UiO : Department of Mathematics
University of Oslo

Recommender systems analysis by compressed sensing

Kristofer Munsterhjelm
Master's Thesis, Spring 2018

Abstract

Recommender systems are algorithms that suggest content or products to users on the internet. These are becoming ever more important due to the massive growth of content on popular web sites, yet their design is often only guided by empirical results. This has two drawbacks: mathematical analysis lags behind the use of the methods, and the methods may focus too much on immediate results instead of taking a wider perspective, leading to unintended consequences such as social media polarization.

To help offset those drawbacks, this thesis considers both how one may analyze recommender systems more rigorously, as well as how they may be improved by optimizing not just for short-term results. The thesis approaches recommender systems from a compressed sensing perspective, starting with an explanation of compressed sensing as the study of how to approximate the cardinality minimization problem. It then proceeds to give a review of how compressed sensing can be generalized to approximate two matrix-valued problems, called matrix sensing and matrix completion. The application of matrix completion to the bilinear factorization model used in recommender systems follows, and we finish by investigating improvements to the basic bilinear factorization model, as well as suggesting other directions of improvement.

Acknowledgements

I would like to thank my advisors: Øyvind Ryan and Anders Hansen. My frequent review meetings with Øyvind have been of great help, and he has always been quick to respond to any questions by mail. Both of my advisors also contributed to me choosing compressed sensing as the subject of this thesis: Øyvind, through his applications of linear algebra course; and Anders, through his lecture series on compressed sensing.

Thanks also to my friends and family, for their support and for giving me much needed breaks from academic matters, so that I could recover and keep going without exhausting myself.

Kristofer
May 2018

Contents

Notation and notes

A rectangular matrix is by default $M \in \mathbb{R}^{m\times n}$, so that m is consistently the number of rows and n is the number of columns. We're interested in underdetermined systems, so $m \leq n$.

If the matrix is square, I use $M \in \mathbb{R}^{n\times n}$.

Vectors (but not matrices, matrix rows or columns) are denoted by boldface, like this: $A\mathbf{x} = \mathbf{y}$, and are column vectors unless otherwise specified. Here $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. Not denoting matrix rows or columns by boldface might be a bit inconsistent in retrospect, as they can be considered vectors, but I chose not to do it.

Index sets $[k]$ are defined as follows:

$$[k] \quad = \quad \{x \in \mathbb{N} : 1 \leq x \leq k\}$$

and a vector restriction $\mathbf{x}_P$ where P is a set of integers, is defined as

$$(x_P)_i \quad = \quad \begin{cases} x_i & i \in P \\ 0 & \text{otherwise} \end{cases}$$

and a vector $\mathbf{x}$ for which $\mathbf{x} = \mathbf{x}_P$ is said to be *supported on* P.

I use $\mid X \mid$ for a set X to denote the cardinality of X.

I have restricted myself to the case of real numbers, as it makes more sense given my subject. Some of the results given in the thesis hold in the complex domain as well (e.g. the null space property (definition 7) is identical for $A \in \mathbb{C}^{m\times n}$, and in this case, theorem 8 holds for all $\mathbf{x} \in \mathbb{C}^n$). To stay consistent, I have only considered real variables for these results, e.g. real matrices for the NSP.

The null space of a matrix A or operator $\mathcal{A}$ is consistently denoted $\ker(A)$ or $\ker(\mathcal{A})$ in this thesis even though some papers use ker and other papers use $Null$.

When I refer to particular page numbers of papers, these numbers are of the publication in question. E.g. Vandenberghe and Boyd [1996] is pages 49-95 of *SIAM review 38*. A reference to Vandenberghe and Boyd [1996, p. 49] refers to the first page of the paper proper, which is page 49 of *SIAM review 38*, rather than to the 49th page of the paper.

I use both $X_{i,j}$ and X_{ij} to index the element at the ith row and jth column of a matrix X. I usually use the latter notation unless it would be ambiguous.

1 Introduction

The internet is growing at an extraordinary rate. In 1994 there were around 2800 web sites on the internet [Gray, 1996]; now there are around 170 million active sites [Netcraft, 2018]. This rapid growth has led to an information explosion. The very first video on YouTube was uploaded on April 2005; and now, YouTube serves a billion hours of video to its users a day [YouTube, 2018].

With this kind of information explosion, it becomes increasingly harder to decide what web pages to visit, what music to listen to, or what videos to watch. In 1994, Yahoo worked by manually cataloging web pages in a hierarchy; but this is clearly impractical with 170 million active sites. Instead, the manual approach, which doesn't scale with the information explosion, has to a great degree been replaced by the use of algorithms.

One of these types of algorithm is the recommender system. A recommender system suggests items to users based on the users' past behavior: what items they have visited in the past, and how they've rated those items. The purpose of the system is to direct a user's attention towards a few recommended items of generally good quality from the point of view of that user, instead of leaving him to manually choose from thousands or millions of items, some of which he might not even know exists. Recommender systems can be used to increase sales in an online store, or to improve users' experience on publishing sites (like YouTube) and on social networks.

While algorithms scale better than the manual approach, they only do what they've been constructed to do. Thus, they are limited by their assumptions and can fail to take indirect effects into account. As recommender systems are being used by increasingly larger stores and publishing sites, this becomes a more serious matter. For instance, recommender systems might assume honesty in the data given to them, and thus be vulnerable to manipulation. Furthermore, since the algorithms are often the result of empirical rather than analytical design, we may not know under what conditions they produce a good result. We usually know that they give good results on standard datasets, but we can't ensure that the results generalize unless we have worst-case bounds.

Fortunately, a particular type of recommender system model, the bilinear or low rank matrix model, lends itself to analysis based on compressed sensing. Compressed sensing, as a field, started with the observation that sparse signals $\mathbf{x}$ can be recovered from observations $\mathbf{y}$ under the assumption that $A\mathbf{x} = \mathbf{y}$ for a known matrix A, if A obeys certain properties, by relaxing a hard problem to an easier one. This strategy was later generalized to the case of matrices, where we instead face the problem of recovering an unknown matrix X under the condition that

$$\mathcal{A}(X) \quad = \quad \mathcal{A}(Y)$$

for a known linear operator $\mathcal{A}$; and the generalization produced similar results for the matrix domain as was found for the vector domain. The results were further generalized to the case where one might not be able to control $\mathcal{A}$ precisely, which is just the situation we have in recommender systems.

In the recommender system setting, we have ratings of items by users, but naturally, not every item has been rated by every user. We want to extrapolate the ratings to the unknown user-item pairs to make a good guess as to what items a user would like, so we can present those items to that user. The ratings we alredy know form Y, and the operator $\mathcal{A}$ zeroes out the unrated user-item pairs.

By using the generalized compressed sensing results, we can get some analytical bounds on when it is possible to get a full matrix X that's close to Y. The algorithms most often used in practice have weaker bounds.

1.1 Purpose and structure of the thesis

I started with some knowledge of compressed sensing and a wish to investigate recommender systems from a more formal mathematical perspective. Most web pages about collaborative filtering and recommender systems are written from an implementation perspective, giving information about how to implement particular algorithms, or what libraries to use; but there's relatively speaking fewer articles approaching recommender systems from a formally theoretical angle.

Since I'm interested in finding out how to improve recommender systems without making alterations *ad hoc*, I have focused on the theory in this thesis. I have, however, not let entirely go of the practical: it turns out that the type of algorithm that gives the best formal bounds also is impractical to run on large scale data as is produced by the information explosion, hence I have also considered the (weaker) bounds that have been found for more practical algorithms.

The thesis as such has four parts. In the first part, I go through compressed sensing on vectors, starting with the ideal problem we'd like to solve, and then showing the difficulty of solving this problem on a computer. I then investigate how the problem can be relaxed, and what bounds exist for recovery on the relaxed problem.

In the second part, I consider two generalizations of compressed sensing, focusing on matrix sensing: the situation where $\mathcal{A}$ is known, Y partially so, and we want to recover a low rank X so that $\mathcal{A}(X) = \mathcal{A}(Y)$. I also give a definition of matrix completion, which is the case where $\mathcal{A}$ is not entirely known in advance.

In the third part, I investigate matrix completion in a recommender systems setting. Matrix completion is more realistic because, if $\mathcal{A}$ retains the known user-item pairs, we don't have complete control of $\mathcal{A}$; it depends on which users rate which items. Two algorithms end up being of interest: the semidefinite programming problem, which exactly solves the relaxation of the minimum rank problem, but is very slow; and the alternating least squares algorithm, which is practical (and was used in e.g. the Netflix challenge), but for which the analytical bounds are looser.

In the fourth part, I explain two ways to extend recommender systems: one to make the bilinear matrix model more accurate, and another to help keep the system from developing blind spots by focusing too much on the information it already knows. I give a summary of an article relevant to each type of extension, and then finish by suggesting future directions of research; one regarding compressed sensing in general, and two on improving recommender systems.

1.2 Code

I have not written much code for this thesis. I have written code to construct a few figures, as well as to attempt an empirical test of an eigenvalue-based function f detailed in section 18. These pieces of code may be found at this GitHub repository.

Part I

Compressed sensing on vectors

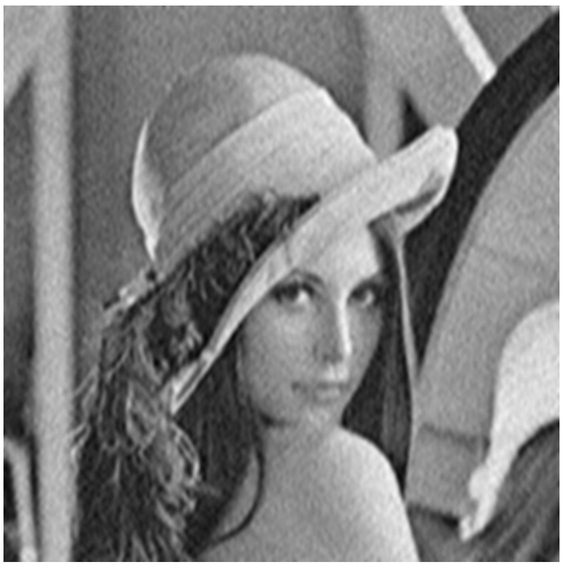
2 Introduction to compressed sensing on vectors

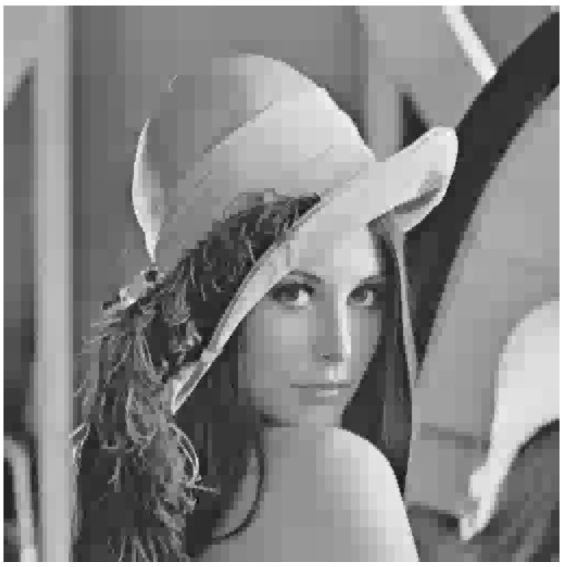
Signals from physical processes are often compressible. The existence of compression software and multimedia codecs shows this to be the case: lossless compression can reduce the storage space of audio signals to 50%-70% [van Beurden, 2013], while lossy compression can be near-transparent at compression rates in excess of 1:6 for music[1] and 1:40 for speech [Hoene et al., 2013]. Image and video compress even further: JPEG encoded pictures can attain compression rates of 1:25 with very low distortion[2]; while DVD video is usually compressed at 3-9 Mbit/s at a resolution of 720x576 at 25 frames per second, which if stored uncompressed would amount to 248 Mbit/s.

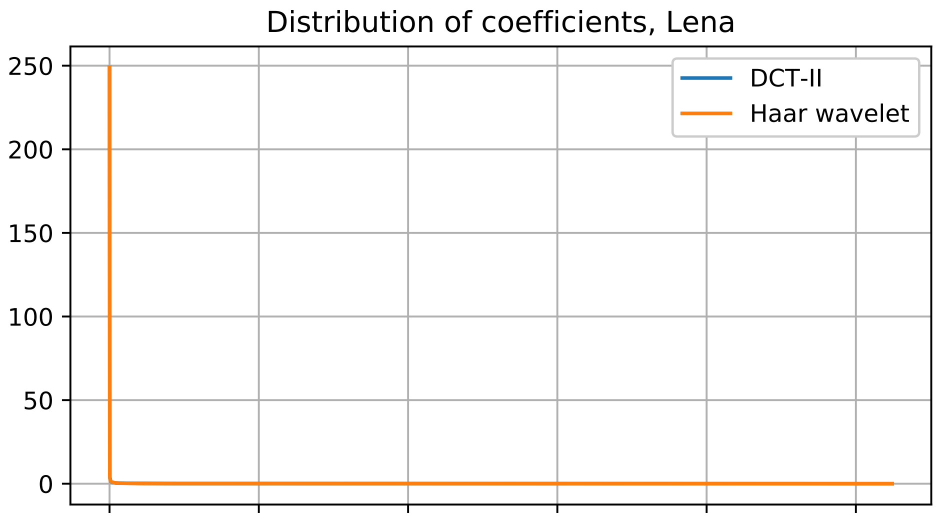
[1]MP3 format, see Pan [1995]
[2]JPEG quality 80

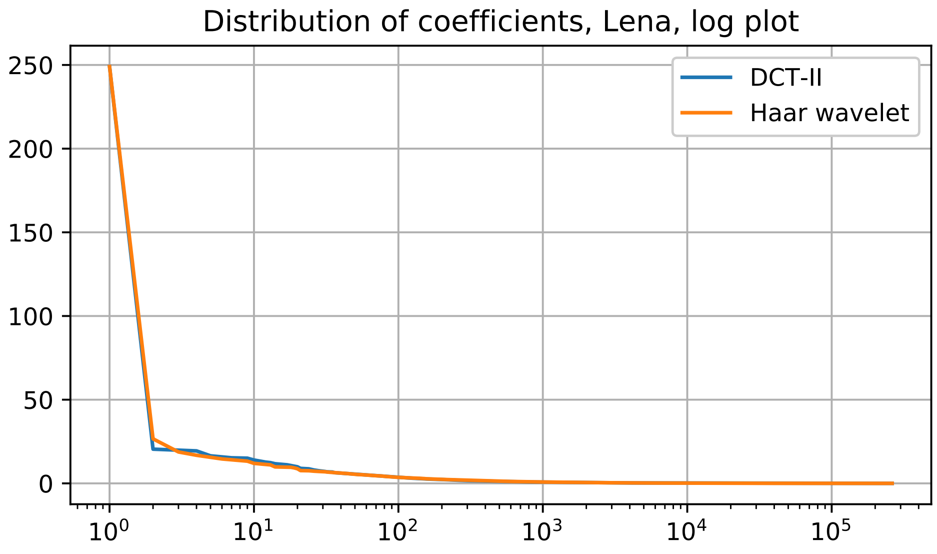
In a linear system, compressibility takes the form of the signal in question being expressible in some basis, with a few of the basis vectors contributing to a large proportion of that signal. When the change of basis moves more of the contribution to a few vectors, the effect is called energy compaction.

An example of energy compaction can be seen below. Here we run the standard Lena picture through a discrete cosine or wavelet transform, observe the distribution of the coefficient magnitudes after the change of basis, and then set 98% of the coefficients to zero. The inverse transformation produces pictures that are still of reasonable quality.









As the pictures and plots show, both the DCT and the Haar wavelet show energy compaction.

However, even though compression provides great reduction of storage space once a signal has been recorded, an ordinary change of basis still requires that we have all the data before we perform the transform. Thus, the vast majority of implementations take the seemingly paradoxical approach of first sampling a very detailed signal, and then throwing most of it away.

In some cases, this is no problem: sound equipment can record audio sufficiently quickly for the acquisition itself to work, cameras have sufficient resolution, and modern computers have enough space to store either the raw sampled audio signal or image.

But in other cases, acquiring all the data can take a lot of time or be outright impossible. An example of the former is magnetic resonance imaging for medical purposes, where the sampling process is slow and the patient may not be able to keep still throughout. And in a recommender system, the data to be acquired consists of users rating items (movies or products, say); and it's clearly unreasonable to expect every user to rate every item in advance.

This naturally motivates the question of whether it's possible to perform both the sample acquisition and the basis change as one step, if all we know is that the target basis would concentrate most of the information in just a few coefficients. In the words of Candès and Plan, 2010, we would like to reconstruct a signal by sampling only a number of times nearly proportional to its information content, rather than to its bandwidth.

It turns out that compressed sensing lets us do just that, if a few more conditions hold.

3 Sparse signal theory

Suppose we have a signal that we suspect is the sum of a small number of contributions; for instance, a time-domain signal that is the sum of a few distinct frequencies, but we do not know which ones. The signal can be represented as a vector, $\mathbf{y} \in \mathbb{R}^m$, and by our assumptions, it is the sum of a few basis vectors in the space of contributions. Going by our example of a time-domain signal, $\mathbf{y}$ is our sampled intensities over time and the basis is the Fourier space, since each basis vector in the Fourier domain encodes a different frequency.

Formally, if we let $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}$ be the basis vectors for the space in question, we have that for the signal $\mathbf{y}$,

$$A\mathbf{x} = \mathbf{y} \tag{3.1}$$

for an unknown $\mathbf{x}$, where the basis matrix A is defined as

$$A = \left[\begin{array}{c|c|c|c|c} & & & & \\ \mathbf{x_1} & \mathbf{x_2} & \mathbf{x_3} & \ldots & \mathbf{x_n} \\ & & & & \end{array}\right]$$

If the number of basis vectors n is greater than the vector length m, we have an underdetermined system. If the system is consistent (has any solution at all), it has an infinite number of them. We thus can't single out any one solution without adding more preconditions.

If the signal $\mathbf{y}$ is indeed a sum of only a few distinct basis vectors, then the number of nonzero elements of $\mathbf{x}$ is also limited, which gives us an additional criterion we can use to narrow down the number of solutions to (3.1).

We would like to more generally deal with the concept of the number of nonzero elements being bounded, so we define the support set $supp(x)$ of a vector as follows:
Definition 1. The support set of $\mathbf{x}$, $supp(\mathbf{x})$ is defined as:

$$supp(\mathbf{x}) \quad = \quad \{j \in [n] : x_j \neq 0\}$$

This is the set of indices j where x_j is nonzero.

This definition lets us formalize the concept of sparsity: we say that $\mathbf{x}$ is *k-sparse* if $supp(\mathbf{x}) \leq k$, i.e. $\mathbf{x}$ contains at most k nonzero elements.

If $\mathbf{y}$ indeed is the sum of a few distinct basis vectors, then when we're given $A\mathbf{x} = \mathbf{y}$, we would prefer to find $\mathbf{x}$ so that $\mid supp(\mathbf{x}) \mid$ is minimized. We can restate this as an optimization problem:

$$\min \mid supp(\mathbf{x}) \mid \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{y}$$

In the optimization problem, we're interested in the optimal solution $\mathbf{x}^\sharp$ rather than the value of the objective function at that optimum. This problem is called ℓ_0 *minimization* or *cardinality minimization* [Recht et al., 2010].

3.1 The name ℓ_0 minimization

The cardinality of the support set, $\mid supp(\mathbf{x}) \mid$, is usually denoted $\parallel \mathbf{x} \parallel_0$. This is somewhat of an abuse of notation. The use of this term arises from that if $\parallel \mathbf{x} \parallel_p$ is the ℓ_p norm defined by

$$\parallel \mathbf{x} \parallel_p \quad = \quad \sqrt[\frac{1}{p}]{\sum_{i=1}^{n} \mid x_i \mid^p}$$

then we have that

$$\begin{aligned} \parallel \mathbf{x} \parallel_p^p \quad &= \quad \sum_{i=1}^{n} \mid x_i \mid^p \\ \lim_{p \to 0} \parallel \mathbf{x} \parallel_p^p \quad &= \quad \sum_{i=1}^{n} \lim_{p \to 0} \mid x_i \mid^p \end{aligned}$$

When $x_i = 0$, we get that $\mid x_i \mid^p$ converges to 0 if $x_i = 0$ since

$$\lim_{p \to 0} 0^p = 0$$

and when $x_i \neq 0$, $\lim_{p \to 0} \mid x_i \mid^p = \mid x_i \mid^0 = 1$. Hence

$$\lim_{p \to 0} \parallel \mathbf{x} \parallel_p^p \;=\; \sum_{i=1}^{n} \begin{cases} 1 & x_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

Properly speaking, $\parallel \mathbf{x} \parallel_0^0 = \mid supp(\mathbf{x}) \mid$ in the limit.

In compressed sensing literature, $\lim_{p \to 0} \parallel \mathbf{x} \parallel_p^p$ is usually referred to as $\parallel \mathbf{x} \parallel_0$ even though it is not a proper norm. For instance, the $\parallel \mathbf{x} \parallel_0$ "norm" violates the absolute homogeneity property of $\mid \alpha \mid \parallel \mathbf{x} \parallel = \parallel \alpha \mathbf{x} \parallel$.

Armed with this concept and notation, we arrive at the following notation for cardinality minimization:

$$\min \parallel \mathbf{x} \parallel_0 \text{ subject to } A\mathbf{x} = \mathbf{y} \tag{3.2}$$

The problem also admits a decision formulation, which will be useful later for proving complexity properties:

$$\begin{array}{ll} \text{given } A \in \mathbb{Q}^{m \times n}, \mathbf{y} \in \mathbb{Q}^m, k \in \mathbb{N} & \\ \text{does there exist an } \mathbf{x} \text{ so that} & \parallel \mathbf{x} \parallel_0 \leq k,\ A\mathbf{x} = \mathbf{y}? \end{array} \tag{3.3}$$

We'll get to why the decision formulation is defined on the rationals, next.

4 Complexity and hardness results

We next consider whether the optimization problem can be efficiently solved in practice. While having a formalized notion of what we're trying to accomplish can be useful, it is still of limited practical purpose if the problem itself turns out to be intractable.

4.1 Concepts from complexity theory

4.1.1 Abstract computers and Boolean functions

To do so, we first need some concepts from complexity theory. In particular, we need to quantify how long time an algorithm takes to run on a computer. As actual computers would be hard to formalize, we start with an idealized computer. Since complexity theory isn't the main subject of this thesis, the definitions will be somewhat brief; more detailed definitions can be found in [Arora and Barak, 2009].

A Turing machine is an idealized computer, and consists of one or more infinitely long tapes and a state machine that can move a tape left, right, read or write to a current tape position, as well as transition to a *halting state* where no further state transitions are possible. We'll assume for convenience that our Turing machines have one input tape and one output tape each.

Next we need concepts of what it is we want to calculate, and how we may calculate it. The important part is to distinguish the abstract concept of a function or mapping, which doesn't go into the matter of implementation, from an algorithm, which is a particular implementation that realizes one of these functions or mappings.

The Boolean set $\mathbf{B}$ is defined as $\mathbf{B} = \{0, 1\}$. A Boolean function may take as input a string of n Boolean values $\mathbf{B}^n$, or the set of all possible strings of Boolean values, $\mathbf{B}^* = \cup_{i=0}^{\infty} \mathbf{B}^i$; and returns a single Boolean value, i.e. $f : \mathbf{B}^n \to \mathbf{B}$ and $g : \mathbf{B}^* \to \mathbf{B}$ are Boolean functions.

A decision problem is a problem of the form of a yes/no question. This is what we want to calculate: it takes some input and provides an answer (either yes or no). One example of a decision problem is the decision problem for ℓ_0 optimization 3.3: for that decision problem, the inputs are the matrix A and observed vector $\mathbf{y}$.

However, since we want to analyze how a Turing machine might calculate a decision problem, decision problems are often defined to only take Boolean inputs and to return some Boolean output. In other words, they're represented by a Boolean function f, or alternatively as a set of values for which $f = 1$ (i.e. $L_f = \{x : f(x) = 1\}$).

4.1.2 Implicit encoding of decision problems

Since I've been using the term "decision problem" in a more indirect sense, to refer to the problem or question itself, I'll give a definition that fits that more indirect meaning. This may be nonstandard with respect to formal computer science terminology.

That the direct definition of a decision problem only refers to Boolean functions means we have a problem if we want to answer questions about, for instance, whether there exists a vector $\mathbf{x}$ so that for given A,$\mathbf{y}$ and k, $A\mathbf{x} = \mathbf{y}$ and $\parallel \mathbf{x} \parallel_0 \leq k$. We need an encoding, in this case from $(A, \mathbf{y}, k)$ to a sequence of Boolean values. Then the decision problem itself can be considered a Boolean function taking this encoding of $(A, \mathbf{y}, k)$ and returning either "yes" or "no".

To specify this encoding directly would sidetrack from the complexity analysis, so in my indirect use of the term "decision problem", I have omitted specifying such an encoding function. A decision problem in this thesis is thus a mapping from a set $\mathbf{Y} \to \mathbf{B}$, where $\mathbf{Y}$ depends on the question at hand, so that the encoding function is implicit: there's a $e_f : \mathbf{Y} \to \mathbf{B}^*$ and $f : \mathbf{B}^* \to \mathbf{B}$, where the formal computer science decision problem is f, and the indirect use here refers to $e_f \circ f$.

There is however, one aspect to the Boolean encoding that we'll have to specify directly. We want to analyze the time complexity for algorithms that calculate f on a Turing machine. For the time complexity to be finite for every finite input $y \in \mathbf{Y}$, e_f must encode finite y to $\mathbf{B}^n$ for some finite n (usually depending on y). Thus $\mathbf{Y}$ cannot be an uncountably infinite set; otherwise n could be infinite for some choices of finite y. The problems we'll consider take arguments that are either real numbers or based on them (e.g. for the ℓ_0 optimization problem, A is a matrix of real numbers, $\mathbf{y}$ is a vector, and k is an integer). For the implicit e_f to map finite cardinality inputs to finite cardinality outputs, it can only take arguments from some countable subset of the reals rather than from the reals themselves, as $\mathbb{R}$ is uncountably infinite.

Which countable subset we choose may influence the complexity of the problem.[3] Thus I'll specify the subsets of the reals used as inputs to the decision problems when defining those problems.

4.1.3 Algorithms and runtime

A particular Turing machine can be defined by its initial tapes, initial state, and the state transition machine that acts as rules for how the Turing machine proceeds to work. A particular Turing machine can be designed to calculate a particular problem in this way: different state machine and initial settings, different results.

An algorithm computing f is a Turing machine so that for any $x \in \mathbf{B}^*$, if the machine is initialized with the start of its input state set to x, it will reach the halting state in a finite number of steps, with the output tape set to $f(x)$. The runtime of the algorithm is the number of steps it takes, as a function of n.

We also need a concept of runtime that's not unduly influenced by the particular nature of the computer. Suppose that we have a function f that we want to find out the runtime of. We want a measure of runtime that is not affected by different hardware or by hard-coded speedups for a small subset of the inputs (e.g. a lookup table). Thus we arrive at the following concepts:

We say that $f(x)$ is $O(g(x))$ if there exist constants $M > 0$ and $x_0 > 0$ so that for all $x > x_0$, $f(x) < Mg(x)$.

Let then $T_M(x)$ be equal to the number of steps required for the Turing machine M to reach a halting state if given x as input. The algorithm represented by M has runtime $Q(n)$ if $T_M(x)$ is $O(Q(\mid x \mid)$; and a function f has runtime $Q(n)$ if there exists an algorithm calculating f whose associated Turing machine has runtime $Q(n)$.

[3]For instance, numerically determining if $\mid x_1 + \cdots + x_n \mid > k$ is easy if $\mathbf{x} \in \mathbb{N}^n, k \in \mathbb{N}$, but if we consider algebraic numbers instead of natural numbers, doing so is much harder. [Allender et al., 2009, thm 1.4]

4.1.4 Complexity classes

With the above in mind, we can then define two complexity classes.
Definition 2. The complexity class P (for "polynomial time") is the set of functions that can be computed by a Turing machine in time $T(n)$ where T is a polynomial of n. In simpler terms, it's the set of functions that have runtime T(n).

This class is usually considered to contain problems that are tractable, i.e. can be computed in reasonable time, although strictly speaking, one could argue that polynomial time does not exactly correspond to reasonable time. Section 1.5.1 of Arora and Barak [2009] discusses polynomial time and tractability, and to what degree they correspond.

Next, we need to define a more general class that is about whether we can verify solutions quickly, rather than whether we can solve problems quickly. To do so, we have to establish some other concepts first.

Let $p(n)$ mean that a function that is a polynomial of n. Let a verifier of a decision problem f, $g_f : \mathbf{B}^*, \mathbf{B}^* \to \mathbf{B}$ be a function that takes an input of $\mathbf{B}^*$, which is the Boolean encoding of the input to a decision problem, and another also of $\mathbf{B}^*$, which is called a certificate.

For g_f to be a verifier of f, it must furthermore satisfy the following properties:

- If $f(a) = 1$, then there exists a certificate $b \in \mathbf{B}^{p(|a|)}$ so that $g(a, b) = 1$
- If $f(a) \neq 1$, then for every possible $b \in \mathbf{B}^*$, $g(a, b) = 0$.
- g_f is in P.

We can then proceed with the definition:
Definition 3. The complexity class NP (for "non-deterministic polynomial time") is the set of decision problems whose outputs can be verified in polynomial time. That is, a decision problem $f(a) : \mathbf{B}^* \to \mathbf{B}$ is in NP iff there exists a verifier g_f for it.

We also need to introduce the complexity class *coNP*. *coNP* can most easily be defined by reference to NP. A decision problem $g(a) : \mathbf{B}^* \to \mathbf{B}$ is in *coNP* iff there exists another decision problem f in NP so that

$$g(a) = \begin{cases} 0 & \text{if } f(a) = 1 \\ 1 & \text{if } f(a) = 0 \end{cases}$$

for all $a \in \mathbf{B}^*$.

Informally, a problem f is in NP if it's easy to verify a "yes" answer, and in *coNP* if it's easy to verify a "no" answer. Yet *coNP* and NP are not disjoint, since there are problems in both NP and *coNP*.[4]

[4]One such example can be produced from factoring: is input integer x prime? If yes, a primality certificate proves this, otherwise, a list of the prime factors of x certifies that x is composite.

4.1.5 Complexity class hardness

We next proceed to define classes that consist of the most difficult problems in some class, in the sense that if we can solve one of those most difficult problems, we can solve every problem in that class.

First, we need a way of showing that f is at least as difficult to solve as g:
Definition 4. A polynomial-time transformation from a decision problem $f \in NP$ to another problem $g \in NP$ is a function $t_{f\to g} : \mathbf{B}^* \to \mathbf{B}^*$ in P so that for all x, $f(x) = 1$ iff $g(t_{f\to g}(x)) = 1$.

If there exists such a transformation from f to g, then that implies that if we can use g to solve every problem in f with only a polynomial time penalty. This lets us define particularly hard problems:
Definition 5. A decision problem g is NP-hard if, for every problem f in NP, there exists a polynomial-time transformation from f to g.

Solving an NP-hard problem in polynomial time is equivalent to solving every problem in NP in polynomial time. Finally, we have the hardest possible problems still in NP:
Definition 6. A decision problem f is NP-complete if it is in NP and is NP-hard.

Solving any NP-complete problem in polynomial time would be sufficient to show $P = NP$.

4.2 Complexity of ℓ_0 optimization

We thus have a foundation for determining whether a given problem (in the sense of a question) can be solved (i.e. answered) in reasonable time.

The ℓ_0 optimization decision problem (3.3) is in NP: if $r \in \mathbf{B}^*$ is a binary encoding of the arguments, then the verifier

$$g(a,\, b) \;=\; \begin{cases} 1 & \text{if b encodes } x,\ a \text{ encodes } A \text{ and } \mathbf{y},\ \|\ \mathbf{x}\ \|_0 \leq k, \text{ and } A\mathbf{x} = \mathbf{y} \\ 0 & \text{otherwise} \end{cases}$$

can be evaluated in polynomial time. Determining whether $\|\ \mathbf{x}\ \|_0 \leq k$ can be done in a number of steps proportional to the length of x times the max number of bits required to encode an element in $\mathbf{x}$, and $A\mathbf{x}$ can be calculated in $O(n^2)$ multiplications.

However, the ℓ_0 problem is also NP-hard, and thus NP-complete.

To prove that the ℓ_0 decision problem is NP-hard, I'll construct a use a reduction from another NP-hard problem, the partition problem.[5]

[5]This proof is Exercise 2.10 from Foucart and Rauhut, 2013.

The partition problem is that of determining whether, given a multiset (set with repetitions allowed) of rationals $S = \{s_1, \ldots, s_n\}$, if there exist disjoint S_1, S_2 with $S_1 \cup S_2 = S$ and $\sum_{x\in S_1} x = \sum_{y\in S_2} y$. The decision problem returns 1 if this is the case, 0 otherwise. The partition problem is in NP since given candidates S_1 and S_2, one can simply compare sums. We can also find a proof that the partition problem is NP-hard in Karp [1972].

We'll show a reduction to any ℓ_p decision problem with $0 \le p < 1$. This produces a more general result with $p = 0$ as a special case, and will later be useful for eliminating a large number of problems as potential worst-case tractable approximations to the ℓ_0 problem.

We want to show that given a set S and p with $0 \le p < 1$, we can construct A, $\mathbf{y}$, k so that the analogous decision problem

$$\text{given } A \in \mathbb{Q}^{m\times n},\ \mathbf{y} \in \mathbb{Q}^m, \text{and } k \in \mathbb{N}, \text{does there exist an } \mathbf{x} \text{ so that } \parallel x \parallel_p^p \le k,\ A\mathbf{x} = \mathbf{y}?$$

is true iff the partition problem has a solution (i.e. there exist S_1, S_2 satisfying the criteria given for the partition problem for that particular S). Note here that I have used $\parallel x \parallel_p^p$ since $\parallel x \parallel_0$ is somewhat of an abuse of notation and it makes more sense to consider it as $\parallel x \parallel_0^0$.

Let $S = \{s_1, \cdots s_n\}$ as before and let the variables to the ℓ_p optimization problem be

$$\begin{aligned} A &= \left[\begin{array}{ccccc|ccccc} s_1 & s_2 & s_3 & \cdots & s_n & -s_1 & -s_2 & -s_3 & \cdots & -s_n \\ \hline & & & & & & & & & \\ & & I_n & & & & & I_n & & \\ & & & & & & & & & \end{array}\right] \\ y &= \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \end{pmatrix}^T \\ k &= n \end{aligned}$$

We first want to show that the only valid solutions to the ℓ_p optimization problem for $\parallel x \parallel_p \le n$ with $x = (x_1, \cdots, x_{2n})$ must have $x_i \in \{0, 1\}$. First note that due to the identity matrix components of A, we have that $x_i + x_{n+i} = 1$ for all $0 \le i \le n$.

Since the ℓ_p-norm is concave when $p < 1$, the minimum attained for $\mid x_i \mid^p + \mid x_{n+i} \mid^p$ subject to $x_i + x_{n+i} = 1$ is at either $x_i = 0$ or $x_{n+i} = 0$. Thus, if every $x_i \in \{0, 1\}$, $\parallel x \parallel_p^p = \sum_{i=1}^n \mid x_i \mid^p + \mid x_{n+i} \mid^p = n$. Since each $\mid x_i \mid^p + \mid x_{n+i} \mid^p$ is at its minimum in this case, any $x_i \notin \{0, 1\}$ would increase the objective value above n and so would not be a valid solution.

Now suppose that we have a solution to the partition problem. We can define indicator variables $a_1, \cdots, a_n$ and $b_1, \cdots, b_n \in \{0, 1\}$ so that $S_1 = \{s_j \mid 0 \le j \le n,\ a_j = 1\}$ and $S_2 = \{s_j \mid 0 \le k \le n,\ b_j = 1\}$. Since every member of S is either in S_1 or S_2, $a_j + b_j = 1$ for all $0 \le k \le n$, and since S_1 and S_2 solve the partition problem, we have $\sum_{x\in S_1} x = \sum_{y\in S_2} y$ or $\sum_{i=1}^n a_i s_i - b_i s_i = 0$. $x = (a_1, \cdots, a_n, b_1, \cdots b_n)$ is thus a valid solution to the ℓ_0 decision problem, since $\parallel x \parallel_0 = \parallel x \parallel_p^p = n \le k$ and $Ax = y$.

In the other direction, any x that solves the ℓ_p optimization problem gives a partition of S into S_1 and S_2. Since $\| x \|_p^p = n$ subject to $x_i + x_{i+n} = 1$, we know that every element of x must be either 0 or 1. Then, since $y_1 = 0$, we have that $\sum_{i=1}^n s_i x_i - s_i x_{i+n} = 0$. Thus

$$\sum_{i=1}^{n} s_i x_i = \sum_{i=1}^{n} s_i x_{i+n}$$

and with multisets

$$\begin{aligned} S_1 &= \{s_i \mid 0 \leq i \leq n,\ x_i = 1\} \\ S_2 &= \{s_{i+n} \mid 0 \leq i \leq n,\ x_{i+n} = 1\} \end{aligned}$$

that is equivalent to

$$\sum_{x \in S_1} x = \sum_{y \in S_2} y$$

which is the constraint for the partition problem, and thus S_1 and S_2 solve this instance of the partition problem.

5 Convex relaxation

We have thus determined that the decision form of ℓ_p-minimization is NP-complete for $0 \leq p < 1$. Any algorithm that can exactly answer the ℓ_p decision problem can also be used to solve the partition problem, which is also NP-complete.

The key ingredient of the proof is that the ℓ_p quasinorm is concave. This is what forces the nonzero values of the optimal vector towards ± 1, thus solving the partition problem. For $p \geq 1$, this forcing no longer works, as the norm is a proper norm and thus convex.

We might then consider using ℓ_p minimization with $p \geq 1$ as a proxy for solving ℓ_0 minimization. The convex relaxation should be significantly easier to solve since convexity implies that all local minima are global minima. We're still left with the question of which p to use, however. Fortunately, there is a result that excludes every choice but $p = 1$ for sparse recovery purposes.

Suppose we choose to replace (3.2) by

$$\min \| \mathbf{x} \|_p \text{ subject to } A\mathbf{x} = \mathbf{y}$$

with $p > 1$. Then for any $A \in \mathbb{R}^{m \times n}$ with $m < n$, we can choose $\mathbf{y}$ so that $A\mathbf{e_i} = \mathbf{y}$, with $\mathbf{e_i}$ some unit vector, but where $\mathbf{e_i}$ is not an optimal solution to ℓ_p minimization. This implies that the ℓ_p minimization problem will, in the worst case, fail to recover even a 1-sparse solution and so is not suited to perform sparse recovery.

Proof. Since $m < n$, the set $\ker(A) \setminus \mathbf{0}$ is non-empty. So choose a vector $\mathbf{v} \in \ker(A) \setminus \mathbf{0}$ and some i so that $v_i \neq 0$, and then let $\mathbf{y} = A\mathbf{e_i}$. For contradiction, suppose that $\mathbf{e_i}$ will be an optimal solution to the ℓ_p optimization problem for $p > 1$ with this choice of $\mathbf{y}$. Now consider linear combinations $\mathbf{e_i} + t\mathbf{v}$ of $\mathbf{e_i}$ with $t \geq 0$.

First note that all such combinations also pass the equality constraint of $A\mathbf{x} = \mathbf{y}$ since $At\mathbf{v} = \mathbf{0}$.

Then consider the ℓ_p norm of such a combination:

$$\begin{aligned} \| \mathbf{e_i} + t\mathbf{v} \|_q^q &= | 1 + tv_i |^q + \sum_{k \neq i} | tv_k |^q \\ &= | 1 + tv_i |^q + | t |^q \sum_{k \neq i} | v_k |^q \end{aligned}$$

If $\mathbf{e_i}$ is to be an optimum, then $t = 0$ must be the minimum for $\| \mathbf{e_i} + t\mathbf{v} \|_q^q$. If $\| \mathbf{e_i} + t\mathbf{v} \|_q^q$ is differentiable with respect to t at $t = 0$, then its derivative must be equal to 0 there.

Suppose $| t |< \frac{1}{v_i}$, which is possible since $v_i \neq 0$ by definition. Define

$$\begin{aligned} f_-(t) &= (1 + tv_i)^q + t^q \sum_{k \neq i} | v_k |^q \\ f_+(t) &= (1 + tv_i)^q - t^q \sum_{k \neq i} | v_k |^q \\ f(t) &= \begin{cases} f_+(t) & t \geq 0 \\ f_-(t) & t \leq 0 \end{cases} \end{aligned}$$

and we see that if $| t |< \frac{1}{v_i}$, then $f(t) = \| \mathbf{e_i} + t\mathbf{v} \|_q^q$.

The derivatives of f are

$$\begin{aligned} f_+'(t) &= qv_i(1 + tv_i)^{q-1} + qt^{q-1} \sum_{k \neq i} | v_k |^q \\ f_-'(t) &= qv_i(1 + tv_i)^{q-1} - qt^{q-1} \sum_{k \neq i} | v_k |^q \end{aligned}$$

and

$$\begin{aligned} \lim_{t \to 0+} f_+'(t) &= \lim_{t \to 0-} f_-'(t) \\ &= qv_i \end{aligned}$$

The limits agree, so $f'(0) = qv_i$. Since we chose $\mathbf{v}$ so that $v_i \neq 0$, $f'(0) \neq 0$ and thus $t = 0$ is not a minimum. $\square$

Remark. Note that if we try to do this proof for $q = 1$, we get

$$\begin{aligned} f_+(t) &= 1 + tv_i + t\sum_{k\neq i} \mid v_k \mid^q \\ f_-(t) &= 1 + tv_i - t\sum_{k\neq i} \mid v_k \mid^q \end{aligned}$$

and

$$\begin{aligned} f'_+(t) &= v_i + \sum_{k\neq i} \mid v_k \mid^q \\ f'_-(t) &= v_i - \sum_{k\neq i} \mid v_k \mid^q \end{aligned}$$

In this case, $\lim_{t\to 0+} f'_+(t) \neq \lim_{t\to 0-} f'_-(t)$ and the proof fails. We'll later see that ℓ_p with $p = 1$ can work.

Every $p > 1$ is disqualified in the sense that there are some 1-sparse solutions that ℓ_p minimization can't recover no matter what matrix A happens to be. We have also seen that all $p < 1$ are worst-case intractable unless $P = NP$, due to the reduction from the partition problem.

We are left with the option of $p = 1$, which is the closest we can get to ℓ_0 without losing convexity. The corresponding optimization problem becomes

$$\min \parallel \mathbf{x} \parallel_1 \text{ subject to } A\mathbf{x} = \mathbf{y} \tag{5.1}$$

with the decision problem

$$\begin{array}{c} \text{Given } A \in \mathbb{Q}^{m\times n},\ \mathbf{y} \in \mathbb{Q}^m,\ k \in \mathbb{N}, \\ \text{does there exist an } \mathbf{x} \text{ so that } \quad \parallel \mathbf{x} \parallel_1 \leq k,\ A\mathbf{x} = \mathbf{y}? \end{array} \tag{5.2}$$

The ℓ_1 optimization problem is called basis pursuit.

Basis pursuit is a bit more difficult to handle than for $p > 1$ because the absolute value function is not differentiable. This is not insurmountable, however, and linear programming yields a simple formulation:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i + z_i \\ \text{subject to} \quad & A(\mathbf{x} - \mathbf{z}) = \mathbf{y} \\ & \mathbf{x} \geq 0 \\ & \mathbf{z} \geq 0 \end{aligned}$$

where the inequality constraints on the vectors $\mathbf{x}$ and $\mathbf{z}$ are element-wise.[6] The main drawback to linear programming is that it is slow, and so other algorithms have been developed to solve (5.1) more quickly. We won't go into detail here, but rather proceed to investigate what kind of signals can be recovered by ℓ_1 optimization.

5.1 Recovery of sparse vectors

We can then proceed to determine when ℓ_1 optimization will recover the same vector as ℓ_0 optimization. Broadly, the results are that there are tight bounds on when the solutions agree, but the results depend on a matrix property that is difficult to calculate. Thus, one uses more loose bounds in practice.

First, let $\mathbf{v}$ be a vector of n elements, $S \subset [n]$ is a set of indices with $\bar{S} \subset [n]$ its complement, and $\mathbf{v}_S$ is the restriction of $\mathbf{v}$ to S.

Then the null space property of a matrix A is defined as follows:
Definition 7. A matrix $A \in \mathbb{R}^{m \times n}$ satisfies the null space property relative to some $S \subset [n]$ if

$$\| \mathbf{v}_S \|_1 < \| \mathbf{v}_{\bar{S}} \|_1 \qquad \forall \mathbf{v} \in \text{ker(A)} \backslash \{\mathbf{0}\}$$

and the matrix satisfies the null space property of order s if the above holds for every S with cardinality at most s.

We can then use the null space property to prove when the ℓ_1 minimizer coincides with the ℓ_0 minimizer. To do so, we first need an auxiliary theorem:
Theorem 8. *Given $A \in \mathbb{R}^{m \times n}$, if the null space property relative to S holds for A, then whenever the ℓ_1-optimal solution is supported on S, it is the only solution. When the ℓ_1 solution is unique whenever it is supported on S, then A satisfies the null space property relative to S.*

Proof. This is an expansion of the proof of Foucart and Rauhut [2013, thm 4.4]. Suppose that the null space property holds relative to S and that we have two different solutions to $A\mathbf{x} = \mathbf{y}$, i.e. $A\mathbf{x} = A\mathbf{z} = \mathbf{y}$ with $\mathbf{x} \neq \mathbf{z}$. Let $\mathbf{v} = \mathbf{x} - \mathbf{z}$ and note that $\mathbf{v} \in \ker(A) \backslash \{\mathbf{0}\}$. We have the following:

$$\| \mathbf{x} \|_1 \quad \leq \quad \| \mathbf{x} - \mathbf{z}_S \|_1 + \| \mathbf{z}_S \|_1$$

by the reverse triangle inequality. Continuing:

$$\begin{aligned} &= \quad \| \mathbf{v}_S \|_1 + \| \mathbf{z}_S \|_1 \\ &< \quad \| \mathbf{v}_{\bar{S}} \|_1 + \| \mathbf{z}_S \|_1 \end{aligned}$$

[6]This is a fairly common way of minimizing absolute value terms in linear programming, but note that it doesn't work for *maximization.*

by the null space property relative to S, so

$$\begin{aligned} \| \mathbf{x} \|_1 &< \| \mathbf{v}_{\bar{S}} \|_1 + \| \mathbf{z}_S \|_1 \\ &= \| (\mathbf{x} - \mathbf{z})_{\bar{S}} \|_1 + \| \mathbf{z}_S \|_1 \\ &= \| -\mathbf{z}_{\bar{S}} \|_1 + \| \mathbf{z}_S \|_1 \\ &= \| \mathbf{z} \|_1 \end{aligned}$$

Thus for any other $\mathbf{z}$ satisfying $A\mathbf{x} = A\mathbf{z} = \mathbf{y}$, $\| \mathbf{x} \|_1 < \| \mathbf{z} \|_1$. Thus $\| \mathbf{z} \|_1$ can't be another optimal solution to the ℓ_1 optimization problem.

In the other direction, suppose that whenever $\mathbf{x}$ is supported on S and $A\mathbf{x} = \mathbf{y}$, it is the unique solution to (5.1). Since it holds across all different $\mathbf{y}$, it also holds if $\mathbf{y}$ is chosen so that $A\mathbf{v}_S = \mathbf{y}$, where $\mathbf{v}$ is an arbitrary vector in $\ker(A)\backslash\{\mathbf{0}\}$.

Since $A\mathbf{v_S} = \mathbf{y}$, then by the assumption we started with, $\mathbf{v}_S$ is the unique solution to (5.1) due to being supported on S. But since $\mathbf{v} \in \ker(A)\backslash\{\mathbf{0}\}$, we have

$$\begin{aligned} A(\mathbf{v}_S + \mathbf{v}_{\bar{S}}) &= \mathbf{0} \\ A\mathbf{v}_S &= -A\mathbf{v}_{\bar{S}} \end{aligned}$$

so $\mathbf{x} = -\,\mathbf{v}_{\bar{S}}$ is also a solution to (5.1) with this particular choice of $\mathbf{y}$. Since $\mathbf{v}_S$ is the unique solution, we must have that

$$\begin{aligned} \| \mathbf{v}_S \|_1 &< \| -\mathbf{v}_{\bar{S}} \|_1 \\ &= \| \mathbf{v}_{\bar{S}} \|_1 \end{aligned}$$

and since $\mathbf{v}$ was chosen arbitrarily from $\ker(A)\backslash\{\mathbf{0}\}$,

$$\| \mathbf{v}_S \|_1 < \| \mathbf{v}_{\bar{S}} \|_1 \ \forall \mathbf{v} \in \ker(\mathrm{A})\backslash\{\mathbf{0}\}$$

which is the NSP relative to S. □

The auxiliary theorem then admits a simple generalization:

Given $A \in \mathbb{R}^{m \times n}$, let $\mathbf{y} \in \mathbb{R}^m$ be arbitrary. Suppose an s-sparse vector $\mathbf{x} \in \mathbb{R}^n$ is a solution to $A\mathbf{x} = \mathbf{y}$. Then $\mathbf{x}$ is the only solution to (5.1) iff A satisfies the null space property of order s, and $\mathbf{x}$ is also the optimal solution to (3.2).

Suppose otherwise. Then we have some $\mathbf{z}$ so that $A\mathbf{x} = A\mathbf{z} = \mathbf{y}$. Since $\mathbf{x}$ is s-sparse, and the null space property of order s holds for A, $\mathbf{x}$ is supported on some S, $\mathrm{card}(S) \leq s$. An argument completely analogous to the first proof for the auxiliary theorem suffices to show $\| \mathbf{x} \|_1 < \| \mathbf{z} \|_1$.

To show that in this case, the ℓ_0 optimizer coincides with the ℓ_1 optimizer, let $\mathbf{x}$ be the ℓ_1 optimizer and $\mathbf{z}$ the ℓ_0 optimizer. Then since $\| \mathbf{z} \|_0 \leq \| \mathbf{x} \|_0$, $\mathbf{z}$ must also be s-sparse. Let $\mathbf{x}$ be supported on S and $\mathbf{z}$ be supported on T. If $\mathbf{x} \neq \mathbf{z}$, then the null space property relative to S implies that $\| \mathbf{x} \|_1 < \| \mathbf{z} \|_1$. But the null space property relative to T implies $\| \mathbf{z} \|_1 < \| \mathbf{x} \|_1$. This is clearly impossible, so $\mathbf{x} = \mathbf{z}$.

6 Handling noisy signals

We've looked at exactly sparse signals: vectors $\mathbf{y}$ that are sparse when transformed to the basis provided by A, i.e. for which $supp(\mathbf{x}) \ll n$ and $A\mathbf{x} = \mathbf{y}$. But real world signals will rarely be exactly sparse. There are multiple factors that can keep a signal from being sparse, such as:

- Fundamental properties of the signal in question: the signal $\mathbf{x}$ isn't exactly s-sparse but the energy falls off rapidly, e.g. when the signal is the result of an energy compaction transformation like the picture examples in the introduction.
- Sampling and sensor noise: any random small-magnitude contribution to our input signal $\mathbf{y}$. This can either be modeled as statistical noise (e.g. additive Gaussian random noise), or in a worst case sense, as being arbitrary within the constraints given by small magnitude, usually $\mathbf{y} = \mathbf{y_0} + \mathbf{n}$ with $\| \mathbf{n} \|$ less than some fixed noise magnitude t.
- Limited precision of the A matrix, either due to physical effects or computer representation problems such as floating point quantization error, as in approximating a Fourier matrix in $\mathbb{Q}$.

We'll be considering the first and second points here.

In the first case, if we're using a model where $\mathbf{x}$ isn't exactly s-sparse but close to it, we call the signal $\mathbf{x}$ *compressible*. In the second case, a slight noise contribution to $\mathbf{y}$, as in $\tilde{\mathbf{y}} = A\mathbf{z} + \mathbf{e}$, may perturb solutions $\mathbf{x}$ to $A\mathbf{x} = \tilde{\mathbf{y}}$ to not be s-sparse for any s, even if $\| \mathbf{e} \|$ is bounded arbitrarily close to zero. Thus we have to directly alter the optimization problem, and an optimization problem that takes noise into account is called *robust*.

The robust setting is also useful where our model is strictly speaking wrong, e.g. where both linear and nonlinear effects contribute to the signal $\mathbf{y}$, but where the nonlinear contribution is small compared to the linear contribution. In such a case, we can treat the nonlinear contribution as some noise term $\mathbf{e}$, and bound its magnitude even though we don't know the statistical properties of the nonlinear process that generates $\mathbf{e}$.

6.1 Compressible signals

Since a compressible signal is nearly s-sparse, it makes sense to define an error measure of how far a signal is from exact sparsity. Let

$$\sigma_s(\mathbf{x})_p \quad = \quad \inf\{\| \mathbf{x} - \mathbf{z} \|_p,\, \mathbf{z} \in \mathbb{R}^n,\, \mathbf{z}\, \text{s-sparse}\}$$

give the error, in ℓ_p-norm, of the best possible s-sparse approximation to $\mathbf{x}$. The infimum is attained by letting $\mathbf{z}$ equal $\mathbf{x}$ for the s elements of $\mathbf{x}$ with greatest absolute values, and zero everywhere else. One way of formalizing that is as follows:

Define the non-increasing rearrangement of $\mathbf{x}$ as $\mathbf{x}^*$ where

$$x_1^* \geq x_2^* \geq \cdots \geq x_N^* \geq 0$$

Then let $\pi : [n] \to [n]$ be a permutation so that

$$x_j^* \quad = \quad | \, x_{\pi(j)} \, |$$

We can then define σ_s equivalently as

$$\sigma_s(\mathbf{x})_p \quad = \quad \| \, (x_{s+1}^*, \, x_{s+2}^*, \, \ldots, \, x_n^*) \, \|_p$$

The exact value of $\sigma_s(\mathbf{x})_p$ may vary depending on p, but for any given $\mathbf{x}$ and s, the best approximation $\mathbf{z}$ is the same no matter the particular p value. [Foucart and Rauhut, 2013]

The function $\sigma_s(\mathbf{x})_p$ will become useful once we start discussing recovery bounds on the recovery of signals that aren't exactly sparse. We'll find that the degree to which ℓ_1 optimization can recover the exact ℓ_0 result depends on $\sigma_s(\mathbf{x})_p$.

6.2 Robust optimization

Suppose that we're dealing with a noisy observation $\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{e}$ where $\mathbf{e}$ is a noise term with $0 <\| \mathbf{e} \| \leq r$. We might hope that we could employ the same strategy as in 6.1 and ensure recovery for sparsity k with noise as long as we have recovery for sparsity $s > k$ without noise. However, that is impossible for a large class of A.

Suppose that for A and a noise-free $\mathbf{y}$ (that we're not given access to), $A\mathbf{x} = \mathbf{y}$ has a unique s-sparse solution $\mathbf{x}^\sharp$.

If A is surjective, then in the worst case, there exists an $\mathbf{e}$ so that even though $A\mathbf{x} = \mathbf{y}$ has an s-sparse solution, $A\mathbf{x} = \tilde{\mathbf{y}}$ has no q-sparse solution for any $q < n$, where $\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{e}$.

To show this, define $\mathbf{v}$ so that it satisfies

$$A\mathbf{v}_k \quad = \quad \begin{cases} 1 & {\mathbf{x}^\sharp}_k = 0 \\ \text{sign}({\mathbf{x}^\sharp}_k) & {\mathbf{x}^\sharp}_k \neq 0 \end{cases}$$

Then letting $\tilde{\mathbf{y}} = \mathbf{y} + r\frac{\mathbf{v}}{\|\mathbf{v}\|}$, there is no q-sparse solution to $A\mathbf{x} = \tilde{\mathbf{y}}$, and we can't use exact sparse recovery to undo the noise and recover $\mathbf{x}^\sharp$, as the noise perturbs the corresponding solution to no longer be sparse for any $q < n$.

Since most candidate matrices A for compressed sensing are surjective, this is a serious drawback to exact recovery in the presence of noisy $\mathbf{y}$. We have to change the optimization problem itself to circumvent the result.

6.3 Bounds on recovery

6.3.1 Recovery of compressible vectors

For a compressible signal, we would expect to pay some penalty to recover the signal as it is no longer exactly sparse. Our intuition would in that case be correct, as we'll see.

The analogous null space property for a compressible signal is called the *stable null space property* in Foucart and Rauhut [2013]. It is defined as follows:

A matrix $A \in \mathbb{R}^{m \times n}$ satisfies the stable null space property relative to some $S \subset [n]$ and constant $0 < \rho < 1$ if

$$\| \mathbf{v}_S \|_1 < \rho \| \mathbf{v}_{\bar{S}} \|_1 \qquad \forall \mathbf{v} \in \ker(\mathrm{A}) \backslash \{0\}$$

The matrix satisfies the null space property of order s if this holds relative to all S for which $| S | \leq s$.

We then have the following result (again from Foucart and Rauhut [2013]):
Theorem 9. *Suppose that a matrix $A \in \mathbb{R}^{m \times n}$ satisfies the stable null space property of order s and with constant $0 < \rho < 1$, with arbitrary $\mathbf{y} \in \mathbb{R}^m$ as before. For any $\mathbf{x}$ so that $A\mathbf{y} = \mathbf{x}$, the ℓ_1 optimal solution $\mathbf{z}$ (with $A\mathbf{y} = A\mathbf{z} = \mathbf{x}$) approximates $\mathbf{x}$ with a bound on the ℓ_1 error*

$$\| \mathbf{x} - \mathbf{z} \|_1 \quad \leq \quad \frac{2(1+\rho)}{1-\rho} \sigma_s(\mathbf{x})_1$$

and if there are multiple solutions with the same ℓ_1 norm, then the bound holds for all of them.

Here the compressibility of the unknown target vector $\mathbf{x}$ comes into play in the $\sigma_s(\mathbf{x})_1$ term; the further away from exact sparsity $\mathbf{x}$ is, the greater the worst case error becomes.

6.3.2 Robust recovery

In 6.2, we found that we need to alter the optimization problem in case of arbitrary bounded additive noise. We can do that by relaxing the equality constraint.

The robust recovery model is that we observe $\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{e}$. For any solution $\mathbf{x_s}$ satisfying $A\mathbf{x} = \mathbf{y}$, we have $\| \mathbf{e} \| = \| A\mathbf{x_s} - \tilde{\mathbf{y}} \|$. We can thus ensure $\mathbf{x_s}$ is a feasible solution in the face of noise if we generalize the constraint $A\mathbf{x} = \mathbf{y}$ to $\| A\mathbf{x} - \mathbf{y} \| < \| \mathbf{e} \|$. Doing so gives *quadratically constrained basis pursuit* or QCBP:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \| \mathbf{x} \|_1 \text{ subject to } \| A\mathbf{x} - \mathbf{y} \|_2 \leq \eta \tag{6.1}$$

with η being an upper bound on $\| \mathbf{e} \|_2$. While we know $\mathbf{x_s}$ will be a feasible solution to QCBP, we don't know if it will still be the optimum. In full generality, it may not be. The following definition and result from Foucart and Rauhut [2013] shows how close QCBP optima can get to $\mathbf{x_s}$:

Definition 10. A matrix $A \in \mathbb{R}^{m\times n}$ satisfies the robust null space property with respect to the norm $\| \cdot \|$ and set $S \subset [n]$ with constants $0 < \rho < 1$ and $\tau > 0$ if

$$\| \mathbf{v}_S \|_1 \leq \rho \| \mathbf{v}_{\bar{S}} \|_1 + \tau \| A\mathbf{v} \| \text{ for all } \mathbf{v} \in \mathbb{R}^n$$

and satisfies the robust null space property of order s with the same constants if the robust null space property holds for A for any $S \subset [N]$ with cardinality at most s.

Note that the robust null space property implies the stable null space property since if $\mathbf{v} \in \ker(A)\backslash\{0\}$, then the $\tau \| A\mathbf{v} \|$ term disappears as $A\mathbf{v} = \mathbf{0}$ by definition of $\ker(A)$. So the robust null space property implies the stable null space property of the same constant ρ.

We then have [Foucart and Rauhut, 2013, thm. 4.20] that A satisfies the robust null space property with constants $0 < \rho < 1$ and $\tau > 0$ relative to $S \subset [n]$ iff

$$\| \mathbf{x} - \mathbf{z} \|_1 \leq \frac{1+\rho}{1-\rho}(\| \mathbf{z} \|_1 - \| \mathbf{x} \|_1 + 2 \| \mathbf{x}_{\bar{S}} \|_1) + \frac{2\tau}{1-\rho} \| A(\mathbf{z} - \mathbf{x}) \|$$

for all vectors $\mathbf{x}$, $\mathbf{z} \in \mathbb{R}^n$. Furthermore, we have the following theorem:

Theorem 11. *Suppose $A \in \mathbb{R}^{m\times n}$ satisfies the robust null space property of order s with constants ρ and τ. For the problem $\mathbf{y} = A\mathbf{x} + \mathbf{e}$ and $\| \mathbf{e} \| \leq \eta$, no matter what $\mathbf{x} \in \mathbb{R}^n$ is, the optimal solution (or solutions) $\mathbf{x}^\sharp$ to* (6.1) *approximate $\mathbf{x}$ to within*

$$\| \mathbf{x} - \mathbf{x}^\sharp \|_1 \quad \leq \quad \frac{2(1+\rho)}{1-\rho}\sigma_s(\mathbf{x})_1 + \frac{4\tau}{1-\rho}\eta$$

Proof. We prove this by letting $\mathbf{z} = \mathbf{x}^\sharp$ and applying theorem 4.20 of Foucart and Rauhut with the robust null space holding for order s, i.e. for all $S \subseteq [N]$ with $| S | \leq s$:

$$\| \mathbf{x} - \mathbf{x}^\sharp \|_1 \quad \leq \quad \frac{1+\rho}{1-\rho}(\| \mathbf{x}^\sharp \|_1 - \| \mathbf{x} \|_1 + 2\sigma_s(\mathbf{x})) + \frac{2\tau}{1-\rho} \| A(\mathbf{x}^\sharp - \mathbf{x}) \|$$

Since $\mathbf{x}^\sharp$ is an ℓ_1 minimum, we have that $\| \mathbf{x}^\sharp \| \leq \| \mathbf{x} \|$, and from the constraint that's part of QCBP, we also have that $\| A\mathbf{x}^\sharp - y \| \leq \eta$. It follows that

$$\begin{aligned}
\| \mathbf{x} - \mathbf{x}^\sharp \|_1 \quad &\leq \quad \frac{1+\rho}{1-\rho}(\| \mathbf{x} \|_1 - \| \mathbf{x} \|_1 + 2\sigma_s(\mathbf{x})) + \frac{2\tau}{1-\rho} \| A(\mathbf{x}^\sharp - \mathbf{x}) \| \\
&= \quad \frac{2(1+\rho)}{1-\rho}\sigma_s(\mathbf{x}) + \frac{2\tau}{1-\rho} \| A\mathbf{x}^\sharp - \mathbf{y} + \mathbf{e} \| \\
&\leq \quad \frac{2(1+\rho)}{1-\rho}\sigma_s(\mathbf{x}) + \frac{2\tau}{1-\rho} \| A\mathbf{x}^\sharp - \mathbf{y} \| + \| \mathbf{e} \| \\
&\leq \quad \frac{2(1+\rho)}{1-\rho}\sigma_s(\mathbf{x}) + \frac{2\tau}{1-\rho}2\eta \\
&= \quad \frac{2(1+\rho)}{1-\rho}\sigma_s(\mathbf{x}) + \frac{4\tau}{1-\rho}\eta
\end{aligned}$$

as desired. □

6.3.3 A tractable approximation to the NSP

Unfortunately, determining whether any given matrix A satisfies the null space property or stable null space property of order s is $coNP$-complete [Tillmann and Pfetsch, 2014]. [7]

This understandably makes it difficult to determine whether we can usefully employ basis pursuit with some given A. Instead we have to use a more computationally practical property. The standard property to use for this purpose is coherence.

Coherence is defined as follows:

Definition 12. Suppose $A \in \mathbb{R}^{m\times n}$ is a matrix where every column vector $\mathbf{a}_i$ are normalized in the ℓ_2-norm. Then the coherence of that matrix is defined as

$$\mu(A) \quad = \quad \max_{1\leq i\neq j\leq n} \mid \langle \mathbf{a}_i, \mathbf{a}_j \rangle \mid$$

i.e. the absolute value of the inner product with greatest such value.

Determining the coherence of a matrix can be done in polynomial time by just trying every combination i, j.

Coherence can both be used to probabilistically bound the number of samples required to reconstruct a signal by using basis pursuit, and to determine sufficient criteria for the ℓ_0 optimum to coincide with the ℓ_1 optimum.

Adcock et al., 2017 refer to a sampling result:

Theorem 13. *Suppose $U \in \mathbb{R}^{n\times n}$ is an isometry defined by $u_{ij} = \langle \varphi_i, \psi_j \rangle$, and suppose $\mathbf{x}$ is s-sparse in the basis given by $\{\phi_j\}_{j=1}^n$. Let the rows of A be m rows from U chosen uniformly at random, and $\mathbf{y} \in \mathbb{R}^m$ be given by $A\mathbf{x} = \mathbf{y}$. Then $\mathbf{x}$ can be recovered from $\mathbf{y}$ with probability exceeding $1 - \epsilon$ if*

$$m \quad \geq \quad C_0 \cdot \mu(U) \cdot n \cdot s \cdot (1 + \log(\frac{1}{\epsilon})) \cdot \log(n)$$

where C_0 is a numerical constant independent of the parameters. [8]

This is useful in a sampling situation where a sensor can be set to record a sample along a specified dimension of the vector space $\psi = \{\psi_j\}_{j=1}^n$. By randomly varying the dimension to sample from, if the matrix U has coherence of order $\frac{1}{n}$, we can recover the target signal (no matter what it is) by taking a number of samples nearly proportional to the sparsity of the target signal. We only have to a constant factor independent of the parameters,

[7] Even determining whether A satisfies the null space property relative to a given set S is tricky. I was unable to find the complexity of solving that problem; we end up getting a set of linear absolute value equations of the form "find $\mathbf{v}$ so that $\sum_{i\in S} \mid v_i \mid + \sum_{i\in \bar{S}} \mid v_i \mid = 1$ subject to $A\mathbf{v} = \mathbf{0}$". The general problem "solve $A\mathbf{x} + B \mid \mathbf{x} \mid = \mathbf{c}$" is NP-complete: see Mangasarian [2007].

[8] The phrasing of this theorem in Adcock et al. is reminiscent of the matrix completion problem we'll investigate later, but I have given it in a cardinality minimization form here.

and a further logarithmic factor $\log(n)$, in excess of the number of observations needed if we knew in advance which elements of $\mathbf{x}$ were zero.[9]

Deterministic bounds linking the NSP and incoherence are weaker, but we have from Fuchs, 2003:

Theorem 14. *If for a vector* $\mathbf{x}$ *satisfying* $A\mathbf{x} = \mathbf{y}$, *with* $A \in \mathbb{R}^{m \times n}$ *composed of* n *column vectors of unit* ℓ_2 *norm, the following holds:*

$$\| \mathbf{x} \|_0 \quad \leq \quad \frac{1}{2}\big(1 + \frac{1}{\mu(A)}\big)$$

then $\mathbf{z}$ *is the unique* ℓ_0 *and* ℓ_1 *minimum and can thus be recovered by solving* (5.1).

Since every such $\mathbf{x}$ must be $\frac{1}{2}(1 + \frac{1}{\mu(A)})$-sparse (by definition of $\| \cdot \|_0$), A must satisfy the null space property of order $\frac{1}{2}(1 + \frac{1}{\mu(A)})$.

6.4 A note on the restricted isometry property

Besides the null space property, the restricted isometry property is frequently used in the compressed sensing literature. As we will later encounter a linear operator analog of this property, I will define the restricted isometry property for vectors here:

Definition 15. Suppose $A \in \mathbb{R}^{m \times n}$ and let $1 \leq s \leq n$. A satisfies the restricted isometry property (or RIP) of order s with constant δ_s if

$$(1 - \delta_s) \| \mathbf{x} \|_2^2 \leq \quad \| A\mathbf{x} \|_2^2 \leq \quad (1 + \delta_s) \| \mathbf{x} \|_2^2$$

for all s-sparse vectors $\mathbf{x} \in \mathbb{R}^n$.

When one refers to just "the restricted isometry constant δ_s", this is usually taken to mean the smallest δ_s so that A satisfies the RIP of order s with that constant. [Tillmann and Pfetsch, 2014]

While the restricted isometry property can be useful for proving other results in compressed sensing, it is not as directly useful. Deciding whether $\delta_s < 1$ for some given $A \in \mathbb{Q}^{m \times n}$ and s is $coNP$-complete [Tillmann and Pfetsch, 2014]. This means that the RIP can't be used as a quick worst-case approximation the way incoherence can be.

For theoretical purposes, the NSP and the RIP each have their benefits and drawbacks. Given any matrix A with $\delta_s = k$ for some s and k, it's possible to construct another matrix with the same null space, but no longer satisfying the RIP with that constant. On the other hand, RIP can be used to give bounds on robust recovery that don't depend on the magnitude of the true solution $\| \mathbf{x} \|$; see Cahill et al., 2016.

[9] A drawback of this analysis is that some basis matrices are coherent. E.g. in MRI with $U = U_{dft} V_{dwt}^{-1}$ modeling sparsity in the wavelet transform V_{dwt} domain based on observations in the discrete Fourier domain U_{dft}, U has constant coherence. By changing the distribution of the random draws from U, it's possible to circumvent this problem in many cases; see Adcock et al. for details.

7 Other convex relaxations of robust recovery

Before we extend the compressed sensing framework to matrix recovery, we'll note other convex relaxations for performing robust recovery. These are useful for algorithmic purposes, e.g. if the solver being used only solves unconstrained problems.

We've already seen quadratically constrained basis pursuit:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \| \mathbf{x} \|_1 \text{ subject to } \| A\mathbf{x} - \mathbf{y} \|_2 \leq \eta$$

In statistics, we may encounter the lasso [Tibshirani, 1996]:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \| A\mathbf{x} - \mathbf{y} \|_2 \text{ s.t. } \| \mathbf{x} \|_1 \leq t$$

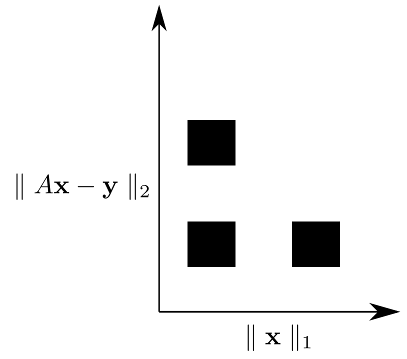
with $\lambda > 0$. Adopting a Lagrangian approach to the lasso gives us basis pursuit denoising or BPDN [Chen et al., 2001]:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \lambda \| \mathbf{x} \|_1 + \frac{1}{2} \| A\mathbf{x} - \mathbf{y} \|_2^2$$

7.1 Relations between robust recovery methods

The three formulations are equivalent in the sense that for each formulation's solution of a particular problem, if there's a unique solution, there exist parameter choices for the other two formulations so that the unique solution is part of their solution set (though possibly not unique).

However, this equivalence may fail to hold for solution sets as a whole. We can motivate the relationship with an illustration:



Suppose the squares indicate the three possible solutions to $A\mathbf{x} = \mathbf{y}$ for some given A and $\mathbf{y}$. The axis arrows point in the direction of increasing values. Since QCBP's objective value only depends on $\| \mathbf{x} \|_1$, its set of optimal solutions will either include at least the two solutions to the left, or will be empty. Similarly, the lasso's objective value depends only on $\| A\mathbf{x} - \mathbf{y} \|_2$ and so the set of optimal solutions is either empty or contains at least the two bottom solutions of the illustration.

The BPDN objective function considers all solutions lying along a line to be equally good, where the slope of the line depends on λ. For finite positive values of λ, BPDN will pick the solution at the lower left of the above illustration since its objective function will always consider moving closer to the origin along both axes to be an improvement.

Formally, we have the following relations:

- If $\mathbf{x}$ is an optimal solution to BPDN for some $\lambda > 0$, then there exists a $\eta \geq 0$ so that $\mathbf{x}$ is also an optimal solution to QCBP.
- If $\mathbf{x}$ is a unique optimal solution to QCBP for some $\eta \geq 0$, then there exists a $\tau \geq 0$ so that $\mathbf{x}$ is also the unique optimal solution to the Lasso.
- If $\mathbf{x}$ is an optimal solution to Lasso for some $\tau > 0$, then there exists a $\lambda \geq 0$ so that $\mathbf{x}$ is also an optimal solution to BPDN.

I'll prove the first two of these relations and give a brief overview of the proof to the third, since proving it fully requires machinery from duality theory. The proofs are due to Foucart and Rauhut [2013, prop 3.2].[10]

Basis pursuit denoising $\Rightarrow$ quadratically constrained basis pursuit Let $\mathbf{x}^\sharp$ be an optimal solution to basis pursuit denoising for some $\lambda > 0$. Let $\eta_\sharp = \| A\mathbf{x}^\sharp - \mathbf{y} \|_2$ and consider some other candidate solution $\mathbf{z} \neq \mathbf{x}^\sharp$ to QCBP with $\eta = \eta_\sharp$. Since $\mathbf{x}^\sharp$ is the optimal solution to BPDN, we have that

$$\lambda \| \mathbf{x}^\sharp \|_1 + \frac{1}{2} \| A\mathbf{x}^\sharp - \mathbf{y} \|_2^2 \quad \leq \quad \lambda \| \mathbf{z} \|_1 + \frac{1}{2} \| A\mathbf{z} - \mathbf{y} \|_2^2$$

Since $\mathbf{z}$ is a candidate solution to QCBP, we also have

$$\| A\mathbf{z} - \mathbf{y} \|_2^2 \quad \leq \quad \| A\mathbf{x}^\sharp - \mathbf{y} \|_2^2$$

hence

$$\begin{aligned} \lambda \| \mathbf{z} \|_1 + \frac{1}{2} \| A\mathbf{z} - \mathbf{y} \|_2^2 \quad &\leq \quad \lambda \| \mathbf{z} \|_1 + \frac{1}{2} \| A\mathbf{x}^\sharp - \mathbf{y} \|_2^2 \\ \lambda \| \mathbf{x}^\sharp \|_1 + \frac{1}{2} \| A\mathbf{x}^\sharp - \mathbf{y} \|_2^2 \quad &\leq \quad \lambda \| \mathbf{z} \|_1 + \frac{1}{2} \| A\mathbf{x}^\sharp - \mathbf{y} \|_2^2 \\ \| \mathbf{x}^\sharp \|_1 \quad &\leq \quad \| \mathbf{z} \|_1 \end{aligned}$$

so $\mathbf{x}^\sharp$ is an optimal solution to QCBP.

[10]Note that the authors omit the scaling factor of $\frac{1}{2}$ in their definition of basis pursuit denoising.

Unique QCBP solution $\Rightarrow$ unique Lasso solution Let $\mathbf{x}^\sharp$ be the optimal solution to QCBP for some η. Let $t_\sharp = \| \mathbf{x}^\sharp \|_1$, and consider another candidate vector $\mathbf{z} \neq \mathbf{x}^\sharp$. For $\mathbf{z}$ to be a solution to the Lasso with $t = t_\sharp$, $\| \mathbf{z} \|_1 \leq \| \mathbf{x}^\sharp \|_1$. Since $\mathbf{x}^\sharp$ is the unique optimum of QCBP, $\| A\mathbf{z} - \mathbf{y} \|_2 > \| A\mathbf{x} - \mathbf{y} \|_2$ (otherwise $\mathbf{z}$ would also have been an optimal solution to QCBP). Thus $\mathbf{x}^\sharp$ is also the unique optimal solution to the Lasso.

Lasso $\Rightarrow$ BPDN The Lasso is equivalent to

$$\min \| A\mathbf{x} - \mathbf{y} \|_2^2 \text{ s.t. } \| \mathbf{x} \|_2 \leq \tau \tag{7.1}$$

since any optimal solution set $\mathbf{X}$ to the Lasso can be turned into the optimal solution set to this problem by letting $\tau = \max_{\mathbf{x} \in \mathbf{X}} \| \mathbf{x} \|_2$.

The Lagrangian form of (7.1) is

$$\min \| A\mathbf{x} - \mathbf{y} \|_2^2 + \xi(\| \mathbf{x} \|_2 - \tau) \tag{7.2}$$

This problem exhibits strong duality, so there exists a dual optimal $\xi^\sharp \geq 0$ for this problem that gives the same optimal solution set as (7.1) with a given τ.

By expanding terms and removing the resulting constant $-\xi\tau$, we can rewrite (7.2) as

$$\min \| A\mathbf{x} - \mathbf{y} \|_2^2 + \xi \| \mathbf{x} \|_2 \tag{7.3}$$

This is equivalent to BPDN by a similar trick as for (7.1): if $\mathbf{X}$ is the optimal solution set for 7.3, then setting $t = \max_{\mathbf{x} \in \mathbf{X}} \| \mathbf{x} \|_1$ will suffice to make $\mathbf{X}$ the solution set for BPDN.

Part II

Matrix sensing and completion

8 Generalizing compressed sensing to unknown matrices

The matrix sensing and completion problems are generalizations of the cardinality minimization problem which the field of compressed sensing started off investigating. While the cardinality minimization problem,

$$\min_{\mathbf{x}} \quad \| \mathbf{x} \|_0 \quad \text{subject to } A\mathbf{x} = \mathbf{y}$$

involves recovering a vector, matrix sensing and completion involve recovering a matrix.

The difference between matrix sensing and completion problems is that matrix sensing considers a fully general linear operator $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$, giving the following problem:

$$\min_X \operatorname{rank}(X) \,\text{s.t.}\, \mathcal{A}(X) = \mathcal{A}(Y) \tag{8.1}$$

for a given matrix $Y \in \mathbb{R}^{m\times n}$.

The matrix completion problem replaces $\mathcal{A}$ with a projection operator $P_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$. It is defined based on a set Ω, as follows:

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

The matrix completion problem is otherwise similar:

$$\min_X \quad \operatorname{rank}(X) \quad \text{s.t. } P_\Omega(X) = P_\Omega(Y)$$

As is the case for cardinality minimization, both matrix sensing and completion have corresponding decision problems, namely

$$\begin{array}{c}\text{given } \mathcal{A} : \mathbb{Q}^{m\times n} \to \mathbb{Q}^p,\, Y \in \mathbb{Q}^{m\times n},\, r \in N \\ \text{does there exist an } X \in \mathbb{Q}^{m\times n}, \text{,rank(X)} \le r \text{ so that} \quad \mathcal{A}(X) = \mathcal{A}(Y)?\end{array} \tag{8.2}$$

for matrix sensing and

$$\begin{array}{c}\text{given } P_\Omega : \mathbb{Q}^{m\times n} \to \mathbb{Q}^{m\times n},\, Y \in \mathbb{Q}^{m\times n},\, r \in N \\ \text{does there exist an } X \in \mathbb{Q}^{m\times n}, \text{,rank(X)} \le r \text{ so that} \quad P_\Omega(X) = P_\Omega(Y)?\end{array} \tag{8.3}$$

for matrix completion, with maximum rank r given as an additional input to the problem.

Without additional restrictions on Y or $\mathcal{A}$, matrix completion is a subset of matrix sensing, because for any P_Ω, one can easily create a flattened linear operator $\mathcal{A}_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\cdot n}$ by letting $\mathcal{A}_\Omega(X)_{an+b} = P_\Omega(X)_{a,b}$. However, in practice, the setting differs: in matrix sensing, the structure of $\mathcal{A}$ is usually known in advance, whereas in matrix completion, Ω might not be known in advance, or may be random. We'll later be considering matrix completion in the context of recommender systems, where Ω contains the indices corresponding to known observations, and thus depends upon user behavior.

Note that the (in some sense) degenerate case of matrix completion with all entries of Y known is easy to solve: we can find the optimal rank k approximation to Y by decomposing Y into $U\Sigma V^T$ by the SVD, setting the k-1 smallest diagonal values of Σ to 0, thus getting Σ_T (for *truncated*) and letting $X = U\Sigma_T V^T$.

Let us next determine whether minimum rank matrix sensing is hard to solve.

9 Hardness results

9.1 Matrix sensing

Just as (3.3) is NP-complete, it also turns out that (8.2) is. In addition, the way this is shown indicates where cardinality minimization is hiding in the matrix sensing problem.

To show that the decision problem is NP-complete, we need to show that it is in NP and that it is NP-hard.

It's relatively easy to show that (8.2) is in NP: if we're given a candidate matrix $\hat{X}$, it's possible to verify the rank of $\hat{X}$ in polynomial time (e.g. by a singular value decomposition), and it's also possible to calculate $\mathcal{A}(\hat{X}) - \mathcal{A}(Y)$ in polynomial time.

Showing that (8.2) is NP-hard can be done by a reduction from cardinality minimization. I will use functions that operate on $\mathbb{R}$ for this reduction even though the decision problems are defined on $\mathbb{Q}$ to show that the reduction is not an artifact of choosing $\mathbb{Q}$ as the domain for the decision problems.

The reduction goes as follows:

Let $A = (\mathbf{a}_1, \ldots, \mathbf{a}_n)$, k, $\mathbf{y}$ be the inputs to the cardinality minimization decision problem, and define the diagonal functions as follows:

$$\begin{aligned} \operatorname{diag}(X) : \mathbb{R}^{n\times n} \to \mathbb{R}^n \quad &= \quad (X_{11}, \ldots, X_{nn}) \\ \operatorname{diag}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^{n\times n} \quad \text{so that} \quad &\operatorname{diag}(\mathbf{x})_{ij} = \begin{cases} x_i & i = j \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Let $Y = \operatorname{diag}(\mathbf{y})$, i.e. the $n \times n$ matrix with $\mathbf{y}$ along the diagonal, and define the linear mapping $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\cdot n}$ as

$$\mathcal{A}(X)_{in+j} \quad = \quad \begin{cases} A \operatorname{diag}(X)_i & a = b \\ X_{ij} & \text{otherwise} \end{cases}$$

Then we want to show that we have $X \in \mathbb{R}^{nxn}$ of rank $\leq$ k with $\mathcal{A}(X) = Y$ iff $\parallel \mathbf{x} \parallel_0 \leq k$ subject to $A\mathbf{x} = \mathbf{y}$.

Suppose that X is a solution to (8.2) with $r = k$. Then by the definition of the decision problem, $\operatorname{rank}(X) \leq k$ and $\mathcal{A}(X) = Y$. Since Y is a diagonal matrix and $\mathcal{A}$ preserves off-diagonal elements of X, X must also be a diagonal matrix. Hence $\mathcal{A}(X) = Y$ is equivalent to $A\mathbf{x} = \mathbf{y}$ for $\mathbf{y}$ defined above and $\mathbf{x} = \operatorname{diag}(X)$. Since the rank of a diagonal matrix is equal to the number of non-zero elements along the diagonal, $\operatorname{rank}(X) \leq k$ is equivalent to $\parallel \mathbf{x} \parallel_0 \leq k$. We thus have a solution to the vector decision problem

$$\text{does there exist an}\,\mathbf{x}\,\text{so that} \quad \parallel \mathbf{x} \parallel_0 \leq k,\ A\mathbf{x} = \mathbf{y}?$$

since $\mathbf{x}$ satisfies the criteria.

The other direction is analogous: Suppose $\mathbf{x}$ is a solution to the vector decision problem with $r = k$. Then $X = \operatorname{diag}(\mathbf{x})$ has rank $\leq k$ due to the aforementioned equivalence between rank and zero norm. Since all off-diagonal elements of X are zero and $A\mathbf{x} = \mathbf{y}$, $\mathcal{A}(X) = Y$ for $Y = \operatorname{diag}(\mathbf{y})$ given.

This implies that solving (8.2) is at least as hard as solving (3.3), and since the latter is NP-hard, so is the former.

Intuition of the result I'll later give results that correspond to the null space property results for cardinality minimization, but where the singular values of all matrices $M \in \ker \mathcal{A}$ take the place of the values of all vectors $\mathbf{x} \in \ker A$ of the results in compressed sensing. On an intuitive level, then, matrix sensing consists of finding some solution to $\mathcal{A}(X) = \mathcal{A}(Y)$ and then adding a weighted combination of matrices from $\ker \mathcal{A}$ so as to minimize the rank of the solution. The cardinality minimization problem is embedded in this problem since a low rank matrix necessarily must have the vector of its singular values be minimal cardinality among all possible solutions.

In the case of the reduction above, the analogy is more clear, since all diagonal matrices have SVD with U and V being permutation matrices. Thus the singular value vector $\operatorname{diag}(\Sigma)$ is simply $\mathbf{x}$ in the vector compressed sensing problem, albeit with its elements rearranged.

9.2 Matrix completion

Proving that matrix sensing is at least as hard as cardinality minimization still leaves open the possibility that the more restricted matrix completion problem would be exempt. The proof as shown does not apply to matrix completion since the construction of P_Ω does not allow any multiplication by the matrix A, which we need to perform the reduction.

However, Gillis and Glineur [2011] give references to a proof that matrix completion is NP-hard, and provide proofs of their own for robust recovery generalizations of matrix sensing and completion.

10 Convex relaxation

In the previous part of the thesis, we found that ℓ_0 optimization was NP-hard. We then examined the closest convex relaxation, namely basis pursuit, finding that we could still recover the ℓ_0 optimum in certain cases.

Since we've determined that the minimum rank optimization problem is NP-hard, it seems natural to attempt the same strategy here.

To do so, we first need to find two matrix norms where the first gives the rank of the matrix, and the second, a convex relaxation of the first, gives the ℓ_1 norm of the diagonal vector when given a diagonal matrix. Then we can use the former norm where we used the ℓ_0 norm in compressed sensing, and the latter norm where we used the ℓ_1 norm, and see where that gets us.

Motivated by the hardness reduction in subsection 9.1, we can rewrite (8.1) as

$$\min \|\text{diag}(\Sigma)\ \|_0 \text{ subject to } \mathcal{A}(X) = \mathcal{A}(Y)$$

where Σ is the diagonal matrix in the singular value decomposition $X = U\Sigma V^T$.

More generally, define

$$\| A \|_{S_p} = \Big(\sum^{min(m,n)} \sigma_i(A)^p \Big)^{\frac{1}{p}} \tag{10.1}$$

the Schatten p-norm on a matrix $A \in \mathbb{R}^{mxn}$, where σ_i is the i-th singular value. Then let (in a similar nonstandard use to the vector case) $\| A \|_0 = \lim_{k \to 0} \| A \|_{S_k}$.

The p-norm matrix sensing problem

$$\min \|\text{X}\|_{S_p} \text{ subject to } \mathcal{A}(X) = \mathcal{A}(Y)$$

contains the ℓ_p optimization problem

$$\min \| \mathbf{x} \|_p \text{ subject to } A\mathbf{x} = \mathbf{y}$$

as a special case, so the former is at least as hard as the latter, complexity wise. This can be shown by an analogous argument to the hardness result. The singular values of a diagonal matrix consists of the absolute values of the vector on the diagonal, and the choice of $\mathcal{A}$ in the hardness result forces all candidate matrices to be diagonal.

It is thus natural to consider the following convex relaxation of (8.1):

$$\min \|\text{X}\|_* \text{ subject to } \mathcal{A}(X) = Y \tag{10.2}$$

where $\| X \|_*$ is the nuclear norm, equivalent to the p-Schatten norm with $p = 1$. As long as we're dealing with actual norms, which is the case for $p \geq 1$, convexity follows from the triangle inequality. Thus we know that the nuclear norm minimization problem is convex.

Another way of seeing that the nuclear norm minimization problem is convex is to express it as a semidefinite programming problem; if this is possible, the problem is convex since semidefinite programs are convex. Furthermore, a semidefinite program gives a way to solve the problem, albeit with prohibitive runtime for anything beyond small problem instances.

Theorem 16. *The following semidefinite programming problem, due to Vandenberghe and Boyd [1996, p. 66], is equivalent to* (10.2)*:*

$$\min_{X \in \mathbb{R}^{mxn},\, Y \in \mathbb{R}^{mxm},\, Z \in \mathbb{R}^{nxn}} \quad Trace(Y) + Trace(Z) \quad \textit{subject to } \mathcal{A}(X) = \mathcal{A}(Y) \textit{ and } \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \succcurlyeq 0$$

Here $A \succcurlyeq 0$ means $A \in \mathbb{R}^{pxp}$ for some p is positive semidefinite, i.e. that $\mathbf{x}^* A \mathbf{x} \geq 0$ for all $\mathbf{x} \in \mathbb{C}^p$.

10.1 Proof of SDP formulation

The proof strategy broadly follows Vandenberghe and Boyd.

For simplicity's sake, let's disregard the constraint $\mathcal{A}(X) = \mathcal{A}(Y)$ for the time being. Our objective is then to prove that

$$\min_{X\in\mathbb{R}^{mxn},\, Y\in\mathbb{R}^{mxm},\, Z\in\mathbb{R}^{nxn}} \quad Trace(Y) + Trace(Z) \qquad \text{subject to} \quad \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \succcurlyeq 0$$

is equivalent to

$$\min \parallel X \parallel_*$$

This is a strictly more general claim and so if we can prove it for all X, we automatically have that it also holds for all X where $\mathcal{A}(X) = \mathcal{A}(Y)$.

The next step of our strategy is to show that for every matrix $X \in \mathbb{R}^{mxn}$, there exist matrices Y, Z so that $\begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \succcurlyeq 0$ and $Trace(Y) + Trace(Z) = \alpha$ iff $\parallel X \parallel_* \leq \frac{\alpha}{2}$. If we can show this, then any decrease in the trace term implies a decrease in the nuclear norm. Thus minimizing the trace terms under the constraints given is equivalent to minimizing the nuclear norm of X.

To prove the claim, we need to prove the implication in both directions. We first start with

$$\exists Y, Z : \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \succcurlyeq 0 \,\text{and}\, Trace(Y) + Trace(Z) = \alpha \quad \Rightarrow \quad \parallel X \parallel_* \leq \frac{\alpha}{2}$$

Let the singular value decomposition of X be given by $X = U\Sigma V^T$. Σ is a diagonal $p \times p$ matrix, where $p = \min(m, n)$. Furthermore, $Trace(\Sigma) = \parallel X \parallel_*$.

Furthermore let

$$\begin{aligned} W &= \begin{bmatrix} UU^T & -UV^T \\ -VU^T & VV^T \end{bmatrix} \\ P &= \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \\ A &= \begin{bmatrix} U^T & -V^T \end{bmatrix} \end{aligned}$$

W is positive semidefinite since $W = A^T A$, and P is positive semidefinite by constraint. We thus have that $Trace(W) \geq 0$ and $Trace(P) \geq 0$, and we would like to show that $Trace(WP) \geq 0$ follows.

To do so, we need an auxiliary result, that $Trace(A) \geq 0$ and $Trace(B) \geq 0$ implies $Trace(AB) \geq 0$ for any positive semidefinite matrices A, B. Let L_A and L_B be the Cholesky decomposition of A and B respectively. Since A and B are both positive semidefinite, such decompositions exist. Then we have

$$\begin{aligned} trace(AB) &= Trace(L_A L_A^T L_B L_B^T) \\ &= Trace(L_B^T L_A L_A^T L_B) \\ &= Trace((L_A^T L_B)^T (L_A^T L_B)) \end{aligned}$$

which is on the form $C^T C$ with $C = L_A^T L_B$. Thus $trace(AB) \geq 0$ as well, and in particular, $trace(WP) \geq 0$. Now we have that

$$\begin{aligned} WP &= \begin{bmatrix} UU^T & -UV^T \\ -VU^T & VV^T \end{bmatrix} \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \\ &= \begin{bmatrix} UU^TY - UV^TX^T & UU^TX - UV^TZ \\ -VU^TY - VV^TX^T & -VU^TX + VV^TZ \end{bmatrix} \end{aligned}$$

and since the trace of a matrix is the sum of its diagonal elements, $Trace(WP) \geq 0$ implies $Trace(UU^TY - UV^TX^T) + Trace(-VU^TX + VV^TZ) \geq 0$. Rewriting,

$$\begin{aligned} & Trace(UU^TY - UV^TX^T) + \\ & \quad Trace(-VU^TX + VV^TZ) = Trace(UU^TY) - Trace(UV^TX^T) \\ & \qquad -Trace(VU^TX) + Trace(VV^TZ) \\ & \quad = Trace(UU^TY) + Trace(VV^TZ) - 2 \cdot Trace(UV^TX^T) \\ & \quad \geq 0 \end{aligned}$$

and we can simplify further by

$$\begin{aligned} Trace(UV^TX^T) &= Trace(UV^T(U\Sigma V^T)^T) \\ &= Trace(UV^TV\Sigma U^T) \\ &= Trace(U\Sigma U^T) \\ &= Trace(\Sigma) \end{aligned}$$

so we have

$$Trace(UU^TY) + Trace(VV^TZ) - 2 \cdot Trace(\Sigma) \quad \geq \quad 0$$

Since UU^T is a projection, $Trace(I - UU^T) \geq 0$ and so $Trace((I - UU^T)Y) \geq 0$. By the constraint of the SDP, P is positive semidefinite, which implies $Trace(Y) \geq 0$. Hence

$$\begin{aligned} Trace(Y - UU^TY) &\geq 0 \\ Trace(Y) - Trace(UU^TY) &\geq 0 \\ Trace(Y) &\geq Trace(UU^TY) \end{aligned}$$

so we can further simplify to

$$Trace(Y) + Trace(Z) - 2 \cdot Trace(\Sigma) \quad \geq \quad 0$$

(showing $Trace(Z) \geq Trace(UU^TZ)$ is completely analogous). Finally, making use of $Trace(Y) + trace(Z) = \alpha$ and $Trace(\Sigma) = \parallel X \parallel_*$, we get

$$\alpha - 2 \parallel X \parallel_* \quad \geq \quad 0$$

i.e.

$$\parallel X \parallel_* \quad \leq \quad \frac{\alpha}{2}$$

as desired.

In the other direction, we want to show that

$$\parallel X \parallel_* \leq \frac{\alpha}{2} \quad \Rightarrow \quad \exists Y, Z : \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} \succcurlyeq 0 \,\text{and}\, Trace(Y) + Trace(Z) = \alpha$$

Let $X = U\Sigma V^T$ be the SVD of X as before, and

$$\begin{aligned} \gamma &= \frac{\alpha - 2 \parallel X \parallel_*}{n + m} \\ Y &= U\Sigma U^T + \gamma I \\ Z &= V\Sigma V^T + \gamma I \end{aligned}$$

Note that

$$\begin{aligned} \gamma &\geq \frac{\alpha - 2\frac{\alpha}{2}}{n + m} \\ &= 0 \end{aligned}$$

so

$$\begin{aligned} Trace(Y) + Trace(Z) &= 2 \cdot Trace(\Sigma) + (n+m)\gamma \\ &= 2 \parallel X \parallel_* + \alpha - 2 \parallel X \parallel_* \\ &= \alpha \end{aligned}$$

satisfying the second constraint.

Furthermore, we have that

$$\begin{aligned} \begin{bmatrix} Y & X \\ X^T & Z \end{bmatrix} &= \begin{bmatrix} U\Sigma U^T + \gamma I & X \\ X^T & V\Sigma V^T + \gamma I \end{bmatrix} \\ &= \begin{bmatrix} U\Sigma U^T & U\Sigma V^T \\ V\Sigma U^T & V\Sigma V^T \end{bmatrix} + \gamma I \\ &= \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma \end{bmatrix} \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} + \gamma I \\ &\succcurlyeq 0 \end{aligned}$$

The last step holds since we have a sum of two positive semidefinite matrices. $\begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma \end{bmatrix} \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix}$ is of the form $M^T D M$ with D a diagonal matrix of nonnegative entries, and so positive semidefinite, and since $\gamma \geq 0$, γI is positive semidefinite as well.

11 Matrix sensing problems and results

We have determined a close analog and generalization to the norm minimization problem of compressed sensing, which reduces to the vector problem for diagonal matrices. The next thing to do is to find out when we can recover the hard problem (low rank recovery) by using the easy problem (nuclear norm minimization).

We proceed in a manner analogous to the vector recovery case, first considering situations where the desired matrix X is exactly low rank, and then when it is approximately low rank. We then consider the case where we cannot determine the linear operator $\mathcal{A}$ ahead of time, which is of interest for recommender systems where $\mathcal{A}$ depends on which items have been rated by which users.

11.1 Bounds on recovery

For $\mathcal{A}(X)$ and Y known, we have the following bounds:

11.1.1 Recovery of exactly low rank matrices

Here I provide two results that generalize the results for compressed sensing on vectors. The first uses a null space property for linear operators analogous to the null space property for matrices in the vector case, while the second uses the spherical section property.

By the null space property We first need to define the null space.
Definition 17. For a linear operator $\mathcal{A}$, define its null space as

$$\ker(\mathcal{A}) \quad = \quad \{X : \mathcal{A}(X) = \mathbf{0}\}$$

where $\mathbf{0} \in \mathbb{R}^p$ is the zero vector.

Oymak and Hassibi [2010] give a generalization of the null space property to the case of matrix sensing. The result is for square matrices, but can easily be extended to rectangular ones to give the following theorem:
Theorem 18. *Let $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ be a linear map. Any matrix X of rank at most r that satisfies $\mathcal{A}(X) = \mathcal{A}(Y)$ is the unique optimal solution to* (8.1) *for that choice of $\mathcal{A}$ and Y iff, for all $Z \in \ker(\mathcal{A}) \setminus \{0\}$, its singular values $\sigma_i(Z)$ satisfy*

$$\sum_{i=1}^{r} \sigma_i(Z) \quad < \quad \sum_{i=r+1}^{N} \sigma_i(Z)$$

or equivalently

$$2\sum_{i=1}^{r} \sigma_i(Z) \quad < \quad \sum_{i=1}^{N} \sigma_i(Z)$$

where $N = \min(m, n)$.

Proof. Suppose the property above holds and the optimal solution X_0 has rank $\leq r$. Consider a possible other solution X_1 that satisfies $\mathcal{A}(X) = Y$, $X_0 \neq X_1$. Since $rank(X_0) \leq r$, $\sigma_i(X_0) = 0$ for all $i > r$. Let $X_0 - X_1 = W \in \ker(\mathcal{A}) \setminus \{0\}$.

From lemma A.18 in Foucart and Rauhut [2013], we have that

$$\sum_{i=1}^{k} \sigma_i(X - Y) \quad \geq \quad \sum_{i=1}^{k} \mid \sigma_i(X) - \sigma_i(Y) \mid$$

for any $X \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{m\times n}$ and $k \leq \min(m, n)$, which in our case implies

$$\parallel X_0 + W \parallel_* \quad \geq \quad \sum_{i=1}^{N} \mid \sigma_i(X_0) - \sigma_i(W) \mid$$

It then follows that

$$\begin{aligned}
\sum_{i=1}^{N} \mid \sigma_i(X_0) - \sigma_i(W) \mid &\geq \sum_{i=1}^{r} \big(\sigma_i(X_0) - \sigma_i(W)\big) + \sum_{i=r+1}^{N} \mid \sigma_i(X_0) - \sigma_i(W) \mid \\
&= \sum_{i=1}^{r} \big(\sigma_i(X_0) - \sigma_i(W)\big) + \sum_{i=r+1}^{N} \sigma_i(W) \\
&= \sum_{i=1}^{n} \sigma_i(X_0) + \sum_{i=1}^{n} \sigma_i(W) - 2\sum_{i=1}^{r} \sigma_i(W) + \sigma_r(W) \\
&\geq \parallel X_0 \parallel_* + \parallel W \parallel_* -2\sum_{i=1}^{r} \sigma_i(W) \\
&> \parallel X_0 \parallel_*
\end{aligned}$$

Thus $\parallel X_1 \parallel_* = \parallel X + W \parallel_* > \parallel X_0 \parallel_*$. Since X_1 is an arbitrary solution to $\mathcal{A}(X) = Y$ different from X_0, X_0 must be the unique optimal solution to (10.2).

In the other direction, suppose the property above isn't satisfied, i.e. that there exists at least one $W \in \ker(\mathcal{A}) \setminus \{0\}$ so that

$$\sum_{i=1}^{r} \sigma_i(W) \quad \geq \quad \sum_{i=r+1}^{N} \sigma_i(W) \tag{11.1}$$

Choose Y in (10.2) so that $X = -W_r$ satisfies $\mathcal{A}(X) = Y$, where W_r is the matrix produced by setting all but the largest r singular values of W to zero. Then $X + W = W - W_r$ has only the r largest singular values set to zero, and because $W \in \ker(\mathcal{A}) \setminus \{0\}$, $X + W$ is also a solution. We have

$$\begin{aligned}
\parallel X + W \parallel_* &= \sum_{i=r+1}^{N} \sigma_i(W) \\
\parallel X \parallel_* &= \sum_{i=1}^{r} \sigma_i(W)
\end{aligned}$$

and (11.1) implies

$$\parallel X + W \parallel_* \quad \leq \quad \parallel X \parallel_*$$

even though $rank(X) \leq r \leq rank(X + W)$. Thus either (10.2) has both $X + W$ and X as optimal solutions, or only $X + W$. In neither case is X the unique optimal solution. $\square$

By the spherical section property This generalization follows Dvijotham and Fazel [2010] and Oymak et al. [2011].

Define the null space of $\mathcal{A}$ as above, and the spherical section property as follows:

Definition 19. $\ker(\mathcal{A})$ satisfies the spherical section property with constant Δ if

$$\frac{\| Z \|_*}{\| Z \|_F} \geq \sqrt{\Delta} \, \forall \, Z \in \ker(\mathcal{A}) \setminus \mathbf{0}$$

Then theorem 2.1 of Dvijotham and Fazel [2010] states that if $\ker(\mathcal{A})$ satisfies the spherical section property with constant Δ, then X_0 is the unique optimal solution to (8.1) if $\mathcal{A}(X_0) = Y$ and $rank(X_0) < \frac{\Delta}{2}$.

Proof. Suppose for contradiction that there is another solution, X_1 to (8.1), and $X_0 \neq X_1$, $rank(X_1) \leq rank(X_0) < \frac{\Delta}{2}$. Let

$$Z = X_0 - X_1$$

and since $\mathcal{A}(X_0) = \mathcal{A}(X_1) = Y$, we have $Z \in \ker(\mathcal{A}) \setminus 0$.

By the spherical section property,

$$\| Z \|_* \geq \| Z \|_F \sqrt{\Delta}$$

and for all real matrices A we have

$$\sqrt{rank(A)} \cdot \| A \|_F \geq \| A \|_* \geq \| A \|_F$$

which implies

$$\| Z \|_F \geq \frac{1}{\sqrt{rank(Z)}} \| Z \|_*$$

or

$$\| Z \|_F \sqrt{\Delta} \geq \frac{\sqrt{\Delta}}{\sqrt{rank(Z)}} \| Z \|_*$$

hence

$$\begin{aligned} \| Z \|_* &\geq \frac{\sqrt{\Delta}}{\sqrt{rank(Z)}} \| Z \|_* \\ 1 &\geq \frac{\sqrt{\Delta}}{\sqrt{rank(Z)}} \\ rank(Z) &\geq \Delta \end{aligned}$$

However, we also have that

$$\begin{aligned} rank(Z) &\leq rank(X_0) + rank(X_1) \\ &< \Delta \end{aligned}$$

hence a contradiction. So no such X_1 can exist. □

From Oymak et al. [2011] we further have that if $\mathcal{A}(X_0) = Y$ and $rank(X_0) < \frac{\Delta}{4}$, X_0 is the only optimal solution to (10.2). See the paper for proof.

Together, these results imply that if $\ker(\mathcal{A})$ satisfies the spherical section property with constant Δ, then if there exists at least one X for which $\mathcal{A}(X) = Y$ and $rank(X) < \frac{\Delta}{2}$, then this X is the unique optimal solution to the low rank matrix recovery problem, and if we furthermore have $rank(X) < \frac{\Delta}{4}$, then nuclear norm minimization will recover that X.

11.1.2 Recovery of approximately low rank matrices

By the null space property Foucart and Rauhut [2013] provide a stable generalization of the matrix null space property.

Definition 20. A linear operator $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ satisfies the stable rank null space property of order r with constant $0 < \rho < 1$ if

$$\sum_{i=1}^{r} \sigma_i(M) \leq \quad \rho \textstyle\sum_{i=r+1}^{N} \sigma_i(M) \quad \forall M \in \ker(\mathcal{A})\backslash\{0\}$$

Note the similarity of the stable null space property for matrices with that for vectors:

$$\| \mathbf{v}_S \|_1 < \rho \| \mathbf{v}_{\bar{S}} \|_1 \qquad \forall \mathbf{v} \in \ker(\mathrm{A})\backslash\{0\}$$

Since the convention for singular values is that they are sorted by magnitude (i.e. $k < j \Rightarrow \sigma_k \geq \sigma_j$), we don't need to consider every possible index set S as in the vector case: it's sufficient to consider the case where $S = 1, \dots, r$ for order r.

If the stable rank null space property holds, we have a bound on the distance of every nuclear norm minimization optimum $X^\sharp$ from every other matrix X satisfying $\mathcal{A}(X) = \mathcal{A}(Y)$. Since the minimum rank matrix satisfying $\mathcal{A}(X) = \mathcal{A}(Y)$ is also one of those, the result gives a bound on the distance between the nuclear norm minimum and the minimum rank solution.

Theorem 21. *If a linear operator $\mathcal{A}$ satisfies the stable rank null space property of order r with constant ρ, then for every nuclear norm minimization optimum $X^\sharp$ and every matrix X satisfying $\mathcal{A}(X) = \mathcal{A}(Y)$,*

$$\| X - X^\sharp \|_* \quad \leq \quad \frac{2(1+\rho)}{1-\rho} \sum_{i=r+1}^{N} \sigma_i(X)$$

The bound is essentially the same as in Theorem 9, with $\| \cdot \|_*$ taking the place of $\| \cdot \|_1$, and $\sum_{i=r+1}^{N} \sigma_i(X)$ taking the place of $\sigma_s(\mathbf{x})_1$. The proof is similar as well; lemma A.18 in Foucart and Rauhut makes it possible to use the same strategy.

11.2 Other nuclear norm problems

In a similar way to how we generalized basis pursuit in section Section 8, we can generalize the nuclear norm minimization problem (8.1). These generalizations are necessary to handle robust recovery problems where the output $\mathcal{A}(Y)$ may be contaminated by additive noise of bounded magnitude. They can also be used to balance between multiple criteria of consideration, like simplicity versus accuracy.

We can generalize the problems as follows, for some vector norm $\| \cdot \|$:

- The quadratically constrained basis pursuit analog:

$$\min \parallel X \parallel_* \quad \text{subject to} \quad \parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel \leq \delta$$

- The lasso analog:

$$\min \parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel \quad \text{subject to} \quad \parallel X \parallel_* \leq t$$

- The basis pursuit denoising analog:

$$\min \lambda \parallel X \parallel_* \; + \; \frac{1}{2} \parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel^2$$

If we choose the ℓ_2-norm for the inaccuracy penalty, i.e. $\| \cdot \| = \| \cdot \|_2$, we get something very close to the robust vector recovery problems of section 7, apart from that linear operators take the place of matrices, and the nuclear norm takes the place of the ℓ_1 norm.

It is possible to impose a $\parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel$ constraint as a semidefinite program, or to assign the value of this expression to a variable in such a program; thus all of these generalizations can also be expressed as semidefinite programs. I omit these for the sake of brevity, but see de Azevedo [2017] for an example of modeling a norm constraint as an SDP.

For these robust problems, we next need some bounds on recovery.

11.2.1 Robust null space property for linear operators

In 11.1.2, we saw that the stable null space property for matrix sensing was quite similar to that for compressed sensing. This suggests that it is also possible to generalize the robust null space property for compressed sensing to the matrix sensing domain. That turns out to be the case, as shown in Foucart and Rauhut. First define the matrix sensing analog:

Definition 22. A linear operator $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^p$ satisfies the robust rank null space property of order r with constant $0 < \rho < 1$ and $\tau > 0$ for some norm $\| \cdot \|$ if

$$\sum_{i=1}^{r} \sigma_i(M) \leq \quad \rho \textstyle\sum_{i=r+1}^{N} \sigma_i(M) + \tau \parallel \mathcal{A}(M) \parallel \quad \forall M \in \mathbb{R}^{m \times n}$$

where $N = \min(m, n)$.

The robust recovery theorem comparable to theorem 11 becomes:
Theorem 23. *Suppose the linear operator $\mathcal{A}$ satisfies the robust rank null space property of order r with constants ρ and τ. Let $\mathbf{y} = \mathcal{A}(X_0) + \mathbf{e}$ be the observed vector contaminated by additive noise with $\| \mathbf{e} \|_2 \leq \delta$. Then for any optimum $X^\sharp$ to the QCBP nuclear norm optimization problem*

$$\min \| X \|_* \quad \textit{subject to} \quad \| \mathcal{A}(X) - \mathbf{y} \|_2 < \delta$$

and any other feasible solution X to this problem, we have

$$\| X - X^\sharp \|_* \leq \frac{2(1+\rho)}{1-\rho} \sum_{i=r+1}^{N} \sigma_i(X) + \frac{4\tau}{1-\rho}\eta$$

As with the recovery result from robust compressed sensing, we're particularly interested in the case where X is the unknown low rank matrix X_0.

To prove this, we need the following theorem
Theorem 24. *For all $X, Z \in \mathbb{R}^{m\times n}$, iff $\mathcal{A}$ satisfies the robust rank null space property of order r with constants ρ and τ, it holds that*

$$\| X - Z \|_* \quad \leq \quad \frac{1+\rho}{1-\rho}\big(\| Z \|_* - \| X \|_* + 2 \sum_{i=r+1}^{N} \sigma_i(X)\big) + \frac{2\tau}{1-\rho} \| \mathcal{A}(Z - X) \|$$

The proof of that theorem is exercise 4.20 in Foucart and Rauhut. Given the theorem, proving theorem 23 is completely analogous to proving theorem 11: we let $Z = X^\sharp$ and use the linearity of $\mathcal{A}$ as well as the triangle inequality to get

$$\| X - X^\sharp \|_* \quad \leq \quad \frac{1+\rho}{1-\rho}\big(2 \sum_{i=r+1}^{N} \sigma_i(X)\big) + \frac{2\tau}{1-\rho} \| \mathcal{A}(X^\sharp) - \mathbf{y} \| + \| \mathbf{e} \|$$

and since both $\| \mathcal{A}(X^\sharp) - \mathbf{y} \|$ and $\| \mathbf{e} \|$ are bounded by δ, the result follows.

12 Matrix completion and variable $\mathcal{A}(X)$

The previous cases have all considered the linear operator $\mathcal{A}$, which corresponds to the matrix A in ordinary compressed sensing, to be known in advance. If we're trying to recover a recover a sparse signal $\mathbf{x}$ by an observation $\mathbf{y}$ of that signal in another basis, we usually know what kind of domain our observation apparatus uses, and we either know the space where $\mathbf{x}$ is sparse, or can make a good guess of one based on energy compaction properties.

However, in some matrix sensing situations, we might not know the exact nature of $\mathcal{A}(X)$ ahead of time. In matrix completion, we have a projection P_Ω that can be flattened into $\mathcal{A}$. If the known entries are based on user input, as in a recommender system, we don't know the exact nature of Ω in advance. In an explicit rating recommender system, Ω is the set of user and item pairs where the user has rated that item; and that could be anything. Some users might not have rated anything, and some items might not have been rated at all.

In an implicit rating recommender system, we don't even have ratings, we just know whether a user has shown interest in an item or not, and so $\mathcal{A}$ becomes something to the effect of

$$\mathcal{A}(X)_{an+b} \quad = \quad C_{ab}X_{ab}$$

where $C \in \mathbb{R}^{m \times n}$ is a confidence matrix so that C_{ij} roughly corresponds to the precision (inverse variance) of our belief in the observation X_{ij}.

In either case, the linear operator $\mathcal{A}$ is chosen by nature: the users decide its construction when they choose what items to provide feedback on.

We would thus like to know whether the nuclear norm minimization problem can recover the minimum rank matrix over a whole class of linear operators $\mathcal{A}$. For a recommender system, this class could be $\mathcal{A}_{\Omega,k}$ containing all flattened P_Ω operators with $\mid \Omega \mid \leq k$.

For unknown/variable linear operators, we have both possibility and impossibility results.

In the general case where $\mathcal{A}$ is an arbitrarily weighted generalization $\mathcal{A}_\Omega$ of the flattened P_Ω, i.e.

$$\mathcal{A}_\Omega(X)_{in+j} \quad = \quad \begin{cases} C_{ij}X_{ij} & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

for some matrix of constants C, low rank matrix completion is NP-hard. This holds even if $C \in \{0,1\}^{m \times n}$, which implies that low rank matrix completion for the whole class of linear operators $\mathcal{A}_\Omega$ is NP-hard if the set Ω is adversarially chosen [Gillis and Glineur, 2011]. There is thus no algorithm for low rank matrix completion that is poly-time for recommender systems in the worst case, unless $P = NP$.

On the other hand, if each user-item combination is revealed with a fixed probability p, and the probability that any one user-item combination is revealed is independent of every other, then we have the random orthogonal model, and nuclear norm minimization can recover the optimal low rank matrix. See part III.

Part III

Matrix completion in recommender systems

13 Introduction to recommender systems

In many situations, users of a service find themselves having to decide what items of some sort to pay attention to, and where the sheer amount of items makes it impossible to go through all of them. The most obvious example is digital media of any form: what videos to watch, what music to listen to, which users to follow on social media; but users can find themselves in a similar situation when having to decide what product, if any, they want to buy from an internet store.

Service providers might thus want to help the user decide. As the number of items (videos, music, products) increase, it becomes increasingly difficult for users to find the items they in retrospect would want to have seen. Manual recommendation, where the service provider goes through the items and finds particularly good items, can work, but such an effort becomes more difficult as the number of items increase, as well. Other, more automated methods, are needed.

A recommender system is one such method. Recommender systems attempt to predict what items an user might be interested in, given data about what items other users have shown interest in, and potentially external information, like music genre or actors participating in a movie. I'll here be using "recommender system" as another name for collaborative filtering, which strictly speaking are recommender systems that don't make use of external information.

In a recommender systems setting, we're given a number of users and a number of items, and we wish to predict the preferences of the users for the different items based on the users' past behavior. By providing personalized recommendations, the system would help users arrive at interesting items with less effort.

Recommender systems have been employed both for digital content and for product suggestions. Netflix held a well known competition to create a better recommender system for its movie service, and many web stores (like Amazon: Smith and Linden, 2017) use recommender systems to suggest products to their customers. Less obvious uses for recommender systems include suggesting jokes [Gupta et al., 1999].

In some systems, users and items can coincide, such as in a social network where we might want to recommend interesting people to follow. Thus, in some settings, an item is a user. Strictly speaking, the items would be what the users produce (e.g. posts or notifications), but the sites usually don't let users subscribe or follow individual posts or notifications; instead, a user can follow another user and indirectly be notified of events produced by that user.

The common features where it makes sense to use a recommender system are:

- There are many users and items, too many items for the users to manually go through in feasible time.
- The users prefer the different items to differing degrees; they're not ambivalent to what they're looking at.
- The users can provide feedback on what items they prefer or don't prefer. The feedback may be explicit, as in rating a movie, or implicit, as in stopping the movie playback before it's done.
- It's possible to provide customized recommendations (e.g. not a traditional TV broadcast, where the only option is whether to watch what programming is running).

13.1 Matrix factorization as a recommender system model

A model used by a recommender systems should ideally satisfy both of the following properties:

- It should have a relatively small state space, or there should be regularization options, so that it doesn't overfit.
- It should be useful: it should provide relatively good precision.

It's also an beneficial if the model can be interpreted, as such a capability can aid the store or service in predicting what may be well-liked in the future, even if the items don't exist yet.[11]

The matrix factorization model appeared relatively quickly during the Netflix Prize challenge and was a part of the winning entry [Koren et al., 2009]. In its simplest form, the matrix factorization model (called "SVD" by the Netflix Prize contestants) predicts a user u's rating of an item i, $r_{u,i}$ as a sum of interaction terms or *latent factors*:

$$r_{u,i} \quad = \quad \sum_{x=1}^{f} a_{u,x} \cdot b_{i,x}$$

with f being the number of factors, which is a parameter that needs to be set ahead of time. Training the system then consists of solving the following optimization problem:

$$\min_{a_{1,1},\cdots,a_{m,f},b_{1,1},\cdots,b_{n,f}} \sum_{(u,i)\in\Omega} \big(r_{u,i} - \sum_{x=1}^{f} a_{u,x}b_{i,x}\big)^2$$

[11]See for instance Netflix's use of its data to predict that the series *House of Cards* would be a success.

where Ω is the set of indices (u, i) where $r_{u,i}$ is already known, and m and n are the number of users and items respectively.

This is a rather simple model, but the users behaved close enough to something linear for it to prove useful.[12]

It is also relatively interpretable, since each item variable $b_{i,x}$ can be seen as the degree to which item i exhibits feature x, and $a_{u,x}$ how much user u likes feature x.

The matrix factorization model is also interesting to us because it can be cast as a matrix completion problem. Let $U_{xy} = a_{x,y}$ and $V_{xy} = b_{y,x}$. The rating prediction becomes

$$r_{u,i} \quad = \quad U_u \cdot V_i^T$$

or

$$R \quad = \quad UV^T$$

Denote the matrix of observed ratings as M, and $P_\Omega(X)$ defined as in section Section 8. The optimization problem ends up being

$$\min_{U,V^T} \parallel P_\Omega(UV^T) - P_\Omega(M) \parallel_F \tag{13.1}$$

with $U \in \mathbb{R}^{mxf}$, $V \in \mathbb{R}^{nxf}$. As we'll find out, a regularized version of this problem can be shown as equivalent to nuclear norm minimization.

14 Solving matrix completion recommender problems

14.1 Limitations of semidefinite programming

Many libraries exist for solving semidefinite programs.[13] However, references to the asymptotic complexity of semidefinite programming algorithms is usually scarce in the literature, so it is hard to compare the algorithms' performance. Kulis et al. [2007] state that general purpose semidefinite programming solvers have $O(N^3)$ performance or worse (depending on the solver), where $X \in \mathbb{R}^{N\times N}$ is the matrix that is constrained to be positive semidefinite. For the nuclear norm minimization program, $N = n + m$, thus the sum of the number of items and users in a recommender system. As both n and m can be of the order 10^4 or greater, semidefinite programming is not a practical method for a recommender system.

We thus need more practical algorithms to perform matrix completion if we're to use the matrix factorization model in a recommender system. From the practical side of

[12]Less simple variants introduced regularization analogous to robust recovery; and time preference, where products rated recently count more than those rated a long time ago.

[13]An incomplete list is CVX, CVXOPT, DSDP, CSDP, SDPNAL, and SDPNAL+.

things, recommender systems designers have used gradient descent or an algorithm called alternating least squares, which we'll talk about later. The use of these algorithms in the Netflix competition was motivated by performance rather than any analytical results. This thus raises the question of how we can characterize fast algorithms beyond just "works well in practice".

14.2 A fast algorithm for matrix completion

An algorithm often used to find a good matrix factorization in a recommender system is the alternating least squares method. Consider the following problem [Mazumder et al., 2010]:

$$\min_{U,V} \frac{1}{2} \sum_{(i,j)\in\Omega} (Y_{ij} - (UV^T)_{ij})^2 + \frac{\lambda}{2}\big(\parallel U \parallel_F^2 + \parallel V \parallel_F^2 \big) \tag{14.1}$$

or in short,

$$\min_{U,V} \parallel P_\Omega(Y) - P_\Omega(UV^T) \parallel_F^2 + \lambda\big(\parallel U \parallel_F^2 + \parallel V \parallel_F^2 \big)$$

where $X = UV^T$, $U \in \mathbb{R}^{mxr}$, $V \in \mathbb{R}^{nxr}$, and the maximum rank $r \leq \min(n, m)$. Without loss of generality, we'll let $r = \min(n, m)$, although r can be reduced for computational purposes if the actual rank of X (or an upper bound for the rank) is known for a particular λ.

Mazumder et al. further show
Lemma 25.

$$\parallel X \parallel_* \quad = \quad \min_{U,V:X=UV^T} \frac{1}{2}(\parallel U \parallel_F^2 + \parallel V \parallel_F^2)$$

when U and V are defined as above.

Proof. Let $M = \begin{pmatrix} U \\ V \end{pmatrix} \begin{pmatrix} U^T & V^T \end{pmatrix} = \begin{pmatrix} UU^T & X \\ X^T & VV^T \end{pmatrix}$. By construction, $M \succcurlyeq 0$.

Subsection (10.1) then gives that $Trace(UU^T) + Trace(VV^T) = 2 \parallel X \parallel_*$.

Recognizing $Trace(UU^T) = \parallel U \parallel_F^2$ and $Trace(VV^T) = \parallel V \parallel_F^2$ completes the proof. □

Thus the minimization problem of (14.1) is equivalent to

$$\min_X \frac{1}{2} \sum_{(i,j)\in\Omega} (Y_{ij} - X_{ij})^2 + \lambda \parallel X \parallel_*$$

The optimization problem (14.1) is not convex when minimizing over both U and V, but if we fix either U or V, optimizing for the other is a convex problem. This observation leads to the alternating squares method. As suggested, it consists of alternating between optimizing U with V fixed and optimizing V with U fixed.

Suppose U is fixed. Our problem becomes

$$\min_V \quad \tfrac{1}{2}\big(\textstyle\sum_{(i,j)\in\Omega}(Y_{ij}-(UV^T)_{ij})^2\big)+ \quad \displaystyle\frac{\lambda}{2}\sum_{i=1}^{n} \parallel V_i \parallel^2$$

We have a linear regression - the $\sum_{(i,j)\in\Omega}(Y_{ij}-(UV^T)_{ij})^2\big)$ term - and an ℓ_2 regularization term - $\frac{\lambda}{2}\sum_{i=1}^{n} \parallel V_i \parallel^2$. So the problem is an ℓ_2-regularized linear regression, i.e. a ridge regression problem.

The presence of unknown values $(i,j)\notin\Omega$ makes expressing the problem in matrix form tougher than an ordinary ridge regression. First, define a matrix C so that

$$\begin{aligned} C_{ij} &= \begin{cases} 1 & (i,j)\in\Omega \\ 0 & \text{otherwise} \end{cases} \\ C_{i,:} &= \operatorname{diag}(C_i) \\ C_{:,j} &= \operatorname{diag}(C_j^T) \end{aligned}$$

and a zero-completed extension of Y,

$$\tilde{Y}_{ij} = \begin{cases} Y_{ij} & (i,j)\in\Omega \\ 0 & \text{otherwise} \end{cases}$$

Let $\bar{U},\bar{V}$ be the previous iteration's approximations to U and V respectively, and $\hat{U}$, $\hat{V}$ be this iteration's approximations to U and V. In other words, using our notation, for iteration k, $\bar{U}=U_{k-1}$, $\bar{V}=V_{k-1}$, $\hat{U}=U_k$, $\hat{V}=V_k$. I do this to avoid otherwise very messy notation when indexing a matrix that itself has a subscript giving its iteration number.

Then Hu et al. [2008] give that

$$\begin{aligned} \hat{U}_i &= \big(\bar{V}^T C_{i,:}\bar{V}+\lambda I\big)^{-1}\bar{V}^T C_{i,:}\tilde{Y}_i \\ \hat{V}_j &= \big(\hat{U}^T C_{:,j}\hat{U}+\lambda I\big)^{-1}\hat{U}^T C_{:,j}\tilde{Y}_j^T \end{aligned}$$

In practice, implementations of the method use multiple speedups, some of which are given in Hu et al..

The benefit of the alternating least squares method is that it is practical for large matrices, as opposed to solving a semidefinite program. However, the best known bounds on when we will recover the minimum rank solution are weaker than for nuclear norm minimization.

15 Bounds on exact recovery using SDP and ALS

15.1 Prerequisite definitions

We'll need some definitions first. First two different types of coherence:

Definition 26. Coherence

Let the coherence of a matrix $U \in \mathbb{R}^{nxn}$ with respect to rank k be defined as

$$\mu(U) \quad = \quad \max_{i \in [U]} \frac{n}{k} \parallel \mathbf{e_i^T} U \parallel_2^2 \tag{15.1}$$

Jain et al. [2013, definition 2.4] define a different type of coherence:
Definition 27. A matrix $M \in \mathbb{R}^{m \times n}$ is μ-incoherent with parameter $\mu = \mu_0$ if

$$\parallel U_i \parallel \leq \frac{\mu_0 \sqrt{k}}{\sqrt{m}} \forall i \in [m], \quad \parallel V_i \parallel \leq \tfrac{\mu_0 \sqrt{k}}{\sqrt{n}} \forall j \in [n] \tag{15.2}$$

where $M = U \Sigma V^T$ is the singular value decomposition of M.

Next is the restricted isometry for linear operators, which is completely analogous to definition 15:
Definition 28. $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^d$ satisfies the $k - RIP$ with constant δ_k if, for all $X \in \mathbb{R}^{m \times n}$ where $rank(X) \leq k$, the following holds:

$$(1 - \delta_k) \parallel X \parallel_F^2 \leq \parallel \mathcal{A}(X) \parallel_2^2 \leq (1 + \delta_k) \parallel X \parallel_F^2$$

To prove a matrix sensing result for alternating least squares, we also need a notion of distance between subspaces:
Definition 29. Let $\hat{X}, \hat{Y} \in \mathbb{R}^{m \times n}$ be given matrices and let X, Y be orthonormal basis matrices of respectively $Span(\hat{X})$ and $Span(\hat{Y})$. Let $X_\perp$, $Y_\perp$ be orthonormal basis matrices of the perpendicular spaces $Span(\hat{X})^\perp$ and $Span(\hat{Y})^\perp$ respectively.

The principal angle distance *dist* between subspaces is defined as

$$\begin{aligned} \text{dist}(\hat{X}, \hat{Y}) &= \parallel X_\perp^T Y \parallel_2 \\ &= \parallel Y_\perp^T X \parallel_2 \end{aligned}$$

with $\parallel \cdot \parallel_2$ being the spectral norm.

If A and B span the same space, then $(A_\perp)_i \cdot B_j = \mathbf{0}$ for all (i, j) by definition of perpendicularity, and thus $dist(A, B) = 0$. If the ranks of A and B are not equal, $dist(A, B) = 1$.

Proof. Without loss of generality let $rank(\hat{B}) > rank(\hat{A})$. Choose orthonormal bases $A_\perp, B$ so that there exists a basis vector b_i also in $A_\perp$. Then since $\|A\|_2 = \max_{\mathbf{x}} \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$, $\|A_\perp^T B\|_2 \geq \|A_\perp^T B e_j\|_2 = 1$. Since the bases are orthonormal, $\frac{\|A_\perp^T B\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq 1$ for all $\mathbf{x}$, so $\|A_\perp^T B\|_2 = 1$. □

15.2 Matrix sensing

Let $M \in \mathbb{R}^{m\times n}$ with singular values $\sigma_1, \ldots, \sigma_k$ be the unknown rank k matrix to be recovered, and define $N = \max(m, n)$. As semidefinite programming exactly solves nuclear norm minimization, see 11.1 for the relevant bounds for SDP.

15.2.1 Alternating least squares

Jain et al. [2013] give the following bound for the alternating least squares algorithm:

If $\mathcal{A}$ satisfies the $2k - RIP$ with constant

$$\delta_{2k} \quad < \quad \frac{\sigma_k^2}{\sigma_1^2} \cdot \frac{1}{100k}$$

and the ALS algorithm has a properly prepared first estimate U_0,[14] it converges geometrically to the optimal solution to (8.1).

15.2.2 Proof

We first need some decompositions.

Let $M = U_{*D}\Sigma_{*D}V_{*D}^T$ with $M \in \mathbb{R}^{m\times n}$, $U_{*D} \in \mathbb{R}^{m\times m}$, $\Sigma_{*D} \in \mathbb{R}^{m\times n}$, $V_{*D} \in \mathbb{R}^{n\times n}$ be the singular value decomposition of M. Since M has rank $k < \min(m, n)$, some of the entries along the diagonal of Σ_{*D} will be zero. We can thus truncate these and remove the extraneous corresponding columns of U_{*D} and V_{*D} to get $M = U_{*F}\Sigma_{*F}V_{*F}^T$ with $U_{*F} \in \mathbb{R}^{k\times m}$, $\Sigma_{*F} \in \mathbb{R}^{k\times k}$, $V_{*F} \in \mathbb{R}^{k\times n}$. The latter is called the singular value factorization of M.

Jain et al., thm 2.2 show that if $\mathcal{A}$ satisfies the $2k$-RIP, the alternating least squares algorithm (with a particular first step), produces a series of approximations $\hat{U}_t, \hat{V}_t$ whose product converges geometrically towards M. Its formal statement is
Theorem. *Let M be a rank k matrix with nonzero singular values $\sigma_1 \geq \cdots \geq \sigma_k$. Let the linear measurement operator $\mathcal{A}(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}^d$ satisfy 2k-RIP with constant $\delta_{2k} \leq \frac{(\sigma_k)^2}{(\sigma_1)^2} \cdot \frac{1}{100k}$. Then using the AltMinSense algorithm[15], for all iterations $t > T = 2\log(\frac{\|M\|_F}{\epsilon})$, we have that the iterates $\hat{U}_t$, $\hat{V}_t$ satisfy $\| M - \hat{U}_t\hat{V}_t^T \|_F \leq \epsilon$.*

Analysis makes use of theorem 4.2 of the paper, which states that the distance between the subspaces covered by U_t and V_t (again my notation), which are the outputs after time step t in ALS, and the subspaces U_{*F}, V_{*F} covered by the factorization $M = U_{*F}\Sigma_{*F}V_{*F}^T$,

[14]See the paper for details on how U_0 is prepared.

[15]AltMinSense is alternating minimization with properly prepared U_0

diminishes geometrically given appropriate starting points. More formally [Jain et al., 2013, p. 8], we have:

$$\begin{aligned}\text{dist}(V_{t+1}, V_{*F}) &\leq \frac{1}{4} \cdot \text{dist}(U_{t+1}, U_{*F}) \\ \text{dist}(U_{t+1}, U_{*F}) &\leq \frac{1}{4} \cdot \text{dist}(V_{t+1}, V_{*F})\end{aligned}$$

with the distance between subspaces defined below.[16] I won't go into detail in proving theorem 4.2. itself, but I'll show how it can be used to show that for a given ϵ, then after $T = 2\log(\frac{\|M\|_F}{\epsilon})$ steps, $\| M - \hat{U_T}\hat{V_T^T} \|_F^2 \leq \epsilon$.

15.2.3 Proof using theorem 4.2

Let $\hat{U}_t, \hat{W}_t \in \mathbb{R}^{m\times k}$ be matrices generated by ALS at iteration t, and let U_t, W_t be orthonormal basis matrices of $Span(\hat{U}_t)$ and $Span(\hat{W}_t)$. Let $U_{t_\perp}$, $W_{t_\perp}$ be orthonormal basis matrices of the perpendicular spaces $Span(\hat{U}_t)^\perp$ and $Span(\hat{W}_t)^\perp$, and *dist* the distance measure between subspaces defined previously.

RIP says that if

$$(1-\delta_k) \| X \|_F^2 \quad \leq \| \mathcal{A}(X) \|_2^2 \quad \leq (1+\delta_k \| X \|_F^2$$

holds for all X of rank k for some $0 \leq \delta_k < 1$, we say k-RIP is satisfied with constant δ_k. Recall that M satisfies the $2k$-RIP with constant δ_{2k}, and consider a particular $X = M - \hat{U_T}\hat{V_T^T}$. We get

$$(1-\delta_{2k}) \| M - \hat{U_T}\hat{V_T^T} \|_F^2 \quad \leq \quad \| \mathcal{A}(M - \hat{U_T}\hat{V_T^T}) \|_2^2$$

Dividing by $1-\delta_{2k}$ gives that the following holds since M satisfies the $2k$-RIP with δ_{2k}:

$$\| M - \hat{U_T}\hat{V_T^T} \|_F^2 \quad \leq \quad \frac{1}{1-\delta_{2k}} \| \mathcal{A}(M - \hat{U_T}\hat{V_T^T}) \|_2^2$$

We get

$$\begin{aligned}\| M - \hat{U_T}\hat{V_T^T} \|_F^2 &\leq \frac{1}{1-\delta_{2k}} \| \mathcal{A}(M - \hat{V_T}\hat{V_T^T}) \|_2^2 \\ &= \frac{1}{1-\delta_{2k}} \| \mathcal{A}(M(I - \hat{V_T}\hat{V_T^T})) \|_2^2\end{aligned}$$

This follows because $\hat{U_T}$ was the optimal solution for $\min_Y \| \mathcal{A}(M - Y\hat{V_T^T}) \|_2^2$, so anything else inserted as Y must either keep the spectral norm value the same, or increase it.

[16]Note that we have a convergence between subspaces, not a convergence between matrices. So even though U_{t+1} covers increasingly more of the subspace that U_{*F} does, this does not imply that $\hat{U}_T\hat{V}_T^T = U_{*F}V_{*F}^T$.

Inserting the result into the RIP Using the RIP again, with the rearranged term in the middle,

$$\frac{1-\delta_{2k}}{1-\delta_{2k}} \parallel M - \hat{V}_T \hat{V}_T^T \parallel_F^2 \ \leq \tfrac{1}{1-\delta_{2k}} \parallel \mathcal{A}(M - \hat{V}_T \hat{V}_T^T) \parallel_2^2 \ \leq \frac{1+\delta_{2k}}{1-\delta_{2k}} \parallel M - \hat{V}_T \hat{V}_T^T \parallel_F^2$$

and we thus get

$$\parallel M - \hat{U}_T \hat{V}_T^T \parallel_F^2 \quad \leq \quad \frac{1+\delta_{2k}}{1-\delta_{2k}} \parallel M - \hat{V}_T \hat{V}_T^T \parallel_F^2$$

and factoring out M and subsequently inserting $M = U_{*F} \Sigma_{*F} V_{*F}^T$,

$$\leq \quad \frac{1+\delta_{2k}}{1-\delta_{2k}} \parallel U_{*F} \Sigma_{*F} V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_F^2$$

Making use of dist By definition of *dist*,

$$\text{dist}(\hat{U}_t, \hat{W}_t) \quad = \quad \parallel U_t^T W_{t_\perp} \parallel_2$$

The Frobenius norm is consistent, so we have

$$\begin{aligned} \parallel U_{*F} \Sigma_{*F} V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_F^2 \quad &\leq \quad \parallel U_{*F} \Sigma_{*F} \parallel_F^2 \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_F^2 \\ &= \quad \parallel U_{*D} \Sigma_{*D} \parallel_F^2 \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_F^2 \end{aligned}$$

then by unitary invariance of the Frobenius norm,

$$\begin{aligned} &= \quad \parallel U_{*D} \Sigma_{*D} V_{*D}^T \parallel_F^2 \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_F^2 \\ &= \quad \parallel M \parallel_F^2 \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_F^2 \end{aligned}$$

then equivalence $\parallel A \parallel_F \leq \sqrt{rank(A)} \parallel A \parallel_2$ along with $rank(A) \geq 1$ implies $\parallel A \parallel_F^2 \leq \parallel A \parallel_2^2$, so

$$\leq \quad \parallel M \parallel_F^2 \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_2^2$$

Now note that $(I - \hat{V}_T \hat{V}_T^T)W$ will project W down on $V_{T_\perp}$. Let $W = \begin{bmatrix} V_{T_\perp} & V_T \end{bmatrix}$. Then, since the spectral norm is also unitary invariant, we have

$$\begin{aligned} \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) \parallel_2^2 \quad &= \quad \parallel V_{*F}^T (I - \hat{V}_T \hat{V}_T^T) W \parallel_2^2 \\ &= \quad \parallel V_{*F}^T \, [V_{T_\perp}, 0] \parallel_2^2 \\ &= \quad \parallel V_{*F}^T V_{T_\perp} \parallel_2^2 \\ &= \quad \parallel V_{*F}^T V_{T_\perp} \parallel_2^2 \\ &= \quad \text{dist}(V_{*F}, V_T)^2 \end{aligned}$$

so we arrive at the conclusion that

$$\begin{aligned}\| M - \hat{U_T}\hat{V_T^T} \|_F^2 &\leq \| U_* \Sigma_* V_*^T (I - \hat{V_T}\hat{V_T^T}) \|_F^2 \\ &\leq \| M \|_F^2 \operatorname{dist}(V_{*F}, V_T)^2\end{aligned}$$

Finally, by theorem 4.2. we have that

$$\| M \|_F^2 \, dist(V_{*F}, V_T)^2 \quad \leq \quad \epsilon$$

15.3 Matrix completion

In section 12, the fully general matrix completion problem was shown to be NP-hard. Unless $P = NP$, nuclear norm minimization can't possibly recover the hidden low rank matrix X in every possible case of P_Ω. In addition, under some Ω sets, we simply can't recover the matrix as there's not enough information.

Suppose Ω fails to contain any observation from column j, and M has rank one, $M_{ij} = \mathbf{x}_i \mathbf{y}_j$. Then there's no way to recover $\mathbf{y}_j$, because the information simply isn't there [Candès and Recht, 2008].

Thus we need additional assumptions on both M and P_Ω.

One common assumption is that M is incoherent in some way (e.g. $\mu(M)$ or $\max(\mu(U), \mu(V))$ being small with $M = U\Sigma V^T$ being the SVD), and that each entry of M is known with independent probability p so that $\mid \Omega \mid = O(pmn)$. Call this the incoherent uniform model. The exact type of incoherence may differ; in some situations, we use standard incoherence of (15.1); in others, μ-incoherence of (15.2).

15.3.1 Nuclear norm minimization

Candès and Recht [2008] introduces the random orthogonal model, where the coherence of M is not explicitly stated, but M is said to have rank k and $M = U\Sigma V^T$ with $\{U_i\}_{i=1}^k$ and $\{V_i\}_{i=1}^k$ is chosen uniformly at random from all families of k orthonormal vectors.

For the random orthogonal model, they prove that there exist constants C_1 and c so that the nuclear norm minimization problem, (8.1), recovers the minimum rank matrix with probability $1 - cN^{-3}$ if

$$\mid \Omega \mid \quad \geq \quad C_1 \, N^{5/4} k \, \log N$$

If the rank is particularly low compared to the matrix dimensions, i.e. $k \leq N^{1/5}$, then we get an even better bound of

$$\mid \Omega \mid \geq C_1 \, N^{6/5} k \, \log N \tag{15.3}$$

The authors then generalize this to (a restriction of) the incoherent uniform model. Consider a matrix $M \in \mathbb{R}^{m\times n}$ of rank k with $M = U\Sigma V^T$, $W = UV^T$ and $N = \max(m, n)$, and let μ_0 and μ_1 be defined so that

- $\max\big(\mu(U), \mu(V)\big) \leq \mu_0$
- $\max_{i,j} \mid W_{ij} \mid \leq \mu_1\sqrt{\frac{k}{mn}}$s
- $\mu_1 \geq 1$

Then there exist constants C and c so that for all $\beta > 2$, nuclear norm minimization can recover the minimum rank matrix with probability $1 - cN^{-\beta}$ if

$$\mid \Omega \mid \quad \geq \quad C\max(\mu_1^2, \sqrt{\mu_0}\mu_1, \mu_0 N^{1/4})Nk(\beta \log N)$$

and for particularly low rank, $k \leq \frac{1}{\mu_0}N^{\frac{1}{5}}$, we get a better bound of

$$\mid \Omega \mid \quad \geq \quad C\mu_0 N^{6/5}k(\beta \log N)$$

If we let $\beta = 3$, then we get essentially the same result as (15.3), except for the $\frac{1}{\mu_0}$ factor in the particularly low rank constraint, and the μ_0 factor in the bound itself.

Since this bound does not contain any μ_1 terms, it holds for the incoherent uniform model in general, not just the one where there's a restriction on the magnitude of the W_{ij} elements. The particularly low rank constraint might be too hard, however: for a recommender system matrix of 10000 users and items, $N^{1/5} = 6.3$. Even for a Netflix-scale dataset with 10^6 users and items, $N^{1/5}$ is still only around 15.85.

We can generalize the restricted bound to hold for the general incoherent uniform model by making use of Candès and Recht's observation that $\mu_1 = \mu_0\sqrt{k}$ always holds. Inserting and letting $\beta = 3$ gives that there exist constants D and c so that nuclear norm minimization can recover the minimum rank matrix with probability $1 - cN^{-3}$ if

$$\mid \Omega \mid \quad \geq \quad D\max(\mu_0^2 k, \mu_0^{3/2}\sqrt{k}, \mu_0 N^{1/4})Nk(\log N)$$

15.3.2 Generalized alternating least squares

Hardt [2014, p. 3] provides an ALS-based algorithm that outputs matrices X and Y so that $\|M - XY^T\|_F \leq \epsilon\|M\|_F$ for the unknown rank k matrix M, with high probability if

$$pN \quad \geq \quad k(k + \log(\frac{N}{\epsilon})\mu(U)(\frac{\parallel M \parallel_F}{\sigma_k})^2$$

for symmetric matrices $M = U\Lambda U^T$ where Λ is a diagonal matrix, and also gives a way to generalize this to any matrix (rectangular nonsymmetric).

Note that unlike the bounds for nuclear norm minimization, the bound above does not just involve the incoherence $\mu(U)$, size N and rank k, but also the desired accuracy ϵ and Frobenius norm of M. Unlike nuclear norm minimization, we don't either recover fully or not at all; whether we recover to some accuracy ϵ depends on ϵ itself. Of course, the bounds may not be tight, and the acutal situation may be better than it seems here.

15.3.3 Alternating least squares

For the standard ALS algorithm, we have the following bounds and proof from Jain et al. [2013]:

We assume here that we want to recover a μ-incoherent matrix $M \in \mathbb{R}^{m \times n}$ of rank k.

In the incoherent uniform model, with M having singular values $\sigma_1, \cdots, \sigma_k$, if M satisfies the $2k$-RIP with constant $\delta_{2k} \leq \frac{\sigma_k}{12k\sigma_1}$, then theorem 2.5 of Jain et al. [2013] gives the following result:

There exist constants C and C' so that if the probability p of observing a random element satisfies

$$p \geq C\frac{\left(\frac{\sigma_1}{\sigma_k}\right)^2 \mu^2 k^{2.5} \log n \log \frac{k\|M\|_F}{\epsilon}}{m\delta_{2k}^2}$$

then after $T = C' \log \frac{\|M\|_F}{\epsilon}$ steps of the alternating least squares algorithm, supposing the initial estimates are prepared in a particular manner, the error between the true matrix M and the approximation will be reduced to $\| M - \hat{U}_T \hat{V}_T^T \|_F \leq \epsilon$.

This is equivalent to saying that, in the incoherent uniform model, if M is k-incoherent and satisfies the $2k$-RIP with $\delta_{2k} = \frac{\sigma_k^*}{12k\sigma_1^*}$, then $| \Omega | = O(\left(\frac{\sigma_1^*}{\sigma_k^*}\right)^4 \mu^2 k^{4.5} n \log n \log \frac{k\|M\|_F}{\epsilon})$ will suffice to recover M within ϵ distance by Frobenius norm. $O()$, being an order-of operation, hides the dependence on the unknown global constant C.

Proof of this equivalence We start with the initial bound from Jain et al.,

$$p \geq C\frac{\left(\frac{\sigma_1}{\sigma_k}\right)^2 \mu^2 k^{2.5} \log n \log \frac{k\|M\|_F}{\epsilon}}{m\delta_{2k}^2}$$

We then multiply by mn on both sides to get the expected number of elements on the left ($\mathbb{E}[| \Omega |] = pmn$):

$$pmn \geq Cn\frac{\left(\frac{\sigma_1}{\sigma_k}\right)^2 \mu^2 k^{2.5} \log n \log \frac{k\|M\|_F}{\epsilon}}{\delta_{2k}^2}$$

Since we're assuming $2k$-RIP holds with constant $\delta_{2k} = \frac{\sigma_k}{12k\sigma_1}$, we can replace the former with the latter:

$$\begin{aligned}
&\geq Cn\frac{\left(\frac{\sigma_1}{\sigma_k}\right)^2 \mu^2 k^{2.5} \log n \log \frac{k\|M\|_F}{\epsilon}}{\left(\frac{\sigma_k}{\sigma_1}\right)^2 \left(\frac{1}{12}\right)^2 \left(\frac{1}{k}\right)^2} \\
&= \frac{C}{\left(\frac{1}{12}\right)^2} n \left(\frac{\sigma_1}{\sigma_k}\right)^4 \mu^2 k^{4.5} \log n \log \frac{k \parallel M \parallel_F}{\epsilon}
\end{aligned}$$

Now let D be another global constant, $D = 12^2 C$, and we get that

$$\begin{aligned}
pmn &\geq D\left(\frac{\sigma_1}{\sigma_k}\right)^4 \mu^2 k^{4.5} n \log n \log \frac{k \parallel M \parallel_F}{\epsilon} \\
&= O\left(\left(\frac{\sigma_1}{\sigma_k}\right)^4 \mu^2 k^{4.5} n \log n \log \frac{k \parallel M \parallel_F}{\epsilon}\right)
\end{aligned}$$

which was what was wanted.

15.3.4 Observations and comments

For nuclear norm minimization, we found that it's impossible to ensure recovery in full generality (over all P_Ω). We then considered the incoherent uniform model, where it's possible to give bounds that ensure recovery with high probability if the unknown matrix M is incoherent and we are given distinct elements drawn uniformly at random from M.

The bounds for nuclear norm minimization depend on how many such elements we're given, the rank of the unknown matrix M, and the coherence of M, as well as a bound on the singular vectors of M if rank is not very low. On the other hand, for alternating least squares, we see also dependence on $\|M\|_F$, on the accuracy we want for the recovery, and on either σ_k or the condition number $\frac{\sigma_1}{\sigma_k}$. Thus, while semidefinite programming is slow, we can't get an exact solution from alternating least squares, and its bounds are more complex.

Part IV

Extending recommender systems

While matrix factorization can be employed to quite a good effect in collaborative filtering, there are still limitations to what it can do. There are broadly speaking two types of limitations:

- Prediction accuracy limits to the model, e.g. users behave in a nonlinear fashion whereas matrix factorization is explicitly linear (or bilinear if we consider both the user feature and item feature spaces as distinct linear spaces).

- Limits to the application: even if the model were to have perfect prediction accuracy, it would lack certain desirable features, and could possibly lead to undesirable consequences when used on its own.

In simpler terms, the first limitation arises when the system gives a wrong answer, and the second arises when the system gives a right answer - but to the wrong question.

I'll give an example of both of these, and how a practical collaborative filtering method might be improved by addressing these limitations.

16 Generalizations of the model

It is not entirely realistic to suppose that people's preferences are the result of a bilinear process such as it is modeled in ordinary matrix factorization. Matrix factorization still has predictive power, because the users' behavior can be approximated by a linear model, but we could get closer by a more complex choice of model.

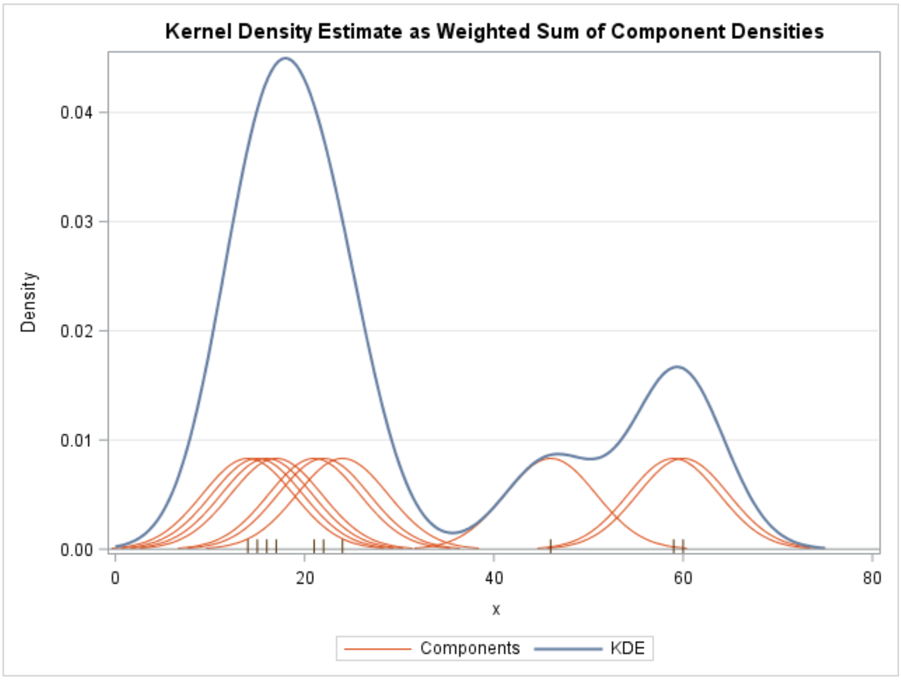
16.1 Local low rank approximation

The idea behind local low rank approximation One way of generalizing the low rank assumption is to suppose it only holds in certain neighborhoods, as is done in Lee et al. [2013].

In the article, the authors combine matrix factorization and nonparametric statistics. Their more general model supposes that there exists some distance metric $d((u, i), (u', i'))$ between pairs of users and items, so that groups of user-item pairs that have mutually short distance behave similarly. Under that assumption, the users and items in a particular neighborhood (area of short mutual distances) are considered to behave in a low-rank manner, though taken as a whole, the predicted ratings matrix the recommender system generates is not low rank.

The idea of combining many functions with simple structure to obtain a more complex structure can be illustrated by kernel density estimation, which is a generalization of a histogram.

Kernel density estimation concerns itself with estimating a probability density function based on a number of observations drawn from that distribution. Instead of grouping the observations into distinct bins and estimating the same density for every point in a certain bin, like a histogram does, kernel density estimation centers a function on each observation and lets the sum of these functions constitute the estimated density. The following illustration from Wicklin [2016] shows the principle:



The component function (or kernel) in the KDE above is a Gaussian probability distribution function. In local low rank approximation (LLORMA), the kernel is a weighting matrix, called $K_h^{(a,b)}$ in the paper, which gives a high weight to user-item pairs in the neighborhood of user a and item b. As neighborhoods differ, the $K_h^{(a,b)}$ matrices also differ based on what user-item pairs are most like (a, b) according to the distance function d.

The weight matrices are used to solve weighted nuclear norm minimization problems of the form

$$\min \quad \| X \|_* \quad \text{s.t} \ \| K_h^{(a,b)} \cdot P_\Omega(X - M) \|_F < \delta$$

where M is the matrix of observed entries, $A \cdot B$ is the element-wise multiplication operator, and P_Ω is the projection upon the observed pairs we've seen earlier. This gives one local low-rank matrix $\hat{T}_{(a,b)}$ for each (a, b) pair chosen for the minimization problem. Thus each $\hat{T}_{(a,b)}$ matrix is a low rank matrix that's accurate in the vicinity of the neighborhood where (a, b) is the center.

Computational problems Suppose $M \in \mathbb{R}^{m \times n}$, i.e. we have m users and n items in our recommender system. Then if we had unlimited computing power and a distance function d specified beforehand, we could determine $\hat{T}_{(a,b)}$ for all $m \cdot n$ possible choices of (a, b). Then we could let the predicted recommendations matrix $\hat{M}$ be defined by $\hat{M}_{ij} = (\hat{T}_{(i,j)})_{ij}$.

However, this is computationally impractical, and the authors give another approach. Instead of calculating a $\hat{T}_{(a,b)}$ for each possible pair, they suggest choosing a limited number - on the order of 10 - anchor points, calculate $\hat{T}_{(a,b)}$ only for these anchor points, and interpolating between them to get the full predicted recommendations matrix $\hat{\hat{M}}$.

Required parameters To perform the low rank matrix approximation for a given observations matrix M, we need a distance function d between user-item pairs, the choice of error δ, the bandwidth h used for computing $K_h^{(a,b)}$, and a choice of anchor points. The paper authors suggest the following:

- For d, first decompose the function into something that is more intuitive, a product of user similarity and item similarity. That is, for users a and c, and items c, d, $d((a,b),(c,d)) = d_u(a,c) \cdot d_i(b,d)$. Then factorize M into $M = UV^T$ by ordinary nuclear norm minimization, and let $d_u(a,c) = \arccos\left(\frac{U_a \cdot U_c}{\|U_a\| \cdot \|U_c\|}\right)$, $d_i(b,d) = \arccos\left(\frac{V_b \cdot V_d}{\|V_b\| \cdot \|V_d\|}\right)$.[17]
- For the locality parameter or bandwidth h, the authors show the performance of the method for different choices of h, but do not give a particular choice. This parameter, as well as δ, could probably be found by cross-validation.
- For the anchor points, page 19 of the paper gives results for different selection strategies, and the simple strategy of choosing randomly at uniform from all possible pairs (a,b), or from Ω, do just as well as more complex strategies.

Performance and observations Lee et al. run their implementation of local low rank approximation on several datasets, and compare the results to "SVD" method, which consists of solving the non-regularized problem of (13.1). The paper does not refer to the algorithm used to solve or approximately solve (13.1), but it's reasonable to assume by the name of "SVD", that it is alternating least squares. The paper also includes results for some other approaches I have not investigated, such as Accelerated Proximal Gradient (APG) and Divide-and-Conquer Matrix Factorization (DFC).

On each of the tested datasets, local low rank approximation shows lower RMS error compared to the other methods, although not every alternate method was tried on every dataset. For the Netflix dataset, local low rank approximation with rank $r = 20$ attained an RMSE of 0.834 compared to the winner of the Netflix contest at 0.857.

Thus, according to the paper, local low rank approximation seems to produce quite respectable results. If it had been submitted to the Netflix contest, it would have won the grand prize.

[17]It would be interesting to investigate whether we could iteratively refine d in a setting where local low rank matrix approximation would be run multiple times. One could imagine defining d on the $k+1$th run of LLORMA as a function of the $\hat{T}_{(a,b)}$ matrices from the kth run.

The drawback is that interpretability suffers: since local low rank approximation uses many individually low-rank matrices, it's no longer as easy to determine what the matrices mean. For a bilinear model, we can think of the rows of U as user preferences for various factors, and the rows of V as to what degree each item contains the different factors, each factor being a property like "percussive music" or "movie with outdoors scenes". However, when we're predicting based on a weighted combination of many low-rank matrices, what the factors for each low rank matrix means is no longer as clear.

In addition, if the anchor points are chosen at random, that reduces interpretability further. For an interpretable model, perhaps it would be better to choose anchor points according to side data (such as music genre). Then each local low rank matrix consists of features important to the center where the anchor point is placed. As an example, if the anchor point is placed at a user who is a fan of classical music (as the user component of the tuple defining an anchor point), and a well known piece of classical music (as the item component of the pair), then the low rank matrix based on this center would contain features that fans of classical music consider to be of particular interest. Many of these features would differ from those of fans of black metal music.

17 Problems due to the application

When used as a recommender system, matrix completion methods have a significant flaw: they take the observations as given, instead of using the recommendations to actively acquire more information about the user who receives those recommendations. There are three particular settings where the limitations of this strategy are evident:

- The so-called "cold start" problem, where the system has no information about users who have just joined, or items that have just been produced, and thus doesn't know what items it should suggest to the new user, or who it should suggest the new item to.
- Self-reinforcing situations where a user has only expressed interest in some type of item (e.g. heist movies), but would enjoy some other type of item (say, political thrillers) if it were recommended to him. The system doesn't know what it doesn't know, and so can't experiment with providing the user more diverse suggestions to gain more information.
- Saturation or time preference problems, where a user may have recently made use of many items of one type, but is now bored and would like greater variety, but the system keeps recommending the same type of item. A common example is a shopping site that keeps recommending refrigerators even after a customer has purchased one. Another is a music site that recommends loud music even past the point where the user gets exhausted and would prefer something more quiet.

The flaw arises when the matrix completion methods are used directly to form recommendations. Since the data we use to generate recommendations is based on observation, there's some uncertainty as to how much the observations represent the users' behavior; the less data we have, the greater the uncertainty. But a plain matrix completion method doesn't know about this uncertainty, as it takes the data at face value. Showing a user the items with greatest predicted rating is the best move short term (knowing what we already know), but can lead to a self-reinforcing situation where the system stays ignorant about data that would otherwise lead it to give better predictions later.

17.1 Adding an exploration component

In reinforcement learning, there is a trade-off between doing what the current data suggests is the best move, and exploring uncertainty to get a better idea of what might be best in the future. The two extremes (only doing what seems best at any given point, or only reducing uncertainty) is called exploitation and exploration, respectively.

In that light, the problem with directly using a matrix completion method is that it employs too much exploitation. The problem can be reduced by introducing exploration, with the idea that balancing the two leads to better outcomes than focusing on either alone. And since exploration involves reducing uncertainty, it's natural to adopt a statistical approach for modeling that uncertainty.

Xing et al. [2014] set up a statistical model based on matrix completion and a forgetting model, and compare the results of using a greedy strategy entirely focused on exploitation to one that balances exploration and exploitation. The experiment shows that while the exploitation-only strategy starts off nearly as good as the balanced model, the users tire of the recommendations and the average rating for the recommended items suffer.

For simplicity, Xing et al. use a sequential recommendation setting: the user is given an item to consider (in their particular case, a song to listen to), and is then asked to rate that item before proceeding to the next. Such a setting may be useful for an internet radio or dynamic music playlist service, which serves one song at a time; or it may be used as the basis for something more complex.

Modeling the uncertainty of ratings To incorporate exploration, the authors start with a bilinear matrix factorization model of the type we've seen can be fitted by low rank matrix completion. They add a forgetting term to model how users may get tired of seeing the same item all the time, ending up with the following, in our notation:

$$\mathbb{E}[R_{ui}] = X_{ui} \quad = \quad (\boldsymbol{\theta_u} V_i^T)(1 - e^{-t_{ui}/s_u})$$

where R_{ui} is a random variable predicting the rating by user u for item i. Suppose $M = UV^T$ is the matrix factorization from ALS or nuclear norm minimization. Then $\boldsymbol{\theta_u}$ is a vector of random variables with a role analogous to U_u, so that θ_{uf} is a random variable giving how much user u likes factor f, while V_i^T is the (deterministic) vector giving how much item i exhibits each factor (V_{fi} gives how much item i exhibits factor f). t_{ui} is the number of seconds since user u visited item i, while s_u is a random variable giving the forgetting factor of user u: a higher forgetting factor means that the user doesn't have to stop listening to a song for very long before he starts liking it again, while a smaller forgetting factor implies that the user tires easily.

The authors' uncertainty model is thus user-focused: it implicitly says that we can't be sure of how much a given user likes a given factor f, but we are entirely sure of how much some item i exhibits f. This limits the model, but it seems the reason they do this is to not make the system too complex.

Determining the random variables The authors first use alternating least squares, solving (14.1) to obtain a matrix factorization $M = UV^T$, $U \in \mathbb{R}^{mxr}$, $V \in \mathbb{R}^{nxr}$, and after this they treat V as known. If we didn't have a forgetting term and θ was also to be deterministic, regularized linear regression would suffice to determine θ. But because θ is random and there is a forgetting term, things become considerably more complex.

To determine the random variables, the authors use Bayesian statistics with the following model:

$$\begin{aligned}
R_{ui} \mid V_i, t_{ui}, \boldsymbol{\theta_u}, s_u, \sigma_u^2 &\sim N\big((\boldsymbol{\theta_u} V_i^T)(1 - e^{-t_{ui}/s_u}), \sigma_u^2\big) \\
\boldsymbol{\theta_u} \mid \sigma_u^2 &\sim N(0, a_0 \sigma_u^2 I_r) \\
s_u &\sim Gamma(b_0, c_0) \\
\frac{1}{\sigma^2} &\sim Gamma(d_0, e_0)
\end{aligned}$$

Here I_r is the $r \times r$ identity matrix, and a_0, b_0, c_0, d_0, e_0 are global parameters to be decided by some other process (e.g. cross-validation). σ_u^2 is a random variable specific to the user.

$\boldsymbol{\theta_u}$ starts off with no information about what features the user likes, so the next step is for the system to have a way of updating this vector based on user behavior. The user's behavior is recorded in an ordered history set, one for each user. Let $\mathcal{D}_{uh}$ be the ordered set for user u up to the hth item. It is defined as follows:

$$\mathcal{D}_{uh} = \{(V_i^T, t_{ui}, r_{ui})\}_{i=1}^h$$

where V_i^T is the feature vector for the item being used at time t_{ui} by user u, and rated r_{ui} by that user afterwards. The incorporation of history then takes the form of posterior probability rules:

$$p((\boldsymbol{\theta_u}, s_u) \mid \mathcal{D}_{uh}) \propto p((\boldsymbol{\theta_u}, s_u))p(\mathcal{D}_{uh} \mid (\boldsymbol{\theta_u}, s_u))$$

and estimate R_{ui} by integrating over all possible $(\boldsymbol{\theta_u}, s_u)$:

$$p(R_{ui} \mid \mathcal{D}_{uh}) = \int p(R_{ui} \mid (\boldsymbol{\theta_u}, s_u))p((\boldsymbol{\theta_u}, s_u) \mid \mathcal{D}_{uh})$$

The authors first try to use MCMC for this purpose, but the method is too slow and they find another Gibbs-based approach that's much quicker.

Strategies and results With random variables R_{ui} in hand, the authors can now test two strategies:

- Pure exploitation: When the user is to be shown a new song to rate, choose the one with maximum expected rating.
- Balanced: Use a multi-armed bandit algorithm to balance exploration and exploitation, and choose the song it selects.

The authors also test an earlier balanced content-based strategy, but as that is not as relevant to the kind of recommendation system I'm investigating, I've chosen to omit that strategy here.

The multi-armed bandit algorithm (Bayes-UCB) in essence chooses items with greater variance (more uncertainty about ratings) more often, thus incorporating exploration into the method. The name comes from the imagined "multi-armed bandit" setting: a person, wishing to maximize return, is facing a row of slot machines that pay out at different rates. He has to determine which slot machine ("one-armed bandit") to play, taking into account both the known payouts so far (exploitation) and the uncertainty of the estimates (exploration). Details about how the Bayes-UCB algorithm works can be found in the paper.

Xing et al., p. 450 shows the results: the balanced strategies get consistently positive feedback from the users, while that of the greedy algorithm declines with time as it focuses too much on early good results. As the authors put it, the greedy algorithm "is quickly trapped at a local optima, repeatedly recommending the few songs with initial good ratings".

Like local low rank approximation, the balanced strategy thus appears to be an improvement upon plain matrix factorization, but for another reason. The balanced strategy escapes self-reinforcing situations and manages time preference better than the pure exploitation strategy.

However, it still has limitations. Since it only models users stochastically, it presumably would not do as well in a new item or cold-start setting. If a new song is released, we don't know the song's feature vector V_i^T until at least a few users have started rating that song; and it might make sense to recommend the song to more users than a user-based balanced strategy would, just to become more certain of what the feature vector is.

In statistical terms, that would entail modeling not just $\boldsymbol{\theta_u}$ as random variables, but also, say $\boldsymbol{\nu_i}$ analogous to V_i^T the way $\boldsymbol{\theta_u}$ is analogous to U_u. However, the simple bootstrapping strategy of first using an ordinary matrix completion algorithm to obtain V, and then estimate $\boldsymbol{\theta_u}$ based on that V, would fail, since the item feature vectors would depend on the user feature vectors and vice versa. It is thus understandable why the authors didn't stochastically model both users and items. Instead of using the bootstrapping strategy, a Bayesian analog of nuclear norm minimization would be needed.

18 Future work ideas

18.1 Search for other tractable subproblems of ℓ_p $p < 1$ optimization

While the ℓ_p optimization problem is NP-hard for $p < 1$, this only bounds its worst case complexity (assuming $P \neq NP$). This implies that there might exist subsets of problem instances for which ℓ_p optimization is easy, even though the optimization problem is NP-hard in full generality, and if we can determine such subsets, we can solve the ℓ_0 optimization problem with greater success.

The compressed sensing results for ℓ_1-optimization can be considered to determine a set of problem instances for which solving the ℓ_0 optimization problem is easy in this manner. Thus, research that extends the domain of compressed sensing (e.g. Adcock et al. 2017) could be considered progress in this area, as it discovers characteristics of more problem instances where ℓ_0 optimization is easy.

However, beyond this direct approach, we might make use of that the decision variant of the ℓ_0 optimization problem is in NP, so it can be reduced to any other NP-complete problem. A program that solves an NP-complete problem such as 3-CNF-SAT or Hamiltonian circuit often does better than its worst-case performance would suggest, and sometimes very much so: Applegate [2006] lists exact solution records for the traveling salesman problem, including a 24978 node instance that would have taken on the order of $24978^2 \cdot 2^{24978}$ operations using the simple Held-Karp algorithm.

We could thus explore potential feasible subsets by reducing the generally NP-complete ℓ_p-optimization problem with $p < 1$ into an NP-complete problem for which general solvers exist, and then investigate the performance of these solvers on the reduced problem. Possible things to investigate could include:

- Do the solvers do well on instances where the null space property holds or coherence is low? I.e. do they in some sense detect that these problems are easy to solve, as we know they are from compressed sensing theory?
- Do the solvers do well on instances where plain compressed sensing fails, but more recent results like Adcock et al. 2017 show that the problem instances can still be efficiently solved?
- Are there other unusual instances where the solvers do well but no theoretical results yet exist to indicate that the instances should be easy to solve?

Such investigation could be done both directly on ℓ_0 and on ℓ_p, $p < 1$. There might, for instance, be problem instances where ℓ_0-optimization is hard, but ℓ_p, $0 < p < 1$ is easy. Although the approach of solving ℓ_p-optimization would fail to ensure the optimally sparse solution in that case, we could still approach such a solution better than if we were limited to ℓ_1 optimization.

One problem with this approach is that we would have to know the ℓ_0 optima beforehand. Otherwise, solving the decision problem for ℓ_0 on some particular $A\mathbf{x} = \mathbf{y}$ could appear to be easy down to, say $\| \mathbf{x} \|_0 = 20$, and then the solvers fail to find any lower value for $\| \mathbf{x} \|_0$ in reasonable time. If we don't already know the minimum, we can't tell whether $\| \mathbf{x} \|_0 = 20$ is the true minimum or whether there is a better minimizer that the solver approach just doesn't find. This is due to that ℓ_0 optimization is in NP but not in $coNP$ unless $NP = coNP$.

Some work roughly along these lines can be found in Ge et al. [2011]. The paper gives an algorithm for finding a local optimum to the ℓ_p optimization problem for $p < 1$, and through numerical experiments, shows how often the local minima coincide with the ℓ_0 minimum.

Finally, according to Mazumder et al. [2010, p. 2290], the ℓ_1 optimum might be preferred to the ℓ_0 solution in certain statistical applications.

18.2 Balance recommender system exploitation in other ways

In 17.1, we investigated a collaborative filtering system that balanced exploration and exploitation to provide suggestions that the users wouldn't tire of as easily. This system shows that it's possible to balance the objectives of a recommender system away from an exclusive focus on short-term accuracy.

We also know that recommender systems can have an effect on the network structure, such as which users or items become popular [Su et al., 2016]. In light of concerns that personalized systems increase polarization, it could be fruitful to investigate whether recommender systems can be debiased to counter unintended dynamics while still providing high quality recommendations.

One general approach would be to, instead of maximizing accuracy, maximize a combination of accuracy and some other goal, reminiscent of how basis pursuit denoising balances minimizing the ℓ_1 norm of the solution $\mathbf{x}$, and the error term $\| A\mathbf{x} - \mathbf{y} \|_2^2$.

A most clear parallel to BPDN would be a denoising problem of the type:

$$\min_X \quad \alpha f(X) + \beta \parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel + \parallel X \parallel_*$$

with the associated hard constraint problem being

$$\begin{aligned} \min_X \quad & \alpha f(X) + \parallel X \parallel_* \\ \text{subject to} \quad & \parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel \quad \leq \delta \end{aligned}$$

in either case, with $f(X)$ being a measure that we want to minimize.

If we want to counteract polarization, $f(X)$ could be the degree to which there exist distinct groups of users, each of whom share only a few preferred item with other such groups of users; and if we want to balance out the dynamic that popular items get more popular, $f(X)$ could be an inequality measure on the items. In the former case, the parameter α lets us balance between emphasizing cross-cutting items (i.e. items that otherwise disparate groups like) and short-term accuracy; while in the latter, increasing α would promote less known items to users to reduce the rich-get-richer effect.

Jaggi and Sulovský [2010] gives an algorithm for minimizing a Lasso-type constraint for any convex loss function, i.e

$$\min_X \qquad f(X) + \lambda \parallel X \parallel_*$$

for any convex f. This gives considerable freedom as to how to design a debiasing term, as long as f itself is convex.

I'll give two examples of using Markov matrices to counter unintended dynamics:

Rich-get-richer effect Suppose we have a collaborative publishing system with m users and n items, with each item having an author (user) associated with it. Users also rate items on a scale from 0 to r inclusive. We thus have a (partially observed) ratings matrix $Y \in \mathbb{R}^{m \times n}$ that we want to extrapolate into a fully known matrix X using a recommender system.

We also have a fully observed authorship matrix $A \in \mathbf{B}^{m \times n}$ where $A_{ij} = 1$ if user i produced item j, 0 otherwise. Let $a(j)$ be the i so that $A_{ij} = 1$, i.e. $a(j)$ denotes the index of the author of j.

To analyze the dynamics, we can then create a simple statistical model of a user that acts somewhat similar to how most users do in aggregate. Drawing from Page et al. [1998], we create a random surfer model. While no particular user acts like a random surfer (either ours or Google's), the random surfer may, in the vicinity of any user u, locally emulate behavior of users who are interested in what u produces. Thus, in aggregate (over all users), general traffic may behave similar to such a model.

When starting at some user i, our random surfer visits one of i's recommendations with probability proportional to the value the recommender system assigns that item, then visits one of the author's recommendations, then one of that recommendation's author's recommendations and so on. The Markov chain produced by this model is

$$p(t+1 = j \mid t = i) \quad = \quad \frac{X_{a(i),j}}{\sum_{k=1}^{n} X_{a(i),k}}$$

We may add a slight offset to this in order to make the Markov chain easier to analyze, so that with some probability p_{rand}, the user just chooses at random. This gives

$$p(t+1=j \mid t=i) \quad = \quad \frac{p_{rand}}{n} + (1-p_{rand})\frac{X_{a(i),j}}{\sum_{k=1}^{n} X_{a(i),k}}$$

The Markov chain has an associated matrix representation, which by convention is right stochastic. Let $M \in \mathbb{R}^{m\times m}$ be this matrix, defined so that $M_{ij} = p(t+1=j \mid t=i)$. The convention in Markov chain analysis in statistics is to assume that vectors are row vectors, but here I will, to be consistent with the rest of the thesis, keep using column vectors.

The stationary probabilities of the Markov chain defined by M will be less even the more inherent popularity exists. If we want to moderate the influence of popularity upon recommendations, we then need a function that measures this unevenness or dispersion.

Since the stationary probabilities are indeed probabilities, we can use information theory entropy: $H(\boldsymbol{\pi}) = -\sum_{i=1}^{n} \pi_i \log(\pi_i)$. This measures surprise, so it's maximized when we have a flat distribution. Thus, if we're minimizing $\alpha\, f(X) + \beta \parallel \mathcal{A}(X) - \mathcal{A}(Y) \parallel + \parallel X \parallel_*$, we need to negate H for our f.

As the stationary probabilities of the chain are given by the principal eigenvector (with eigenvalue 1), we end up with $f(X)$ being $-H(\mathbf{v_1})$, where $\mathbf{v_1}$ is the principal eigenvector of the Markov matrix for X and the model above.

Some problems with this, which would have to be investigated, include:

- $f(X)$ might not be convex. The easiest way of recovering $\mathbf{v_1}$ is probably to solve the equation for a stationary distribution: $\mathbf{v_1}^T M = \mathbf{v_1}^T$. But since we're optimizing over both M (indirectly by optimizing X) and $\mathbf{v_1}$, this turns into a set of quadratic constraints and may not be convex.
- M is not linearly derived from X - the problem is the normalizing term $\sum_{k=1}^{n} X_{a(i),k}$. This might introduce another nonconvexity.

If that doesn't work, a much simpler model would be to nudge recommendation values down according to popularity. Let $\mathbf{p} \in \mathbb{N}^m$ be a count of the number of times each item has been viewed so far, then let $D \in \mathbb{R}^{m\times n}$ be defined by $D_{ij} = X_{ij} \cdot \frac{p_j}{\sum_i p_i}$. We can then set $f(X) = \sum_i \sum_j D_{ij}$ as the additional component to incorporate into the objective to minimize.

Since $\mathbf{p}$ only uses known values, the optimization problem should be much easier than the one involving Markov matrices. Nonlinear transformations of $\mathbf{p}$ (e.g. centering, normalizing) are also possible without making the optimization problem any harder. The two models mainly differ in that using the stationary probabilities as an objective would not only deweight popular items, but also serve less popular items to more popular users, so that when the popular users tell other users what they've seen, this gives an additional boost to otherwise less popular items.

Community polarization Suppose we have a social network with n users who are also items, with a follow set $\Omega = \{(i,j) : \text{i follows j}\}$, and the partially observed follow matrix $Y \in \mathbb{R}^{n\times n}$ defined by

$$Y_{ij} = \begin{cases} 1 & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

This is an implicit recommender setting, as given in section 12: we know whether user i follows user j, but if user i doesn't follow j, we don't know if that's because they don't know of one another or i dislikes j. So let $\mathcal{A}(K)_{an+b} = C_{ab}K_{ab}$ for a matrix K, and C defined as $C = \tau Y$ for an uncertainty constant τ which gives how much i not following j should count as i disliking j.

The recommender system is meant to extrapolate from Y to a fully known matrix X. For any such completed X we can define a Markov matrix M similar to above, but the model is simpler than in the last example: the notional random surfer picks a random user according to how likely it is that the current user would follow that user, i.e.

$$p(t+1 = j \mid t = i) = \frac{p_{rand}}{n} + (1 - p_{rand})\frac{X_{i,j}}{\sum_{k=1} X_{i,k}}$$

The Markov model could then be used to measure polarization, by reasoning that it'll take a longer time for the random surfer to visit every user if there's a lot of polarization in the network. Another way of saying this is that if groups keep mostly to themselves, there will be bottlenecks between the groups. Formally, we would expect the mixing time of the Markov chain to be longer if there are few between-group links than if there are many.

A standard way of determining the stationary distribution of a Markov chain is to use the power method. Let $\boldsymbol{\pi}$ be the stationary probability vector ($\mathbf{v_1}$ in the previous example). If M is diagonalizable and positive, we have that

$$\lim_{t\to\infty} \mathbf{1}^T M^t = \boldsymbol{\pi}^T$$

with $\mathbf{1}$ the vector of all ones.

Suppose M is diagonalizable. Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of M with $\lambda_1 = 1 > \mid \lambda_2 \mid \geq \dots \geq \mid \lambda_n \mid$, and the associated eigenvectors be $\mathbf{v_1}, \dots, \mathbf{v_n}$. Then there exist $c_1, \dots, c_n$ so that $\mathbf{1} = \sum_{k=1}^n c_k \mathbf{v_k}$. Hence

$$\begin{aligned} \mathbf{1}^T M^t &= \sum_{i=1}^n c_i \mathbf{v_i}^T M^t \\ &= \sum_{i=1}^n c_i \mathbf{v_i}^T \lambda_i^t \\ &= c_1 \boldsymbol{\pi}^T + \sum_{i=2}^n c_i \mathbf{v_i}^T \lambda_i^t \end{aligned}$$

which is a combination of the desired stationary probability vector $\mathbf{v_1} = \boldsymbol{\pi}$ and some error term $\sum_{i=2}^{n} c_i \mathbf{v_i}^T \lambda_i^t \leq\mid \lambda_2 \mid^t \sum_{i=2}^{n} c_i \mathbf{v_i}^T$. Since $\mid \lambda_2 \mid< 1$, the error term converges to zero as $t \to \infty$, and how quickly it does so depends on the magnitude of λ_2.

Hence $\mid \lambda_2 \mid$ is directly related to the rate of convergence of the power method, with the intuition above being that for polarized networks, this rate is slower than for evenly mixed ones. So we can use $f(X) =\mid \lambda_2 \mid$ as a measure of polarization, or equivalently $f(X) =\mid \lambda_1 \mid + \mid \lambda_2 \mid$. As $\lambda_1 = 1$ for all Markov matrices, this doesn't change anything as far as optimization goes.

I have not been able to verify that $f(X)$ is convex over all right stochastic matrices X, but I haven't been able to find any counterexamples either. The second issue above, that M is not linearly derived from X, still remains, and I haven't proven that the Markov matrices we're considering are always diagonalizable either. If they're not, some other approach has to be done to show that convergence time still depends on $\mid \lambda_2 \mid$. It is known that not all positive Markov matrices are diagonalizable, see e.g. Myerson [2013] .

The study of properties of a graph by eigenvalues and eigenvectors is called spectral graph theory. There may be results within spectral graph theory that would permit us to use X directly, or use some transformation of X that would be convex, if the Markov chain transformation turns out to be nonconvex. Boyd [2006] gives some results for undirected graphs; however, social network graphs are not undirected. Furthermore, the extrapolated matrix X provides us with uncertain edges: if $Y_{ij} = 1$ whenever user i follows user j, then $0 < X_{jk} < 1$ implies that the system thinks there's some chance that user j would follow user k if made aware of k. As standard graph theory does not consider uncertain edges, it may have to be adapted to our setting.

Another approach to reducing polarization can be found in Musco et al., 2017. Their approach is based on a concrete opinion dynamics model, and they construct an optimization problem that balances disagreement (analogous to the lack of accuracy) and polarization. However, the problem is only convex for a particular balance of the two.

18.3 Harden recommender systems against adversarial noise

The use of recommender systems in settings where the same users consume and provide content may induce undesirable incentives that distort the data. For instance, a publisher on a video sharing site might create a fake user who highly rates videos a particular group likes, and then in addition highly rates the publisher's own videos. Under a simple user-user similarity recommender system, this would lead the system to recommend the publisher's videos to that group, if the group is not too large. Matrix factorization methods might be susceptible to more advanced strategies.

To discourage tactical rating in a matrix factorization context, we could consider the matrix to not only be incomplete, but to consider some proportion of its revealed entries to also be arbitrarily corrupted, i.e. be adversarially chosen as to minimize the accuracy of the recommender system itself. Assuming the corrupted entries are chosen that way works as a worst case assumption, since any other strategy will (by definition) not degrade the system as much.

A simple, yet potentially impractical approach is to run the recommender system $\binom{r}{(1-p)r}$ times, each time excluding pr of the r known entries, and choosing the final predicted matrix to be the returned matrix that minimizes the sum of distances to every other predicted matrix, according to some metric $d(A, B)$, e.g. $d(A, B) = \| A - B \|$. More formally, that can be cast as an optimization problem for the final predicted matrix M based on the observed set of ratings Ω and partial matrix A. Let λ be specified ahead of time and

$$\begin{aligned} \Phi &= \{X \subset \Omega : | X | = \lfloor pr \rfloor\} \\ f(P, X) &= \arg\min_X \| P(X) - X \| + \lambda \| X \|_* \end{aligned}$$

and then the optimal solution is

$$\begin{aligned} \phi_{opt} &= \arg\min_{\tilde{\phi}} \sum_{\phi \in \Phi \setminus \{\tilde{\phi}\}} \| f(P_{\Omega \setminus \phi}, A) - f(P_{\Omega \setminus \tilde{\phi}}, A) \| \\ M &= f(P_{\Omega \setminus \phi_{opt}}, A) \end{aligned}$$

However, this might be completely impractical: there may not be any good ways to simplify the problem to a poly-time algorithm; and we don't know the robustness guarantees analogous to the breakdown point of a robust estimator.

Another approach is to minimize the rank or nuclear norm subject to that a finite number of observations can be attenuated or discarded. This approach is sometimes used in background extraction problems in image processing, where the assumption is that the background can be represented by a low rank matrix, whereas the foreground will interrupt the natural regularity of that background and thus will be discarded by the algorithm. See for instance Xiong et al. [2011]. Background extraction rarely involves missing values, however, and so focuses on just finding outlier points, not on both finding outliers while completing missing values. The closest I've found is Shang et al. [2015], which investigates the optimization problem

$$\min_{L, S} \quad \| S \|_1 + \lambda \| L \|_* \quad \text{subject to } \mathcal{A}(L + S) = \mathcal{A}(X)$$

for a given observed matrix X, where $\| \cdot \|_1$ is the entrywise ℓ_1 norm. This problem handles a small number of corrupted entries by creating an "adjustment matrix" L that cancels out these entries, but it assumes that $L + S$ is exactly low rank. As mentioned earlier, this may not be a good assumption for real world data, and so a natural extension would be to introduce approximate matching, e.g.

$$\min_{L, S} \| S \|_1 + \lambda_1 \| L \|_* + \lambda_2 \| \mathcal{A}(L + S) - \mathcal{A}(X) \|$$

Note that unless there is some way of limiting the number of users that a real person can create, or unless we can make each user creation costly, all recommender systems are arbitrarily manipulable in the worst case. A malicious user can simply create as many users as he desires, so that the real uncorrupted users constitute a vanishingly small proportion of the total number of users. The malicious user can then fill the fake users with information to make the recommender output what he wants it to output.

19 Summary and conclusion

19.1 Summary

In the first part of this thesis, we reviewed the basic theory of compressed sensing, focusing on the fundamental problems: cardinality minimization and basis pursuit. We investigated complexity theory to get the tools required to say whether a problem is asymptotically easy or hard to calculate on a computer, and then continued by proving a necessary and sufficient property for compressed sensing recovery, namely the null space property. Finally, we considered how to handle noise, and found tractable approximations to the null space property.

The second part continued where the first left off, by taking compressed sensing into the matrix domain in the form of low rank matrix sensing and completion. We used the complexity theory tools to show that the nonconvex matrix problems are at least as worst-case hard as the analogous vector problems, which motivated a convex relaxation by using the same strategy as for vectors: relaxing a nonconvex operator to a norm. We showed how the relaxed problem could be written as a semidefinite program, and, following the layout of the first part, investigated generalizations of the null space property. We then moved onto matrix completion by giving hardness results for that setting in particular.

In the third part, we focused on matrix completion and gave a connection between this setting and a model used for recommender systems: the bilinear matrix factorization (or latent factor model). Even though that is a simple model, it's been useful in recommender systems. After doing so, we compared the analytical elegance of exact nuclear norm minimization to the more pragmatic world of actual recommender systems, the former exemplified by semidefinite programming and the latter by the alternating least squares algorithm. We found the former easier to analyze but the latter faster in practice.

The fourth part moved beyond matrix factorization and latent factor models to consider how recommender systems might be improved by extending its scope. We investigated two papers with different suggestions: the very well performing local low rank approximation system, and the balanced exploration system that used statistical modeling to provide novel recommendations. Finally we saw some ideas for future work: to use solvers of computer science problems to guide the search for other easy problems in compressed sensing, and ideas for discouraging unintended consequences that may otherwise result from the use of recommender systems. The latter contained ideas of how to measure, and counter, polarization in social networks.

19.2 Concluding remarks

I have learned quite a bit of compressed sensing while writing this thesis. There are some results, however, that are lurking just beneath the surface, that I have not been able to draw fully out into the open.

For instance, there seems to be a parallel between the matrix completion problem and statistical recovery in compressed sensing. These are results similar to (13), which involve observing $\mathbf{x} \cdot \psi_j$ with j chosen at random according to some distribution. In that setting, we don't know ahead of time which ψ_j are going to be the bases for our observations, in a similar way that we can't pin down the exact structure of P_Ω in advance for matrix completion. Since I've focused on deterministic compressed sensing and thus on the null space property rather than on the RIP, I can't draw the parallel as well as I would like.

There is also still a gap between empirical results and analytical bounds. To quote Mazumder et al. [2010], "We note however that the conditions under which the nuclear-norm regularization is theoretically meaningful are not met on the Netflix data set". Hence we should be aware that more research is needed before we can accurately predict the performance of recommender systems algorithms on real datasets, or even how well the methods can capture the linear component of user behavior.

References

Ben Adcock, Anders C Hansen, Clarice Poon, and Bogdan Roman. Breaking the coherence barrier: A new theory for compressed sensing. In *Forum of Mathematics, Sigma*, volume 5. Cambridge University Press, 2017.

Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM Journal on Computing*, 38(5):1987–2006, 2009.

David L Applegate. *The traveling salesman problem: a computational study.* Princeton university press, 2006.

Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

Stephen Boyd. Convex optimization of graph laplacian eigenvalues. In *Proceedings of the International Congress of Mathematicians*, volume 3, pages 1311–1319, 2006.

Jameson Cahill, Xuemei Chen, and Rongrong Wang. The gap between the null space property and the restricted isometry property. *Linear Algebra and its Applications*, 501:363–375, 2016.

Emmanuel J. Candès and Yaniv Plan. A probabilistic and ripless theory of compressed sensing. *CoRR*, abs/1011.3854, 2010. URL `http://arxiv.org/abs/1011.3854`.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *CoRR*, abs/0805.4471, 2008. URL `http://arxiv.org/abs/0805.4471`.

Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

Rodrigo de Azevedo. Spectral norm minimization via semidefinite programming. Mathematics Stack Exchange, 2017. URL `https://math.stackexchange.com/q/2137408`. URL:https://math.stackexchange.com/q/2137408 (version: 2017-04-26).

Krishnamurthy Dvijotham and Maryam Fazel. A nullspace analysis of the nuclear norm heuristic for rank minimization. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 3586–3589. IEEE, 2010.

Simon Foucart and Holger Rauhut. *A mathematical introduction to compressive sensing*, volume 1. Birkhäuser Basel, 2013.

Jean Jacques Fuchs. More on sparse representations in arbitrary bases. *IFAC Proceedings Volumes*, 36(16):1315–1320, 2003.

Dongdong Ge, Xiaoye Jiang, and Yinyu Ye. A note on the complexity of lp minimization. *Mathematical programming*, 129(2):285–299, 2011.

Nicolas Gillis and François Glineur. Low-rank matrix approximation with weights or missing data is np-hard. *SIAM Journal on Matrix Analysis and Applications*, 32(4): 1149–1165, 2011.

Matthew Gray. Measuring the growth of the web. `http://www.mit.edu/people/mkgray/growth/`, 6–Aug 1996. [Online; accessed 26-May-2018].

Dhruv Gupta, Mark Digiovanni, Hiro Narita, and Ken Goldberg. Jester 2.0 (poster abstract): Evaluation of an new linear time collaborative filtering algorithm. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 291–292, New York, NY, USA, 1999. ACM. ISBN 1-58113-096-1. doi: 10.1145/312624.312718. URL `http://doi.acm.org/10.1145/312624.312718`.

Moritz Hardt. Understanding alternating minimization for matrix completion. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 651–660. IEEE, 2014.

Christian Hoene, Jean-Marc Valin, Koen Vos, and Jan Skoglund. Summary of opus listening test results. Internet-Draft draft-ietf-codec-results-03, IETF Secretariat, May 2013. URL `https://tools.ietf.org/html/draft-ietf-codec-results-03`.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

Martin Jaggi and Marek Sulovský. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 471–478, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL `http://dl.acm.org/citation.cfm?id=3104322.3104383`.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

Brian Kulis, Arun C. Surendran, and John C. Platt. Fast low-rank semidefinite programming for embedding and clustering. In Marina Meila and Xiaotong Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 235–242, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR. URL `http://proceedings.mlr.press/v2/kulis07a.html`.

Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local low-rank matrix approximation. In *International Conference on Machine Learning*, pages 82–90, 2013.

OL Mangasarian. Absolute value programming. *Computational Optimization and Applications*, 36(1):43–53, 2007.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322, 2010.

Cameron Musco, Christopher Musco, and Charalampos E Tsourakakis. Minimizing polarization and disagreement in social networks. *arXiv preprint arXiv:1712.09948*, 2017.

Gerry Myerson. Example of a markov chain transition matrix that is not diagonalizable? Mathematics Stack Exchange, 2013. URL `https://math.stackexchange.com/q/332700`. URL:https://math.stackexchange.com/q/332700 (version: 2015-04-10).

Netcraft. April 2018 web server survey. `https://news.netcraft.com/archives/2018/04/26/april-2018-web-server-survey.html`, 24–Apr 2018. [Online; accessed 26-May-2018].

Samet Oymak and Babak Hassibi. New null space results and recovery thresholds for matrix rank minimization. *arXiv preprint arXiv:1011.6326*, 2010.

Samet Oymak, Karthik Mohan, Maryam Fazel, and Babak Hassibi. A simplified approach to recovery conditions for low rank matrices. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 2318–2322. IEEE, 2011.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1998.

Davis Pan. A tutorial on mpeg/audio compression. *IEEE MultiMedia*, 2(2):60–74, June 1995. ISSN 1070-986X. doi: 10.1109/93.388209. URL `http://dx.doi.org/10.1109/93.388209`.

Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52 (3):471–501, 2010.

Fanhua Shang, Yuanyuan Liu, Hanghang Tong, James Cheng, and Hong Cheng. Robust bilinear factorization with missing and grossly corrupted observations. *Information Sciences*, 307:53–72, 2015.

Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *IEEE Internet Computing*, 21(3):12–18, 2017.

Jessica Su, Aneesh Sharma, and Sharad Goel. The effect of recommendations on network structure. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1157–1167. International World Wide Web Conferences Steering Committee, 2016.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Andreas M Tillmann and Marc E Pfetsch. The computational complexity of the restricted isometry property, the nullspace property, and related concepts in compressed sensing. *IEEE Transactions on Information Theory*, 60(2):1248–1259, 2014.

Martijn van Beurden. Lossless audio codec comparison, August 2013. URL `https://xiph.org/flac/comparison.pdf`.

Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38 (1):49–95, 1996.

Rick Wicklin. How to visualize a kernel density estimate. The DO Loop, SAS blog, 27 July 2016. URL `https://blogs.sas.com/content/iml/2016/07/27/visualize-kernel-density-estimate.html`.

Zhe Xing, Xinxi Wang, and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *Ismir*, pages 445–450, 2014.

Liang Xiong, Xi Chen, and Jeff Schneider. Direct robust matrix factorizatoin for anomaly detection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 844–853. IEEE, 2011.

YouTube. Press - youtube. `https://www.youtube.com/intl/en-GB/yt/about/press/`, 2018. [Online; accessed 26-May-2018].