ECE 269 Mini Project 1: Finding Sparse Solutions via Orthogonal Matching Pursuit (OMP)

Daniel Chen
Dept. of Electrical and Computer Engineering
University of California San Diego
La Jolla, CA 92093
Email: dychen@ucsd.edu

I. Introduction

The goal of this project is to implement a Orthogonal Matching Pursuit algorithm in order solve for the most sparse linear combination of a signal using a dictionary of redundant atoms. We demonstrate that this has applications in signal reconstruction across image and audio. The applications of such greedy algorithms are important in the field of digital signal processing. In this project, we will show how OMP performs across various levels of complexity including the number of measurements, sparsity, and how it performs great under noiseless environments, but may be limited under noisy environments.

II. Problem Formulation

In class, we learned about how signals can be reconstructed as a linear combination of basis vectors. This holds true, but the key that makes this interesting in real life is that we do not know the dimension our signal can be represented as. I like to think of this notion as if we had a very simple signal that only has two coefficients on its basis. So the signal can be represented in R^2. However, if that signal is in a much higher dimension such as in R^{50}, how can you efficiently tell how many vectors can least make up our signal if our signal is in such a high dimension. The key here is unique representation under a basis. We don't need to find the exact standard basis it was made, as long as we find some two linearly independent basis from even a random dictionary of basis vectors (even if they have repeats aka dependent aka non trivial null space) we can still represent and therefore recover our signal in that original 2 vectors worth of information.

Now we describe Orthogonal Matching Pursuit (OMP) as presented in *Greed is Good: Algorithmic Results for Sparse Approximation* by Tropp [1]. In their paper, they describes the history of greedy and basic pursuit algorithms. It explains their terminology of using a large dependent dictionary with atoms, which can be thought of as our basis that we want to minimally select from to reconstruct our signal.

III. 1. Performance Metric

In order for OMP algorithm to work, we need to have a performance metric that is similar to a loss function we use in least squares and any other optimization problem. However, here we define a normalized error for OMP that takes the difference of the estimate and the actual x from OMP and averages over 2000 Monte Carlo runs to minimize deviation

$$\text{Normalized Error} = \frac{\|x - \hat{x}\|_2}{\|x\|_2}. \tag{1}$$

IV. 2. Experimental Setup

We consider the standard compressed sensing measurement model

$$y = Ax + n \tag{2}$$

where

- $y \in \mathbb{R}^M$ is the measurement vector
- $A \in \mathbb{R}^{M \times N}$ is the measurement matrix
- $x \in \mathbb{R}^N$ is the unknown sparse signal to be recovered, with support cardinality s
- $n \in \mathbb{R}^M$ is the additive noise from the measurement

A. Greed is good

Approximating a signal with the best linear combination of elements from redundant is a method of sparse approximation/highly nonlinear approximation. Given a redundant dictionary, we want to design fast algorithm to calculate nearly optimal sparse representation of arbitrary input signal. OMP is a iterative greedy algorithm that selects at each step the dictionary element that best correlated with the residual part of the signal to produce a new approximant by projecting signal onto the dictionary element that already selected. The Matching Pursuit (MP) extends from trivial greedy algorithm. We have a fixed redundant dictionary of elementary signals/atoms where the signal is represented by a linear combination of atoms in order to identify the representation with sparest/least atoms.

A_{opt} - columns are atoms of optimal representation

$$A_{\text{opt}}^{+} = A_{\text{opt}}^{*} A_{\text{opt}}^{-1} A_{\text{opt}}$$

$\mu = \max$ absolute inner product of two distinct atoms (how much alike)

theorem A:

$$\max_{\psi} \ \|\phi_{\text{opt}}^{+} \psi\| < 1$$

theorem B: ERC condition holds for evert=y signal with an m-term representation provided that

$$m < 0.5(\mu^{-1} + 1) \quad \text{or} \quad \mu_1(m-1) + \mu_1(m) < 1$$

theorem B -

$$\|s - a_m\|_2 \leq \sqrt{1+6m}\|s - a_{\text{opt}}\|_2$$

a_{opt} = optimal m-term approximant of input signal

Definitions:
Hermitian Inner Product

$$\langle s, x \rangle = x^* s$$

Euclidean norm

$$\|s\|_2 = \sqrt{(s,s)}$$

Dictionary is a finite collection D of unit-norm vectors that span whole space

Atoms are members of dictionary ϕ_ω, ω is drawn from an index set Ω

$$D = \{\phi_\omega : \omega \in \Omega\}$$

$$N = \text{size of dictionary}, \quad N = |D| = |\Omega|$$

Representation is a linear combination of atoms that constructs the signal. All coeffcients in representation are non-zero so m-term representation uses m-atoms

$$\min_{|\Lambda|=m} \min_{\{b_\lambda\}} \left\| s - \sum_{\lambda \in \Lambda} b_\lambda \varphi_\lambda \right\|_2$$

Now to describe how the OMP algorithm works, we first generate a dictionary of shape A. We input that into our algorithm, we take the dimensions which are included and will take the initial residual and set it to create a set of lambdas, which are where our atoms will be, and now we are going to try to solve for those atoms. I will be using the following notation in my code which was an aiming to match the notation from the paper and the assignment. I think I found a good middle ground that makes sense to me and our class conventions:

- $r^{(k)}$ - residuals are the yet to be explained measurements
- $\Lambda^{(k)}$ - is the support set containing selected atom idexes,
- $x^{(k)}$ - is the estimate of the sparse signal.

$$r^{(0)} = y, \tag{3}$$
$$\Lambda^{(0)} = \emptyset, \tag{4}$$
$$x^{(0)} = 0. \tag{5}$$

Next we need to iterate through to find all the atoms. Knowing when to stop is extremely crucial and I will be discussing this further when I talk about the noiseless and noisy cases with and without sparsity knowledge, but essentially we have a stopping condition when we feel confident our error has reached a certain threshold or success criteria.

We next want to find the best atom, which is the key to a greedy algorithm. We do this by calculating the correlation between our residual $r^{(k)}$ and the all the atoms A. This can be done with taking the absolute value of their multiplication:

$$correlation_j = |\langle a_j, r^{(k)} \rangle|, \quad j = 1, \ldots, N, \tag{6}$$

We find the largest correlation and add that to our atoms dictionary. We then calculate the least squares of the selected basis vector along with the y and append that to our list of λ. We then use least squares to do a orthogonal projection mapping:

$$x_k = \arg\min_z \|y - A_k z\|_2^2. \tag{7}$$

The solution can solved with what looks a lot like our projection matrix we learned in class!

$$x_k = (A_k^T A_k)^{-1} \cdot A_k^T \cdot y \tag{8}$$

Next we can update our residual with our calculated least squares:

$$r^k = y - A_k x_k. \tag{9}$$

If we have noise or not, we check the L_2 norm against either the tolerance, which is when we say it is close to zero, or we check against the noise threshold, which is typically a bit higher since we have noise. When we have noise, instead of waiting for exact matches, where we approximate a match when the normalized error is below a very small tolerance, well below for the range we are sampling from to be by chance, I went with a tolerance for a difference of 0 to be 10^{-6} for noiseless. However, when dealing with noisy conditions, we can never get to zero zero. This means we must have a success threshold we can define that tells us when we can stop. We also can use two ways of stopping. In part 4a, we test when we know the sparsity we are aiming for. So that means we can use a the known sparsity as our stopping rule. The point of the greedy algorithm is that it selects the best mathching that is also orthogonal to all the previous atoms. This means that if it is going to be able to find the sparse vectors of a signal, it needs that many iterations. We use 10^{-3} as a our success criteria. To test, we can use multiple different levels of error to see how it performs. This can be modeled by the standard deviation σ I choose 0.001 and 0.1 originally for some diversity, but 0.001 was getting too close to the noiseless so I ultimately chose to show 0.005 versus 0.1 which better shows the drop off in OMP success when introduced to levels of noise. We can also test the same levels of error for when we do not know the sparsity. We use the fact that we know our noise n. Since from our model we had:

$$y = Ax + n$$
$$r = y - Ax = n$$

The best solution we can get still has a residual $||r^{(k)}||_2$ of $||n||_2$. This means that with the known noise, we can now

try stopping when we have hit a residual of than known error. This can arguably happen earlier or later than when we have our known sparsity. I think this might be a more realistic scenario in real life when we know less about the signal's sparsest construction but we do know about the noise of our measurement tools. However, I can also see a real life scenario where we may known sparsity. In fact, if we were to say have an audio signal of known sparsity, that would be a great example of known sparsity with noise!

V. Results

Now that we have the OMP algorithm, we can use it to perform our experiments under noiseless and noisy conditions. Starting off with the noiseless, this is easy by not defining any noise n to add to our y. We can try reconstructing the signal of differing sparsities s with different number of measurements m for different size signal vectors N. Let us start with 20, 50, and 100 n size vectors. I wrote a function that originally skips every other sparsity s but that maybe the transition more jagged. To show the full picture, I let my monte carlo runs run over night and I was able to get very smooth phase transitions for all n test cases including for $n = 100$.

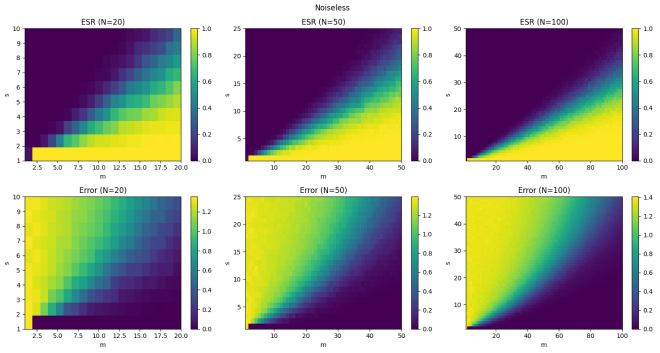

Fig. 1. Noiseless Summary

We need to test for each of these three under noiseless conditions but only when we have measurements less than n. That means we should test trying to recover from 1-N. Then we try to recover different level of sparsity, the more sparse, the higher chance we recover so we can see in the results that we were able to recover a triangular area of signals when our measurements is about 4 times the size of our sparsity of that signal. We notice that having the correct ratio of sparsity to signal size along with the right amount of measurements is crucial. Ideally, we could test across more N to show that it become harder and harder to recover signals when they are in a larger search space. We can see that visualized by the yellow triangle in N = 100 under noiseless. This is a sharp ESR phase transition for the phase transition but the error phase transition is more gradual. I originally was skipping over any test case where m is less than s because it should not be possible to recover a signal of sparsity s with less than that amount of measurements. However, for completeness I also graphed them and like I thought, most of their ESR is near 0 which makes sense because you should not be able to reconstruct a signal when you have less samples than the sparsity it has.

Now let us add some normal noise and test when we know the sparsity s, which means we know when we should stop which is when we reach s iterations:

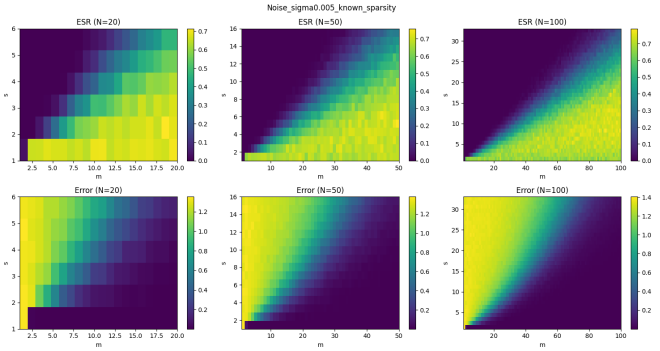

Fig. 2. Noise sigma 0.005 known sparsity summary

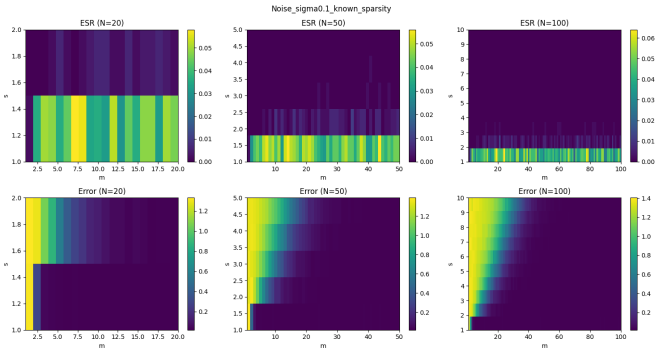

Fig. 3. Noise sigma 0.1 known sparsity summary

This is sampled from a normal distribution and highlights the difficulty of the OMP algorithm in handling noisy signals. It struggles with signals that have noise with more than sigma = 0.1, not being able to recover any under this m¡n condition. I assume if we sample more measurements, we might be able to do better. However, with a smaller sigma of 0.005, we have less noise and can almost recover some signals under known and unknown sparsity. It seems like knowing the sparsity helped with solving time as we knew when we could stop, our stop condition would just be the sparsity, we know when to give up. However, the we get to a probability of 0.7 but not able to really recover any signals. I was able to recover many signals when I used a sigma of 0.001 but that seems so negligible amount of noise, it was not a fair test. This noisy test is to show the limitations of how OMP struggles with noise.

However, with the known and unknown number of sparsity, the error graphs and different, we are closer to solving with the known sparsity size for the same given other variables.

Now to show the success of OMP, we can try applying this to some compressed images and audio. depending on the

number of measurements or samples and the sparsity, we can see how it performs. For the same given audio, it would have the same sparsity but if we measure less of it, we have lower m and lower chance of recovery.

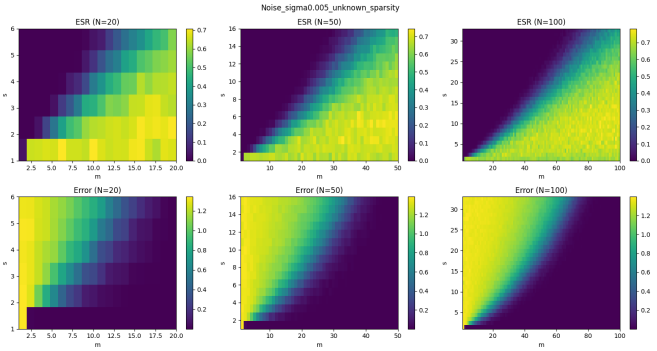

Fig. 4. Noise sigma 0.005 unknown sparsity summary

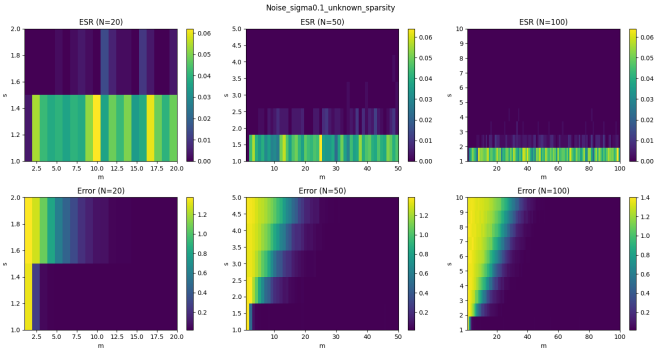

Fig. 5. Noise sigma 0.1 unknown sparsity summary

The first thing I noticed after running the Monte Carlo runs unknown sparsity is that it struggled a lot more. It is difficult to tell whether it was stopping earlier or later, but either way it will be off from the tru sparisty by some distribution about it. I was orignally confused on how we got the stopping condition because in real life how do you calculate that, but I realized it should be near your noise. so that is why I tested with 0.001, 0.05 and 0.1. For the 0.001 case, I think it makes sense that we can recover, because the noise is just about our "known" noise of 0.01. With these three different standard deviation levels of noise compared to our stopping condition, we can see it struggles greatly when the noise is greater than stopping condition. This makes a lot of sense because it is like trying to stabilize something that is jittering more than your capability of stabalizing. I'm thinking of it like taking a picture on a bumpy road, where it's more bumpy than you can correct for it. We can evidently see this in the poor performance of both when $\sigma = 0.1$ or $\sigma = 0.005$. However, we can see that OMP does a pretty decent job for both known and unknown sparisty when $\sigma = 0.001$:

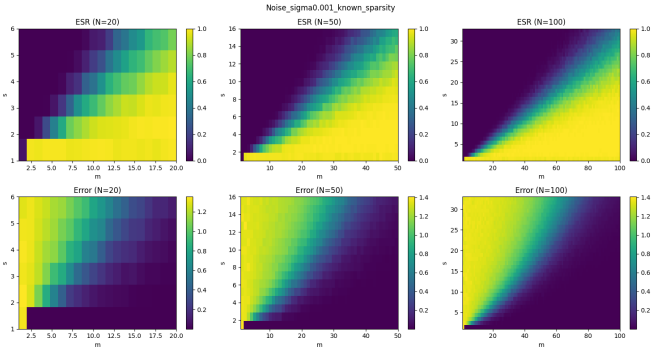

Fig. 6. Noise sigma 0.001 unknown sparsity summary

Notice I also tested a smaller range of sparsity simply because the higher the noise, the more impossible it becomes to recover the signal even as M approachs n. I wonder for M¿n, we can probably recover a good amount but I'm sure it gets bounded at some point if our noise is too high. I think I made reasonable cut offs for sparsities tested because no successful reconstructions are cut off, especially for the high 0.1 noise.

A. *Image Compression*

Next let us try actually applying this to a compressed image. We start with 3 compressed files of unknown dimensions, so it's near impossible to decipher the compressed image because you would not be given the dimensions from the raw data alone.

Let's formalize some variables, $A \in \mathbb{R}^{M \times N}$ is a random matrix so we can use that as our dictionary. $X \in \mathbb{R}^N$ is our sparse image in vector form. we can later transform it to a 2D representation with the known dimension information. This gives us a relationship of:

$$Y_i = A_i X, \quad i = 1, 2, 3$$

Using our OMP algorithm in the setting that we do NOT know our sparsity, we can guess different error n's for our stopping condition. I tried a few methodical ways but I was not able to get a good condition dependent on the measurement size. I tried initially just tried all the columns of A. But I was realizing since it's a redundant dictionary, there are fewer rows than columns. so I went with the rows. This was I think still too many, it was trying to overfit and it missed the actual good atoms since it might be trying to construct with too many axioms and you actually end up with a worse reconstruction than the most sparse. I tried with fewer by just manually setting them to lower ones. Since the A's all have increasing size with more fidelity.

After playing around with my OMP algorithm parameters, I noticed that the message decoding benefited greatly from an early stop, I looked at the different shapes of the images which corresponds to the number of measurements M.

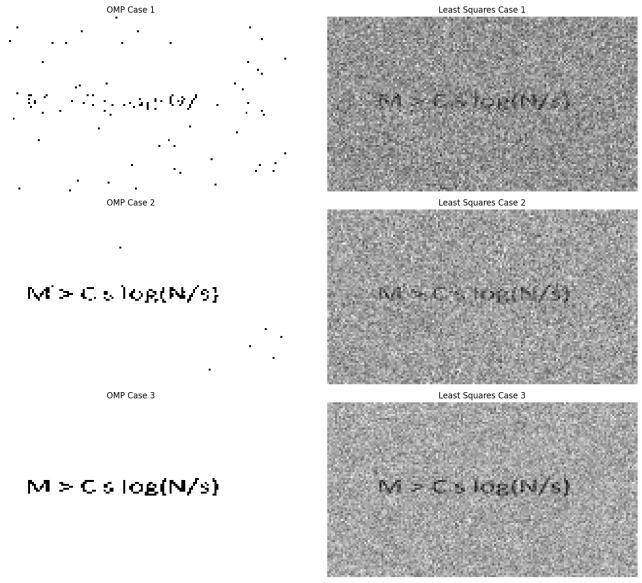

Fig. 7. Secret Message!

They increase from case 1 to 2 to 3 from 960, 1440, 2880 respectively. The case 3 was the least compressed so it had the most information with the dictionary of row 2880. However, this must have led to some overfitting to incorrect basis or discarding of our correct basis because it was running for too many cycles. I played around until I found really good results with 500.

c) Which corrupted image gave you the best result for OMP? Can you explain why?

I got clear results with case 3 which had the most measurements and somewhat good results with case 2 which almost 50% less measurements. However it was not possible for me to recover the signal from case 1, there was a lot of noise seen as the specks. These are likely due to incorrect atoms being picked. It reminds me of ECE 172 when we did tried Discrete Fourier transform (DFT) reconstructions of images. I believe DFT is essentially reconstructing signals with the right coefficients of a basis which I thought was an interesting connection. You can recover a lot of the image from a few frequencies and higher and lower frequencies keep different information since higher frequency vectors preserve the rapidly changing pixels, it preserves the details while while lower frequencies get the structure and constrast. Here we were able to reconstruct a black and white image with clear borders meaning we likely captured both high and low frequency atoms. I'm not 100% sure if this is a valid way to look at OMP signal reconstruction but I think it might have some analogs to DFT.

d) (For Fun) Can you make an (educated) guess about the meaning of this message?

I think since this is an inequality of M ¿ Cs log(N/s) it has a lot of variables that we have seen. I'm guessing it represent a minimum number of measurements M you need to recover a signal with a sparsity of s in a signal of size N or field of size N. I think the logrithmic formula shows that the number of measurements you need increases but logrithmically as you have a larger N to s ratio or level of sparsity meaning how many s vectors you need to represent your signal in N.

Decode a Compressed Audio Signal

Next we work on reconstructing an unknown audio signal x with frame per second (fps) of 7350 containing a message has a sparse representation over a special basis D:

$$x = Ds$$

We have that the sparsity s is a 100-sparse vector of size 15980.

This is a great example of when we know the sparsity so we can use our OMP in the configuration that assumes noise and has max iterations of 100. We still compare our L2 norm of our residual to the tolerance but we stop when we reach 100 iterations.

The audio signal x was compressed with a random matrix A that we can use as our dictionary to produce compressed signal y:

$$y = Ax = ADs$$

This matrix D is normalized multiplied by our random matrix A which are all known.

(a) Can you guess the message by simply playing the compressed signal?
With the audio signal, I was not able to decipher, the compressed audio signal just sounded like noise. This is to be expected because multiplying by a random matrix is like having random linear projections of our original signal.

(b) For all values of $k \in K = \{10, 50, 100, 200, 300, 1000, 2000, 3000\}$, select the first k elements of y

$$y_k = [y]_{1:k}$$

We want to use OMP in our max iterations of $s = 100$ to recover s from y_k for all $k \in K$ and play

$$x = Ds$$

The message is "I love linear algebra" It started sounding clearer than the least squares version and increases with the number of measurements K we use from 10 being noise to 3000 being crystal clear

(c) What is the minimum number of measurements (call it $K_{\min}$) that is enough to understand the message from the reconstructed audio signal (based on your perception)? Can you explain what parameters of the problem this number depends on? What is the relation of this number to the size and sparsity of s? Can you comment on what advantages more measurements (over $K_{\min}$) offer?
The number of measurements we use represented by K is the key here, as we increase in K, the OMP reconstruction improved significantly, the first K = 10 still sounded like noise but as we incrased in K, it became less and less noisy until it was crystal clear with no background static in K3000 compared to the

k3000 under least squares which still had static. This minimum K value needed depends on the sparsity of our vector as well as the search space N. It is also affected by maybe how distorted the signal gets after AD but I'm not sure how to best quantify that. Lastly the amount of noise n also affects the minimum value of k we need to reconstruct the signal. If we increase the measurements we use, the better our audio gets. the minimum measurements seemed to be K = 50 with static, but it started sounding better and better than the least square vesrsion, where at k3000, we can hear it crystal clear.

VI. Conclusion

In conclusion, I learned a lot about signal processing through this project. It was a lot of fun going from reading a very dense paper to realizing you learned a lot of the basis (pun intended) and then applying the algorithm to test how different number of samples m against how sparse s the vector is. I wonder if we can empirclally show the secret message $M > C \cdot s \cdot log(N/s)$ is true from the noise and noiseless plots. I'm not sure how noise applies to it but since it is an inequality it would make sense that there is a minimum. It kind of reminds me of other minimum rules in signal processing like Nyquist sampling theorem where you need strictly greater than 2 times the highest frequency in your signal to reconstruct without aliasing.

References

[1] J. A. Tropp, "Greed is Good: Algorithmic Results for Sparse Approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.