CS 229r: Algorithms for Big Data Fall 2015

Lecture 21 — November 12, 2015

Prof. Jelani Nelson *Scribe: Zezhou Liu*

1 Overview

Previously, we looked at basis pursuit and *iterative hard thresholding (IHT)* for ℓ_2/ℓ_1 sparse signal recovery. Recall that from before the ℓ_2/ℓ_1 guarantee: time per iteration depends on matrix-vector multiplication time by Π, Π^T in the IHT algorithm. To make this fact, the only way we know is to use a fast such Π, such as sampling rows from the Fourier matrix. However, for those Π we do not know how to get RIP with only $O(k\log(n/k))$ sampled rows (the best known proof currently requires an extra factor of $\log^2 k$ in the number of measurements).

Today, we take a look at ℓ_1/ℓ_1 sparse signal recovery using expanders. Our goal is to get a good recovery guarantee using $O(k\log(n/k))$ measurements with fast time per iteration in an iterative algorithm; for this though we will relax from achieving the ℓ_2/ℓ_1 guarantee to achieving the weaker ℓ_1/ℓ_1 guarantee (as you will show on the current problem set, ℓ_1/ℓ_1 is indeed a weaker guarantee). Specifically, we aim for the guarantee of finding $\tilde{x}$ such that

$$\|x - \tilde{x}\| \leq C \cdot \|x_{tail(k)}\|_1 \tag{1}$$

2 RIP_1 matrices for signal recovery

Definition 1. *A matrix Π satisfies the (ε, k)-RIP_1 property if for all k-sparse vectors x*

$$(1-\varepsilon)\|x\|_1 \leq \|\Pi x\|_1 \leq \|x\|_1$$

Definition 2. *Let $G = (U, V, E)$ be a left d-regular bipartite graph (every node on left is adjacent to d nodes on the right) with left vertices U, right vertices V and edges E. The graph G is a (k,ε)-expander if for all $S \subseteq U$ where $|S| \leq k$:*

$$|\Gamma(S)| \geq (1-\varepsilon)d|S|.$$

Here, $\Gamma(S)$ denotes the neighborhood of S. Intuitively, this means that no matter what S you choose (as long as it is relatively small $|S| \leq k$), you won't have too many collisions amongst the neighbors of vertices in S.

Claim 3. *There exist d-regular (k,ε)-expanders satisfying*

- $n = |U|$
- $m \lesssim |V| = \mathcal{O}\left(\frac{k}{\varepsilon^2}\log\left(\frac{n}{k}\right)\right)$

- $d \le \mathcal{O}\big(\frac{1}{\varepsilon^2}\log\big(\frac{n}{k}\big)\big)$

This claim can be proven by picking G at random with the specified n, d, m, then showing that G satisfies the expansion condition with high probability (by union bounding over all $S \subset [n]$ of size at most k). Thus, this approach is not constructive, although it does provide a simple Monte Carlo randomized algorithm to get a good expander with high probability.

In fact [4] shows that you can construct d-regular (k,ε)-expanders deterministically with:

- $n = |U|$
- $m = |V| = \mathcal{O}\big(d^2 k^{1+\alpha}\big)$
- $d = \mathcal{O}\big(\frac{1}{\varepsilon}\log(k)\log(n)\big)^{1+\frac{1}{\varepsilon}}$

where α is an arbitrarily small constant.

Given a d-regular (k,ε)-expander G, we can construct the following $\Pi = \Pi_G$:

$$\Pi = \Pi_G = \frac{1}{d}A_G$$

where $A_G \in \mathbb{R}^{m\times n}$ is the bipartite adjacency matrix for the expander G and each column of A_G contains exactly d non-zero (1) entries and the rest zeros. Then, $\Pi_G \in \mathbb{R}^{|V|\times|U|}$ has exactly d non-zero entries (equal to $1/d$) in each column. The average number of non-zero entries per row will be nd/m; and indeed, if Π is picked as a random graph as above then each row will have $O(nd/m)$ non-zeroes with high probability. Thus Π is both row-sparse and column-sparse.

Theorem 4 ([1, Theorem 1])**.** *If G is a d-regular (k,ε)-expander, then Π_G is $(2\varepsilon,k)$-RIP$_1$.*

Definition 5. *The matrix Π satisfies the "C-restricted nullspace property of order k" if for all $\eta \in \mathrm{Ker}(\Pi)$ and for all $S \subseteq [n]$ where $|S| = k$*

$$\|\eta\|_1 \le C\|\eta_{\bar{S}}\|_1.$$

It is known that if Π satisfies $(C,2k)$-RNP, then for small enough C, if $\tilde{x}$ is the solution form basis pursuit, then

$$\|x-\tilde{x}\|_1 \le O(1)\cdot\|x_{tail(k)}\|_1$$

For example see the proof in [5].

Note the above is not nice for a few reasons. First, it is an abstraction violation! We would like to say RIP$_1$ alone suffices for basis pursuit to give a good result, but unfortunately one can come up with a counter-example showing that's not true. For example, Michael Cohen provided the counter example where Π is $n\times(n-1)$ with ith column e_1 for $i = 1,\ldots,n-1$, and the last column is $(n-1)^{-1}(1,\ldots,1)$. This Π is RIP$_1$ with good constant even for $k \in \Theta(n)$, but it does not have the restricted nullspace property (consider how it acts on the vector $(1,1,\ldots,1,-(n-1))$ in its kernel).

Second, it is slow: we are trying to avoid solving basis pursuit (we could already solve basis pursuit with subgaussian RIP matrices with $O(k\log(n/k))$ rows that provide us with the stronger ℓ_2/ℓ_1 guarantee!). Thus, we will switch to iterative recovery algorithms.

3 Iterative Recovery Algorithms

There are a number of iterative recovery algorithms. Here, we include a couple of them as well as their results.

- Expander Matching Pursuit (EMP) [IR08] does not use the RIP_1 but relies directly on Π coming from an expander; it gets $C = 1+\varepsilon$ for ℓ_1/ℓ_1 recovery using an $(O(k), O(\varepsilon))$-expander. This is the best known iterative ℓ_1/ℓ_1-sparse recovery algoithm in terms of theoretically proven bounds.
- Sparse Matching Pursuit (SMP) [1] also does not use the RIP_1 abstraction but relies on expanders; it gets $C = O(1)$ using an $(O(k), O(1))$-expander. SMP performs better than EMP in practice, despite the theoretical results proven being not as good.
- Sequential Sparse Matching Pursuit (SSMP) [2] was originally analyzed with a reliance on expanders; it gets $C = O(1)$ using an $(O(k), O(1))$-expander. Price [6] later showed how to analyze the same algorithm relying only on Π being $(O(1), O(k))$-RIP$_1$, thus not relying on expanders explicitly.

In the remainder we describe SSMP and the analysis of [6].

4 Sequential Sparse Matching Pursuit (SSMP)

Here is the pseudocode for SSMP recovery given $b = \Pi x + e$, where Π an RIP$_1$ matrix. The vector $e \in \mathbb{R}^m$ is the error vector.

1. Initialize $x^{[0]} \leftarrow 0$
2. for $j = 1$ to T:
   (a) $x^{[j,0]} \leftarrow x^{[j-1]}$
   (b) for $a = 1$ to $(c-1)k$:
      i. $(i, z) = argmin_{(i,z)}(\|b - \Pi(x^j + z \cdot e_i)\|_1)$
      ii. $x^{[j,a]} \leftarrow x^{[j,a-1]} + z \cdot e_i$
   (c) $x^{[j]} \leftarrow H(x^{[j,(c-1)k]})$
3. return $x^{[T]}$

Note finding the (i, z) minimizing the above expression can actually be done quickly using a balanced binary search tree. Suppose Π is d-sparse in each column and r-sparse in each row. We create a max priority queue Q in which there are n keys $1, \ldots, n$, where key i has value equal to how much $\|b - \Pi(x^j + z \cdot e_i)\|_1$ is decreased when z is chosen optimally. Then to find (i, z) which is the argmin above, we remove the key in the tree with the smallest value then make the corresponding update by adding $z \cdot e_i$ to our current iterate. Note that doing this affects the search tree values for every other $j \in [n]$ such that the jth column of Π and the ith column of Π both have a non-zero entry in the same row for at least one row. Thus the total number of j whose values are affected is at most dr. Finding the new optimal z and calculating the new value for each takes $O(d)$ time for each one. When using an expander to construct Π, r is $O(nd/m)$. Thus we may have to adjust keys for $O(nd^2/m)$ other elements in the priority queue, taking $O((nd^2/m)(d + \log n))$ time total per iteration of the inner loop. There are $O(k)$ iterations through the inner loop, and thus each outer loop takes time $O((nd^2k/m)(d + \log n))$. For an optimal (k, ε)-expander we have $m = \Theta(kd)$ and $d = O(\log n)$, so the total time per outer loop is $O(nd \log n) = O(n \log(n/k) \log n)$.

Now for the error analysis. Without loss of generality, x is k-sparse. If not and $x = x^k + (x - x^k)$ (where x^k is the best k-sparse approximation to x), then we can rewrite $\Pi x + e = \Pi x^k + (e + \Pi(x - x^k))$. Now define $\tilde{e} = (e + \Pi(x - x^k))$. Then

$$\|\tilde{e}\|_1 \leq \|e\|_1 + \|\Pi(x - x^k)\|_1 \leq \|e\|_1 + \|x - x^k\|_1 \tag{2}$$

Although $x - x^k$ is not k-sparse, note that $\|\Pi z\|_1 \leq \|z\|_1$ for *any* vector z, since we can just break up z into a sum of sparse vectors (by partitioning coordinates) then applying the triangle inequality.

Henceforth we assume x is k-sparse, and our goal is to show:

$$\|x - x^{[T+1]}\|_1 \leq C \cdot [2^{-T} \cdot \|x - x^{[0]}\|_1 + \|e\|_1] \tag{3}$$

Note when x isn't actually k-sparse, the $2^{-T}\|x - x^{[0]}\|$ term is actually $2^{-T}\|H_k(x)|$, where H_k is the hard thresholding operator from last lecture. With the exception of the $2^{-T} \cdot \|H_k(x)\|_1$ term (which should decrease exponentially with increasing iterations T), this is our ℓ_1/ℓ_1-error. This means at any particular iteration $j + 1$, we have the following subgoal:

$$\|x - x^{[j+1]}\|_1 \leq 1/2 \cdot \|x - x^{[j]}\|_1 + c' \cdot \|e\|_1 \tag{4}$$

If we can show this, then (3) follows by induction on j (and making C big enough as a function of c' to make the inductive step go through).

5 SSMP Proof

First, let's allow some notation $y^{[j]} = x - x^{[j]}$ and $y^{[j,a]} = x - x^{[j,a]}$. We will show this proof in four steps.

5.1 Step 1.

We show each iteration of the inner loop decreases error by $(1 - \frac{1}{2_{k+a}})^{1/2}$. This is to say, as long as the error is at least $c''\|e\|_1$ for some c'',

$$\|\Pi x^{[j+1,a]} - b\|_1 \leq (1 - \frac{1}{2_{k+a}})^{1/2} \cdot \|\Pi x^{[j+1,a]} - b\|_1 \tag{5}$$

Note the interesting case is indeed when the error is still at least $c''\|e\|_1$ (since iterating beyond that point just always keeps us at error $O(\|e\|_1)$. Let's assume the above is true for the rest of the proof.

5.2 Step 2.

Given Step 1 of the proof, we show that since we run the inner loop $(c-1)k$ times, then at the end of the loop when $a = (c-1)k$, we have:

$$\|\Pi x^{[j+1,(c-1)k]} - b\|_1 \leq [\prod_{a=0}^{t-1}(1 - \frac{1}{2_{k+a}})^{1/2} \cdot \|\Pi x^{[j]} - b\|_1$$

Then $(1 - \frac{1}{2_{k+a}}) = \frac{2k+a-1}{2k+a}$, which implies that when $c = 127$ we get something like $t = (c-1)k = 126k$, which makes this coefficient at most $\frac{1}{8}$. As a result, we get:

$$\|\Pi x^{[j+1,(c-1)k]} - b\|_1 \leq \frac{1}{8}\|\Pi x^{[j]} - b\|_1 \tag{6}$$

5.3 Step 3.

Recall that $b = \Pi x + e$, we can substitute it back in, and then use triangle inequality to show

$$\begin{aligned}
\|\Pi x^{[j+1,t]} - b\|_1 &= \|\Pi(x^{[j+1,t]} - x) - e\|_1 &(7)\\
&\geq \|\Pi(x^{[j+1,t]} - x)\|_1 - \|e\|_1 &(8)\\
&\geq (1-\varepsilon) \cdot \|x^{[j+1,t]} - x\|_1 - \|e\|_1 &(9)
\end{aligned}$$

Re-arranging the equation and then applying some arithmetic and using $\varepsilon < 1/2$ gives us:

$$\begin{aligned}
\|x^{[j+1,t]} - x\|_1 &\leq \frac{1}{1-\varepsilon} \cdot (\|\Pi x^{[j+1,t]} - b\|_1 + \|e\|_1)\\
&\leq 2\|\Pi x^{[j+1,t]} - b\|_1 + 2\|e\|_1\\
&\leq \frac{1}{4}\|\Pi x^{[j]} - b\|_1 + 2\|e\|_1\\
&\leq \frac{1}{4}\|\Pi(x^{[j]} - x)\|_1 + \frac{9}{4}\|e\|_1\\
&\leq \frac{1}{4}\|x^{[j]} - x\|_1 + \frac{9}{4}\|e\|_1
\end{aligned}$$

5.4 Step 4.

Here, we show that the previous result implies $\|x^{[j+1]} - x\|_1 \leq 1/2 \cdot \|x^{[j]} - x]|_1 + \frac{9}{2}\|e\|_1$. Notice the first step is by adding an identity, the second is by triangle inequality, and the last is by using the results form step 3.

$$\begin{aligned}
\|x^{[j+1]} - x\| &= \|x^{[j+1]} - (x^{[j+1,t]} + x^{[j+1,t]}) - x\|_1 \\
&\leq \|x^{[j+1]} - x^{[j+1,t]}\| + \|x^{[j+1,t]} - x\|_1 \\
&\leq 2\|x - x^{[j+1,t]}\|_1 \qquad (10) \\
&\leq 1/2 \cdot \|x^{[j]} - x]\|_1 + \frac{9}{2}\|e\|_1
\end{aligned}$$

where (10) follows since $x^{[j+1]}$ is the best k-sparse approximation to $x^{[j+1,t]}$, whereas x is some other k-sparse vector.

6 Lemmas

Now it just suffices to establish Step 1. It relies on a few lemmas, which are proven in [6].

Lemma 6. *Suppose you have a bunch of vectors $r_1, ..., r_s \in R^m$ and $z = \mu + \sum_{i=1}^{s} r_i$ where $\|\mu\|_1 \leq c \cdot \|z\|_1$ then if*

$$(1-\delta)\sum \|r_i\|_1 \leq \|\sum r_i\|_1 \leq \sum \|r_i\|_1$$

then there exists i such that

$$\|z - r_i\|_1 \leq (1 - \frac{1}{s}(1 - 2\delta - 5c))\|z\|_1$$

Intuitively the condition on the r_i implies that there is not much cancellation when they are summed up, so not much ℓ_1 mass is lost by summing. In the case when there is no cancellation at all, then obviously (if μ were zero, say) any non-zero r_i could be subtracted from the sum to decrease $\|z\|_1$. The above lemma captures this intuition even when there can be a small amount of cancellation, and a small norm μ is added as well (think of c as being a very small constant).

Now we have the next lemma, which can be proven by the previous one.

Lemma 7. *If Π is (s, 1/10)-RIP$_1$ and $s > 1$, then if y is s-sparse, $\|w\|_1 \leq 1/30\|y\|_1$ then there exists a 1-sparse z such that $\|\Pi(y - z) + w\|_1 \leq (1 - \frac{1}{s})^{1/2}\|Piy + w\|_1$.*

This is to say that at every step choose the best 1-sparse to add to decrease the error.

Now, step 1 follows by noting that $x^{j,a}$ is $(k + a)$-sparse, so $x^{j,a} - x$ is $2k + a \leq (c + 1)k$ sparse. Thus there is one-sparse update (by the above lemma) which decreases the error (and note SSMP finds the best one-sparse update in each iteration of the inner loop, so it does at least as well).

References

[IR08] Piotr Indyk and Milan Ružić. Near-optimal sparse recovery in the l1 norm. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 199–207. IEEE, 2008.

[1] Radu Berinde, Anna Gilbert, Piotr Indyk, Howard Karloff, and Martin Strauss. Combining geometry and combinatorics: a unified approach to sparse signal recovery. *Allerton*, 2008.

[2] Radu Berinde, Piotr Indyk. Sequential sparse matching pursuit. *Allerton*, 2009.

[3] Thomas Blumensath, Mike E. Davies. Iterative hard thresholding for compressed sensing. *Appl. Comput. Harmon. Anal.*, 27:265–274, 2009.

[4] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. In *IEEE Conference on Computational Complexity*, pages 96–108. IEEE Computer Society, 2007.

[5] Piotr Indyk, Ronitt Rubinfeld. Sublinear algorithms. `http://stellar.mit.edu/S/course/6/sp13/6.893/courseMaterial/topics/topic2/lectureNotes/riplp/riplp.pdf`

[6] Eric Price. Improved analysis of Sequential Sparse Matching Pursuit. *Unpublished manuscript*, 2010.