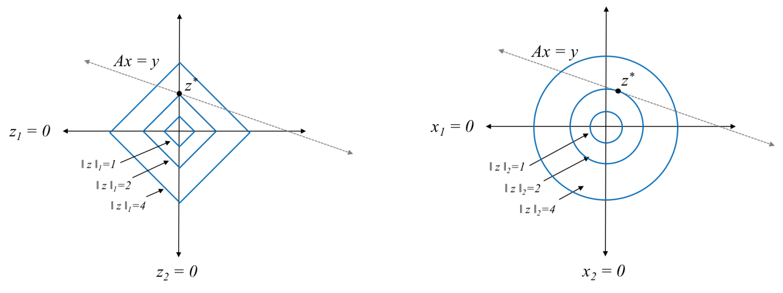
Problem 1 relaxes the ℓ_0 minimization problem of Theorem 1 by replacing the non-convex objective function $\|z\|_0$ with $\|z\|_1$. Why should this be a good relaxation?

In general, an ℓ_1 norm objective promotes sparsity when combined with a linear constraint. Figure 1 gives some intuition. In blue, level sets of $\|z\|_1$ are drawn. Finding the z with smallest ℓ_1 norm is equivalent to finding the point on the line defined by $Ax = y$ which intersects the innermost level set (in this case, we have $n = 2$, so choose A to have just one row).

As we can see, for lines in general position (i.e. random A), with overwhelming probability, the minimum ℓ_1 norm solution will lie at a vertex of a level set, meaning that it will be 1-sparse. This same intuition extends roughly to high dimensions: low dimensional lines/planes are more likely to intersect vertices or edges of the ℓ_1 level sets, which means that we often obtain sparse solutions from ℓ_1 minimization.

This stands in contrast to e.g. ℓ_2 minimization, which as picture in Figure 1, does not promote sparsity.

Our intuition is formalize in the following theorem:



(a) Vertices of level sets of ℓ_1 norm correspond to sparse solutions.

(b) This is not the case e.g. for the ℓ_2 norm.

Figure 1: The goal of interior point methods is to move towards an optimal solution while avoiding immediately approaching the boundary of $\mathcal{K}$.

Theorem 2. *If A is $(3k, \epsilon)$-RIP for $\epsilon < .17$ and $\|x\|_0 = k$, then $z^\star = x$ is the unique optimal solution of Problem 1 (the basis pursuit linear program).*

This theorem only increases the parameter requirements for the RIP property by a constant factor. So for random Gaussian matrices or subsampled Fourier matrices, it still allows recovery with $O(k \log^c n)$ samples. However, the recovery algorithm now runs in polynomial time.

Our proof is similar to that of Theorem 1. If we can show that any two solutions to the basis pursuit linear program must differ by a sparse vector, then we will obtain a contradiction.

In particular, assume there exists some minimizer z for Problem 1 where $z \neq x$. Write $z = x + \Delta$. If $z \neq x$, it must be that $\|\Delta\|_2 > 0$. At the same time $\|A\Delta\|_2 = 0$. Since A satisfies an RIP condition, both of these statements cannot be true when Δ is sparse. To prove Theorem 2, we will extend this argument to when Δ is approximately sparse – i.e. it's ℓ_2 norm is concentrated on a small number of indices.

We start with the following claim:

Claim 1. *Let $S \subset \{1, \dots, n\}$ be the set of indices in x with non-zero value. I.e. if $i \in S$, $x_i \neq 0$. Let $\bar{S}$ denote $\{1, \dots, n\} \setminus S$. Since x is k-sparse, $|S| \leq k$. We claim:*

$$\|\Delta_S\|_1 \geq \|\Delta_{\bar{S}}\|_1$$

Here Δ_S denotes the vector restricted to indices in S (i.e. if you zero'd out all other indices). $\Delta_{\bar{S}}$ denotes the vector restricted to indices in $\bar{S}$.

Proof.

$$\|x\|_1 \geq \|x + \Delta\|_1 = \|x_S + \Delta_S\|_1 + \|\Delta_{\bar{S}}\|_1 \geq \|x_S\|_1 - \|\Delta_S\|_1 + \|\Delta_{\bar{S}}\|_1$$

Rearranging terms proves. □

This bound already suggests that Δ is somewhat well concentrated on the k indices in S. Our next step is to convert this bound to one involving ℓ_2 norms:

Claim 2. *Let $T_1, T_2, \dots$ be sets of size $2k$. T_1 contains the indices of the first $2k$ largest entries in $\Delta_{\bar{S}}$. T_2 contains the indices of the next $2k$ largest entries, so on and so forth. We claim:*

$$\|\Delta_S\|_2 \geq \sqrt{2} \sum_{j \geq 2} \|\Delta_{T_j}\|_2$$

Proof. In this proof we use that fact that for all b with $\|b\|_0 \leq w$,

$$\|b\|_2 \leq \|b\|_1 \leq \sqrt{w}\|b\|_2.$$

The lower bound $\|b\|_2 = \|b\|_1$ is achieved when b is zero on all indices besides one. The upper bound $\|b\|_1 = \sqrt{w}\|b\|_2$ is achieved when b has equal value on exactly w vertices. From the previous claim, we have:

$$\|\Delta_S\|_1 = \|\Delta_{\bar{S}}\|_1 \geq \sum_{j \geq 1} \|\Delta_{T_j}\|_1. \tag{1}$$

We then have that for all $j \geq 1$,

$$\|\Delta_{T_j}\|_1 \geq \sqrt{2k}\|\Delta_{T_{j+1}}\|_2 \tag{2}$$

Why? Every entry in $\Delta_{T_{j+1}}$ is less than the minimum entry in Δ_{T_j}, which is $\leq \|\Delta_{T_j}\|_1/2k$. So we have;

$$\|\Delta_{T_j}\|_2^2 = \sum_{i \in T_j} |\Delta_i|^2 \leq \sum_{i \in T_j} \left(\frac{\|\Delta_{T_{j+1}}\|_1}{2k} \right)^2 \leq 2k \cdot \frac{\|\Delta_{T_{j+1}}\|_1^2}{(2k)^2}.$$

This establishes 2, which substituted into (1) gives:

$$\|\Delta_S\|_1 \geq \sqrt{2k} \sum_{j \geq 2} \|\Delta_{T_j}\|_2.$$

Claim 2 follows because $\|\Delta_S\|_1 \leq \sqrt{k}\|\Delta_S\|_2$. □

We are now ready to prove the main result:

Proof of Theorem 2.

$$\begin{aligned}
0 = \|A\Delta\|_2 &\geq \|A\Delta_{s \cup T_1}\|_2 - \sum_{j \geq 2} \|\Delta_{T_j}\|_2 && \text{(triangle inequality)} \\
&\geq (1-\epsilon)\|\Delta_{s \cup T_1}\|_2 - (1+\epsilon) \sum_{j \geq 2} \|\Delta_{T_j}\|_2 && \text{RIP property of } A \\
&\geq (1-\epsilon)\|\Delta_s\|_2 - \frac{(1+\epsilon)}{\sqrt{2}} \|\Delta_s\|_2 \\
&\geq \left(1 - \epsilon - \frac{(1+\epsilon)}{\sqrt{2}}\right) \|\Delta_s\|_2
\end{aligned}$$

which is a contradiction if $\left(1 - \epsilon - \frac{(1+\epsilon)}{\sqrt{2}}\right) \geq 0$, which happens as long as $\epsilon \lesssim .17$. □

Other considerations

Noise Robustness

The astute student who has been reading engineering papers in French from the 18th century might have noticed that in fact, it is has long been known how to recover a k sparse signal x from *exactly* $2k$ Fourier measurements! This was show by Prony in 1795 [5] using a deterministic, polynomial time algorithm. You can find an analysis here: `http://pages.cs.wisc.edu/~brecht/cs838docs/Lecture06.pdf`.

[5]

What gives? With Prony's method in hand, why does the field of compressed sensing exists at all? The major reason is *noise*. Prony's method and related approaches (e.g. the matrix pencil method) depend on solving highly ill-conditioned linear systems. They often fail due to finite precision computation or when linear measurements are collected in an inexact way imprecise. Moreover, Prony's method fails whenever x is not exactly k-sparse, even if it is very close to k-sparse.

As mentioned, this is nearly always the case in practice. The Basis Pursuit approach adapts naturally to the noisy problem with an analysis very similar to the one seen here (see e.g. `http://people.seas.harvard.edu/~minilek/cs229r/fall13/lec/lec16.pdf`.)

Another major advantage of Basis Pursuit and related compressed sensing methods is that they are not specific to Fourier matrices – they work for much broader classes of measurement matrices (e.g. anything satisfying RIP).

Faster Methods

While Basis Pursuit gives a beautifully simple and robust polynomial time algorithm, there has been a lot of interest in developing even faster algorithms that avoid using the "heavy hammer" of linear programming. One example of such an algorithm is Iterative Hard Thresholding [6], which looks a lot like the projected gradient descent method we saw in class. The idea is to solve $\min_z \|z - Ax\|$ via a gradient method, while continually projection z back to the set of k-sparse vectors. While this set is *not convex* it's possible to show that the method still converges very rapidly. See `http://people.seas.harvard.edu/~minilek/cs229r/fall13/lec/lec18.pdf` for an analysis. The runtime of the method is approximately $O(nk \log n)$ for Gaussian measurement matrices and $O(n \log n)$ for subsampled Fourier matrices. Other "first order" type methods obtain similar running times. If you are interested in learning more, some algorithms to look up include "Orthogonal Matching Pursuit", "CoSaMP", and "Subspace Pursuit".

[6]

Even Faster Methods

Remarkably, for a subsampled Fourier measurement matrix, it's actually possible to go even faster! In particular, the linear dependence on n can be reduced to a logarithmic dependence, giving algorithms that run in $O(k \log^c n)$ time. See [7] and the earlier results cited in that paper.

[7]

Methods achieving this sort of runtime are refered to as "Sparse Fourier Transform" algorithms. Notably, they run faster than the time to compute the Fourier transform of a length n signal. Accordingly, SFT algorithms are useful beyond compressed sensing. Suppose we have the entire vector $y = Dx$ in hand (i.e. there is no measurement acquisition cost) and we believe that x is sparse (as is often the case in practice). We can compute x more quickly by using an SFT algorithm (which begins by selecting a subset of entries in y) than by computing the inverse Fourier transform $x = D^{-1}y$.

Even these faster algorithms have a linear dependence on n. There's no obvious reason this should be necessary as our input Ax and output x are both of size $O(k \log^c n)$.

Note that all of these methods are specific to Fourier measurement matrices – they do not work for general RIP matrices.

Bibliography

[1] Rudelson, M. and Vershynin, R., On sparse reconstruction from Fourier and Gaussian measurements. Comm. Pure Appl. Math., 61: 1025-1045, 2008.

[2] Haviv, Ishay and Regev, Oded, The Restricted Isometry Property of Subsampled Fourier Matrices. Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 288–297, 2016.

[3] Jean Bourgain, Stephen Dilworth, Kevin Ford, Sergei Konyagin, and Denka Kutzarova, Explicit constructions of RIP matrices and related problems. Duke Math. J., 159:1 145–185, 2011.

[4] Thomas Blumensath and Mike E. Davies, Iterative hard thresholding for compressed sensing. Applied and Computational Harmonic Analysis, 27:3, 265–274, 2009.

[5] Gaspard Riche de Prony, Essay experimental et analytique: sur les lois de la dilatabilite de fluides elastique et sur celles de la force expansive de la vapeur de l'alcool, a differentes temperatures. Journal de l'Ecole Polytechnique, 24–76. 1795.

[6] Haitham Hassanieh, Piotr Indyk, Dina Katabi, Eric Price, Nearly optimal sparse fourier transform. Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing (STOC), 2012.