4. MSRI Graduate Summer School Notes
Jeff Blanchard, 2018
Fundamental Decoders

In Chap. 3 we were introduced to the ℓ_0 decoder and derived sufficient conditions based on spark and coherence that guarantee the ℓ_0 decoder will return the sparsest solution to our underdetermined compressed sensing problem. Although we omitted the proof of the NP hardness of the ℓ_0 decoder, it is relatively clear that employing a naïve search for the solution to this combinatorial optimization problem is simply not reasonable for anything but the smallest values of n. This chapter pulls together some results that give us hope of something better than an exhaustive search. The following exploration and guided tour through some of these proofs is based on work of many researchers. It certainly was not the case that it all appeared this easy in the beginning. This is the best of computational mathematics: a seemingly intractable problem will be solved with very elementary algorithms which boast sufficient conditions guaranteeing convergence to the correct solution.

Before we begin think of the first time you learned Newton's method for root finding. It was possibly in a first course on calculus. If you weren't amazed by this ridiculously efficient method of finding the solution to a nonlinear equation, you simply didn't reflect on it long enough. Do so now! Say you want to find the root of 37^{th} degree polynomial with rational coefficients. Can you do this by hand? Sure, provided you are willing to spend the rest of your life factoring. This is akin to an exhaustive search. Instead, take the derivative of the polynomial using the fantastically simple power rule. Now guess a decent approximation of the root you are after and employ Newton's method. In a few iterations, you'll have your solution with very good precision. We're about to do the same thing here.

4.1. **Thresholding.** The first algorithm is so simple there should be absolutely no way it ever works. Surprisingly, it will work quite well provided the encoder has a reasonable coherence. Here is the idea. If the vector x is very sparse, we'll pretend A is unitary and multiply the measurements y by A^*. If A was indeed unitary, we would have the solution. Since it isn't even square, it certainly is not unitary, but from this very rough approximation of the pseudo inverse we will decide to select the locations of largest entries of $|A^*y|$ of our incorrect solution as the support set of our approximation. Then, we simply project the measurements y onto this support by solving an overdetermined least squares problem.

This task of choosing, or detecting, a support set will appear often in our algorithms. It appears often enough to give it a name. The subroutine $\text{PrincipalSupport}_k(z)$ finds indices of the k largest magnitude values of the vector z:

$$S = \text{PrincipalSupport}_k(z) = \{i : |z_i| \text{ is one of the } k \text{ largest values in } |z|\}.$$

This notation implies that the subroutine knows k before hand. In fact, the subroutine must be told the value of k but it will always be invoked in an algorithm which does indeed have a value for k.

Algorithm 1 Thresholding Decoder

Input: A, y, k
Output: A k-sparse approximation $\hat{x}$ of the target signal x

1: $x = A^*y$ (rough approximation of inverse)
2: $T = \text{PrincipalSupport}_k(x)$ (estimate for support set)
3: $\hat{x}_T = A_T^\dagger y$ and $\hat{x}_{T^c} = 0$ (projection onto k subspace defined by T)

This simple algorithm has a few computational advantages. If we simply solved the least squares problem on the underdetermined system of equations, we would likely end up with dense solution. This method ensures we end up with a k-sparse approximation. The first step in Alg. 1 is a simple matrix-vector multiplication and requires $O(mn)$ operations rather than computing the pseudo inverse via the full SVD. Then, after determining the support set, we compute the pseudo inverse via the SVD of a much smaller submatrix, $A_T \in \mathbb{R}^{m\times k}$. This is only $O(mk^2)$ rather than $O(mn^2)$. Since k is assumed to be much smaller than n, this is an enormous computational savings. Best of all, this algorithm returns the same solution as the ℓ_0 decoder far more often than one could possibly hope.

To prove a sufficient condition for exact recovery of a k sparse vector, we need to define an important quantity for the vector x. This ℓ_∞ ratio describes the range of magnitudes of the nonzero elements of x.

Definition 4.1. The ℓ_∞ *ratio* of a vector $x \in \mathbb{R}^n$ is the ratio of the smallest and largest magnitude in x:

$$\nu_\infty(x) = \frac{\min_{j \in \text{supp}(x)} |x_j|}{\|x\|_\infty}.$$

We are now prepared to state and prove our first sufficient condition for a computationally tractable decoder in compressed sensing. The thresholding decoder is guaranteed to find the sparest solution to the underdetermined problem $y = Ax$ whenever A has small coherence and the measured vector x has an appropriate ℓ_∞ ratio. Note that the following result is similar, although stricter, than the result for the ℓ_0 decoder, Thm. 3.3.2. However, when the ℓ_∞ ratio of x is 1, in other words when all nonzeros in x have the same magnitude, the results are identical!

Theorem 4.1.1. *Let $x \in \mathbb{R}^n$ be a k-sparse vector, $A \in \mathbb{R}^{m \times n}$ have unit ℓ_2 norm columns, and $y = Ax$. The approximation $\hat{x}$ returned by the Thresholding decoder (Alg. 1) satisfies $\hat{x} = x$ provided*

$$k < \frac{\nu_\infty(x) \cdot \mu(A)^{-1} + 1}{2}.$$

We prove this theorem with a series of four exercises.

Exercise 4.1. Let $w = A^* y = A^* A x$. Prove that for $i \notin \text{supp}(x)$,

$$|w_i| \leq k\mu(A)\|x\|_\infty.$$

Exercise 4.2. Let $w = A^* y = A^* A x$. Prove that for $i \in \text{supp}(x)$,

$$|w_i| \geq |x_i| - (k-1)\mu(A)\|x\|_\infty.$$

Exercise 4.3. Using Exer. 4.1 and 4.2, prove that if

$$|x_i| - (k-1)\mu(A)\|x\|_\infty > k\mu(A)\|x\|_\infty,$$

then $i \in \text{PrincipalSupport}_k(A^* y) \cap \text{supp}(x)$.

Exercise 4.4. Use Exers. 4.1–4.3 to prove Thm. 4.1.1.

This should be rather satisfying! If x has all equal magnitude entries, the sufficient coherence condition on A which ensures the ℓ_0 decoder will find the correct solution also implies that we can do something much simpler. The same condition ensures the Thresholding decoder will return exactly the same solution. You should give this a try (and we will in Exer. 4.14). The (theoretical) penalty for using this simple algorithm to solve this seemingly intractable problem is when the smallest nonzero element is much smaller than the largest, our guarantee is far weaker.

4.2. **Orthogonal Matching Pursuit.** One potential reason to be dissatisfied with Thresholding is that the result is impacted by the range of magnitudes in the nonzero values in our sparse vector, i.e. the ℓ_∞ ratio. We can see from our proof that this appears in the analysis because in order to select the correct indices for the support, we must have the minimum magnitude of $|A^* y|$ on the support be larger than maximum magnitude of $|A^* y|$ off the support. Well, surely there is a better chance to select a correct index if we do this one at a time. In that case we only need that the *maximum* magnitude of $|A^* y|$ on the support is greater than the maximum magnitude off the support.

This next algorithm will select the support set one entry at a time. As it builds the support set, it is simultaneously selecting the columns of A that are involved in encoding x. Therefore, we are building a basis for the column space of A one column at a time. When we have selected the columns of A that correspond to the support of x we have exactly selected the subspace of the column space in which y resides. At each step along the way, we want the best approximation of y to the current subspace, so we solve a least squares problem. This guarantees that once we select an index for our approximate support set, we will never select it again. Why not?

This is the first algorithm we can be truly satisfied with. With good coherence properties, we can ignore the ℓ_∞ ratio and guarantee that OMP will recover the k-sparse vector exactly. How good must the coherence properties be? Well, that's the amazing part: it is exactly the same condition we required to guarantee the

Algorithm 2 Orthogonal Matching Pursuit (OMP)

Input: A, y, k
Output: A k-sparse approximation $\hat{x}$ of the target signal x

Initialization: **Set** $x^0 = 0, r^0 = y, T^0 = \{\}$;

Iteration:

1: **for** $j = 1, 2, \ldots, k$ **do**
2: $i = \text{argmax}_l |A_l^* r^{j-1}|$; (identify column of A most correlated to residual)
3: $T^j = T^{j-1} \cup \{i\}$; (add the new column index to the index set)
4: $x^j_{T^j} = A^\dagger_{T^j} y$ and $x^j_{(T^j)^c} = 0$; (project the measurements onto the T subspace)
5: $r^j = y - Ax^j$; (update the residual)
6: **end for**
7: **return** $\hat{x} = x^k$ (return the k-sparse vector x^k)

ℓ_0 decoder will find the exact solution. What we really require is the k-sparse vector x is the unique solution to the combinatorial optimization problem.

Theorem 4.2.1. *Let $y = Ax$ with $x \in \mathbb{R}^n$, $\|x\|_0 = k$, and $A \in \mathbb{R}^{m \times n}$ with unit ℓ_2 columns. If*

$$k < \frac{\mu(A)^{-1} + 1}{2},$$

then after k iterations of the Orthogonal Matching Pursuit decoder (Alg. 2), $\hat{x} = x^k = x$.

The tour through this proof takes just a few more steps, but two of them are very similar to exercises we have already been completed. For each of the following exercises, assume the hypotheses of Thm. 4.2.1 and the notation from Alg. 2.

Exercise 4.5. Prove that for any $j < k$, $A_l^* r^j \neq 0$ for some $l \in \{1, 2, \ldots, n\}$. (What if this were not true? What would that imply about x^j? Why is this a contradiction?)

Exercise 4.6. Prove that for $j \leq k$ and any $l \in T^j$, $A_l^* r^j = 0$. (This is more of an observation. Invoke a fact from Section 1.3.)

Exercise 4.7. Prove that $\text{supp}(x^j) \subset \text{supp}(x)$ for all $j \leq k$. Follow this outline:

a. Establish a base case and make the inductive hypothesis $T_j \subset \text{supp}(x)$.
b. Use the inductive hypothesis to show that if $i \notin \text{supp}(x)$ and $w = A^* r^j$,

$$|w_i| \leq k\mu(A)\|x - x^j\|_\infty.$$

c. Use the inductive hypothesis to show that if $i \in \text{supp}(x)$ and $w = A^* r^j$,

$$|w_i| \geq |(x - x^j)_i| - (k-1)\mu(A)\|x - x^j\|_\infty.$$

d. Employ the bounds from the two preceding steps to show that Alg. 2, Step 2 will correctly select an index in the support set.

Exercise 4.8. Combine Exers. 4.5–4.7 to prove Thm. 4.2.1.

4.3. **ℓ_1-minimization.** Our first decoder was the ℓ_0 decoder (Def. 3.6) which solved the combinatorial optimization problem:

$$x^\star = \text{argmin}_{z \in \mathbb{R}^n} \|z\|_0 \quad \text{subject to } y = Az = Ax.$$

We have now also seen that Thresholding and Orthogonal Matching Pursuit will find the same solution as the ℓ_0 decoder under the sufficient coherence based conditions on the encoder A. In this section, we investigate the most widely studied compressed sensing decoder, the ℓ_1 decoder. The ℓ_1 decoder solves a convex optimization problem and is called the *convex relaxation* or *convex envelop* of the ℓ_0 problem.

Definition 4.2 ℓ_1 decoder. Suppose $y = Ax \in \mathbb{R}^m$. The solution to the convex optimization problem

$$x^\star = \text{argmin}_{z \in \mathbb{R}^n} \|z\|_1 \quad \text{subject to } y = Az = Ax. \tag{5}$$

is the output of the ℓ_1 decoder.

4.3.1. *The Null Space Property.* There is beautiful geometric theory which provides a necessary and sufficient condition on the encoder A in terms of the polytope formed by projecting the unit ℓ_1 ball in $\mathbb{R}^n$ into $\mathbb{R}^m$. In this section we will study an alternative necessary and sufficient condition on A known as the *Null Space Property*.

Definition 4.3 Null Space Property. A matrix $A \in \mathbb{R}^{m\times n}$ satisfies the the *null space property of order* k, abbreviated $NSP(k)$, if for every set $T \subset \{1,\dots,n\}$ with $|T| \le k$ and for every $v \in \text{null}(A) - \{0\}$,

$$\|v_T\|_1 < \|v_{T^c}\|_1. \tag{6}$$

The null space property or order k can be shown to be both necessary and sufficient for showing that every k-sparse vector is the unique solution to (5). For our purposes, we will focus on the sufficient conditions although you are encouraged to establish that it is also necessary. We ultimately want to know when the output of the ℓ_1 decoder is equal to the output of the ℓ_0 decoder.

Theorem 4.3.1. *Suppose $x \in \mathbb{R}^n$ with $\|x\|_0 = k$. Suppose $A \in \mathbb{R}^{m\times n}$ satisfies the null space property of order k. Let $y = Ax \in \mathbb{R}^m$. Then x is the unique output of the ℓ_1 decoder, namely $\hat{x} = x$.*

Exercise 4.9. Prove Thm. 4.3.1 following this outline:
a. Let $T = \text{supp}(x)$ and suppose $z \in \mathbb{R}^n$ satisfies $z \neq x$ but $Az = Ax$. Show $\|(x-z)_{T^c}\|_1 = \|z_{T^c}\|_1$.
b. Observe that $\|x\|_1 = \|x_T\|_1$ and use the triangle inequality and $NSP(k)$ to show $\|x\|_1 < \|z\|_1$.

4.3.2. *Coherence and ℓ_1 minimization.* We now establish that the same coherence condition that guarantees k-sparse recovery for the ℓ_0 and OMP decoders is also sufficient for the ℓ_1 decoder.

Theorem 4.3.2. *Suppose $A \in \mathbb{R}^{m\times n}$ has unit ℓ_2 norm columns and coherence $\mu(A)$. Let $x \in \chi_n(k)$ and let $y = Ax$. If*

$$k < \frac{\mu(A)^{-1} + 1}{2}$$

then x is the unique solution to the ℓ_1 decoder, namely $\hat{x} = x$. Therefore, every k-sparse vector is exactly recovered by the ℓ_1 decoder.

The following exercises are the proof of Thm. 4.3.2. The goal to is to establish that the coherence bound on k is enough to guarantee that A satisfies $NSP(k)$. For these exercises, let $T \subset \{1,\dots,n\}$ with $|T| = k$ and let $v \in \text{null}(A) - \{0\}$.

Exercise 4.10. Use the fact that $0 = Av = \sum_{j=1}^n v_j A_j$ to observe that $0 = \left\langle A_i, \sum_{j=1}^n v_j A_j \right\rangle$ for any $i \in T$. Use this to establish

$$|v_i| \le \sum_{j\neq i, j\in T} |v_j|\,|\langle A_i, A_j\rangle| + \sum_{j\in T^c} |v_j|\,|\langle A_i, A_j\rangle|.$$

Exercise 4.11. Use the previous result to show that

$$\|v_T\|_1 \le (k-1)\mu(A)\|v_T\|_1 + k\mu(A)\|v_{T^c}\|_1.$$

Exercise 4.12. Now establish $NSP(k)$ to complete the proof.

4.4. **Summary.** We set out to "solve" an underdetermined system of equations $y = Ax$ where we want the sparsest solution. We also assume that we measured the sparsest solution. The spark and coherence of A are conditions that establish how well A encodes x in the sense that we will be able to distinguish x from the infinitely many other vectors z that will satisfy $y = Ax = Az$. When $\|x\|_0 = k$, the condition

$$k < \frac{\mu(A)^{-1} + 1}{2}$$

tells us that x is the unique sparsest vector whose measurements will be y, and it simultaneously tells us that we can recover (decode) x from y and A by

- the ℓ_0 decoder;
- the OMP decoder;
- the ℓ_1 decoder;
- the Thresholding decoder if the nonzeros of x are essentially all the same magnitude.

This is actually amazing in the sense that this ill posed problem can not only be solved, it can be solved in several different ways and with a reasonable computational cost.

However, we have to interpret our results correctly. The following exercise will put a temporary damper on our party.

Exercise 4.13. Suppose you have a fantastic encoder $A \in \mathbb{R}^{m\times n}$ with $\mu(A) = 1/\sqrt{m}$. Describe the relationship between k and m required for A to satisfy the condition

$$k < \frac{\mu(A)^{-1} + 1}{2}.$$

Is this really what we want? What is the relationship between k and m that we actually desire.

4.5. **Additional Exercises.**

Exercise 4.14. Using your previous SVD least squares code, implement Alg. 1. Create some random problems and find values of (k, m, n) that returns the correct solution. What can you say about the relationship between k and m? Remember, we assume A has unit ℓ_2 norm columns, so after creating a Gaussian matrix with $\texttt{randn}(m, n)$ you need to normalize the columns. Also, this won't work well if n is very small. Try $n = 256$ for starters. For the PrincipalSupport$_k$ step, use help to understand the sort function. To create the output vector $\hat{x}$ with zeros everywhere but on the set T, simply use $\hat{x} = \texttt{zeros}(n, 1)$ and then $\hat{x}(T) = \texttt{svdsolve}(y, T)$.

Exercise 4.15. Repeat Exercise 4.14 for OMP. (To ensure your code is actually working, you might want to start with smaller problems for OMP. Be sure to go back to the case where $n = 256$ and compare how long it takes to solve a problem with OMP and Thresholding.

There are several conditions which imply the null space property. One such condition used later in a different proof is established in the following exercise.

Exercise 4.16. Prove the following for a matrix $A \in \mathbb{R}^{m\times n}$. If for every $v \in \text{null}(A) - \{0\}$ and every index set $T \in \{1, \ldots, n\}$ with $|T| = k$,

$$\|v_T\|_1 < \frac{1}{2}\|v\|_1$$

then A satisfies $NSP(k)$. (The converse is also true; feel free to prove it.)

Here you prove the necessity of the null space property.

Exercise 4.17. Suppose $A \in \mathbb{R}^{m\times n}$ satisfies that for any $x \in \chi_n(k)$ and $y = Ax \in \mathbb{R}^m$, x is the unique output of the ℓ_1 decoder. Prove that A satisfies the null space property of order k. (*Hint*: Choose an arbitrary index set, an arbitrary vector from the null space, and write the vector as the sum of the vectors supported on and off your index set.)