5-2010

Compressive Sensing

Yue Mao
Clemson University, yuem@clemson.edu

Compressive Sensing

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Sciences
Mathematical Sciences

---

by
Yue Mao
May 2010

Accepted by:
Dr. Shuhong Gao, Committee Chair
Dr. Taufiquar R Khan
Dr. Xiaoqian Sun

Abstract

Compressive sensing is a novel paradigm for acquiring signals and has a wide range of applications. The basic assumption is that one can recover a sparse or compressible signal from far fewer measurements than traditional methods. The difficulty lies in the construction of efficient recovery algorithms. In this thesis, we review two main approaches for solving the sparse recovery problem in compressive sensing: l_1-minimization methods and greedy methods. Our contribution is that we look at compressive sensing from a different point of view by connecting it with sparse interpolation. We introduce a new algorithm for compressive sensing called generalized eigenvalues (GE). GE uses the first m consecutive rows of discrete Fourier matrix as its measurement matrix. GE solves for the support of a sparse signal directly by considering generalized eigenvalues of Hankel systems. Under Fourier measurements, we compare GE with iterated hard thresholding (IHT) that is one of the state-of-the-art greedy algorithms. Our experiment shows that GE has a much higher probability of success than IHT when the number of measurements is small while GE is a bit more sensitive for signals with clustered entries. To address this problem, we give some observations from the experiment that suggests GE can be potentially improved by taking adaptive Fourier measurements. In addition, most greedy algorithms assume that the sparsity k is known. As sparsity depends on the signal and we may not be able to know the sparsity unless we have some prior information about the signal. However, GE doesn't need any prior information on the sparsity and can determine the sparsity by simply computing the rank of the Hankel system.

Key words: compressive sensing, sparse recovery, generalized eigenvalues, sparse interpolation, Hankel system, Fourier measurements

Dedication

I dedicate this dissertation to all my loved ones, those with me today and those who have passed on.

Acknowledgments

I thank my advisor Shuhong Gao. His guidance, support, patience, and humor have made this thesis possible. I also thank Frank Volny IV for his helpful discussion throughout the development of this thesis and his comments and suggestions. I appreciate everyone at the Department of Mathematical Sciences in Clemson University for giving me an environment where research is fun. Mingfu Zhu been exceptionally helpful with my graduate life, thank you. I am grateful to my family for their love, support, and inspiration. Finally, I appreciate Xiaoqian Sun and Taufiquar R Khan for their comments as my committee members.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Conventional approaches to sampling signals follow Shannon's theorem: in order to construct a signal without error, the sampling rate must be at least the Nyquist rate that is defined as twice the bandwidth of the signal. As we know, many signals are sparse or compressible under some orthonormal basis. For example, smooth images are compressible in the Fourier basis, while piecewise smooth images are compressible in the wavelets basis. In practice, one samples a signal at the Nyquist rate and compress it via Fourier or wavelets transforms. This is done in JPEG and JPEG2000 [21].

However, the process of acquiring all the data seems wasteful since most of the data will be thrown away later. This raises an interesting question: is it possible to acquire signals in a compressed form? If this is possible, then we don't need to spend so much effort acquiring all the data. Recent work in compressive sensing (also referred as compressive sampling or compressed sampling or CS) theory indicates the answer is positive.

Instead of taking samples at the Nyquist rate, compressive sensing uses a much smaller sampling rate. Pioneering contributions in this area are given by Candès, Romberg, Tao [6] [7] [3] and Donoho [13]. They use l_1-minimization methods. Another approach is via greedy methods. Compressive sampling matching pursuit (CoSaMP) [24], subspace pursuit (SP) [10], and iterated hard thresholding (IHT) [2] are three of the state-of-the-art greedy algorithms. The phase transition framework of [1] indicates IHT is the best one among them because IHT has the lowest computational cost and requires fewer measurements than CoSaMP and SP. Inspired by [14], we propose a new algorithm called "generalized eigenvalues (GE)." GE takes Fourier measurement and considers

generalized eigenvalues of Hankel system.

1.1 Basic Problem

We regard a signal as a a real-valued column vector in $\mathbb{R}^{N\times 1}$. Through out this thesis, for simplicity, we denote $\mathbb{R}^{N\times 1}$ ($\mathbb{C}^{N\times 1}$) by $\mathbb{R}^N$ ($\mathbb{C}^N$).

Definition 1.1.1 *(Support Set, Sparse Signal, Sparsity) Given a signal $x \in \mathbb{R}^N$. The **support set** of x are the indices of nonzero entries in x, say $T \subset \{1, 2, \ldots, N\}$. A signal is **sparse** if it is only supported on a small set. i.e., $k = |T| \ll N$. Then above k is called the **sparsity** of x and we say x is a k-sparse signal.*

Definition 1.1.2 *(Measurements) Suppose we want to sense/sample a signal $x \in \mathbb{R}^N$. Let $a_1, a_2, \ldots, a_m \in \mathbb{R}^N$, then **measurements** are linear combinations of the entries of x:*

$$y_i =< a_i, x >, \qquad i = 1, 2, \ldots, m,$$

where $\langle \cdot, \cdot \rangle$ is the inner product and m is the number of measurements. We prefer using convenient matrix notation. Define $A \in \mathbb{R}^{m\times N}$ as

$$A = \begin{pmatrix} a_1^t \\ a_2^t \\ \vdots \\ a_m^t \end{pmatrix},$$

*called **measurement matrix** (also referred as **sensing matrix**) where a_i^t is the conjugate of a_i. Also define $y \in \mathbb{R}^m$ as*

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix},$$

called the measurement vector. Then the sampling/sensing process is simply

$$y = Ax. \tag{1.1}$$

Definition 1.1.3 *(Undersampling) If the the number of measurements is less than the length of the signal, i.e., $m < N$, then above sampling process is called* ***undersampling****.*

Consider recovering x from an undersampling in (1.1). Note the number of measurements is less than the signal length, i.e., $m < N$, so (1.1) is under-determined. Then (1.1) would have infinitely many candidates. However, if we know that x is sparse, then we are only interested in the sparsest solution among these candidates. Hence, we actually want to solve the following minimization problem.

Problem 1.1.1 *(l_0-minimization) Given a matrix $A \in \mathbb{R}^{m\times N}$ and $y \in \mathbb{R}^m$, solve the minimization problem*

$$\min_{\tilde{x}\in\mathbb{R}^N} ||\tilde{x}||_0 \quad s.t. \quad A\tilde{x} = y, \tag{1.2}$$

where $||\cdot||_0$ counts the number of nonzero entries. Assume that $y = Ax$ where $x \in \mathbb{R}^N$ is sparse. The problem is for which A, $\tilde{x} = x$? And what's the recovery algorithm?

In practice, signals may not be exactly sparse. Most time we would deal with "compressible signals" which is defined as follows.

Definition 1.1.4 *(Compressible Signals, Power-law and Best k-sparse Approximation) A signal $x \in \mathbb{R}^N$ is* ***compressible*** *if its entries decay rapidly when sorted by magnitude,*

$$|x|_{(1)} \geq |x|_{(2)} \geq \cdots \geq |x|_{(N)}.$$

The i-th largest entry obeys the ***power-law*** *$|x|_{(i)} \leq C \cdot i^{-p}$, for $1 \leq i \leq N$ and some constant $C > 0$, where the parameter $p > 1$ controls the speed of decay. The larger p, the faster the decay. Let $x^{(k)} \in \mathbb{R}^N$ denote the vector which take the largest k entries of x with others being zeros. Then $x^{(k)}$ is the* ***best k-sparse approximation*** *of x. The k-sparse approximation $x^{(k)}$ is denoted by x_s in this thesis.*

By classical calculus,

$$||x - x_s||_1 = ||x - x^{(k)}||_1 \leq \sum_{i=k+1}^{N} |x|_{(i)} \leq \sum_{i=k+1}^{N} C \cdot i^{-1/p} \leq C' \cdot k^{1-p},$$

for some constant C'. Hence we can choose some proper sparsity k to control the approximation error.

Most natural signals are compressible if they are represented under certain orthogonal basis such as Fourier basis and wavelets basis. Those bases are also referred as sparse dictionaries. In this thesis, we shall not discuss how to find sparse dictionaries, instead refer the readers to the papers [16], [19] and [20].

Definition 1.1.5 *(Noisy Measurements) Due to limited precision of our measurements and other inevitable human or machine mistakes, there always exists errors, say $e \in \mathbb{R}^m$. Suppose we are measuring a signal $x \in \mathbb{R}^N$ (not necessarily sparse), we get a noisy measurement vector*

$$y = Ax + e. \tag{1.3}$$

In the following we modify (1.2) and define the sparse recovery problem in general.

Problem 1.1.2 *(Sparse Recovery from Noisy Measurement) Given a measurement matrix $A \in \mathbb{R}^{m\times N}$ and a noisy measurement vector $y = Ax + e \in \mathbb{R}^m$ in (1.3). Our goal is to recover a sparse approximation x_s of x. This can be proceeded as the following minimization problem:*

$$\min_{\tilde{x}\in\mathbb{R}^N} ||\tilde{x}||_0 \quad s.t. \quad ||A\tilde{x} - y||_2 < \epsilon \tag{1.4}$$

for some small positive $\epsilon \in \mathbb{R}$.

In next chapter, we will discuss the main ideas to solve above sparse recovery problem (1.4).

1.2 Main Ideas

Unfortunately, problem (1.2) and problem (1.4) are combinatorial programmings that are NP-hard [22]. However, by recent compressive sensing theory, it is shown that exact recovery is possible under certain condition. There are two main approaches to solve it. One is via l_1-minimization methods and the other is via greedy methods. And for both of them, the number of measurements m is nearly linear in the sparsity k,

$$m \approx k \log^{O(1)}(N). \tag{1.5}$$

Here we briefly describe the main ideas of l_1-minimization methods and greedy methods.

1.2.1 l_1-minimization Methods

Since minimizing l_0-norm is NP-hard, several authors have considered its convex relaxation [4], that is, to solve the following convex programming problem:

$$\min_{\tilde{x}} ||\tilde{x}||_1 \quad s.t. \quad ||A\tilde{x} - y||_2 < \epsilon. \tag{1.6}$$

A difficult problem is to know when the solutions to (1.4) and (1.6) coincide. For this purpose, Candès and Tao introduced "restricted isometry property (RIP)" [8]. It roughly says that the matrix A acts as an almost isometry on all sparse vectors of certain sparsity, i.e., the columns of A are almost orthonormal. This approach is slow but can be guaranteed to find the correct solution for a wide range of parameters. We will describe the "restricted isometry property" in detail in section 2.1.2.

1.2.2 Greedy Methods

Another approach is to use greedy methods. A greedy algorithm is any algorithm that follows the problem solving metaheuristic of making the locally optimal choice at each stage with the hope of finding the global optimum. Instead of finding a convex relaxation, greedy algorithms try to recover the supports iteratively.

There are three state-of-the-art greedy algorithms: compressive sampling matching pursuit (CoSaMP), subspace pursuit (SP) and iterated hard thresholding (IHT). These algorithms are closely connected.

The main idea of all three greedy algorithms is: during i-th iteration, they compute a temporary estimate $\hat{x}$ based on the information of previous iteration. Then they update the k-sparse estimate as $x^{[i]}$ as well as the corresponding support set $T^{[i]}$ by selecting largest k entries of $\hat{x}$ (hard thresholding). Their differences lie in how to apply the hard thresholding and how to determine the support set based on information of previous iteration. Section 2.2.2 presents more details.

1.3 Organization of the Thesis

This thesis is organized as follows. In Chapter 2, we briefly recall several key concepts in compressive sensing. We also describe main recovery approaches including l_1-minimization methods and greedy methods. In Chapter 3, we briefly discuss the applications and extensions of compressive sensing including imaging, error correction, and robust principle component analysis. In Chapter 4, we look at compressive sensing from a different point of view by connecting compressive sensing with sparse interpolation. We introduce an algorithm called generalized eigenvalues (GE). Then, we consider the case of Fourier measurements and give detailed experiments comparing GE with IHT. We also give some interesting observations from the experiments and suggest that GE can be potentially improved by take adaptive Fourier measurements.

Chapter 2

Main Approaches in Compressive Sensing

Compressive sensing has two steps. The first step is to sense a signal by taking a few measurements. If we write it using matrix notation, then it is

$$y = Ax,$$

where $y \in \mathbb{R}^m$ is the measurements, $A \in \mathbb{R}^{m \times N}$ is the measurement matrix and $x \in \mathbb{R}^N$ is a real-valued compressible signal. In compressive sensing, signals are undersampled ($m < N$). In other words, signals are already sampled in a compressed form. Hence, the the second step is to recover sparse approximations of the original signals. We can also view the first step as "encoding" and the second step as "decoding." Hence the main problems are:

- constructions of matrices which provide guarantees for exact recovery for sparse or compressible signals,
- constructions of efficient and stable recovery algorithms.

In this chapter we discuss "measurements" in the first section and "recovery algorithms" in the second section.

2.1 Measurements

2.1.1 RIP

Candès and Tao introduced an important property on a measurement matrix which ensures exact recovery, that is so called "restricted isometry property (RIP)." Roughly speaking, the matrix A having RIP are nearly orthonormal when operating on sparse vectors x. The columns with indices corresponding to the support of sparse vector x should be nearly orthogonal. Given the sparsity k, the number of measurements m, the dimension N for the input signal x, we can define RIP as follows.

Definition 2.1.1 *(restricted isometry property (RIP)) A matrix $A \in \mathbb{R}^{m \times N}$ satisfies RIP if for any k-sparse nonzero vector $x \in \mathbb{R}^N$, the following inequality holds,*

$$1 - \delta(k, m, N) \leq \frac{||Ax||_2^2}{||x||_2^2} \leq 1 + \delta(k, m, N), \tag{2.1}$$

where the constant $\delta(k, m, N) \in (0, 1)$ is called "restricted isometry constant (RIC)."

We have some intuitive explanations for the above definition.

- The smaller RIC, the better performance in compressive sensing. Consider two extreme cases. If $\delta(k, m, N) = 0$, then for any k columns of A, each column will be orthogonal to the other columns. It follows that every column is orthogonal to the other columns. For flat matrices ($m < N$), $\delta(k, m, N)$ is always nonzero. Hence, A must be an $N \times N$ orthonormal matrix. If $\delta(k, m, N) = 1$, then there exists a nonzero k-sparse vector x such that $Ax = 0$. We can write $x = x_1 - x_2$ where both x_1 and x_2 are non-negative. $Ax = 0$ implies $Ax_1 = Ax_2$. Note the sparsities of x_1 and x_2 are both less than k but we are not able to distinguish x_1 and x_2 for they have the same measurements 0. Hence A with $\delta(k, m, N) = 1$ can not recover all k-sparse signals and it is a bad matrix for sparse recovery.

- It is not hard to see that if m increases while we fix k and N, then $\delta(k, m, N)$ will decrease. Intuitively, it is because that we get more information when we take more measurements then we have more chance to recover the sparse signal. Similarly, if we k increases while we fix m and N, then $\delta(k, m, N)$ increases. The reason is that all k_1-sparse vectors are a subset of all k_2-sparse vectors for $k_1 < k_2$. Then by the definition of RIC, $\delta(k_1, m, N) < \delta(k_2, m, N)$.

- Suppose a signal x is supported on T, then $y = Ax = A|_T x|_T$, where $x|_T(A|_T)$ is a sub-vector(sub-matrix) of $x(A)$ restricted on indices T.

We shall see in section 2.2, for both l_1-minimization methods and greedy methods, RIP plays an extremely important role for exact recovery.

Although many papers use RIP, it is important to know that RIP is only a sufficient condition to ensure exact recovery but not necessary. Another big problem is that testing RIP is NP-hard. Random matrices (see next chapter) are shown theoretically to have RIPs with a high probability as long as we have enough measurements. So it is desired to find a property that is not only a necessary and sufficient condition but also testable. One example is the "Null Space Property (NSP)" in [11].

2.1.2 Measurement Matrices

A nature question is to ask what types of measurement matrices have good RIPs? So far, most of measurement matrices are constructed using a random processing such as Gaussian matrix, binary matrix, partial Fourier matrix and incoherent matrix pairs. They have good RIPs with a high probability as long as we have enough measurements. It turns out that the construction of random matrices have the advantages in speed, storage as well as the guarantee for exact recovery. But for implementation, it is still interesting to understand what can be done with purely deterministic sampling. In the following we will discuss these measurement matrices.

We assume our measurement matrix $A \in \mathbb{R}^{m \times N}$. It means the signal has dimension N and we take m measurements.

- **Gaussian matrix**. The entries of measurement matrix A are identically sampled from normal distribution with mean 0 and variance $1/m$.

- **Binary matrix**. The entries of measurement matrix A are identically sampled from symmetric Bernoulli distribution $P(A_{ij} = \pm 1/\sqrt{m}) = 1/2$.

- **Partial Fourier matrix**. First we define the discrete Fourier matrix F. Its entry $[F]_{ij}$ at position (i, j) is

$$[F]_{i,j} = e^{2\pi\sqrt{-1}(i-1)(j-1)/N}.$$

  The partial Fourier matrix A is just taking m rows of the discrete Fourier matrix F. Then we normalize the columns of A by multiplying $1/\sqrt{m}$. Explicitly, let $\{i_1, i_2, \ldots, i_m\}$ be a random

subset of $\{1, 2 \ldots, N\}$, then

$$[A]_{l,j} = [F]_{i_l,j} \quad \text{for } 1 \leq l \leq m.$$

Remember we assume the measurement matrix should have real entries. So essentially, we are using the submatrix of discrete Cosine matrix. However, it is a usually it is more convenient to consider above partial Fourier matrix with complex entries and it won't affect our problems.

- **Incoherent matrix pairs**. Define the coherence of the matrix pair (Φ, Ψ) as

$$\mu(\Phi, \Psi) = \sqrt{N} \max_{i,j} | \langle \phi_i, \psi_j \rangle |,$$

where ϕ_is are the column vectors of Φ and ψ_j^*s are the row vectors of Ψ. The matrix pair (Φ, Ψ) is said to be incoherent pair if $\mu(\Phi, \Psi)$ is small. Then $A = \Phi\Psi$ is our measurement matrix. Actually, Φ refers to global sensing matrix and Ψ refers to sparse dictionaries (orthonormal basis).

- **Deterministic matrix**. For example, [18] and [12].

2.2 Recovery Algorithms

So far, there are many algorithms to solve sparse recovery problems. The two main approaches are l_1-minimization methods and greedy methods. l_1-minimization methods relax the NP-Hard l_0-minimization problem to an l_1-minimization problem, which can be solved by linear programming. Although the stability is good, the speed of linear programming is slow as the dimension increases. An alternative approach to our sparse recovery problem is via greedy algorithms. Early greedy algorithms have the advantages in speed but they are bad in stability. Recent greedy algorithms has bridged the gap between early greedy methods and l_1-minimization methods. These greedy algorithms borrowed the idea of restricted isometry property (RIP) from l_1-minimization. They performed better in both speed and stability.

2.2.1 l_1-minimization Methods

As we mentioned in section 1.2, the solution to problem (1.4) and problem (1.6) coincide under some condition on the matrix A. An sufficient condition is exactly the restricted isometry property mentioned before.

Theorem 2.2.1 *(Main Result of l_1-minimization Methods [4]) Assume $x \in \mathbb{R}^N$ is a compressible signal and it has the best k-sparse approximation x_s. We are given an $m \times N$ measurement matrix A and a noisy measurement vector $y = Ax + e$. Suppose $\tilde{x}$ is the solution to the l_1-minimization problem (1.6). If the restricted isometry constant (RIC) of A satisfies $\delta(2k, m, N) < \sqrt{2} - 1$, then*

$$||\tilde{x} - x||_2 \leq C_0 \cdot ||x - x_s||_1/\sqrt{k} + C_1 \cdot \epsilon, \tag{2.2}$$

for some constants C_0 and C_1. i.e., problem (1.4) and problem (1.6) are equivalent. The constants C_0 and C_1 are typically small. With $\delta(2k, m, N) = 1/4$ for example, $C_0 \leq 5.5$ and $C_1 \leq 6$.

2.2.2 Greedy Methods

Three of the state-of-the-art greedy algorithms are compressive sampling matching pursuit (CoSaMP) [24], subspace pursuit (SP) [10] and iterated hard thresholding (IHT) [2]. Here we discuss the ideas of these algorithms.

Suppose that we are given a measurement matrix A and a noisy measurement vector $y = Ax + e$ and suppose x_s is the true best k-sparse approximation of x. Let T denote the support set of the true sparse vector x_s. Although the sparsity k should depend on the signal, here we assume it is given (prior information). Our goal is to recover $\tilde{x} = x_s$.

Notations. During i-th iteration,

- **Estimate:** the k-sparse estimate denoted by $x^{[i]}$.
- **Support Set:** the corresponding support set of $x^{[i]}$ is denoted by $T^{[i]}$.
- **Residue:** the residue is defined as $r^{[i]} = y - Ax^{[i]}$.

Stopping Criteria: $||r^{[i]}||_2 < \varepsilon$ for some small $\varepsilon > 0$.

Algorithm 2.2.2 *(CoSaMP)*

- ***Input:*** A, y, k
- ***Output:*** *A best k-sparse approximation* $\tilde{x}$ *of the target signal* x
- ***Initialization:***
  1. *Set* $T^{[0]} = \emptyset$
  2. *Set* $r^{[0]} = y$
- ***Iteration:*** *During iteration* i, ***do***
  1. $\hat{T}^{[i]} = T^{[i-1]} \bigcup \{2k \text{ indices of largest magnitude entries of } A^* r^{[i-1]}\}$
  2. $\hat{x} = A|_{\hat{T}^{[i]}}{}^{\dagger} y$
  3. $T^{[i]} = \{k \text{ indices of largest magnitude entries of } \hat{x}\}$
  4. $r^{[i]} = y - A|_{T^{[i]}} \hat{x}|_{T^{[i]}}$
  5. ***until*** $||r^{[i]}||_2 < \epsilon$ ***then***
  6. ***return*** $\tilde{x}$ *defined by* $\tilde{x}|_{\{1,2,\ldots,N\}-T^{[i]}} = 0$ *and* $\tilde{x}|_{T^{[i]}} = \hat{x}|_{T^{[i]}}$

Algorithm 2.2.3 *(SP)*

- ***Input:*** A, y, k
- ***Output:*** *A best k-sparse approximation* $\tilde{x}$ *of the target signal* x
- ***Initialization:***
  1. *Set* $T^{[0]} = \emptyset$
  2. *Set* $r^{[0]} = y - A|_{T_{[0]}} A|^{\dagger}_{T_{[0]}} y$
- ***Iteration:*** *During iteration* i, ***do***
  1. $\hat{T}^{[i]} = T^{[i-1]} \bigcup \{k \text{ indices of largest magnitude entries of } A^* r^{[i-1]}\}$
  2. $\hat{x} = A|_{\hat{T}^{[i]}}{}^{\dagger} y$
  3. $T^{[i]} = \{k \text{ indices of largest magnitude entries of } \hat{x}\}$
  4. $r^{[i]} = y - A|_{T^{[i]}} A|_{T^{[i]}}{}^{\dagger} y$
  5. ***until*** $||r^{[i]}||_2 < \epsilon$ ***then***

6. ***return*** $\tilde{x}$ *defined by* $\tilde{x}|_{\{1,2,\ldots,N\}-T^{[i]}} = 0$ *and* $\tilde{x}|_{T^{[i]}} = A^{\dagger}_{T^{[i]}} y$

Algorithm 2.2.4 *(IHT)*

- ***Input:*** A, y, k
- ***Output:*** *A best* k*-sparse approximation* $\tilde{x}$ *of the target signal* x
- ***Initialization:***
  1. *Set* $x^{[0]} = 0$
  2. *Set* $T^{[0]} = \emptyset$
  3. *Set* $r^{[0]} = y$
- ***Iteration:*** *During iteration* i, ***do***
  1. $x^{[i]} = x^{[i-1]}|_{T^{[i-1]}} + wA^* r^{[i-1]}$
  2. $T^{[i]} = \{k$ *indices of largest magnitude entries of* $x^{[i]}\}$
  3. $r^{[i]} = y - A|_{T^{[i]}} x^{[i]}|_{T^{[i]}}$
  4. ***until*** $||r^{[i]}||_2 < \epsilon$ ***then***
  5. ***return*** $\tilde{x}$ *defined by* $\tilde{x}|_{\{1,2,\ldots,N\}-T^{[i]}} = 0$ *and* $\tilde{x}|_{T^{[i]}} = x^{[i]}|_{T^{[i]}}$

Key idea of CoSaMP, SP and IHT: During i-th iteration, three greedy algorithms compute a temporary estimate $\hat{x}$ based on the information of previous iteration. Then they update the k-sparse estimate as $x^{[i]}$ as well as the corresponding support set $T^{[i]}$ by selecting largest k entries of $\hat{x}$ (hard thresholding). The following Table is a comparison of CoSaMP, SP and IHT.

From Table 2.2.2, CoSaMP and SP are almost the same. The only difference is that, in step 1, CoSaMP adds $2k$ indices to previous support set while SP adds k indices. Note in step 2, for both of CoSaMP and SP, we solve a least square problem based on the temporary support set $\hat{T}$. Hence they both need to compute the pseudoinverse of $A|_{\hat{T}}$ that is approximately $m \times k$ dimensional. IHT is a bit different. IHT doesn't compute any pseudoinverse. Hence IHT is computationally more efficient. But essentially, all of CoSaMP, SP and IHT use the information of $A^* r^{[i-1]}$ to get the temporary estimate $\hat{x}$ then apply hard thresholding. In this sense, all of them are deeply connected.

Since SP is almost the same as CoSaMP, in the following we only review the main results of CoSaMP and IHT.

Step 1	$T^{[i-1]} \rightarrow$ temporary $\hat{T}$
CoSaMP	Add $2k$ indices of largest entries of $A^* r^{[i-1]}$
SP	Add k indices of largest entries of $A^* r^{[i-1]}$
IHT	/
Step 2	Compute temporary $\hat{x}$
CoSaMP	Least square solution based on $\hat{T}$
SP	Least square solution based on $\hat{T}$
IHT	Add $wA^* r^{[i-1]}$ to $x^{[i-1]}$
Step 3	Hard Thresholding
CoSaMP	Select largest k entries of $\hat{x}$, update to $x^{[i]}$, update to $T^{[i]}$
SP	Select largest k entries of $\hat{x}$, update to $x^{[i]}$, update to $T^{[i]}$
IHT	Select largest k entries of $\hat{x}$, update to $x^{[i]}$, update to $T^{[i]}$

Table 2.1: Comparison of CoSaMP, SP, IHT During i-th Iteration.

Theorem 2.2.5 *(Main result of CoSaMP [2]) Given a noisy measurement vector $y = Ax + e$, where x is compressible and it has the best k-sparse approximation x_s. Let $x^{[i]}$ be the estimate during i-th iteration for CoSaMP. If RIC satisfies $\delta(4k, m, N) < 0.1$, then*

$$||x - x^{[i]}||_2 < 2^{-i}||x_s||_2 + 20\epsilon_s, \tag{2.3}$$

where $\epsilon_s = ||x - x_s||_2 + \frac{1}{\sqrt{k}}||x - x_s||_1 + ||e||_2$.

Theorem 2.2.6 *(Main result of IHT [2]) Given a noisy measurement $y = Ax + e$, where x is compressible and it has the best k-sparse approximation x_s. Let $x^{[i]}$ be the estimate during i-th iteration for IHT. If RIC satisfies $\delta(3k, m, N) < 1/\sqrt{32}$, then*

$$||x - x^{[i]}||_2 < 2^{-i}||x_s||_2 + 6\epsilon_s, \tag{2.4}$$

where $\epsilon_s = ||x - x_s||_2 + \frac{1}{\sqrt{k}}||x - x_s||_1 + ||e||_2$.

The experiment in [1] indicates that IHT has the lowest among the three algorithms, and requires fewer measurements than CoSaMP and SP.

Also we would like to remark that, in our experience, the bottleneck for sparse recovery is finding supports of signals. As long as the system does not have dramatic errors, if our k-sparse estimate and the true k-sparse solution x_s happen to be the same, i.e., $T^{[i]} = T$, then $x^{[i]}$ would be very close to the true approximation x_s. Roughly speaking, the reason is that once we know the

support of x_s, the system $y = Ax + e \approx Ax_s$ becomes over-determined since we have m equations and only $k < m$ unknown nonzero coefficients of x_s. It's easy to solve the least square problem by multiplying the pseudoinverse of a sub-matrix of A on y. In other words, if we are able to figure out the support of x_s, then the algorithms immediately converge.

Chapter 3

Applications and Extensions of Compressive Sensing

In practice, there are a lot of sparse or compressible signals. So compressive sensing has a wide range of applications and extensions. In this chapter, we discuss two direct applications that are imaging and error correction. Also, we discuss one extension that is robust principle component analysis.

3.1 Imaging

Let an image be represented by a column vector $f \in \mathbb{R}^N$. Suppose f is represented by x under certain orthogonal basis such as Fourier basis and wavelets basis. Say the basis is $\psi_1, \psi_2, \ldots, \psi_N$. Then

$$f = \Psi x, \tag{3.1}$$

where Ψ is an $N \times N$ matrix with waveforms ψ_i as columns. The entries of x decays significantly fast if they are sorted by magnitudes. Say x has the best k-sparse approximation x_s. Suppose we have a noisy measurement vector

$$y = \Phi f + e, \tag{3.2}$$

where Φ is an $m \times N$ matrix with ϕ_i^* as its rows (see Definition 1.1.2) and e is error.

Combine (3.1) and (3.2) and define $A = \Phi\Psi$, we get

$$y = Ax + e. \tag{3.3}$$

Hence, the imaging problem is equivalent to the sparse recovery problem 1.1.2. An important question is: does matrix $A = \Phi\Psi$ have a good RIP? Recall the definition of coherence of the matrix pair (Φ, Ψ),

$$\mu(\Phi, \Psi) = \sqrt{N} \max_{i,j} | \langle \phi_i, \psi_j \rangle |,$$

where ϕ_i is the column vector of Φ and ψ_j is the row vector of Ψ. It's not hard to see that $\mu(\Phi, \Psi) \in [1, \sqrt{N}]$. It's incoherent if if $\mu(\Phi, \Psi)$ is close to 1. It's called maximum incoherent when $\mu(\Phi, \Psi) = 1$. Then we have two typical choices [4]:

1. Φ is spike basis and Ψ is Fourier basis (maximum incoherent);
2. Φ is noiselet and Ψ is wavelets basis [9].

3.2 Error Correction

The error correction problem is modeled as follows [23] [8]. Suppose we have a message vector $f \in \mathbb{R}^K$. Let C be the N by K coding matrix, where $N > K$. Then compute the codeword vector $Cf \in \mathbb{R}^N$. The coding matrix is a tall matrix, i.e., the length N of codeword is larger than the length K of message vector. Let $y \in \mathbb{R}^N$ be received codeword with some error $e \in \mathbb{R}^N$, say

$$g = Cf + e. \tag{3.4}$$

In digital communication, the error vector $e \in \mathbb{R}^N$ has two parts. One is sparse dramatic error (i.e., some entries in g are highly corrupted) and the other is just small nonsparse random Gaussian noise. If there are k dramatic errors in the received codeword, then e has k large entries. Suppose e has the best k-sparse approximation e_s. For decoding, we are interested in locating the sparse dramatic errors. In other words, we want to solve for the best k-sparse approximation e_s. It's easy to construct a matrix $A \in \mathbb{R}^{m \times N}$ of rank $m = N - K$ such that $AC = 0$. Then (3.4) becomes

$$Ag = ACf + Ae. \tag{3.5}$$

Let $y = Ag$. As $AC = 0$, we have

$$y = Ae. \tag{3.6}$$

This is exactly a sparse recovery problem. The error vector e can be found by recovery algorithms in previous section, provided A is the measurement matrix and y is the measurement vector. For example we locate the error e by l_1-minimization:

$$\min_{\tilde{e}\in\mathbb{R}^N} ||\tilde{e}||_1 \quad s.t. \quad ||y - A\tilde{e}||_2 < \epsilon. \tag{3.7}$$

3.3 Robust Principle Component Analysis

Robust principle component analysis can be viewed as an extension of compressive sensing. Suppose we are given an n by n data matrix M which can be decomposed as

$$M = L_0 + S_0,$$

where L_0 is a low-rank matrix and S_0 a is sparse matrix. Sometimes the data matrix is highly corrupted. i.e., We have only available a few entries. Let $\Omega \subset [n] \times [n]$ and P_Ω be the orthogonal projection onto the linear space of matrices supported on Ω,

$$(P_\Omega X)_{ij} = \begin{cases} X_{ij} & (i,j) \in \Omega \\ 0 & (i,j) \notin \Omega \end{cases}$$

We observe

$$Y = P_\Omega(L_0 + S_0) = P_\Omega L_0 + S_0'. \tag{3.8}$$

The question is whether it is possible to recover the low-rank component L_0 from (3.8). It is showed in [5] that under certain conditions, L_0 can be recovered from (3.8) with high probability by solving the following problem:

$$\min_{L,S} ||L||_* + \frac{1}{\sqrt{n}}||S||_1 \quad s.t. \quad Y = P_\Omega(L + S), \tag{3.9}$$

where L has low rank and S is sparse. The nuclear norm $||L||_*$ is defined as the sum of all singular values of L:

$$||L||_* = \sum \sigma_i(L),$$

and the l_1-norm of S treats the sparse matrix S as a long $n \times n$ dimensional vector,

$$||S||_1 = \sum |S_{ij}|.$$

It is assumed that [5]:

1. Ω is uniformly distributed among all subsets of cardinality m obeying $m = 0.1n^2$
2. The incoherence condition on L_0. It requires the low-rank matrix M be non-sparse.

The incoherence condition is necessary. The following example is given in [5]. Let $M = e_1 e_1^*$. Then M only has one nonzero entry. It is both sparse and low-rank. How can we decide whether it is low-rank or sparse? In other words, the separation may not be unique. Actually, the incoherence condition ensures the uniqueness of the solution to (3.9). Also, we can not recover all types of low-rank matrices. We can only recover those low-rank matrices which satisfy the incoherence condition.

Chapter 4

Generalized Eigenvalues Approach

In this chapter, we starts the case when the measurement matrix is a partial Fourier matrix. Fourier measurements are important in digital communications, in MRI and probably other applications. Under this assumption, we show how the recovery problem is related to sparse interpolation. In the first section, we discuss sparse interpolation problem via generalized eigenvalues method. In the second section, we modify the generalized eigenvalues method by taking more than $2k$ measurements and apply it to compressive sensing. We call our new algorithm generalized eigenvalues (GE). Then we compare GE with Iterated Hard Thresholding (IHT). The result indicates GE performs much better if the number of Fourier measurements is small while IHT is better when the number of measurements is very large. Finally, our experiments indicate that GE may be improved by using adaptive measurements.

4.1 Generalized Eigenvalues for Sparse Interpolation with $m = 2k$

4.1.1 Sparse Interpolation

Problem 4.1.1 *(Sparse Interpolation) Suppose we are given a black box for a k-sparse univariate polynomial* $f \in \mathbb{R}[t]$

$$f(t) = \sum_{j=1}^{k} c_j t^{d_j},$$

where $c_1, c_2, \ldots, c_t \in \mathbb{R}$ *and* $d_1, d_2, \ldots, d_k \in \mathbb{Z}_{\geq 0}$. *We assume that the degree of the polynomial is less than* N,

$$0 \leq d_1 < d_2 < \cdots < d_k \leq N - 1.$$

Evaluating

$$y_1 = f(\nu_1), \quad y_2 = f(\nu_2), \quad y_3 = f(\nu_3), \quad \ldots, \quad y_m = f(\nu_m) \tag{4.1}$$

at our own choice of points $\nu_1, \nu_2, \ldots, \nu_m \in \mathbb{C}$, *where* $m = O(k)$, *we want to determine the exponents* $d_1, d_2, \ldots, d_k$ *and the coefficients* $c_1, c_2, \ldots, c_k$.

4.1.2 Generalized Eigenvalues

Let ω_0 be the N-th root of unity,

$$\omega_0 = e^{\frac{2\pi i}{N}}.$$

We evaluate at $1, \omega_0, \omega_0^2, \ldots, \omega_0^{2k-1}$ to get our first $2k$ Fourier coefficients,

$$y_1 = f(1), \quad y_2 = f(\omega_0), \quad y_3 = f(\omega_0^2), \quad \ldots, \quad y_{2k} = f(\omega_0^{2k-1}). \tag{4.2}$$

Let

$$b_1 = \omega_0^{d_1}, \quad , \ldots, \quad b_k = \omega_0^{d_k}. \tag{4.3}$$

If we write the evaluation process in the form of matrix multiplication, then (4.2) is equivalent to

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2k} \end{pmatrix} = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{2k-1} & b_2^{2k-1} & \ldots & b_k^{2k-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{pmatrix}. \tag{4.4}$$

Following the approach [15], we form the Hankel Systems

$$H_0 = \begin{pmatrix} y_1 & y_2 & \cdots & y_k \\ y_2 & y_3 & \cdots & y_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_k & y_{k+1} & \cdots & y_{2k-1} \end{pmatrix}, \quad H_1 = \begin{pmatrix} y_2 & y_3 & \cdots & y_{k+1} \\ y_3 & y_4 & \cdots & y_{k+2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{k+1} & y_{k+2} & \cdots & y_{2k} \end{pmatrix}. \tag{4.5}$$

Then

$$H_0 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{k-1} & b_2^{k-1} & \dots & b_k^{k-1} \end{pmatrix} \begin{pmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_k \end{pmatrix} \begin{pmatrix} 1 & b_1 & \dots & b_1^{k-1} \\ 1 & b_2 & \dots & b_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_k & \dots & b_k^{k-1} \end{pmatrix} \tag{4.6}$$

$$H_1 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{k-1} & \dots & \dots & b_k^{k-1} \end{pmatrix} \begin{pmatrix} b_1c_1 & 0 & \dots & 0 \\ 0 & b_2c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & b_kc_k \end{pmatrix} \begin{pmatrix} 1 & b_1 & \dots & b_1^{k-1} \\ 1 & b_2 & \dots & b_2^{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_k & \dots & b_k^{k-1} \end{pmatrix}. \tag{4.7}$$

One can check that the solution for $z \in \mathbb{C}$ in the generalized eigenvalue problem

$$H_1 v = z H_0 v \tag{4.8}$$

are $b_1, b_2, \dots, b_k$, where corresponding $v_j \in \mathbb{C}^{t\times 1}, 1 \le j \le k$ are generalized eigenvectors. We can solve for the exponents $d_1, d_2, \dots, d_k$ by taking logarithms

$$d_j = log_{\omega_0} b_j,$$

for $1 \leq j \leq k$. And from (4.4), we can also solve for $c_1, c_2, \ldots, c_k$ by multiplying the inverse of Vandermonde matrix V on the left, where

$$V = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{k-1} & b_2^{k-1} & \ldots & b_k^{k-1} \end{pmatrix} \tag{4.9}$$

is called a Vandermonde matrix.

4.1.3 Randomization

Stability is an important issue for an algorithm. However, the generalized eigenvalues method is very sensitive when the terms of $f(x)$ are clustered, i.e., $|d_i - d_j|$ is very small for some $1 \leq i < j \leq k$. Essentially, the reason is that the condition number of the Vandermonde matrix is bad when the terms are clustered [14].

Theorem 4.1.1 *(Main result for Generalized Eigenvalues [14]) Suppose*

$$H_1 v = z H_0 v$$

has generalized eigenvalues $b_1, b_2, \ldots, b_k \in \mathbb{C}$. *Consider the perturbed problem*

$$(H_1 + \epsilon \hat{H_1})v = z(H_0 + \epsilon \hat{H_0})v \tag{4.10}$$

for normalized perturbations $\hat{H_0}, \hat{H_1} \in \mathbb{C}^{k \times k}$ *with* $||\hat{H_0}||_2 = ||H_0||_2$ *and* $||\hat{H_1}||_2 = ||H_1||_2$. *Then (4.10) has generalized eigenvalues* $\tilde{b_1}, \tilde{b_2}, \ldots, \tilde{b_k} \in \mathbb{C}$, *with*

$$|\tilde{b_j} - b_j| < \epsilon \cdot \frac{2k^2 ||(c_1, c_2, \ldots, c_k)||_\infty ||V^{-1}||_2^2}{|c_j|} \tag{4.11}$$

for $1 \leq j \leq k$.

From the theorem, $|\tilde{b_j} - b_j| \propto ||V^{-1}||_2^2$. Note that $cond_2(V) = ||V||_2 ||V^{-1}||_2 \propto ||V^{-1}||_2$ since $||V||_2$ is a constant here. In order to improve the sensitivity of generalized eigenvalues, we have to improve the condition number of Vandermonde matrix V. By [17], a Vandermonde matrix is well-conditioned

if the nodes $d_1, d_2, \ldots, d_k$ are not clustered. Hence [14] suggests to evaluate $f(x)$ at random roots of unity, i.e., we replace ω_0 by $\omega = \omega_0^r$ in (4.2) and (4.4), where r is an integer and $2 \le r \le N-1$.

(Sparse interpolation with randomization) Let $f \in \mathbb{R}[t]$ be a k-sparse polynomial with degree less than N, $f(t) = \sum_{j=1}^{k} c_j t^{d_j}$, where $c_1, c_2, \ldots, c_t \in \mathbb{C}$ and $0 \le d_1 < d_2 < \cdots < d_k \le N-1$. We want to interpolate $f(x)$ by $2k$ evaluations at random primitive roots of unity,

$$y_1 = f(1), \quad y_2 = f(\omega), \quad y_3 = f(\omega^2), \quad \ldots, \quad y_{2k} = f(\omega^{(2k-1)}), \tag{4.12}$$

where $\omega = \omega_0^r = (e^{\frac{2\pi i}{N}})^r$ and $r \in \{2, 3, \ldots, N-1\}$. We need $\gcd(N, r) = 1$ so that $\omega = \omega_0^r$ is a primitive N-th root of unity.

Here we give an intuitive understanding for the randomization. Recall $f(t) = \sum_{j=1}^{k} c_j t^{d_j}$ and suppose we have chosen the integer r. Define another polynomial

$$\hat{f}(t) = \sum_{j=1}^{k} c_j t^{\hat{d_j}},$$

where $0 \le \hat{d_j} \le N-1$ and $\hat{d_j}$ satisfies

$$\hat{d_j} = r d_j \mod N, \quad j = 0, 1, \ldots, k-1. \tag{4.13}$$

Note $0 \le d_1 < d_2 < \cdots < d_k \le N-1$ while $\hat{d_i}$s are not increasing. Then

$$f(\omega^j) = f(\omega_0^{jr}) = \hat{f}(\omega_0^j), \tag{4.14}$$

for $j = 0, 1, \ldots, 2k-1$. Since the evaluations $f(1), f(\omega), f(\omega^{2r}), \ldots, f(\omega^{(2k-1)})$ are known, by (4.14), $\hat{f}(1), \hat{f}(\omega_0), \ldots, \hat{f}(\omega_0^{2k-1})$ are known. Hence, we can interpolate $\hat{f}(t)$ based on evaluations $\hat{f}(1), \hat{f}(\omega_0), \ldots, \hat{f}(\omega_0^{2k-1})$. Note that coefficients of $f(t)$ and $\hat{f}(t)$ remain the same but the exponents have changed. (4.13) gives a one-to-one correspondence between terms of $f(t)$ and $\hat{f}(t)$. After we get the exponents of the new polynomial $\hat{f}(t)$, we can solve for the exponents of the original polynomial $f(t)$ through (4.13). i.e., we can solve for d_js from $\hat{d_j}$s. Since $\gcd(N, r) = 1$, the solution for the system of congruence equations is unique. Therefore the problem of recovering the polynomial $f(t)$ with exponents d_js has changed to recovering $\hat{f}(t)$ with exponents $\hat{d_j}$s.

Let $c_i e^{d_i}$ and $c_j e^{d_j}$ be any two terms of $f(t)$. Define the distance for the two terms as $d_{ij} = \min\{|d_i - d_j|, N - |d_i - d_j|\}$. For example, if $d_i = 0$ and $d_j = N - 1$, then $d_{ij} = 1$ by the definition and we consider them as clustered terms. Let Δ be the minimal distance for terms of $f(t)$,

$$\Delta = \min_{i,j} d_{ij}.$$

Then $f(t)$ has clustered terms if Δ is small.

Similarly, let $\hat{c_i} e^{\hat{d_i}}$ and $\hat{c_j} e^{\hat{d_j}}$ be any two terms of $\hat{f}(t)$. Define their distance as $\hat{d_{ij}} = \min\{|\hat{d_i} - \hat{d_j}|, N - |\hat{d_i} - \hat{d_j}|\}$. Let $\hat{\Delta}$ be the minimal distance for terms of $\hat{f}(t)$,

$$\hat{\Delta} = \min_{i,j} \hat{d_{ij}}.$$

As we mentioned before, the generalized eigenvalues method is especially sensitive in the case that the polynomial has clustered terms (i.e., when Δ is very small). Hence, if Δ happens to be small, then we are supposed to choose a proper integer r to increase $\hat{\Delta}$ as much as possible. But the problem is that we don't have any prior information about how the terms are clustered.

If we randomly choose an integer r, how likely $\hat{\Delta}$ is not small? We have the following theorem.

Theorem 4.1.2 *[14] Let r be chosen uniformly and randomly from $\{1, 2, \ldots, N-1\}$. Then with probability at least 1/2,*

$$\hat{\Delta} > \lfloor \frac{N}{k^2} \rfloor.$$

4.1.4 Relationship between Sparse Interpolation and Compressive Sensing

In this section, we claim sparse interpolation problem is a special sparse recovery problem in compressive sensing. The evaluations in sparse interpolation can actually be viewed as Fourier measurements in compressive sensing.

For sparse interpolation (4.1), we want to determine the k-sparse polynomial

$$f(t) = \sum_{j=1}^{k} c_j t^{d_j} \in \mathbb{R}[t]$$

with exponents at most $N-1$ from m evaluations $f(\nu_1), f(\nu_2), \ldots, f(\nu_m)$. Note $\{f(t) \in \mathbb{R}[t] : \deg f(t) \leq N-1\}$ is a linear space with basis $\{1, t, t^2, \ldots, t^{N-1}\}$. Under this basis, we can represent a k-sparse polynomial $f(t)$ using a k-sparse vector $x \in \mathbb{R}^N$. We write evaluations $f(\nu_1), f(\nu_2), \ldots, f(\nu_m)$ in a column vector $y \in \mathbb{C}^m$. Then

$$y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = Ax = \begin{pmatrix} 1 & \nu_1 & \ldots & \nu_1^{N-1} \\ 1 & \nu_2 & \ldots & \nu_2^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \nu_m & \ldots & \nu_m^{N-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix},$$

where

$$x_j = \begin{cases} c_i & j = d_i + 1 \quad \text{for some} \quad 1 \leq i \leq k \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

Then the sparse interpolation problem becomes recovering k-sparse vector $x \in \mathbb{R}^N$ from $y = Ax \in \mathbb{C}^m$. Therefore, sparse interpolation is a special case of sparse recovery problems of compressive sensing.

In particular, sparse interpolation uses following $m = 2k$ evaluations

$$\nu_1 = 1, \quad \nu_2 = \omega_0, \quad \ldots, \quad \nu_{2k} = \omega_0^{2k-1}.$$

Then the measurement matrix is

$$A = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1 & \omega_0 & \ldots & \omega_0^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_0^{2k-1} & \ldots & \omega_0^{(2k-1)(N-1)} \end{pmatrix}, \tag{4.16}$$

which is formed by the first $2k$ consecutive rows of discrete Fourier matrix.

4.2 Generalized Eigenvalues with $m > 2k$

Note that $2k$ measurements are minimal for reconstructing of $f(t)$, since the k-sparse polynomial $f(t)$ has k unknowns for the exponents $d_1, d_2, \ldots, d_k$ and k unknowns for the magnitudes

$c_1, c_2, \ldots, c_k$. In the previous section, we discussed generalized eigenvalues method, which can solve the sparse interpolation problems by taking only $2k$ Fourier measurements. In section 2.2, we reviewed main recovery algorithms for compressive sensing including l_1-minimization methods and greedy methods. Usually these compressive sensing algorithms take $m = k \log^{O(1)}(N)$ measurements. At first glance, generalized eigenvalues method looks better for requiring fewer measurements. However, based on our experiments, the generalized eigenvalues method with $2k$ measurements is very sensitive to noise. Although [14] has showed robustness of the generalized eigenvalues method with randomization $\omega = \omega_0^r = (e^{\frac{2\pi i}{N}})^r$, the error in its experiment is too small: the random Gaussian noise is between 10^{-12} and 10^{-9} while the nonzero coefficients are between ± 1. However, in practice, we may deal with much larger error than that.

So in this section, we modify the generalized eigenvalues method of section 4.1 and present a new algorithm called generalized eigenvalues (GE).

4.2.1 Algorithm with $m > 2k$

From section 2.1, we know that the measurements could be any form as long as it has good RIP. But generalized eigenvalues method only works for the Fourier measurement matrix. In order to compare them, in our experiments, we would restrict the measurement matrix to be Fourier. We modify (4.16) to form $A \in \mathbb{C}^{m \times N}$ $(m > 2k)$ by taking first m consecutive rows of discrete Fourier matrix as our measurement matrix,

$$A = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 1 & \omega_0 & \ldots & \omega_0^{2(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_0^{m-1} & \ldots & \omega_0^{(m-1)(N-1)} \end{pmatrix}, \tag{4.17}$$

where $\omega_0 = e^{\frac{2\pi i}{N}}$. Let x be a compressible signal and x_s is its best k-sparse approximation. By section 4.1.4, we assume x_s is supported at $T = \{d_1 + 1, d_2 + 1, \ldots, d_k + 1\}$ and $x_{d_j+1} = c_j$ for $j = 1, 2, \ldots, k$. Let $y = Ax + e$ be a noisy measurement vector where e is error. We want to recover x_s from y. Remember the the Hankel matrices H_0 and H_1 in (4.5) are square but here we form

rectangular matrices

$$H_0 = \begin{pmatrix} y_1 & y_2 & \cdots & y_k \\ y_2 & y_3 & \cdots & y_{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m-k} & y_{m-k+1} & \cdots & y_{m-1} \end{pmatrix}, \quad H_1 = \begin{pmatrix} y_2 & y_3 & \cdots & y_{k+1} \\ y_3 & y_4 & \cdots & y_{k+2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{m-k+1} & y_{m-k+2} & \cdots & y_m \end{pmatrix} \tag{4.18}$$

Recall $b_j = \omega_0^{d_j}$ for $1 \le j \le k$. Then similarly, we can solve the generalized eigenvalue problem

$$H_1 v = z H_0 v$$

for eigenvalues. If there is no error, then H_0 and H_1 would have the following decompositions,

$$H_0 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{m-k-1} & b_2^{m-k-1} & \dots & b_k^{m-k-1} \end{pmatrix} \begin{pmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_k \end{pmatrix} \begin{pmatrix} 1 & b_1 & \dots & b_1^{m-k-1} \\ 1 & b_2 & \dots & b_2^{m-k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_k & \dots & b_k^{m-k-1} \end{pmatrix} \tag{4.19}$$

$$H_1 = \begin{pmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{m-k-1} & \dots & \dots & b_k^{m-k-1} \end{pmatrix} \begin{pmatrix} b_1 c_1 & 0 & \dots & 0 \\ 0 & b_2 c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & b_k c_k \end{pmatrix} \begin{pmatrix} 1 & b_1 & \dots & b_1^{m-k-1} \\ 1 & b_2 & \dots & b_2^{m-k-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_k & \dots & b_k^{m-k-1} \end{pmatrix}. \tag{4.20}$$

Hence, if there is no error, then $b_1, b_2, \dots, b_k$ would be exactly the eigenvalues. Then we get $d_1, d_2, \dots, d_k$ by taking logarithms

$$d_j = \log_{\omega_0} b_j,$$

for $1 \le j \le k$.

When the support of x is determined, say $T = \{d_1 + 1, d_2 + 1, \dots, d_k + 1\}$. Then one can

solve for coefficients $c_1, c_2, \ldots, c_k$. Let V be the rectangular Vandermonde matrix,

$$V = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ b_1 & b_2 & \ldots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{m-k-1} & b_2^{m-k-1} & \ldots & b_k^{m-k-1} \end{pmatrix} \tag{4.21}$$

Note $V = A|_T$ and $x|_T = x_s|T = (c_1, c_2, \ldots, c_k)^t$, then

$$y = Ax + e \approx Ax \approx Ax_s = A|_T x_s|T = V \cdot (c_1, c_2, \ldots, c_k)^t$$

The system is over determined since we have m equations and only $k < m$ unknowns. Hence it's easy to solve for c_js as follows:

$$(c_1, c_2, \ldots, c_k)^t = V^\dagger y, \tag{4.22}$$

where $V^\dagger$ is the pseudoinverse of V. Finally reconstruct the signal using (4.15).

4.2.2 Determining the Sparsity

Greedy algorithms in section 2.2 use sparsity k as an input. However, k is usually unknown and it has to be determined. In some paper, they use estimates. For example, in [24], they take $k \approx m/(2 \log N)$. One of the advantages of our generalized eigenvalues method is that we are able to determine the sparsity from the rank of a series of square Hankel matrices. Suppose from a k-sparse signal x we have measured $y_1, y_2, \ldots, y_m$ which are not noisy. Let $1 \leq j \leq \lfloor \frac{m}{2} \rfloor$. We consider the rank of following square Hankel matrix,

$$H^{[j]} = \begin{pmatrix} y_1 & y_2 & \cdots & y_j \\ y_2 & y_3 & \cdots & y_{j+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_j & y_{j+1} & \cdots & y_{2j-1} \end{pmatrix}.$$

If $j \geq k$, then $H^{[j]}$ has the following decomposition

$$H^{[j]} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ b_1 & b_2 & \dots & b_k \\ \vdots & \vdots & \ddots & \vdots \\ b_1^{j-1} & b_2^{j-1} & \dots & b_k^{j-1} \end{pmatrix} \begin{pmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & c_k \end{pmatrix} \begin{pmatrix} 1 & b_1 & \dots & b_1^{j-1} \\ 1 & b_2 & \dots & b_2^{j-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & b_k & \dots & b_k^{j-1} \end{pmatrix}$$

Hence, if $j \geq k$, the rank of $H^{[j]}$ is equal to the sparsity k. If $j < k$, then the rank of $H^{[j]}$ is obviously less than k. Therefore, we determine the sparsity by computing the ranks of $H^{[1]}, H^{[2]}, \dots$ until they remain a finite integer k. Then integer k is the sparsity. Actually, let $j = \lfloor \frac{m}{2} \rfloor$. If m is large enough, then $j > k$. We can determine the sparsity k by only computing the rank of $H^{[j]}$.

However, if the measurement vector y is noisy, then it would be a bit harder to determine the rank of a singular matrix since it usually becomes a full-rank matrix. In this case, we determine an approximate rank as follows. Suppose the eigenvalues of $H^{[j]}$ are

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k \geq \cdots \geq \lambda_j \geq 0.$$

As long as their is no dramatic error,

$$\{\lambda_1, \lambda_2, \dots, \lambda_k\} \approx \{c_1, c_2, \dots, c_k\}, \quad \lambda_{k+1} \approx 0, \quad \dots, \quad \lambda_j \approx 0.$$

Let $g_j, j = 1, 2, \dots, j-1$ be the"gaps" between those eigenvalues,

$$g_1 = \frac{\lambda_1}{\lambda_2}, \quad g_2 = \frac{\lambda_2}{\lambda_3}, \quad \cdots \quad g_{j-1} = \frac{\lambda_{j-1}}{\lambda_j}$$

If the error is a bit large, a safe way is to choose a larger integer k as the sparsity. A larger sparsity k won't affect the result of our generalized eigenvalues method. Also, after we have determined the sparsity k, we replace k by $2k$ and then solve for $d_1, d_2, \dots, d_{2k}$ by generalized eigenvalues method. Since d_js are supposed to integer, from them we select $\hat{d}_1, \hat{d}_2, \dots, \hat{d}_k$ with smallest imaginary parts.

4.2.3 Algorithm

Now we present our algorithm of generalized eigenvalues (GE)

Algorithm 4.2.1 ***(GE)***

- ***Input:*** *A measurement vector $y \in \mathbb{C}^m$, the signal dimension N.*
- ***Output:*** *support and corresponding coefficients $(c_1, d_1+1), (c_2, d_2+1), \ldots, (c_k, d_k+1)$.*

  1. ***Determine an approximate sparsity*** *k. Let $j = \lfloor \frac{m}{2} \rfloor$, construct $H^{[j]}$ and compute its rank using the method of section 4.2.2. Set the sparsity k as an integer that is slightly larger than the rank of $H^{[j]}$.*
  2. ***Construct Hankel Systems*** *H_0* ***and*** *H_1. Use the k from step 1 to form $H_0, H_1 \in \mathbb{C}^{(m-2k)\times 2k}$ as in (4.18) where k is replaced by $2k$.*
  3. ***Solve a generalized eigenvalue problem.*** *Solve $2k$ generalized eigenvalues $b_1, b_2, \ldots, b_{2k}$ from $H_1 v = z H_0 v$.*
  4. ***Take logarithms.*** *$d_j = log_{\omega_0} b_j$, where $\omega_0 = e^{\frac{2\pi i}{N}}$.*
  5. ***Determine support.*** *From above $d_1, d_2, \ldots, d_{2k}$, select $\hat{d_1}, \hat{d_2}, \ldots, \hat{d_k}$ with smallest imaginary parts. Set $\{\hat{d_1}+1, \hat{d_2}+1, \ldots, \hat{d_k}+1\}$ as the support.*
  6. ***Determine coefficients.*** *Construct a rectangular Vandermonde matrix V as in (4.21) and solve for nonzero coefficients $c_1, c_2, \ldots, c_k$ by (4.22).*

4.2.4 GE vs IHT

By section 2.2.2, the greedy algorithms (CosaMP, SP, IHT) are fast and have good stability. And IHT is the best one among them because IHT has the lowest computational cost, and requires fewer measurements than CoSaMP and SP [1]. Hence, in this section, we compare our generalized eigenvalues method (GE) with Iterated Hard Thresholding (IHT).

We randomly generate a k-sparse signal x of length $N = 1009$. We first pick k random positions as the support set of x and for each position, we set the corresponding entry of x via uniform distribution $U(-1, 1)$. Then for GE, we take first m consecutive rows of discrete Fourier matrix as our measurement matrix. For IHT, we randomly select m rows from discrete Fourier matrix and then normalized the columns. In Figure (a), we add complex Gaussian noise of level 10^-5 on the measurements. In Figure (b), we add complex Gaussian noise of level 10^-4 on the measurement. In Figure (c), we add complex Gaussian noise of level 10^-3 on the measurements. And for all Figure (a), (b) and (c), the x-axis is the number of measurements m, which varies from

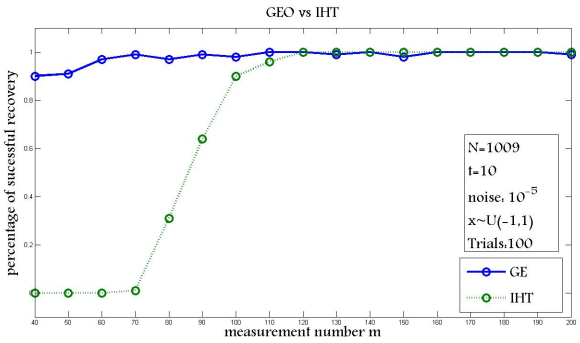

Figure 4.1: (a) GE vs IHT when noise level is 10^{-5}

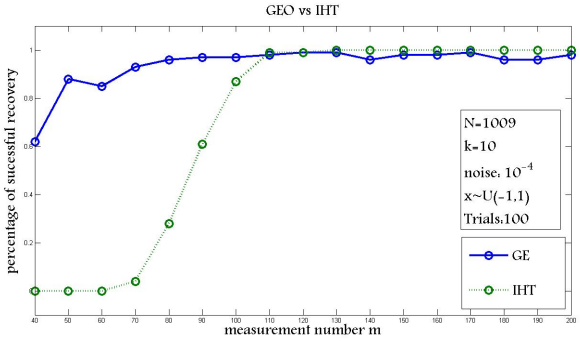

Figure 4.2: (b) GE vs IHT when noise level is 10^{-4}

40 to 200; the y-axis is the percentage of successful recovery over 100 trials (we consider a trial as a success if the 2-norm of the residue is less than 0.1).

In Figure (a), the noise $e \sim 10^{-5}$ is very small while the entries are from $U(-1, 1)$. Both GE and IHT perform well if the number of measurements $m \geq 120$. When $40 \leq m \leq 120$, GE is better than IHT. In Figure (b), the noise $e \sim 10^{-4}$ is a bit larger. If we take m measurements where $m \geq 110$, then GE and IHT perform well. GE is still better than IHT when $m \leq 100$. But GE doesn't have 100% success even if $m \geq 110$. In figure (c), the noise $e \sim 10^{-3}$ is pretty large. When $m \leq 90$, GE is better. But when $m \geq 100$, GE fails more while IHT still have 100% success. To sum up, we have following conclusions.

- IHT is better than GE when number of measurements is very large.
- GE has a much higher probability of success than IHT when the number of measurements is small.

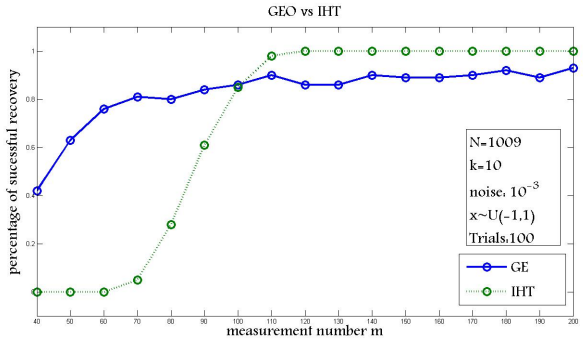

Figure 4.3: (c) GE vs IHT when noise level is 10^{-3}

Above results raise interesting questions:

- Why does GE fail even if the number of measurements is large? For what types of x does GE fail?
- Any possible improvement for GE?

The following are several typical outputs for GE. Recall the assumption in our experiments: $N = 1009, k = 10$. Suppose the error level is 10^{-3} as in Figure (c). For $m = 60$, the successful recovery probability for IHT is almost 0. But GE tells us a lot of information. Let $x \in \mathbb{R}^N$ the generated true k-sparse vector and $\tilde{x} \in \mathbb{C}^N$ be the k-sparse vector recovered by GE. In both x and $\tilde{x}$, the first column means the support and the second column corresponding coefficients.

- Example (a), (Perfect Case.)

$$x = \begin{pmatrix} (76) & -0.8152 \\ (234) & -0.7779 \\ (289) & -0.4052 \\ (388) & -0.5655 \\ (616) & -0.3449 \\ (628) & 0.1609 \\ (663) & -0.5620 \\ (734) & -0.5508 \\ (881) & -0.9275 \\ (976) & -0.9033 \end{pmatrix} \qquad \tilde{x} = \begin{pmatrix} (76) & -0.8153 + 0.0001i \\ (234) & -0.7778 - 0.0002i \\ (289) & -0.4052 - 0.0000i \\ (388) & -0.5656 - 0.0001i \\ (616) & -0.3449 + 0.0000i \\ (628) & 0.1607 - 0.0001i \\ (663) & -0.5619 + 0.0000i \\ (734) & -0.5509 + 0.0001i \\ (881) & -0.9274 + 0.0001i \\ (976) & -0.9033 - 0.0001i \end{pmatrix}$$

- Example (b), (Tail Entries.)

$$x = \begin{pmatrix} (90) & 0.8423 \\ (345) & 0.4390 \\ (363) & -0.0283 \\ (377) & 0.7961 \\ (398) & 0.1152 \\ (465) & -0.8289 \\ (554) & -0.6652 \\ (628) & 0.9431 \\ (761) & 0.6483 \\ (840) & -0.0665 \end{pmatrix} \qquad \tilde{x} = \begin{pmatrix} (90) & 0.8420 + 0.0001i \\ (345) & 0.4376 - 0.0001i \\ (377) & 0.8009 + 0.0033i \\ (398) & 0.1132 + 0.0002i \\ (465) & -0.8293 - 0.0001i \\ (554) & -0.6658 + 0.0004i \\ (628) & 0.9432 + 0.0004i \\ (679) & -0.0002 + 0.0003i \\ (761) & 0.6479 + 0.0002i \\ (840) & -0.0671 - 0.0003i \end{pmatrix}$$

- Example (c), (Clustered Entries.)

$$x = \begin{pmatrix} (130) & -0.1259 \\ (206) & 0.8247 \\ (379) & 0.1281 \\ (519) & -0.1742 \\ (520) & -0.4870 \\ (550) & 0.9199 \\ (569) & 0.1359 \\ (637) & 0.2221 \\ (697) & 0.2754 \\ (838) & 0.1764 \end{pmatrix} \qquad \tilde{x} = \begin{pmatrix} (83) & 0.0001 - 0.0003i \\ (130) & -0.1256 - 0.0003i \\ (206) & 0.8248 + 0.0001i \\ (379) & 0.1285 + 0.0003i \\ (520) & -0.6582 + 0.0318i \\ (550) & 0.9158 - 0.0028i \\ (569) & 0.1329 - 0.0006i \\ (637) & 0.2206 - 0.0003i \\ (697) & 0.2757 - 0.0006i \\ (838) & 0.1762 - 0.0004i \end{pmatrix}$$

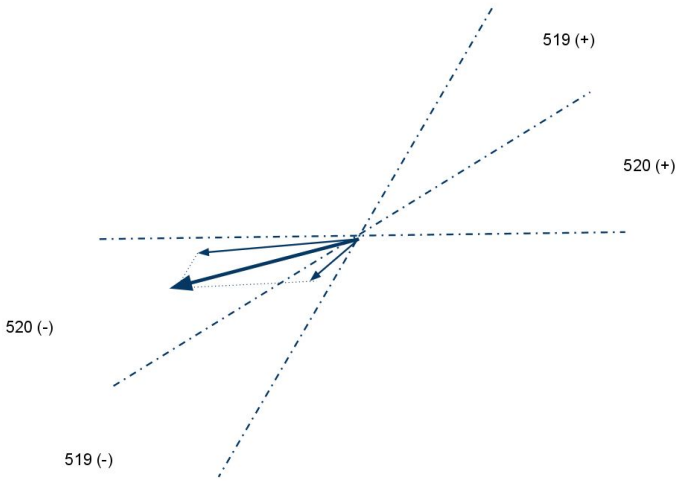

Figure 4.4: (a) Intuitive Explanation of Example (c)

- Example (d), (More Complicated Clustered Entries.)

$$x = \begin{pmatrix} (160) & -0.2052 \\ (162) & 0.5751 \\ (163) & 0.8435 \\ (267) & -0.5448 \\ (514) & 0.1851 \\ (609) & 0.4641 \\ (642) & -0.4350 \\ (790) & 0.6388 \\ (852) & 0.7099 \\ (857) & -0.5934 \end{pmatrix} \qquad \tilde{x} = \begin{pmatrix} (29) & 0.0001 + 0.0001i \\ (159) & -0.0278 + 0.0487i \\ (163) & 1.2288 - 0.0523i \\ (267) & -0.5442 + 0.0000i \\ (514) & 0.1850 - 0.0002i \\ (609) & 0.4644 + 0.0001i \\ (642) & -0.4347 + 0.0002i \\ (790) & 0.6389 + 0.0002i \\ (852) & 0.7101 - 0.0005i \\ (857) & -0.5931 + 0.0007i \end{pmatrix}$$

Example (a) is a perfect recovery. The support of $\tilde{x}$ is exactly the same as the support of true x. The coeffients of x and $\tilde{x}$ are also very close. Note the imaginary parts of $\tilde{x}$ are all close to zeros.

In example (b), the generated true vector x has a tail entry at 363 with a very small magnitude. Since we assume the sparsity is k, the recovered vector $\tilde{x}$ also has a tail entry at 679 with a very small magnitude. Although the recovered support turns out to be "wrong," $||x - \tilde{x}||_2$ is

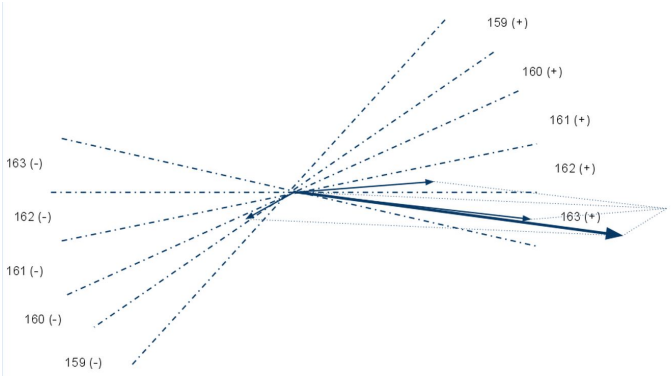

Figure 4.5: (a) Intuitive Explanation of Example (d)

small. Hence we still regard this case as a succuss.

Example (c) is bad case. x has clustered entries at 519 and 520. The recovered vector $\tilde{x}$ only has the entry at 520 with a wrong magnitude. Roughly, if we think of Fourier basis as "directions," then those sparse entries are "vectors." Then $\tilde{x}_{520}$ looks like the sum of x_{519} and x_{520}. See Figure 4.2.4. Note the imaginary part of $\tilde{x}_{520}$ is very large, this actually tells us $\tilde{x}_{520}$ is wrong and x is clustered near 520.

Example (d) has more complicated cluster entries. x has clustered entries at 160,162 and 163 while $\tilde{x}$ are only clustered at 159 and 163. Similarly as example (c), here $\tilde{x}_{159} + \tilde{x}_{163} \approx x_{160} + x_{162} + x_{163}$. Alternatively, as $\tilde{x}_{159}$ is small, it doesn't matter if we regard it as a tail entry. Then $\tilde{x}_{163} \approx x_{160} + x_{162} + x_{163}$. See Figure 4.2.4. And note the imaginary part of $\tilde{x}_{163}$ is very large, this tells us $\tilde{x}_{163}$ is wrong and x is clustered near position (163)!

To sum up, suppose the recovered k-sparse vector $\tilde{x}$ has a nonzero entry $\tilde{x}_i$, then

	If real part of $\tilde{x}_i$ is small	If real part of $\tilde{x}_i$ is large
If imaginary part of $\tilde{x}_i$ is small	Then $\tilde{x}_i$ is a tail entry	Then x_s is clustered near position (i)
If imaginary part of $\tilde{x}_i$ is large	Then $\tilde{x}_i$ is a tail entry	Then $\tilde{x}_i$ is correct

Table 4.1: Conclusion of example (a)-(d)

4.2.5 Adaptive Measurements?

The problem we have not addressed thus far is the recovery of signals with clustered entries. While do not offer a complete solution to this, we note that randomization in section 4.1.3 is of

potential help. Let $\tilde{x}$ be the recovered vector using GE algorithm. Suppose $\tilde{x}_{d_1+1}, \tilde{x}_{d_2+1}, \ldots, \tilde{x}_{d_j+1}$ have both large real parts and imaginary parts, where $\{d_1+1, d_2+1, \ldots, d_j+1\} \subseteq [C-\Theta, C+\Theta]$. Then by Table 4.2.4, x_s has clustered entries within $[C-\gamma\Theta, C+\gamma\Theta]$ for some $\gamma > 1$. Although we known where the the entries are clustered, we are not able to separate them using current measurements. Recall the idea of randomization in section 4.1.3, if we do another round of measurements with randomization, then $\{d_1+1, d_2+1, \ldots, d_j+1\}$ becomes $\{rd_1+1, rd_2+1, \ldots, rd_j+1\}(\mod N)$. And it's possible that these clustered entries $\tilde{x}_{d_1+1}, \tilde{x}_{d_2+1}, \ldots, \tilde{x}_{d_j+1}$ can be recovered. Note, our measurements are adaptive, i.e., we take a second round of measurements based on the information of the first round of measurements.

Chapter 5

Summary and Future Work

Compressive sensing is a novel paradigm for acquiring signals. In Chapter 2, we briefly reviewed previous work of compressive sensing including main concepts like RIP and Measurements matrices and main recovery algorithms like l_1-minimization methods and greedy methods. In Chapter 3, we briefly discussed the applications and extensions of compressive sensing including imaging, error correction and robust principle component analysis. Chapter 4 is the major part of this thesis which contains our contribution to this field. In Chapter 4, we looked at compressive sensing from a different point of view by connecting it to sparse interpolation. We modified the method in sparse interpolation and constructed a new algorithm for compressive sensing that is called generalized eigenvalues (GE). Then we compared it with iterated hard thresholding (IHT). The result suggest GE performs better if the number of Fourier measurements is small while it's sensitive for signals with clustered entries. During our experiments, we found some interesting observations which showed taking adaptive measurements may potentially improves GE. Hence, our future work lies in improving GE such that it can be more stable for clustered signals.

Appendices

Appendix A MATLAB Codes

```
%1, main function

clear;

clc;

N=1009;

t=10;

trials=100;

numj=20;

recResGei=zeros(numj,trials);

recResIht=zeros(numj,trials);


for j=1:numj

m=10*j;

%error:

%(m1,l1):dramatic error

%(m2,l2):small error

m1=0;

l1=1;

l2=3;


for k=1:trials

[x_true]=rvec(N,t);

row_iht=sort(randsample(N,m));

y_iht_true=N*ifft(x_true);

y_iht_true=y_iht_true(row_iht)/sqrt(m);


row_gei=[1:m];

y_gei_true=N*ifft(x_true);

y_gei_true=y_gei_true(row_gei);
```

```
e=err(m,m1,l1,l2);
y_iht=y_iht_true+e;
y_gei=y_gei_true+e;

[T_gei]=supp(y_gei,N,t);
[x_gei]=lsq(T_gei,y_gei,N);
recResGei(j,k)=norm(x_gei-x_true,2);

x_iht=iht(y_iht,N,t,row_iht);
recResIht(j,k)=norm(x_iht-x_true,2);
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%2, Generating error e=e1+e2;
%e1 is large error vector; e2 is small error vector;
%Both e1 and e2 have length m;
%e1 has m1 nonzero entries~10^(-l1);
%It returns e=e1+e2;
function [e]=err(m,m1,l1,l2)
emtx1=randn([m,2]);
a=10^(-l1)*(emtx1(:,1)+1i*emtx1(:,2))/sqrt(2);
row=sort(randsample(m,m1));
e1=zeros(m,1);
e1(row)=a(row);
emtx2=randn([m,2]);
e2=10^(-l2)*(emtx2(:,1)+1i*emtx2(:,2))/sqrt(2);
e=e1+e2;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%3, solving x use IHT
function x=iht(y,N,t,row)
m=size(y,1);
%Initilization
x=zeros(N,1);
r=y;

%Iteration
for j=1:400
r_long=zeros(N,1); % a longer r
r_long(row)=r;
x_temp=x+fft(r_long)./sqrt(m);
[mag,idx]=sort(abs(x_temp),'descend');
x=zeros(N,1);
x(idx(1:t))=x_temp(idx(1:t));
y1=ifft(x).*N./sqrt(m);
y1=y1(row);
r=y-y1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%4, solving a least square problem
function [x]=lsq(T,y,N)
T=T-1;
m=size(y,1);
t=size(T,1);
b=exp(2*pi*1i*T/N);
V=zeros(m,t);
```

```
for j=1:m
    for k=1:t
        V(j,k)=b(k).^(j-1);
    end
end
c=lscov(V,y);
x=zeros(N,1);
T=T+1;
for j=1:t
    x(T(j))=c(j);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%5, This function solves the support
function [I]=supp(r,N,t)
m=size(r,1);
% we choose a larger sparsity t1=2*t
t1=2*t;
H= zeros(m-t1,t1+1);
for j = 1: m-t1
    for k = 1: t1+1
        H(j,k) = r(j+k-1);
    end
end
b=eig(pinv(H(1:m-t1,1:t1))*H(1:m-t1,2:t1+1));
I_temp=log(b).*N./(2*pi*1i);
%we don't use tan because this may delete support 0;
%tan_I =imag(I_temp)./real(I_temp);
%[mag,idx]=sort(abs(tan_I),'ascend');
imag_I =imag(I_temp);
```

```
[mag,idx]=sort(abs(imag_I),'ascend');
I=I_temp(idx(1:t));
I=round(real(I));
I=mod(I,N)+1;
I=sort(I);
I=dele(I);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%6. Generating uniform random vectors
function [x,d]=rvec(N,t)
d = sort(randsample(N,t));
x = zeros(N,1);
for j = 1:t
x(d(j,1),1)=rand(1)*(-1)^(randi([0,1]));
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%7. This function delete repeating entries
%input: sorted integer vector with repeating entries
%output: sorted integer vector without repeating entries
function [d2]=dele(d1)
d1_min=min(d1);
d1_max=max(d1);
range=d1_max-d1_min+1;
count=histc(d1,d1_min-0.5:1:d1_max+0.5);
d2=zeros(0,1);
for j=1:range
    if count(j)>0
```

```
    d2=[d2;d1_min+j-1];
    end
end
```

Bibliography

[1] Jeffrey D. Blanchard, Coralia Cartis B, Jared Tanner B, and Andrew Thompson B. Phase transitions for greedy sparse approximation algorithms. submitted, 2009.

[2] Thomas Blumensath and Mike E. Davies. Iterative hard thresholding for compressed sensing. CoRR, abs/0805.0510, 2008.

[3] Emmanuel C, Justin Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements, 2005.

[4] E. J. Candes. Compressive sampling. Proceedings of the International Congress of Mathematicians, Madrid, Spain. 2006.

[5] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? CoRR, abs/0912.3599, 2009.

[6] Emmanuel J. Candes and Justin Romberg. Quantitative robust uncertainty principles and optimally sparse decompositions. Found. Comput. Math., 6(2):227–254, 2006.

[7] Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory, 52(2):489–509, 2006.

[8] Emmanuel J. Candès and Terence Tao. Decoding by linear programming. CoRR, abs/math/0502327, 2005.

[9] R. Coifman, F. Geshwind, and Y. Meyer. Noiselets. Appl. Comput. Harmon. Anal., 10(1):27–44, 2001.

[10] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. IEEE Trans. Inf. Theor., 55(5):2230–2249, 2009.

[11] A. d'Aspremont and L. El Ghaoui. Testing the nullspace property using semidefinite programming. Math. Progr., February 2010. Special issue on machine learning.

[12] Ronald A. DeVore. Deterministic constructions of compressed sensing matrices. J. Complex., 23(4-6):918–925, 2007.

[13] David L. Donoho. Compressed sensing. IEEE Transactions on Information Theory, 52(4):1289–1306, 2006.

[14] Mark Giesbrecht, George Labahn, and Wen-shin Lee. Symbolic-numeric sparse interpolation of multivariate polynomials. In ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation, pages 116–123, New York, NY, USA, 2006. ACM.

[15] Gene H. Golub, Peyman Milanfar, and James Varah. A stable numerical method for inverting shape from moments. SIAM J. Sci. Comput, 21:1222–1243, 1999.

[16] Rémi Gribonval and Karin Schnass. Dictionary identification - sparse matrix-factorisation via l_1-minimisation. CoRR, abs/0904.4774, 2009.

[17] Nicholas J. Higham. Accuracy and Stability of Numerical Algorithms. Second edition, 2002.

[18] M. A. Iwen. Simple deterministically constructible rip matrices with sublinear fourier sampling requirements. In in in Proc. of Proceedings of CISS 2008, 2008.

[19] Kenneth Kreutz-Delgado, Joseph F. Murray, Bhaskar D. Rao, Kjersti Engan, Te-Won Lee, and Terrence J. Sejnowski. Dictionary learning algorithms for sparse representation. Neural Comput., 15(2):349–396, 2003.

[20] Julien Mairal, Guillermo Sapiro, and Michael Elad. Multiscale sparse image representationwith learned dictionaries. In ICIP (3), pages 105–108, 2007.

[21] Stephane Mallat. A Wavelet Tour of Signal Processing. AP Professional, London, 1997.

[22] B. K. Natarajan. Sparse approximate solutions to linear systems. SIAM J. Comput., 24(2):227–234, 1995.

[23] Mark Rudelson and Roman Vershynin. Geometric approach to error correcting codes and reconstruction of signals. INT. MATH. RES. NOT, 64:4019–4041, 2005.

[24] Joel A. Tropp and Deanna Needell. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. CoRR, abs/0803.2392, 2008.