Foundations and Trends® in Machine Learning
Vol. 10, No. 3-4 (2017) 142–336

DOI: 10.1561/2200000058

Non-convex Optimization for Machine Learning

Prateek Jain
Microsoft Research India
prajain@microsoft.com

Purushottam Kar
IIT Kanpur
purushot@cse.iitk.ac.in

Contents

Abstract

A vast majority of machine learning algorithms train their models and perform inference by solving optimization problems. In order to capture the learning and prediction problems accurately, structural constraints such as sparsity or low rank are frequently imposed or else the objective itself is designed to be a non-convex function. This is especially true of algorithms that operate in high-dimensional spaces or that train non-linear models such as tensor models and deep networks.

The freedom to express the learning problem as a non-convex optimization problem gives immense modeling power to the algorithm designer, but often such problems are NP-hard to solve. A popular workaround to this has been to relax non-convex problems to convex ones and use traditional methods to solve the (convex) *relaxed* optimization problems. However this approach may be lossy and nevertheless presents significant challenges for large scale optimization.

On the other hand, direct approaches to non-convex optimization have met with resounding success in several domains and remain the methods of choice for the practitioner, as they frequently outperform relaxation-based techniques – popular heuristics include projected gradient descent and alternating minimization. However, these are often poorly understood in terms of their convergence and other properties.

This monograph presents a selection of recent advances that bridge a long-standing gap in our understanding of these heuristics. We hope that an insight into the inner workings of these methods will allow the reader to appreciate the unique marriage of task structure and generative models that allow these heuristic techniques to (provably) succeed. The monograph will lead the reader through several widely used non-convex optimization techniques, as well as applications thereof. The goal of this monograph is to both, introduce the rich literature in this area, as well as equip the reader with the tools and techniques needed to analyze these simple procedures for non-convex problems.

P. Jain and P. Kar. *Non-convex Optimization for Machine Learning*. Foundations and Trends® in Machine Learning, vol. 10, no. 3-4, pp. 142–336, 2017.
DOI: 10.1561/2200000058.

Preface

Optimization as a field of study has permeated much of science and technology. The advent of the digital computer and a tremendous subsequent increase in our computational prowess has increased the impact of optimization in our lives. Today, tiny details such as airline schedules all the way to leaps and strides in medicine, physics and artificial intelligence, all rely on modern advances in optimization techniques.

For a large portion of this period of excitement, our energies were focused largely on convex optimization problems, given our deep understanding of the structural properties of convex sets and convex functions. However, modern applications in domains such as signal processing, bio-informatics and machine learning, are often dissatisfied with convex formulations alone since there exist non-convex formulations that better capture the problem structure. For applications in these domains, models trained using non-convex formulations often offer excellent performance and other desirable properties such as compactness and reduced prediction times.

Examples of applications that benefit from non-convex optimization techniques include gene expression analysis, recommendation systems, clustering, and outlier and anomaly detection. In order to get satisfactory solutions to these problems, that are scalable and accurate, we require a deeper understanding of non-convex optimization problems that naturally arise in these problem settings.

Such an understanding was lacking until very recently and non-convex optimization found little attention as an active area of study, being regarded as intractable. Fortunately, a long line of works have recently led areas such as computer science, signal processing, and statistics to realize that the general abhorrence to non-convex optimization problems hitherto practiced, was misled. These works demonstrated in a beautiful way, that although non-convex optimization problems do suffer from intractability in general, those that arise in *natural settings* such as machine learning and signal processing, possess additional structure that allow the intractability results to be circumvented.

The first of these works still religiously stuck to convex optimization as the method of choice, and instead, sought to show that certain classes of non-convex problems which possess suitable additional structure as offered by natural instances of those problems, could be converted to convex problems without any loss. More precisely, it was shown that the original non-convex problem and the modified convex problem possessed a common optimum and thus, the solution to the convex problem would automatically solve the non-convex problem as well! However, these approaches had a price to pay in terms of the time it took to solve these so-called *relaxed* convex problems. In several instances, these relaxed problems, although not intractable to solve, were nevertheless challenging to solve, at large scales.

It took a second wave of still more recent results to usher in provable non-convex optimization techniques which abstained from relaxations, solved the non-convex problems in their native forms, and yet seemed to offer the same quality of results as relaxation methods did. These newer results were accompanied with a newer realization that, for a range of domains such as sparse recovery, matrix completion and robust learning, these direct techniques are faster, often by an order of magnitude or more, than their relaxation-based cousins.

This monograph wishes to tell the story of this realization and the wisdom we gained from it from the point of view of machine learning and signal processing applications. The monograph will introduce the reader to a lively world of non-convex optimization problems with rich structure that can be exploited to obtain extremely scalable solutions to these problems. Put a bit more dramatically, it will seek to show how problems that were once avoided, having been shown to be NP-hard to solve, now have solvers that operate in near-linear time, by carefully analyzing and exploiting additional task structure! It will seek to inform the reader on how to look for such structure in diverse application areas, as well as equip the reader with a sound background in fundamental tools and concepts required to analyze such problem areas and come up with newer solutions.

How to use this monograph We have made efforts to make this monograph as self-contained as possible while not losing focus of the main topic of non-convex optimization techniques. Consequently, we have devoted entire sections to present a tutorial-like treatment to basic concepts in convex analysis and optimization, as well as their non-convex counterparts. As such, this monograph can be used for a semester-length course on the basics of non-convex optimization with applications to machine learning.

On the other hand, it is also possible to cherry pick portions of the monograph, such the section on sparse recovery, or the EM algorithm, for inclusion in a broader course. Several courses such as those in machine learning, optimization, and signal processing may benefit from the inclusion of such topics. However, we advise that relevant background sections (see Figure 1) be covered beforehand.

While striving for breadth, the limits of space have constrained us from looking at some topics in much detail. Examples include the construction of design matrices that satisfy the RIP/RSC properties and pursuit style methods, but there are several others. However, for all such omissions, the bibliographic notes at the end of each section can always be consulted for references to details of the omitted topics. We have also been unable to address several application areas such as dictionary learning, advances in low-rank tensor decompositions, topic modeling and community detection in graphs but have provided pointers to prominent works in these application areas too.

The organization of this monograph is outlined below with Figure 1 presenting a suggested order of reading the various sections.

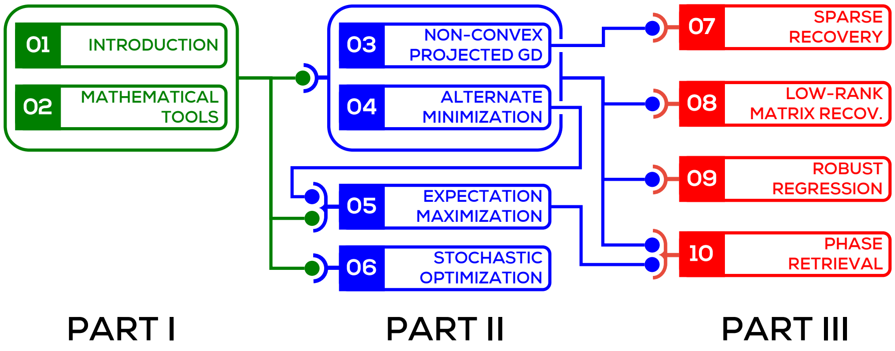

01 INTRODUCTION
02 MATHEMATICAL TOOLS
03 NON-CONVEX PROJECTED GD
04 ALTERNATE MINIMIZATION
05 EXPECTATION MAXIMIZATION
06 STOCHASTIC OPTIMIZATION
07 SPARSE RECOVERY
08 LOW-RANK MATRIX RECOV.
09 ROBUST REGRESSION
10 PHASE RETRIEVAL
PART I
PART II
PART III

Figure 1: A schematic showing the suggested order of reading the sections. For example, concepts introduced in § 3 and 4 are helpful for § 9 but a thorough reading of § 6 is not required for the same. Similarly, we recommend reading § 5 after going through § 4 but a reader may choose to proceed to § 7 directly after reading § 3.

Part I: Introduction and Basic Tools

This part will offer an introductory note and a section exploring some basic definitions and algorithmic tools in convex optimization. These sections are recommended to readers not intimately familiar with basics of numerical optimization.

Section 1 - Introduction This section will give a more relaxed introduction to the area of non-convex optimization by discussing applications that motivate the use of non-convex formulations. The discussion will also clarify the scope of this monograph.

Section 2 - Mathematical Tools This section will set up notation and introduce some basic mathematical tools in convex optimization. This section is basically a handy repository of useful concepts and results and can be skipped by a reader familiar with them. Parts of the section may instead be referred back to, as and when needed, using the cross-referencing links in the monograph.

Part II: Non-convex Optimization Primitives

This part will equip the reader with a collection of primitives most widely used in non-convex optimization problems.

Section 3 - Non-convex Projected Gradient Descent This section will introduce the simple and intuitive projected gradient descent method in the context of non-convex optimization. Variants of this method will be used in later sections to solve problems such as sparse recovery and robust learning.

Section 4 - Alternating Minimization This section will introduce the principle of alternating minimization which is widely used in optimization problems over two or more (groups of) variables. The methods introduced in this section will be later used in later sections to solve problems such as low-rank matrix recovery, robust regression, and phase retrieval.

Section 5 - The EM Algorithm This section will introduce the EM algorithm which is a widely used optimization primitive for learning problems with latent variables. Although EM is a form of alternating minimization, given its significance, the section gives it special attention. This section will discuss some recent advances in the analysis and applications of this method and look at two case studies in learning Gaussian mixture models and mixed regression to illustrate the algorithm and its analyses.

Section 6 - Stochastic Non-convex Optimization This section will look at some recent advances in using stochastic optimization techniques for solving optimization problems with non-convex objectives. The section will also introduce the problem of tensor factorization as a case study for the algorithms being studied.

Part III - Applications
This part will take a look at four interesting applications in the areas of machine learning and signal processing and explore how the non-convex optimization techniques introduced earlier can be used to solve these problems.

Section 7 - Sparse Recovery This section will look at a very basic non-convex optimization problem, that of performing linear regression to fit a sparse model to the data. The section will discuss conditions under which it is possible to do so in polynomial time and show how the non-convex projected gradient descent method studied earlier can be used to offer provably optimal solutions. The section will also point to other techniques used to solve this problem, as well as refer to extensions and related results.

Section 8 - Low-rank Matrix Recovery This section will address the more general problem of low rank matrix recovery with specific emphasis on low-rank matrix completion. The section will gently introduce low-rank matrix recovery as a generalization of sparse linear regression that was studied in the previous section and then move on to look at matrix completion in more detail. The section will apply both the non-convex projected gradient descent and alternating minimization methods in the context of low-rank matrix recovery, analyzing simple cases and pointing to relevant literature.

Section 9 - Robust Regression This section will look at a widely studied area of machine learning, namely robust learning, from the point of view of regression. Algorithms that are robust to (adversarial) corruption in data are sought after in several areas of signal processing and learning. The section will explore how to use the projected gradient and alternating minimization techniques to solve the robust regression problem and also look at applications of robust regression to robust face recognition and robust time series analysis.

Section 10 - Phase Retrieval This section will look at some recent advances in the application of non-convex optimization to phase retrieval. This problem lies at the heart of several imaging techniques such as X-ray crystallography and electron microscopy. A lot remains to be understood about this problem and existing algorithms often struggle to cope with the retrieval problems presented in practice.

The area of non-convex optimization has considerably widened in both scope and application in recent years and newer methods and analyses are being proposed at a rapid pace. While this makes researchers working in this area extremely happy, it also makes summarizing the vast body of work in a monograph such as this, more challenging. We have striven to strike a balance between presenting results that are the best known, and presenting them in a manner accessible to a newcomer. However, in all cases, the bibliography notes at the end of each section do contain pointers to the state of the art in that area and can be referenced for follow-up readings.

Prateek Jain, Bangalore, India
Purushottam Kar, Kanpur, India
December 2, 2017

Mathematical Notation

- The set of real numbers is denoted by $\mathbb{R}$. The set of natural numbers is denoted by $\mathbb{N}$.

- The cardinality of a set S is denoted by $|S|$.

- Vectors are denoted by boldface, lower case alphabets for example, $\mathbf{x}, \mathbf{y}$. The zero vector is denoted by $\mathbf{0}$. A vector $\mathbf{x} \in \mathbb{R}^p$ will be in column format. The transpose of a vector is denoted by $\mathbf{x}^\top$. The i^{th} coordinate of a vector $\mathbf{x}$ is denoted by $\mathbf{x}_i$.

- Matrices are denoted by upper case alphabets for example, A, B. A_i denotes the i^{th} column of the matrix A and A^j denotes its j^{th} row. A_{ij} denotes the element at the i^{th} row and j^{th} column.

- For a vector $\mathbf{x} \in \mathbb{R}^p$ and a set $S \subset [p]$, the notation $\mathbf{x}_S$ denotes the vector $\mathbf{z} \in \mathbb{R}^p$ such that $\mathbf{z}_i = \mathbf{x}_i$ for $i \in S$, and $\mathbf{z}_i = 0$ otherwise. Similarly for matrices, A_S denotes the matrix B with $B_i = A_i$ for $i \in S$ and $B_i = \mathbf{0}$ for $i \neq S$. Also, A^S denotes the matrix B with $B^i = A^i$ for $i \in S$ and $B^i = \mathbf{0}^\top$ for $i \neq S$.

- The support of a vector $\mathbf{x}$ is denoted by $\operatorname{supp}(x) := \{i : \mathbf{x}_i \neq 0\}$. A vector x is referred to as s-sparse if $|\operatorname{supp}(x)| \leq s$.

- The canonical directions in $\mathbb{R}^p$ are denoted by $\mathbf{e}_i,\ i = 1, \ldots, p$.

- The identity matrix of order p is denoted by $I_{p\times p}$ or simply I_p. The subscript may be omitted when the order is clear from context.

- For a vector $\mathbf{x} \in \mathbb{R}^p$, the notation $\left\|x\right\|_q = \sqrt[q]{\sum_{i=1}^{p} |\mathbf{x}_i|^q}$ denotes its L_q norm. As special cases we define $\left\|\mathbf{x}\right\|_\infty := \max_i \ |\mathbf{x}_i|$, $\left\|\mathbf{x}\right\|_{-\infty} := \min_i \ |\mathbf{x}_i|$, and $\left\|\mathbf{x}\right\|_0 := |\mathrm{supp}(\mathbf{x})|$.

- Balls with respect to various norms are denoted as $\mathcal{B}_q(r) := \left\{\mathbf{x} \in \mathbb{R}^p, \left\|\mathbf{x}\right\|_q \leq r\right\}$. As a special case the notation $\mathcal{B}_0(s)$ is used to denote the set of s-sparse vectors.

- For a matrix $A \in \mathbb{R}^{m\times n}$, $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_{\min\{m,n\}}(A)$ denote its singular values. The Frobenius norm of A is defined as $\left\|A\right\|_F := \sqrt{\sum_{i,j} A_{ij}^2} = \sqrt{\sum_i \sigma_i(A)^2}$. The nuclear norm of A is defined as $\left\|A\right\|_* := \sum_i \sigma_i(A)$.

- The trace of a square matrix $A \in \mathbb{R}^{m\times m}$ is defined as $\mathrm{tr}(A) = \sum_{i=1}^{m} A_{ii}$.

- The spectral norm (also referred to as the operator norm) of a matrix A is defined as $\left\|A\right\|_2 := \max_i \sigma_i(A)$.

- Random variables are denoted using upper case letters such as X, Y.

- The expectation of a random variable X is denoted by $\mathbb{E}\left[X\right]$. In cases where the distribution of X is to be made explicit, the notation $\mathbb{E}_{X\sim\mathcal{D}}\left[X\right]$, or else simply $\mathbb{E}_{\mathcal{D}}\left[X\right]$, is used.

- $\mathsf{Unif}(\mathcal{X})$ denotes the uniform distribution over a compact set $\mathcal{X}$.

- The standard *big-Oh* notation is used to describe the asymptotic behavior of functions. The *soft-Oh* notation is employed to hide poly-logarithmic factors i.e., $f = \widetilde{\mathcal{O}}\left(g\right)$ will imply $f = \mathcal{O}\left(g\log^c(g)\right)$ for some absolute constant c.

Part I

Introduction and Basic Tools

1

Introduction

This section will set the stage for subsequent discussions by motivating some of the non-convex optimization problems we will be studying using real life examples, as well as setting up notation for the same.

1.1 Non-convex Optimization

The generic form of an analytic optimization problem is the following

$$\begin{aligned} &\min_{\mathbf{x} \in \mathbb{R}^p} \; f(\mathbf{x}) \\ &\text{s.t. } \mathbf{x} \in \mathcal{C}, \end{aligned}$$

where $\mathbf{x}$ is the *variable* of the problem, $f : \mathbb{R}^p \to \mathbb{R}$ is the *objective function* of the problem, and $\mathcal{C} \subseteq \mathbb{R}^p$ is the *constraint set* of the problem. When used in a machine learning setting, the objective function allows the algorithm designer to encode proper and expected behavior for the machine learning model, such as fitting well to training data with respect to some loss function, whereas the constraint allows restrictions on the model to be encoded, for instance, restrictions on model size.

An optimization problem is said to be *convex* if the objective is a convex function, as well as the constraint set is a convex set. We refer the reader to § 2 for formal definitions of these terms. An optimization problem that violates either one of these conditions, i.e., one that has a non-convex objective, or a non-convex constraint set, or both, is called a *non-convex* optimization problem. In this monograph, we will discuss non-convex optimization problems with non-convex objectives and convex constraints (§ 4, 5, 6, and 8), as well as problems with non-convex constraints but convex objectives (§ 3, 7, 9, 10, and 8). Such problems arise in a lot of application areas.

1.2 Motivation for Non-convex Optimization

Modern applications frequently require learning algorithms to operate in extremely high dimensional spaces. Examples include web-scale document classification problems where n-gram-based representations can have dimensionalities in the millions or more, recommendation systems with millions of items being recommended to millions of users, and signal processing tasks such as face recognition and image processing and bio-informatics tasks such as splice and gene detection, all of which present similarly high dimensional data.

Dealing with such high dimensionalities necessitates the imposition of structural constraints on the learning models being estimated from data. Such constraints are not only helpful in regularizing the learning problem, but often essential to prevent the problem from becoming ill-posed. For example, suppose we know how a user rates some items and wish to infer how this user would rate other items, possibly in order to inform future advertisement campaigns. To do so, it is essential to impose some structure on how a user's ratings for one set of items influences ratings for other kinds of items. Without such structure, it becomes impossible to infer any new user ratings. As we shall soon see, such structural constraints often turn out to be non-convex.

In other applications, the natural objective of the learning task is a non-convex function. Common examples include training deep neural networks and tensor decomposition problems. Although non-convex objectives and constraints allow us to accurately model learning problems, they often present a formidable challenge to algorithm designers.

This is because unlike convex optimization, we do not possess a handy set of tools for solving non-convex problems. Several non-convex optimization problems are known to be NP-hard to solve. The situation is made more bleak by a range of non-convex problems that are not only NP-hard to solve optimally, but NP-hard to solve approximately as well [Meka et al., 2008].

1.3 Examples of Non-Convex Optimization Problems

Below we present some areas where non-convex optimization problems arise naturally when devising learning problems.

Sparse Regression The classical problem of linear regression seeks to recover a linear model which can effectively predict a response variable as a linear function of covariates. For example, we may wish to predict the average expenditure of a household (the response) as a function of the education levels of the household members, their annual salaries and other relevant indicators (the covariates). The ability to do allows economic policy decisions to be more informed by revealing, for instance, how does education level affect expenditure.

More formally, we are provided a set of n covariate/response pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$. The linear regression approach makes the modeling assumption $y_i = \mathbf{x}_i^\top \mathbf{w}^* + \eta_i$ where $\mathbf{w}^* \in \mathbb{R}^p$ is the underlying linear model and η_i is some benign additive noise. Using the data provided $\{\mathbf{x}_i, y_i\}_{i=1,\ldots,n}$, we wish to recover back the model $\mathbf{w}^*$ as faithfully as possible.

A popular way to recover $\mathbf{w}^*$ is using the *least squares* formulation

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}\in\mathbb{R}^p}{\arg\min} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^\top \mathbf{w}\right)^2.$$

The linear regression problem as well as the least squares estimator, are extremely well studied and their behavior, precisely known. However, this age-old problem acquires new dimensions in situations where, either we expect only a few of the p features/covariates to be actually relevant to the problem but do not know their identity, or else are working in extremely data-starved settings i.e., $n \ll p$.

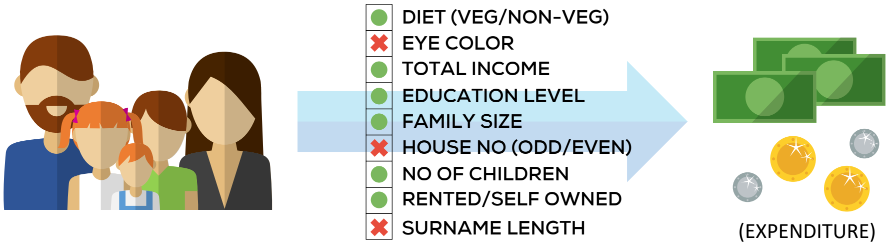


Figure 1.1: Not all available parameters and variables may be required for a prediction or learning task. Whereas the family size may significantly influence family expenditure, the eye color of family members does not directly or significantly influence it. Non-convex optimization techniques, such as sparse recovery, help discard irrelevant parameters and promote compact and accurate models.

The first problem often arises when there is an excess of covariates, several of which may be spurious or have no effect on the response. § 7 discusses several such practical examples. For now, consider the example depicted in Figure 1.1, that of expenditure prediction in a situation when the list of indicators include irrelevant ones such as whether the family lives in an odd-numbered house or not, which should arguably have no effect on expenditure. It is useful to eliminate such variables from consideration to promote consistency of the learned model.

The second problem is common in areas such as genomics and signal processing which face moderate to severe *data starvation* and the number of data points n available to estimate the model is small compared to the number of model parameters p to be estimated, i.e., $n \ll p$. Standard statistical approaches require at least $n \geq p$ data points to ensure a consistent estimation of all p model parameters and are unable to offer accurate model estimates in the face of data-starvation.

Both these problems can be handled by the *sparse recovery* approach, which seeks to fit a sparse model vector (i.e., a vector with say, no more than s non-zero entries) to the data. The least squares formulation, modified as a sparse recovery problem, is given below

$$\widehat{\mathbf{w}}_{\text{sp}} = \underset{\mathbf{w}\in\mathbb{R}^p}{\arg\min} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^\top \mathbf{w}\right)^2$$
$$\text{s.t. } \mathbf{w} \in \mathcal{B}_0(s),$$

Although the objective function in the above formulation is convex, the constraint $\|\mathbf{w}\|_0 \leq s$ (equivalently $\mathbf{w} \in \mathcal{B}_0(s)$ – see list of mathematical notation at the beginning of this monograph) corresponds to a non-convex constraint set[1]. Sparse recovery effortlessly solves the twin problems of discarding irrelevant covariates and countering data-starvation since typically, only $n \geq s \log p$ (as opposed to $n \geq p$) data points are required for sparse recovery to work which drastically reduces the data requirement. Unfortunately however, sparse-recovery is an NP-hard problem [Natarajan, 1995].

Recommendation Systems Several internet search engines and e-commerce websites utilize recommendation systems to offer items to users that they would benefit from, or like, the most. The problem of recommendation encompasses benign recommendations for songs etc, all the way to critical recommendations in personalized medicine.

To be able to make accurate recommendations, we need very good estimates of how each user likes each item (song), or would benefit from it (drug). We usually have first-hand information for some user-item pairs, for instance if a user has specifically rated a song or if we have administered a particular drug on a user and seen the outcome. However, users typically rate only a handful of the hundreds of thousands of songs in any commercial catalog and it is not feasible, or even advisable, to administer every drug to a user. Thus, for the vast majority of user-item pairs, we have no direct information.

It is useful to visualize this problem as a *matrix completion* problem: for a set of m users $u_1, \ldots, u_m$ and n items $a_1, \ldots, a_n$, we have an $m \times n$ *preference matrix* $A = [A_{ij}]$ where A_{ij} encodes the preference of the i^{th} user for the j^{th} item. We are able to directly view only a small number of entries of this matrix, for example, whenever a user explicitly rates an item. However, we wish to recover the remaining entries, i.e., complete this matrix. This problem is closely linked to the *collaborative filtering* technique popular in recommendation systems.

Now, it is easy to see that unless there exists some structure in matrix, and by extension, in the way users rate items, there would be

[1] See Exercise 2.6.

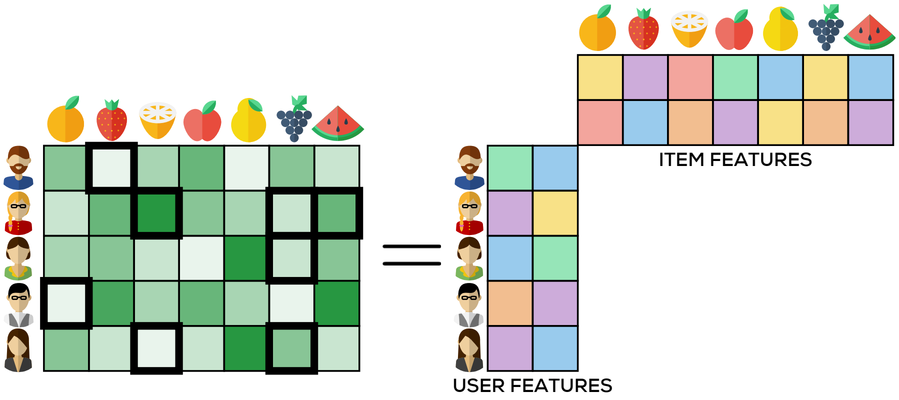


Figure 1.2: Only the entries of the ratings matrix with thick borders are observed. Notice that users rate infrequently and some items are not rated even once. Non-convex optimization techniques such as low-rank matrix completion can help recover the unobserved entries, as well as reveal hidden features that are descriptive of user and item properties, as shown on the right hand side.

no relation between the unobserved entries and the observed ones. This would result in there being no unique way to complete the matrix. Thus, it is essential to impose some structure on the matrix. A structural assumption popularly made is that of low rank: we wish to fill in the missing entries of A assuming that A is a low rank matrix. This can make the problem well-posed and have a unique solution since the additional low rank structure links the entries of the matrix together. The unobserved entries can no longer take values independently of the values observed by us. Figure 1.2 depicts this visually.

If we denote by $\Omega \subset [m] \times [n]$, the set of observed entries of A, then the low rank matrix completion problem can be written as

$$\widehat{A}_{\text{lr}} = \underset{X \in \mathbb{R}^{m \times n}}{\arg\min} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2$$

$$\text{s.t. } \operatorname{rank}(X) \leq r,$$

This formulation also has a convex objective but a non-convex rank constraint[2]. This problem can be shown to be NP-hard as well. Interestingly, we can arrive at an alternate formulation by imposing the

[2] See Exercise 2.7.

low-rank constraint indirectly. It turns out that[3] assuming the ratings matrix to have rank at most r is equivalent to assuming that the matrix A can be written as $A = UV^\top$ with the matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ having at most r columns. This leads us to the following alternate formulation

$$\widehat{A}_{\text{lv}} = \underset{\substack{U\in\mathbb{R}^{m\times r}\\ V\in\mathbb{R}^{n\times r}}}{\arg\min} \sum_{(i,j)\in\Omega} \left(U_i^\top V_j - A_{ij}\right)^2.$$

There are no constraints in the formulation. However, the formulation requires joint optimization over a pair of variables (U, V) instead of a single variable. More importantly, it can be shown[4] that the objective function is non-convex in (U, V).

It is curious to note that the matrices U and V can be seen as encoding r-dimensional descriptions of users and items respectively. More precisely, for every user $i \in [m]$, we can think of the vector $U^i \in \mathbb{R}^r$ (i.e., the i-th row of the matrix U) as describing user i, and for every item $j \in [n]$, use the row vector $V^j \in \mathbb{R}^r$ to describe the item j in vectoral form. The rating given by user i to item j can now be seen to be $A_{ij} \approx \langle U^i, V^j\rangle$. Thus, recovering the rank r matrix A also gives us a bunch of r-dimensional latent vectors describing the users and items. These latent vectors can be extremely valuable in themselves as they can help us in understanding user behavior and item popularity, as well as be used in "content"-based recommendation systems which can effectively utilize item and user features.

The above examples, and several others from machine learning, such as low-rank tensor decomposition, training deep networks, and training structured models, demonstrate the utility of non-convex optimization in naturally modeling learning tasks. However, most of these formulations are NP-hard to solve exactly, and sometimes even approximately. In the following discussion, we will briefly introduce a few approaches, classical as well as contemporary, that are used in solving such non-convex optimization problems.

[3] See Exercise 3.3.

[4] See Exercise 4.1.

1.4 The Convex Relaxation Approach

Faced with the challenge of non-convexity, and the associated NP-hardness, a traditional workaround in literature has been to modify the problem formulation itself so that existing tools can be readily applied. This is often done by *relaxing* the problem so that it becomes a convex optimization problem. Since this allows familiar algorithmic techniques to be applied, the so-called *convex relaxation* approach has been widely studied. For instance, there exist relaxed, convex problem formulations for both the recommendation system and the sparse regression problems. For sparse linear regression, the relaxation approach gives us the popular LASSO formulation.

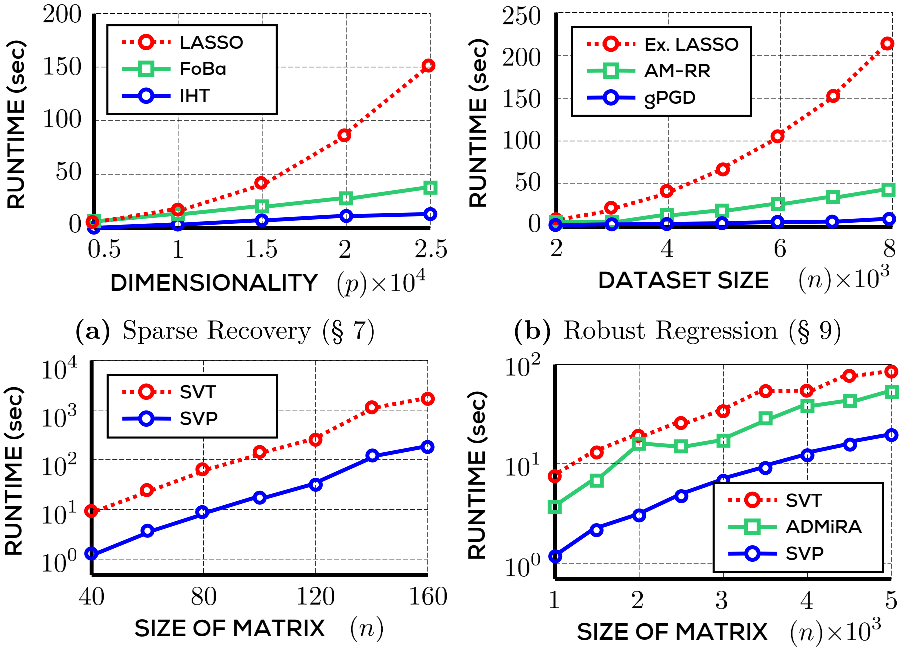
Now, in general, such modifications change the problem drastically, and the solutions of the relaxed formulation can be poor solutions to the original problem. However, it is known that if the problem possesses certain nice structure, then under careful relaxation, these distortions, formally referred to as a“relaxation gap”, are absent, i.e., solutions to the relaxed problem would be optimal for the original non-convex problem as well.

Although a popular and successful approach, this still has limitations, the most prominent of them being scalability. Although the relaxed convex optimization problems are solvable in polynomial time, it is often challenging to solve them *efficiently* for large-scale problems.

1.5 The Non-Convex Optimization Approach

Interestingly, in recent years, a new wisdom has permeated the fields of machine learning and signal processing, one that advises not to relax the non-convex problems and instead solve them directly. This approach has often been dubbed the *non-convex optimization* approach owing to its goal of optimizing non-convex formulations directly.

Techniques frequently used in non-convex optimization approaches include simple and efficient primitives such as projected gradient descent, alternating minimization, the expectation-maximization algorithm, stochastic optimization, and variants thereof. These are very fast in practice and remain favorites of practitioners.



(a) Sparse Recovery (§ 7)

(b) Robust Regression (§ 9)

(c) Matrix Recovery (§ 8)

(d) Matrix Completion (§ 8)

Figure 1.3: An empirical comparison of run-times offered by various approaches to four different non-convex optimization problems. LASSO, extended LASSO, SVT are relaxation-based methods whereas IHT, gPGD, FoBa, AM-RR, SVP, ADMiRA are non-convex methods. In all cases, non-convex optimization techniques offer routines that are faster, often by an order of magnitude or more, than relaxation-based methods. Note that Figures 1.3c and 1.3d, employ a y-axis at logarithmic scale. The details of the methods are present in the sections linked with the respective figures.

At first glance, however, these efforts seem doomed to fail, given to the aforementioned NP-hardness results. However, in a series of deep and illuminating results, it has been repeatedly revealed that if the problem possesses nice structure, then not only do relaxation approaches succeed, but non-convex optimization algorithms do too. In such nice cases, non-convex approaches are able to only avoid NP-hardness, but actually offer provably optimal solutions. In fact, in practice, they often handsomely outperform relaxation-based approaches in terms of speed and scalability. Figure 1.3 illustrates this for some applications that we will investigate more deeply in later sections.

Very interestingly, it turns out that problem structures that allow non-convex approaches to avoid NP-hardness results, are very similar to those that allow their convex relaxation counterparts to avoid distortions and a large relaxation gap! Thus, it seems that if the problems possess nice structure, convex relaxation-based approaches, as well as non-convex techniques, both succeed. However, non-convex techniques usually offer more scalable solutions.

1.6 Organization and Scope

Our goal of this monograph is to present basic tools, both algorithmic and analytic, that are commonly used in the design and analysis of non-convex optimization algorithms, as well as present results which best represent the non-convex optimization philosophy. The presentation should enthuse, as well as equip, the interested reader and allow further readings, independent investigations, and applications of these techniques in diverse areas.

Given this broad aim, we shall appropriately restrict the number of areas we cover in this monograph, as well as the depth in which we cover each area. For instance, the literature abounds in results that seek to perform optimizations with more and more complex structures being imposed - from sparse recovery to low rank matrix recovery to low rank tensor recovery. However, we shall restrict ourselves from venturing too far into these progressions. Similarly, within the problem of sparse recovery, there exist results for recovery in the simple least squares setting, the more involved setting of sparse M-estimation, as well as the still more involved setting of sparse M-estimation in the presence of outliers. Whereas we will cover sparse least squares estimation in depth, we will refrain from delving too deeply into the more involved sparse M-estimation problems.

That being said, the entire presentation will be self contained and accessible to anyone with a basic background in algebra and probability theory. Moreover, the bibliographic notes given at the end of the sections will give pointers that should enable the reader to explore the state of the art not covered in this monograph.

2

Mathematical Tools

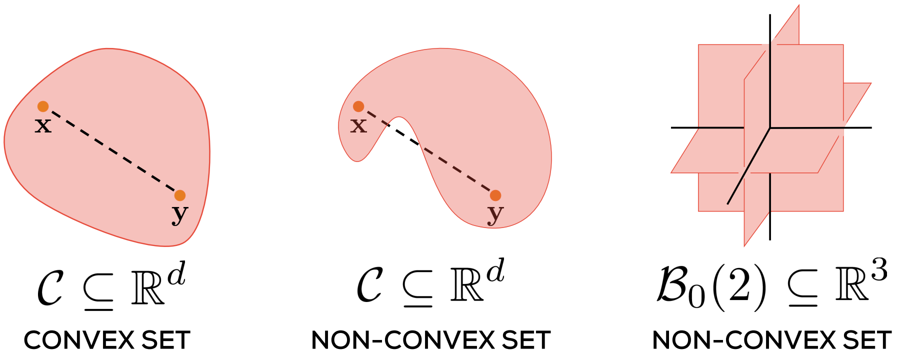
This section will introduce concepts, algorithmic tools, and analysis techniques used in the design and analysis of optimization algorithms. It will also explore simple convex optimization problems which will serve as a warm-up exercise.

2.1 Convex Analysis

We recall some basic definitions in convex analysis. Studying these will help us appreciate the structural properties of non-convex optimization problems later in the monograph. For the sake of simplicity, unless stated otherwise, we will assume that functions are continuously differentiable. We begin with the notion of a convex combination.

Definition 2.1 (Convex Combination)**.** A convex combination of a set of n vectors $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1 \ldots n$ in an arbitrary real space is a vector $\mathbf{x}_{\boldsymbol{\theta}} := \sum_{i=1}^n \theta_i \mathbf{x}_i$ where $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$, $\theta_i \geq 0$ and $\sum_{i=1}^n \theta_i = 1$.

A set that is closed under arbitrary convex combinations is a convex set. A standard definition is given below. Geometrically speaking, convex sets are those that contain all line segments that join two points inside the set. As a result, they cannot have any inward “bulges”.



$\mathcal{C} \subseteq \mathbb{R}^d$ CONVEX SET

$\mathcal{C} \subseteq \mathbb{R}^d$ NON-CONVEX SET

$\mathcal{B}_0(2) \subseteq \mathbb{R}^3$ NON-CONVEX SET

Figure 2.1: A convex set is closed under convex combinations. The presence of even a single uncontained convex combination makes a set non-convex. Thus, a convex set cannot have inward "bulges". In particular, the set of sparse vectors is non-convex.

Definition 2.2 (Convex Set)**.** A set $\mathcal{C} \in \mathbb{R}^p$ is considered convex if, for every $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ and $\lambda \in [0, 1]$, we have $(1 - \lambda) \cdot \mathbf{x} + \lambda \cdot \mathbf{y} \in \mathcal{C}$ as well.

Figure 2.1 gives visual representations of prototypical convex and non-convex sets. A related notion is that of convex functions which have a unique behavior under convex combinations. There are several definitions of convex functions, those that are more basic and general, as well as those that are restrictive but easier to use. One of the simplest definitions of convex functions, one that does not involve notions of derivatives, defines convex functions $f : \mathbb{R}^p \to \mathbb{R}$ as those for which, for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ and every $\lambda \in [0, 1]$, we have $f((1 - \lambda) \cdot \mathbf{x} + \lambda \cdot \mathbf{y}) \leq (1 - \lambda) \cdot f(\mathbf{x}) + \lambda \cdot f(\mathbf{y})$. For continuously differentiable functions, a more usable definition follows.

Definition 2.3 (Convex Function)**.** A continuously differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ is considered convex if for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$ we have $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$, where $\nabla f(\mathbf{x})$ is the gradient of f at $\mathbf{x}$.

A more general definition that extends to non-differentiable functions uses the notion of *subgradient* to replace the gradient in the above expression. A special class of convex functions is the class of *strongly convex* and *strongly smooth* functions. These are critical to the study of algorithms for non-convex optimization. Figure 2.2 provides a handy visual representation of these classes of functions.

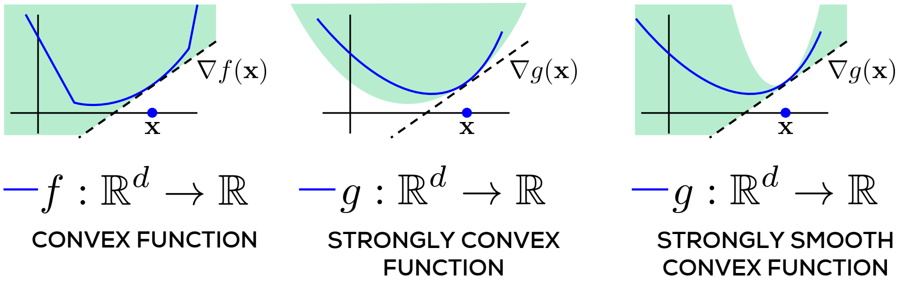


Figure 2.2: A convex function is lower bounded by its own tangent at all points. Strongly convex and smooth functions are, respectively, lower and upper bounded in the rate at which they may grow, by quadratic functions and cannot, again respectively, grow too slowly or too fast. In each figure, the shaded area describes regions the function curve is permitted to pass through.

Definition 2.4 (Strongly Convex/Smooth Function)**.** A continuously differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ is considered α-strongly convex (SC) and β-strongly smooth (SS) if for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, we have

$$\frac{\alpha}{2} \left\| \mathbf{x} - \mathbf{y} \right\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{\beta}{2} \left\| \mathbf{x} - \mathbf{y} \right\|_2^2 .$$

It is useful to note that strong convexity places a quadratic lower bound on the growth of the function at every point – the function must rise up at least as fast as a quadratic function. How fast it rises is characterized by the SC parameter α. Strong smoothness similarly places a quadratic upper bound and does not let the function grow too fast, with the SS parameter β dictating the upper limit.

We will soon see that these two properties are extremely useful in forcing optimization algorithms to rapidly converge to optima. Note that whereas strongly convex functions are definitely convex, strong smoothness does not imply convexity[1]. Strongly smooth functions may very well be non-convex. A property similar to strong smoothness is that of Lipschitzness.

Definition 2.5 (Lipschitz Function)**.** A function $f : \mathbb{R}^p \to \mathbb{R}$ is B-Lipschitz if for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, we have

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq B \cdot \left\| \mathbf{x} - \mathbf{y} \right\|_2 .$$

[1] See Exercise 2.1.

Notice that Lipschitzness places a upper bound on the growth of the function that is linear in the perturbation i.e., $\|\mathbf{x}-\mathbf{y}\|_2$, whereas strong smoothness (SS) places a quadratic upper bound. Also notice that Lipschitz functions need not be differentiable. However, differentiable functions with bounded gradients are always Lipschitz[2]. Finally, an important property that generalizes the behavior of convex functions on convex combinations is the Jensen's inequality.

Lemma 2.1 (Jensen's Inequality)**.** If X is a random variable taking values in the domain of a convex function f, then $\mathbb{E}\left[f(X)\right] \geq f(\mathbb{E}\left[X\right])$

This property will be useful while analyzing iterative algorithms.

2.2 Convex Projections

The projected gradient descent technique is a popular method for constrained optimization problems, both convex as well as non-convex. The *projection* step plays an important role in this technique. Given any closed set $\mathcal{C} \subset \mathbb{R}^p$, the projection operator $\Pi_{\mathcal{C}}(\cdot)$ is defined as

$$\Pi_{\mathcal{C}}(\mathbf{z}) := \underset{\mathbf{x}\in\mathcal{C}}{\arg\min} \;\; \|\mathbf{x}-\mathbf{z}\|_2 .$$

In general, one need not use only the L^2-norm in defining projections but is the most commonly used one. If $\mathcal{C}$ is a convex set, then the above problem reduces to a convex optimization problem. In several useful cases, one has access to a closed form solution for the projection.

For instance, if $\mathcal{C} = \mathcal{B}_2(1)$ i.e., the unit L_2 ball, then projection is equivalent[3] to a normalization step

$$\Pi_{\mathcal{B}_2(1)}(\mathbf{z}) = \begin{cases} \mathbf{z}/\left\|\mathbf{z}\right\|_2 & \text{if } \|\mathbf{z}\| > 1 \\ \mathbf{z} & \text{otherwise} \end{cases}.$$

For the case $\mathcal{C} = \mathcal{B}_1(1)$, the projection step reduces to the popular *soft thresholding* operation. If $\widehat{\mathbf{z}} := \Pi_{\mathcal{B}_1(1)}(\mathbf{z})$, then $\widehat{\mathbf{z}}_i = \max\left\{\mathbf{z}_i - \theta, 0\right\}$, where θ is a threshold that can be decided by a sorting operation on the vector [see Duchi et al., 2008, for details].

[2] See Exercise 2.2.

[3] See Exercise 2.3.

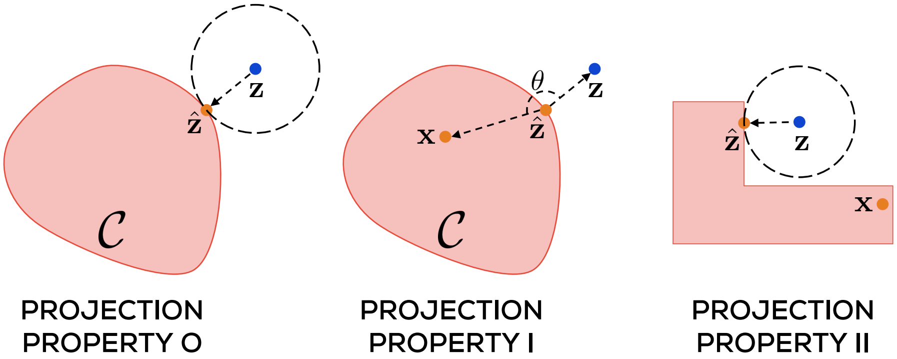


Figure 2.3: A depiction of projection operators and their properties. Projections reveal a closest point in the set being projected onto. For convex sets, projection property I ensures that the angle θ is always non-acute. Sets that satisfy projection property I also satisfy projection property II. Projection property II may be violated by non-convex sets. Projecting onto them may take the projected point $\mathbf{z}$ closer to certain points in the set (for example, $\widehat{\mathbf{z}}$) but farther from others (for example, $\mathbf{x}$).

Projections onto convex sets have some very useful properties which come in handy while analyzing optimization algorithms. In the following, we will study three properties of projections. These are depicted visually in Figure 2.3 to help the reader gain an intuitive appeal.

Lemma 2.2 (Projection Property-O)**.** For any set (convex or not) $\mathcal{C} \subset \mathbb{R}^p$ and $\mathbf{z} \in \mathbb{R}^p$, let $\widehat{\mathbf{z}} := \Pi_{\mathcal{C}}(\mathbf{z})$. Then for all $\mathbf{x} \in \mathcal{C}$, $\|\widehat{\mathbf{z}} - \mathbf{z}\|_2 \leq \|\mathbf{x} - \mathbf{z}\|_2$.

This property follows by simply observing that the projection step solves the the optimization problem $\min_{\mathbf{x}\in\mathcal{C}} \|\mathbf{x} - \mathbf{z}\|_2$. Note that this property holds for all sets, whether convex or not. However, the following two properties necessarily hold only for convex sets.

Lemma 2.3 (Projection Property-I)**.** For any convex set $\mathcal{C} \subset \mathbb{R}^p$ and any $\mathbf{z} \in \mathbb{R}^p$, let $\widehat{\mathbf{z}} := \Pi_{\mathcal{C}}(\mathbf{z})$. Then for all $\mathbf{x} \in \mathcal{C}$, $\langle \mathbf{x} - \widehat{\mathbf{z}}, \mathbf{z} - \widehat{\mathbf{z}} \rangle \leq 0$.

Proof. To prove this, assume the contra-positive. Suppose for some $\mathbf{x} \in \mathcal{C}$, we have $\langle \mathbf{x} - \widehat{\mathbf{z}}, \mathbf{z} - \widehat{\mathbf{z}} \rangle > 0$. Now, since $\mathcal{C}$ is convex and $\widehat{\mathbf{z}}, \mathbf{x} \in \mathcal{C}$, for any $\lambda \in [0, 1]$, we have $\mathbf{x}_\lambda := \lambda \cdot \mathbf{x} + (1 - \lambda) \cdot \widehat{\mathbf{z}} \in \mathcal{C}$. We will now show that for some value of $\lambda \in [0, 1]$, it must be the case that $\|\mathbf{z} - \mathbf{x}_\lambda\|_2 < \|\mathbf{z} - \widehat{\mathbf{z}}\|_2$. This will contradict the fact that $\widehat{\mathbf{z}}$ is the closest point in the convex set to $\mathbf{z}$ and prove the lemma. All that remains to be done is to find such a value of λ. The reader can verify that any value of $0 < \lambda < \min\left\{1, \frac{2\langle \mathbf{x}-\widehat{\mathbf{z}}, \mathbf{z}-\widehat{\mathbf{z}}\rangle}{\left\|\mathbf{x}-\widehat{\mathbf{z}}\right\|_2^2}\right\}$ suffices. Since we assumed $\langle \mathbf{x}-\widehat{\mathbf{z}}, \mathbf{z}-\widehat{\mathbf{z}}\rangle > 0$, any value of λ chosen this way is always in $(0, 1]$. □

Projection Property-I can be used to prove a very useful *contraction property* for convex projections. In some sense, a convex projection brings a point closer to *all* points in the convex set simultaneously.

Lemma 2.4 (Projection Property-II)**.** For any convex set $\mathcal{C} \subset \mathbb{R}^p$ and any $\mathbf{z} \in \mathbb{R}^p$, let $\widehat{\mathbf{z}} := \Pi_{\mathcal{C}}(\mathbf{z})$. Then for all $\mathbf{x} \in \mathcal{C}$, $\left\|\widehat{\mathbf{z}} - \mathbf{x}\right\|_2 \leq \left\|\mathbf{z} - \mathbf{x}\right\|_2$.

Proof. We have the following elementary inequalities

$$\begin{aligned}
\left\|\mathbf{z}-\mathbf{x}\right\|_2^2 &= \left\|(\widehat{\mathbf{z}}-\mathbf{x}) - (\widehat{\mathbf{z}}-\mathbf{z})\right\|_2^2 \\
&= \left\|\widehat{\mathbf{z}}-\mathbf{x}\right\|_2^2 + \left\|\widehat{\mathbf{z}}-\mathbf{z}\right\|_2^2 - 2\left\langle \widehat{\mathbf{z}}-\mathbf{x}, \widehat{\mathbf{z}}-\mathbf{z}\right\rangle \\
&\geq \left\|\widehat{\mathbf{z}}-\mathbf{x}\right\|_2^2 + \left\|\widehat{\mathbf{z}}-\mathbf{z}\right\|_2^2 \qquad \text{(Projection Property-I)} \\
&\geq \left\|\widehat{\mathbf{z}}-\mathbf{x}\right\|_2^2 \qquad \square
\end{aligned}$$

Note that Projection Properties-I and II are also called *first order* properties and can be violated if the underlying set is non-convex. However, Projection Property-O, often called a *zeroth order* property, always holds, whether the underlying set is convex or not.

2.3 Projected Gradient Descent

We now move on to study the projected gradient descent algorithm. This is an extremely simple and efficient technique that can effortlessly scale to large problems. Although we will apply this technique to non-convex optimization tasks later, we first look at its behavior on convex optimization problems as a warm up exercise. We warn the reader that the proof techniques used in the convex case do not apply directly to non-convex problems. Consider the following optimization problem:

$$\begin{aligned}
&\min_{\mathbf{x}\in\mathbb{R}^p} \; f(\mathbf{x}) \\
&\;\;\text{s.t. } \mathbf{x} \in \mathcal{C}.
\end{aligned} \qquad \text{(CVX-OPT)}$$

Algorithm 1 Projected Gradient Descent (PGD)

```
**Input:** Convex objective $f$, convex constraint set $\mathcal{C}$, step lengths $\eta_t$
**Output:** A point $\widehat{\mathbf{x}} \in \mathcal{C}$ with near-optimal objective value
$\mathbf{x}^1 \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots, T$ do
    $\mathbf{z}^{t+1} \leftarrow \mathbf{x}^t - \eta_t \cdot \nabla f(\mathbf{x}^t)$
    $\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{C}}(\mathbf{z}^{t+1})$
end for
(OPTION 1) return $\widehat{\mathbf{x}}_{\text{final}} = \mathbf{x}^T$
(OPTION 2) return $\widehat{\mathbf{x}}_{\text{avg}} = (\sum_{t=1}^{T} \mathbf{x}^t)/T$
(OPTION 3) return $\widehat{\mathbf{x}}_{\text{best}} = \arg\min_{t\in[T]} f(\mathbf{x}^t)$
```

In the above optimization problem, $\mathcal{C} \subset \mathbb{R}^p$ is a convex constraint set and $f : \mathbb{R}^p \to \mathbb{R}$ is a convex objective function. We will assume that we have oracle access to the gradient and projection operators, i.e., for any point $\mathbf{x} \in \mathbb{R}^p$ we are able to access $\nabla f(\mathbf{x})$ and $\Pi_{\mathcal{C}}(\mathbf{x})$.

The projected gradient descent algorithm is stated in Algorithm 1. The procedure generates iterates $\mathbf{x}^t$ by taking steps guided by the gradient in an effort to reduce the function value locally. Finally it returns either the final iterate, the average iterate, or the best iterate.

2.4 Convergence Guarantees for PGD

We will analyze PGD for objective functions that are either a) convex with bounded gradients, or b) strongly convex and strongly smooth. Let $f^* = \min_{\mathbf{x}\in\mathcal{C}}\ f(\mathbf{x})$ be the optimal value of the optimization problem. A point $\widehat{\mathbf{x}} \in \mathcal{C}$ will be said to be an ϵ-optimal solution if $f(\widehat{\mathbf{x}}) \leq f^* + \epsilon$.

2.4.1 Convergence with Bounded Gradient Convex Functions

Consider a convex objective function f with bounded gradients over a convex constraint set $\mathcal{C}$ i.e., $\left\| f(\mathbf{x}) \right\|_2 \leq G$ for all $\mathbf{x} \in \mathcal{C}$.

Theorem 2.5. Let f be a convex objective with bounded gradients and Algorithm 1 be executed for T time steps with step lengths $\eta_t = \eta = \frac{1}{\sqrt{T}}$. Then, for any $\epsilon > 0$, if $T = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$, then $\frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^t) \leq f^* + \epsilon$.

We see that the PGD algorithm in this setting ensures that the function value of the iterates approaches f^* *on an average*. We can use this result to prove the convergence of the PGD algorithm. If we use OPTION 3, i.e., $\widehat{\mathbf{x}}_{\text{best}}$, then since by construction, we have $f(\widehat{\mathbf{x}}_{\text{best}}) \leq f(\mathbf{x}^t)$ for all t, by applying Theorem 2.5, we get

$$f(\widehat{\mathbf{x}}_{\text{best}}) \leq \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^t) \leq f^* + \epsilon,$$

If we use OPTION 2, i.e., $\widehat{\mathbf{x}}_{\text{avg}}$, which is cheaper since we do not have to perform function evaluations to find the best iterate, we can apply Jensen's inequality (Lemma 2.1) to get the following

$$f(\widehat{\mathbf{x}}_{\text{avg}}) = f\left(\frac{1}{T}\sum_{t=1}^{T} \mathbf{x}^t\right) \leq \frac{1}{T}\sum_{t=1}^{T} f(\mathbf{x}^t) \leq f^* + \epsilon.$$

Note that the Jensen's inequality may be applied only when the function f is convex. Now, whereas OPTION 1 i.e., $\widehat{\mathbf{x}}_{\text{final}}$, is the cheapest and does not require any additional operations, $\widehat{\mathbf{x}}_{\text{final}}$ does not converge to the optimum for convex functions in general and may oscillate close to the optimum. However, we shall shortly see that $\widehat{\mathbf{x}}_{\text{final}}$ does converge if the objective function is strongly smooth. Recall that strongly smooth functions may not grow at a faster-than-quadratic rate.

The reader would note that we have set the step length to a value that depends on the total number of iterations T for which the PGD algorithm is executed. This is called a *horizon-aware* setting of the step length. In case we are not sure what the value of T would be, a *horizon-oblivious* setting of $\eta_t = \frac{1}{\sqrt{t}}$ can also be shown to work[4].

Proof (of Theorem 2.5). Let $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{C}}\ f(\mathbf{x})$ denote any point in the constraint set where the optimum function value is achieved. Such a point always exists if the constraint set is closed and the objective function continuous. We will use the following *potential function* $\Phi_t = f(\mathbf{x}^t) - f(\mathbf{x}^*)$ to track the progress of the algorithm. Note that Φ_t measures the sub-optimality of the t-th iterate. Indeed, the statement of the theorem is equivalent to claiming that $\frac{1}{T}\sum_{t=1}^{T}\Phi_t \leq \epsilon$.

[4] See Exercise 2.4.

(Apply Convexity) We apply convexity to upper bound the potential function at every step. Convexity is a global property and very useful in getting an upper bound on the level of sub-optimality of the current iterate in such analyses.

$$\Phi_t = f(\mathbf{x}^t) - f(\mathbf{x}^*) \leq \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \right\rangle$$

We now do some elementary manipulations

$$\begin{aligned}
&\left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \right\rangle = \frac{1}{\eta} \left\langle \eta \cdot \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \right\rangle \\
&= \frac{1}{2\eta} \left(\left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2 + \left\| \eta \cdot \nabla f(\mathbf{x}^t) \right\|_2^2 - \left\| \mathbf{x}^t - \eta \cdot \nabla f(\mathbf{x}^t) - \mathbf{x}^* \right\|_2^2 \right) \\
&= \frac{1}{2\eta} \left(\left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2 + \left\| \eta \cdot \nabla f(\mathbf{x}^t) \right\|_2^2 - \left\| \mathbf{z}^{t+1} - \mathbf{x}^* \right\|_2^2 \right) \\
&\leq \frac{1}{2\eta} \left(\left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2 + \eta^2 G^2 - \left\| \mathbf{z}^{t+1} - \mathbf{x}^* \right\|_2^2 \right),
\end{aligned}$$

where the first step applies the identity $2ab = a^2 + b^2 - (a+b)^2$, the second step uses the update step of the PGD algorithm that sets $\mathbf{z}^{t+1} \leftarrow \mathbf{x}^t - \eta_t \cdot \nabla f(\mathbf{x}^t)$, and the third step uses the fact that the objective function f has bounded gradients.

(Apply Projection Property) We apply Lemma 2.4 to get

$$\left\| \mathbf{z}^{t+1} - \mathbf{x}^* \right\|_2^2 \geq \left\| \mathbf{x}^{t+1} - \mathbf{x}^* \right\|_2^2$$

Putting all these together gives us

$$\Phi_t \leq \frac{1}{2\eta} \left(\left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2 - \left\| \mathbf{x}^{t+1} - \mathbf{x}^* \right\|_2^2 \right) + \frac{\eta G^2}{2}$$

The above expression is interesting since it tells us that, apart from the $\eta G^2/2$ term which is small as $\eta = \frac{1}{\sqrt{T}}$, the current sub-optimality Φ_t is small if the consecutive iterates $\mathbf{x}^t$ and $\mathbf{x}^{t+1}$ are close to each other (and hence similar in distance from $\mathbf{x}^*$).

This observation is quite useful since it tells us that once PGD stops making a lot of progress, it actually converges to the optimum! In hindsight, this is to be expected. Since we are using a constant step length, only a vanishing gradient can cause PGD to stop progressing.

However, for convex functions, this only happens at global optima. Summing the expression up across time steps, performing telescopic cancellations, using $\mathbf{x}^1 = \mathbf{0}$, and dividing throughout by T gives us

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\Phi_t &\leq \frac{1}{2\eta T}\left(\|\mathbf{x}^*\|_2^2 - \|\mathbf{x}^{T+1} - \mathbf{x}^*\|_2^2\right) + \frac{\eta G^2}{2}\\ &\leq \frac{1}{2\sqrt{T}}\left(\|\mathbf{x}^*\|_2^2 + G^2\right),\end{aligned}$$

where in the second step, we have used the fact that $\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2 \geq 0$ and $\eta = 1/\sqrt{T}$. This gives us the claimed result. □

2.4.2 Convergence with Strongly Convex and Smooth Functions

We will now prove a stronger guarantee for PGD when the objective function is strongly convex and strongly smooth (see Definition 2.4).

Theorem 2.6. Let f be an objective that satisfies the α-SC and β-SS properties. Let Algorithm 1 be executed with step lengths $\eta_t = \eta = \frac{1}{\beta}$. Then after at most $T = \mathcal{O}\left(\frac{\beta}{\alpha}\log\frac{\beta}{\epsilon}\right)$ steps, we have $f(\mathbf{x}^T) \leq f(\mathbf{x}^*) + \epsilon$.

This result is particularly nice since it ensures that the final iterate $\widehat{\mathbf{x}}_{\text{final}} = \mathbf{x}^T$ converges, allowing us to use OPTION 1 in Algorithm 1 when the objective is SC/SS. A further advantage is the accelerated rate of convergence. Whereas for general convex functions, PGD requires $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ iterations to reach an ϵ-optimal solution, for SC/SS functions, it requires only $\mathcal{O}\left(\log\frac{1}{\epsilon}\right)$ iterations.

The reader would notice the insistence on the step length being set to $\eta = \frac{1}{\beta}$. In fact the proof we show below crucially uses this setting. In practice, for many problems, β may not be known to us or may be expensive to compute which presents a problem. However, as it turns out, it is not necessary to set the step length exactly to $1/\beta$. The result can be shown to hold even for values of $\eta < 1/\beta$ which are nevertheless large enough, but the proof becomes more involved. In practice, the step length is tuned globally by doing a grid search over several η values, or per-iteration using line search mechanisms, to obtain a step length value that assures good convergence rates.

Proof (of Theorem 2.6). This proof is a nice opportunity for the reader to see how the SC/SS properties are utilized in a convergence analysis. As with convexity in the proof of Theorem 2.5, the strong convexity property is a global property that will be useful in assessing the progress made so far by relating the optimal point $\mathbf{x}^*$ with the current iterate $\mathbf{x}^t$. Strong smoothness on the other hand, will be used locally to show that the procedure makes significant progress between iterates.

We will prove the result by showing that after at most $T = \mathcal{O}\left(\frac{\beta}{\alpha}\log\frac{1}{\epsilon}\right)$ steps, we will have $\left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2^2 \leq \frac{2\epsilon}{\beta}$. This already tells us that we have reached very close to the optimum. However, we can use this to show that $\mathbf{x}^T$ is ϵ-optimal in function value as well. Since we are very close to the optimum, it makes sense to apply strong smoothness to upper bound the sub-optimality as follows

$$f(\mathbf{x}^T) \leq f(\mathbf{x}^*) + \left\langle \nabla f(\mathbf{x}^*), \mathbf{x}^T - \mathbf{x}^* \right\rangle + \frac{\beta}{2}\left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2^2.$$

Now, since $\mathbf{x}^*$ is an optimal point for the constrained optimization problem with a convex constraint set $\mathcal{C}$, the first order optimality condition [see Bubeck, 2015, Proposition 1.3] gives us $\langle \nabla f(\mathbf{x}^*), \mathbf{x} - \mathbf{x}^* \rangle \leq 0$ for any $\mathbf{x} \in \mathcal{C}$. Applying this condition with $\mathbf{x} = \mathbf{x}^T$ gives us

$$f(\mathbf{x}^T) - f(\mathbf{x}^*) \leq \frac{\beta}{2}\left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2^2 \leq \epsilon,$$

which proves that $\mathbf{x}^T$ is an ϵ-optimal point. We now show $\left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2^2 \leq \frac{2\epsilon}{\beta}$. Given that we wish to show convergence in terms of the iterates, and not in terms of the function values, as we did in Theorem 2.5, a natural potential function for this analysis is $\Phi_t = \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2$.

(Apply Strong Smoothness) As discussed before, we use it to show that PGD always makes significant progress in each iteration.

$$\begin{aligned}
f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) &\leq \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle + \frac{\beta}{2}\left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|_2^2 \\
&= \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^{t+1} - \mathbf{x}^* \right\rangle + \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle + \frac{\beta}{2}\left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|_2^2 \\
&= \frac{1}{\eta}\left\langle \mathbf{x}^t - \mathbf{z}^{t+1}, \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle + \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle + \frac{\beta}{2}\left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|_2^2
\end{aligned}$$

(Apply Projection Rule) The above expression contains an unwieldy term $\mathbf{z}^{t+1}$. Since this term only appears during projection steps, we eliminate it by applying Projection Property-I (Lemma 2.3) to get

$$\begin{aligned}\left\langle \mathbf{x}^t - \mathbf{z}^{t+1}, \mathbf{x}^{t+1} - \mathbf{x}^* \right\rangle &\leq \left\langle \mathbf{x}^t - \mathbf{x}^{t+1}, \mathbf{x}^{t+1} - \mathbf{x}^* \right\rangle \\ &= \frac{\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 - \left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|_2^2 - \left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2}{2}\end{aligned}$$

Using $\eta = 1/\beta$ and combining the above results gives us

$$f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) \leq \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle + \frac{\beta}{2}\left(\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 - \left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2\right)$$

(Apply Strong Convexity) The above expression is perfect for a telescoping step but for the inner product term. Fortunately, this can be eliminated using strong convexity.

$$\left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle \leq f(\mathbf{x}^*) - f(\mathbf{x}^t) - \frac{\alpha}{2}\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2$$

Combining with the above this gives us

$$f(\mathbf{x}^{t+1}) - f(\mathbf{x}^*) \leq \frac{\beta - \alpha}{2}\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 - \frac{\beta}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2.$$

The above form seems almost ready for a telescoping exercise. However, something much stronger can be said here, especially due to the $\frac{-\alpha}{2}\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2$ term. Notice that we have $f(\mathbf{x}^{t+1}) \geq f(\mathbf{x}^*)$. This means

$$\frac{\beta}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2 \leq \frac{\beta - \alpha}{2}\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2,$$

which can be written as

$$\Phi_{t+1} \leq \left(1 - \frac{\alpha}{\beta}\right)\Phi_t \leq \exp\left(-\frac{\alpha}{\beta}\right)\Phi_t,$$

where we have used the fact that $1 - x \leq \exp(-x)$ for all $x \in \mathbb{R}$. What we have arrived at is a very powerful result as it assures us that the potential value goes down by a constant fraction at every iteration! Applying this result recursively gives us

$$\Phi_{t+1} \leq \exp\left(-\frac{\alpha t}{\beta}\right)\Phi_1 = \exp\left(-\frac{\alpha t}{\beta}\right)\left\|\mathbf{x}^*\right\|_2^2,$$

since $\mathbf{x}^1 = \mathbf{0}$. Thus, we deduce that $\Phi_T = \left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2^2 \leq \frac{2\epsilon}{\beta}$ after at most $T = \mathcal{O}\left(\frac{\beta}{\alpha}\log\frac{\beta}{\epsilon}\right)$ steps which finishes the proof □

We notice that the convergence of the PGD algorithm is of the form $\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2 \leq \exp\left(-\frac{\alpha t}{\beta}\right) \left\|\mathbf{x}^*\right\|_2^2$. The number $\kappa := \frac{\beta}{\alpha}$ is the *condition number* of the optimization problem. The concept of condition number is central to numerical optimization. Below we give an informal and generic definition for the concept. In later sections we will see the condition number appearing repeatedly in the context of the convergence of various optimization algorithms for convex, as well as non-convex problems. The exact numerical form of the condition number (for instance here it is β/α) will also change depending on the application at hand. However, in general, all these definitions of condition number will satisfy the following property.

Definition 2.6 (Condition Number - Informal). The condition number of a function $f : \mathcal{X} \rightarrow \mathbb{R}$ is a scalar $\kappa \in \mathbb{R}$ that bounds how much the function value can change relative to a perturbation of the input.

Functions with a small condition number are stable and changes to their input do not affect the function output values too much. However, functions with a large condition number can be quite jumpy and experience abrupt changes in output values even if the input is changed slightly. To gain a deeper appreciation of this concept, consider a differentiable function f that is also α-SC and β-SS. Consider a stationary point for f i.e., a point $\mathbf{x}$ such that $\nabla f(\mathbf{x}) = \mathbf{0}$. For a general function, such a point can be a local optima or a saddle point. However, since f is strongly convex, $\mathbf{x}$ is the (unique) global minima[5] of f. Then we have, for any other point $\mathbf{y}$

$$\frac{\alpha}{2} \left\|\mathbf{x} - \mathbf{y}\right\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) \leq \frac{\beta}{2} \left\|\mathbf{x} - \mathbf{y}\right\|_2^2$$

Dividing throughout by $\frac{\alpha}{2} \left\|\mathbf{x} - \mathbf{y}\right\|_2^2$ gives us

$$\frac{f(\mathbf{y}) - f(\mathbf{x})}{\frac{\alpha}{2} \left\|\mathbf{x} - \mathbf{y}\right\|_2^2} \in \left[1, \frac{\beta}{\alpha}\right] := [1, \kappa]$$

Thus, upon perturbing the input from the global minimum $\mathbf{x}$ to a point $\left\|\mathbf{x} - \mathbf{y}\right\|_2 =: \epsilon$ distance away, the function value does change much – it

[5] See Exercise 2.5.

goes up by an amount at least $\frac{\alpha\epsilon^2}{2}$ but at most $\kappa\cdot\frac{\alpha\epsilon^2}{2}$. Such well behaved response to perturbations is very easy for optimization algorithms to exploit to give fast convergence.

The condition number of the objective function can significantly affect the convergence rate of algorithms. Indeed, if $\kappa = \frac{\beta}{\alpha}$ is small, then $\exp\left(-\frac{\alpha}{\beta}\right) = \exp\left(-\frac{1}{\kappa}\right)$ would be small, ensuring fast convergence. However, if $\kappa \gg 1$ then $\exp\left(-\frac{1}{\kappa}\right) \approx 1$ and the procedure might offer slow convergence.

2.5 Exercises

Exercise 2.1. Show that strong smoothness does not imply convexity by constructing a non-convex function $f : \mathbb{R}^p \to \mathbb{R}$ that is 1-SS.

Exercise 2.2. Show that if a differentiable function f has bounded gradients i.e., $\|\nabla f(\mathbf{x})\|_2 \leq G$ for all $\mathbf{x} \in \mathbb{R}^d$, then f is Lipschitz. What is its Lipschitz constant?
Hint: use the mean value theorem.

Exercise 2.3. Show that for any point $\mathbf{z} \notin \mathcal{B}_2(r)$, the projection onto the ball is given by $\Pi_{\mathcal{B}_2(r)}(\mathbf{z}) = \frac{r}{\|\mathbf{z}\|_2} \cdot \mathbf{z}$.

Exercise 2.4. Show that a *horizon-oblivious* setting of $\eta_t = \frac{1}{\sqrt{t}}$ while executing the PGD algorithm with a convex function with bounded gradients also ensures convergence.
Hint: the convergence rates may be a bit different for this setting.

Exercise 2.5. Show that if $f : \mathbb{R}^p \to \mathbb{R}$ is a strongly convex function that is differentiable, then there is a unique point $\mathbf{x}^* \in \mathbb{R}^p$ that minimizes the function value f i.e., $f(\mathbf{x}^*) = \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x})$.

Exercise 2.6. Show that the set of sparse vectors $\mathcal{B}_0(s) \subset \mathbb{R}^p$ is non-convex for any $s < p$. What happens when $s = p$?

Exercise 2.7. Show that $\mathcal{B}_{\text{rank}}(r) \subseteq \mathbb{R}^{n\times n}$, the set of $n \times n$ matrices with rank at most r, is non-convex for any $r < n$. What happens when $r = n$?

Exercise 2.8. Consider the Cartesian product set $\mathcal{C} = \mathbb{R}^{m \times r} \times \mathbb{R}^{n \times r}$. Show that it is convex.

Exercise 2.9. Consider a least squares optimization problem with a strongly convex and smooth objective. Show that the condition number of this problem is equal to the condition number of the Hessian matrix of the objective function.

Exercise 2.10. Show that if $f : \mathbb{R}^p \to \mathbb{R}$ is a strongly convex function that is differentiable, then optimization problems with f as an objective and a convex constraint set $\mathcal{C}$ always have a unique solution i.e., there is a unique point $\mathbf{x}^* \in \mathcal{C}$ that is a solution to the optimization problem $\arg\min_{\mathbf{x} \in \mathcal{C}} \ f(x)$. This generalizes the result in Exercise 2.5.
Hint: use the first order optimality condition (see proof of Theorem 2.6)

2.6 Bibliographic Notes

The sole aim of this discussion was to give a self-contained introduction to concepts and tools in convex analysis and descent algorithms in order to seamlessly introduce non-convex optimization techniques and their applications in subsequent sections. However, we clearly realize our inability to cover several useful and interesting results concerning convex functions and optimization techniques given the paucity of scope to present this discussion. We refer the reader to literature in the field of optimization theory for a much more relaxed and deeper introduction to the area of convex optimization. Some excellent examples include [Bertsekas, 2016, Boyd and Vandenberghe, 2004, Bubeck, 2015, Nesterov, 2003, Sra et al., 2011].

Part II

Non-convex Optimization Primitives

3

Non-Convex Projected Gradient Descent

In this section we will introduce and study gradient descent-style methods for non-convex optimization problems. In § 2, we studied the projected gradient descent method for convex optimization problems. Unfortunately, the algorithmic and analytic techniques used in convex problems fail to extend to non-convex problems. In fact, non-convex problems are NP-hard to solve and thus, no algorithmic technique should be expected to succeed on these problems in general.

However, the situation is not so bleak. As we discussed in § 1, several breakthroughs in non-convex optimization have shown that non-convex problems that possess nice additional structure can be solved not just in polynomial time, but rather efficiently too. Here, we will study the inner workings of projected gradient methods on such structured non-convex optimization problems.

The discussion will be divided into three parts. The first part will take a look at constraint sets that, despite being non-convex, possess additional structure so that projections onto them can be carried out efficiently. The second part will take a look at structural properties of objective functions that can aid optimization. The third part will present and analyze a simple extension of the PGD algorithm for non-convex problems. We will see that for problems that do possess nicely structured objective functions and constraint sets, the PGD-style algorithm does converge to the global optimum in polynomial time with a linear rate of convergence.

We would like to point out to the reader that our emphasis in this section will be on generality and exposition of basic concepts. We will seek to present easily accessible analyses for problems that have non-convex objectives. However, the price we will pay for this generality is in the fineness of the results we present. The results discussed in this section are not the best possible and more refined and problem-specific results will be discussed in subsequent sections where specific applications will be discussed in detail.

3.1 Non-Convex Projections

Executing the projected gradient descent algorithm with non-convex problems requires projections onto non-convex sets. Now, a quick look at the projection problem

$$\Pi_{\mathcal{C}}(\mathbf{z}) := \underset{\mathbf{x}\in\mathcal{C}}{\arg\min}\ \left\|\mathbf{x}-\mathbf{z}\right\|_2$$

reveals that this is an optimization problem in itself. Thus, when the set $\mathcal{C}$ to be projected onto is non-convex, the projection problem can itself be NP-hard. However, for several well-structured sets, projection can be carried out efficiently despite the sets being non-convex.

3.1.1 Projecting into Sparse Vectors

In the sparse linear regression example discussed in § 1,

$$\widehat{\mathbf{w}} = \underset{\|\mathbf{w}\|_0\leq s}{\arg\min}\ \sum_{i=1}^{n}\left(y_i - \mathbf{x}_i^\top \mathbf{w}\right)^2,$$

applying projected gradient descent requires projections onto the set of s-sparse vectors i.e., $\mathcal{B}_0(s) := \{\mathbf{x}\in\mathbb{R}^p, \left\|\mathbf{x}\right\|_0 \leq s\}$. The following result shows that the projection $\Pi_{\mathcal{B}_0(s)}(\mathbf{z})$ can be carried out by simply sorting the coordinates of the vector $\mathbf{z}$ according to magnitude and setting all except the top-s coordinates to zero.

Lemma 3.1. For any vector $\mathbf{z} \in \mathbb{R}^p$, let σ be the permutation that sorts the coordinates of $\mathbf{z}$ in decreasing order of magnitude, i.e., $\left|\mathbf{z}_{\sigma(1)}\right| \geq \left|\mathbf{z}_{\sigma(2)}\right| \geq \ldots \geq \left|\mathbf{z}_{\sigma(p)}\right|$. Then the vector $\widehat{\mathbf{z}} := \Pi_{\mathcal{B}_0(s)}(\mathbf{z})$ is obtained by setting $\widehat{\mathbf{z}}_i = \mathbf{z}_i$ if $\sigma(i) \leq s$ and $\widehat{\mathbf{z}}_i = 0$ otherwise.

Proof. We first notice that since the function $x \mapsto x^2$ is an increasing function on the positive half of the real line, we have $\underset{\mathbf{x}\in\mathcal{C}}{\arg\min}\ \|\mathbf{x}-\mathbf{z}\|_2 = \underset{\mathbf{x}\in\mathcal{C}}{\arg\min}\ \|\mathbf{x}-\mathbf{z}\|_2^2$. Next, we observe that the vector $\widehat{\mathbf{z}} := \Pi_{\mathcal{B}_0(s)}$ must satisfy $\widehat{\mathbf{z}}_i = \mathbf{z}_i$ for all $i \in \operatorname{supp}(\widehat{\mathbf{z}})$ otherwise we can decrease the objective value $\|\widehat{\mathbf{z}}-\mathbf{z}\|_2^2$ by ensuring this. Having established this gives us $\|\widehat{\mathbf{z}}-\mathbf{z}\|_2^2 = \sum_{i\notin\operatorname{supp}(\widehat{\mathbf{z}})} \mathbf{z}_i^2$. This is clearly minimized when $\operatorname{supp}(\widehat{\mathbf{z}})$ has the coordinates of $\mathbf{z}$ with largest magnitude. □

3.1.2 Projecting into Low-rank Matrices

In the recommendation systems problem, as discussed in § 1

$$\widehat{A}_{\mathrm{lr}} = \underset{\operatorname{rank}(X)\leq r}{\arg\min} \sum_{(i,j)\in\Omega} \left(X_{ij} - A_{ij}\right)^2,$$

we need to project onto the set of low-rank matrices. Let us first define this problem formally. Consider matrices of a certain order, say $m \times n$ and let $\mathcal{C} \subset \mathbb{R}^{m\times n}$ be an arbitrary set of matrices. Then, the projection operator $\Pi_{\mathcal{C}}(\cdot)$ is defined as follows: for any matrix $A \in R^{m\times n}$,

$$\Pi_{\mathcal{C}}(A) := \underset{X\in\mathcal{C}}{\arg\min}\ \|A - X\|_F,$$

where $\|\cdot\|_F$ is the Frobenius norm over matrices. For low rank projections we require $\mathcal{C}$ to be the set of low rank matrices $\mathcal{B}_{\mathrm{rank}}(r) := \{A \in \mathbb{R}^{m\times n}, \operatorname{rank}(A) \leq r\}$. Yet again, this projection can be done efficiently by performing a *Singular Value Decomposition* on the matrix A and retaining the top r singular values and vectors. The Eckart-Young-Mirsky theorem proves that this indeed gives us the projection.

Theorem 3.2 (Eckart-Young-Mirsky theorem)**.** For any matrix $A \in \mathbb{R}^{m\times n}$, let $U\Sigma V^\top$ be the singular value decomposition of A such that $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)})$ where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(m,n)}$. Then for any $r \leq \min(m,n)$, the matrix $\widehat{A}_{(r)} := \Pi_{\mathcal{B}_{\mathrm{rank}}(r)}(A)$ can be obtained as $U_{(r)}\Sigma_{(r)}V_{(r)}^{\top}$ where $U_{(r)} := [U_1 U_2 \ldots U_r]$, $V(r) := [V_1 V_2 \ldots V_r]$, and $\Sigma_{(r)} := \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$.

Although we have stated the above result for projections with the Frobenius norm defining the projections, the Eckart-Young-Mirsky theorem actually applies to any unitarily invariant norm including the Schatten norms and the operator norm. The proof of this result is beyond the scope of this monograph.

Before moving on, we caution the reader that the ability to efficiently project onto the non-convex sets mentioned above does not imply that non-convex projections are as nicely behaved as their convex counterparts. Indeed, none of the projections mentioned above satisfy projection properties I or II (Lemmata 2.3 and 2.4). This will pose a significant challenge while analyzing PGD-style algorithms for non-convex problems since, as we would recall, these properties were crucially used in all convergence proofs discussed in § 2.

3.2 Restricted Strong Convexity and Smoothness

In § 2, we saw how optimization problems with convex constraint sets and objective functions that are convex and have bounded gradients, or else are strongly convex and smooth, can be effectively optimized using PGD, with much faster rates of convergence if the objective is strongly convex and smooth. However, when the constraint set fails to be convex, these results fail to apply.

There are several workarounds to this problem, the simplest being to convert the constraint set into a convex one, possibly by taking its *convex hull*[1], which is what relaxation methods do. However, a much less drastic alternative exists that is widely popular in non-convex optimization literature.

The intuition is a simple one and generalizes much of the insights we gained from our discussion in § 2. The first thing we need to notice[2]

[1] The convex hull of any set $\mathcal{C}$ is the "smallest" convex set $\overline{\mathcal{C}}$ that contains $\mathcal{C}$. Formally, we define $\overline{\mathcal{C}} = \bigcap_{\substack{S \supseteq \mathcal{C} \\ S \text{ is convex}}} S$. If $\mathcal{C}$ is convex then it is its own convex hull.

[2] See Exercise 3.1.

is that the convergence results for the PGD algorithm in § 2 actually do not require the objective function to be convex (or strongly convex/strongly smooth) over the entire $\mathbb{R}^p$. These properties are only required to be satisfied over the constraint set being considered. A natural generalization that emerges from this insight is the concept of *restricted* properties that are discussed below.

Definition 3.1 (Restricted Convexity). A continuously differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ is said to satisfy restricted convexity over a (possibly non-convex) region $\mathcal{C} \subseteq \mathbb{R}^p$ if for every $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ we have $f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle$, where $\nabla f(\mathbf{x})$ is the gradient of f at $\mathbf{x}$.

As before, a more general definition that extends to non-differentiable functions, uses the notion of subgradient to replace the gradient in the above expression.

Definition 3.2 (Restricted Strong Convexity/Smoothness). A continuously differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ is said to satisfy α-restricted strong convexity (RSC) and β-restricted strong smoothness (RSS) over a (possibly non-convex) region $\mathcal{C} \subseteq \mathbb{R}^p$ if for every $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, we have

$$\frac{\alpha}{2} \left\| \mathbf{x} - \mathbf{y} \right\|_2^2 \leq f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq \frac{\beta}{2} \left\| \mathbf{x} - \mathbf{y} \right\|_2^2 .$$

Note that, as Figure 3.1 demonstrates, even non-convex functions can demonstrate the RSC/RSS properties over suitable subsets. Conversely, functions that satisfy RSC/RSS need not be convex. It turns out that in several practical situations, such as those explored by later sections, the objective functions in the non-convex optimization problems do satisfy the RSC/RSS properties described above, in some form.

We also remind the reader that the RSC/RSS definitions presented here are quite generic and presented to better illustrate basic concepts. Indeed, for specific non-convex problems such as sparse recovery, low-rank matrix recovery, and robust regression, the later sections will develop more refined versions of these properties that are better tailored to those problems. In particular, for sparse recovery problems, the RSC/RSS properties can be shown to be related[3] to the well-known restricted isometry property (RIP).

[3] See Exercise 7.4.

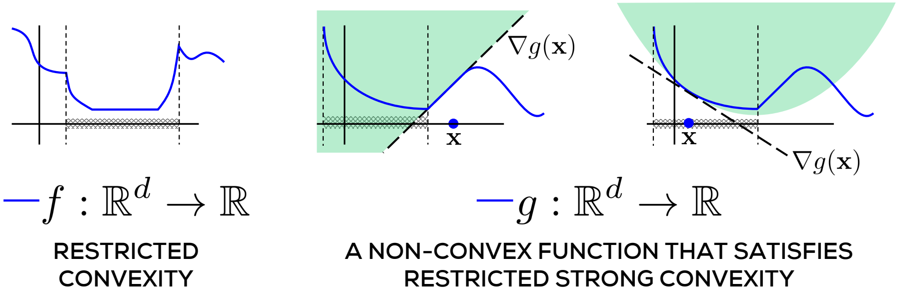

$\nabla g(\mathbf{x})$
$\mathbf{x}$
$\mathbf{x}$
$\nabla g(\mathbf{x})$
$f : \mathbb{R}^d \to \mathbb{R}$
RESTRICTED CONVEXITY
$g : \mathbb{R}^d \to \mathbb{R}$
A NON-CONVEX FUNCTION THAT SATISFIES RESTRICTED STRONG CONVEXITY

Figure 3.1: A depiction of restricted convexity properties. f is clearly non-convex over the entire real line but is convex within the cross-hatched region bounded by the dotted vertical lines. g is a non-convex function that satisfies restricted strong convexity. Outside the cross-hatched region (again bounded by the dotted vertical lines), g fails to even be convex as its curve falls below its tangent, but within the region, it actually exhibits strong convexity.

3.3 Generalized Projected Gradient Descent

We now present the generalized projected gradient descent algorithm (gPGD) for non-convex optimization problems. The procedure is outlined in Algorithm 2. The reader would find it remarkably similar to the PGD procedure in Algorithm 1. However, a crucial difference is in the projections made. Whereas PGD utilized convex projections, the gPGD procedure, if invoked with a non-convex constraint set $\mathcal{C}$, utilizes non-convex projections instead.

We will perform the convergence analysis for the gPGD algorithm assuming that the projection step in the algorithm is carried out exactly. As we saw in the preceding discussion, this can be accomplished efficiently for non-convex sets arising in several interesting problem settings. However, despite this, the convergence analysis will remain challenging due to the non-convexity of the problem.

Firstly, we will not be able to assume that the objective function we are working with is convex over the entire $\mathbb{R}^p$. Secondly, non-convex projections do not satisfy projection properties I or II. Finally, the first order optimality condition ([Bubeck, 2015, Proposition 1.3]) we used to prove Theorem 2.6 also fails to hold for non-convex constraint sets. Since the analyses for the PGD algorithm crucially used these

Algorithm 2 Generalized Projected Gradient Descent (gPGD)

```
Input: Objective function $f$, constraint set $\mathcal{C}$, step length $\eta$
Output: A point $\widehat{\mathbf{x}} \in \mathcal{C}$ with near-optimal objective value
$\mathbf{x}^1 \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots, T$ do
    $\mathbf{z}^{t+1} \leftarrow \mathbf{x}^t - \eta \cdot \nabla f(\mathbf{x}^t)$
    $\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{C}}(\mathbf{z}^{t+1})$
end for
return $\widehat{\mathbf{x}}_{\text{final}} = \mathbf{x}^T$
```

results, we will have to find workarounds to all of them. We will denote the optimal function value as $f^* = \min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x})$ and any optimizer as $\mathbf{x}^* \in \mathcal{C}$ such that $f(\mathbf{x}^*) = f^*$.

To simplify the presentation we will assume that $\nabla f(\mathbf{x}^*) = \mathbf{0}$. This assumption is satisfied whenever the objective function is differentiable and the optimal point $\mathbf{x}^*$ lies in the interior of the constraint set $\mathcal{C}$. However, many sets such as $\mathcal{B}_0(s)$ do not possess an interior (although they may still possess a *relative* interior) and this assumption fails by default on such sets. Nevertheless, this assumption will greatly simplify the presentation as well as help us focus on the key issues. Moreover, convergence results can be shown without making this assumption too.

Theorem 3.3 gives the convergence proof for gPGD. The reader will notice that the convergence rate offered by gPGD is similar to the one offered by the PGD algorithm for convex optimization (see Theorem 2.6). However, the gPGD algorithm requires a more careful analysis of the structure of the objective function and constraint set since it is working with a non-convex optimization problem.

Theorem 3.3. Let f be a (possibly non-convex) function satisfying the α-RSC and β-RSS properties over a (possibly non-convex) constraint set $\mathcal{C}$ with $\beta/\alpha < 2$. Let Algorithm 2 be executed with a step length $\eta = \frac{1}{\beta}$. Then after at most $T = \mathcal{O}\left(\frac{\alpha}{2\alpha-\beta} \log \frac{1}{\epsilon}\right)$ steps, $f(\mathbf{x}^T) \leq f(\mathbf{x}^*) + \epsilon$.

This result holds even when the step length is set to values that are large enough but yet smaller than $1/\beta$. However, setting $\eta = \frac{1}{\beta}$ simplifies the proof and allows us to focus on the key concepts.

Proof (of Theorem 3.3). Recall that the proof of Theorem 2.5 used the SC/SS properties for the analysis. We will replace these by the RSC/RSS properties – we will use RSC to track the global convergence of the algorithm and RSS to locally assess the progress made by the algorithm in each iteration. We will use $\Phi_t = f(\mathbf{x}^{t+1}) - f(\mathbf{x}^*)$ as the potential function.

(Apply Restricted Strong Smoothness) Since both $\mathbf{x}^t, \mathbf{x}^{t+1} \in \mathcal{C}$ due to the projection steps, we apply the β-RSS property to them.

$$\begin{aligned} f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) &\leq \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle + \frac{\beta}{2} \left\| \mathbf{x}^t - \mathbf{x}^{t+1} \right\|_2^2 \\ &= \frac{1}{\eta} \left\langle \mathbf{x}^t - \mathbf{z}^{t+1}, \mathbf{x}^{t+1} - \mathbf{x}^t \right\rangle + \frac{\beta}{2} \left\| \mathbf{x}^t - \mathbf{x}^{t+1} \right\|_2^2 \\ &= \frac{\beta}{2} \left(\left\| \mathbf{x}^{t+1} - \mathbf{z}^{t+1} \right\|_2^2 - \left\| \mathbf{x}^t - \mathbf{z}^{t+1} \right\|_2^2 \right) \end{aligned}$$

Notice that this step crucially uses the fact that $\eta = 1/\beta$.

(Apply Projection Property) We are again stuck with the unwieldy $\mathbf{z}^{t+1}$ term. However, unlike before, we cannot apply projection properties I or II as non-convex projections do not satisfy them. Instead, we resort to Projection Property-O (Lemma 2.2), that all projections (even non-convex ones) must satisfy. Applying this property gives us

$$\begin{aligned} f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) &\leq \frac{\beta}{2} \left(\left\| \mathbf{x}^* - \mathbf{z}^{t+1} \right\|_2^2 - \left\| \mathbf{x}^t - \mathbf{z}^{t+1} \right\|_2^2 \right) \\ &= \frac{\beta}{2} \left(\left\| \mathbf{x}^* - \mathbf{x}^t \right\|_2^2 + 2 \left\langle \mathbf{x}^* - \mathbf{x}^t, \mathbf{x}^t - \mathbf{z}^{t+1} \right\rangle \right) \\ &= \frac{\beta}{2} \left\| \mathbf{x}^* - \mathbf{x}^t \right\|_2^2 + \left\langle \mathbf{x}^* - \mathbf{x}^t, \nabla f(\mathbf{x}^t) \right\rangle \end{aligned}$$

(Apply Restricted Strong Convexity) Since both $\mathbf{x}^t, \mathbf{x}^* \in \mathcal{C}$, we apply the α-RSC property to them. However, we do so in two ways:

$$\begin{aligned} f(\mathbf{x}^*) - f(\mathbf{x}^t) &\geq \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle + \frac{\alpha}{2} \left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2 \\ f(\mathbf{x}^t) - f(\mathbf{x}^*) &\geq \left\langle \nabla f(\mathbf{x}^*), \mathbf{x}^t - \mathbf{x}^* \right\rangle + \frac{\alpha}{2} \left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2 \geq \frac{\alpha}{2} \left\| \mathbf{x}^t - \mathbf{x}^* \right\|_2^2, \end{aligned}$$

where in the second line we used the fact that we assumed $\nabla f(\mathbf{x}^*) = \mathbf{0}$. We recall that this assumption can be done away with but makes the proof more complicated which we wish to avoid. Simple manipulations with the two equations give us

$$\left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle + \frac{\beta}{2}\left\|\mathbf{x}^* - \mathbf{x}^t\right\|_2^2 \leq \left(2 - \frac{\beta}{\alpha}\right)\left(f(\mathbf{x}^*) - f(\mathbf{x}^t)\right)$$

Putting this in the earlier expression gives us

$$f(\mathbf{x}^{t+1}) - f(\mathbf{x}^t) \leq \left(2 - \frac{\beta}{\alpha}\right)\left(f(\mathbf{x}^*) - f(\mathbf{x}^t)\right)$$

The above inequality is quite interesting. It tells us that the larger the gap between $f(\mathbf{x}^*)$ and $f(\mathbf{x}^t)$, the larger will be the drop in objective value in going from $\mathbf{x}^t$ to $\mathbf{x}^{t+1}$. The form of the result is also quite fortunate as it assures us that we will cover a constant fraction $\left(2 - \frac{\beta}{\alpha}\right)$ of the remaining "distance" to $\mathbf{x}^*$ at each step! Rearranging this gives

$$\Phi_{t+1} \leq (\kappa - 1)\Phi_t,$$

where $\kappa = \beta/\alpha$. Note that we always have $\kappa \geq 1$[4] and by assumption $\kappa = \beta/\alpha < 2$, so that we always have $\kappa - 1 \in [0, 1)$. This proves the result after simple manipulations. □

We see that the condition number has yet again played in crucial role in deciding the convergence rate of the algorithm, this time for a non-convex problem. However, we see that the condition number is defined differently here, using the RSC/RSS constants instead of the SC/SS constants as we did in § 2.

The reader would notice that while there was no restriction on the condition number κ in the analysis of the PGD algorithm (see Theorem 2.6), the analysis of the gPGD algorithm does require $\kappa < 2$. It turns out that this restriction can be done away with for specific problems. However, the analysis becomes significantly more complicated. Resolving this issue in general is beyond the scope of this monograph but we will revisit this question in § 7 when we study sparse recovery in ill-conditioned settings. i.e., with large condition numbers.

[4] See Exercise 3.2.

In subsequent sections, we will see more refined versions of the gPGD algorithm for different non-convex optimization problems, as well as more refined and problem-specific analyses. In all cases we will see that the RSC/RSS assumptions made by us can be fulfilled in practice and that gPGD-style algorithms offer very good performance on practical machine learning and signal processing problems.

3.4 Exercises

Exercise 3.1. Verify that the basic convergence result for the PGD algorithm in Theorem 2.5, continues to hold when the constraint set $\mathcal{C}$ is convex and f only satisfies *restricted convexity* over $\mathcal{C}$ (i.e., f is not convex over the entire $\mathbb{R}^p$). Verify that the result for strongly convex and smooth functions in Theorem 2.6, also continues to hold if f satisfies RSC and RSS over a convex constraint set $\mathcal{C}$.

Exercise 3.2. Let the function f satisfy the α-RSC and β-RSS properties over a set $\mathcal{C}$. Show that the condition number $\kappa = \frac{\beta}{\alpha} \geq 1$. Note that the function f and the set $\mathcal{C}$ may both be non-convex.

Exercise 3.3. Recall the recommendation systems problem we discussed in § 1. Show that assuming the ratings matrix to be rank-r is equivalent to assuming that with every user $i \in [m]$ there is associated a vector $\mathbf{u}_i \in \mathbb{R}^r$ describing that user, and with every item $j \in [n]$ there is associated a vector $\mathbf{v}_i \in \mathbb{R}^r$ describing that item such that the rating given by user i to item j is $A_{ij} = \mathbf{u}_i^\top \mathbf{v}_j$.
Hint: Use the singular value decomposition for A.

4

Alternating Minimization

In this section we will introduce a widely used non-convex optimization primitive, namely the alternating minimization principle. The technique is extremely general and its popular use actually predates the recent advances in non-convex optimization by several decades. Indeed, the popular Lloyd's algorithm [Lloyd, 1982] for k-means clustering and the EM algorithm [Dempster et al., 1977] for latent variable models are problem-specific variants of the general alternating minimization principle. The technique continues to inspire new algorithms for several important non-convex optimization problems such as matrix completion, robust learning, phase retrieval and dictionary learning.

Given the popularity and breadth of use of this method, our task to present an introductory treatment will be even more challenging here. To keep the discussion focused on core principles and tools, we will refrain from presenting the alternating minimization principle in all its varieties. Instead, we will focus on showing, in a largely problem-independent manner, what are the challenges that face alternating minimization when applied to real-life problems, and how they can be overcome. Subsequent sections will then show how this principle can be applied to various machine learning and signal processing tasks. In particular, § 5 will be devoted to the EM algorithm which embodies the alternating minimization principle and is extremely popular for latent variable estimation problems in statistics and machine learning.

The discussion will be divided into four parts. In the first part, we will look at some useful structural properties of functions that frequently arise in alternating minimization settings. In the second part, we will present a general implementation of the alternating minimization principle and discuss some challenges faced by this algorithm in offering convergent behavior in real-life problems. In the third part, as a warm-up exercise, we will show how these challenges can be overcome when the optimization problem being solved is convex. Finally in the fourth part, we will discuss the more interesting problem of convergence of alternating minimization for non-convex problems.

4.1 Marginal Convexity and Other Properties

Alternating Minimization is most often utilized in settings where the optimization problem concerns two or more (groups of) variables. For example, recall the matrix completion problem in recommendation systems from § 1 which involved two variables U and V denoting respectively, the latent factors for the users and the items. In several such cases, the optimization problem, more specifically the objective function, is not *jointly convex* in all the variables.

Definition 4.1 (Joint Convexity). A continuously differentiable function in two variables $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ is considered jointly convex if for every $(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2) \in \mathbb{R}^p \times \mathbb{R}^q$ we have

$$f(\mathbf{x}^2, \mathbf{y}^2) \geq f(\mathbf{x}^1, \mathbf{y}^1) + \left\langle \nabla f(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2) - (\mathbf{x}^1, \mathbf{y}^1) \right\rangle,$$

where $\nabla f(\mathbf{x}^1, \mathbf{y}^1)$ is the gradient of f at the point $(\mathbf{x}^1, \mathbf{y}^1)$.

The definition of joint convexity is not different from the one for convexity Definition 2.3. Indeed the two coincide if we assume f to be a function of a single variable $\mathbf{z} = (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{p+q}$ instead of two variables. However, not all multivariate functions that arise in applications are jointly convex. This motivates the notion of *marginal convexity.*

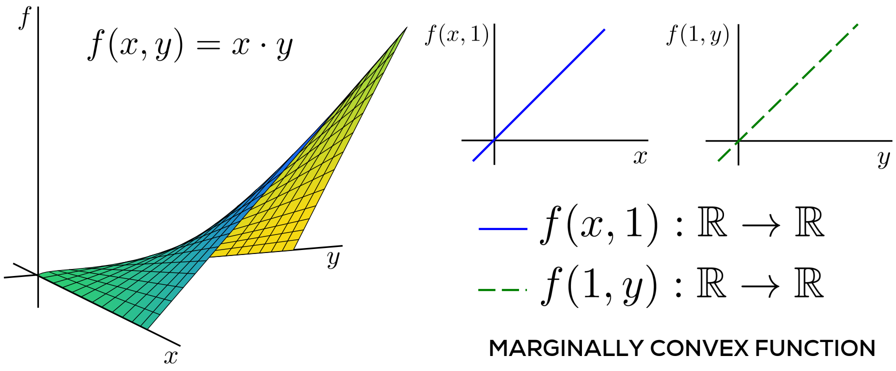


Figure 4.1: A marginally convex function is not necessarily (jointly) convex. The function $f(x, y) = x \cdot y$ is marginally linear, hence marginally convex, in both its variables, but clearly not a (jointly) convex function.

Definition 4.2 (Marginal Convexity). A continuously differentiable function of two variables $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ is considered marginally convex in its first variable if for every value of $\mathbf{y} \in \mathbb{R}^q$, the function $f(\cdot, \mathbf{y}) : \mathbb{R}^p \to \mathbb{R}$ is convex, i.e., for every $\mathbf{x}^1, \mathbf{x}^2 \in \mathbb{R}^p$, we have

$$f(\mathbf{x}^2, \mathbf{y}) \geq f(\mathbf{x}^1, \mathbf{y}) + \left\langle \nabla_{\mathbf{x}} f(\mathbf{x}^1, \mathbf{y}), \mathbf{x}^2 - \mathbf{x}^1 \right\rangle,$$

where $\nabla_{\mathbf{x}} f(\mathbf{x}^1, \mathbf{y})$ is the partial gradient of f with respect to its first variable at the point $(\mathbf{x}^1, \mathbf{y})$. A similar condition is imposed for f to be considered marginally convex in its second variable.

Although the definition above has been given for a function of two variables, it clearly extends to functions with an arbitrary number of variables. It is interesting to note that whereas the objective function in the matrix completion problem mentioned earlier is not jointly convex in its variables, it is indeed marginally convex in both its variables[1].

It is also useful to note that even though a function that is marginally convex in all its variables need not be a jointly convex function (see Figure 4.1), the converse is true[2]. We will find the following notions of *marginal strong convexity and smoothness* to be especially useful in our subsequent discussions.

[1] See Exercise 4.1.

[2] See Exercise 4.2.

Definition 4.3 (Marginally Strongly Convex/Smooth Function)**.** A continuously differentiable function $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ is considered (uniformly) α-marginally strongly convex (MSC) and (uniformly) β-marginally strongly smooth (MSS) in its first variable if for every value of $\mathbf{y} \in \mathbb{R}^q$, the function $f(\cdot, \mathbf{y}) : \mathbb{R}^p \to \mathbb{R}$ is α-strongly convex and β-strongly smooth, i.e., for every $\mathbf{x}^1, \mathbf{x}^2 \in \mathbb{R}^p$, we have

$$\frac{\alpha}{2}\left\|\mathbf{x}^2 - \mathbf{x}^1\right\|_2^2 \leq f(\mathbf{x}^2, \mathbf{y}) - f(\mathbf{x}^1, \mathbf{y}) - \left\langle \mathbf{g}, \mathbf{x}^2 - \mathbf{x}^1 \right\rangle \leq \frac{\beta}{2}\left\|\mathbf{x}^2 - \mathbf{x}^1\right\|_2^2,$$

where $\mathbf{g} = \nabla_{\mathbf{x}} f(\mathbf{x}^1, \mathbf{y})$ is the partial gradient of f with respect to its first variable at the point $(\mathbf{x}^1, \mathbf{y})$. A similar condition is imposed for f to be considered (uniformly) MSC/MSS in its second variable.

The above notion is a "uniform" one as the α, β parameters do not depend on the $\mathbf{y}$ coordinate. It is instructive to relate MSC/MSS to the RSC/RSS properties from § 2. MSC/MSS extend the idea of functions that are not "globally" convex (strongly or otherwise) but do exhibit such properties under "qualifications". MSC/MSS use a different qualification than RSC/RSS did. Note that a function that is MSC with respect to *all* its variables, need not be a convex function[3].

4.2 Generalized Alternating Minimization

The alternating minimization algorithm (gAM) is outlined in Algorithm 3 for an optimization problem on two variables constrained to the sets $\mathcal{X}$ and $\mathcal{Y}$ respectively. The procedure can be easily extended to functions with more variables, or have more complicated constraint sets[4] of the form $\mathcal{Z} \subset \mathcal{X} \times \mathcal{Y}$. After an initialization step, gAM alternately fixes one of the variables and optimizes over the other.

This approach of solving several intermediate *marginal* optimization problems instead of a single big problem is the key to the practical success of gAM. Alternating minimization is mostly used when these marginal problems are easy to solve. Later, we will see that there exist simple, often closed form solutions to these marginal problems for applications such as matrix completion, robust learning etc.

[3] See Exercise 4.3.

[4] See Exercise 4.4.

Algorithm 3 Generalized Alternating Minimization (gAM)

```
Input: Objective function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$
Output: A point $(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) \in \mathcal{X} \times \mathcal{Y}$ with near-optimal objective value
(x^1, y^1) ← INITALIZE()   $(\mathbf{x}^1, \mathbf{y}^1) \leftarrow \textsf{INITALIZE}()$
for $t = 1, 2, \ldots, T$ do
    $\mathbf{x}^{t+1} \leftarrow \arg\min_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}, \mathbf{y}^t)$
    $\mathbf{y}^{t+1} \leftarrow \arg\min_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}^{t+1}, \mathbf{y})$
end for
return $(\mathbf{x}^T, \mathbf{y}^T)$
```

There also exist "descent" versions of gAM which do not completely perform marginal optimizations but take gradient steps along the variables instead. The alternating descent updates look like the following

$$\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t - \eta_{t,1} \cdot \nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t)$$
$$\mathbf{y}^{t+1} \leftarrow \mathbf{y}^t - \eta_{t,2} \cdot \nabla_{\mathbf{y}} f(\mathbf{x}^{t+1}, \mathbf{y}^t)$$

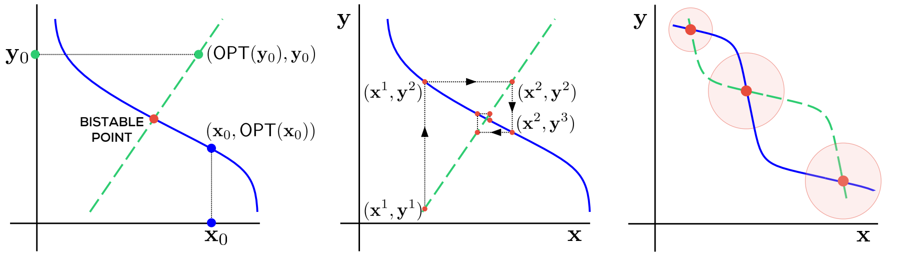
These descent versions are often easier to execute but may also converge more slowly. If the problem is nicely structured, then progress made on the intermediate problems offers fast convergence to the optimum. However, from the point of view of convergence, gAM faces several challenges. To discuss those, we first introduce some more concepts.

Definition 4.4 (Marginally Optimum Coordinate)**.** Let f be a function of two variables constrained to be in the sets $\mathcal{X}, \mathcal{Y}$ respectively. For any point $\mathbf{y} \in \mathcal{Y}$, we say that $\widetilde{\mathbf{x}}$ is a marginally optimal coordinate with respect to $\mathbf{y}$, and use the shorthand $\widetilde{\mathbf{x}} \in \mathsf{mOPT}_f(\mathbf{y})$, if $f(\widetilde{\mathbf{x}}, \mathbf{y}) \leq f(\mathbf{x}, \mathbf{y})$ for all $\mathbf{x} \in \mathcal{X}$. Similarly for any $\mathbf{x} \in \mathcal{X}$, we say $\widetilde{\mathbf{y}} \in \mathsf{mOPT}_f(\mathbf{x})$ if $\widetilde{\mathbf{y}}$ is a marginally optimal coordinate with respect to $\mathbf{x}$.

Definition 4.5 (Bistable Point)**.** Given a function f over two variables constrained within the sets $\mathcal{X}, \mathcal{Y}$ respectively, a point $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ is considered a *bistable* point if $\mathbf{y} \in \mathsf{mOPT}_f(\mathbf{x})$ and $\mathbf{x} \in \mathsf{mOPT}_f(\mathbf{y})$ i.e., both coordinates are marginally optimal with respect to each other.

It is easy to see[5] that the optimum of the optimization problem must be a bistable point. The reader can also verify that the gAM

[5] See Exercise 4.5.


$\mathbf{y}_0$
$(\mathsf{OPT}(\mathbf{y}_0), \mathbf{y}_0)$
BISTABLE POINT
$(\mathbf{x}_0, \mathsf{OPT}(\mathbf{x}_0))$
$\mathbf{x}_0$
$\mathbf{y}$
$(\mathbf{x}^1, \mathbf{y}^2)$
$(\mathbf{x}^2, \mathbf{y}^2)$
$(\mathbf{x}^2, \mathbf{y}^3)$
$(\mathbf{x}^1, \mathbf{y}^1)$
$\mathbf{x}$
$\mathbf{y}$
$\mathbf{x}$

MARGINAL OPTIMALITY PLOT

gAM ITERATES CONVERGE TO THE BISTABLE POINT

MULTIPLE BISTABLE POINTS WITH RESPECTIVE REGIONS OF ATTRACTION

Figure 4.2: The first plot depicts the marginally optimal coordinate curves for the two variables whose intersection produces bistable points. gAM is adept at converging to bistable points for well-behaved functions. Note that gAM progresses only along a single variable at a time. Thus, in the 2-D example in the second plot, the progress lines are only vertical or horizontal. A function may have multiple bistable points, each with its own region of attraction, depicted as shaded circles.

procedure must stop after it has reached a bistable point. However, two questions arise out of this. First, how fast does gAM approach a bistable point and second, even if it reaches a bistable point, is that point guaranteed to be (globally) optimal?

The first question will be explored in detail later. It is interesting to note that the gAM procedure has no parameters, such as step length. This can be interpreted as a benefit as well as a drawback. While it relieves the end-user from spending time tweaking parameters, it also means that the user has less control over the progress of the algorithm. Consequently, the convergence of the gAM procedure is totally dependent on structural properties of the optimization problem. In practice, it is common to switch between gAM updates as given in Algorithm 3 and descent versions thereof discussed earlier. The descent versions do give a step length as a tunable parameter to the user.

The second question requires a closer look at the interaction between the objective function and the gAM process. Figure 4.2 illustrates this with toy bi-variate functions over $\mathbb{R}^2$. In the first figure, the bold solid curve plots the function $g : \mathcal{X} \to \mathcal{X} \times \mathcal{Y}$ with $g(\mathbf{x}) = (\mathbf{x}, \mathsf{mOPT}_f(\mathbf{x}))$ (in this toy case, the marginally optimal coordinates are taken to be unique for simplicity, i.e., $|\mathsf{mOPT}_f(\mathbf{x})| = 1$ for all $\mathbf{x} \in \mathcal{X}$). The bold dashed curve similarly plots $h : \mathcal{Y} \to \mathcal{X} \times \mathcal{Y}$ with $h(\mathbf{y}) = (\mathsf{mOPT}_f(\mathbf{y}), \mathbf{y})$. These plots are quite handy in demonstrating the convergence properties of the gAM algorithm.

It is easy to see that bistable points lie precisely at the intersection of the bold solid and the bold dashed curves. The second illustration shows how the gAM process may behave when instantiated with this toy function – clearly gAM exhibits rapid convergence to the bistable point. However, the third illustration shows that functions may have multiple bistable points. The figure shows that this may happen even if the marginally optimal coordinates are unique i.e., for every $\mathbf{x}$ there is a unique $\widetilde{\mathbf{y}}$ such that $\widetilde{\mathbf{y}} = \mathsf{mOPT}_f(\mathbf{x})$ and vice versa.

In case a function taking bounded values possesses multiple bistable points, the bistable point to which gAM eventually converges depends on where the procedure was initialized. This is exemplified in the third illustration where each bistable region has its own “region of attraction”. If initialized inside a particular region, gAM converges to the bistable point corresponding to that region. This means that in order to converge to the globally optimal point, gAM must be initialized inside the region of attraction of the global optimum.

The above discussion shows that it may be crucial to properly initialize the gAM procedure to ensure convergence to the global optimum. Indeed, when discussing gAM-style algorithms for learning latent variable models, matrix completion and phase retrieval in later sections, we will pay special attention to initialize the procedure “close” to the optimum. The only exception will be that of robust regression in § 9 where it seems that the problem structure ensures a unique bistable point and so, a careful initialization is not required.

4.3 A Convergence Guarantee for gAM for Convex Problems

As usual, things do become nice when the optimization problems are convex. For instance, for differentiable convex functions, all bistable points are global minima[6] and thus, converging to any one of them is sufficient. In fact, approaches similar to gAM, commonly known

[6] See Exercise 4.6.

as *Coordinate Minimization* (CM), are extremely popular for large scale convex optimization problems. The CM procedure simply treats a single p-dimensional variable $\mathbf{x} \in \mathbb{R}^p$ as p one-dimensional variables $\{\mathbf{x}_1, \ldots, \mathbf{x}_p\}$ and executes gAM-like steps with them, resulting in the intermediate problems being uni-dimensional. However, it is worth noting that gAM can struggle with non-differentiable objectives.

In the following, we will analyze the convergence of the gAM algorithm for the case when the objective function is jointly convex in both its variables. We will then see what happens to Algorithm 3 if the function f is non-convex. To keep the discussion focused, we will consider an unconstrained optimization problem.

Theorem 4.1. Let $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ be jointly convex, continuously differentiable, satisfy β-MSS in both its variables, and $f^* = \min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}, \mathbf{y}) > -\infty$. Let the region $S_0 = \{\mathbf{x}, \mathbf{y} : f(\mathbf{x}, \mathbf{y}) \leq f(\mathbf{0}, \mathbf{0})\} \subset \mathbb{R}^{p+q}$ be bounded, i.e., satisfy $S_0 \subseteq \mathcal{B}_2((\mathbf{0}, \mathbf{0}), R)$ for some $R > 0$. Let Algorithm 3 be executed with the initialization $(\mathbf{x}^1, \mathbf{y}^1) = (\mathbf{0}, \mathbf{0})$. Then after at most $T = \mathcal{O}\left(\frac{1}{\epsilon}\right)$ steps, we have $f(\mathbf{x}^T, \mathbf{y}^T) \leq f^* + \epsilon$.

Proof. The first property of the gAM algorithm that we need to appreciate is *monotonicity*. It is easy to see that due to the marginal minimizations carried out, we have at all time steps t,

$$f(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) \leq f(\mathbf{x}^{t+1}, \mathbf{y}^t) \leq f(\mathbf{x}^t, \mathbf{y}^t)$$

The region S_0 is the *sublevel set* of f at the initialization point. Due to the monotonicity property, we have $f(\mathbf{x}^t, \mathbf{y}^t) \leq f(\mathbf{x}^1, \mathbf{y}^1)$ for all t i.e., $(\mathbf{x}^t, \mathbf{y}^t) \in S_0$ for all t. Thus, gAM remains restricted to the bounded region S_0 and does not diverge. We notice that this point underlies the importance of proper initialization: gAM benefits from being initialized at a point at which the sublevel set of f is bounded.

We will use $\Phi_t = \frac{1}{f(\mathbf{x}^t, \mathbf{y}^t) - f^*}$ as the potential function. This is a slightly unusual choice of potential function but its utility will be clear from the proof. Note that $\Phi_t > 0$ for all t and that convergence is equivalent to showing $\Phi_t \to \infty$. We will, as before, use smoothness to analyze the per-iteration progress made by gAM and use convexity for global convergence analysis. For any time step $t \geq 2$, consider the hypothetical update we could have made had we done a gradient step instead of the marginal minimization step gAM does in step 3.

$$\widetilde{\mathbf{x}}^{t+1} = \mathbf{x}^t - \frac{1}{\beta}\nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t)$$

(Apply Marginal Strong Smoothness) We get

$$\begin{aligned} f(\widetilde{\mathbf{x}}^{t+1}, \mathbf{y}^t) &\leq f(\mathbf{x}^t, \mathbf{y}^t) + \left\langle \nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t), \widetilde{\mathbf{x}}^{t+1} - \mathbf{x}^t, + \right\rangle \frac{\beta}{2} \left\| \widetilde{\mathbf{x}}^{t+1} - \mathbf{x}^t \right\|_2^2 \\ &= f(\mathbf{x}^t, \mathbf{y}^t) - \frac{1}{2\beta} \left\| \nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t) \right\|_2^2 \end{aligned}$$

(Apply Monotonicity of gAM) Since $\mathbf{x}^{t+1} \in \mathsf{mOPT}_f(\mathbf{y}^t)$, we must have $f(\mathbf{x}^{t+1}, \mathbf{y}^t) \leq f(\widetilde{\mathbf{x}}^{t+1}, \mathbf{y}^t)$, which gives us

$$f(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) \leq f(\mathbf{x}^{t+1}, \mathbf{y}^t) \leq f(\mathbf{x}^t, \mathbf{y}^t) - \frac{1}{2\beta} \left\| \nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t) \right\|_2^2$$

Now since $t \geq 2$, we must have had $\mathbf{y}^t \in \arg\min_{\mathbf{y}} f(\mathbf{x}^t, \mathbf{y})$. Since f is differentiable, we must have ([see Bubeck, 2015, Proposition 1.2]) $\nabla_{\mathbf{y}} f(\mathbf{x}^t, \mathbf{y}^t) = \mathbf{0}$. Applying the Pythagoras' theorem now gives us as a result, $\left\| \nabla f(\mathbf{x}^t, \mathbf{y}^t) \right\|_2^2 = \left\| \nabla_{\mathbf{x}} f(\mathbf{x}^t, \mathbf{y}^t) \right\|_2^2$.

(Apply Convexity) Since f is jointly convex, we can state

$$f(\mathbf{x}^t, \mathbf{y}^t) - f^* \leq \left\langle \nabla f(\mathbf{x}^t, \mathbf{y}^t), (\mathbf{x}^t, \mathbf{y}^t) - (\mathbf{x}^*, \mathbf{y}^*) \right\rangle \leq 2R \left\| \nabla f(\mathbf{x}^t, \mathbf{y}^t) \right\|_2,$$

where we have used the Cauchy-Schwartz inequality and the fact that $(\mathbf{x}^t, \mathbf{y}^t), (\mathbf{x}^*, \mathbf{y}^*) \in S_0$. Putting these together gives us

$$f(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) \leq f(\mathbf{x}^t, \mathbf{y}^t) - \frac{1}{4\beta R^2} \left(f(\mathbf{x}^t, \mathbf{y}^t) - f^* \right)^2,$$

or in other words,

$$\frac{1}{\Phi_{t+1}} \leq \frac{1}{\Phi_t} - \frac{1}{4\beta R^2} \frac{1}{\Phi_t^2} \leq \frac{1}{\Phi_t} - \frac{1}{4\beta R^2} \frac{1}{\Phi_t \Phi_{t+1}},$$

where the second step follows from monotonicity. Rearranging gives us

$$\Phi_{t+1} - \Phi_t \geq \frac{1}{4\beta R^2},$$

which upon telescoping, and using $\Phi_2 \geq 0$ gives us

$$\Phi_T \geq \frac{T}{4\beta R^2},$$

which proves the result. Note that the result holds even if f is jointly convex and satisfies the MSS property only *locally* in the region S_0. □

4.4 A Convergence Guarantee for gAM under MSC/MSS

We will now investigate what happens when the gAM algorithm is executed with a non-convex function. Note that the previous result crucially uses convexity and will not extend here. Moreover, for non-convex functions, there is no assurance that all bistable points are global minima. Instead we will have to fend for fast convergence to a bistable point, as well as show that the function is globally minimized there.

Doing the above will require additional structure on the objective function. In the following, we will denote $f^* = \min_{\mathbf{x},\mathbf{y}} f(\mathbf{x}, \mathbf{y})$ to be the optimum value of the objective function. We will fix $(\mathbf{x}^*, \mathbf{y}^*)$ to be any point such that $f(\mathbf{x}^*, \mathbf{y}^*) = f^*$ (there may be several). We will also let $\mathcal{Z}^* \subset \mathbb{R}^p \times \mathbb{R}^q$ denote the set of all bistable points for f.

First of all, notice that if a continuously differentiable function $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ is marginally convex (strongly or otherwise) in both its variables, then its bistable points are exactly its stationary points.

Lemma 4.2. A point $(\mathbf{x}, \mathbf{y})$ is bistable with respect to a continuously differentiable function $f : \mathbb{R}^p \times \mathbb{R}^q$ that is marginally convex in both its variables iff $\nabla f(\mathbf{x}, \mathbf{y}) = \mathbf{0}$.

Proof. It is easy to see that partial derivatives must vanish at a bistable point since the function is differentiable ([see Bubeck, 2015, Proposition 1.2]) and thus we get $\nabla f(\mathbf{x}, \mathbf{y}) = [\nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}), \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y})] = \mathbf{0}$. Arguing the other way round, if the gradient, and by extension the partial derivatives, vanish at $(\mathbf{x}, \mathbf{y})$, then by marginal convexity, for any $\mathbf{x}'$

$$f(\mathbf{x}', \mathbf{y}) - f(\mathbf{x}, \mathbf{y}) \geq \langle \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}), \mathbf{x}' - \mathbf{x} \rangle = 0$$

Similarly, $f(\mathbf{x}, \mathbf{y}') \geq f(\mathbf{x}, \mathbf{y})$ for any $\mathbf{y}'$. Thus $(\mathbf{x}, \mathbf{y})$ is bistable. □

The above tells us that $\mathcal{Z}^*$ is also the set of all stationary points of f. However, not all points in $\mathcal{Z}^*$ may be global minima. Addressing this problem requires careful initialization and problem-specific analysis, that we will carry out for problems such as matrix completion etc in later sections. For now, we introduce a generic *robust* bistability property that will be very useful in the analysis. Similar properties are frequently used in the analysis of gAM-style algorithms.

Definition 4.6 (Robust Bistability Property). A function $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ satisfies the C-robust bistability property if for some $C > 0$, for every $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^q$, $\widetilde{\mathbf{y}} \in \mathsf{mOPT}_f(\mathbf{x})$ and $\widetilde{\mathbf{x}} \in \mathsf{mOPT}_f(\mathbf{y})$, we have

$$f(\mathbf{x}, \mathbf{y}^*) + f(\mathbf{x}^*, \mathbf{y}) - 2f^* \leq C \cdot (2f(\mathbf{x}, \mathbf{y}) - f(\mathbf{x}, \widetilde{\mathbf{y}}) - f(\widetilde{\mathbf{x}}, \mathbf{y})).$$

The right hand expression captures how much one can reduce the function value *locally* by performing marginal optimizations. The property suggests[7] that if not much local improvement can be made (i.e., if $f(\mathbf{x}, \widetilde{\mathbf{y}}) \approx f(\mathbf{x}, \mathbf{y}) \approx f(\widetilde{\mathbf{x}}, \mathbf{y})$) then we are close to the optimum. This has a simple corollary that all bistable points achieve the (globally) optimal function value. We now present a convergence analysis for gAM.

Theorem 4.3. Let $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ be a continuously differentiable (but possibly non-convex) function that, within the region $S_0 = \{\mathbf{x}, \mathbf{y} : f(\mathbf{x}, \mathbf{y}) \leq f(\mathbf{0}, \mathbf{0})\} \subset \mathbb{R}^{p+q}$, satisfies the properties of α-MSC, β-MSS in both its variables, and C-robust bistability. Let Algorithm 3 be executed with the initialization $(\mathbf{x}^1, \mathbf{y}^1) = (\mathbf{0}, \mathbf{0})$. Then after at most $T = \mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ steps, we have $f(\mathbf{x}^T, \mathbf{y}^T) \leq f^* + \epsilon$.

Note that the MSC/MSS and robust bistability properties need only hold within the sublevel set S_0. This again underlines the importance of proper initialization. Also note that gAM offers rapid convergence despite the non-convexity of the objective. In order to prove the result, the following consequence of C-robust bistability will be useful.

Lemma 4.4. Let f satisfy the properties mentioned in Theorem 4.3. Then for any $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^p \times \mathbb{R}^q$, $\widetilde{\mathbf{y}} \in \mathsf{mOPT}_f(\mathbf{x})$ and $\widetilde{\mathbf{x}} \in \mathsf{mOPT}_f(\mathbf{y})$,

$$\|\mathbf{x} - \mathbf{x}^*\|_2^2 + \|\mathbf{y} - \mathbf{y}^*\|_2^2 \leq \frac{C\beta}{\alpha}\left(\|\mathbf{x} - \widetilde{\mathbf{x}}\|_2^2 + \|\mathbf{y} - \widetilde{\mathbf{y}}\|_2^2\right)$$

[7] See Exercise 4.7.

Proof. Applying MSC/MSS repeatedly gives us

$$\begin{aligned} f(\mathbf{x}, \mathbf{y}^*) + f(\mathbf{x}^*, \mathbf{y}) &\geq 2f^* + \frac{\alpha}{2}\left(\|\mathbf{x} - \mathbf{x}^*\|_2^2 + \|\mathbf{y} - \mathbf{y}^*\|_2^2\right) \\ 2f(\mathbf{x}, \mathbf{y}) &\leq f(\mathbf{x}, \widetilde{\mathbf{y}}) + f(\widetilde{\mathbf{x}}, \mathbf{y}) + \frac{\beta}{2}\left(\|\mathbf{x} - \widetilde{\mathbf{x}}\|_2^2 + \|\mathbf{y} - \widetilde{\mathbf{y}}\|_2^2\right) \end{aligned}$$

Applying robust stability then proves the result. □

It is noteworthy that Lemma 4.4 relates local convergence to global convergence and assures us that reaching an almost bistable point is akin to converging to the optimum. Such a result can be crucial, especially for non-convex problems. Indeed, similar properties are used in other proofs concerning coordinate minimization as well, for example, the *local error bound* used in [Luo and Tseng, 1993].

Proof (of Theorem 4.3). We will use $\Phi_t = f(\mathbf{x}^t, \mathbf{y}^t) - f^*$ as the potential function. Since the intermediate steps in gAM are marginal optimizations and not gradient steps, we will actually find it useful to apply marginal strong convexity at a local level, and apply marginal strong smoothness at a global level instead.

(Apply Marginal Strong Smoothness) As $\nabla_{\mathbf{x}} f(\mathbf{x}^*, \mathbf{y}^*) = \mathbf{0}$, applying MSS gives us

$$f(\mathbf{x}^{t+1}, \mathbf{y}^*) - f(\mathbf{x}^*, \mathbf{y}^*) \leq \frac{\beta}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2.$$

Further, the gAM updates ensure $\mathbf{y}^{t+1} \in \mathsf{mOPT}_f(\mathbf{x}^{t+1})$, which gives

$$\Phi_{t+1} = f(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) - f^* \leq f(\mathbf{x}^{t+1}, \mathbf{y}^*) - f^* \leq \frac{\beta}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2,$$

(Apply Marginal Strong Convexity) Since $\nabla_{\mathbf{x}} f(\mathbf{x}^{t+1}, \mathbf{y}^t) = \mathbf{0}$,

$$\begin{aligned} f(\mathbf{x}^t, \mathbf{y}^t) &\geq f(\mathbf{x}^{t+1}, \mathbf{y}^t) + \frac{\alpha}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|_2^2 \\ &\geq f(\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) + \frac{\alpha}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|_2^2, \end{aligned}$$

which gives us

$$\Phi_t - \Phi_{t+1} \geq \frac{\alpha}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|_2^2.$$

This shows that appreciable progress is made in a single step. Now, with $(\mathbf{x}^t, \mathbf{y}^t)$ for any $t \geq 2$, due to the nature of the gAM updates, we know that $\mathbf{y}^t \in \mathsf{mOPT}_f(\mathbf{x}^t)$ and $\mathbf{x}^{t+1} \in \mathsf{mOPT}_f(\mathbf{y}^t)$. Applying Lemma 4.4 then gives us the following inequality

$$\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 \leq \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 + \left\|\mathbf{y}^t - \mathbf{y}^*\right\|_2^2 \leq \frac{C\beta}{\alpha}\left\|\mathbf{x}^t - \mathbf{x}^{t+1}\right\|_2^2$$

Putting these together and using $(a+b)^2 \leq 2(a^2+b^2)$ gives us

$$\begin{aligned}\Phi_{t+1} &\leq \frac{\beta}{2}\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2 \leq \beta\left(\left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|_2^2 + \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2\right)\\ &\leq \beta(1+C\kappa)\left\|\mathbf{x}^{t+1} - \mathbf{x}^t\right\|_2^2 \leq 2\kappa(1+C\kappa)\left(\Phi_t - \Phi_{t+1}\right),\end{aligned}$$

where $\kappa = \frac{\beta}{\alpha}$ is the effective condition number of the problem. Rearranging gives us

$$\Phi_{t+1} \leq \eta_0 \cdot \Phi_t,$$

where $\eta_0 = \frac{2\kappa(1+C\kappa)}{1+2\kappa(1+C\kappa)} < 1$ which proves the result. □

Notice that the condition number κ makes an appearance in the convergence rate of the algorithm but this time, with a fresh definition in terms of the MSC/MSS parameters. As before, small values of κ and C ensure fast convergence, whereas large values of κ, C promote $\eta_0 \to 1$ which slows the procedure down.

Before we conclude, we remind the reader that in later sections, we will see more precise analyses of gAM-style approaches, and the structural assumptions will be more problem specific. However, we hope the preceding discussion has provided some insight into the inner workings of alternating minimization techniques.

4.5 Exercises

Exercise 4.1. Recall the low-rank matrix completion problem in recommendation systems from § 1

$$\widehat{A}_{\mathrm{lv}} = \min_{\substack{U\in\mathbb{R}^{m\times r}\\ V\in\mathbb{R}^{n\times r}}} \sum_{(i,j)\in\Omega}\left(U_i^\top V_j - A_{ij}\right)^2.$$

Show that the objective in this optimization problem is not jointly convex in U and V. Then show that the objective is nevertheless, marginally convex in both the variables.

Exercise 4.2. Show that a function that is jointly convex is necessarily marginally convex as well. Similarly show that a (jointly) strongly convex and smooth function is marginally so as well.

Exercise 4.3. Marginal strong convexity does not imply convexity. Show this by giving an example of a function $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ that is marginally strongly convex in *both* its variables, but non-convex.
Hint: use the fact that the function $f(x) = x^2$ is 2-strongly convex.

Exercise 4.4. Design a variant of the gAM procedure that can handle a general constraint set $\mathcal{Z} \subset \mathcal{X} \times \mathcal{Y}$. Attempt to analyze the convergence of your algorithm.

Exercise 4.5. Show that $(\mathbf{x}^*, \mathbf{y}^*) \in \arg\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y}} f(\mathbf{x}, \mathbf{y})$ must be a bistable point for any function even if f is non-convex.

Exercise 4.6. Let $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ be a differentiable, *jointly* convex function. Show that any bistable point of f is a global minimum for f.
Hint: first show that directional derivatives vanish at bistable points.

Exercise 4.7. For a robustly bistable function f, any *almost* bistable point is *almost* optimal as well. Show this by proving, for any $(\mathbf{x}, \mathbf{y})$, $\widetilde{\mathbf{y}} \in \mathsf{mOPT}_f(\mathbf{x})$, $\widetilde{\mathbf{x}} \in \mathsf{mOPT}_f(\mathbf{y})$ such that $\max\{f(\mathbf{x}, \widetilde{\mathbf{y}}), f(\widetilde{\mathbf{x}}, \mathbf{y})\} \leq f(\mathbf{x}, \mathbf{y}) + \epsilon$, that $f(\mathbf{x}, \mathbf{y}) \leq f^* + \mathcal{O}(\epsilon)$. Conclude that if f satisfies robust bistability, then any bistable point $(\mathbf{x}, \mathbf{y}) \in \mathcal{Z}^*$ is optimal.

Exercise 4.8. Show that marginal strong convexity is additive i.e., if $f, g : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ are two functions such that f is respectively α_1 and α_2-MSC in its two variables and g is $\overline{\alpha}_1$ and $\overline{\alpha}_2$-MSC in its variables, then the function $f + g$ is $(\alpha_1 + \overline{\alpha}_1)$ and $(\alpha_2 + \overline{\alpha}_2)$-MSC in its variables.

Exercise 4.9. The alternating minimization procedure may oscillate if the optimization problem is not well-behaved. Suppose for an especially nasty problem, the gAM procedure enters into the following loop

$$(\mathbf{x}^t, \mathbf{y}^t) \to (\mathbf{x}^{t+1}, \mathbf{y}^t) \to (\mathbf{x}^{t+1}, \mathbf{y}^{t+1}) \to (\mathbf{x}^t, \mathbf{y}^{t+1}) \to (\mathbf{x}^t, \mathbf{y}^t)$$

Show that all four points in the loop are bistable and share the same function value. Can you draw a hypothetical set of marginally optimal coordinate curves which may cause this to happen (see Figure 4.2)?

4.6 Bibliographic Notes

The descent version for CM is aptly named *Coordinate Descent* (CD) and only takes descent steps along the coordinates [Saha and Tewari, 2013]. There exist versions of CM and CD for constrained optimization problems as well [Luo and Tseng, 1993, Nesterov, 2012]. A variant of CM/CD splits variables into *blocks* of multi-dimensional variables. The resulting algorithm is appropriately named *Block Coordinate Descent* and minimizes over one block of variables at each time step.

The coordinate/block being processed at each step is chosen carefully to ensure rapid convergence. Several "rules" exist for this, for example, the Gauss-Southwell rule (that chooses the coordinate/block along which the objective gradient is the largest), the cyclic rule that simply performs a round-robin selection of coordinates/blocks, and random choice that chooses a random coordinate/block at each time step, independent of previous such choices.

For several problem areas such support vector machines, CM/CD methods are at the heart of some of the fastest solvers available due to their speed and ease of implementation [Fan et al., 2008, Luo and Tseng, 1992, Shalev-Shwartz and Zhang, 2013].

5

The EM Algorithm

In this section we will take a look at the *Expectation Maximization* (EM) principle. The principle forms the basis for widely used learning algorithms such as those used for learning Gaussian mixture models, the Baum-Welch algorithm for learning hidden Markov models (HMM), and mixed regression. The EM algorithm is also a close cousin to the Lloyd's algorithm for clustering with the k-means objective.

Although the EM algorithm, at a surface level, follows the alternating minimization principle which we studied in § 5, given its wide applicability in learning latent variable models in probabilistic learning settings, we feel it is instructive to invest in a deeper understanding of the EM method. To make the reading experience self-contained, we will first devote some time developing intuitions and notation in probabilistic learning methods.

Notation A parametric distribution over a domain $\mathcal{X}$ with parameter $\boldsymbol{\theta}$ is denoted by $f(\cdot\,|\,\boldsymbol{\theta})$ or $f_{\boldsymbol{\theta}}$. The notation is abused to let $f(\mathbf{x}\,|\,\boldsymbol{\theta})$ denote the probability mass or density function (as the case may be) of the distribution at the point $\mathbf{x} \in \mathcal{X}$. The notation is also abused to let $X \sim f(\cdot\,|\,\boldsymbol{\theta})$ or $X \sim f_{\boldsymbol{\theta}}$ denote a sample drawn from this distribution.

5.1 A Primer in Probabilistic Machine Learning

Suppose we observe i.i.d. samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ of a random variable $X \in \mathcal{X}$ drawn from an unknown distribution f^*. Suppose also, that it is known that the distribution generating these data samples belongs to a *parametric family* of distributions $\mathcal{F} = \{f_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ such that $f^* = f_{\boldsymbol{\theta}^*}$ for some unknown parameter $\boldsymbol{\theta}^* \in \Theta$.

How may we recover (an accurate estimate of) the true parameter $\boldsymbol{\theta}^*$, using only the samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$? There are several ways to do so, popular among them being the *maximum likelihood* estimate. Since the samples were generated independently, one can, for any parameter $\boldsymbol{\theta}^0 \in \Theta$, write their joint density function as follows

$$f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \,|\, \boldsymbol{\theta}^0) = \prod_{i=1}^{n} f(\mathbf{x}_i \,|\, \boldsymbol{\theta}^0)$$

The above quantity is also known as the *likelihood* of the data parametrized on $\boldsymbol{\theta}^0$ as it captures the probability that the observed data was generated by the parameter $\boldsymbol{\theta}^0$. In fact, we can go ahead and define the likelihood function for any parameter $\boldsymbol{\theta} \in \Theta$ as follows

$$\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) := f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \,|\, \boldsymbol{\theta})$$

The maximum likelihood estimate (MLE) of $\boldsymbol{\theta}^*$ is simply the parameter that maximizes the above likelihood function i.e, the parameter which seems to be the "most likely" to have generated the data.

$$\widehat{\boldsymbol{\theta}}_{\text{MLE}} := \underset{\boldsymbol{\theta} \in \Theta}{\arg\max}\ \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$$

It is interesting to study the convergence of $\widehat{\boldsymbol{\theta}}_{\text{MLE}} \to \boldsymbol{\theta}^*$ as $n \to \infty$ but we will not do so in this monograph. We note that there do exist other estimation techniques apart from MLE, such as the *Maximum a Posteriori* (MAP) estimate that incorporates a *prior* distribution over $\boldsymbol{\theta}$, but we will not discuss those here either.

Least Squares Regression As a warmup exercise, let us take the example of linear regression and reformulate it in a probabilistic setting to better understand the above framework. Let $y \in \mathbb{R}$ and $\mathbf{w}, \mathbf{x} \in \mathbb{R}^p$

and consider the following parametric distribution over the set of reals, parametrized by $\mathbf{w}$ and $\mathbf{x}$

$$f(y \,|\, \mathbf{x}, \mathbf{w}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w})^2}{2}\right),$$

Note that this distribution exactly encodes the responses in a linear regression model with unit variance Gaussian noise. More specifically, if $y \sim f(\cdot \,|\, \mathbf{x}, \mathbf{w})$, then

$$y \sim \mathcal{N}(\mathbf{x}^\top \mathbf{w}, 1)$$

The above observation allows us to cast linear regression as a parameter estimation problem. Consider the parametric distribution family

$$\mathcal{F} = \{f_{\mathbf{w}} = f(\cdot \,|\, \cdot, \mathbf{w}) : \|\mathbf{w}\|_2 \leq 1\}.$$

Suppose now that we have n covariate samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ and there is a true parameter $\mathbf{w}^*$ such that the distribution $f_{\mathbf{w}^*} \in \mathcal{F}$ (i.e., $\|\mathbf{w}^*\|_2 \leq 1$) is used to generate the responses i.e., $y_i \sim f(\cdot \,|\, \mathbf{x}_i, \mathbf{w}^*)$. It is easy to see that the likelihood function in this setting is[1]

$$\mathcal{L}(\mathbf{w}; \{(\mathbf{x}_i, y_i)\}_{i=1}^n) = \prod_{i=1}^n f(y_i \,|\, \mathbf{x}_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi}} \prod_{i=1}^n \exp\left(-\frac{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}{2}\right)$$

Since the logarithm function is a strictly increasing function, maximizing the *log-likelihood* will also yield the MLE, i.e.,

$$\widehat{\mathbf{w}}_{\text{MLE}} = \underset{\|\mathbf{w}\|_2 \leq 1}{\arg\max}\; \log \mathcal{L}(\mathbf{w}; \{(\mathbf{x}_i, y_i)\}_{i=1}^n) = \underset{\|\mathbf{w}\|_2 \leq 1}{\arg\min}\; \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \mathbf{w})^2$$

Thus, the MLE for linear regression under Gaussian noise is nothing but the common least squares estimate! The theory of maximum likelihood estimates and their consistency properties is well studied and under suitable conditions, we indeed have $\widehat{\mathbf{w}}_{\text{MLE}} \to \mathbf{w}^*$. ML estimators are members of a more general class of estimators known as *M-estimators* [Huber and Ronchetti, 2009].

[1] The reader would notice that we are modeling only the process that generates the responses *given* the covariates. However, this is just for sake of simplicity. It is possible to model the process that generates the covariates $\mathbf{x}_i$ as well using, for example, a mixture of Gaussians (that we will study in this very section). A model that accounts for the generation of both $\mathbf{x}_i$ and y_i is called a *generative* model.

5.2 Problem Formulation

In practical applications, the probabilistic models one encounters are often more challenging than the one for linear regression due to the presence of *latent variables* which necessitates the use of more careful routines like the EM algorithm to even calculate the MLE.

Consider a statistical model that generates two random variables $Y \in \mathcal{Y}$ and $Z \in \mathcal{Z}$, instead of one, using a distribution from a parametric family $\mathcal{F} = \{f_{\boldsymbol{\theta}} = f(\cdot, \cdot \,|\, \boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$ i.e., $(Y, Z) \sim f_{\boldsymbol{\theta}^*}$ for some $\boldsymbol{\theta}^* \in \Theta$. However, we only get to see realizations of the Y components and not the Z components. More specifically, although the (unknown) parameter $\boldsymbol{\theta}^*$ generates pairs of samples $(\mathbf{y}_1, \mathbf{z}_1), (\mathbf{y}_2, \mathbf{z}_2), \ldots, (\mathbf{y}_n, \mathbf{z}_n)$, only $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ are revealed to us. The missing Z components are often called *latent variables* since they are hidden from us.

The above situation arises in several practical situations in data modeling, clustering, and analysis. We encourage the reader to momentarily skip to § 5.6 to look at a few nice examples before returning to proceed with this discussion. A first attempt at obtaining the MLE in such a scenario would be to maximize the *marginal likelihood* function instead. Assume for the sake of simplicity that the support set of the random variable Z, i.e., $\mathcal{Z}$, is discrete. Then the marginal likelihood function is defined as the following

$$\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n) = \prod_{i=1}^{n} f(\mathbf{y}_i \,|\, \boldsymbol{\theta}) = \prod_{i=1}^{n} \sum_{\mathbf{z}_i \in \mathcal{Z}} f(\mathbf{y}_i, \mathbf{z}_i \,|\, \boldsymbol{\theta}).$$

In most practical situations, using the marginal likelihood function $\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ to perform ML estimation becomes intractable since the expression on the right hand side, when expanded as a sum, contains $|\mathcal{Z}|^n$ terms which makes it difficult to even write down the expression fully let alone optimize using it as an objective function!

For comparison, the log-likelihood expression for the linear regression problem with n data points (the least squares expression) was a summation of n terms. Indeed, the problem of maximizing the marginal likelihood function $\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ is often NP-hard and as a consequence, direct optimization techniques for finding the MLE fail for even small scale problems.

Algorithm 4 AltMax for Latent Variable Models (AM-LVM)

```
Input: Data points $\mathbf{y}_1, \ldots, \mathbf{y}_n$
Output: An approximate MLE $\widehat{\boldsymbol{\theta}} \in \Theta$
1: $\boldsymbol{\theta}^1 \leftarrow$ INITALIZE()
2: for $t = 1, 2, \ldots$ do
3:     for $i = 1, 2, \ldots, n$ do
4:         $\widehat{\mathbf{z}}_i^t \leftarrow \arg\max_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{z} \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)$          (Estimate latent variables)
5:     end for
6:     $\boldsymbol{\theta}^{t+1} \leftarrow \arg\max_{\boldsymbol{\theta} \in \Theta} \log \mathcal{L}(\boldsymbol{\theta}; \{(\mathbf{y}_i, \widehat{\mathbf{z}}_i^t)\}_{i=1}^n)$          (Update parameter)
7: end for
8: return $\mathbf{w}^t$
```

5.3 An Alternating Maximization Approach

Notice that the key reason for the intractability of the MLE problem in the previous discussion was the missing information about the latent variables. Had the latent variables $\mathbf{z}_1, \ldots, \mathbf{z}_n$ been magically provided to us, it would have been simple to find the MLE solution as follows

$$\widehat{\boldsymbol{\theta}}_{\text{MLE}} = \underset{\boldsymbol{\theta} \in \Theta}{\arg\max} \; \log \mathcal{L}(\boldsymbol{\theta}; \{(\mathbf{y}_i, \mathbf{z}_i)\}_{i=1}^n)$$

However, notice that it is also true that, had the identity of the true parameter $\boldsymbol{\theta}^*$ been provided to us (again magically), it would have been simple to estimate the latent variables using a maximum posterior probability estimate for $\mathbf{z}_i$ as follows

$$\widehat{\mathbf{z}}_i = \underset{\mathbf{z} \in \mathcal{Z}}{\arg\max} \; f(\mathbf{z} \,|\, \mathbf{y}_i, \boldsymbol{\theta}^*)$$

Given the above, it is tempting to apply a gAM-style algorithm to solve the MLE problem in the presence of latent variables. Algorithm 4 outlines such an adaptation of gAM to the latent variable learning problem. Note that steps 4 and 6 in the algorithm can be very efficiently carried out for several problem cases. In fact, it can be shown[2] that for the Gaussian Mixture modeling problem, AM-LVM reduces to the popular Llyod's algorithm for k-means clustering.

[2] See Exercise 5.1.

However, the AM-LVM algorithm has certain drawbacks, especially when the space of latent variables $\mathcal{Z}$ is large. At every time step t, AM-LVM makes a "hard assignment", assigning the data point $\mathbf{y}_i$ to just one value of the latent variable $\mathbf{z}_i^t \in \mathcal{Z}$. This can amount to throwing away a lot of information, especially when there may be other values $\mathbf{z}' \in \mathcal{Z}$ present such that $f(\mathbf{z}' \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)$ is also large but nevertheless $f(\mathbf{z}' \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t) < f(\mathbf{z}_i^t \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)$ so that AM-LVM neglects $\mathbf{z}'$. The EM algorithm tries to remedy this.

5.4 The EM Algorithm

Given the drawbacks of the hard assignment approach adopted by the AM-LVM algorithm, the EM algorithm presents an alternative approach that can be seen as making "soft" assignments.

At a very high level, the EM algorithm can be seen as doing the following. At time step t, instead of assigning the point $\mathbf{y}_i$ to a single value of the latent variable $\widehat{\mathbf{z}}_i^t = \underset{\mathbf{z} \in \mathcal{Z}}{\arg\max}\, f(\mathbf{z} \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)$, the EM algorithm chooses to make a partial assignment of $\mathbf{y}_i$ to *all* possible values of the latent variable in the set $\mathcal{Z}$. The EM algorithm "assigns" $\mathbf{y}_i$ to a value $\mathbf{z} \in \mathcal{Z}$ with affinity/weight equal to $f(\mathbf{z} \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)$.

Thus, $\mathbf{y}_i$ gets partially assigned to all possible values of the latent variable, to some with more weight, to others with less weight. This avoids any loss of information. Note that these weights are always positive and sum up to unity since $\sum_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{z} \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t) = 1$. Also note that EM still assigns $\widehat{\mathbf{z}}_i^t = \underset{\mathbf{z} \in \mathcal{Z}}{\arg\max}\, f(\mathbf{z} \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)$ the highest weight. In contrast, AM-LVM can be now seen as putting all the weight on $\widehat{\mathbf{z}}_i^t$ alone and zero weight on any other latent variable value.

We now present a more formal derivation of the EM algorithm. Instead of maximizing the likelihood in a single step, the EM algorithm tries to efficiently *encourage* an increase in the likelihood over several steps. Define the *point-wise likelihood* function as

$$\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}) = \sum_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta}).$$

Note that we can write the marginal likelihood function as $\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n) = \prod_{i=1}^n \mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_i)$. Our goal is to maximize

$\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ but doing so directly is too expensive. So the next best thing is to do so indirectly. Suppose we had a proxy function that lower bounded the likelihood function but was also easy to optimize. Then maximizing the proxy function would also lead to an increase in the likelihood if the proxy were really good.

This is the key to the EM algorithm: it introduces a proxy function called the *Q-function* that lower bounds the marginal likelihood function and casts the parameter estimation problem as a bi-variate problem, the two variables being the parameter $\boldsymbol{\theta}$ and the Q-function.

Given an initialization, $\boldsymbol{\theta}^0 \in \Theta$, EM constructs a Q-function out of it, uses that as a proxy to obtain a better parameter $\boldsymbol{\theta}^1$, uses the newly obtained parameter to construct a better Q-function, uses the better Q-function to obtain a still better parameter $\boldsymbol{\theta}^2$, and so on. Thus, it essentially performs alternating optimization steps, with better estimations of the $\boldsymbol{\theta}$ parameter leading to better constructions of the Q-function and vice versa.

To formalize this notion, we will abuse notation to let $f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0)$ denote the conditional probability function for the random variable Z given the variable Y and the parameter $\boldsymbol{\theta}^0$. Then we have

$$\log \mathcal{L}(\boldsymbol{\theta}; \mathbf{y}) = \log \sum_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta}) = \log \sum_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0) \frac{f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta})}{f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0)}$$

The summation in the last expression can be seen to be simply an expectation with respect to the random variable Z being sampled from the conditional distribution $f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^0)$. Using this, we get

$$\begin{aligned}
\log \mathcal{L}(\boldsymbol{\theta}; \mathbf{y}) &= \log \mathbb{E}_{\mathbf{z} \sim f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^0)} \left[\frac{f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta})}{f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0)} \right] \\
&\geq \mathbb{E}_{\mathbf{z} \sim f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^0)} \left[\log \frac{f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta})}{f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0)} \right] \\
&= \underbrace{\mathbb{E}_{\mathbf{z} \sim f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^0)} \left[\log f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta}) \right]}_{Q_{\mathbf{y}}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^0)} - \underbrace{\mathbb{E}_{\mathbf{z} \sim f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^0)} \left[\log f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0) \right]}_{R_{\mathbf{y}}(\boldsymbol{\theta}^0)}.
\end{aligned}$$

The inequality follows from Jensen's inequality as the logarithm function is concave. The function $Q_{\mathbf{y}}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^0)$ is called the *point-wise* Q-

function. Now, the Q-function can be interpreted as a *weighted* point-wise likelihood function

$$Q_{\mathbf{y}}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^0) = \sum_{\mathbf{z} \in \mathcal{Z}} w_{\mathbf{z}} \cdot \log f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta}),$$

with weights $w_{\mathbf{z}} = f(\mathbf{z} \,|\, \mathbf{y}, \theta^0)$. Notice that this exactly corresponds to assigning $\mathbf{y}$ to every $\mathbf{z} \in \mathcal{Z}$ with weight $w_{\mathbf{z}}$ instead of assigning it to just one value $\widehat{\mathbf{z}} = \arg\max_{\mathbf{z} \in \mathcal{Z}} \; f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^0)$ (with weight 1) as AM-LVM does. We will soon see that due to the way the Q-function is used, the EM algorithm can be seen as performing AM-LVM with a soft-assignment step instead of a hard assignment step.

Using the point-wise Q-function, we define the Q-function.

$$Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^0) = \frac{1}{n} \sum_{i=1}^{n} Q_{\mathbf{y}_i}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^0)$$

The Q-function has all the properties we desired from our proxy: parameters maximizing the function $Q(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^0)$ do indeed improve the likelihood $\mathcal{L}(\boldsymbol{\theta}; \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$. More importantly, for several applications, it is possible to both construct as well as optimize the Q-function, very efficiently.

Algorithm 5 gives an overall skeleton of the EM algorithm. Implementing EM requires two routines, one to construct the Q-function corresponding to the current iterate (the *Expectation step* or E-step), and the other to maximize the Q-function (the *Maximization step* or M-step) to get the next iterate. We will next give precise constructions of these steps.

EM works well in many situations where the marginal likelihood function is inaccessible. This is because the Q-function only requires access to the conditional probability function $f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^0)$ and the joint probability function $f(\cdot, \cdot \,|\, \boldsymbol{\theta})$, both of which are readily accessible in several applications. We will see examples of such applications shortly.

5.5 Implementing the E/M steps

We will now look at various implementations of the E and the M steps in the EM algorithm. Some would be easier to implement in practice whereas others would be easier to analyze.

Algorithm 5 Expectation Maximization (EM)

```
Input: Implementations of the E-step $E(\cdot)$, and the M-step $M(\cdot)$
Output: A good parameter $\widehat{\boldsymbol{\theta}} \in \Theta$
$\boldsymbol{\theta}^1 \leftarrow$ INITALIZE()
for $t = 1, 2, \ldots$ do
    $Q_t(\cdot \,|\, \boldsymbol{\theta}^t) \leftarrow E(\boldsymbol{\theta}^t)$                    (E-step)
    $\boldsymbol{\theta}^{t+1} \leftarrow M(\boldsymbol{\theta}^t, Q_t)$                    (M-step)
end for
return $\mathbf{w}^t$
```

E-step Constructions Given the definition of the point-wise Q-function, one can use it to construct the Q-function in several ways. The first construction is of largely theoretical interest but reveals a lot of the inner workings of the EM algorithm by simplifying the proofs. This *population* construction requires access to the marginal distribution on the Y variable i.e., the probability function $f(\mathbf{y} \,|\, \boldsymbol{\theta}^*)$. Recall that $\boldsymbol{\theta}^*$ is the true parameter generating these samples. Given this, the population E-step constructs the Q-function as

$$\begin{aligned}
Q_t^{\text{pop}}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^t) &= \mathbb{E}_{\mathbf{y} \sim f_{\boldsymbol{\theta}^*}} Q_{\mathbf{y}}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^t) \\
&= \mathbb{E}_{\mathbf{y} \sim f_{\boldsymbol{\theta}^*}} \mathbb{E}_{\mathbf{z} \sim f(\cdot \,|\, \mathbf{y}, \boldsymbol{\theta}^t)} \left[\log f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta}) \right] \\
&= \sum_{\mathbf{y} \in \mathcal{Y}} \sum_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{y} \,|\, \boldsymbol{\theta}^*) \cdot f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^t) \cdot \log f(\mathbf{y}, \mathbf{z} \,|\, \boldsymbol{\theta})
\end{aligned}$$

Clearly this construction is infeasible in practice. A much more realistic *sample* construction works with just the observed samples $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ (note that these were indeed drawn from the distribution $f(\mathbf{y} \,|\, \boldsymbol{\theta}^*)$). The sample E-step constructs the Q-function as

$$\begin{aligned}
Q_t^{\text{sam}}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^t) &= \frac{1}{n} \sum_{i=1}^{n} Q_{\mathbf{y}_i}(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^t) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{\mathbf{z} \sim f(\cdot \,|\, \mathbf{y}_i, \boldsymbol{\theta}^t)} \left[\log f(\mathbf{y}_i, \mathbf{z} \,|\, \boldsymbol{\theta}) \right] \\
&= \frac{1}{n} \sum_{i=1}^{n} \sum_{\mathbf{z} \in \mathcal{Z}} f(\mathbf{z} \,|\, \mathbf{y}, \boldsymbol{\theta}^t) \cdot \log f(\mathbf{y}_i, \mathbf{z} \,|\, \boldsymbol{\theta})
\end{aligned}$$

Note that this expression has $n \cdot |\mathcal{Z}|$ terms instead of the $|\mathcal{Z}|^n$ terms which the marginal likelihood expression had. This drastic reduction is a key factor behind the scalability of the EM algorithm.

M-step Constructions Recall that in § 4 we considered alternating approaches that fully optimize with respect to a variable, as well as those that merely perform a descent step, improving the function value along that variable but not quite optimizing it.

Similar variants can be developed for EM as well. Given a Q-function, the simplest strategy is to optimize it completely with the M-step simply returning the maximizer of the Q-function. This is the *fully corrective* version of EM. It is useful to remind ourselves here that whereas in previous sections we looked at minimization problems, the problem here is that of likelihood *maximization*.

$$M^{\mathrm{fc}}(\boldsymbol{\theta}^t, Q_t) = \arg\max_{\boldsymbol{\theta} \in \Theta} \; Q_t(\boldsymbol{\theta} \,|\, \boldsymbol{\theta}^t)$$

Since this can be expensive in large scale optimization settings, a *gradient descent* version exists that makes the M-step faster by performing just a gradient step with respect to the Q-function i.e.,

$$M^{\mathrm{grad}}(\boldsymbol{\theta}^t, Q_t) = \boldsymbol{\theta}^t + \alpha_t \cdot \nabla Q_t(\boldsymbol{\theta}^t \,|\, \boldsymbol{\theta}^t).$$

Stochastic EM Construction A highly scalable version of the algorithm is the *stochastic update* version that uses the point-wise Q-function of a single, randomly chosen sample $Y_t \sim \mathsf{Unif}[n]$ to execute a gradient update at each time step t. It can be shown[3] that on expectation, this executes the sample E-step and the gradient M-step.

$$M^{\mathrm{sto}}(\boldsymbol{\theta}^t) = \boldsymbol{\theta}^t + \alpha_t \cdot \nabla Q_{Y_t}(\boldsymbol{\theta}^t \,|\, \boldsymbol{\theta}^t)$$

5.6 Motivating Applications

We will now look at a few applications of the EM algorithm and see how the E and M steps are executed efficiently. We will revisit these applications while discussing convergence proofs for the EM algorithm.

[3] See Exercise 5.2.

5.6.1 Gaussian Mixture Models

Mixture models are ubiquitous in applications such as clustering, topic modeling, and segmentation tasks. Let $\mathcal{N}(\cdot; \boldsymbol{\mu}, \Sigma)$ denote the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^p$ and covariance $\Sigma \in \mathbb{R}^{p \times p}$. A mixture model can be constructed by combining two such normal distributions to obtain a density function of the form

$$f(\cdot, \cdot \mid \left\{\phi_i, \boldsymbol{\mu}^{*,i}, \Sigma_i\right\}_{i \in \{0,1\}}) = \phi_0^* \cdot \mathcal{N}_0 + \phi_1^* \cdot \mathcal{N}_1,$$

where $\mathcal{N}_i = \mathcal{N}(\cdot; \boldsymbol{\mu}^{*,i}, \Sigma_i^*), i = 0, 1$ are the *mixture components* and $\phi_i^* \in (0, 1), i = 0, 1$ are the *mixture coefficients*. We insist that $\phi_0^* + \phi_1^* = 1$ to ensure that f is indeed a probability distribution. Note that we consider a mixture with just two components for simplicity. Mixture models with larger number of components can be similarly constructed.

A sample $(\mathbf{y}, z) \in \mathbb{R}^p \times \{0, 1\}$ can be drawn from this distribution by first tossing a Bernoulli coin with bias ϕ_1 to choose a component $z \in \{0, 1\}$ and then drawing a sample $\mathbf{y} \sim \mathcal{N}_z$ from that component.

However, despite drawing the samples $(\mathbf{y}_1, z_1), (\mathbf{y}_2, z_2), \ldots, (\mathbf{y}_n, z_n)$, what is presented to us is $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ i.e., the identities z_i of the components that actually resulted in these draws is hidden from us. For instance in topic modeling tasks, the underlying topics being discussed in documents is hidden from us and we only get to see surface realizations of words in the documents that have topic-specific distributions. The goal here is to recover the mixture components as well as coefficients in an efficient manner from such partially observed draws.

For the sake of simplicity, we will look at a balanced, isotropic mixture i.e., where we are given that $\phi_0^* = \phi_1^* = 0.5$ and $\Sigma_0^* = \Sigma_1^* = I_p$. This will simplify our updates and analysis as the only unknown parameters in the model are $\boldsymbol{\mu}^{*,0}$ and $\boldsymbol{\mu}^{*,1}$. Let $\mathbf{M} = (\boldsymbol{\mu}^0, \boldsymbol{\mu}^1) \in \mathbb{R}^{p \times p}$ denote an ensemble describing such a parametric mixture. Our job is to recover $\mathbf{M}^* = (\boldsymbol{\mu}^{*,0}, \boldsymbol{\mu}^{*,1})$.

E-step Construction For any $\mathbf{M} = (\boldsymbol{\mu}^0, \boldsymbol{\mu}^1)$, we have the mixture

$$f(\cdot, \cdot \mid \mathbf{M}) = 0.5 \cdot \mathcal{N}(\cdot; \boldsymbol{\mu}^0, I_p) + 0.5 \cdot \mathcal{N}(\cdot; \boldsymbol{\mu}^1, I_p),$$

i.e., $\mathcal{N}_i = \mathcal{N}(\cdot; \boldsymbol{\mu}^i, I_p)$. In this case, the Q-function actually has a closed form expression. For any $\mathbf{y} \in \mathbb{R}^p$, $z \in \{0,1\}$, and $\mathbf{M}$, we have

$$\begin{aligned} f(\mathbf{y}, z \,|\, \mathbf{M}) &= \mathcal{N}_z(\mathbf{y}) = \exp\left(-\frac{\|\mathbf{y} - \boldsymbol{\mu}^z\|_2^2}{2}\right) \\ f(z \,|\, \mathbf{y}, \mathbf{M}) &= \frac{f(\mathbf{y}, z \,|\, \mathbf{M})}{f(\mathbf{y} \,|\, \mathbf{M})} = \frac{f(\mathbf{y}, z \,|\, \mathbf{M})}{f(\mathbf{y}, z \,|\, \mathbf{M}) + f(\mathbf{y}, 1-z \,|\, \mathbf{M})}. \end{aligned}$$

Thus, even though the marginal likelihood function was inaccessible, the point-wise Q-function has a nice closed form. For the E-step construction, for any two ensembles $\mathbf{M}^t = (\boldsymbol{\mu}^{t,0}, \boldsymbol{\mu}^{t,1})$ and $\mathbf{M} = (\boldsymbol{\mu}^0, \boldsymbol{\mu}^1)$,

$$\begin{aligned} Q_{\mathbf{y}}(\mathbf{M} \,|\, \mathbf{M}^t) &= \mathbb{E}_{z \sim f(\cdot \,|\, \mathbf{y}, \mathbf{M}^t)} \left[\log f(\mathbf{y}, z \,|\, \mathbf{M})\right] \\ &= f(0 \,|\, \mathbf{y}, \mathbf{M}^t) \cdot \log f(\mathbf{y}, 0 \,|\, \mathbf{M}) + f(1 \,|\, \mathbf{y}, \mathbf{M}^t) \cdot \log f(\mathbf{y}, 1 \,|\, \mathbf{M}) \\ &= -\frac{1}{2}\left(w_t^0(\mathbf{y}) \cdot \left\|\mathbf{y} - \boldsymbol{\mu}^0\right\|_2^2 + w_t^1(\mathbf{y}) \cdot \left\|\mathbf{y} - \boldsymbol{\mu}^1\right\|_2^2\right), \end{aligned}$$

where $w_t^z(\mathbf{y}) = e^{-\frac{\|\mathbf{y} - \boldsymbol{\mu}^{t,z}\|_2^2}{2}} \left(e^{-\frac{\|\mathbf{y} - \boldsymbol{\mu}^{t,0}\|_2^2}{2}} + e^{-\frac{\|\mathbf{y} - \boldsymbol{\mu}^{t,1}\|_2^2}{2}}\right)^{-1}$. Note that $w_t^z(\mathbf{y}) \geq 0$ for $z = 0, 1$ and $w_t^0(\mathbf{y}) + w_t^1(\mathbf{y}) = 1$. Also note that $w_t^z(\mathbf{y})$ is larger if $\mathbf{y}$ is closer to $\boldsymbol{\mu}^{t,z}$ i.e., it measures the affinity of a point to the center. Given a sample of data points $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$, the Q-function is

$$\begin{aligned} Q(\mathbf{M} \,|\, \mathbf{M}^t) &= \frac{1}{n}\sum_{i=1}^n Q_{\mathbf{y}_i}(\mathbf{M} \,|\, \mathbf{M}^t) \\ &= -\frac{1}{2n}\sum_{i=1}^n w_t^0(\mathbf{y}_i) \cdot \left\|\mathbf{y}_i - \boldsymbol{\mu}^0\right\|_2^2 + w_t^1(\mathbf{y}_i) \cdot \left\|\mathbf{y}_i - \boldsymbol{\mu}^1\right\|_2^2 \end{aligned}$$

M-step Construction This also has a closed form solution in this case[4]. If $\mathbf{M}^{t+1} = (\boldsymbol{\mu}^{t+1,0}, \boldsymbol{\mu}^{t+1,1}) = \arg\max_{\mathbf{M}} Q(\mathbf{M} \,|\, \mathbf{M}^t)$, then

$$\boldsymbol{\mu}^{t+1,z} = \sum_{i=1}^n w_t^z(\mathbf{y}_i)\mathbf{y}_i$$

Note that the M-step can be executed in linear time and does not require an explicit construction of the Q-function at all! One just needs to use the M-step repeatedly – the Q-function is implicit in the M-step.

[4] See Exercise 5.3.

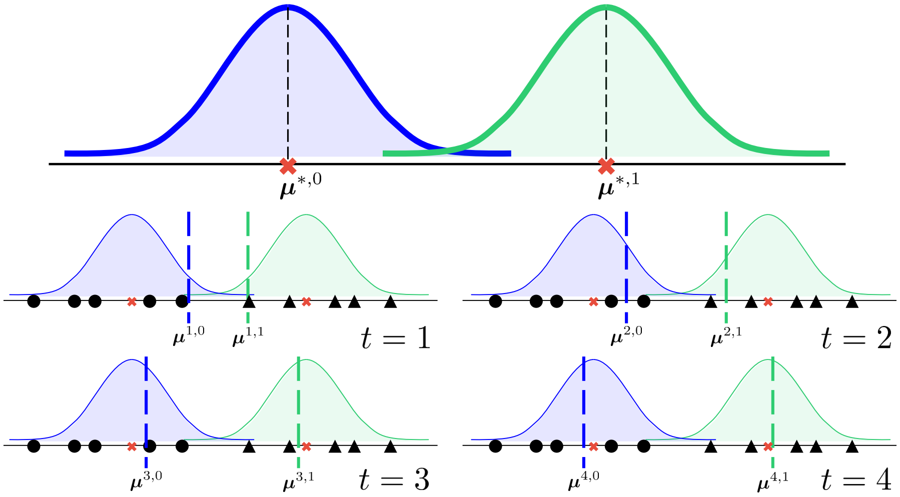


Figure 5.1: The data is generated from a mixture model: circle points from $\mathcal{N}(\boldsymbol{\mu}^{*,0}, I_p)$ and triangle points from $\mathcal{N}(\boldsymbol{\mu}^{*,1}, I_p)$ but their origin is unknown. The EM algorithm performs soft clustering assignments to realize this and keeps increasing w_t^0 values for circle points and increasing w_t^1 values for triangle points. As a result, the estimated means $\boldsymbol{\mu}^{t,z}$ rapidly converge to the true means $\boldsymbol{\mu}^{*,z}$, $z = 0, 1$.

The reader would notice a similarity between the EM algorithm for Gaussian mixture models and the Llyod's algorithm [Lloyd, 1982] for k-means clustering. In fact[5], the Llyod's algorithm implements exactly the AM-LVM algorithm (see Algorithm 4) that performs "hard" assignments, assigning each data point completely to one of the clusters whereas the EM algorithm makes "soft" assignments, allowing each point to have different levels of affinity to different clusters. Figure 5.1 depicts the working of the EM algorithm on a toy GMM problem.

5.6.2 Mixed Regression

As we have seen in § 1, regression problems and their variants have several applications in data analysis and machine learning. One such variant is that of mixed regression. Mixed regression is especially useful when we suspect that our data is actually composed of several *sub-populations* which cannot be explained well using a single model.

[5] See Exercise 5.1.

For example, consider the previous example of predicting family expenditure. Although we may have data from families across a nation, it may be unwise to try and explain it using a single model due to various reasons. The prices of various commodities and services may vary across urban and rural areas and similar consumption in two regions may very well result in different expenditures. Moreover, there may exist parameters such as total income which are not revealed in a survey due to privacy issues, but nevertheless influence expenditure.

Thus, there may actually be several models, each corresponding to a certain income bracket or a certain geographical region, which *together* explain the data very well. This poses a challenge since the income bracket, or geographical location of a family may not have been recorded as a part of the survey due to privacy or other reasons!

To formalize the above scenario, consider two linear models $\mathbf{w}^{*,0}, \mathbf{w}^{*,1} \in \mathbb{R}^p$. For each data point $\mathbf{x}_i \in \mathbb{R}^p$, first one of the models is selected by performing a Bernoulli trial with bias ϕ_1 to get $z_i \in \{0, 1\}$ and then the response is generated as

$$y_i = \mathbf{x}_i^\top \mathbf{w}^{*,z_i} + \eta_i$$

where η_i is i.i.d. Gaussian noise $\eta_i \sim \mathcal{N}(0, \sigma_{z_i}^2)$. This can be cast as a parametric model by considering density functions of the form

$$f(\cdot \,|\, \cdot, \{\phi_z, \mathbf{w}^{*,z}, \sigma_z\}_{z=0,1}) = \phi_0 \cdot g(\cdot \,|\, \cdot, \mathbf{w}^{*,0}, \sigma_0) + \phi_1 \cdot g(\cdot \,|\, \cdot, \mathbf{w}^{*,1}, \sigma_1),$$

where $\sigma_z, \phi_z > 0$, $\phi_0 + \phi_1 = 1$, and for any $(\mathbf{x}, y) \in \mathbb{R}^p \times \mathbb{R}$, we have

$$g(y \,|\, \mathbf{x}, \mathbf{w}^{*,z}, \sigma_z) = \exp\left(-\frac{(y - \mathbf{x}^\top \mathbf{w}^{*,z})^2}{2\sigma_z^2}\right).$$

Note that although such a model generates data in the form of triplets $(\mathbf{x}_1, y_1, z_1), (\mathbf{x}_2, y_2, z_2), \ldots, (\mathbf{x}_n, y_n, z_n)$, we are only allowed to observe $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$ as the data. For the sake of simplicity, we will yet again look at the special case when the Bernoulli trials are fair i.e., $\phi_0 = \phi_1 = 0.5$ and $\sigma_1 = \sigma_2 = 1$. Thus, the only unknown parameters are the models $\mathbf{w}^{*,0}$ and $\mathbf{w}^{*,1}$. Let $\mathbf{W} = (\mathbf{w}^0, \mathbf{w}^1) \in \mathbb{R}^{p \times p}$ denote the parametric mixed model. Our job is to recover $\mathbf{W}^* = (\mathbf{w}^{*,0}, \mathbf{w}^{*,1})$.

A particularly interesting special case arises when we further impose the constraint $\mathbf{w}^{*,0} = -\mathbf{w}^{*,1}$, i.e. the two models in the mixture are tied together to be negative of each other. This model is especially useful in the *phase retrieval* problem. Although we will study this problem in more generality in § 10, we present a special case here.

Phase retrieval is a problem that arises in several imaging situations such as X-ray crystallography where, after data $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$ has been generated as $y_i = \langle \mathbf{w}^*, \mathbf{x}_i \rangle$, the sign of the response (or more generally, the phase of the response if the response is complex-valued) is omitted and we are presented with just $\{(\mathbf{x}_i, |y_i|)\}_{i=1,\ldots,n}$. In such a situation, we can use the latent variable $z_i = \text{sign}(y_i)$ to denote the omitted sign information. In this setting, it can be seen that the mixture model with $\mathbf{w}^{*,0} = -\mathbf{w}^{*,1}$ is very appropriate since each data point $(\mathbf{x}_i, |y_i|)$ will be nicely explained by either $\mathbf{w}^*$ or $-\mathbf{w}^*$ depending on the value of $\text{sign}(y_i)$.

We will revisit this problem in detail in § 10. For now we move on to discuss the E and M-step constructions for the mixed regression problem. We leave details of the constructions as an exercise[6].

E-step Construction The point-wise Q-function has a closed form expression. Given two ensembles $\mathbf{W}^t = (\mathbf{w}^{t,0}, \mathbf{w}^{t,1})$ and $\mathbf{W} = (\mathbf{w}^0, \mathbf{w}^1)$,

$$Q_{(\mathbf{x},y)}(\mathbf{W}|\mathbf{W}^t) = -\frac{1}{2}\left(\alpha^0_{t,(\mathbf{x},y)} \cdot (y - \mathbf{x}^\top \mathbf{w}^0)^2 - \alpha^1_{t,(\mathbf{x},y)} \cdot (y - \mathbf{x}^\top \mathbf{w}^1)^2\right),$$

where $\alpha^z_{t,(\mathbf{x},y)} = e^{-\frac{(y-\mathbf{x}^\top \mathbf{w}^{t,z})^2}{2}}\left[e^{-\frac{(y-\mathbf{x}^\top \mathbf{w}^{t,0})^2}{2}} + e^{-\frac{(y-\mathbf{x}^\top \mathbf{w}^{t,1})^2}{2}}\right]^{-1}$. Note that $\alpha^z_{t,(\mathbf{x},y)} \geq 0$ for $z = 0, 1$ and $\alpha^0_{t,(\mathbf{x},y)} + \alpha^1_{t,(\mathbf{x},y)} = 1$. Also note that $\alpha^z_{t,(\mathbf{x},y)}$ is larger if $\mathbf{w}^{t,z}$ gives less regression error for the point $(\mathbf{x}, y)$ than $\mathbf{w}^{t,1-z}$ i.e., if $(y - \mathbf{x}^\top \mathbf{w}^{t,z})^2 < (y - \mathbf{x}^\top \mathbf{w}^{t,1-z})^2$. Thus, the data point $(\mathbf{x}, y)$ feels greater affinity to the model that fits it better, which is intuitively, an appropriate thing to do.

M-step Construction The maximizer of the Q-function has a closed form solution in this case as well. If $\mathbf{W}^{t+1} = (\mathbf{w}^{t+1,0}, \mathbf{w}^{t+1,1}) = \arg\max_{\mathbf{W}} Q(\mathbf{W}|\mathbf{W}^t)$, where $Q(\mathbf{W}|\mathbf{W}^t)$ is the sample Q-function created from a data sample $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, then it is easy to see that the M-step update is given by the solution to two weighted

[6] See Exercise 5.4.

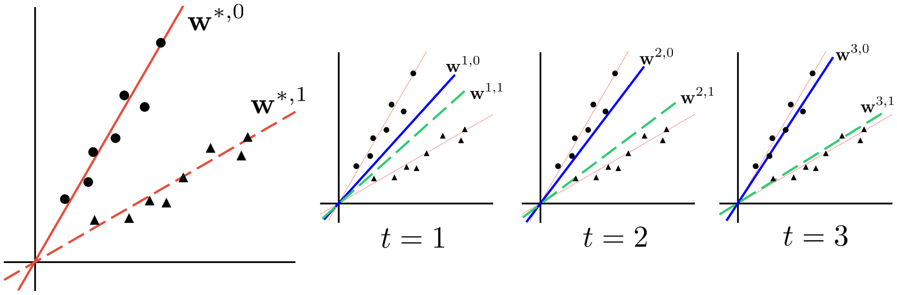


Figure 5.2: The data contains two sub-populations (circle and triangle points) and cannot be properly explained by a single linear model. EM rapidly realizes that the circle points should belong to the solid model and the triangle points to the dashed model. Thus, α_t^0 keeps going up for circle points and α_t^1 keeps going up for triangle points. As a result, only circle points contribute significantly to learning the solid model and only triangle points contribute significantly to the dashed model.

least squares problems with weights given by $\alpha^z_{t,(\mathbf{x}_i,y_i)}$ for $z = \{0, 1\}$, which have closed form solutions given by

$$\mathbf{w}^{t+1,z} = \left(\sum_{i=1}^{n} \alpha^z_{t,(\mathbf{x}_i,y_i)} \cdot \mathbf{x}_i \mathbf{x}_i^\top \right)^{-1} \sum_{i=1}^{n} \alpha^z_{t,(\mathbf{x}_i,y_i)} \cdot y_i \mathbf{x}_i$$

Note that to update $\mathbf{w}^{t,0}$, for example, the M-step essentially performs least squares only over those points $(\mathbf{x}_i, y_i)$ such that $\alpha^0_{t,(\mathbf{x},y)}$ is large and ignores points where $\alpha^0_{t,(\mathbf{x},y)} \approx 0$. This is akin to identifying points that belong to sub-population 0 strongly and performing regression only over them. One need not construct the Q-function explicitly here either and may simply keep repeating the M-step. Figure 5.2 depicts the working of the EM algorithm on a toy mixed regression problem.

Note on Initialization: when executing the EM algorithm, it is important to initialize the models properly. As we saw in § 4, this is true of all alternating strategies. Careless initialization can lead to poor results. For example, if the EM algorithm is initialized for the mixed regression problem with $\mathbf{W}^1 = (\mathbf{w}^{1,0}, \mathbf{w}^{1,1})$ such that $\mathbf{w}^{1,0} = \mathbf{w}^{1,1}$ then it is easy to see that the EM algorithm will never learn two distinct models and we will have $\mathbf{w}^{t,0} = \mathbf{w}^{t,1}$ for all t. We will revisit initialization issues shortly when discussing convergence analyses for the EM.

We refer the reader to [Balakrishnan et al., 2017, Yang et al., 2015] for examples of the EM algorithm applied to other problems such as learning hidden Markov models and regression with missing covariates.

5.7 A Monotonicity Guarantee for EM

We now present a simple but useful monotonicity property of the EM algorithm that guarantees that the procedure never worsens the likelihood of the data during its successive updates. This assures us that EM does not diverge, although it does not ensure convergence to the true parameter $\boldsymbol{\theta}^*$ either. For gAM, a similar result was immediate from the nature of the alternating updates.

Theorem 5.1. The EM algorithm (Algorithm 5), when executed with a population E-step and fully corrective M-step, ensures that the population likelihood never decreases across iterations, i.e., for all t,

$$\mathbb{E}_{\mathbf{y}\sim f_{\boldsymbol{\theta}^*}} f(\mathbf{y}\,|\,\boldsymbol{\theta}^{t+1}) \geq \mathbb{E}_{\mathbf{y}\sim f_{\boldsymbol{\theta}^*}} f(\mathbf{y}\,|\,\boldsymbol{\theta}^{t}),$$

If executed with the sample E-step on data $\mathbf{y}_1, \ldots, \mathbf{y}_n$, EM ensures that the sample likelihood never decreases across iterations, i.e., for all t,

$$\mathcal{L}(\boldsymbol{\theta}^{t+1}; \mathbf{y}_1, \ldots, \mathbf{y}_n) \geq \mathcal{L}(\boldsymbol{\theta}^{t}; \mathbf{y}_1, \ldots, \mathbf{y}_n).$$

A similar result holds for gradient M-steps too but we do not consider that here. To prove this result, we will need the following simple observation[7]. Recall that for any $\boldsymbol{\theta}^0 \in \Theta$ and $\mathbf{y} \in \mathcal{Y}$, we defined the terms $Q_{\mathbf{y}}(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}^0) = \underset{\mathbf{z}\sim f(\cdot\,|\,\mathbf{y},\boldsymbol{\theta}^0)}{\mathbb{E}} [\log f(\mathbf{y}, \mathbf{z}\,|\,\boldsymbol{\theta})]$ and $R_{\mathbf{y}}(\boldsymbol{\theta}^0) = \mathbb{E}_{\mathbf{z}\sim f(\cdot\,|\,\mathbf{y},\boldsymbol{\theta}^0)}\left[\log f(\mathbf{z}\,|\,\mathbf{y}, \boldsymbol{\theta}^0)\right]$.

Lemma 5.2. For any $\boldsymbol{\theta}^0 \in \Theta$, we have

$$\log f(\mathbf{y}\,|\,\boldsymbol{\theta}^0) = Q_{\mathbf{y}}(\boldsymbol{\theta}^0\,|\,\boldsymbol{\theta}^0) - R_{\mathbf{y}}(\boldsymbol{\theta}^0)$$

This is a curious result since we have earlier seen that for any $\boldsymbol{\theta} \in \Theta$, we have the inequality $\log f(\mathbf{y}\,|\,\boldsymbol{\theta}) \geq Q_{\mathbf{y}}(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}^0) - R_{\mathbf{y}}(\boldsymbol{\theta}^0)$ by an application of the Jensen's inequality (see § 5.4). The above lemma suggests that the inequality is actually tight. We now prove Theorem 5.1.

[7] See Exercise 5.5.

Proof (of Theorem 5.1). Consider the sample E-step with $Q(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}^t) = \frac{1}{n}\sum_{i=1}^n Q_{\mathbf{y}_i}(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}^t)$. A similar argument works for population E-steps. The fully corrective M-step ensures $Q(\boldsymbol{\theta}^{t+1}\,|\,\boldsymbol{\theta}^t) \geq Q(\boldsymbol{\theta}^t\,|\,\boldsymbol{\theta}^t)$ i.e.,

$$\frac{1}{n}\sum_{i=1}^n Q_{\mathbf{y}_i}(\boldsymbol{\theta}^{t+1}\,|\,\boldsymbol{\theta}^t) \geq \frac{1}{n}\sum_{i=1}^n Q_{\mathbf{y}_i}(\boldsymbol{\theta}^t\,|\,\boldsymbol{\theta}^t).$$

Subtracting the same terms from both sides gives us

$$\frac{1}{n}\sum_{i=1}^n \left(Q_{\mathbf{y}_i}(\boldsymbol{\theta}^{t+1}\,|\,\boldsymbol{\theta}^t) - R_{\mathbf{y}_i}(\boldsymbol{\theta}^t)\right) \geq \frac{1}{n}\sum_{i=1}^n \left(Q_{\mathbf{y}_i}(\boldsymbol{\theta}^t\,|\,\boldsymbol{\theta}^t) - R_{\mathbf{y}_i}(\boldsymbol{\theta}^t)\right).$$

Using the inequality $\log f(\mathbf{y}\,|\,\boldsymbol{\theta}^{t+1}) \geq Q_{\mathbf{y}}(\boldsymbol{\theta}^{t+1}\,|\,\boldsymbol{\theta}^0) - R_{\mathbf{y}}(\boldsymbol{\theta}^0)$ and applying Lemma 5.2 gives us

$$\frac{1}{n}\sum_{i=1}^n \log f(\mathbf{y}_i\,|\,\boldsymbol{\theta}^{t+1}) \geq \frac{1}{n}\sum_{i=1}^n \log f(\mathbf{y}_i\,|\,\boldsymbol{\theta}^t),$$

which proves the result. □

5.8 Local Strong Concavity and Local Strong Smoothness

In order to prove stronger convergence guarantees for the EM algorithm, we need to introduce a few structural properties of parametric distributions. Let us recall the EM algorithm with a population E-step and fully corrective M-step for sake of simplicity.

1. (E-step) $Q(\cdot\,|\,\boldsymbol{\theta}^t) = \mathbb{E}_{\mathbf{y}\sim f_{\boldsymbol{\theta}^*}} Q_{\mathbf{y}}(\cdot\,|\,\boldsymbol{\theta}^t)$
2. (M-step) $\boldsymbol{\theta}^{t+1} = \underset{\boldsymbol{\theta}\in\Theta}{\arg\max}\ Q_t(\boldsymbol{\theta}\,|\,\boldsymbol{\theta}^t)$

For any $\boldsymbol{\theta}^0 \in \Theta$, we will use $q_{\boldsymbol{\theta}^0}(\cdot) = Q(\cdot\,|\,\boldsymbol{\theta}^0)$ as a shorthand for the Q-function with respect to $\boldsymbol{\theta}^0$ (constructed at the population or sample level depending on the E-step) and let $M(\boldsymbol{\theta}^0) := \underset{\boldsymbol{\theta}\in\Theta}{\arg\max}\ q_{\boldsymbol{\theta}^0}(\boldsymbol{\theta})$ denote the output of the M-step if $\boldsymbol{\theta}^0$ is the current parameter. Let $\boldsymbol{\theta}^*$ denote a parameter that optimizes the population likelihood

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta}\in\Theta}{\arg\max} \underset{\mathbf{y}\sim f(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbb{E}}[\mathcal{L}(\boldsymbol{\theta};\mathbf{y})] = \underset{\boldsymbol{\theta}\in\Theta}{\arg\max} \underset{\mathbf{y}\sim f(\mathbf{y}|\boldsymbol{\theta}^*)}{\mathbb{E}}[f(\mathbf{y}\,|\,\boldsymbol{\theta})]$$

Recall that our overall goal is indeed to recover a parameter such as $\boldsymbol{\theta}^*$ that maximizes the population level likelihood (or else sample likelihood if using sample E-steps). Now the Q-function satisfies[8] the following *self-consistency* property.

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta}\in\Theta}{\arg\max}\; q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta})$$

Thus, if we could somehow get hold of the Q-function $q_{\boldsymbol{\theta}^*}(\cdot)$, then a single M-step would solve the problem! However, this is a circular argument since getting hold of $q_{\boldsymbol{\theta}^*}(\cdot)$ would require finding $\boldsymbol{\theta}^*$ first.

To proceed along the previous argument, we need to refine this observation. Not only should the M-step refuse to deviate from the optimum $\boldsymbol{\theta}^*$ if initialized there, it should behave in relatively calm manner in the neighborhood of the optimum as well. The following properties characterize "nice" Q functions that ensure this happens.

For sake of simplicity, we will assume that all Q-functions are continuously differentiable, as well as that the estimation problem is unconstrained i.e., $\Theta = \mathbb{R}^p$. Note that this ensures $\nabla q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^*) = \mathbf{0}$ due to self-consistency. Also note that since we are looking at maximization problems, we will require the Q function to satisfy "concavity" properties instead of "convexity" properties.

Definition 5.1 (Local Strong Concavity)**.** A statistical estimation problem with a population likelihood maximizer $\boldsymbol{\theta}^*$ satisfies the (r, α)-Local Strong Concavity (LSC) property if there exist $\alpha, r > 0$, such that the function $q_{\boldsymbol{\theta}^*}(\cdot)$ is α-strongly concave in neighborhood ball of radius r around $\boldsymbol{\theta}^*$ i.e., for all $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2 \in \mathcal{B}_2(\boldsymbol{\theta}^*, r)$,

$$q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^1) - q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^2) - \left\langle \nabla q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^2), \boldsymbol{\theta}^1 - \boldsymbol{\theta}^2 \right\rangle \leq -\frac{\alpha}{2} \left\| \boldsymbol{\theta}^1 - \boldsymbol{\theta}^2 \right\|_2^2$$

The reader would find LSC similar to restricted strong convexity (RSC) in Definition 3.2, with the "restriction" being the neighborhood $\mathcal{B}_2(\boldsymbol{\theta}^*, r)$ of $\boldsymbol{\theta}^*$. Also note that only $q_{\boldsymbol{\theta}^*}(\cdot)$ is required to satisfy the LSC property, and not Q-functions corresponding to every $\boldsymbol{\theta}$.

We will also require a counterpart to restricted strong smoothness (RSS). For that, we introduce the notion of *Lipschitz* gradients.

[8] See Exercise 5.6.

Definition 5.2 (Lipschitz Gradients)**.** A differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ is said to have β-Lipschitz gradients if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, we have

$$\left\| \nabla f(\mathbf{x}) - \nabla f(\mathbf{y}) \right\|_2 \leq \beta \cdot \left\| \mathbf{x} - \mathbf{y} \right\|_2 .$$

We advise the reader to relate this notion to that of Lipschitz functions (Definition 2.5). It can be shown[9] that all functions with L-Lipschitz gradients are also L-strongly smooth (Definition 2.4). Using this notion we are now ready to introduce our next properties.

Definition 5.3 (Local Strong Smoothness)**.** A statistical estimation problem with a population likelihood maximizer $\boldsymbol{\theta}^*$ satisfies the (r, β)-Local Strong Smoothness (LSS) property if there exist $\beta, r > 0$, such that for all $\boldsymbol{\theta}^1, \boldsymbol{\theta}^2 \in \mathcal{B}_2(\boldsymbol{\theta}^*, r)$, the function $q_{\boldsymbol{\theta}^*}(\cdot)$ satisfies

$$\left\| \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}^1)) - \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}^2)) \right\|_2 \leq \beta \cdot \left\| \boldsymbol{\theta}^1 - \boldsymbol{\theta}^2 \right\|_2$$

The above property ensures that in the restricted neighborhood around the optimum, the Q-function $q_{\boldsymbol{\theta}^*}(\cdot)$ is strongly smooth. The similarity to RSS is immediate. Note that this property also generalizes the self-consistency property we saw a moment ago.

Self-consistency forces $\nabla q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^*) = \mathbf{0}$ at the optimum. LSS forces such behavior to extend around the optimum as well. To see this, simply set $\boldsymbol{\theta}^2 = \boldsymbol{\theta}^*$ with LSS and observe the corollary $\left\| \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}^1)) \right\|_2 \leq \beta \cdot \left\| \boldsymbol{\theta}^1 - \boldsymbol{\theta}^* \right\|_2$. The curious reader may wish to relate this corollary to the Robust Bistability property (Definition 4.6) and the *Local Error Bound* property introduced by Luo and Tseng [1993].

The LSS property offers, as another corollary, a useful property of statistical estimation problems called the *First Order Stability* property (introduced by Balakrishnan et al. [2017] in a more general setting).

Definition 5.4 (First Order Stability [Balakrishnan et al., 2017])**.** A statistical estimation problem with a population likelihood maximizer $\boldsymbol{\theta}^*$ satisfies the (r, γ)-First Order Stability (FOS) property if there exist $\gamma > 0, r > 0$ such that the the gradients of the functions $q_{\boldsymbol{\theta}}(\cdot)$ are stable in a neighborhood of $\boldsymbol{\theta}^*$ i.e., for all $\boldsymbol{\theta} \in \mathcal{B}_2(\boldsymbol{\theta}^*, r)$,

$$\left\| \nabla q_{\boldsymbol{\theta}}(M(\boldsymbol{\theta})) - \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})) \right\|_2 \leq \gamma \cdot \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^* \right\|_2$$

[9] See Exercise 5.7.

Lemma 5.3. A statistical estimation problem that satisfies the (r, β)-LSS property, also satisfies the (r, β)-FOS property.

Proof. Since $M(\boldsymbol{\theta})$ maximizes the function $q_{\boldsymbol{\theta}}(\cdot)$ due to the M-step, and the problem is unconstrained and the Q-functions differentiable, we get $\nabla q_{\boldsymbol{\theta}}(M(\boldsymbol{\theta})) = \mathbf{0}$. Thus, we have, using the triangle inequality,

$$\begin{aligned}
&\left\|\nabla q_{\boldsymbol{\theta}}(M(\boldsymbol{\theta})) - \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}))\right\|_2 = \left\|\nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}))\right\|_2 \\
&\leq \left\|\nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})) - \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}^*))\right\|_2 + \left\|\nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta}^*))\right\|_2 \\
&\leq \beta \cdot \left\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\right\|_2,
\end{aligned}$$

using self-consistency to get $M(\boldsymbol{\theta}^*) = \boldsymbol{\theta}^*$ and $\nabla q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^*) = \mathbf{0}$. □

5.9 A Local Convergence Guarantee for EM

We will now prove a convergence result much stronger than Theorem 5.1: if EM is initialized "close" to the optimal parameter $\boldsymbol{\theta}^*$ for "nice" problems, then it approaches $\boldsymbol{\theta}^*$ at a linear rate. For the sake of simplicity, we will analyze EM with population E-steps and fully corrective M-steps. We refer the reader to [Balakrishnan et al., 2017] for analyses of sample E-steps and stochastic M-steps. The following *contraction* lemma will be crucial in the analysis of the EM algorithm.

Lemma 5.4. Suppose we have a statistical estimation problem with a population likelihood maximizer $\boldsymbol{\theta}^*$ that, for some $\alpha, \beta, r > 0$, satisfies the (r, α)-LSC and (r, β)-LSS properties. Then in the region $\mathcal{B}_2(\boldsymbol{\theta}^*, r)$, the M operator corresponding to the fully corrective M-step is contractive, i.e., for all $\boldsymbol{\theta} \in \mathcal{B}_2(\boldsymbol{\theta}^*, r)$,

$$\left\|M(\boldsymbol{\theta}) - M(\boldsymbol{\theta}^*)\right\|_2 \leq \frac{\beta}{\alpha} \cdot \left\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\right\|_2$$

Since by the self consistency property, we have $M(\boldsymbol{\theta}^*) = \boldsymbol{\theta}^*$ and the EM algorithm sets $\boldsymbol{\theta}^{t+1} = M(\boldsymbol{\theta}^t)$ due to the M-step, Lemma 5.4 immediately guarantees the following local convergence property[10].

Theorem 5.5. Suppose a statistical estimation problem with population likelihood maximizer $\boldsymbol{\theta}^*$ satisfies the (r, α)-LSC and (r, β)-LSS

[10] See Exercise 5.8.

properties such that $\beta < \alpha$. Let the EM algorithm (Algorithm 5) be initialized with $\boldsymbol{\theta}^1 \in \mathcal{B}_2(\boldsymbol{\theta}^*, r)$ and executed with population E-steps and fully corrective M-steps. Then after at most $T = \mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ steps, we have $\left\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^*\right\|_2 \leq \epsilon$.

Note that the above result holds only if $\beta < \alpha$, in other words, if the condition number $\kappa = \beta/\alpha < 1$. We hasten to warn the reader that whereas in previous sections we always had $\kappa \geq 1$, here the LSC and LSS properties are defined differently (LSS involves the M-step whereas LSC does not) and thus it is possible that we have $\kappa < 1$.

Also, since all functions satisfy $(0, 0)$-LSC, it is plausible that for well behaved problems, even for $\alpha > \beta$, there should exist some small radius $r(\alpha)$ so that the $(r(\alpha), \alpha)$-LSC property holds. This may require the EM algorithm to be initialized closer to the optimum for the convergence properties to kick in. We now prove Lemma 5.4 below.

Proof of Lemma 5.4. Since we have differentiable Q-functions and an unconstrained estimation problem, we immediately get a lot of useful results. We note that this lemma holds even for constrained estimation problems but the arguments are more involved which we wish to avoid. Let $\boldsymbol{\theta} \in \mathcal{B}_2(\boldsymbol{\theta}^*, r)$ be any parameter in the r-neighborhood of $\boldsymbol{\theta}^*$.

(Apply Local Strong Concavity) Upon a two sided application of LSC and using $\nabla q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^*) = \mathbf{0}$, we get

$$q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^*) - q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})) - \langle \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})), \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \rangle \leq -\frac{\alpha}{2} \left\| M(\boldsymbol{\theta}) - \boldsymbol{\theta}^* \right\|_2^2$$
$$q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})) - q_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}^*) \leq -\frac{\alpha}{2} \left\| \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \right\|_2^2,$$

adding which gives us the inequality

$$\langle \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})), \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \rangle \geq \alpha \cdot \left\| M(\boldsymbol{\theta}) - \boldsymbol{\theta}^* \right\|_2^2$$

(Apply Local Strong Smoothness) Since $M(\boldsymbol{\theta})$ maximizes the function $q_{\boldsymbol{\theta}}(\cdot)$ due to the M-step, we get $\nabla q_{\boldsymbol{\theta}}(M(\boldsymbol{\theta})) = \mathbf{0}$. Thus,

$$\begin{aligned} \langle \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})), \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \rangle &= \langle \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})) - \nabla q_{\boldsymbol{\theta}}(M(\boldsymbol{\theta})), \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \rangle \\ &\leq \left\| \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})) - \nabla q_{\boldsymbol{\theta}}(M(\boldsymbol{\theta})) \right\|_2 \left\| \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \right\|_2 \end{aligned}$$

Using Lemma 5.3 to invoke the (r, β)-FOS property further gives us

$$\langle \nabla q_{\boldsymbol{\theta}^*}(M(\boldsymbol{\theta})), \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \rangle \leq \beta \cdot \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^* \right\|_2 \left\| \boldsymbol{\theta}^* - M(\boldsymbol{\theta}) \right\|_2$$

Combining the two results proved above gives us

$$\alpha \cdot \left\| M(\boldsymbol{\theta}) - \boldsymbol{\theta}^* \right\|_2^2 \leq \beta \cdot \left\| M(\boldsymbol{\theta}) - \boldsymbol{\theta}^* \right\|_2 \cdot \left\| \boldsymbol{\theta} - \boldsymbol{\theta}^* \right\|_2,$$

which finish the proof. □

5.9.1 A Note on the Application of Convergence Guarantees

We conclude the discussion with some comments on the feasibility of the structural assumptions we used to prove the convergence guarantees, in practical settings. Rigorous proofs that these properties are indeed satisfied in practical applications is beyond the scope of this monograph. These can be found in [Balakrishnan et al., 2017]. Note that for the convergence guarantees (specifically Lemma 5.4) to hold, a problem need only satisfy the LSC and FOS properties, with LSS being equivalent to FOS due to Lemma 5.3.

Gaussian Mixture Models To analyze the LSC and FOS properties, we need to look at the population version of the Q-function. Given the point-wise Q-function derivation of the in § 5.6, we get for $\mathbf{M} = (\boldsymbol{\mu}^0, \boldsymbol{\mu}^1)$

$$Q(\mathbf{M} \,|\, \mathbf{M}^*) = -\frac{1}{2}\mathbb{E}_{\mathbf{y} \sim f_{\mathbf{M}^*}} \left[w^0(\mathbf{y}) \cdot \left\| \mathbf{y} - \boldsymbol{\mu}^0 \right\|_2^2 + w^1(\mathbf{y}) \cdot \left\| \mathbf{y} - \boldsymbol{\mu}^1 \right\|_2^2 \right],$$

where $w^z(\mathbf{y}) = e^{-\frac{\|\mathbf{y}-\boldsymbol{\mu}^{*,z}\|_2^2}{2}} \left[e^{-\frac{\left\|\mathbf{y}-\boldsymbol{\mu}^{*,0}\right\|_2^2}{2}} + e^{-\frac{\left\|\mathbf{y}-\boldsymbol{\mu}^{*,1}\right\|_2^2}{2}} \right]^{-1}$ for $z = 0, 1$.

It can be seen that the function $q_{\mathbf{M}^*}(\cdot)$ satisfies $\nabla^2 q_{\mathbf{M}^*}(\cdot) \succeq w \cdot I$ where $w = \min\left\{ \mathbb{E}\left[w^0(\mathbf{y})\right], \mathbb{E}\left[w^1(\mathbf{y})\right] \right\} > 0$ and hence this problem satisfies the (∞, w)-LSC property i.e., it is globally strongly concave.

Establishing the FOS property is more involved. However, it can be shown that the problem does satisfy the (r, α)-FOS property with $r = \Omega(\|\mathbf{M}^*\|_2)$ and $\alpha = \exp(-\Omega\left(\|\mathbf{M}^*\|_2^2\right))$ under suitable conditions.

Mixed Regression We again use the point-wise Q-function construction to construct the population Q-function. For any $\mathbf{W} = (\mathbf{w}^0, \mathbf{w}^1)$,

$$Q_{(\mathbf{x},y)}(\mathbf{W} | \mathbf{W}^*) = -\frac{1}{2}\mathbb{E}\left[\alpha^0_{(\mathbf{x},y)} \cdot (y - \mathbf{x}^\top \mathbf{w}^0)^2 - \alpha^1_{(\mathbf{x},y)} \cdot (y - \mathbf{x}^\top \mathbf{w}^1)^2\right],$$

where $\alpha^z_{(\mathbf{x},y)} = e^{-\frac{(y-\mathbf{x}^\top \mathbf{w}^{*,z})^2}{2}}\left[e^{-\frac{(y-\mathbf{x}^\top \mathbf{w}^{*,0})^2}{2}} + e^{-\frac{(y-\mathbf{x}^\top \mathbf{w}^{*,1})^2}{2}}\right]^{-1}$. Assuming $\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right] = I$ for sake of simplicity, we get $\nabla^2 q_{\mathbf{W}^*}(\cdot) \succeq \alpha \cdot I$ where $\alpha = \min\left\{\lambda_{\min}\left(\mathbb{E}\left[w^0(\mathbf{x}, y) \cdot \mathbf{x}\mathbf{x}^\top\right]\right), \lambda_{\min}\left(\mathbb{E}\left[w^1(\mathbf{x}, y) \cdot \mathbf{x}\mathbf{x}^\top\right]\right)\right\}$ i.e., the problem satisfies the (∞, w)-LSC property i.e., it is globally strongly concave. Establishing the FOS property is more involved but the problem does satisfy the (r, α)-FOS property with $r = \Omega(\|\mathbf{W}^*\|_2)$ and $\alpha = \Omega(1)$ under suitable conditions.

5.10 Exercises

Exercise 5.1. Show that for Gaussian mixture models with a balanced isotropic mixture, the AM-LVM algorithm (Algorithm 4) implements exactly recovers Lloyd's algorithm for k-means clustering. Note that AM-LVM in this case prescribes setting $w^0_t(\mathbf{y}) = 1$ if $\|\mathbf{y} - \boldsymbol{\mu}^{t,0}\|_2 \leq \|\mathbf{y} - \boldsymbol{\mu}^{t,1}\|_2$ and 0 otherwise and also setting $w^1_t(\mathbf{y}) = 1 - w^0_t(\mathbf{y})$.

Exercise 5.2. Show that on expectation, the stochastic EM update rule is, on expectation, equivalent to the sample E and the gradient M-step i.e., $\mathbb{E}\left[M^{\text{sto}}(\boldsymbol{\theta}^t, Q_t) \,|\, \boldsymbol{\theta}^t\right] = M^{\text{grad}}(\boldsymbol{\theta}^t, Q^{\text{sam}}_t(\cdot \,|\, \boldsymbol{\theta}^t))$.

Exercise 5.3. Derive the fully corrective and gradient M-step in the Gaussian mixture modeling problem. Show that they have closed forms.

Exercise 5.4. Derive the E and M-step constructions for the mixed regression problem with fair Bernoulli trials.

Exercise 5.5. Prove Lemma 5.2.

Exercise 5.6. Let $\widehat{\boldsymbol{\theta}}$ be a population likelihood maximizer i.e.,

$$\widehat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta} \in \Theta} \mathop{\mathbb{E}}_{\mathbf{y} \sim f(\mathbf{y} | \boldsymbol{\theta}^*)} \left[f(\mathbf{y} \,|\, \boldsymbol{\theta})\right]$$

Then show that $\widehat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}\in\Theta} q_{\widehat{\boldsymbol{\theta}}}(\boldsymbol{\theta})$. *Hint*: One way to show this result is to use Theorem 5.1 and Lemma 5.2.

Exercise 5.7. Show that a function f (whether convex or not) that has L-Lipschitz gradients is necessarily L-strongly smooth. Also show that for any given $L > 0$, there exist functions that do not have L-Lipschitz gradients which are also not L-strongly smooth.
Hint: Use the fundamental theorem for calculus for line integrals for the first part. For the second part try using a quadratic function.

Exercise 5.8. For any statistical estimation problem with population likelihood maximizer $\boldsymbol{\theta}^*$ that satisfies the LSC and LSS properties with appropriate constants, show that parameters close to $\boldsymbol{\theta}^*$ are approximate population likelihood maximizers themselves. Show this by finding constants $\epsilon_0, D > 0$ (that may depend on the LSC, LSS constants) such that for any $\boldsymbol{\theta}$, if $\left\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\right\|_2 \leq \epsilon < \epsilon_0$, then

$$\mathop{\mathbb{E}}_{\mathbf{y}\sim f(\mathbf{y}\,|\,\boldsymbol{\theta}^*)}\left[f(\mathbf{y}\,|\,\boldsymbol{\theta})\right] \geq \mathop{\mathbb{E}}_{\mathbf{y}\sim f(\mathbf{y}\,|\,\boldsymbol{\theta}^*)}\left[f(\mathbf{y}\,|\,\boldsymbol{\theta}^*)\right] - D\cdot\epsilon.$$

Hint: Use Lemma 5.2 and Exercise 5.7.

Exercise 5.9. Similar to how the EM algorithm ensures monotone progress with respect to the likelihood objective (see § 5.7), show that the the AM-LVM algorithm (see Algorithm 4) ensures monotone progress with respect to the following optimization problem

$$\max_{\substack{\boldsymbol{\theta}\in\Theta \\ \widehat{\mathbf{z}}_i,\ldots,\widehat{\mathbf{z}}_n\in\mathcal{Z}}} \mathcal{L}\left(\boldsymbol{\theta}; \{(\mathbf{y}_i, \widehat{\mathbf{z}}_i)\}_{i=1}^n\right).$$

More specifically, show that if the iterates obtained by AM-LVM at time t are $\boldsymbol{\theta}^t, \{\widehat{\mathbf{z}}_i^t\}_{i=1}^n$, then we have for all $t \geq 1$

$$\mathcal{L}\left(\boldsymbol{\theta}^{t+1}; \left\{(\mathbf{y}_i, \widehat{\mathbf{z}}_i^{t+1})\right\}_{i=1}^n\right) \geq \mathcal{L}\left(\boldsymbol{\theta}^{t}; \left\{(\mathbf{y}_i, \widehat{\mathbf{z}}_i^{t})\right\}_{i=1}^n\right)$$

5.11 Bibliographic Notes

The EM algorithm was given its name and formally introduced in the seminal work of Dempster et al. [1977]. Although several results exist that attempt to characterize the convergence properties of the EM algorithm, prominent among them is the work of Wu [1983] which showed that for problems with a uni-modal likelihood function and some regularity conditions that make the estimation problem well-posed, the EM algorithm converges to the (unique) global optimum.

We have largely followed the work of Balakrishnan et al. [2017] in the treatment of the topic in this section. Recent results have attempted to give local convergence results for the EM algorithm, similar to the ones we discussed. Examples include analyses of the Baum-Welch algorithm used to learn hidden Markov models [Yang et al., 2015], the EM algorithm [Balakrishnan et al., 2017], and the more general problem of M-estimation [Andresen and Spokoiny, 2016] (recall that maximum likelihood estimators are members of the broader class of M-estimators). The recurring message here is that upon proper initialization, the EM algorithm does converge to the global optimum.

For certain problems, such as mixed regression, it is known how to get such initializations in polynomial time under regularity assumptions [Yi et al., 2014] which makes the entire procedure a polynomial time solution to get globally consistent parameters. Other recent advances in the area include extensions to high dimensional estimation estimation problems by way of regularization [Yi and Caramanis, 2015] or directly incorporating sparsity structure into the procedure [Wang et al., 2015]. We refer the reader to the work of Balakrishnan et al. [2017] and other more recent works for a more detailed review of literature in this area.

As a closing note, we reiterate our previous comment on similarities between EM and alternating minimization. Both have an alternating pattern in their execution with each alternation typically being cheap and easy to execute. Indeed, for several areas in which the EM approach has been found successful, such as learning mixture models, regression with missing or corrupted data, phase retrieval etc, there also exist efficient alternating minimization algorithms. We will look at some of them in later sections. Having said that, the EM algorithm does distinguish itself within the general alternating minimization approach in its single minded goal of approaching the MLE by promoting parameters that increase the likelihood of the data.

6

Stochastic Optimization Techniques

In previous sections, we have looked at specific instances of optimization problems with non-convex objective functions. In § 4 we looked at problems in which the objective can be expressed as a function of two variables, whereas in § 5 we looked at objective functions with latent variables that arose in probabilistic settings. In this section, we will look at the problem of optimization with non-convex objectives in a more general setting.

Several machine learning and signal processing applications such as deep learning, topic modeling etc, generate optimization problems that have non-convex objective functions. The global optimization of non-convex objectives, i.e., finding the global optimum of the objective function, is an NP hard problem in general. Even the seemingly simple problem of minimizing quadratic functions of the kind $\mathbf{x}^\top A\mathbf{x}$ over convex constraint sets becomes NP-hard the moment the matrix A is allowed to have even one negative eigenvalue.

As a result, a much sought after goal in applications with non-convex objectives is to find a local minimum of the objective function. The main hurdle in achieving local optimality is the presence of saddle points which can mislead optimization methods such as gradient descent by stalling their progress. Saddle points are best avoided as they signal inflection in the objective surface and unlike local optima, they need not optimize the objective function in any meaningful way.

The recent years have seen much interest in this problem, particularly with the advent of deep learning where folklore wisdom tells us that in the presence of sufficient data, even locally optimal solutions to the problem of learning the edge weights of a network perform quite well [Choromanska et al., 2015]. In these settings, techniques such as convex relaxations, and non-convex optimization techniques that we have studied such as EM, gAM, gPGD, do not apply directly. In these settings, one has to attempt optimizing a non-convex objective directly.

The problem of avoiding or escaping saddle points is actually quite challenging in itself given the wide variety of configurations saddle points can appear in, especially in high dimensional problems. It should be noted that there exist saddle configurations, bypassing which is intractable in itself. For such cases, even finding locally optimal solutions is an NP-hard problem [Anandkumar and Ge, 2016].

In our discussion, we will look recent results which show that if the function being optimized possesses certain nice structural properties, then an application of very intuitive algorithmic techniques can guarantee local optimality of the solutions. This will be yet another instance of a result where the presence of a structural property (such as RSC/RSS, MSC/MSS, or LSC/LSS as studied in previous sections) makes the problem well behaved and allows efficient algorithms to offer provably good solutions. Our discussion will largely aligned to the work of Ge et al. [2015]. The bibliographic notes will point to other works.

6.1 Motivating Applications

A wide range of problems in machine learning and signal processing generate optimization problems with non-convex objectives. Of particular interest to us would be the problem of *Orthogonal Tensor Decomposition.* This problem has been shown to be especially useful in modeling a large variety of learning problems including training deep Recurrent Neural Networks [Sedghi and Anandkumar, 2016], Topic Modeling, learning Gaussian Mixture Models and Hidden Markov Models, Independent Component Analysis [Anandkumar et al., 2014], and reinforcement learning [Azizzadenesheli et al., 2016].

The details of how these machine learning problems can be reduced to tensor decomposition problems will involve getting into the details which will distract us from our main objective. To keep the discussion focused and brief, we request the reader to refer to these papers for the reductions to tensor decomposition. We ourselves will be most interested in the problem of tensor decomposition itself.

We will restrict our study to 4^{th}-order tensors which can be interpreted as 4-dimensional arrays. Tensors are easily constructed using *outer products*, also known as *tensor products*. An outer product of 2^{nd} order produces a 2^{nd} order tensor which is nothing but a matrix. For any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^p$, their outer product is defined as $\mathbf{u} \otimes \mathbf{v} := \mathbf{u}\mathbf{v}^\top \in \mathbb{R}^{p \times p}$ which is a $p \times p$ matrix, whose (i, j)-th entry is $\mathbf{u}_i \mathbf{v}_j$.

We can similarly construct a 4^{th}-order tensors. For any $\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x} \in \mathbb{R}^p$, let $T = \mathbf{u} \otimes \mathbf{v} \otimes \mathbf{w} \otimes \mathbf{x} \in \mathbb{R}^{p \times p \times p \times p}$. The (i, j, k, l)-th entry of this tensor, for any $i, j, k, l \in [p]$, will be $T_{i,j,k,l} = \mathbf{u}_i \cdot \mathbf{v}_j \cdot \mathbf{w}_k \cdot \mathbf{x}_l$. The set of 4^{th}-order tensors is closed under addition and scalar multiplication. We will study a special class of 4^{th}-order tensors known as *orthonormal* tensors which have an *orthonormal decomposition* as follows

$$T = \sum_{i=1}^{r} \mathbf{u}_i \otimes \mathbf{u}_i \otimes \mathbf{u}_i \otimes \mathbf{u}_i,$$

where the vectors $\mathbf{u}_i$ are orthonormal *components* of the tensor T i.e., $\mathbf{u}_i^\top \mathbf{u}_j = 0$ if $i \neq j$ and $\|\mathbf{u}_i\|_2 = 1$. The above tensor is said to have rank r since it has r components in its decomposition. If an orthonormal decomposition of a tensor exists, it can be shown to be unique.

Just as a matrix $A \in \mathbb{R}^{p \times p}$ defines a bi-linear form $A : (\mathbf{x}, \mathbf{y}) \mapsto \mathbf{x}^\top A \mathbf{y}$, similarly a tensor defines a *multi-linear form*. For orthonormal tensors, the multilinear form has a simple expression. In particular, if T has the orthonormal form described above, we have

$$\begin{aligned}
T(\mathbf{v}, \mathbf{v}, \mathbf{v}, \mathbf{v}) &= \textstyle\sum_{i=1}^{r} (\mathbf{u}_i^\top \mathbf{v})^4 \in \mathbb{R} \\
T(I, \mathbf{v}, \mathbf{v}, \mathbf{v}) &= \textstyle\sum_{i=1}^{r} (\mathbf{u}_i^\top \mathbf{v})^3 \cdot \mathbf{u}_i \in \mathbb{R}^p \\
T(I, I, \mathbf{v}, \mathbf{v}) &= \textstyle\sum_{i=1}^{r} (\mathbf{u}_i^\top \mathbf{v})^2 \cdot \mathbf{u}_i \mathbf{u}_i^\top \in \mathbb{R}^{p \times p} \\
T(I, I, I, \mathbf{v}) &= \textstyle\sum_{i=1}^{r} (\mathbf{u}_i^\top \mathbf{v}) \cdot (\mathbf{u}_i \otimes \mathbf{u}_i \otimes \mathbf{u}_i) \in \mathbb{R}^{p \times p \times p}
\end{aligned}$$

The problem of orthonormal tensor decomposition involves recovering all r components of a rank-r, 4-th order tensor T. An intuitive way to do this is to iteratively find all the components.

First we recover, say w.l.o.g., the first component $\mathbf{u}_1$. Then we perform a *peeling/deflation* step by subtracting that component and creating a new tensor $T^{(1)} = T - \mathbf{u}_1 \otimes \mathbf{u}_1 \otimes \mathbf{u}_1 \otimes \mathbf{u}_1$ and repeating the process. Note that the rank of the tensor $T^{(1)}$ is only $r - 1$ and thus, this procedure terminates in just r steps.

To execute the above algorithm, all we are required is to solve the individual steps of recovering a single component. This requires us to solve the following optimization problem.

$$\begin{array}{rl} \max & T(\mathbf{u}, \mathbf{u}, \mathbf{u}, \mathbf{u}) = \sum_{i=1}^{r} (\mathbf{u}_i^\top \mathbf{u})^4 \\ \text{s.t.} & \|\mathbf{u}\|_2 = 1, \end{array} \tag{LRTD}$$

This is the non-convex optimization problem[1] that we will explore in more detail. We will revisit this problem later after looking at some techniques to optimize non-convex objectives.

6.2 Saddles and why they Proliferate

To better understand the challenges posed by saddle points, let us take good old gradient descent as an optimization algorithm. As we studied in § 2, the procedure involves repeatedly taking steps away from the direction of the gradient.

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta_t \cdot \nabla f(\mathbf{x}^t)$$

Now, it can be shown[23], that the procedure is guaranteed to make progress at *every* time step, provided the function f is strongly smooth and the step length is small enough. However, the procedure stalls at *stationary points* where the gradient of the function vanishes i.e., $\nabla f(\mathbf{x}) = \mathbf{0}$. This includes local optima, which are of interest, and saddle points, which simply stall descent algorithms.

[1] See Exercise 6.1.

[2] See Exercise 6.2.

[3] See Exercise 6.3.

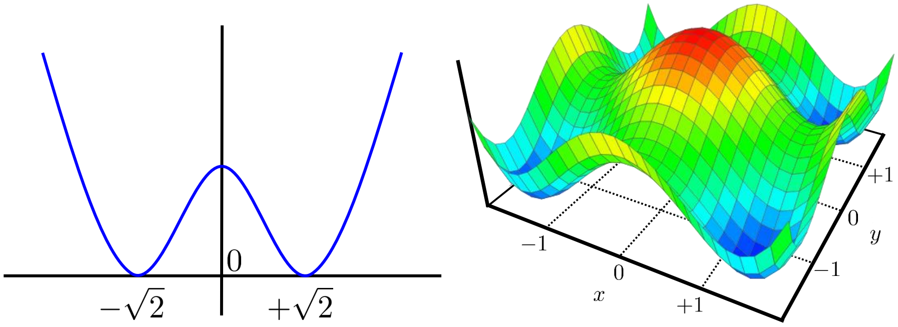


Figure 6.1: The function on the left $f(x) = x^4 - 4 \cdot x^2 + 4$ has two global optima $\left\{-\sqrt{2}, \sqrt{2}\right\}$ separated by a local maxima at 0. Using this function, we construct on the right, a higher dimensional function $g(x, y) = f(x) + f(y) + 8$ which now has 4 global minima separated by 4 saddle points. The number of such minima and saddle points can explode exponentially in learning problems with symmetry (indeed $g(x, y, z) = f(x) + f(y) + f(z) + 12$ has 8 local minima and saddle points). Plot on the right courtesy `academo.org`

One way to distinguish saddle points from local optima is by using the *second derivative test*. The Hessian of a doubly differentiable function has only positive eigenvalues at local minima and only negative ones at local maxima. Saddles on the other hand are unpredictable. The *simple saddles* which we shall study here, reveal themselves by having both positive and negative eigenvalues in the Hessian. The bibliographic notes discuss more complex saddle structures.

The reasons for the origin of saddle points is quite intriguing too. Figure 6.1 shows how saddles may emerge and their numbers increase exponentially with increasing dimensionality. Consider the tensor decomposition problem in ((LRTD)). It can be shown[4] that all the r components are optimal solutions to this problem. Thus, the problem possesses a beautiful symmetry which allows us to recover the components in any order we like. However, it is also easy to show[5] that general convex combinations of the components are not optimal solutions to this problem. Thus, we automatically obtain r isolated optima spread out in space, interspersed with saddle points.

[4] See Exercise 6.4.

[5] See Exercise 6.5.

The applications we discussed, such as Gaussian mixture models, also have such an internal symmetry – the optimum is unique only up to permutation. Indeed, it does not matter in which order do we recover the components of a mixture model, so long as we recover all of them. However, this very symmetry gives rise to saddle points [Ge et al., 2015], since taking two permutations of the optimal solution and taking a convex combination of them is in general not an optimal solution as well. This gives us, in general, an exponential number of optima, separated by (exponentially many) saddle points.

Before moving forward, we remind the reader that techniques we have studied so far for non-convex optimization, namely EM, gAM, and gPGD are far too specific to be applied to non-convex objectives in general, and to the problems we encounter with tensor decomposition in particular. We need more generic solutions for the task of local optimization of non-convex objectives.

6.3 The Strict Saddle Property

Given that the Hessian of the function is the first point of inquiry when trying to distinguish saddle points from other stationary points, it seems natural to use second order methods to escape points. Indeed, the bibliographic notes discuss several such approaches that use the Hessian, or estimates thereof, to escape saddle points. However, these are expensive and do not scale to high dimensional problems. Consequently, we would be more interested in scalable first order methods.

Given the above considerations, a natural course of action is to identify properties that an objective function should satisfy in order to allow gradient descent-style techniques to escape its saddle points. The recent works of Ge et al. [2015], Sun et al. [2015] give an intuitive answer to this question. They observe that if a saddle point $\mathbf{x}$ for a function f contains directions of steep descent, then it is possible, at least in principle, for a gradient descent procedure to discover this direction and "fall" along it. The existence of such directions makes a saddle unstable – it behaves like a local maxima along these directions and a slight perturbation is very likely to cause gradient descent to roll down

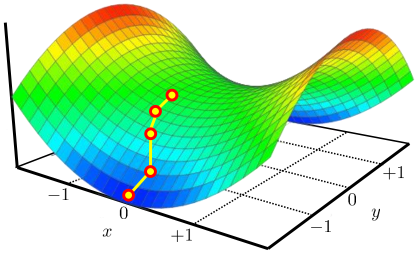


Figure 6.2: The function $f(x, y) = x^2 - y^2$ exhibits a saddle at the origin $(0, 0)$. The Hessian of this function at the origin is $-2 \cdot I$ and since $\lambda_{\min}(\nabla^2 f((0, 0))) = -2$, the saddle satisfies the strict-saddle property. Indeed, the saddle does offer a prominent descent path along the y axis which can be used to escape the saddle point. In § 6.5 we will see how the NGD algorithm is able to provably escape this saddle point. Plot courtesy `academo.org`

the function surface. We notice that this will indeed be the case if for some direction $\mathbf{u} \in \mathbb{R}^p$, we have $\mathbf{u}^\top \nabla^2 f(\mathbf{x})\mathbf{u} \ll 0$. Figure 6.2 depicts a toy case with the function $f(x, y) = x^2 - y^2$ which exhibits a saddle at the origin but nevertheless, also presents a direction of steep descent.

Consider the following unconstrained optimization problem.

$$\min_{\mathbf{x} \in \mathbb{R}^p} f(\mathbf{x}) \tag{NOPT}$$

The following *strict saddle property* formalizes the requirements we have discussed in a more robust manner.

Definition 6.1 (Strict Saddle Property [Ge et al., 2015])**.** A twice differentiable function $f(\mathbf{x})$ is said to satisfy the $(\alpha, \gamma, \kappa, \xi)$-strict saddle (SSa) property, if for every local minimum $\mathbf{x}^*$ of the function, the function is α-strongly convex in the region $\mathcal{B}_2(\mathbf{x}^*, 2\xi)$ and moreover, every point $\mathbf{x}_0 \in \mathbb{R}^p$ satisfies at least one of the following properties:

1. (Non-stationary Point) $\|\nabla f(\mathbf{x}_0)\|_2 \geq \kappa$
2. (Strict Saddle Point) $\lambda_{\min}(\nabla^2 f(\mathbf{x}_0)) \leq -\gamma$
3. (Approx. Local Min.) For some local minimum $\mathbf{x}^*$, $\|\mathbf{x}_0 - \mathbf{x}^*\|_2 \leq \xi$.

The above property places quite a few restrictions on the function. The function must be strongly-convex in the neighborhood of every local optima and every point that is not an approximate local minimum, must offer a direction of steep descent. This the point may do by having a steep gradient (case 1) or else (if the point is a saddle point) have its Hessian offer an eigenvector with a large negative eigenvalue which then offers a steep descent direction (case 2). We shall later see that there exist interesting applications that do satisfy this property.

6.4 The Noisy Gradient Descent Algorithm

Given that the strict-saddle property assures us of the existence of a steep descent direction until we reach close to an approximate-local minimum, it should come as no surprise that simple techniques should be able to achieve local optimality on functions that are strict saddle. We will now look at a one such simple approach to exploit the strict saddle property and achieve local optimality.

The idea behind the approach is very simple: at every saddle point which may stall gradient descent, the SSa property ensures that there exists a direction of steep descent. If we perturb the gradient, there is a chance it will point in the general direction of the steep descent and escape the saddle. However, if we are at a local minimum or a non-stationary point, then this perturbation would not affect us much.

Algorithm 6 gives the details of this approach. At each time step, it perturbs the gradient using a unit vector pointing at a random direction in the hope that if we are currently stuck at a saddle point, the perturbation will cause us to discover the steep (enough) descent direction, allowing us to continue making progress. One easy way to obtain a random point on the unit sphere is to sample a random standard Gaussian vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, I_{p\times p})$ and normalize it $\boldsymbol{\zeta} = \frac{1}{\|\mathbf{w}\|_2} \cdot \mathbf{w}$. Notice that since the perturbation $\boldsymbol{\zeta}^t$ is chosen uniformly from the unit sphere, we have $\mathbb{E}\left[\boldsymbol{\zeta}^t\right] = \mathbf{0}$ and by linearity of expectation, $\mathbb{E}\left[\mathbf{g}^t \,|\, \mathbf{x}^t\right] = \nabla f(\mathbf{x}^t)$.

Such an unbiased estimate $\mathbf{g}^t$ is often called a *stochastic gradient* and is widely used in machine learning and optimization. Recall that even the EM algorithm studied in § 5 had a variant that used stochastic gradients. In several machine learning applications, the objective can be written as a finite sum $f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}; \boldsymbol{\theta}^i)$ where $\boldsymbol{\theta}^i$ may denote

Algorithm 6 Noisy Gradient Descent (NGD)

```
Input: Objective $f$, max step length $\eta_{\max}$, tolerance $\epsilon$
Output: A locally optimal point $\widehat{\mathbf{x}} \in \mathbb{R}^p$
$\mathbf{x}^1 \leftarrow \mathsf{INITALIZE}()$
Set $T \leftarrow 1/\eta^2$, where $\eta = \min\left\{\epsilon^2/\log^2(1/\epsilon), \eta_{\max}\right\}$
for $t = 1, 2, \ldots, T$ do
    Sample perturbation $\boldsymbol{\zeta}^t \sim S^{p-1}$ //Random pt. on unit sphere
    $\mathbf{g}^t \leftarrow \nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t$
    $x^{t+1} \leftarrow \mathbf{x}^t - \eta \cdot \mathbf{g}^t$
end for
return $\mathbf{x}^T$
```

the i-th data point. This allows us to construct a stochastic gradient estimate in an even more inexpensive manner. At each time step, simply sample a data point $I_t \sim \mathsf{Unif}([n])$ and let

$$\mathbf{g}^t = \nabla f(\mathbf{x}^t, \boldsymbol{\theta}^{I_t}) + \boldsymbol{\zeta}^t$$

Note that we still have $\mathbb{E}\left[\mathbf{g}^t \,|\, \mathbf{x}^t\right] = \nabla f(\mathbf{x}^t)$ but with a much cheaper construction for $\mathbf{g}^t$. However, in order to simplify the discussion, we will continue to work with the setting $\mathbf{g}^t = \nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t$.

We note that we have set the step lengths to be around $1/\sqrt{T}$, where T is the total number of iterations we are going to execute the NGD algorithm. This will seem similar to what we used in the projected gradient descent approach (see Algorithm 1 and Theorem 2.5). Although in practice one may set the step length to $\eta_t \approx 1/\sqrt{t}$ here as well, the analysis becomes more involved.

6.5 A Local Convergence Guarantee for NGD

We will now analyze the NGD algorithm for its convergence properties. We note that the complete proof requires results that are quite technical and beyond the scope of this monograph. We will instead, present the essential elements of the proof and point the curious reader to [Ge et al., 2015] for the complete analysis. To simplify the notation, we will often omit specifying the exact constants as well.

In the following, we will assume that the function f satisfies the strict saddle property and is β-strongly smooth (see Definition 2.4). We will also assume that the function is bounded i.e., $|f(\mathbf{x})| \leq B$ for all $\mathbf{x} \in \mathbb{R}^p$ and has ρ-Lipschitz Hessians i.e., for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$, we have $\|\nabla^2_{\mathbf{w}} f(\mathbf{x}) - \nabla^2_{\mathbf{w}} f(\mathbf{y})\|_2 \leq \rho \cdot \|\mathbf{x} - \mathbf{y}\|_2$ where $\|\cdot\|_2$ for matrices denotes the spectral/operator norm of the matrix.

Before commencing with the actual proof, we first present an overview of the proof. The definition of the SSa property clearly demarcates three different regimes:

1. Non-stationary points, i.e., points where gradient is “large” enough: in this case, standard (stochastic) gradient descent is powerful enough to ensure a large enough decrease in the objective function value in a single step[6]

2. Saddle points, i.e., points where the gradient is close to $\mathbf{0}$. Here, the SSa property ensures that at least one highly ‘negative” Hessian direction exists: in this case, traditional (stochastic) gradient descent may fail but the additional noise ensures an escape from the saddle point with high probability

3. Local minima, i.e., points where gradient is close to $\mathbf{0}$ but the have a positive semi definite Hessian due to strong convexity: in this case, standard (stochastic) gradient descent by itself would converge to the corresponding local minima

The above three regimes will be formally studied below in three separate lemmata. Note that the analysis for non-stationary points as well as for points near local minima is similar to the standard stochastic gradient descent analysis for convex functions. However, the analysis for saddle points is quite interesting and shows that the added random noise ensures an escape from the saddle point.

To further understand the inner workings of the NGD algorithm, let us perform a warm-up exercise by showing that the NGD algorithm will, with high probability, escape the saddle point in the function $f(x, y) = x^2 - y^2$ that we considered in Figure 6.2.

[6] See Exercise 6.2.

Theorem 6.1. Consider the function $f(x, y) = x^2 - y^2$ on two variables. If initialized at the saddle point $(0, 0)$ of this function with step length $\eta < 1$, with high probability, we have $f(x^t, y^t) \to -\infty$ as $t \to \infty$.

Proof. For an illustration, see Figure 6.2. Note that the function $f(x, y) = x^2 - y^2$ has trivial minima at the limiting points $(0, \pm\infty)$ where the function value approaches $-\infty$. Thus, the statement of the theorem claims that NGD approaches the "minimum" function value.

In any case, we are interested in showing that NGD escapes the saddle point $(0, 0)$. The gradient of f is $\mathbf{0}$ at the origin $(0, 0)$. Thus, if a gradient descent procedure is initialized at the origin for this function, it will remain stuck there forever making no non-trivial updates.

The NGD algorithm on the other hand, when initialized at the saddle point $(0, 0)$, after t iterations, can be shown to reach the point (x^t, y^t) where $x^t = \sum_{\tau=0}^{t-1}(1-\eta)^{t-\tau-1}\zeta_1^\tau$ and $y^t = \sum_{\tau=0}^{t-1}(1+\eta)^{t-\tau-1}\zeta_2^\tau$ and $(\zeta_1^\tau, \zeta_2^\tau) \in \mathbb{R}^2$ is the noise vector added to the gradient at each step. Since $\eta < 1$, as $t \to \infty$, it is easy to see that with high probability, we have $x^t \to 0$ while $|y^t| \to \infty$ which indicates a successful escape from the saddle point, as well as progress towards the global optima. □

We now formalize the intuitions developed above. The following lemma shows that even if we are at a saddle point, NGD will still ensure a large drop in function value in not too many steps. The proof of this result is a bit subtle and we will just provide a sketch.

Lemma 6.2. If NGD is executed on a function $f : \mathbb{R}^p \to \mathbb{R}$ that is β-strongly smooth, satisfies the $(\alpha, \gamma, \kappa, \xi)$-SSa property and has ρ-Lipschitz Hessians, with step length $\eta \leq 1/(\beta+\rho)^2$, then if an iterate $\mathbf{x}^t$ satisfies $\left\|\nabla f(\mathbf{x}^t)\right\|_2 \leq \sqrt{\eta\beta}$ and $\lambda_{\min}\nabla^2 f(\mathbf{x}^t) \leq -\gamma$, then NGD ensures that after at most $s \leq \frac{3\log p}{\eta\gamma}$ steps,

$$\mathbb{E}\left[f(\mathbf{x}^{t+s}) \,|\, \mathbf{x}^t\right] \leq f(\mathbf{x}^t) - \eta/4$$

Proof. For the sake of notational simplicity, let $t = 0$. The overall idea of the proof is to rely on that one large negative eigendirection in the Hessian to induce a drop in function value. The hope is that random fluctuations will eventually nudge NGD in the steep descent direction and upon discovery, the larger and larger gradient values will accumulate to let the NGD procedure escape the saddle.

Since the effects of the Hessian are most apparent in the second order Taylor expansion, we will consider the following function

$$\widehat{f}(\mathbf{x}) = f(\mathbf{x}^0) + \left\langle \nabla f(\mathbf{x}^0), \mathbf{x} - \mathbf{x}^0 \right\rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}^0)^\top \mathcal{H}(\mathbf{x} - \mathbf{x}^0),$$

where $\mathcal{H} = \nabla^2 f(\mathbf{x}^0) \in \mathbb{R}^{p \times p}$. The proof will proceed by first imagining that NGD was executed on the function $\widehat{f}(\cdot)$ instead, showing that the function value indeed drops, and then finishing off by showing that things do not change too much if NGD is executed on the function $f(\cdot)$ instead. Note that to make this claim, we will need Hessians to vary smoothly which is why we assumed the Lipschitz Hessian condition. This seems to be a requirement for several follow-up results as well [Jin et al., 2017, Agarwal et al., 2017]. An interesting and challenging open problem is to obtain similar results for non-convex optimization without the Lipschitz-Hessian assumption.

We will let $\mathbf{x}^t, t \geq 1$ denote the iterates of NGD when executed on $f(\cdot)$ and $\widehat{\mathbf{x}}^t$ denote the iterates of NGD when executed on $\widehat{f}(\cdot)$. We will fix $\widehat{\mathbf{x}}^0 = \mathbf{x}^0$. Using some careful calculations, we can get

$$\widehat{\mathbf{x}}^t - \widehat{\mathbf{x}}^0 = -\eta \sum_{\tau=0}^{t-1} (I - \eta\mathcal{H})^\tau \nabla f(\mathbf{x}^0) - \eta \sum_{\tau=0}^{t-1} (I - \eta\mathcal{H})^{t-\tau-1} \boldsymbol{\zeta}^\tau$$

We note that both terms in the right hand expression above correspond to “small” vectors. The first term is small as we know that $\widehat{\mathbf{x}}^0$ satisfies $\left\|\nabla f(\widehat{\mathbf{x}}^0)\right\|_2 \leq \sqrt{\eta\beta}$ by virtue of being close to a stationary point as is assumed in the statement of this result. The second term is small as $\boldsymbol{\zeta}^\tau$ are random unit vectors with expectation $\mathbf{0}$. Using these intuitions, we will first show that $\widehat{f}(\widehat{\mathbf{x}}^t)$ is significantly smaller than $\widehat{f}(\mathbf{x}^0) = f(\mathbf{x}^0)$ after sufficiently many steps. Then using the property of Lipschitz Hessians, we will obtain obtain a descent guarantee for $f(x^t)$.

Now, notice that NGD chooses the noise vectors $\boldsymbol{\zeta}^\tau$ for any $\tau \geq 0$, independently of $\mathbf{x}^0$ and $\mathcal{H}$. Moreover, for any two $\tau \neq \tau'$, the vectors $\boldsymbol{\zeta}^\tau, \boldsymbol{\zeta}^{\tau'}$ are also chosen independently. We also know that the noise vectors are isotropic i.e., $\mathbb{E}\left[\boldsymbol{\zeta}^\tau (\boldsymbol{\zeta}^\tau)^\top\right] = I_p$. These observations and some straightforward calculations give us the following upper bound on the suboptimality of $\mathbf{x}^T$ with respect to the Taylor approximation $\widehat{f}$. As we have noted, this upper bound can be converted to an upper bound on the suboptimality of $\mathbf{x}^T$ with respect to the actual function f using some more effort.

$$\begin{aligned}
&\mathbb{E}[\widehat{f}(\widehat{\mathbf{x}}^T) - f(\widehat{\mathbf{x}}^0)]\\
&\qquad = -\eta\nabla f(\mathbf{x}^0)^\top \sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^\tau \nabla f(\mathbf{x}^0)\\
&\qquad\quad + \frac{\eta^2}{2}\nabla f(\mathbf{x}^0)^\top \sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^\tau H \sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^\tau \nabla f(\mathbf{x}^0)\\
&\qquad\quad + \frac{\eta^2}{2}\mathrm{tr}\left(\sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^{2\tau}\mathcal{H}\right)\\
&\qquad = -\eta\nabla f(\mathbf{x}^0)^\top B\nabla f(\mathbf{x}^0) + \frac{\eta^2}{2}\mathrm{tr}\left(\sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^{2\tau}\mathcal{H}\right),
\end{aligned}$$

where $\mathrm{tr}(\cdot)$ is the trace operator and $B = \sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^\tau - \frac{\eta}{2}\sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^\tau\mathcal{H}\sum_{\tau=0}^{t-1}(I_p - \eta\mathcal{H})^\tau$. It is easy to verify that $B \succeq 0$ for all step lengths $\eta \leq \frac{1}{\|\mathcal{H}\|_2}$ i.e., all $\eta \leq \frac{1}{\beta}$.

For any value of $t \geq \frac{3\log p}{\eta\gamma}$ (which is the setting for the parameter s in the statement of the theorem), the second term in the final expression above can be simplified to give us

$$\frac{\eta^2}{2}\mathrm{tr}\left(\sum_{\tau=0}^{t-1}(I - \eta\mathcal{H})^{2\tau}\mathcal{H}\right) = \sum_{i=1}^{p}\lambda_i(\sum_{\tau=0}^{t-1}(1 - \eta\lambda_i)^\tau) \leq -\frac{\eta}{2},$$

where λ_i is the i-th eigenvalue of $\mathcal{H}$, by using $\lambda_p \leq -\gamma$. This gives us

$$\mathbb{E}\left[\widehat{f}(\widehat{\mathbf{x}}^T) - f(\mathbf{x}^0)\right] \leq -\frac{\eta}{2}.$$

Note that the above equation only shows descent for $\widehat{f}(\widehat{x}^T)$. One can now show [Ge et al., 2015, Lemma 19] that the iterates obtained by NGD on $\widehat{f}(\cdot)$ do not deviate too far from those obtained on the actual function $f(\cdot)$ using the Lipschitz-Hessian property. The proof is concluded by combining these two results. □

Lemma 6.3. If NGD is executed on a function that is β-strongly smooth and satisfies the $(\alpha, \gamma, \kappa, \xi)$-SSa property, with step length $\eta \leq \frac{1}{\beta} \cdot \min\left\{1, \kappa^2\right\}$, then for any iterate $\mathbf{x}^t$ that satisfies $\left\|\nabla f(\mathbf{x}^t)\right\|_2 \geq \sqrt{\eta\beta}$, NGD ensures that

$$\mathbb{E}\left[f(\mathbf{x}^{t+1}) \mid \mathbf{x}^t\right] \leq f(\mathbf{x}^t) - \frac{\beta}{2} \cdot \eta^2.$$

Proof. This is the most carefree case as we are neither close to any local optima, nor a saddle point. Unsurprisingly, the proof of this lemma follows from standard arguments as we have assumed the function to be strongly smooth. Since $\mathbf{x}^{t+1} = \mathbf{x}^t - \eta \cdot (\nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t)$, we have, by an application of the strong smoothness property (see Definition 2.4)

$$f(\mathbf{x}^{t+1}) \leq f(\mathbf{x}^t) - \left\langle \nabla f(\mathbf{x}^t), \eta \cdot (\nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t)\right\rangle + \frac{\beta\eta^2}{2} \cdot \left\|\nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t\right\|_2^2.$$

Using the facts $\left\|\boldsymbol{\zeta}^t\right\|_2 = 1$, $\mathbb{E}\left[\boldsymbol{\zeta}^t \mid \mathbf{x}^t\right] = \mathbf{0}$, we get

$$\mathbb{E}\left[f(\mathbf{x}^{t+1}) \mid \mathbf{x}^t\right] \leq f(\mathbf{x}^t) - \eta\left(1 - \frac{\beta\eta}{2}\right)\left\|\nabla f(\mathbf{x}^t)\right\|_2^2 + \frac{\beta\eta^2}{2},$$

Since we have $\eta \leq \frac{1}{\beta} \cdot \min\left\{1, \kappa^2\right\}$ by assumption and $\left\|f(\mathbf{x}^t)\right\|_2 \geq \sqrt{\eta\beta}$, we get

$$\mathbb{E}\left[f(\mathbf{x}^{t+1} \mid \mathbf{x}^t\right] \leq f(\mathbf{x}^t) - \frac{\beta}{2} \cdot \eta^2,$$

which proves the result. □

The final intermediate result is an *entrapment* lemma. It shows that once NGD gets sufficiently close to a local optimum, it gets trapped there for a really long time. Although the function f satisfies strong convexity and smoothness properties in the neighborhood $\mathcal{B}_2(\mathbf{x}^*, 2\xi)$, the proof of this result is still non-trivial due to the perturbations $\boldsymbol{\zeta}^t$. Had the perturbations not been there, we could have utilized the analysis of the PGD algorithm[7] to show that we would converge to the local optimum $\mathbf{x}^*$ at a linear rate.

The problem is that the perturbations do not diminish – we always have $\left\|\boldsymbol{\zeta}^t\right\|_2 = 1$. This prevents us from ever converging to the local optima. Moreover, a sequence of unfortunate perturbations may have

[7] See Exercise 6.3.

us kicked out of this nice neighborhood. The next result shows that we will not get kicked out of the neighborhood of $\mathbf{x}^*$ for a really long time.

Lemma 6.4. If NGD is executed on a function that is β-strongly smooth and satisfies the $(\alpha, \gamma, \kappa, \xi)$-SSa property, with step length $\eta \leq \min\left\{\frac{\alpha}{\beta^2}, \xi^2 \log^{-1}(\frac{1}{\delta\xi})\right\}$ for some $\delta > 0$, then if some iterate $\mathbf{x}^t$ satisfies $\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2 \leq \xi$, then NGD ensures that with probability at least $1 - \delta$, for all $s \in \left[t, t + \frac{1}{\eta^2}\log\frac{1}{\delta}\right]$, we have

$$\left\|\mathbf{x}^s - \mathbf{x}^*\right\|_2 \leq \sqrt{\eta \log \frac{1}{\eta\delta}} \leq \xi.$$

Proof. Using strong convexity in the neighborhood of $\mathbf{x}^*$ and the fact that $\mathbf{x}^*$, being a local minimum, satisfies $\nabla f(\mathbf{x}^*) = \mathbf{0}$, gives us

$$f(\mathbf{x}^t) \geq f(\mathbf{x}^*) + \frac{\alpha}{2}\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2$$
$$f(\mathbf{x}^*) \geq f(\mathbf{x}^t) + \left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^* - \mathbf{x}^t \right\rangle + \frac{\alpha}{2}\left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2.$$

Together, the above two expressions give us

$$\left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \right\rangle \geq \alpha \cdot \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2.$$

Since f is β-smooth, using the co-coercivity of the gradient for smooth convex functions (recall that f is strongly convex, in this neighborhood of $\mathbf{x}^*$), we conclude that f has β-Lipschitz gradients, which gives us

$$\left\|\nabla f(\mathbf{x}^t)\right\|_2 = \left\|\nabla f(\mathbf{x}^t) - \nabla f(\mathbf{x}^*)\right\|_2 \leq \beta \cdot \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2.$$

Using the above results, $\mathbb{E}\left[\boldsymbol{\zeta}^t \mid \mathbf{x}^t\right] = \mathbf{0}$ and $\eta \leq \frac{\alpha}{\beta^2}$ gives us

$$\begin{aligned}
\mathbb{E}\left[\left\|\mathbf{x}^{t+1} - \mathbf{x}^*\right\|_2^2 \;\middle|\; \mathbf{x}^t\right] &= \mathbb{E}\left[\left\|\mathbf{x}^t - \eta(\nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t) - \mathbf{x}^*\right\|_2^2 \;\middle|\; \mathbf{x}^t\right] \\
&= \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 - 2\eta\left\langle \nabla f(\mathbf{x}^t), \mathbf{x}^t - \mathbf{x}^* \right\rangle \\
&\quad + \eta^2\left\|\nabla f(\mathbf{x}^t)\right\|_2^2 + \eta^2 \\
&\leq (1 - 2\eta\alpha + \eta^2\beta^2) \cdot \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 + \eta^2 \\
&\leq (1 - \eta\alpha) \cdot \left\|\mathbf{x}^t - \mathbf{x}^*\right\|_2^2 + \eta^2,
\end{aligned}$$

which, upon some manipulation, gives us

$$\mathbb{E}\left[\left\|\mathbf{x}^{t+1}-\mathbf{x}^*\right\|_2^2-\frac{\eta}{\alpha}\;\middle|\;\mathbf{x}^t\right]\leq(1-\eta\alpha)\left(\left\|\mathbf{x}^t-\mathbf{x}^*\right\|_2^2-\frac{\eta}{\alpha}\right)$$

This tells us that on expectation, the distance of the iterates $\mathbf{x}^t$ from the local optimum $\mathbf{x}^*$ will hover around $\sqrt{\frac{\eta}{\alpha}}$ which indicates towards the claimed result. The subsequent steps in the proof of this result require techniques from martingale theory which we wish to avoid. We refer the reader to [Ge et al., 2015, Lemma 16] for the details. □

Notice that the above result traps the iterates within a radius ξ ball around the local minimum $\mathbf{x}^*$. Also notice that all points that are approximate local minima satisfy the preconditions of this theorem due to the SSa property and consequently the NGD gets trapped for these points. We now present the final convergence guarantee for NGD.

Theorem 6.5. For any $\epsilon, \delta > 0$, suppose NGD is executed on a function that is β-strongly smooth, has ρ-Lipschitz Hessians, and satisfies the $(\alpha, \gamma, \kappa, \xi)$-SSa property, with a step length $\eta < \eta_{\max} = \min\left\{\frac{\epsilon^2}{\log(1/\epsilon\delta)}, \frac{\alpha}{\beta^2}, \frac{\xi^2}{\log(1/\xi\delta)}, \frac{1}{(\beta+\rho)^2}, \frac{\kappa^2}{\beta}\right\}$. Then, with probability at least $1-\delta$, after $T \geq \log p/\eta^2 \cdot \log(2/\delta)$ iterations, NGD produces an iterate $\mathbf{x}^T$ that is ϵ-close to some local optimum $\mathbf{x}^*$ i.e., $\left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2 \leq \epsilon$.

Proof. We partition the space $\mathbb{R}^p$ into 3 regions

1. $\mathfrak{R}_1 = \{\mathbf{x} : \|f(\mathbf{x})\|_2 \geq \sqrt{\eta\beta}\}$
2. $\mathfrak{R}_2 = \{\mathbf{x} : \|f(\mathbf{x})\|_2 < \sqrt{\eta\beta}, \lambda_{\min}(\nabla^2 f(\mathbf{x})) \leq -\gamma\}$
3. $\mathfrak{R}_3 = \mathbb{R}^p\backslash(\mathfrak{R}_1 \cup \mathfrak{R}_2)$

Since $\sqrt{\eta\beta} \leq \kappa$ due to the setting of $\eta_{\max}$, the region $\mathfrak{R}_1$ contains all points considered non-stationary by the SSa property (Definition 6.1) and possibly some other points as well. For this reason, region $\mathfrak{R}_2$ can be shown to contain only saddle points. Since the SSa property assures us that a point that is neither non-stationary nor a saddle point is definitely an approximate local minimum, we deduce that the region $\mathfrak{R}_3$ contains only approximately local minima.

The proof will use the following line of argument: since Lemmata 6.3 and 6.2 assure us that whenever the NGD procedure is in regions $\mathfrak{R}_1$ or $\mathfrak{R}_2$ there is a large drop in function value, we should expect the procedure to enter region $\mathfrak{R}_3$ sooner or later, since the function value cannot go on decreasing indefinitely. However, Lemma 6.4 shows that once we are in region $\mathfrak{R}_3$, we are trapped there. In the following analysis, we will ignore all non-essential constants and log factors.

Recall that we let the NGD procedure last for $T = 1/\eta^2 \log(2/\delta)$ steps. Below we will show that in any sequence of $1/\eta^2$ steps, there is at least a $1/2$ chance of encountering an iterate $\mathbf{x}^t \in \mathfrak{R}_3$. Since the entire procedure lasts $\log(1/\delta)$ such sequences, we will conclude, by union bound, that with probability at least $1 - \delta/2$, we will encounter at least one iterate in the region $\mathfrak{R}_3$ in the T steps we execute.

However, Lemma 6.4 shows that once we enter the $\mathfrak{R}_3$ neighborhood, with probability at least $1 - \delta/2$, we are trapped there for at least T steps. Applying the union bound will establish that with probability at least $1 - \delta$, the NGD procedure will output $\mathbf{x}^T \in \mathfrak{R}_3$. Since we set $\eta \leq \epsilon^2/\log(1/\epsilon\delta)$, this will conclude the proof.

We now left with proving that in every sequence of $1/\eta^2$ steps, there is at least a $1/2$ chance of NGD encountering an iterate $\mathbf{x}^t \in \mathfrak{R}_3$. To do so, we set up the notion of *epochs*. These will basically correspond to the amount of time taken by NGD to reduce the function value by a significant amount. The first epoch starts at time $\tau_1 = 0$. Subsequently, we define

$$\tau_{i+1} = \begin{cases} \tau_i + 1 & \text{if } \mathbf{x}^{\tau_i} \in \mathfrak{R}_1 \cup \mathfrak{R}_3 \\ \tau_i + \frac{1}{\eta} & \text{if } \mathbf{x}^{\tau_i} \in \mathfrak{R}_2 \end{cases}$$

Ignoring constants and other non-essential factors, we can rewrite the results of Lemmata 6.3 and 6.2 as follows

$$\mathbb{E}\left[f(\mathbf{x}^{\tau_{i+1}}) - f(\mathbf{x}^{\tau_i}) \mid \mathbf{x}^{\tau_i} \in \mathfrak{R}_1\right] \leq -\eta^2$$
$$\mathbb{E}\left[f(\mathbf{x}^{\tau_{i+1}}) - f(\mathbf{x}^{\tau_i}) \mid \mathbf{x}^{\tau_i} \in \mathfrak{R}_2\right] \leq -\eta$$

Putting these together gives us

$$\mathbb{E}\left[f(\mathbf{x}^{\tau_{i+1}}) - f(\mathbf{x}^{\tau_i}) \mid \mathbf{x}^{\tau_i} \notin \mathfrak{R}_3\right] \leq -\mathbb{E}\left[(\tau_{i+1} - \tau_i) \mid \mathbf{x}^{\tau_i} \notin \mathfrak{R}_3\right] \cdot \eta^2$$

Define the event $\mathfrak{E}_t := \{\nexists\, j \leq t : \mathbf{x}^j \in \mathfrak{R}_3\}$ and let $\mathbf{1}_E$ denote the indicator variable for event E i.e., $\mathbf{1}_E = 1$ if E occurs and $\mathbf{1}_E = 0$ otherwise.

Then we have

$$\begin{aligned}
&\mathbb{E}\left[f(\mathbf{x}^{\tau_{i+1}})\cdot\mathbf{1}_{\mathfrak{E}_{\tau_{i+1}}}-f(\mathbf{x}^{\tau_i})\cdot\mathbf{1}_{\mathfrak{E}_{\tau_i}}\right]\\
&\quad=\mathbb{E}\left[f(\mathbf{x}^{\tau_{i+1}})\cdot(\mathbf{1}_{\mathfrak{E}_{\tau_{i+1}}}-\mathbf{1}_{\mathfrak{E}_{\tau_i}})\right]+\mathbb{E}\left[(f(\mathbf{x}^{\tau_{i+1}})-f(\mathbf{x}^{\tau_i}))\cdot\mathbf{1}_{\mathfrak{E}_{\tau_i}}\right]\\
&\quad\leq B\cdot(\mathbb{P}\left[\mathfrak{E}_{\tau_{i+1}}\right]-\mathbb{P}\left[\mathfrak{E}_{\tau_i}\right])-\eta^2\cdot\mathbb{E}\left[\tau_{i+1}-\tau_i\mid\mathbf{1}_{\mathfrak{E}_{\tau_i}}\right]\cdot\mathbb{P}\left[\mathfrak{E}_{\tau_i}\right],
\end{aligned}$$

where we have used the fact that $|f(\mathbf{x})|\leq B$ for all $\mathbf{x}\in\mathbb{R}^p$. Since $\mathfrak{E}_{t+1}\Rightarrow\mathfrak{E}_t$, we have $\mathbb{P}\left[\mathfrak{E}_{t+1}\right]\leq\mathbb{P}\left[\mathfrak{E}_t\right]$. Summing the expressions above from $i=1$ to j and using $\mathbf{x}^1\notin\mathfrak{R}_3$ gives us

$$\mathbb{E}\left[f(\mathbf{x}^{\tau_{j+1}})\cdot\mathbf{1}_{\mathfrak{E}_{\tau_{j+1}}}\right]-f(\mathbf{x}^1)\leq-\eta^2\mathbb{E}\left[\tau_{j+1}\right]\cdot\mathbb{P}\left[\mathfrak{E}_{\tau_j}\right]$$

However, since the function is bounded $|f(\mathbf{x})|\leq B$, the left hand side cannot be smaller than $-2B$. Thus, if $\mathbb{E}\left[\tau_{j+1}\right]\geq 4B/\eta^2$ then we must have $\mathbb{P}\left[\mathfrak{E}_{\tau_j}\right]\leq 1/2$. This concludes the proof. □

We have not presented exact constants in the results to avoid clutter. The NGD algorithm actually requires the step length to be set to $\eta<\eta_{\max}\leq\frac{1}{p}$ where p is the ambient dimensionality. Now, since NGD is run for $\Omega\left(1/\eta^2\right)$ iterations and each iteration takes $\mathcal{O}\left(p\right)$ time to execute, the total run-time of NGD is $\mathcal{O}\left(p^3\right)$ which can be prohibitive. The bibliographic notes discuss more recent results that offer run-times that are linear in p. Also note that the NGD procedure requires $\widetilde{\mathcal{O}}\left(1/\epsilon^4\right)$ iterations to converge within an ϵ distance of a local optimum.

6.6 Constrained Optimization with Non-convex Objectives

We will now present extensions to the above discussion to cases where the optimization problem is constrained. We will concentrate on constrained optimization problems with equality constraints.

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^p}\quad & f(\mathbf{x})\\ \text{s.t.}\quad & c_i(\mathbf{x})=0, i\in[m]\end{aligned}\qquad\text{(CNOPT)}$$

Let $\mathcal{W}:=\{\mathbf{x}\in\mathbb{R}^p : c_i(\mathbf{x})=0, i\in[m]\}$ denote the constraint set. In general $\mathcal{W}$ is a *manifold* and we will assume that this manifold is *nice* in that it is smooth and does not have corners. It is natural to attempt

Algorithm 7 Projected Noisy Gradient Descent (PNGD)

```
Input: Objective $f$, max step length $\eta_{\max}$, tolerance $\epsilon$
Output: A locally optimal point $\widehat{\mathbf{x}} \in \mathbb{R}^p$
$\mathbf{x}^1 \leftarrow \mathsf{INITALIZE}()$
Set $T \leftarrow 1/\eta^2$, where $\eta = \min \left\{ \epsilon^2 / \log^2(1/\epsilon), \eta_{\max} \right\}$
for $t = 1, 2, \ldots, T$ do
    Sample perturbation $\boldsymbol{\zeta}^t \sim S^{p-1}$ //Random pt. on unit sphere
    $\mathbf{g}^t \leftarrow \nabla f(\mathbf{x}^t) + \boldsymbol{\zeta}^t$
    $x^{t+1} \leftarrow \Pi_{\mathcal{W}}(\mathbf{x}^t - \eta \cdot \mathbf{g}^t)$        //Project onto constraint set
end for
return $\mathbf{x}^T$
```

to solve the problem using a gPGD-like approach. Indeed, Algorithm 7 extends the NGD algorithm to include a projection step. The algorithm is actually similar to the NGD algorithm save the step that projects the iterates onto the manifold $\mathcal{W}$. This projection step can be tricky but can be efficiently solved, for instance when the constraint functions c_i are linear (see [Boyd and Vandenberghe, 2004, Section 6.2]).

The complete analysis of this algorithm (see [Ge et al., 2015, Appendix B]), although similar in parts to that of the NGD algorithm, is beyond the scope of this monograph. Instead, we just develop the design concepts and intuition used in the analysis. The first step is to convert the above constrained optimization into an unconstrained one so that some of the tools used for NGD may be reapplied here.

A very common way to do so is to first construct the *Lagrangian* [Boyd and Vandenberghe, 2004, Chapter 5] of the problem defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i c_i(\mathbf{x}),$$

where λ_i are *Lagrange multipliers*. It is easy to verify that the solution to the problem ((CNOPT)) coincides with that of the following problem

$$\min_{\mathbf{x} \in \mathbb{R}^p} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \ \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$$

Note that the above problem is unconstrained. For any $\mathbf{x} \in \mathbb{R}^p$, define $\boldsymbol{\lambda}^*(\mathbf{x}) := \arg\min_{\boldsymbol{\lambda}} \ \left\| \nabla f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i \nabla c_i(\mathbf{x}) \right\|$ and $\mathcal{L}^*(\mathbf{x}) :=$

$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*(\mathbf{x}))$. We also define the tangent and normal spaces of the manifold $\mathcal{W}$ as follows.

Definition 6.2 (Normal and Tangent Space)**.** For a manifold $\mathcal{W}$ defined as the intersection of m constraints of the form $c_i(\mathbf{x}) = 0$, given any $\mathbf{x} \in \mathcal{W}$, define its tangent space as $\mathcal{T}(\mathbf{x}) = \{\mathbf{v} \mid \langle \nabla c_i(\mathbf{x}), \mathbf{v}\rangle = 0, i \in [m]\}$ and its normal space as $\mathcal{T}^c(\mathbf{x}) = \text{span}\,\{\nabla c_1(\mathbf{x}), \ldots, \nabla c_m(\mathbf{x})\}$.

If we think of $\mathcal{W}$ as a smooth surface, then at any point, the tangent space defines the *tangent plane* of the surface at that point and the normal space consists of all vectors orthogonal to the tangent plane. The reason behind defining all the above quantities is the following. In unconstrained optimization we have the first and second order optimality conditions: if $\mathbf{x}^*$ is a local minimum for $f(\cdot)$ then $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*) \succ 0$.

Along the same lines, for constrained optimization problems, similar conditions exist characterizing stationary and optimal points: one can show [Wright and Nocedal, 1999] that if $\mathbf{x}^*$ is a local optimum for the constrained problem ((CNOPT)), then we must have $\nabla \mathcal{L}^*(\mathbf{x}^*) = \mathbf{0}$, as well as $\mathbf{v}^\top \nabla^2 \mathcal{L}^*(\mathbf{x}^*)\mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathcal{T}(\mathbf{x}^*)$. This motivates us to propose the following extension of the strict saddle property for constrained optimization problems.

Definition 6.3 (Strict Constrained Saddle Property [Ge et al., 2015])**.** A twice differentiable function $f(\mathbf{x})$ with a constraint set $\mathcal{W}$ is said to satisfy the $(\alpha, \gamma, \kappa, \xi)$-strict constrained saddle (SCSa) property, if for every local minimum $\mathbf{x}^* \in \mathcal{W}$, we have $\mathbf{v}^\top \mathcal{L}^*(\mathbf{x}')\mathbf{v} \geq \alpha$ for all $\mathbf{v} \in \mathcal{T}(\mathbf{x}'), \|\mathbf{v}\|_2 = 1$ and for all $\mathbf{x}'$ in the region $\mathcal{B}_2(\mathbf{x}^*, 2\xi)$, and moreover, any point $\mathbf{x}_0 \in \mathbb{R}^p$ satisfies at least one of the following properties:

1. (Non-Stationary) $\|\nabla \mathcal{L}^*(\mathbf{x}_0)\|_2 \geq \kappa$
2. (Strict Saddle) $\mathbf{v}^\top \nabla^2 \mathcal{L}^*(\mathbf{x}_0)\mathbf{v} \leq -\gamma$ for some $\mathbf{v} \in \mathcal{T}(\mathbf{x}_0), \|\mathbf{v}\|_2 = 1$
3. (Approx. Local Min.) For some local minimum $\mathbf{x}^*$, $\|\mathbf{x}_0 - \mathbf{x}^*\|_2 \leq \xi$.

The following local convergence result can then be shown for the PNGD algorithm.

Theorem 6.6. Suppose PNGD is executed on a constrained optimization problem that satisfies the $(\alpha, \gamma, \kappa, \xi)$-SCSa property, has ρ-Lipschitz Hessians and whose constraint set is a smooth manifold $\mathcal{W}$. Then there exists a constant $\eta_{\max} = \Theta(1)$ such that for any $\epsilon, \delta > 0$, if we set $\eta = \min\left\{\eta_{\max}, \frac{\epsilon^2}{\log(1/\epsilon\delta)}\right\}$, then with probability at least $1 - \delta$, after $T \geq \log p/\eta^2 \cdot \log(2/\delta)$ iterations, PNGD produces an iterate $\mathbf{x}^T$ that is ϵ-close to some local optimum $\mathbf{x}^*$ i.e., $\left\|\mathbf{x}^T - \mathbf{x}^*\right\|_2 \leq \epsilon$.

The PNGD procedure requires $\widetilde{\mathcal{O}}\left(1/\epsilon^4\right)$ iterations to converge within an ϵ distance of a local optimum, similar to NGD. The proof of the above theorem [Ge et al., 2015, Lemma 35] proceeds by showing that the PNGD updates can be rewritten as

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta \cdot (\nabla \mathcal{L}^*(\mathbf{x}^t) + \Pi_{\mathcal{T}(\mathbf{x}^t)}(\boldsymbol{\zeta}^t)) + \boldsymbol{\iota}^t,$$

where $\Pi_{\mathcal{T}(\mathbf{x}^t)}(\cdot)$ is the projection of a vector onto the tangent space $\mathcal{T}(\mathbf{x}^t)$ and $\boldsymbol{\iota}^t$ is an error vector with small norm $\left\|\boldsymbol{\iota}^t\right\|_2 \leq \eta^2$.

6.7 Application to Orthogonal Tensor Decomposition

Recall that we reduced the tensor decomposition problem to solving the following optimization problem ((LRTD))

$$\begin{array}{rl} \max & T(\mathbf{u}, \mathbf{u}, \mathbf{u}, \mathbf{u}) = \sum_{i=1}^r (\mathbf{u}_i^\top \mathbf{u})^4 \\ \text{s.t.} & \|\mathbf{u}\|_2 = 1, \end{array}$$

The above problem has internal symmetry with respect to permutation of the components, as well as sign flips (both $\mathbf{u}_i$ and $-\mathbf{u}_i$ are valid components) which gives rise to saddle points as we discussed earlier. Using some simple but slightly tedious calculations (see [Ge et al., 2015, Theorem 44, Lemmata 45, 46]) it is possible to show that the above optimization problem does satisfy the $(3, 7/p, 1/p^c, 1/p^c)$-SCSa property for some constant $c > 0$. All other requirements for Theorem 6.6 can also be shown to be satisfied.

It is easy to see that any solution to the problem must lie in the span of the components. Since the components $\mathbf{u}_i$ are orthonormal, this means that it suffices to look for solutions of the form $\mathbf{u} = \sum_{i=1}^r x_i \mathbf{u}_i$.

This gives us $T(\mathbf{u}, \mathbf{u}, \mathbf{u}, \mathbf{u}) = \sum_{i=1}^{r} x_i^4$, and $\|\mathbf{u}\|_2 = \|\mathbf{x}\|_2$ where $\mathbf{x} = [x_1, x_2, \ldots, x_r]$. This allows us to formulate the equivalent problem as

$$\begin{array}{ll} \min & -\|\mathbf{x}\|_4^4 \\ \text{s.t.} & \|\mathbf{x}\|_2 = 1, \end{array}$$

Note that the above problem is non-convex as the objective function is concave. However, it is also possible to show[8][9] that the only local minima of the optimization problem ((LRTD)) are $\pm\mathbf{u}_i$. This is most fortunate since it shows that all local minima are actually global minima! Thus, the *deflation* strategy alluded to earlier can be successfully applied. We discover one component, say $\mathbf{u}_1$ by applying the PNGD algorithm to ((LRTD)). Having recovered this component, we create a new tensor $T' = T - \mathbf{u}_1 \otimes \mathbf{u}_1 \otimes \mathbf{u}_1 \otimes \mathbf{u}_1$, apply the procedure again to discover a second component and so on.

The work of Ge et al. [2015] also discusses techniques to recover all r components simultaneously, tensors with different positive weights on the components, as well as tensors of other orders.

6.8 Exercises

Exercise 6.1. Show that the optimization problem in the formulation ((LRTD)) is non-convex in that it has a non-convex objective as well as a non-convex constraint set. Show that constraint set may be convexified without changing the optimum of the problem.

Exercise 6.2. Consider a differentiable function f that is β-strongly smooth but possibly non-convex. Show that if gradient descent is performed on f with a static step length $\eta \leq \frac{2}{\beta}$ i.e.,

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta \cdot \nabla f(\mathbf{x}^t)$$

then the function value f will never increase across iterations i.e., $f(\mathbf{x}^{t+1}) \leq f(\mathbf{x}^t)$. This shows that on smooth functions, gradient descent enjoys monotonic progress whenever the step length is small enough. *Hint*: apply the SS property relating consecutive iterates.

[8] See Exercise 6.4.

[9] See Exercise 6.5.

Exercise 6.3. For the same setting as the previous problem, show that if we have $\eta \in \left[\frac{1}{2\beta}, \frac{1}{\beta}\right]$ instead, then within $T = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ steps, gradient descent is guaranteed to identify an ϵ-stationary point i.e for some $t \leq T$, we must have $\left\|\nabla f(\mathbf{x}^t)\right\|_2 \leq \epsilon$.
Hint: first apply SS to show that $f(x^t) - f(x^{t+1}) \geq \frac{1}{4\beta} \cdot \left\|\nabla f(x^t)\right\|_2^2$. Then use the fact that the total improvement in function value over T time steps cannot exceed $f(x^0) - f^*$.

Exercise 6.4. Show that every component vector $\mathbf{u}_i$ is a globally optimal point for the optimization problem in ((LRTD)).
Hint: Observe the reduction in § 6.7. Show that it is equivalent to finding the minimum L_2 norm vector(s) among unit L_1 norm vectors. Find the Lagrangian dual of this optimization problem and simplify.

Exercise 6.5. Given a rank-r orthonormal tensor T, construct a nontrivial convex combination of its components that has unit L_2 norm but achieves a suboptimal objective value in the formulation ((LRTD)).

6.9 Bibliographic Notes

Progress in this area was extremely rapid at the time of writing this monograph. The following discussion outlines some recent advances.

The general problem of local optimization has been of interest in the optimization community for several years. Some of the earlier results in the area concentrated on second order or quasi-Newton methods, understandably since the Hessians can be used to distinguish between local optima and saddle points. Some key works include those of Nesterov and Polyak [2006], Dauphin et al. [2014], Sun et al. [2015], Yuan [2015]. However, these techniques can be expensive and hence, are not used in high dimensional applications.

Algorithms for specialized applications, such as the EM algorithm we studied in § 5, have received independent attention with respect to their local convergence properties. We refer the reader to the bibliographic notes § 5.11 for the same. In recent years, the problem of optimization of general non-convex problems has been looked at from several perspectives. Below we outline some of the major thrust areas.

First Order Methods Given their efficiency, the question of whether first order methods can offer local optimality has captured the interest of the field. In particular, much attention has been paid to the standard gradient descent and its variants. The work of Lee et al. [2016] showed that when initialized at a random point, gradient descent avoids saddle points almost surely, although the work provides no definite rate of convergence in general.

The subsequent works of Sun et al. [2015], Ge et al. [2015], Anandkumar and Ge [2016] introduced structural properties such as the strict saddle property, and demonstrated that crisp convergence rates can be ensured for problems that do satisfy these structural properties. It is notable that the work of Sun et al. [2015] reconsidered second order methods while Ge et al. [2015] were able to show that noisy stochastic gradient descent itself suffices.

The technique of using randomly perturbed (stochastic) gradients to escape saddle points receives attention in a more general framework of *Langevin Dynamics* which study cases when the perturbations are non-isotropic or else are applied at a scale that adapts to the problem. The recent work of Zhang et al. [2017] shows a powerful result that offers, for empirical risk minimization problems that are ubiquitous in machine learning, a convergence guarantee that ensures convergence to a local optimum of the population risk functional. This is significant since a majority of works in literature focus on identifying local optima of the empirical risk functional, which may correspond to very bad solutions with respect to the population risk.

Non-smooth Optimization All techniques we discussed in thus far in this monograph require the objective function to at least be differentiable. In fact, we go one step ahead to assume (restricted) smoothness. The work of Reddi et al. [2016] shows how variance reduction techniques may be exploited to ensure ϵ-convergence to a first-order stationary point $\mathbf{x}^t$ (effectively a saddle point) where we have $\left\| f(\mathbf{x}^t) \right\|_2 \leq \epsilon$, in no more than $\mathcal{O}\left(1/\epsilon\right)$ iterations of stochastic mini-batch gradient descent even when the objective function is non-convex and non-smooth at the same time. We urge the reader to confer this with Exercise 6.3 where it took $\mathcal{O}\left(1/\epsilon^2\right)$ iterations to do the same, that too using full gradient descent. Also notable is the fact that in our analysis (see Theorem 6.5), it took NGD $\mathcal{O}\left(1/\epsilon^4\right)$ steps to reach a local optimum. However, two caveats exist: 1) the method proposed in [Reddi et al., 2016] only reaches a saddle point, and 2) it assumes a *finite-sum* objective, i.e., one that has a decomposable form.

Accelerated Optimization The works of Carmon et al. [2017], Agarwal et al. [2017] extend the work of Reddi et al. [2016] by offering faster than $\mathcal{O}\left(1/\epsilon^2\right)$ convergence to a stationary point for general (non finite-sum) non-convex objectives. However, it should be noted that these techniques assume a smooth objective and convergence is guaranteed only to a saddle point, not a local minimum. Whereas the work of Carmon et al. [2017] invokes a variant of Nesterov's accelerated gradient technique to offer ϵ-convergence in $\mathcal{O}\left(1/\epsilon^{7/4}\right)$ iterations, the work of Agarwal et al. [2017] employs a second-order method to offer ϵ-convergence, also in $\mathcal{O}\left(1/\epsilon^{7/4}\right)$ iterations.

High-dimensional Optimization When considering problems in high-dimensions, it becomes difficult to execute algorithms whose run-times scale super-linearly in the ambient dimensionality. This is one reason why first-order methods are preferred. However, as we saw, the overall run-time of Theorem 6.5 scales as $\mathcal{O}\left(p^3\right)$ which makes the method prohibitive even for moderate dimensional problems. The work of Jin et al. [2017] proposes another perturbed gradient method that offers a run-time that depends only quasi-linearly on the ambient dimension. It should be noted that the works of Carmon et al. [2017], Agarwal et al. [2017] also offer run-times that are linear in the ambient dimensionality although, as noted earlier, convergence is only guaranteed to a saddle point by these methods.

Escaping Higher-order Saddles In our discussion, we were occupied with the problem of avoiding getting stuck at simple saddle points which readily reveal themselves by having distinctly positive and negative eigenvalues in the Hessian. However, there may exist more complex *degenerate saddle* points where the Hessian has only non-negative eigenvalues and thus, masquerades as a local minima. Such configurations yield complex cases such as *monkey saddles* and *connected saddles*. We did not address these. The work of Anandkumar and Ge [2016] proposes a method based on the Cubic Regularization algorithm of Nesterov and Polyak [2006] which is able to escape some of these more complex saddle points and achieve convergence to a point that enjoys *third order optimality*.

Training Deep Networks Given the popularity of deep networks in several areas of learning and signal processing, as well as the fact that the task of training deep networks corresponds to non-convex optimization, a lot of recent efforts have focused on the problem of efficiently and provably training deep networks using non-convex optimization techniques. Some notable works include provable methods for training multi-layered perceptrons [Goel and Klivans, 2017, Zhong et al., 2017, Li and Yuan, 2017] and non-overlapping convolutional networks [Brutzkus and Globerson, 2017]. Whereas the works [Brutzkus and Globerson, 2017, Zhong et al., 2017, Li and Yuan, 2017] utilize gradient-descent based techniques, Goel and Klivans [2017] uses an application of isotonic regression and kernel techniques. The work of Li and Yuan [2017] shows that the inclusion of an *identity map* into the network eases optimization by making the training problem well-posed.

Part III

Applications

7

Sparse Recovery

In this section, we will take a look at the sparse recovery and sparse linear regression as applications of non-convex optimization. These are extremely well studied problems and find applications in several practical settings. This will be the first of four "application" sections where we apply non-convex optimization techniques to real-world problems.

7.1 Motivating Applications

We will take the following two running examples to motivate the sparse regression problem in two different settings.

Gene Expression Analysis The availability of DNA micro-array gene expression data makes it possible to identify genetic explanations for a wide range of phenotypical traits such as physiological properties or even disease progressions. In such data, we are given say, for n human test subjects participating in the study, the expression levels of a large number p of genes (encoded as a real vector $\mathbf{x}_i \in \mathbb{R}^p$), and the corresponding phenotypical trait $y_i \in \mathbb{R}$. Figure 7.1 depicts this for a hypothetical study on Type-I diabetes. For the sake of simplicity, we

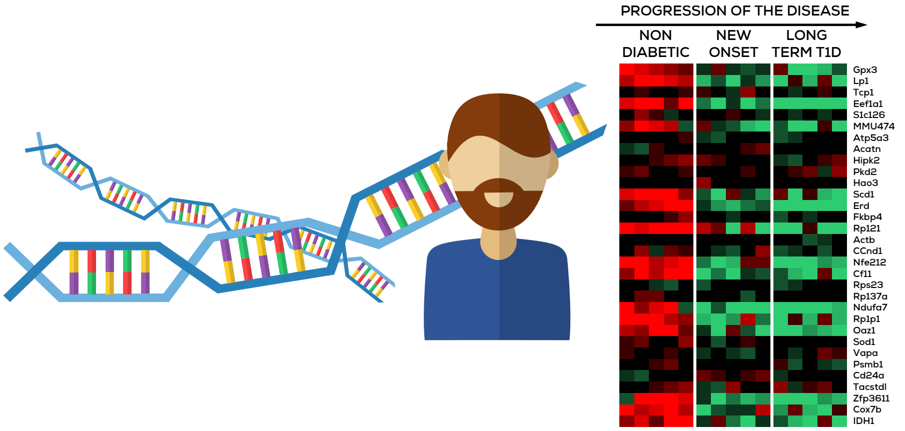


Figure 7.1: Gene expression analysis can help identify genetic bases for physiological conditions. The expression matrix on the right has 32 rows and 15 columns: each row represents one gene being tracked and each column represents one test subject. A bright red (green) shade in a cell indicates an elevated (depressed) expression level of the corresponding gene in the test subject with respect to a reference population. A black/dark shade indicates an expression level identical to the reference population. Notice that most genes do not participate in the progression of the disease in a significant manner. Moreover, the number of genes being tracked is much larger than the number of test subjects. This makes the problem of gene expression analysis, an ideal application for sparse recovery techniques. Please note that names of genes and expression levels in the figure are illustrative and do not correspond to actual experimental observations. Figure adapted from [Wilson et al., 2003].

are considering cases where the phenotypical trait can be modeled as a real number – this real number may indicate the severity of a condition or the level of some other biological measurement. More expressive models exist in literature where the target phenotypical trait is itself represented as a vector [see for example, Jain and Tewari, 2015].

For the sake of simplicity, we assume that the phenotypical response is linearly linked to the gene expression levels i.e. for some $\mathbf{w}^* \in \mathbb{R}^p$, we have $y_i = \mathbf{x}_i^\top \mathbf{w}^* + \eta_i$ where η_i is some noise. The goal then is to use gene expression data to deduce an estimate for $\mathbf{w}^*$. Having access to the model $\mathbf{w}^*$ can be instrumental in discovering possible genetic bases for diseases, traits etc. Consequently, this problem has significant implications for understanding physiology and developing novel medical interventions to treat and prevent diseases/conditions.

However, the problem fails to reduce to a simple linear regression problem for two important reasons. Firstly, although the number of genes whose expression levels are being recorded is usually very large (running into several tens of thousands), the number of samples (test subjects) is usually not nearly as large, i.e. $n \ll p$. Traditional regression algorithms fall silent in such data-starved settings as they usually expect $n > p$. Secondly, and more importantly, we do not expect all genes being tracked to participate in realizing the phenotype. Indeed, the whole objective of this exercise is to identify a small set of genes which most prominently influence the given phenotype. Note that this implies that the vector $\mathbf{w}^*$ is very sparse. Traditional linear regression cannot guarantee the recovery of a sparse model.

Sparse Signal Transmission and Recovery The task of transmitting and acquiring signals is a key problem in engineering. In several application areas such as magnetic resonance imagery and radio communication, *linear* measurement techniques, for example, sampling, are commonly used to acquire a signal. The task then is to reconstruct the original signal from these measurements. For sake of simplicity, suppose we wish to sense/transmit signals represented as vectors in $\mathbb{R}^p$. For various reasons (conserving energy, protection against data corruption etc), we may want to not transmit the signal directly and instead, create a *sensing mechanism* wherein a signal $\mathbf{w} \in \mathbb{R}^p$ is encoded into a signal $\mathbf{y} \in \mathbb{R}^n$ and it is $\mathbf{y}$ that is transmitted. At the receiving end $\mathbf{y}$ must be decoded back into $\mathbf{w}$. A popular way of creating sensing mechanisms – also called *designs* – is to come up with a set of n linear functionals $\mathbf{x}_i : \mathbb{R}^p \to \mathbb{R}$ and for any signal $\mathbf{w} \in \mathbb{R}^p$, record the values $y_i = \mathbf{x}_i^\top \mathbf{w}$. If we denote $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ and $\mathbf{y} = [y_1, \ldots, y_n]^\top$, then $\mathbf{y} = X\mathbf{w}$ is transmitted. Note as a special case that if $n = p$ and $\mathbf{x}_i = \mathbf{e}_i$, then $X = I_{p \times p}$ and $\mathbf{y} = \mathbf{w}$, i.e. we transmit the original signal itself.

If p is very large then we naturally look for designs with $n \ll p$. However, elementary results in algebra dictate that the recovery of $\mathbf{w}$ from $\mathbf{y}$ cannot be guaranteed even if $n = p - 1$. There is irrecoverable loss of information and there could be (infinitely) many signals $\mathbf{w}$ all of which map to the same transmitted signal $\mathbf{y}$ making it impossible to recover the original signal uniquely. A result similar in spirit called the Shannon-Nyquist theorem holds for analog or continuous-time signals. Although this seems to spell doom for any efforts to performed *compressed* sensing and transmission, these negative results can be overcome by observing that in several useful settings, the signals we are interested in, are actually very sparse i.e. $\mathbf{w} \in \mathcal{B}_0(s) \subset \mathbb{R}^p$, $s \ll p$.

This realization is critical since it allows the possibility of specialized design matrices to be used to transmit sparse signals in a highly compressed manner i.e. with $n \ll p$ but without any loss of information. However the recovery problem now requires a sparse vector to be recovered from the transmitted signal $\mathbf{y}$ and the design matrix X.

7.2 Problem Formulation

In both the examples considered above, we wish to recover a sparse linear model $\mathbf{w} \in \mathcal{B}_0(s) \subset \mathbb{R}^p$ that fits the data, i.e., $\mathbf{y}_i \approx \mathbf{x}_i^\top \mathbf{w}$, hence the name *sparse recovery*. In the gene analysis problem, the support of such a model $\mathbf{w}$ is valuable in revealing the identity of the genes involved in promoting the given phenotype/disease. Similarly, in the sparse signal recovery problem, $\mathbf{w}$ is the (sparse) signal itself.

This motivates the sparse linear regression problem. In the following, we shall use $\mathbf{x}_i \in \mathbb{R}^p$ to denote the features (gene expression levels/measurement functionals). Each feature will constitute a data point. There will be a *response* variable $y_i \in \mathbb{R}$ (phenotype response/measurement) associated with each data point. We will assume that the response variables are being generated using some underlying sparse *model* $\mathbf{w}^* \in \mathcal{B}_0(s)$ (gene influence pattern/sparse signal) as $y_i = \mathbf{x}_i^\top \mathbf{w}^* + \eta_i$ where η_i is some benign noise.

In both the gene expression analysis problem, as well as the sparse signal recovery problem, recovering $\mathbf{w}^*$ from the data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ then requires us to solve the following optimization problem: $\min_{\mathbf{w} \in \mathbb{R}^p, \|\mathbf{w}\|_0 \leq s} \sum_{i=1}^n \left(y_i - \mathbf{x}_i^\top \mathbf{w}\right)^2$. Rewriting the above in more succinct notation gives us

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^p \\ \|\mathbf{w}\|_0 \leq s}} \|\mathbf{y} - X\mathbf{w}\|_2^2, \tag{SP-REG}$$

where $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$ and $\mathbf{y} = [y_1, \ldots, y_n]^\top$. It is common to model the additive noise as *white noise* i.e. $\eta_i \sim \mathcal{N}(0, \sigma^2)$ for some $\sigma > 0$. It should be noted that the sparse regression problem in ((SP-REG)) is an NP-hard problem [Natarajan, 1995].

7.3 Sparse Regression: Two Perspectives

Although we cast both the gene analysis and sparse signal recovery problems in the same framework of sparse linear regression, there are subtle but crucial differences between the two problem settings.

Notice that the problem in sparse signal recovery is to come up with both, a design matrix X, as well as a recovery algorithm $\mathcal{A} : \mathbb{R}^n \times \mathbb{R}^{n \times p} \to \mathbb{R}^p$ such that all sparse signals can be accurately recovered from the measurements, i.e. $\forall \mathbf{w} \in \mathcal{B}_0(s) \subset \mathbb{R}^p$, we have $\mathcal{A}(X\mathbf{w}, X) \approx \mathbf{w}$.

On the other hand, in the gene expression analysis task, we do not have as direct a control over the effective design matrix X. In this case, the role of the design matrix is played by the gene expression data of the n test subjects. Although we may choose which individuals we wish to include in our study, even this choice does not give us a direct handle on the properties of the design matrix. Our job here is restricted to coming up with an algorithm $\mathcal{A}$ which can, given the gene expression data for p genes in n test subjects, figure out a sparse set of s genes which collectively promote the given phenotype i.e. for any $\mathbf{w} \in \mathcal{B}_0(s) \subset \mathbb{R}^p$ and given $X \in \mathbb{R}^{n \times p}$, we desire $\mathcal{A}(X\mathbf{w}, X) \approx \mathbf{w}$.

The distinction between the two settings is now apparent: in the first setting, the design matrix is totally in our control. We may design it to have specific properties as required by the recovery algorithm. However, in the second case, the design matrix is mostly given to us. We have no fine control over its properties.

This will make an important difference in the algorithms that operate in these settings since algorithms for sparse signal recovery would be able to make very stringent assumptions regarding the design matrix since we ourselves create this matrix from scratch. However, for the same reason, algorithms working in *statistical learning* settings such as the gene expression analysis problem, would have to work with relaxed

Algorithm 8 Iterative Hard-thresholding (IHT)

```
Input: Data $X, \mathbf{y}$, step length $\eta$, projection sparsity level $k$
Output: A sparse model $\widehat{\mathbf{w}} \in \mathcal{B}_0(k)$
$\mathbf{w}^1 \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ do
    $\mathbf{z}^{t+1} \leftarrow \mathbf{w}^t - \eta \cdot \frac{1}{n} X^\top (X\mathbf{w}^t - \mathbf{y})$
    $\mathbf{w}^{t+1} \leftarrow \Pi_{\mathcal{B}_0(k)}(\mathbf{z}^{t+1})$                //see § 3.1
end for
return $\mathbf{w}^t$
```

assumptions that can be expected to be satisfied by natural data. We will revisit this point later once we have introduced the reader to algorithms for performing sparse regression.

7.4 Sparse Recovery via Projected Gradient Descent

The formulation in ((SP-REG)) looks strikingly similar to the convex optimization problem ((CVX-OPT)) we analyzed in § 2 if we take $f(\mathbf{w}) = \|\mathbf{y} - X\mathbf{w}\|_2^2$ as the (convex) objective function and $\mathcal{C} = \mathcal{B}_0(s)$ as the (non-convex) constraint set. Given this, it is indeed tempting to adapt Algorithm 1 to solve this problem. The only difference would be that the projection step would now have to project onto a non-convex set $\mathcal{B}_0(s)$. However, as we have seen in § 3.1, this can be done efficiently. The resulting algorithm is a variant of the gPGD algorithm (Algorithm 2) that we studied in § 3 and is referred to as *Iterative Hard Thresholding* (IHT) in literature. The IHT algorithm is outlined in Algorithm 8.

It should come as no surprise that Algorithm 8 turns out to be extremely simple to implement, as well as extremely fast in execution, given that only gradient steps are required. Indeed, IHT is a method of choice for practitioners given its ease of use and speed. However, much less is clear about the recovery guarantees of this algorithm, especially since we have already stated that ((SP-REG)) is NP-hard to solve [Natarajan, 1995]. Note that since the problem involves non-convex constraints, Theorem 2.5 no longer applies. This seems to destroy all hope of proving a recovery guarantee, until one observes that the NP-hardness result does not preclude the possibility of solving this problem efficiently when there is special structure in the design matrix X.

Indeed if $X = I_{p \times p}$, it is trivial to recover *any* underlying sparse model $\mathbf{w}^*$ by simply returning $\mathbf{y}$. This toy case actually holds the key to efficient sparse recovery. Notice that when $X = I$, the design matrix is an *isometry* – it completely preserves the geometry of the space $\mathbb{R}^p$. However, one could argue that this is an uninteresting and expensive design with $n = p$. In a long line of illuminating results, which we shall now discuss, it was revealed that even if the design matrix is not a global isometry such as I but a *restricted isometry* that only preserves the geometry of sparse vectors, recovery is possible with $n \ll p$.

The observant reader would be wondering why are we not applying the gPGD analysis from § 3 directly here, and if notions similar to the RSC/RSS notions discussed there make sense here too. We request the reader to read on. We will find that not only do those notions extend here, but have beautiful interpretations. Moreover, instead of directly applying the gPGD analysis (Theorem 3.3), we will see a simpler convergence proof tailored to the sparse recovery problem which also gives a sharper result.

7.5 Restricted Isometry and Other Design Properties

As we observed previously, design matrices such as I_p which preserve the geometry of signals/models seem to allow for recovery of sparse signals. We now formalize this intuition further and develop specific conditions on the design matrix X which guarantee *universal recovery* i.e. for every $\mathbf{w} \in \mathcal{B}_0(s)$, it is possible to uniquely recover $\mathbf{w}$ from the measurements $X\mathbf{w}$.

It is easy to see that a design matrix that *identifies* sparse vectors cannot guarantee universal recovery. Suppose we have a design matrix $X \in \mathbb{R}^{n \times p}$ such that for some $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_0(s)$ and $\mathbf{w}_1 \neq \mathbf{w}_2$, we get $\mathbf{y}_1 = \mathbf{y}_2$ where $\mathbf{y}_1 = X\mathbf{w}_1$ and $\mathbf{y}_2 = X\mathbf{w}_2$. In this case, it is information theoretically impossible to distinguish between $\mathbf{w}_1$ and $\mathbf{w}_2$ on the basis of measurements made using X i.e. using $\mathbf{y}_1$ (or $\mathbf{y}_2$). Consequently, this design matrix cannot be used for universal recovery since it produces measurements that confuse between sparse vectors. It can be seen[1] that such a design matrix will not identify just one pair of sparse vectors but an infinite set of pairs (indeed an entire subspace) of sparse vectors.

Thus, it is clear that the design matrix must preserve the geometry of the set of sparse vectors while projecting them from a high p-dimensional space to a low n-dimensional space. Recall that in sparse recovery settings, we usually have $n \ll p$. The *Nullspace Property* presented below, and the others thereafter, are formalizations of this intuition. For any subset of coordinates $S \subset [p]$, let us define the set $\mathcal{C}(S) := \left\{ \mathbf{w} \in \mathbb{R}^p, \|\mathbf{w}_S\|_1 \geq \|\mathbf{w}_{\overline{S}}\|_1 \right\}$. This is the (convex) set of points that place a majority of their weight on coordinates in the set S. Define $\mathcal{C}(k) := \bigcup_{S:|S|=k} \mathcal{C}(S)$ to be the (non-convex[2]) set of points that place a majority of their weight on some k coordinates. Note that $\mathcal{C}(k) \supset \mathcal{B}_0(k)$ since k-sparse vectors put *all* their weight on some k coordinates.

Definition 7.1 (Nullspace Property [Cohen et al., 2009])**.** A matrix $X \in \mathbb{R}^{n \times p}$ is said to satisfy the null-space property of order k if $\ker(X) \cap \mathcal{C}(k) = \{\mathbf{0}\}$, where $\ker(X) = \{\mathbf{w} \in \mathbb{R}^p : X\mathbf{w} = \mathbf{0}\}$ is the kernel of the linear transformation induced by X (also called its null-space).

If a design matrix satisfies this property, then vectors in its null-space are disallowed from concentrating a majority of their weight on any k coordinates. Clearly no k-sparse vector is present in the null-space either. If a design matrix has the null-space property of order $2s$, then it can never identify two s-sparse vectors[3] – something that we have already seen as essential to ensure global recovery. A strengthening of the Nullspace Property gives us the Restricted Eigenvalue Property.

Definition 7.2 (Restricted Eigenvalue Property [Raskutti et al., 2010])**.** A matrix $X \in \mathbb{R}^{n \times p}$ is said to satisfy the restricted eigenvalue property of order k with constant α if for all $\mathbf{w} \in \mathcal{C}(k)$, we have $\frac{1}{n} \|X\mathbf{w}\|_2^2 \geq \alpha \cdot \|\mathbf{w}\|_2^2$.

This means that not only are k-sparse vectors absent from the null-space, they actually retain a good fraction of their length after projec-

[1] See Exercise 7.1.
[2] See Exercise 7.2.
[3] See Exercise 7.3.

tion as well. This means that if $k = 2s$, then for any $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{B}_0(s)$, we have $\frac{1}{n} \left\|X(\mathbf{w}_1 - \mathbf{w}_2)\right\|_2^2 \geq \alpha \cdot \left\|\mathbf{w}_1 - \mathbf{w}_2\right\|_2^2$. Thus, the distance between *any two* sparse vectors never greatly diminished after projection. Such behavior is the hallmark of an isometry, which preserves the geometry of vectors. The next property further explicates this and is, not surprisingly, called the Restricted Isometry Property.

Definition 7.3 (Restricted Isometry Property [Candès and Tao, 2005])**.** A matrix $X \in \mathbb{R}^{n \times p}$ is said to satisfy the restricted isometry property (RIP) of order k with constant $\delta_k \in [0, 1)$ if for all $\mathbf{w} \in \mathcal{B}_0(k)$, we have

$$(1 - \delta_k) \cdot \left\|\mathbf{w}\right\|_2^2 \leq \frac{1}{n} \left\|X\mathbf{w}\right\|_2^2 \leq (1 + \delta_k) \cdot \left\|\mathbf{w}\right\|_2^2.$$

The above property is most widely used in analyzing sparse recovery and compressive sensing algorithms. However, it is a bit restrictive since it requires the distortion parameters to be of the kind $(1 \pm \delta)$ for $\delta \in [0, 1)$. A generalization of this property that is especially useful in settings where the properties of the design matrix are not strictly controlled by us, such as the gene expression analysis problem, is the following notion of restricted strong convexity and smoothness.

Definition 7.4 (Restricted Strong Convexity/Smoothness Property [Jain et al., 2014, Jalali et al., 2011])**.** A matrix $X \in \mathbb{R}^{n \times p}$ is said to satisfy the α-restricted strong convexity (RSC) property and the β-restricted smoothness (RSS) property of order k if for all $\mathbf{w} \in \mathcal{B}_0(k)$, we have

$$\alpha \cdot \left\|\mathbf{w}\right\|_2^2 \leq \frac{1}{n} \left\|X\mathbf{w}\right\|_2^2 \leq \beta \cdot \left\|\mathbf{w}\right\|_2^2.$$

The only difference between the RIP and the RSC/RSS properties is that the former forces constants to be of the form $1 \pm \delta_k$ whereas the latter does not impose any such constraints. The reader will notice the similarities in the definition of restricted strong convexity and smoothness as given here and Definition 3.2 where we defined restricted strongly convexity and smoothness notions for general functions. The reader is invited to verify[4] that the two are indeed related.

Indeed, Definition 3.2 can be seen as a generalization of Definition 7.4 to general functions [Jain et al., 2014]. For twice differentiable

[4] See Exercise 7.4.

functions, both definitions can be seen as placing restrictions on the (restricted) eigenvalues of the Hessian of the function.

It is a useful exercise to verify[5] that these properties fall in a hierarchy: RSC-RSS ⇒ REP ⇒ NSP for an appropriate setting of constants. We will next establish the main result of this section: if the design matrix satisfies the RIP condition with appropriate constants, then the IHT algorithm does indeed guarantee universal sparse recovery. Subsequently, we will give pointers to recent results that guarantee universal recovery in gene expression analysis-like settings.

7.6 Ensuring RIP and other Properties

Since properties such as RIP, RE and RSC are so crucial for guaranteed sparse recovery, it is important to study problem settings in which these are satisfied by actual data. A lot of research has gone into explicit construction of matrices that provably satisfy the RIP property.

Random Designs: The simplest of these results are the so-called random design constructions which guarantee that if the matrix is sampled from certain well behaved distributions, then it will satisfy the RIP property with high probability. For instance, the work of Baraniuk et al. [2008] shows the following result:

Theorem 7.1. [Baraniuk et al., 2008, Theorem 5.2] Let $\mathcal{D}$ be a distribution over matrices in $\mathbb{R}^{n \times p}$ such that for any fixed $\mathbf{v} \in \mathbb{R}^p, \epsilon > 0$,

$$\underset{X \sim \mathcal{D}^{n \times p}}{\mathbb{P}} \left[\left| \|X\mathbf{v}\|_2^2 - \|\mathbf{v}\|_2^2 \right| > \epsilon \cdot \|\mathbf{v}\|_2^2 \right] \leq 2\exp(-\Omega(n))$$

Then, for any $k < p/2$, matrices X generated from this distribution also satisfy the RIP property at order k with constant δ with probability at least $1 - 2\exp(-\Omega(n))$ whenever $n \geq \Omega\left(\frac{k}{\delta^2} \log \frac{p}{k}\right)$.

Thus, a distribution over matrices that, for every *fixed* vector, acts as an almost isometry with high probability, is also guaranteed to, with very high probability, generate matrices that act as a restricted isometry *simulataneously* over all sparse vectors. Such matrix distributions

[5] See Exercise 7.5.

are easy to construct – one simply needs to sample each entry of the matrix independently according to one of the following distributions:

1. sample each entry from the Gaussian distribution $\mathcal{N}(0, 1/n)$.
2. set each entry to $\pm 1/\sqrt{n}$ with equal probability.
3. set each entry to 0 w.p. 2/3 and $\pm\sqrt{3/n}$ w.p. 1/6.

The work of Agarwal et al. [2012] shows that the RSC/RSS properties are satisfied whenever rows of the matrix X are drawn from a sub-Gaussian distribution over p-dimensional vectors with a non-singular covariance matrix. This result is useful since it shows that real-life data, which can often be modeled as vectors being drawn from sub-Gaussian distributions, will satisfy these properties with high probability. This is crucial for sparse recovery and other algorithms to be applicable to real life problems such as the gene-expression analysis problem.

If one can tolerate a slight blowup in the number of rows of the matrix X, then there exist better constructions with the added benefit of allowing fast matrix vector products. The initial work of Candès and Tao [2005] itself showed that selecting each row of a Fourier transform matrix independently with probability $\mathcal{O}\left(k\frac{\log^6 p}{p}\right)$ results in an RIP matrix with high probability. More recently, this was improved to $\mathcal{O}\left(k\log^2 k\frac{\log p}{p}\right)$ in the work of Haviv and Regev [2017]. A matrix-vector product of a k-sparse vector with such a matrix takes only $\mathcal{O}\left(k\log^2 p\right)$ time whereas a dense matrix filled with Gaussians would have taken up to $\mathcal{O}\left(k^2\log p\right)$ time. There exist more involved hashing-based constructions that simultaneously offer reduced sample complexity and fast matrix-vector multiplications [Nelson et al., 2014].

Deterministic Designs: There exist far fewer and far weaker results for deterministic constructions of RIP matrices. The initial results in this direction all involved constructing *incoherent* matrices. A matrix $X \in \mathbb{R}^{n\times p}$ with unit norm columns is said to be μ-incoherent if for all $i \neq j \in [p]$, $\langle X_i, X_j\rangle \leq \mu$. A μ-incoherent matrix always satisfies[6] RIP at order k with parameter $\delta = (k-1)\mu$.

[6] See Exercise 7.6.

Deterministic constructions of incoherent matrices with $\mu = \mathcal{O}\left(\frac{\log p}{\sqrt{n}\log n}\right)$ are well known since the work of Kashin [1975]. However, such constructions require $n = \widetilde{\Omega}\left(\frac{k^2 \log^2 p}{\delta^2}\right)$ rows which is quadratically more than what random designs require. The first result to improve upon these constructions came from the work of Bourgain et al. [2011] which gave deterministic combinatorial constructions that assured the RIP property with $n = \widetilde{\mathcal{O}}\left(\frac{k^{(2-\epsilon)}}{\delta^2}\right)$ for some constant $\epsilon > 0$. However, till date, substantially better constructions are not known.

7.7 A Sparse Recovery Guarantee for IHT

We will now establish a convergence result for the IHT algorithm. Although the analysis for the gPGD algorithm (Theorem 3.3) can be adapted here, the following proof is much more tuned to the sparse recovery problem and offers a tighter analysis and several problem-specific insights.

Theorem 7.2. Suppose $X \in \mathbb{R}^{n\times p}$ is a design matrix that satisfies the RIP property of order $3s$ with constant $\delta_{3s} < \frac{1}{2}$. Let $\mathbf{w}^* \in B_0(s) \subset \mathbb{R}^p$ be any arbitrary sparse vector and let $\mathbf{y} = X\mathbf{w}^*$. Then the IHT algorithm (Algorithm 8), when executed with a step length $\eta = 1$, and a projection sparsity level $k = s$, ensures $\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq \epsilon$ after at most $t = \mathcal{O}\left(\log \frac{\|\mathbf{w}^*\|_2}{\epsilon}\right)$ iterations of the algorithm.

Proof. We start off with some notation. Let $S^* := \text{supp}(\mathbf{w}^*)$ and $S^t := \text{supp}(\mathbf{w}^t)$. Let $I^t := S^t \cup S^{t+1} \cup S^*$ denote the union of the supports of the two consecutive iterates and the optimal model. The reason behind defining this quantity is that we are assured that while analyzing this update step, the two *error vectors* $\mathbf{w}^t - \mathbf{w}^*$ and $\mathbf{w}^{t+1} - \mathbf{w}^*$, which will be the focal point of the analysis, have support within I^t. Note that $|I^t| \leq 3s$. Please refer to the notation section at the beginning of this monograph for the interpretation of the notation $\mathbf{x}_I$ and A_I for a vector $\mathbf{x}$, matrix A and set I.

With $\eta = 1$, we have (refer to Algorithm 8), $\mathbf{z}^{t+1} = \mathbf{w}^t - \frac{1}{n}X^\top(X\mathbf{w}^t - \mathbf{y})$. However, due to the (non-convex) projection step

$\mathbf{w}^{t+1} = \Pi_{\mathcal{B}_0(k)}(\mathbf{z}^{t+1})$, applying projection property-O gives us

$$\left\|\mathbf{w}^{t+1} - \mathbf{z}^{t+1}\right\|_2^2 \leq \left\|\mathbf{w}^* - \mathbf{z}^{t+1}\right\|_2^2.$$

Note that none of the other projection properties are applicable here since the set of sparse vectors is a non-convex set. Now, by Pythagoras' theorem, for any vector $\mathbf{v} \in \mathbb{R}^p$, we have $\|\mathbf{v}\|_2^2 = \|\mathbf{v}_I\|_2^2 + \|\mathbf{v}_{\overline{I}}\|_2^2$ which gives us

$$\left\|\mathbf{w}_I^{t+1} - \mathbf{z}_I^{t+1}\right\|_2^2 + \left\|\mathbf{w}_{\overline{I}}^{t+1} - \mathbf{z}_{\overline{I}}^{t+1}\right\|_2^2 \leq \left\|\mathbf{w}_I^* - \mathbf{z}_I^{t+1}\right\|_2^2 + \left\|\mathbf{w}_{\overline{I}}^* - \mathbf{z}_{\overline{I}}^{t+1}\right\|_2^2$$

Now it is easy to see that $\mathbf{w}_{\overline{I}}^{t+1} = \mathbf{w}_{\overline{I}}^* = \mathbf{0}$. Hence we have

$$\left\|\mathbf{w}_I^{t+1} - \mathbf{z}_I^{t+1}\right\|_2 \leq \left\|\mathbf{w}_I^* - \mathbf{z}_I^{t+1}\right\|_2$$

Using the fact that $y = X\mathbf{w}^*$, and denoting $\overline{X} = \frac{1}{\sqrt{n}}X$ we get

$$\left\|\mathbf{w}_I^{t+1} - (\mathbf{w}_I^t - \overline{X}_I^\top \overline{X}(\mathbf{w}^t - \mathbf{w}^*))\right\|_2 \leq \left\|\mathbf{w}_I^* - (\mathbf{w}_I^t - \overline{X}_I^\top \overline{X}(\mathbf{w}^t - \mathbf{w}^*))\right\|_2$$

Adding and subtracting $\mathbf{w}^*$ from the expression inside the norm operator on the LHS, rearranging, and applying the triangle inequality for norms gives us

$$\left\|\mathbf{w}_I^{t+1} - \mathbf{w}_I^*\right\|_2 \leq 2\left\|(\mathbf{w}_I^t - \mathbf{w}_I^*) - \overline{X}_I^\top \overline{X}(\mathbf{w}^t - \mathbf{w}^*)\right\|_2$$

As $\mathbf{w}_I^t = \mathbf{w}^t, \mathbf{w}_I^{t+1} = \mathbf{w}^{t+1}$, observing that $X(\mathbf{w}^t - \mathbf{w}^*) = X_I(\mathbf{w}^t - \mathbf{w}^*)$ gives us

$$\begin{aligned}\left\|\mathbf{w}^{t+1} - \mathbf{w}^*\right\|_2 &\leq 2\left\|(I - \overline{X}_I^\top \overline{X}_I)(\mathbf{w}^t - \mathbf{w}^*)\right\|_2 \\ &\leq 2\left(\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 - \left\|\overline{X}_I^\top \overline{X}_I(\mathbf{w}^t - \mathbf{w}^*)\right\|_2\right) \\ &\leq 2\delta_{3s}\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2,\end{aligned}$$

which finishes the proof. The second inequality above follows due to the triangle inequality and the third inequality follows from the fact that RIP implies[7] that for any $|I| \leq 3s$, the smallest eigenvalue of the matrix $\overline{X}_I^\top \overline{X}_I$ is lower bounded by $(1 - \delta_{3s})$. □

[7] See Exercise 7.7.

We note that this result holds even if the hard thresholding level is set to $k > s$. It is easy to see that the condition $\delta_{3s} < \frac{1}{2}$ is equivalent to the *restricted condition number* (over $3s$-sparse vectors) of the corresponding sparse recovery problem being upper bounded by $\kappa_{3s} < 3$. Similar to Theorem 3.3, here also we require an upper bound on the restricted condition number of the problem. It is interesting to note that a direct application of Theorem 3.3 would have instead required $\delta_{2s} < \frac{1}{3}$ (or equivalently $\kappa_{2s} < 2$) which can be shown to be a harsher requirement than what we have achieved. Moreover, applying Theorem 3.3 would have also required us to set the step length to a specific quantity $\eta = \frac{1}{1+\delta_s}$ while executing the gPGD algorithm whereas while executing the IHT algorithm, we need only set $\eta = 1$.

An alternate proof of this result appears in the work of Garg and Khandekar [2009] which also requires the condition $\delta_{2s} < \frac{1}{3}$. The above result extends to a more general setting where there is additive noise in the model $\mathbf{y} = X\mathbf{w}^* + \boldsymbol{\eta}$. In this setting, it is known (see for example, [Jain et al., 2014, Theorem 3] or [Garg and Khandekar, 2009, Theorem 2.3]) that if the objective function in question (for the sparse recovery problem the objective function is the least squares objective) satisfies the (α, β) RSC/RSS properties at level $2s$, then the following is guaranteed for the output $\widehat{\mathbf{w}}$ of the IHT algorithm (assuming the algorithm is run for roughly $\mathcal{O}(\log n)$ iterations)

$$\left\|\widehat{\mathbf{w}} - \mathbf{w}^*\right\|_2 \leq \frac{3\sqrt{s}}{\alpha}\left\|\frac{X^\top \boldsymbol{\eta}}{n}\right\|_\infty$$

The consistency of the above solution can be verified in several interesting situations. For example, if the design matrix has normalized columns i.e. $\left\|X_i\right\|_2 \leq \sqrt{n}$ and the noise η_i is generated i.i.d. and independently of the design X from some Gaussian distribution $\mathcal{N}(0, \sigma^2)$, then the quantity $\left\|\frac{X^\top \boldsymbol{\eta}}{n}\right\|_\infty$ is of the order of $\sigma\sqrt{\frac{\log p}{n}}$ with high probability. In the above setting IHT guarantees with high probability

$$\left\|\widehat{\mathbf{w}} - \mathbf{w}^*\right\|_2 \leq \widetilde{\mathcal{O}}\left(\frac{\sigma}{\alpha}\sqrt{\frac{s\log p}{n}}\right),$$

i.e. $\left\|\widehat{\mathbf{w}} - \mathbf{w}^*\right\|_2 \to 0$ as $n \to \infty$, thus establishing consistency.

7.8 Other Popular Techniques for Sparse Recovery

The IHT method is a part of a larger class of *hard thresholding techniques*, which include algorithms such as Iterative Hard Thresholding (IHT) [Blumensath, 2011], Gradient Descent with Sparsification (GraDeS) [Garg and Khandekar, 2009], and Hard Thresholding Pursuit (HTP) [Foucart, 2011]. Apart from these gradient descent-style techniques, several other approaches have been developed for the sparse recovery problem over the years. Here we briefly survey them.

7.8.1 Pursuit Techniques

A popular non-convex optimization technique for sparse recovery and a few related optimization problems is that of discovering support elements iteratively. This technique is embodied in pursuit-style algorithms. We warn the reader that the popular *Basis Pursuit* algorithm is actually a convex relaxation technique and not related to the other pursuit algorithms we discuss here. The terminology is a bit confusing but seems to be a matter of legacy.

The pursuit family of algorithms includes Orthogonal Matching Pursuit (OMP) [Tropp and Gilbert, 2007], Orthogonal Matching Pursuit with Replacement (OMPR) [Jain et al., 2011], Compressive Sampling Matching Pursuit (CoSaMP) [Needell and Tropp, 2008], and the Forward-backward (FoBa) algorithm [Zhang, 2011].

Pursuit methods work by gradually discovering the elements in the support of the true model vector $\mathbf{w}^*$. At every time step, these techniques add a new support element to an active support set (which is empty to begin with) and solve a traditional least-squares problem on the active support set. This least-squares problem has no sparsity constraints, and is hence a convex problem which can be solved easily.

The support set is then updated by adding a new support element. It is common to add the coordinate where the gradient of the objective function has the highest magnitude among coordinates not already in the support. FoBa-style techniques augment this method by having *backward* steps where support elements that were erroneously picked earlier are discarded when the error is detected.

Pursuit-style methods are, in general, applicable whenever the structure in the (non-convex) constraint set in question can be represented as a combination of a small number of *atoms*. Examples include sparse recovery, where the atoms are individual coordinates: every s-sparse vector is a linear combination of some s of these atoms.

Other examples include low-rank matrix recovery, which we will study in detail in § 8, where the atoms are rank-one matrices. The SVD theorem tells us that every r-rank matrix can indeed be expressed as a sum of r rank-one matrices. There exist works [Tewari et al., 2011] that give generic methods to perform sparse recovery in such structurally constrained settings.

7.8.2 Convex Relaxation Techniques for Sparse Recovery

Convex relaxation techniques have been extremely popular for the sparse recovery problem. In fact they formed the first line of attack on non-convex optimization problems starting with the seminal work of Candès and Tao [2005], Candès et al. [2006], Donoho [2006] that, for the first time, established polynomial time, globally convergent solutions for the compressive sensing problem.

A flurry of work then followed on relaxation-based techniques [Candès, 2008, Donoho et al., 2009, Foucart, 2010, Negahban et al., 2012] that vastly expanded the scope of the problems being studied, the techniques being applied, as well as their analyses. It is important to note that all methods, whether relaxation based or not, have to assume some design property such as NSP/REP/RIP/RSC-RSS that we discussed earlier, in order to give provable guarantees.

The relaxation approach converts non-convex problems to convex problems first before solving them. This approach, applied to the sparse regression problem, gives us the so-called LASSO problem which has been studied extensively. Consider the sparse recovery problem ((SP-REG)).

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^p\\ \|\mathbf{w}\|_0\leq s}} \|\mathbf{y}-X\mathbf{w}\|_2^2 .$$

Non-convexity arises in the problem due to the non-convex constraint $\|\mathbf{w}\|_0 \leq s$ as the sparsity operator is not a valid norm. The relaxation approach fixes this problem by changing the constraint to use the L_1 norm instead i.e.

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^p\\ \|\mathbf{w}\|_1\leq R}} \|\mathbf{y}-X\mathbf{w}\|_2^2, \tag{LASSO-1}$$

or by using its regularized version instead

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{2n}\|\mathbf{y}-X\mathbf{w}\|_2^2+\lambda_n\|\mathbf{w}\|_1. \tag{LASSO-2}$$

The choice of the L_1 norm is motivated mainly by its convexity as well as formal results that assure us that the relaxation gap is small or non-existent. Both the above formulations ((LASSO-1)) and ((LASSO-2)) are indeed convex but include parameters such as R and λ_n that must be tuned properly to ensure proper convergence. Although the optimization problems ((LASSO-1)) and ((LASSO-2)) are vastly different from ((SP-REG)), a long line of beautiful results, starting from the seminal work of Candès and Tao [2005], Candès et al. [2006], Donoho [2006], showed that if the design matrix X satisfies RIP with appropriate constants, and if the parameters of the relaxations R and λ_n are appropriately tuned, then the solutions to the relaxations are indeed solutions to the original problems as well.

Below we state one such result from the recent text by Hastie et al. [2016]. We recommend this text to any reader looking for a well-curated compendium of techniques and results on the relaxation approach to several non-convex optimization problems arising in machine learning.

Theorem 7.3. [Hastie et al., 2016, Theorem 11.1] Consider a sparse recovery problem $\mathbf{y}=X\mathbf{w}^*+\boldsymbol{\eta}$ where the model $\mathbf{w}^*$ is s-sparse and the design matrix X satisfies the restricted-eigenvalue condition (see Definition 7.2) of the order s with constant α, then the following hold

1. Any solution $\widehat{\mathbf{w}}_1$ to ((LASSO-1)) with $R=\|\mathbf{w}^*\|_1$ satisfies

$$\|\widehat{\mathbf{w}}_1-\mathbf{w}^*\|_2\leq\frac{4}{\alpha}\sqrt{s}\left\|\frac{X^\top\boldsymbol{\eta}}{n}\right\|_\infty.$$

2. Any solution $\widehat{\mathbf{w}}_2$ to ((LASSO-2)) with $\lambda_n\geq 2\left\|X^\top\boldsymbol{\eta}/n\right\|_\infty$ satisfies

$$\|\widehat{\mathbf{w}}_1-\mathbf{w}^*\|_2\leq\frac{3}{\alpha}\sqrt{s}\lambda_n.$$

The reader can verify that the above bounds are competitive with the bounds for the IHT algorithm that we discussed previously. We refer the reader to [Hastie et al., 2016, Chapter 11] for more consistency results for the LASSO formulations.

7.8.3 Non-convex Regularization Techniques

Instead of performing a complete convex relaxation of the problem, there exist approaches that only partly relax the problem. A popular approach in this direction uses L_q regularization with $0 < q < 1$ [Chartrand, 2007, Foucart and Lai, 2009, Wang et al., 2011]. The resulting problem still remains non-convex but becomes a little well behaved in terms of having objective functions that are almost-everywhere differentiable. For instance, the following optimization problem may be used to solve the noiseless sparse recovery problem.

$$\begin{aligned}\min_{\mathbf{w}\in\mathbb{R}^p} \ & \left\|\mathbf{w}\right\|_q, \\ \text{s.t. } & \mathbf{y} = X\mathbf{w}\end{aligned}$$

For noisy settings, one may replace the constraint with a soft constraint such as $\left\|\mathbf{y} - X\mathbf{w}\right\|_2 \leq \epsilon$, or else move to an unconstrained version like LASSO with the L_1 norm replaced by the L_q norm. The choice of the regularization norm q is dictated by application and usually any value within a certain range within the interval $(0, 1)$ can be chosen.

There has been interest in characterizing both the global and the local optima of these optimization problems for their recovery properties [Chen and Gu, 2015]. In general, L_q regularized formulations, if solved exactly, can guarantee recovery under much weaker conditions than what LASSO formulations, and IHT require. For instance, the RIP condition that L_q-regularized formulations need in order to guarantee universal recovery can be as weak as $\delta_{2k+1} < 1$ [Chartrand, 2007]. This is very close to the requirement $\delta_{2k} < 1$ that must be made by any algorithm in order to ensure that the solution even be unique. However, solving these non-convex regularized problems at large scale itself remains challenging and an active area of research.

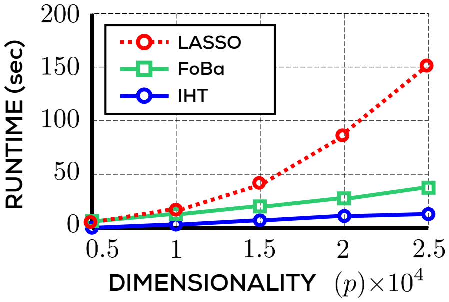


Figure 7.2: An empirical comparison of run-times offered by the LASSO, FoBA and IHT methods on sparse regression problems with varying dimensionality p. All problems enjoyed sparsity $s = 100$ and were offered $n = 2s \cdot \log p$ data points. IHT is clearly the most scalable of the methods followed by FoBa. The relaxation technique does not scale very well to high dimensions. Figure adapted from [Jain et al., 2014].

7.8.4 Empirical Comparison

To give the reader an appreciation of the empirical performance of the various methods we have discussed for sparse recovery, Figure 7.2 provides a comparison of some of these methods on a synthetic sparse regression problem. The graph plots the running time taken by the various methods to solve the same sparse linear regression problem with sparsity $s = 100$ but with dimensionalities increasing from $p = 5000$ to $p = 25000$. The graph indicates that non-convex optimization methods such as IHT and FoBa are far more scalable than relaxation-based methods. It should be noted that although pursuit-style techniques are scalable, they can become sluggish if the true support set size s is not very small since these techniques discover support elements one by one.

7.9 Extensions

In the preceding discussion, we studied the problem of sparse linear regression and the IHT technique to solve the problem. These basic results can be augmented and generalized in several ways. The work of Negahban et al. [2012] greatly expanded the scope of sparse recovery techniques beyond simple least-squares to the more general M-estimation problem. The work of Bhatia et al. [2015] offered solutions to the *robust* sparse regression problem where the responses may be corrupted by an adversary. We will explore the robust regression problem in more detail in § 9. We discuss a few more such extensions below.

7.9.1 Sparse Recovery in Ill-Conditioned Settings

As we discussed before, the bound on the RIP constant $\delta_{3s} < \frac{1}{2}$ as required by Theorem 7.2, effectively places a bound on the *restricted condition number* κ_{3s} of the design matrix. In our case the bound translates to $\kappa_{3s} = \frac{1+\delta_{3s}}{1-\delta_{3s}} < 3$. However, in cases such as the gene expression analysis problem where the design matrix is not totally under our control, the restricted condition number might be much larger than 3.

For instance, it can be shown that if the expression levels of two genes are highly correlated then this results in ill-conditioned design matrices. In such settings, it is much more appropriate to assume that the design matrix satisfies restricted strong convexity and smoothness (RSC/RSS) which allows us to work with design matrices with arbitrarily large condition numbers. It turns out that the IHT algorithm can be modified [see for example, Jain et al., 2014] to work in these ill-conditioned recovery settings.

Theorem 7.4. Suppose $X \in \mathbb{R}^{n \times p}$ is a design matrix that satisfies the restricted strong convexity and smoothness property of order $2k + s$ with constants α_{2k+s} and β_{2k+s} respectively. Let $\mathbf{w}^* \in B_0(s) \subset \mathbb{R}^p$ be any arbitrary sparse vector and let $\mathbf{y} = X\mathbf{w}^*$. Then the IHT algorithm, when executed with a step length $\eta < \frac{2}{\beta_{2k+s}}$, and a projection sparsity level $k \geq 32\left(\frac{\beta_{2k+s}}{\alpha_{2k+s}}\right)^2 s$, ensures $\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq \epsilon$ after $t = \mathcal{O}\left(\frac{\beta_{2k+s}}{\alpha_{2k+s}} \log \frac{\|\mathbf{w}^*\|_2}{\epsilon}\right)$ iterations of the algorithm.

Note that the above result does not place any restrictions on the condition number or the RSC/RSS constants of the problem. The result also mimics Theorem 3.3 in its dependence on the (restricted) condition number of the optimization problem i.e. $\kappa_{2k+s} = \frac{\beta_{2k+s}}{\alpha_{2k+s}}$. The proof of this result is a bit tedious, hence omitted.

7.9.2 Recovery from a Union of Subspaces

If we look closely, the set $\mathcal{B}_0(s)$ is simply a union of $\binom{p}{s}$ linear subspaces, each subspace encoding a specific sparsity pattern. It is natural to wonder whether the methods and analyses described above also hold when the vector to be recovered belongs to a general union of subspaces. More specifically, consider a family of linear subspaces $\mathcal{H}_1, \ldots, \mathcal{H}_L \subset \mathbb{R}^p$ and denote the union of these subspaces by $\mathcal{H} = \bigcup_{i=1}^{L} \mathcal{H}_i$. The restricted strong convexity and restricted strong smoothness conditions can be appropriately modified to suit this setting by requiring a design matrix $X : \mathbb{R}^p \to \mathbb{R}^n$ to satisfy, for every $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{H}$,

$$\alpha \cdot \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2 \leq \|X(\mathbf{w}_1 - \mathbf{w}_2)\|_2^2 \leq L \cdot \|\mathbf{w}_1 - \mathbf{w}_2\|_2^2$$

It turns out that IHT, with an appropriately modified projection operator $\Pi_{\mathcal{H}}(\cdot)$, can ensure recovery of vectors that are guaranteed to reside in a small union of low-dimensional subspaces. Moreover, a linear rate of convergence, as we have seen for the IHT algorithm in the sparse regression case, can still be achieved. We refer the reader to the work of Blumensath [2011] for more details of this extension.

7.9.3 Dictionary Learning

A very useful extension of sparse recovery, or sparse *coding* emerges when we attempt to learn the design matrix as well. Thus, all we are given are observations $\mathbf{y}_1, \ldots, \mathbf{y}_m \in \mathbb{R}^n$ and we wish to learn a design matrix $X \in \mathbb{R}^{n \times p}$ such that the observations $\mathbf{y}_i$ can be represented as sparse combinations $\mathbf{w}_i \in \mathbb{R}^p$ of the columns of the design matrix i.e. $\mathbf{y}_i \approx X\mathbf{w}_i$ such that $\|\mathbf{w}_i\|_0 \leq s \ll p$. The problem has several applications in the fields of computer vision and signal processing and has seen a lot of interest in the recent past.

The alternating minimization technique where one alternates between estimating the design matrix and the sparse representations, is especially popular for this problem. Methods mostly differ in the exact implementation of these alternations. Some notable works in this area include [Agarwal et al., 2016, Arora et al., 2014, Gribonval et al., 2015, Spielman et al., 2012].

7.10 Exercises

Exercise 7.1. Suppose a design matrix $X \in \mathbb{R}^{n \times p}$ satisfies $X\mathbf{w}_1 = X\mathbf{w}_2$ for some $\mathbf{w}_1 \neq \mathbf{w}_2 \in \mathbb{R}^p$. Then show that there exists an entire subspace $\mathcal{H} \subset \mathbb{R}^p$ such that for all $\mathbf{w}, \mathbf{w}' \in \mathcal{H}$, we have $X\mathbf{w} = X\mathbf{w}'$.

Exercise 7.2. Show that the set $\mathcal{C}(k) := \bigcup_{S:|S|=k} \mathcal{C}(S)$ is non-convex.

Exercise 7.3. Show that if a design matrix X satisfies the null-space property of order $2s$, then for any two distinct s-sparse vectors $\mathbf{v}^1, \mathbf{v}^2 \in \mathcal{B}_0(s)$, $\mathbf{v}^1 \neq \mathbf{v}^2$, it must be the case that $X\mathbf{v}^1 \neq X\mathbf{v}^2$.

Exercise 7.4. Show that the RSC/RSS notion introduced in Definition 7.4 is equivalent to the RSC/RSS notion in Definition 3.2 defined in § 3 for an appropriate choice of function and constraint sets.

Exercise 7.5. Show that RSC-RSS ⇒ REP ⇒ NSP i.e. a matrix that satisfies the RSC/RSS condition for some constants, must satisfy the REP condition for some constants which in turn must force it to satisfy the null-space property.

Exercise 7.6. Show that every μ-incoherent matrix satisfies the RIP property at order k with parameter $\delta = (k-1)\mu$.

Exercise 7.7. Suppose the matrix $X \in \mathbb{R}^{n \times p}$ satisfies RIP at order s with constant δ_s. Then show that for any set $I \subset [p], |I| \leq s$, the smallest eigenvalue of the matrix $X_I^\top X_I$ is lower bounded by $(1 - \delta_s)$.

Exercise 7.8. Show that the RIP constant is monotonic in its order i.e. if a matrix X satisfies RIP of order k with constant δ_k, then it also satisfies RIP for all orders $k' \leq k$ with $\delta_{k'} \leq \delta_k$.

7.11 Bibliographic Notes

The literature on sparse recovery techniques is too vast for this note to cover. We have already covered several directions in §§ 7.8 and 7.9 and point the reader to references therein.

8

Low-rank Matrix Recovery

In this section, we will look at the problem of low-rank matrix recovery in detail. Although simple to motivate as an extension of the sparse recovery problem that we studied in § 7, the problem rapidly distinguishes itself in requiring specific tools, both algorithmic and analytic. We will start our discussion with a milder version of the problem as a warm up and move on to the problem of low-rank matrix completion which is an active area of research.

8.1 Motivating Applications

We will take the following two running examples to motivate the problem of low-rank matrix recovery.

Collaborative Filtering Recommendation systems are popularly used to model the preference patterns of users, say at an e-commerce website, for items being sold on that website, although the principle of recommendation extends to several other domains that demand *personalization* such as education and healthcare. Collaborative filtering is a popular technique for building recommendation systems.

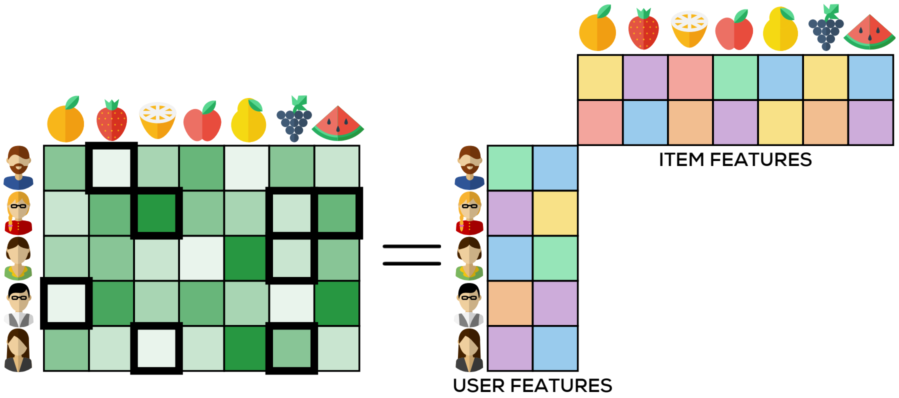


Figure 8.1: In a typical recommendation system, users rate items very infrequently and certain items may not get rated even once. The figure depicts a ratings matrix. Only the matrix entries with a bold border are observed. Low-rank matrix completion can help recover the unobserved entries, as well as reveal hidden features that are descriptive of user and item properties, as shown on the right hand side.

The collaborative filtering approach seeks to exploit co-occurring patterns in the observed behavior across users in order to predict future user behavior. This approach has proven successful in addressing users that interact very sparingly with the system. Consider a set of m users $u_1, \ldots, u_m$, and n items $a_1, \ldots, a_n$. Our goal is to predict the preference score $s_{(i,j)}$ that is indicative of the interest user u_i has in item a_j.

However, we get direct access to (noisy estimates of) actual preference scores for only a few items per user by looking at clicks, purchases etc. That is to say, if we consider the $m \times n$ *preference matrix* $A = [A_{ij}]$ where $A_{ij} = s_{(i,j)}$ encodes the (true) preference of the i^{th} user for the j^{th} item, we get to see only $k \ll m \cdot n$ entries of A, as depicted in Figure 8.1. Our goal is to recover the remaining entries.

The problem of paucity of available data is readily apparent in this setting. In its nascent form, the problem is not even well posed and does not admit a unique solution. A popular way of overcoming these problems is to assume a low-rank structure in the preference matrix.

As we saw in Exercise 3.3, this is equivalent to assuming that there is an r-dimensional vector $\mathbf{u}_i$ denoting the i^{th} user and an r-dimensional vector $\mathbf{a}_j$ denoting the j^{th} such that $s_{(i,j)} \approx \langle \mathbf{u}_i, \mathbf{a}_j \rangle$. Thus, if $\Omega \subset$

$[m] \times [n]$ is the set of entries that have been observed by us, then the problem of recovering the unobserved entries can be cast as the following optimization problem:

$$\min_{\substack{X \in \mathbb{R}^{m \times n} \\ \text{rank}(X) \leq r}} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 .$$

This problem can be shown to be NP-hard [Hardt et al., 2014] but has generated an enormous amount of interest across communities. We shall give special emphasis to this *matrix completion* problem in the second part of this section.

Linear Time-invariant Systems Linear Time-invariant (LTI) systems are widely used in modeling dynamical systems in fields such as engineering and finance. The *response behavior* of these systems is characterized by a model vector $\mathbf{h} = [h(0), h(1), \ldots, h(2N-1)]$. The *order* of such a system is given by the rank of the following Hankel matrix

$$\text{hank}(\mathbf{h}) = \begin{bmatrix} h(0) & h(1) & \ldots & h(N) \\ h(1) & h(2) & \ldots & h(N+1) \\ \vdots & \vdots & \ddots & \vdots \\ h(N-1) & h(N) & \ldots & h(2N-1) \end{bmatrix}$$

Given a sequence of inputs $\mathbf{a} = [a(1), a(2), \ldots, a(N)]$ to the system, the output of the system is given by

$$y(N) = \sum_{t=0}^{N-1} a(N-t)h(t)$$

In order to recover the model parameters of a system, we repeatedly apply i.i.d. Gaussian impulses $a(i)$ to the system for N time steps and then observe the output of the system. This process is repeated, say k times, to yield observation pairs $\left\{(\mathbf{a}^i, y^i)\right\}_{i=1}^{k}$. Our goal now, is to take these observations and identify an LTI vector $\mathbf{h}$ that best fits the data. However, for the sake of accuracy and ease of analysis [Fazel et al., 2013], it is advisable to fit a low-order model to the data. Let the matrix $A \in \mathbb{R}^{k \times N}$ contain the i.i.d. Gaussian impulses applied to the system. Then the problem of fitting a low-order model can be shown to reduce to the following constrained optimization problem with a rank objective and an affine constraint.

$$\begin{array}{ll} \min & \operatorname{rank}(\operatorname{hank}(\mathbf{h})) \\ \text{s.t.} & A\mathbf{h} = \mathbf{y}, \end{array}$$

The above problem is a non-convex optimization problem due to the objective being the minimization of the rank of a matrix. Several other problems in metric embedding and multi-dimensional scaling, image compression, low rank kernel learning and spatio-temporal imaging can also be reduced to low rank matrix recovery problems [Jain et al., 2010, Recht et al., 2010].

8.2 Problem Formulation

The two problems considered above can actually be cast in a single problem formulation, that of *Affine Rank Minimization* (ARM). Consider a low rank matrix $X^* \in \mathbb{R}^{m \times n}$ that we wish to recover and an affine transformation $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^k$. The transformation can be seen as a concatenation of k real valued affine transformations $\mathcal{A}_i : \mathbb{R}^{m \times n} \to \mathbb{R}$. We are given the transformation $\mathcal{A}$, and its (possibly noisy) action $\mathbf{y} = \mathcal{A}(X^*) \in \mathbb{R}^k$ on the matrix X^* and our goal is to recover this matrix by solving the following optimization problem.

$$\begin{array}{ll} \min & \operatorname{rank}(X) \\ \text{s.t.} & \mathcal{A}(X) = \mathbf{y}, \end{array} \tag{ARM}$$

This problem can be shown to be NP-hard due to a reduction to the sparse recovery problem[1]. The LTI modeling problem can be easily seen to be an instance of ARM with the Gaussian impulses being delivered to the system resulting in a k-dimensional affine transformation of the Hankel matrix corresponding to the system. However, the Collaborative Filtering problem is also an instance of ARM. To see this, for any $(i, j) \in [m] \times [n]$, let $O^{(i,j)} \in \mathbb{R}^{m \times n}$ be the matrix such that its (i, j)-th entry $O^{(i,j)}_{ij} = 1$ and all other entries are zero. Then, simply define the affine transformation $\mathcal{A}_{(i,j)} : X \mapsto \operatorname{tr}(X^\top O^{(i,j)}) = X_{ij}$. Thus, if we

[1] See Exercise 8.1.

observe k user-item ratings, the ARM problem effectively operates with a k-dimensional affine transformation of the underlying rating matrix.

Due to its similarity to the sparse recovery problem, we will first discuss the general ARM problem. However, we will find it beneficial to cast the collaborative filtering problem as a *Low-rank Matrix Completion* problem instead. In this problem, we have an underlying low rank matrix X^* of which, we observe entries in a set $\Omega \subset [m] \times [n]$. Then the low rank matrix completion problem can be stated as

$$\min_{\substack{X \in \mathbb{R}^{m \times n} \\ \text{rank}(X) \leq r}} \quad \|\Pi_\Omega(X - X^*)\|_F^2 \, , \tag{LRMC}$$

where $\Pi_\Omega(X)$ is defined, for any matrix X as

$$\Pi_\Omega(X)_{i,j} = \begin{cases} X_{i,j} & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

The above formulation succinctly captures our objective to find a *completion* of the ratings matrix that is both, low rank, as well as agrees on the user ratings that are actually observed. As pointed out earlier, this problem is NP-hard [Hardt et al., 2014].

Before moving on to present algorithms for the ARM and LRMC problems, we discuss some matrix design properties that would be required in the convergence analyses of the algorithms.

8.3 Matrix Design Properties

Similar to sparse recovery, there exist design properties that ensure that the general NP-hardness of the ARM and LRMC problems can be overcome in well-behaved problem settings. In fact given the similarity between ARM and sparse recovery problems, it is tempting to try and import concepts such as RIP into the matrix-recovery setting.

In fact this is exactly the first line of attack that was adopted in literature. What followed was a beautiful body of work that generalized, both structural notions such as RIP, as well as algorithmic techniques such as IHT, to address the ARM problem. Given the generality of these constructs, as well as the smooth transition it offers having studied sparse recovery, we feel compelled to present them to the reader.

8.3.1 The Matrix Restricted Isometry Property

The generalization of RIP to matrix settings, referred to as matrix RIP, follows in a relatively straightforward manner and was first elucidated by Recht et al. [2010]. Quite in line with the sparse recovery setting, the intuition dictates that recovery should be possible only if the affine transformation does not identify two distinct low-rank matrices. A more robust version dictates that no low-rank matrix should be distorted significantly by this transformation which gives us the following.

Definition 8.1 (Matrix Restricted Isometry Property [Recht et al., 2010])**.** A linear map $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^k$ is said to satisfy the matrix restricted isometry property of order r with constant $\delta_r \in [0, 1)$ if for all matrices X of rank at most r, we have

$$(1 - \delta_r) \cdot \|X\|_F^2 \leq \|\mathcal{A}(X)\|_2^2 \leq (1 + \delta_r) \cdot \|X\|_F^2.$$

Furthermore, the work of Recht et al. [2010] also showed that linear maps or affine transformations arising in random measurement models, such as those in image compression and LTI systems, do satisfy RIP with requisite constants whenever the number of affine measurements satisfies $k = \mathcal{O}(nr)$ [Oymak et al., 2015]. Note however, that these are settings in which the design of the affine map is within our control. For settings, where the restricted condition number of the affine map is not within our control, more involved analysis is required. The bibliographic notes point to some of these results.

Given the relatively simple extension of the RIP definitions to the matrix setting, it is all the more tempting to attempt to apply gPGD-style techniques to solve the ARM problem, particularly since we saw how IHT succeeded in offering scalable solutions to the sparse recovery problem. The works of [Goldfarb and Ma, 2011, Jain et al., 2010] showed that this is indeed possible. We will explore this shortly.

8.3.2 The Matrix Incoherence Property

We begin this discussion by warning the reader that there are two distinct notions prevalent in literature, both of which are given the same name, that of the matrix incoherence property. The first of these notions was introduced in § 7.6 as a property that can be used to ensure the RIP property in matrices. However a different property, but bearing the same name, finds application in matrix completion problems which we now introduce. We note that the two properties are not known to be strongly related in a formal sense and the coincidental clash of the names seems to be a matter of legacy.

Nevertheless, the intuition behind the second notion of matrix incoherence is similar to that for RIP in that it seeks to make the problem well posed. Consider the matrix $A = \sum_{t=1}^{r} s_t \cdot \mathbf{e}_{i_t} \bar{\mathbf{e}}_{j_t}^\top \in \mathbb{R}^{m \times n}$ where $\mathbf{e}_i$ are the canonical orthonormal vectors in $\mathbb{R}^m$ and $\bar{\mathbf{e}}_j$ are the canonical orthonormal vectors in $\mathbb{R}^n$. Clearly A has rank at most r.

However, this matrix A is non-zero only at r locations. Thus, it is impossible to recover the entire matrix uniquely unless these very r locations $\{(i_t, j_t)\}_{t=1,\ldots,r}$ are actually observed. Since in recommendation settings, we only observe a few random entries of the matrix, there is a good possibility that none of these entries will ever be observed. This presents a serious challenge for the matrix completion problem – the low rank structure is not sufficient to ensure unique recovery!

To overcome this and make the LRMC problem well posed with a unique solution, an additional property is imposed. This so-called matrix incoherence property prohibits low rank matrices that are also sparse. A side effect of this imposition is that for incoherent matrices, observing a small random set of entries is enough to uniquely determine the unobserved entries of the matrix.

Definition 8.2 (Matrix Incoherence Property [Candès and Recht, 2009])**.** A matrix $A \in \mathbb{R}^{m \times n}$ of rank r is said to be incoherent with parameter μ if its left and right singular matrices have bounded row norms. More specifically, let $A = U\Sigma V^\top$ be the SVD of A. Then μ-incoherence dictates that $\left\|U^i\right\|_2 \leq \frac{\mu\sqrt{r}}{\sqrt{m}}$ for all $i \in [m]$ and $\left\|V^j\right\|_2 \leq \frac{\mu\sqrt{r}}{\sqrt{n}}$ for all $j \in [n]$. A stricter version of this property requires all entries of U to satisfy $|U_{ij}| \leq \frac{\mu}{\sqrt{m}}$ and all entries of V to satisfy $|V_{ij}| \leq \frac{\mu}{\sqrt{n}}$.

A low rank incoherent matrix is guaranteed to be *far*, i.e., well distinguished, from any sparse matrix, something that is exploited by algorithms to give guarantees for the LRMC problem.

Algorithm 9 Singular Value Projection (SVP)

```
Input: Linear map $\mathcal{A}$, measurements $\mathbf{y}$, target rank $q$, step length $\eta$
Output: A matrix $\widehat{X}$ with rank at most $q$
$X^1 \leftarrow \mathbf{0}^{m \times n}$
for $t = 1, 2, \ldots$ do
    $Y^{t+1} \leftarrow X^t - \eta \cdot \mathcal{A}^\top(\mathcal{A}(X^t) - \mathbf{y})$
    Compute top $q$ singular vectors/values of $Y^{t+1}$: $U_q^t, \Sigma_q^t, V_q^t$
    $X^{t+1} \leftarrow U_q^t \Sigma_q^t (V_q^t)^\top$
end for
return $X^t$
```

8.4 Low-rank Matrix Recovery via Proj. Gradient Descent

We will now apply the gPGD algorithm to the ARM problem. To do so, first consider the following reformulation of the ARM problem to make it more compatible to the projected gradient descent iterations.

$$\begin{array}{rl} \min & \frac{1}{2}\left\|\mathcal{A}(X) - \mathbf{y}\right\|_2^2 \\ \text{s.t.} & \operatorname{rank}(X) \leq r \end{array} \tag{ARM-2}$$

Applying the gPGD algorithm to the above formulation gives us the *Singular Value Projection* (SVP) algorithm (Algorithm 9). Note that in this case, the projection needs to be carried out onto the set of low rank matrices. However, as we saw in § 3.1, this can be efficiently done by computing the singular value decomposition of the iterates.

SVP offers ease of implementation and speed similar to IHT. Moreover, it applies to ARM problems in general. If the Matrix RIP property is appropriately satisfied, then SVP guarantees a linear rate of convergence to the optimum, much like IHT. All these make SVP a very attractive choice for solving low rank matrix recovery problems.

Below, we give a convergence proof for SVP in the noiseless case, i.e., when $\mathbf{y} = \mathcal{A}(X^*)$. The proof is similar in spirit to the convergence proof we saw for the IHT algorithm in § 7 but differs in crucial aspects since sparsity in this case is apparent not in the signal domain (the matrix is not itself sparse) but the spectral domain (the set of singular values of the matrix is sparse). The analysis can be extended to noisy measurements as well and can be found in [Jain et al., 2010].

8.5 A Low-rank Matrix Recovery Guarantee for SVP

We will now present a convergence result for the SVP algorithm. As before, although the general convergence result for gPGD can indeed be applied here, we will see, just as we did in § 7, that the problem specific analysis we present here is finer and reveals more insights about the ARM problem structure.

Theorem 8.1. Suppose $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^k$ is an affine transformation that satisfies the matrix RIP property of order $2r$ with constant $\delta_{2r} \leq \frac{1}{3}$. Let $X^* \in \mathbb{R}^{m\times n}$ be a matrix of rank at most r and let $\mathbf{y} = \mathcal{A}(X^*)$. Then the SVP Algorithm (Algorithm 9), when executed with a step length $\eta = 1/(1+\delta_{2r})$, and a target rank $q = r$, ensures $\left\|X^t - X^*\right\|_F^2 \leq \epsilon$ after $t = \mathcal{O}\left(\log \frac{\|\mathbf{y}\|_2^2}{2\epsilon}\right)$ iterations of the algorithm.

Proof. Notice that the notions of *sparsity* and *support* are very different in ARM than what they were for sparse regression. Consequently, the exact convergence proof for IHT (Theorem 7.2) is not applicable here. We will first establish an intermediate result that will show, that after $t = \mathcal{O}\left(\log \frac{\|\mathbf{y}\|_2^2}{2\epsilon}\right)$ iterations, SVP ensures $\left\|\mathcal{A}(X^t) - \mathbf{y}\right\|_2^2 \leq \epsilon$. We will then use the matrix RIP property (Definition 8.1) to deduce

$$\left\|\mathcal{A}(X^t) - \mathbf{y}\right\|_2^2 = \left\|\mathcal{A}(X^t - X^*)\right\|_2^2 \geq (1-\delta_{2r}) \cdot \left\|X^t - X^*\right\|_F^2,$$

which will conclude the proof. To prove this intermediate result, let us denote the objective function as

$$f(X) = \frac{1}{2}\left\|\mathcal{A}(X) - \mathbf{y}\right\|_2^2 = \frac{1}{2}\left\|\mathcal{A}(X - X^*)\right\|_2^2.$$

An application of the matrix RIP property then gives us

$$\begin{aligned} &f(X^{t+1}) \\ &= f(X^t) + \left\langle \mathcal{A}(X^t - X^*), \mathcal{A}(X^{t+1} - X^t)\right\rangle + \frac{1}{2}\left\|\mathcal{A}(X^{t+1} - X^t)\right\|_2^2 \\ &\leq f(X^t) + \left\langle \mathcal{A}(X^t - X^*), \mathcal{A}(X^{t+1} - X^t)\right\rangle + \frac{(1+\delta_{2r})}{2}\left\|X^{t+1} - X^t\right\|_F^2. \end{aligned}$$

The following steps now introduce the intermediate variable Y^{t+1} into the analysis in order to link the successive iterates by using the fact that X^{t+1} was the result of a non-convex projection operation.

$$\begin{aligned}
&\left\langle \mathcal{A}(X^t - X^*), \mathcal{A}(X^{t+1} - X^t)\right\rangle + \frac{(1+\delta_{2r})}{2} \cdot \left\| X^{t+1} - X^t \right\|_F^2 \\
&\quad = \frac{1+\delta_{2r}}{2} \cdot \left\| X^{t+1} - Y^{t+1} \right\|_F^2 - \frac{1}{2(1+\delta_{2r})} \cdot \left\| \mathcal{A}^\top(\mathcal{A}(X^t - X^*)) \right\|_F^2 \\
&\quad \leq \frac{1+\delta_{2r}}{2} \cdot \left\| X^* - Y^{t+1} \right\|_F^2 - \frac{1}{2(1+\delta_{2r})} \cdot \left\| \mathcal{A}^\top(\mathcal{A}(X^t - X^*)) \right\|_F^2 \\
&\quad = \left\langle \mathcal{A}(X^t - X^*), \mathcal{A}(X^* - X^t)\right\rangle + \frac{(1+\delta_{2r})}{2} \cdot \left\| X^* - X^t \right\|_F^2 \\
&\quad \leq \left\langle \mathcal{A}(X^t - X^*), \mathcal{A}(X^* - X^t)\right\rangle + \frac{(1+\delta_{2r})}{2(1-\delta_{2r})} \cdot \left\| \mathcal{A}(X^* - X^t) \right\|_2^2 \\
&\quad = -f(X^t) - \frac{1}{2}\left\| \mathcal{A}(X^* - X^t) \right\|_2^2 + \frac{(1+\delta_{2r})}{2(1-\delta_{2r})} \cdot \left\| \mathcal{A}(X^* - X^t) \right\|_2^2.
\end{aligned}$$

The first step uses the identity $Y^{t+1} = X^t - \eta \cdot \mathcal{A}^\top(\mathcal{A}(X^t) - \mathbf{y})$ from Algorithm 9, the fact that we set $\eta = \frac{1}{1+\delta_{2r}}$, and elementary rearrangements. The second step follows from the fact that $\left\|X^{t+1} - Y^{t+1}\right\|_F^2 \leq \left\|X^* - Y^{t+1}\right\|_F^2$ by virtue of the SVD step which makes X^{t+1} the best rank-$(2r)$ approximation to Y^{t+1} in terms of the Frobenius norm. The third step simply rearranges things in the reverse order of the way they were arranged in the first step, the fourth step uses the matrix RIP property and the fifth step makes elementary manipulations. This, upon rearrangement, and using $\left\|\mathcal{A}(X^t - X^*)\right\|_2^2 = 2f(X^t)$, gives us

$$f(X^{t+1}) \leq \frac{2\delta_{2r}}{1-\delta_{2r}} \cdot f(X^t).$$

Since $f(X^1) = \|\mathbf{y}\|_2^2$, as we set $X^1 = \mathbf{0}^{m\times n}$, if $\delta_{2r} < 1/3$ (i.e., $\frac{2\delta_{2r}}{1-\delta_{2r}} < 1$), we have the claimed convergence result. □

One can, in principle, apply the SVP technique to the matrix completion problem as well. However, on the LMRC problem, SVP is outperformed by gAM-style approaches which we study next. Although the superior performance of gAM on the LMRC problem was well documented empirically, it took some time before a theoretical understanding could be obtained. This was first done in the works of Keshavan [2012], Jain et al. [2013]. These results set off a long line of works that progressively improved both the algorithm, as well as its analysis.

Algorithm 10 AltMin for Matrix Completion (AM-MC)

```
Input: Matrix $A \in \mathbb{R}^{m \times n}$ of rank $r$ observed at entries in the set $\Omega$,
    sampling probability $p$, stopping time $T$
Output: A matrix $\widehat{X}$ with rank at most $r$
Partition $\Omega$ into $2T+1$ sets $\Omega_0, \Omega_1, \ldots, \Omega_{2T}$ uniformly and randomly
$U^1 \leftarrow \text{SVD}(\frac{1}{p}\Pi_{\Omega_0}(A), r)$, the top $r$ left singular vectors of $\frac{1}{p}\Pi_{\Omega_0}(A)$
for $t = 1, 2, \ldots, T$ do
    $V^{t+1} \leftarrow \arg\min_{V \in \mathbb{R}^{n \times r}} \left\|\Pi_{\Omega_t}(U^t V^\top - A)\right\|_F^2$
    $U^{t+1} \leftarrow \arg\min_{U \in \mathbb{R}^{m \times r}} \left\|\Pi_{\Omega_{T+t}}(U(V^{t+1})^\top - A)\right\|_F^2$
end for
return $U^\top (V^\top)^\top$
```

8.6 Matrix Completion via Alternating Minimization

We will now look at the alternating minimization technique for solving the low-rank matrix completion problem. As we have observed before, the LRMC problem admits an equivalent reformulation where the low rank structure constraint is eliminated and instead, the solution is described in terms of two low-rank components

$$\min_{\substack{U \in \mathbb{R}^{m \times k} \\ V \in \mathbb{R}^{n \times k}}} \left\|\Pi_\Omega(UV^\top - X^*)\right\|_F^2. \tag{LRMC*}$$

In this case, fixing either U or V reduces the above problem to a simple least squares problem for which we have very efficient and scalable solvers. As we saw in § 4, such problems are excellent candidates for the gAM algorithm to be applied. The AM-MC algorithm (see Algorithm 10) applies the gAM approach to the reformulated LMRC problem. The AM-MC approach is the choice of practitioners in the context of collaborative filtering [Chen and He, 2012, Koren et al., 2009, Zhou et al., 2008]. However, AM-MC, like other gAM-style algorithms, does require proper initialization and tuning.

8.7 A Low-rank Matrix Completion Guarantee for AM-MC

We will now analyze the convergence properties of the AM-MC method for matrix completion. To simplify the presentation, we will restrict ourselves to the case when the matrix $A \in \mathbb{R}^{m\times n}$ is rank one. This will allow us to present the essential arguments without getting involved with technical details. Let $A = \mathbf{u}^*(\mathbf{v}^*)^\top$ be a μ-incoherent matrix that needs to be recovered, where $\left\|\mathbf{u}^*\right\|_2 = \left\|\mathbf{v}^*\right\|_2 = 1$. It is easy to see that μ-incoherence implies $\left\|\mathbf{u}^*\right\|_\infty \leq \frac{\mu}{\sqrt{m}}$ and $\left\|\mathbf{v}^*\right\|_\infty \leq \frac{\mu}{\sqrt{n}}$.

We will also assume the Bernoulli sampling model i.e., that the set of observed indices Ω is generated by selecting each entry (i, j) for inclusion in Ω in an i.i.d. fashion with probability p. More specifically, $(i, j) \in \Omega$ iff $\delta_{ij} = 1$ where $\delta_{ij} = 1$ with probability p and 0 otherwise.

For simplicity, we will assume that each iteration of the AM-MC procedure receives a fresh set of samples Ω from A. This can be achieved in practice by randomly partitioning the available set of samples into as many groups as the iterations of the procedure. The completion guarantee will proceed in two steps. In the first step, we will show that the initialization step in Algorithm 10 itself brings AM-MC within a constant distance of the optimal solution. Next, we will show that this close initialization is sufficient for AM-MC to ensure a linear rate of convergence to the optimal solution.

Initialization: We will now show that the initialization step (Step 2 in Algorithm 10) provides a point $(\mathbf{u}^1, \mathbf{v}^1)$ which is at most a constant $c > 0$ distance away from $(\mathbf{u}^*, \mathbf{v}^*)$. To this we need a Bernstein-style argument which we provide here for the rank-1 case.

Theorem 8.2. [Tropp, 2012, Theorem 1.6] Consider a finite sequence $\{Z_k\}$ of independent random matrices of dimension $m\times n$. Assume each matrix satisfies $\mathbb{E}\left[Z_k\right] = \mathbf{0}$ and $\left\|Z_k\right\|_2 \leq R$ almost surely and denote

$$\sigma^2 := \max\left\{\left\|\sum_k Z_k Z_k^\top\right\|_2, \left\|\sum_k Z_k^\top Z_k\right\|_2\right\}.$$

Then, for all $t \geq 0$, we have

$$\mathbb{P}\left[\left\|\sum_k Z_k\right\|_2 \geq t\right] \leq (m+n)\cdot\exp\left(\frac{-t^2}{\sigma^2 + Rt/3}\right).$$

Below we apply this inequality to analyze the initialization step for AM-MC in the rank-1 case. We point the reader to [Recht, 2011] and [Keshavan et al., 2010] for a more precise argument analyzing the initialization step in the general rank-r case.

Theorem 8.3. For a rank one matrix A satisfying the μ-incoherence property, let the observed samples Ω be generated with sampling probability p as described above. Let $\mathbf{u}^1, \mathbf{v}^1$ be the singular vectors of $\frac{1}{p}P_\Omega(A)$ corresponding to its largest singular value. Then for any $\epsilon > 0$, if $p \geq \frac{45\mu^2}{\epsilon^2} \cdot \frac{\log(m+n)}{\min\{m,n\}}$, then with probability at least $1 - 1/(m+n)^{10}$:

$$\left\| \frac{1}{p}\Pi_\Omega(A) - A \right\|_2 \leq \epsilon, \quad \left\| \mathbf{u}^1 - \mathbf{u}^* \right\|_2 \leq \epsilon, \quad \left\| \mathbf{v}^1 - \mathbf{v}^* \right\|_2 \leq \epsilon.$$

Moreover, the vectors $\mathbf{u}^1$ and $\mathbf{v}^1$ are also 2μ-incoherent.

Proof. Notice that the statement of the theorem essentially states that once enough entries in the matrix have been observed (as dictated by the requirement $p \geq \frac{45\mu^2}{\epsilon^2} \cdot \frac{\log(m+n)}{\min\{m,n\}}$) an SVD step on the incomplete matrix will yield components $\mathbf{u}^1, \mathbf{v}^1$ that are very close to the components of the complete matrix $\mathbf{u}^*, \mathbf{v}^*$. Moreover, since $\mathbf{u}^*, \mathbf{v}^*$ are incoherent by assumption, the estimated components $\mathbf{u}^1, \mathbf{v}^1$ will be so too.

To apply Theorem 8.2 to prove this result, we will first express $\frac{1}{p}\Pi_\Omega(A)$ as a sum of random matrices. We first rewrite $\frac{1}{p}\Pi_\Omega(A) = \frac{1}{p}\sum_{ij} \delta_{ij} A_{ij} \mathbf{e}_i \mathbf{e}_j^\top = \sum_{ij} W_{ij}$ where $\delta_{ij} = 1$ if $(i,j) \in \Omega$ and 0 otherwise. Note that the Bernoulli sampling model assures us that the random variables $W_{ij} = \frac{1}{p}\delta_{ij} A_{ij} \mathbf{e}_i \mathbf{e}_j^\top$ are independent and that $\mathbb{E}[\delta_{ij}] = p$. This gives us $\mathbb{E}[W_{ij}] = A_{ij}\mathbf{e}_i\mathbf{e}_j^\top$. Note that $\sum_{ij} A_{ij}\mathbf{e}_i\mathbf{e}_j^\top = A$.

The matrices $Z_{ij} = W_{ij} - A_{ij}\mathbf{e}_i\mathbf{e}_j^\top$ shall serve as our *random matrices* in the application of Theorem 8.2. Clearly $\mathbb{E}[Z_{ij}] = \mathbf{0}$. We also have $\max_{ij} \|W_{ij}\|_2 \leq \frac{1}{p}\max_{ij}|A_{ij}| \leq \frac{\mu^2}{p\sqrt{mn}}$ due to the incoherence assumption. Applying the triangle inequality gives us $\max_{ij}\|Z_{ij}\|_2 \leq \max_{ij}\|W_{ij}\|_2 + \max_{ij}\|A_{ij}\mathbf{e}_i\mathbf{e}_j^\top\|_2 \leq \left(1 + \frac{1}{p}\right)\frac{\mu^2}{\sqrt{mn}} \leq \frac{2\mu^2}{p\sqrt{mn}}$.

Moreover, as $A_{ij} = \mathbf{u}_i^*\mathbf{v}_j^*$ and $\|\mathbf{v}^*\|_2 = 1$, we have $\mathbb{E}\left[\sum_{ij} W_{ij}W_{ij}^\top\right] = \frac{1}{p}\sum_i\sum_j A_{ij}^2\mathbf{e}_i\mathbf{e}_i^\top = \frac{1}{p}\sum_i(\mathbf{u}_i^*)^2\mathbf{e}_i\mathbf{e}_i^\top$. Due to incoherence $\|\mathbf{u}^*\|_\infty \leq \frac{\mu}{\sqrt{m}}$, we get $\left\|\mathbb{E}\left[\sum_{ij} W_{ij}W_{ij}^\top\right]\right\|_2 \leq \frac{\mu^2}{p\cdot m}$, which can be shown to give us

$$\left\|\mathbb{E}\left[\sum_{ij} Z_{ij}Z_{ij}^\top\right]\right\|_2 \leq \left(\frac{1}{p}-1\right)\cdot\frac{\mu^2}{m} \leq \frac{\mu^2}{p\cdot m}$$

Similarly, we can also get $\left\|\mathbb{E}\left[\sum_{ij} Z_{ij}^\top Z_{ij}\right]\right\|_2 \leq \frac{\mu^2}{p\cdot n}$. Now using Theorem 8.2 gives us, with probability at least $1-\delta$,

$$\left\|\frac{1}{p}\Pi_\Omega(A) - A\right\|_2 \leq \frac{2\mu^2}{3p\sqrt{mn}}\log\left[\frac{m+n}{\delta}\right] + \sqrt{\frac{\mu^2}{p\cdot\min\{m,n\}}\log\left[\frac{m+n}{\delta}\right]}$$

If $p \geq \frac{45\mu^2\log(m+n)}{\epsilon^2\cdot\min\{m,n\}}$, we have with probability at least $1-1/(m+n)^{10}$,

$$\left\|\frac{1}{p}\Pi_\Omega(A) - A\right\|_2 \leq \epsilon.$$

The proof now follows by applying the Davis-Kahan inequality [Golub and Loan, 1996] with the above bound. It can be shown [Jain and Netrapalli, 2015] that the vectors that are recovered as a result of this initialization are incoherent as well. □

Linear Convergence: We will now show that, given the initialization above, the AM-MC procedure converges to the true solution with a linear rate of convergence. This will involve showing a few intermediate results, such as showing that the alternation steps preserve incoherence. Since the Theorem 8.3 shows that $\mathbf{u}^1$ is 2μ-incoherent, this will establish the incoherence of all future iterates. Preserving incoherence will be crucial in showing the next result which shows that successive iterates get increasingly close to the optimum. Put together, these will establish the convergence result. First, recall that in the t^{th} iteration of the AM-MC algorithm, $\mathbf{v}^{t+1}$ is updated as

$$\mathbf{v}^{t+1} = \arg\min_{\mathbf{v}} \sum_{ij}\delta_{ij}(\mathbf{u}_i^t\mathbf{v}_j - \mathbf{u}_i^*\mathbf{v}_j^*)^2,$$

which gives us

$$\mathbf{v}_j^{t+1} = \frac{\sum_i \delta_{ij}\mathbf{u}_i^*\mathbf{u}_i^t}{\sum_i \delta_{ij}(\mathbf{u}_i^t)^2}\cdot\mathbf{v}_j^*. \tag{8.1}$$

Note that this means that if $\mathbf{u}^* = \mathbf{u}^t$, then $\mathbf{v}^{t+1} = \mathbf{v}^*$. Also, note that if $\delta_{ij} = 1$ for all (i, j) which happens when the sampling probability satisfies $p = 1$, we have $\mathbf{v}^{t+1} = \frac{\langle \mathbf{u}^t, \mathbf{u}^* \rangle}{\|\mathbf{u}^t\|_2^2} \cdot \mathbf{v}^*$. This is reminiscent of the *power method* used to recover the leading singular vectors of a matrix. Indeed if we let $\widetilde{\mathbf{u}} = \mathbf{u}^t / \left\|\mathbf{u}^t\right\|_2$, we get $\left\|\mathbf{u}^t\right\|_2 \cdot \mathbf{v}^{t+1} = \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle \cdot \mathbf{v}^*$ if $p = 1$.

This allows us to rewrite the update (8.1) as a noisy power update.

$$\left\|\mathbf{u}^t\right\|_2 \cdot \mathbf{v}^{t+1} = \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle \cdot \mathbf{v}^* - B^{-1}(\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle B - C)\mathbf{v}^* \tag{8.2}$$

where $B, C \in \mathbb{R}^{n \times n}$ are diagonal matrices with $B_{jj} = \frac{1}{p} \sum_i \delta_{ij} (\widetilde{\mathbf{u}}_i)^2$ and $C_{jj} = \frac{1}{p} \sum_i \delta_{ij} \widetilde{\mathbf{u}}_i \mathbf{u}_i^*$. The following two lemmata show that if $\mathbf{u}^t$ is 2μ incoherent and if p is large enough, then: a) $\mathbf{v}^{t+1}$ is also 2μ incoherent, and b) the angular distance between $\mathbf{v}^{t+1}$ and $\mathbf{v}^*$ decreases as compared to that between $\mathbf{u}^t$ and $\mathbf{u}^*$. The following lemma will aid the analysis.

Lemma 8.4. Suppose $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are two fixed μ-incoherent unit vectors. Also suppose $\delta_i, i \in [n]$ are i.i.d. Bernoulli random variables such that $\delta_i = 1$ with probability p and 0 otherwise. Then, for any $\epsilon > 0$, if $p > \frac{27\mu^2 \log n}{n\epsilon^2}$, then with probability at least $1 - 1/n^{10}$, $\left|\frac{1}{p} \sum_i \delta_i \mathbf{a}_i \mathbf{b}_i - \langle \mathbf{a}, \mathbf{b} \rangle\right| \leq \epsilon$.

Proof. Define $Z_i = \left(\frac{\delta_i}{p} - 1\right) \mathbf{a}_i \mathbf{b}_i$. Using the incoherence of the vectors, we get $\mathbb{E}\left[Z_i\right] = 0$, $\sum_{i=1}^n \mathbb{E}\left[Z_i^2\right] = \left(\frac{1}{p} - 1\right) \sum_{i=1}^n (\mathbf{a}_i \mathbf{b}_i)^2 \leq \frac{\mu^2}{pn}$ since $\|\mathbf{b}\|_2 = 1$, and $|Z_i| \leq \frac{\mu^2}{pn}$ almost surely. Applying the Bernstein inequality gives us

$$\mathbb{P}\left[\left|\frac{1}{p} \sum_i \delta_i \mathbf{a}_i \mathbf{b}_i - \langle \mathbf{a}, \mathbf{b} \rangle\right| > t\right] \leq \exp\left(\frac{-3pnt^2}{6\mu^4 + 2\mu^2 t}\right),$$

which upon simple manipulations, gives us the result. □

Lemma 8.5. With probability at least $\min\left\{1 - 1/n^{10}, 1 - 1/m^{10}\right\}$, if a pair of iterates $(\mathbf{u}^t, \mathbf{v}^t)$ in the execution of the AM-MC procedure are 2μ-incoherent, then so are the next pair of iterates $(\mathbf{u}^{t+1}, \mathbf{v}^{t+1})$.

Proof. Since $\|\widetilde{\mathbf{u}}\|_2 = 1$, using Lemma 8.4 tells us that with high probability, for all j, we have $|B_{jj} - 1| \leq \epsilon$ as well as $|C_{jj} - \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle| \leq \epsilon$.

Also, using triangle inequality, we get $\left\|\mathbf{u}^t\right\|_2 \geq 1-\epsilon$. Using these and the incoherence of $\mathbf{v}^*$ in the update equation for $\mathbf{v}^{t+1}$ (8.2), we have

$$\begin{aligned}
\left|\mathbf{v}_j^{t+1}\right| &= \frac{1}{1-\epsilon}\left|\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle \mathbf{v}_j^* - \frac{1}{B_{jj}}(\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle B_{jj} - C_{jj})\mathbf{v}_j^*\right| \\
&\leq \frac{1}{1-\epsilon}\left|\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle \mathbf{v}_j^*\right| + \frac{1}{1-\epsilon}\left|\frac{1}{B_{jj}}(\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle B_{jj} - C_{jj})\mathbf{v}_j^*\right| \\
&\leq \frac{1}{(1-\epsilon)^2}\left(\left|\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle\right| + \left|\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle(1+\epsilon) - (\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle - \epsilon)\right|\right)\frac{\mu}{\sqrt{n}} \\
&\leq \frac{1+2\epsilon}{(1-\epsilon)^2}\frac{\mu}{\sqrt{n}}
\end{aligned}$$

For $\epsilon < 1/6$, the result now holds. □

We note that whereas Lemma 8.4 is proved for fixed vectors, we seem to have inappropriately applied it to $\tilde{\mathbf{u}}$ in the proof of Lemma 8.5 which is not a fixed vector as it depends on the randomness used in sampling the entries of the matrix revealed to the algorithm. However notice that the AM-MC procedure in Algorithm 10 uses fresh samples Ω_t and Ω_{T+t} for each iteration. This ensures that $\tilde{\mathbf{u}}$ does behave like a fixed vector with respect to Lemma 8.4.

Lemma 8.6. For any $\epsilon > 0$, if $p > \frac{80\mu^2\log(m+n)}{\epsilon^2\min\{m,n\}}$ and $\mathbf{u}^t$ is 2μ-incoherent, the next iterate $\mathbf{v}^{t+1}$ satisfies

$$1 - \left\langle \frac{\mathbf{v}^{t+1}}{\left\|\mathbf{v}^{t+1}\right\|_2}, \mathbf{v}^*\right\rangle^2 \leq \frac{\epsilon}{(1-\epsilon)^3}\left(1 - \left\langle\frac{\mathbf{u}^t}{\left\|\mathbf{u}^t\right\|_2}, \mathbf{u}^*\right\rangle^2\right)$$

Similarly, for any 2μ-incoherent iterate $\mathbf{v}^{t+1}$, the next iterate satisfies

$$1 - \left\langle \frac{\mathbf{u}^{t+1}}{\left\|\mathbf{u}^{t+1}\right\|_2}, \mathbf{u}^*\right\rangle^2 \leq \frac{\epsilon}{(1-\epsilon)^3}\left(1 - \left\langle\frac{\mathbf{v}^{t+1}}{\left\|\mathbf{v}^{t+1}\right\|_2}, \mathbf{v}^*\right\rangle^2\right).$$

Proof. Using the modified form of the update for $\mathbf{u}^{t+1}$ (8.2), we get, for any unit vector $\mathbf{v}_\perp$ such that $\langle\mathbf{v}_\perp, \mathbf{v}^*\rangle = 0$,

$$\begin{aligned}
\left\|\mathbf{u}^t\right\|_2 \cdot \left\langle \mathbf{v}^{t+1}, \mathbf{v}_\perp\right\rangle &= \left\langle \mathbf{v}_\perp, B^{-1}(\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle B - C)\mathbf{v}^*\right\rangle \\
&\leq \left\|B^{-1}\right\|_2 \left\|(\langle\tilde{\mathbf{u}}, \mathbf{u}^*\rangle B - C)\mathbf{v}^*\right\|_2
\end{aligned}$$

$$\leq \frac{1}{1-\epsilon} \left\|(\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle B - C)\mathbf{v}^*\right\|_2,$$

where the last step follows from an application of Lemma 8.4. To bound the other term let $Z_{ij} = \frac{1}{p}\delta_{ij}(\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle (\widetilde{\mathbf{u}}_i)^2 - \widetilde{\mathbf{u}}_i\mathbf{u}^*_i)\mathbf{v}^*_j\mathbf{e}_j \in \mathbb{R}^n$. Clearly $\sum_{i=1}^m \sum_{j=1}^n Z_{ij} = (\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle B - C)\mathbf{v}^*$. Note that due to fresh samples being used by Algorithm 10 at every step, the vector $\widetilde{\mathbf{u}}$ appears as a constant vector to the random variables δ_{ij}. Given this, note that

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^m Z_{ij}\right] &= \sum_{i=1}^m (\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle (\widetilde{\mathbf{u}}_i)^2 - \widetilde{\mathbf{u}}_i\mathbf{u}^*_i)\mathbf{v}^*_j\mathbf{e}_j \\
&= \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle \sum_{i=1}^m (\widetilde{\mathbf{u}}_i)^2\mathbf{v}^*_j\mathbf{e}_j - \sum_{i=1}^m \widetilde{\mathbf{u}}_i\mathbf{u}^*_i\mathbf{v}^*_j\mathbf{e}_j \\
&= (\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle \left\|\widetilde{\mathbf{u}}\right\|_2^2 - \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle)\mathbf{v}^*_j\mathbf{e}_j = \mathbf{0},
\end{aligned}$$

since $\left\|\widetilde{\mathbf{u}}\right\|_2 = 1$. Thus $\mathbb{E}\left[\sum_{ij} Z_{ij}\right] = \mathbf{0}$ as well. Now, we have $\max_i(\widetilde{\mathbf{u}}_i)^2 = \frac{1}{\|\mathbf{u}^t\|_2^2} \cdot \max_i(\mathbf{u}^t_i)^2 \leq \frac{4\mu^2}{m\|\mathbf{u}^t\|_2^2} \leq \frac{\mu^2}{m(1-\epsilon)}$ since $\left\|\mathbf{u}^t - \mathbf{u}^*\right\|_2 \leq \epsilon$. This allows us to bound

$$\begin{aligned}
\left|\mathbb{E}\left[\sum_{ij} Z_{ij}^\top Z_{ij}\right]\right| &= \frac{1}{p}\sum_{i=1}^m\sum_{j=1}^n (\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle (\widetilde{\mathbf{u}}_i)^2 - \widetilde{\mathbf{u}}_i\mathbf{u}^*_i)^2(\mathbf{v}^*_j)^2 \\
&= \frac{1}{p}\sum_{i=1}^m (\widetilde{\mathbf{u}}_i)^2(\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle \widetilde{\mathbf{u}}_i - \mathbf{u}^*_i)^2 \\
&\leq \frac{\mu^2}{pm(1-\epsilon)} \sum_{i=1}^m \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle^2 (\widetilde{\mathbf{u}}_i)^2 + (\mathbf{u}^*_i)^2 - 2\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle \widetilde{\mathbf{u}}_i\mathbf{u}^*_i \\
&\leq \frac{8\mu^2}{pm}(1 - \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle^2),
\end{aligned}$$

where we set $\epsilon = 0.5$. In the same way we can show $\left\|\mathbb{E}\left[\sum_{ij} Z_{ij}^\top Z_{ij}\right]\right\|_2 \leq \frac{8\mu^2}{pm}(1 - \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle^2)$ as well. Using a similar argument we can show $\left\|Z_{ij}\right\|_2 \leq \frac{4\mu^2}{p\sqrt{mn}}\sqrt{1 - \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle^2}$. Applying the Bernstein inequality now tells us, that for any $\epsilon > 0$, if $p > \frac{80\mu^2 \log(m+n)}{\epsilon^2 \min\{m,n\}}$, then with probability at least $1 - 1/n^{10}$, we have

$$\left\|(\langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle B - C)\mathbf{v}^*\right\|_2 \leq \epsilon \cdot \sqrt{1 - \langle \widetilde{\mathbf{u}}, \mathbf{u}^* \rangle^2}.$$

Since $\left\|\mathbf{u}^t\right\|_2 \geq 1-\epsilon$ is guaranteed by the initialization step, we now get,

$$\left\langle \mathbf{v}^{t+1}, \mathbf{v}_\perp \right\rangle \leq \frac{\epsilon}{(1-\epsilon)^2}\sqrt{1-\langle \tilde{\mathbf{u}}, \mathbf{u}^* \rangle^2}.$$

If $\mathbf{v}^{t+1}_\perp$ and $\mathbf{v}^{t+1}_\parallel$ be the components of $\mathbf{v}^{t+1}$ perpendicular and parallel to $\mathbf{v}^*$. Then the above guarantees that $\left\|\mathbf{v}^\perp\right\|_2 \leq c \cdot \sqrt{1-\langle \tilde{\mathbf{u}}, \mathbf{u}^* \rangle^2}$. This gives us, upon applying the Pythagoras theorem,

$$\left\|\mathbf{v}^{t+1}\right\|_2^2 = \left\|\mathbf{v}^{t+1}_\perp\right\|_2^2 + \left\|\mathbf{v}^{t+1}_\parallel\right\|_2^2 \leq \frac{\epsilon}{(1-\epsilon)^2}\left(1-\langle \tilde{\mathbf{u}}, \mathbf{u}^* \rangle^2\right) + \left\|\mathbf{v}^{t+1}_\parallel\right\|_2^2$$

Since $\left\|\mathbf{v}^{t+1}_\parallel\right\|_2 = \langle \mathbf{v}^{t+1}, \mathbf{v}^* \rangle$ and $\left\|\mathbf{v}^{t+1}\right\|_2 \geq 1-\epsilon$ as $\left\|\mathbf{v}^{t+1}-\mathbf{v}^*\right\| \leq \epsilon$ due to the initialization, rearranging the terms gives us the result. □

Using these results, it is easy to establish the main theorem.

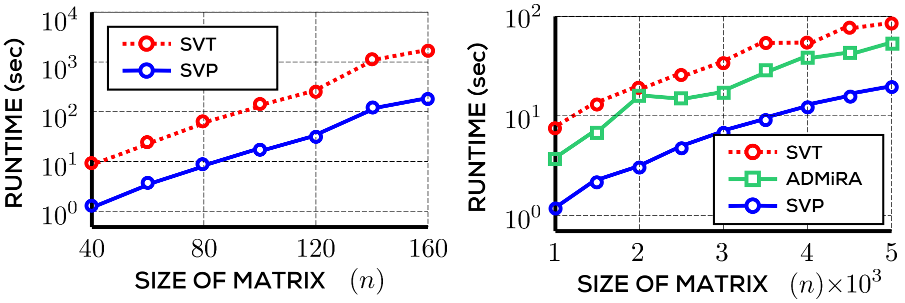
Theorem 8.7. Let $A = \mathbf{u}^*(\mathbf{v}^*)^\top$ be a unit rank matrix where $\mathbf{u}^* \in \mathbb{R}^m$ and $\mathbf{v}^* \in \mathbb{R}^n$ are two μ-incoherent unit vectors. Let the matrix be observed at a set of indices $\Omega \subseteq [m] \times [n]$ where each index is observed with probability p. Then if $p \geq C \cdot \frac{\mu^2 \log(m+n)}{\epsilon^2 \min\{m,n\}}$ for some large enough constant C, then with probability at least $1 - 1/\min\{m,n\}^{10}$, AM-MC generates iterates which are 2μ-incoherent. Moreover, within $\mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ iterations, AM-MC also ensures that $\left\|\frac{\mathbf{u}^t}{\left\|\mathbf{u}^t\right\|_2} - \mathbf{u}^*\right\|_2 \leq \epsilon$ and $\left\|\frac{\mathbf{v}^t}{\left\|\mathbf{v}^t\right\|_2} - \mathbf{v}^*\right\|_2 \leq \epsilon$.

8.8 Other Popular Techniques for Matrix Recovery

As has been the case with the other problems we have studied so far, the first approaches to solving the ARM and LRMC problems were relaxation based approaches [Candès and Recht, 2009, Candès and Tao, 2009, Recht et al., 2010, Recht, 2011]. These approaches relax the non-convex rank objective in the ((ARM)) formulation using the (convex) *nuclear norm*

$$\begin{array}{rl} \min & \left\|X\right\|_* \\ \text{s.t.} & \mathcal{A}(X) = \mathbf{y}, \end{array}$$

where the nuclear norm of a matrix $\left\|X\right\|_*$ is the sum of all singular values of the matrix X. The nuclear norm is known to provide the



(a) Matrix Recovery

(b) Matrix Completion

Figure 8.2: An empirical comparison of run-times offered by the SVT, ADMiRA and SVP methods on synthetic matrix recovery and matrix completion problems with varying matrix sizes. The SVT method due to Cai et al. [2010] is an efficient implementation of the nuclear norm relaxation technique. ADMiRA is a pursuit-style method due to Lee and Bresler [2010]. For the ARM task in Figure 8.2a, the rank of the true matrix was set to $r = 5$ whereas it was set to $r = 2$ for the LRMC task in Figure 8.2b. SVT is clearly the most scalable of the methods in both cases whereas the relaxation-based SVT technique does not scale very well to large matrices. Note however, that for the LRMC problem, AM-MC (not shown in the figure) outperforms even SVP. Figures adapted from [Meka et al., 2008].

tightest convex envelope of the rank function, just as the ℓ_1 norm provides a relaxation to the sparsity norm $\|\cdot\|_0$ [Recht et al., 2010]. Similar to sparse recovery, under matrix-RIP settings, these relaxations can be shown to offer exact recovery [Recht et al., 2010, Hastie et al., 2016].

Also similar to sparse recovery, there exist pursuit-style techniques for matrix recovery, most notable among them being the ADMiRA method [Lee and Bresler, 2010] that extends the orthogonal matching pursuit approach to the matrix recovery setting. However, this method can be a bit sluggish when recovering matrices with slightly large rank since it discovers a matrix with larger and larger rank incrementally.

Before concluding, we present the reader with an empirical performance of these various methods. Figure 8.2 provides a comparison of these methods on synthetic matrix recovery and matrix completion problems with increasing dimensionality of the (low-rank) matrix being recovered. The graphs indicate that non-convex optimization methods such as IHT are far more scalable, often by an order of magnitude, than relaxation-based methods.

8.9 Exercises

Exercise 8.1. Show that low-rank matrix recovery is NP-hard.
Hint: Take the sparse recovery problem in ((SP-REG)) and reduce it to the reformulation ((ARM-2)) of the matrix recovery problem.

Exercise 8.2. Show that the matrix RIP constant is monotonic in its order i.e., if a linear map $\mathcal{A}$ satisfies matrix RIP of order r with constant δ_r, then it also satisfies matrix RIP for all orders $r' \leq r$ with $\delta_{r'} \leq \delta_r$.

8.10 Bibliographic Notes

There are a lot of aspects of low-rank matrix recovery that we have not covered in our discussion. Here we briefly point out some of these.

Similar to the sparse regression setting, the problem of ill-conditioned problems requires special care in the matrix recovery setting as well. For the general ARM problem, the work of Jain et al. [2014] does this by first proposing appropriate versions of RSC-RSS properties (see Definition 7.4) for the matrix case, and the suitably modifying SVP-style techniques to function even in high condition number settings. The final result is similar to the one for sparse recovery (see Theorem 7.4) wherein a more "relaxed" projection step is required by using a rank $q > r$ while executing the SVP algorithm.

It turns out to be challenging to prove convergence results for SVP for the LRMC problem. This is primarily due to the difficulty in establishing the matrix RIP property for the affine transformation used in the problem. The affine map simply selects a few elements of the matrix and reproduces them which makes establishing RIP properties harder in this setting. Specifically, even though the initialization step can be shown to yield a matrix that satisfies matrix RIP [Jain et al., 2010], if the underlying matrix is low-rank and incoherent, it becomes challenging to show that RIP-ness is maintained across iterates. Jain and Netrapalli [2015] overcome this by executing the SVP algorithm in a *stage-wise* fashion which resembles ADMiRA-like pursuit approaches.

Several works have furthered the alternating minimization approach itself by reducing its sample complexity [Hardt, 2014], giving recovery guarantees independent of the condition number of the problem [Hardt and Wootters, 2014, Jain and Netrapalli, 2015], designing universal sampling schemes for recovery [Bhojanapalli and Jain, 2014], as well as tackling settings where some of the revealed entries of the matrix may be corrupted [Chen et al., 2016, Cherapanamjeri et al., 2017].

Another interesting line of work for matrix completion is that of [Ge et al., 2016, Sun and Lu, 2015] which shows that under certain regularization assumptions, the matrix completion problem does not have any non-optimal stationary points once one gets close-enough to the global minimum. Thus, one can use any method for convex optimization such as alternating minimization, gradient descent, stochastic gradient descent, and its variants we studied in § 6, once one is close enough to the global minimum.

9

Robust Linear Regression

In this section, we will look at the problem of robust linear regression. Simply put, it is the task of performing linear regression in the presence of adversarial *outliers* or *corruptions*. Let us take a look at some motivating applications.

9.1 Motivating Applications

The problem of regression has widespread application in signal processing, financial and economic analysis, as well as machine learning and data analytics. In most real life applications, the data presented to the algorithm has some amount of noise in it. However, at times, data may be riddled with missing values and corruptions, and that too of a malicious nature. It is useful to design algorithms that can assure stable operation even in the presence of such corruptions.

Face Recognition The task of face recognition is widely useful in areas such as biometrics and automated image annotation. In biometrics, a fundamental problem is to identify if a new face image belongs to that of a registered individual or not. This problem can be cast as a regression problem by trying to fit various features of the new image to corresponding features of existing images of the individual in the registered database. More specifically, assume that images are represented as n-dimensional feature vectors say, using simple pixel-based features. Also assume that there already exist p images of the person in the database.

Our task is to represent the new image $\mathbf{x}^t \in \mathbb{R}^n$ in terms of the database images $X = [\mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n \times p}$ of that person. A nice way to do this is to perform a linear interpolation as follows

$$\min_{\mathbf{w} \in \mathbb{R}^p} \left\| \mathbf{x}^t - X\mathbf{w} \right\|_2^2 = \sum_{i=1}^{n} (\mathbf{x}_i^t - X^i \mathbf{w})^2.$$

If the person is genuine, then there will exist a combination $\mathbf{w}^*$ such that for all i, we have $\mathbf{x}_i^t \approx X^i \mathbf{w}^*$ i.e., all features can be faithfully reconstructed. Thus, the fit will be nice and we will admit the person. However, the same becomes problematic if the new image has occlusions. For example, if the person is genuine but wearing a pair of sunglasses or sporting a beard. In such cases, some of the pixels $\mathbf{x}_i^t$ will appear corrupted, cause us to get a poor fit, and result in a false alarm. More specifically

$$\mathbf{x}_i^t = X^i \mathbf{w}^* + \mathbf{b}_i^*$$

where $\mathbf{b}_i^* = 0$ on uncorrupted pixels but can take abnormally large and unpredictable values for corrupted pixels, such as those corresponding to the sunglasses. Being able to still correctly identify the person involves computing the least squares fit in the presence of such corruptions. The challenge is to do this without requiring any manual effort to identify the locations of the corrupted pixels. Figure 9.1 depicts this problem setting visually.

Time Series Analysis This is a problem that has received much independent attention in statistics and signal processing due to its applications in modeling sequence data such as weather data, financial data, and DNA sequences. However, the underlying problem is similar to that of regression. A sequence of timestamped observations $\{y_t\}$ for $t = 0, 1, \ldots$ are made which constitute the time series. Note that the ordering of the samples is critical here.

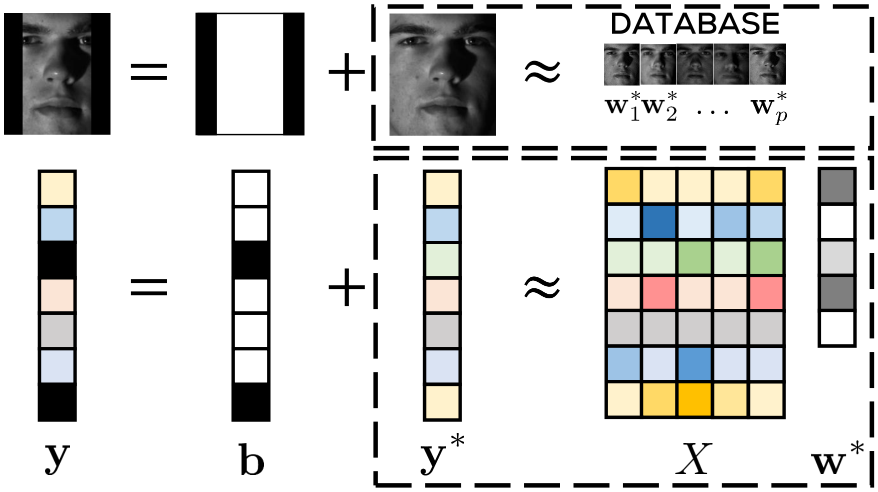


Figure 9.1: A corrupted image $\mathbf{y}$ can be interpreted as a combination of a clean image $\mathbf{y}^*$ and a corruption mask $\mathbf{b}^*$ i.e., $\mathbf{y} = \mathbf{y}^* + \mathbf{b}^*$. The mask encodes the locations of the corrupted pixels as well as the values of the corruptions. The clean image can be (approximately) recovered as an affine combination of existing images in a database as $\mathbf{y}^* \approx X\mathbf{w}^*$. Face reconstruction and recognition in such a scenario constitutes a robust regression problem. Note that the corruption mask $\mathbf{b}^*$ is sparse since only a few pixels are corrupted. Images courtesy the Yale Face Database B.

The popular auto-regressive (AR) time series model uses a generative mechanism wherein the observation y_t at time t is obtained as a fixed linear combination of p previous observations plus some noise.

$$y_t = \sum_{i=1}^{p} \mathbf{w}_i^* y_{t-i} + \eta_t$$

We can cast this into a regression framework by constructing covariates $\mathbf{x}_t := [y_{t-1}, y_{t-2}, \ldots, y_{t-p}]^\top \in \mathbb{R}^p$ and rewriting the above as

$$y_t = \mathbf{x}_t^\top \mathbf{w}^* + \eta_t,$$

where $\mathbf{w}^* \in \mathbb{R}^p$ is an unknown model and η_i is Gaussian noise generated independent of previous observations $\eta_t | \{y_{t-1}, y_{t-2}, \ldots\} \sim \mathcal{N}(0, \sigma^2)$. The number p is known as the *order* of the time series and captures how many historical observations affect the current one.

Is not uncommon to encounter situations where the time series experiences gross corruptions. Examples may include observation or sensing errors or unmodeled factors such as dips and upheavals in stock prices due to political or socio-economic events. Thus, we have

$$y_t = \mathbf{x}_t^\top \mathbf{w}^* + \eta_t + \mathbf{b}_t^*$$

where $\mathbf{b}_t^*$ can take unpredictable values for corrupted time instances and 0 otherwise. In time series literature, two corruption models are popular. In the *additive* model, observations at time steps subsequent to a corruption are constructed using the uncorrupted values i.e., $\mathbf{b}_t^*$ does not influence the values y_τ for $\tau > t$. Of course, observers may detect the corruption at time t but the underlying time series goes on as though nothing happened.

However, in the *innovative* model, observations at time instances subsequent to a corruption use the corrupted value i.e., $\mathbf{b}_t^*$ is involved in constructing the values y_τ for $\tau = t+1, \ldots, t+p$. Innovative corruptions are simpler to handle as although the observation at the moment of the corruption i.e., y_t, appears to deviate from that predicted by the base model i.e., $\mathbf{x}_t^\top \mathbf{w}^*$, subsequent observations fall in line with the predictions once more (unless there are more corruptions down the line). In the additive model however, observations can seem to deviate from the predictions of the base model for several iterations. In particular, y_τ can disagree with $\mathbf{x}_\tau^\top \mathbf{w}^*$ for times $\tau = t, t+1, \ldots, t+p$, even if there is only a single corruption at time t.

A time series analysis technique which seeks to study the "usual" behavior of the model involved, such as stock prices, might wish to exclude such aberrations, whether additive or innovative. However it is unreasonable, as well as error prone, to expect manual exclusion of such corruptions which motivates the problem of robust time series analysis. Note that time series analysis is a more challenging problem than regression since the "covariates" in this case $\mathbf{x}_t, \mathbf{x}_{t+1}, \ldots$ are heavily correlated with each other as they share a large number of coordinates whereas in regression they are usually assumed to be independent.

9.2 Problem Formulation

Let us recall the regression model studied in § 5.1 and § 7.2. In a linear regression setting, responses are modeled as a (possibly sparse) linear function of the data features to which some additive noise is added, i.e., we have

$$y_i = \mathbf{x}_i^\top \mathbf{w}^* + \eta_i.$$

However, notice that in previous discussions, we termed the noise as benign, and even found it appropriate to assume it could be modeled as a Gaussian random variable. This is usually acceptable when the noise is expected to be non-deliberate or the result of small and unstructured perturbations. However, the outliers or corruptions in the examples that we just saw, seem to defy these assumptions.

In the face recognition case, salt and pepper noise due to sensor disturbances, or lighting changes can be considered benign noise. However, it is improper to consider structured occlusions such as sunglasses or beards as benign as some of them may even be a part of a malicious attempt to fool the system. Even in the time series analysis setting, there might be malicious forces at play in the stock market which cause stock prices to significantly deviate from their usual variations and trends.

Such corruptions can severely disrupt the modeling process and hamper our understanding of the system. To handle them, we will need a more refined model that distinguishes between benign and unstructured errors, and errors that are deliberate, malicious and structured. The *robust regression* model best describes this problem setting

$$y_i = \mathbf{x}_i^\top \mathbf{w}^* + \eta_i + \mathbf{b}_i^*$$

where the variable $\mathbf{b}_i^*$ encodes the *additional* corruption introduced into the response. Our goal is to take a set of n (possibly) corrupted data points $(\mathbf{x}_i, y_i)_{i=1}^n$ and recover the underlying parameter vector $\mathbf{w}^*$, i.e.,

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^p, \mathbf{b}\in\mathbb{R}^n \\ \|\mathbf{b}\|_0 \leq k}} \|\mathbf{y} - X\mathbf{w} - \mathbf{b}\|_2^2, \tag{ROB-REG}$$

The variables $\mathbf{b}_i^*$ can be unbounded in magnitude and of arbitrary sign. However, we assume that only a few data points are corrupted i.e., the vector $\mathbf{b}^* = [\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*]$ is sparse $\|\mathbf{b}^*\|_0 \leq k$. Indeed it is impossible[1] to recover the model $\mathbf{w}^*$ if more than half the points are

[1] See Exercise 9.1.

Algorithm 11 AltMin for Robust Regression (AM-RR)

```
Input: Data $X, \mathbf{y}$, number of corruptions $k$
Output: An accurate model $\widehat{\mathbf{w}} \in \mathbb{R}^p$
$\mathbf{w}^1 \leftarrow \mathbf{0}$, $S_1 = [1 : n - k]$
for $t = 1, 2, \ldots$ do
    $\mathbf{w}^{t+1} \leftarrow \arg\min_{\mathbf{w}\in\mathbb{R}^p} \sum_{i \in S_t} (y_i - \mathbf{x}_i^\top \mathbf{w})^2$
    $S_{t+1} \leftarrow \arg\min_{|S|=n-k} \sum_{i \in S} (y_i - \mathbf{x}_i^\top \mathbf{w}^{t+1})^2$
end for
return $\mathbf{w}^t$
```

corrupted i.e., $k \geq n/2$. A worthy goal is to develop algorithms that can tolerate as large a value of k as possible. We will study how two non-convex optimization techniques, namely gAM and gPGD, can be used to solve this problem. We point to other approaches, as well as extensions such as robust sparse recovery, in the bibliographic notes.

9.3 Robust Regression via Alternating Minimization

The key to applying alternating minimization to the robust regression problem is to identify the two critical parameters in this problem. Let us assume that there is no Gaussian noise in the model i.e., $\mathbf{y} = X\mathbf{w}^* + \mathbf{b}^*$ where $\mathbf{b}^*$ is k-sparse but can contain unbounded entries in its support.

It can be seen that $\mathbf{w}^*$ and $\overline{\operatorname{supp}(\mathbf{b}^*)} =: S_*$, i.e., the true model and the locations of the uncorrupted points, are the two most crucial elements since given one, finding the other is very simple. Indeed, if someone were to magically hand us $\mathbf{w}^*$, it is trivial to identify S_* by simply identifying data points where $y_i = \mathbf{x}_i^\top \mathbf{w}^*$. On the other hand, given S_*, it is simple to obtain $\mathbf{w}^*$ by simply solving a least squares regression problem on the set of data points in the set S_*. Thus we can rewrite the robust regression problem as

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^p \\ |S|=n-k}} \left\|\mathbf{y}_S - X^S\mathbf{w}\right\|_2^2 \tag{ROB-REG-2}$$

This gives us a direct way of applying the gAM approach to this problem as outlined in the AM-RR algorithm (Algorithm 11). The work of

Bhatia et al. [2015] showed that this technique and its variants offer scalable solutions to the robust regression problem.

In order to execute the gAM protocol, AM-RR maintains a model estimate $\mathbf{w}^t$ and an *active set* $S_t \subset [n]$ of points that are deemed clean at the moment. At every time step, true to the gAM philosophy, AM-RR first fixes the active set and updates the model, and then fixes the model and updates the active set. The first step turns out to be nothing but least squares over the active set. For the second step, it is easy to see that the optimal solution is achieved simply by taking the $n - k$ data points with the smallest residuals (by magnitude) with respect to the updated model and designating them to be the active set.

9.4 A Robust Recovery Guarantee for AM-RR

To simplify the analysis, we will assume that there is no Gaussian noise in the model i.e., $\mathbf{y} = X\mathbf{w}^* + \mathbf{b}^*$ where $\mathbf{b}^*$ is k-sparse. To present the analysis of the AM-RR algorithm, we will need the notions of subset strong convexity and smoothness.

Definition 9.1 (Subset Strong Convexity/Smoothness Property [Bhatia et al., 2015])**.** A matrix $X \in \mathbb{R}^{n \times p}$ is said to satisfy the α_k-subset strong convexity (SSC) property and the β_k-subset smoothness property (SSS) of order k if for all sets $S \subset [n]$ of size $|S| \leq k$, we have, for all $\mathbf{v} \in \mathbb{R}^p$,

$$\alpha_k \cdot \|\mathbf{v}\|_2^2 \leq \left\|X^S \mathbf{v}\right\|_2^2 \leq \beta_k \cdot \|\mathbf{v}\|_2^2.$$

The SSC/SSS properties require that the design matrix formed by taking any subset of k points from the data set of n points act as an approximate isometry on all p dimensional points. These properties are related to the traditional RSC/RSS properties and it can be shown (see for example, [Bhatia et al., 2015]) that RIP-inducing distributions over matrices (see § 7.6) also produce matrices that satisfy the SSC/SSS properties, with high probability. However, it is interesting to note that whereas the RIP definition is concerned with column subsets of the design matrix, SSC/SSS concerns itself with row subsets.

The nature of the SSC/SSS properties is readily seen to be very appropriate for AM-RR to succeed. Since the algorithm uses only a subset of data points to estimate the model vector, it is essential that smaller subsets of data points of size $n-k$ (in particular the true subset of clean points S_*) also allow the model to be recovered. This is equivalent[2] to requiring that the design matrices formed by smaller subsets of data points not identify distinct model vectors. This is exactly what the SSC property demands.

Given this, we can prove the following convergence guarantee for the AM-RR algorithm. The reader would notice that the algorithm, despite being a gAM-style algorithm, does not require precise and careful initialization of the model and active set. This is in stark contrast to other gAM-style approaches we have seen so far, namely EM and AM-MC, both of which demanded careful initialization.

Theorem 9.1. Let $X \in \mathbb{R}^{n \times p}$ satisfy the SSC property at order $n-k$ with parameter α_{n-k} and the SSS property at order k with parameter β_k such that $\beta_k/\alpha_{n-k} < \frac{1}{\sqrt{2}+1}$. Let $\mathbf{w}^* \in \mathbb{R}^p$ be an arbitrary model vector and $\mathbf{y} = X\mathbf{w}^* + \mathbf{b}^*$ where $\left\|\mathbf{b}^*\right\|_0 \leq k$ is a sparse vector of possibly unbounded corruptions. Then AM-RR yields an ϵ-accurate solution $\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq \epsilon$ in no more than $\mathcal{O}\left(\log \frac{\|\mathbf{b}^*\|_2}{\epsilon}\right)$ steps.

Proof. Let $\mathbf{r}^t = \mathbf{y} - X\mathbf{w}^t$ denote the vector of residuals at time t and let $C_t = (X^{S_t})^\top X^{S_t}$ and $S_* = \overline{\operatorname{supp}(\mathbf{b}^*)}$. Then the model update step of AM-RR solves a least squares problem that ensures that

$$\begin{aligned}
\mathbf{w}^{t+1} &= C_t^{-1}(X^{S_t})^\top \mathbf{y}_{S_t} \\
&= C_t^{-1}(X^{S_t})^\top (X^{S_t}\mathbf{w}^* + \mathbf{b}^*_{S_t}) \\
&= \mathbf{w}^* + C_t^{-1}(X^{S_t})^\top \mathbf{b}^*_{S_t}.
\end{aligned}$$

The residuals with respect to this new model can be computed as

$$\mathbf{r}^{t+1} = \mathbf{y} - X\mathbf{w}^{t+1} = \mathbf{b}^* + XC_t^{-1}(X^{S_t})^\top \mathbf{b}^*_{S_t}.$$

However, the active-set update step selects the set with smallest residuals, in particular, ensuring that

$$\left\|\mathbf{r}^{t+1}_{S_{t+1}}\right\|_2^2 \leq \left\|\mathbf{r}^{t+1}_{S_*}\right\|_2^2.$$

[2] See Exercise 9.2.

Plugging in the expression for $\mathbf{r}^{t+1}$ into both sides of the equation, using $\mathbf{b}^*_{\overline{S}_*} = \mathbf{0}$ and the fact that that for any matrix X and vector $\mathbf{v}$ we have

$$\left\|X^S\mathbf{v}\right\|_2^2 - \left\|X^T\mathbf{v}\right\|_2^2 = \left\|X^{S\backslash T}\mathbf{v}\right\|_2^2 - \left\|X^{T\backslash S}\mathbf{v}\right\|_2^2 \leq \left\|X^{S\backslash T}\mathbf{v}\right\|_2^2,$$

gives us, upon some simplification,

$$\begin{aligned}\left\|\mathbf{b}^*_{S_{t+1}}\right\|_2^2 \leq{}& \left\|X^{S_*\backslash S_{t+1}}C_t^{-1}(X^{S_t})^\top \mathbf{b}^*_{S_t}\right\|_2^2 \\ &- 2(\mathbf{b}^*_{S_{t+1}})^\top X^{S_{t+1}}C_t^{-1}(X^{S_t})^\top \mathbf{b}^*_{S_t} \\ \leq{}& \frac{\beta_k^2}{\alpha_{n-k}^2}\left\|\mathbf{b}^*_{S_t}\right\|_2^2 + 2\cdot\frac{\beta_k}{\alpha_{n-k}}\cdot\left\|\mathbf{b}^*_{S_{t+1}}\right\|_2\cdot\left\|\mathbf{b}^*_{S_t}\right\|_2,\end{aligned}$$

where the last step follows from an application of the SSC/SSS properties by noticing that $|S_*\backslash S_{t+1}| \leq k$ and that $\mathbf{b}^*_{S_t}$ and $\mathbf{b}^*_{S_{t+1}}$ are all k-sparse vectors since $\mathbf{b}^*$ itself is a k-sparse vector. Solving the above equation gives us

$$\left\|\mathbf{b}^*_{S_{t+1}}\right\|_2 \leq (\sqrt{2}+1)\cdot\frac{\beta_k}{\alpha_{n-k}}\cdot\left\|\mathbf{b}^*_{S_t}\right\|_2$$

The above result proves that in $t = \mathcal{O}\left(\log\frac{\|\mathbf{b}^*\|_2}{\epsilon}\right)$ iterations, the alternating minimization procedure will identify an active set S_t such that $\left\|\mathbf{b}^*_{S_t}\right\|_2 \leq \epsilon$. It is easy[3] to see that a least squares step on this active set will yield a model $\widehat{\mathbf{w}}$ satisfying

$$\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 = \left\|C_t^{-1}(X^{S_t})^\top \mathbf{b}^*_{S_t}\right\|_2 \leq \frac{\beta_k}{\alpha_{n-k}}\cdot\epsilon \leq \epsilon,$$

since $\beta_k/\alpha_{n-k} < 1$. This concludes the convergence guarantee. □

The crucial assumption in the previous result is the requirement $\beta_k/\alpha_{n-k} < \frac{1}{\sqrt{2}+1}$. Clearly, as $k \to 0$, we have $\beta_k \to 0$ but if the matrix X is well conditioned we still have $\alpha_{n-k} > 0$. Thus, for small enough k, it is assured that we will have $\beta_k/\alpha_{n-k} < \frac{1}{\sqrt{2}+1}$. The point at which this occurs is the so-called *breakdown point* of the algorithm – it is the largest number k such that the algorithm can tolerate k possibly adversarial corruptions and yet guarantee recovery.

[3] See Exercise 9.3.

Note that the quantity $\kappa_k = \beta_k/\alpha_{n-k}$ acts as the effective condition number of the problem. It plays the same role as the condition number did in the analysis of the gPGD and IHT algorithms. It can be shown that for RIP-inducing distributions (see [Bhatia et al., 2015]), AM-RR can tolerate $k = \Omega(n)$ corruptions.

For the specific case of the design matrix being generated from a Gaussian distribution, it can be shown that we have $\alpha_{n-k} = \Omega\left(n-k\right)$ and $\beta_k = \mathcal{O}\left(k\right)$. This in turn can be used to show that we have $\beta_k/\alpha_{n-k} < \frac{1}{\sqrt{2}+1}$ whenever $k \leq n/70$. This means that AM-RR can tolerate up to $n/70$ corruptions when the design matrix is Gaussian. This indicates a high degree of robustness in the algorithm since these corruptions can be completely adversarial in terms of their location, as well as their magnitude. Note that AM-RR is able to ensure this without requiring any specific initialization.

9.5 Alternating Minimization via Gradient Updates

Similar to the gradient-based EM heuristic we looked at in § 5.5, we can make the alternating minimization process in AM-RR much cheaper by executing a gradient step for the alternations. More specifically, we can execute step 3 of AM-RR as

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \cdot \sum_{i \in S_t} (\mathbf{x}_i^\top \mathbf{w} - y_i)\mathbf{x}_i,$$

for some step size parameter η. It can be shown (see [Bhatia et al., 2015]) that this process enjoys the same linear rate of convergence as AM-RR. However, notice that both alternations (model update as well as active set update) in this gradient-descent version can be carried out in (near-)linear time. This is in contrast with AM-RR which takes super-quadratic time in each alternation to discover the least squares solution. In practice, this makes the gradient version much faster (often by an order of magnitude) as compared to the fully corrective version.

However, once we have obtained a reasonably good estimate of S_*, it is better to execute the least squares solution to obtain the final solution in a single stroke. Thus, the gradient and fully corrective steps can be mixed together to great effect. In practice, such *hybrid* techniques offer the fastest convergence. We refer the reader to § 9.7 for a brief discussion on this and to [Bhatia et al., 2015] for details.

9.6 Robust Regression via Projected Gradient Descent

It is possible to devise an alternate formulation for the robust regression problem that allows us to apply the gPGD technique instead. The work of [Bhatia et al., 2017] uses this alternate formulation to arrive at a solution that enjoys consistency properties. Note that if someone gave us a good estimate $\widehat{\mathbf{b}}$ of the corruption vector, we could use it to clean up the responses as $\mathbf{y} - \widehat{\mathbf{b}}$, and re-estimate the model as

$$\widehat{\mathbf{w}}(\widehat{\mathbf{b}}) = \underset{\mathbf{w}\in\mathbb{R}^p}{\arg\min} \left\|(\mathbf{y} - \widehat{\mathbf{b}}) - X\mathbf{w}\right\|_2^2 = (X^\top X)^{-1} X^\top (\mathbf{y} - \widehat{\mathbf{b}}).$$

The residuals corresponding to this new model estimate would be

$$\left\|\mathbf{y} - \widehat{\mathbf{b}} - X \cdot \widehat{\mathbf{w}}(\widehat{\mathbf{b}})\right\|_2^2 = \left\|(I - P_X)(\mathbf{y} - \widehat{\mathbf{b}})\right\|_2^2,$$

where $P_X = X(X^\top X)^{-1}X^\top$. The above calculation shows that an equivalent formulation for the robust regression problem ((ROB-REG)) is the following

$$\min_{\|\mathbf{b}\|_0 = k} \left\|(I - P_X)(\mathbf{y} - \mathbf{b})\right\|_2^2,$$

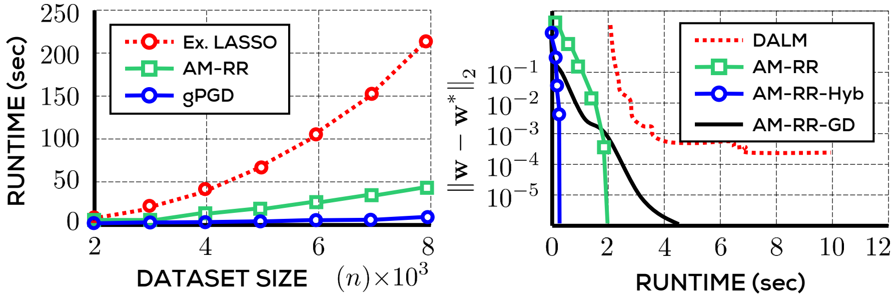
to which we can apply gPGD (Algorithm 2) since it now resembles a sparse recovery problem! This problem enjoys[4] the restricted isometry property whenever the design matrix X is sampled from an RIP-inducing distribution (see § 7.6). This shows that an application of the gPGD technique will guarantee recovery of the optimal corruption vector $\mathbf{b}^*$ at a linear rate. Once we have an ϵ-optimal estimate $\widehat{\mathbf{b}}$ of $\mathbf{b}^*$, a good model $\widehat{\mathbf{w}}$ can be found by solving the least squares problem.

$$\widehat{\mathbf{w}}(\widehat{\mathbf{b}}) = (X^\top X)^{-1} X^\top (\mathbf{y} - \widehat{\mathbf{b}}),$$

as before. It is a simple exercise to show that if $\left\|\widehat{\mathbf{b}} - \mathbf{b}^*\right\|_2 \leq \epsilon$, then

$$\left\|\widehat{\mathbf{w}}(\widehat{\mathbf{b}}) - \mathbf{w}^*\right\| \leq \mathcal{O}\left(\frac{\epsilon}{\alpha}\right),$$

[4] See Exercise 9.4.



(a) Run-time Comparison **(b)** Convergence Comparison

Figure 9.2: An empirical comparison of the performance offered by various approaches for robust regression. Figure 9.2a (adapted from [Bhatia et al., 2017]) compares Extended LASSO, a state-of-the-art relaxation method by Nguyen and Tran [2013b], AM-RR and the gPGD method from § 9.6 on a robust regression problem in $p = 1000$ dimensions with 30% data points corrupted. Non-convex techniques such as AM-RR and gPGD are more than an order of magnitude faster, and scale much better, than Extended LASSO. Figure 9.2b (adapted from [Bhatia et al., 2015]) compares various solvers on a robust regression problem in $p = 300$ dimensions with 1800 data points of which 40% are corrupted. The solvers include the gAM-style solver AM-RR, a variant using gradient-based updates, a hybrid method (see § 9.5), and the DALM method [Yang et al., 2013], a state-of-the-art solver for relaxed LASSO-style formulations. The hybrid method is the fastest of all the techniques. In general, all AM-RR variants are much faster than the relaxation-based method.

where α is the RSC parameter of the problem. Note that this shows how the gPGD technique can be applied to perform recovery not only when the parameter is sparse in the model domain (for instance in the gene expression analysis problem), but also when the parameter is sparse in the data domain, as in the robust regression example.

9.7 Empirical Comparison

Before concluding, we present some discussion on the empirical performance of various algorithms on the robust regression problem. Figure 9.2a compares the running times of various robust regression approaches on synthetic problems as the number of data points available in the dataset increases. Similar trends are seen if the data dimensionality increases. The graph shows that non-convex techniques such as

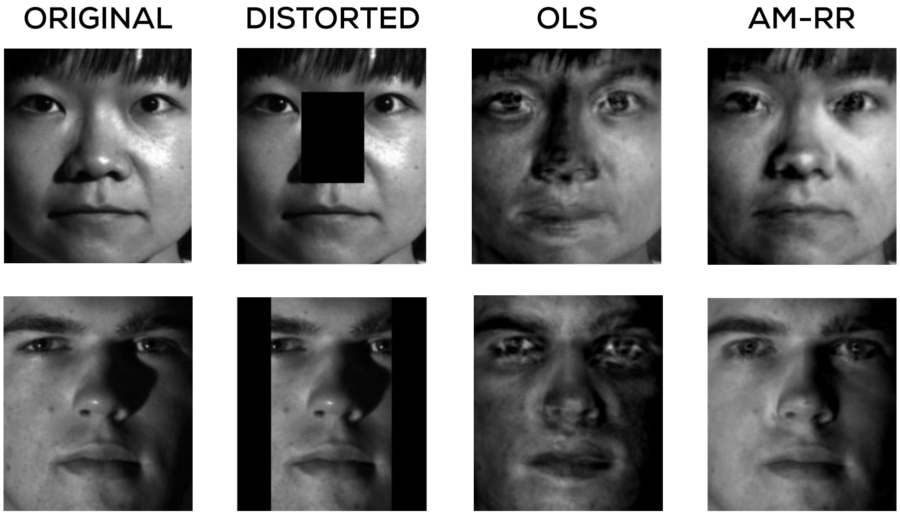


Figure 9.3: An experiment on face reconstruction using robust regression techniques. Two face images were taken and different occlusions were applied to them. Using the model described in § 9.1, reconstruction was attempted using both, ordinary least squares (OLS) and robust regression (AM-RR). It is clear that AM-RR achieves far superior reconstruction of the images and is able to correctly figure out the locations of the occlusions. Images courtesy the Yale Face Database B.

AM-RR and gPGD offer much more scalable solutions than relaxation-based methods. Figure 9.2b similarly demonstrates how variants of the basic AM-RR procedure can offer significantly faster convergence. Figure 9.3 looks at a realistic example of face reconstruction under occlusions and demonstrates how AM-RR is able to successfully recover the face image even when significant portions of the face are occluded.

9.8 Exercises

Exercise 9.1. Show that it is impossible to recover the model vector if a fully adaptive adversary is able to corrupt more than half the responses i.e., if $k \geq n/2$. A fully adaptive adversary is one that is allowed to perform corruptions after observing the clean covariates as well as the uncorrupted responses.
Hint: The adversary can make it impossible to distinguish between two models, the real model, and another one of its choosing.

Exercise 9.2. Show that the SSC property ensures that there exists no subset S of the data, $|S| \leq k$ and no two distinct model vectors $\mathbf{v}^1, \mathbf{v}^2 \in \mathbb{R}^p$ such that $X^S\mathbf{v}^1 = X^S\mathbf{v}^2$.

Exercise 9.3. Show that executing the AM-RR algorithm for a single step starting from an active set S_t such that $\left\|\mathbf{b}^*_{S_t}\right\|_2 \leq \epsilon$ ensures in the very next step that $\left\|\mathbf{w}^t - \mathbf{w}^*\right\|_2 \leq \epsilon$.

Exercise 9.4. Show that if the design matrix X satisfies RIP, then the objective function $f(\mathbf{b}) = \|(I - P_X)(\mathbf{y} - \mathbf{b})\|_2^2$, enjoys RIP (of the same order as X but with possibly different constants) as well.

Exercise 9.5. Prove that ((ROB-REG)) and ((ROB-REG-2)) are equivalent formulations i.e., they yield the same model.

9.9 Bibliographic Notes

The problem of robust estimation has been well studied in the statistics community. Indeed, there exist entire texts devoted to this area [Rousseeuw and Leroy, 1987, Maronna et al., 2006] which look at robust estimators for regression and other problems. However, these methods often involve estimators, such as the least median of squares estimator, that have an exponential time complexity.

It is notable that these infeasible estimators often have attractive theoretical properties such as a high breakdown point. For instance, the work of Rousseeuw [1984] shows that the least median of squares method enjoys a breakdown point of as high as $n/2-p$. In contrast, AM-RR is only able to handle $n/70$ errors. However, whereas the gradient descent version of AM-RR can be executed in near-linear time, the least median of squares method requires time exponential in p.

There have been relaxation based approaches to solving robust regression and time series problems as well. Chief of them include the works of Chen and Dalalyan [2012], Chen et al. [2013] which look at the Dantzig selector methods and the *trimmed product* techniques to perform estimation, and the works of Wright et al. [2009], Nguyen and Tran [2013a]. The work of Chen et al. [2013] considers corruptions not only in the responses, but in the covariates as well. However, these methods tend to scale poorly to really large scale problems owing to the non-smooth nature of the optimization problems that they end up solving. The non-convex optimization techniques we have studied, on the other hand, require linear time or else have closed-form updates.

Recent years have seen the application of non-convex techniques to robust estimation. However these works can be traced back to the classical work of Fischler and Bolles [1981] that developed the RANSAC algorithm that is very widely used in fields such as computer vision. The RANSAC algorithm samples multiple candidate active sets and returns the least squares estimate on the set with least residual error.

Although the RANSAC method does not enjoy strong theoretical guarantees in the face of an adaptive adversary and a large number of corruptions, the method is seen to work well when there are very few outliers. Later works, such as that of She and Owen [2011] applied soft-thresholding techniques to the problem followed by the work of Bhatia et al. [2015] which applied the alternating minimization algorithm we studied here. Bhatia et al. [2015] also looked at the problem of robust sparse recovery.

The time series literature has also seen the application of various techniques for robust estimation including robust M-estimators in both the additive and innovative outlier models [Martin and Zeh, 1978, Stockinger and Dutter, 1987] and least trimmed squares [Croux and Joossens, 2008].

10

Phase Retrieval

In this section, we will take a look at the phase retrieval problem, a non-convex optimization problem with applications in several domains. We briefly mentioned this problem in § 5 when we were discussing mixed regression. At a high level, phase retrieval is equivalent to discovering a complex signal using observations that reveal only the magnitudes of (complex) linear measurements over that signal. The phases of the measurements are not revealed to us.

Over reals, this reduces to solving a system of quadratic equations which is known to be computationally intractable to solve exactly. Fortunately however, typical phase retrieval systems usually require solving quadratic systems that have nice randomized structures in the coefficients of the quadratic equations. These structures can be exploited to efficiently solve these systems. In this section we will look at various algorithms that achieve this.

10.1 Motivating Applications

The problem of phase retrieval arises in areas of signal processing where the phase of a signal being measured is irrevocably lost, or at best, unreliably obtained. The recovery of the signal in such situations becomes closely linked to our ability to perform phase retrieval.

X-ray Crystallography The goal in X-ray crystallography is to find the structure of a small molecule by bombarding it with X-rays from various angles and measuring the trajectories and intensities of the diffracted photons on a film. These quantities can be used to glean the internal three-dimensional structure of the electron cloud within the crystal, revealing the atomic arrangements therein. This technique has been found to be immensely useful in imaging specimens with a crystalline structure and has been historically significant in revealing the interior composition of several compounds, both inorganic and organic.

A notable example is that of nucleic acids such as DNA whose structure was revealed in the seminal work of Franklin and Gosling [1953a] which immediately led Francis and Crick to propose its double helix structure. The reader may be intrigued by the now famous *Photo 51* which provided critical support to the helical structure-theory of DNA [Franklin and Gosling, 1953b]. We refer the reader to the expository work of Lucas [2008] for a technical and historical account of how these discoveries came to be.

Transmission Electron Microscopy (TEM) This technique uses a focused beam of electrons instead of high energy photons to image the object of study. This technique works best with ultra thin specimens which do not absorb electrons but let the beam of electrons pass through. The electrons interact with the atomic structure of the specimen and are captured at the other end using a sensing mechanism. TEM is capable of resolutions far higher than those possible using photonic imaging techniques due to the extremely small de-Broglie wavelength of electrons.

Coherent Diffraction Imaging (CDI) This is a widely used technique for studying nanostructures such as nanotubes, nanocrystals and the like. A highly coherent beam of X-rays is made incident on the object of study and the diffracted rays allowed to interfere to produce a diffraction pattern which is used to recover the structure of the object. A key differentiating factor in CDI is the absence of any lenses to focus light onto the specimen, as opposed to other methods such as TEM/X-Ray crystallography which use optical or electromagnetic lenses to focus the incident beam and then refocus the diffracted beam. The absence of any lenses in CDI is very advantageous since it results in aberration-free patterns. Moreover, this way the resolution of the technique is only dependent on the wavelength and other properties of the incident rays rather than the material of the lens etc.

10.2 Problem Formulation

Bombarding a structure with X-rays or other beams can be shown to be equivalent to taking random Fourier measurements of its internal structure. More specifically, let $\mathbf{w}^* \in \mathbb{C}^p$ denote the vector that represents the density of electrons throughout the crystal, signifying its internal structure. Then, under simplifying regularity assumptions such as perfect periodicity, X-ray crystallography can be modeled as transforming $\mathbf{w}^*$ into $\mathbf{y} = X\mathbf{w}^* \in \mathbb{C}^n$ where $X \in \mathbb{C}^{n \times p}$ has each of its rows sampled from a Fourier measurement matrix, i.e, $X_{ab} = \exp\left(-\frac{2\pi(a-1)(b-1)}{p}\right), b = 1, \ldots, p$ for some random $a \in [p]$.

Unfortunately, the measurements made by the sensors in the above applications are not able to observe the Fourier measurements exactly. Instead of observing complex valued transformations $y_k \in \mathbb{C}$ of the signal $\mathbf{y}$, all we observe are the real value magnitudes $|y_k| \in \mathbb{R}$, the phase being lost in the signal acquisition process. A natural question arises whether it is still possible to recover $\mathbf{w}^*$ or not.

Note that if the signal and the measurement matrices are real then $|y_k|^2 = (\mathbf{x}_k^\top \mathbf{w}^*)^2$. Thus, the goal is simply to recover a model $\mathbf{w}$ by solving a system of n quadratic equations described above. Although problem of solving a system of quadratic equations is intractable in general, in the special case of phase retrieval, it is possible to exploit the fact that the vectors $\mathbf{x}_k$ are sampled randomly, to develop recovery algorithms that are efficient, as well as require only a small number n of measurements.

We note that the state of the art in phase retrieval literature is yet unable to guarantee recovery from random Fourier measurements, which closely model situations arising in crystallography and other imaging applications. Current analyses mostly consider the case when the sensing matrix X has random Gaussian entries. A notable exception to the above is the work of Candès et al. [2015] which is able to address *coded diffraction* measurements. These are more relevant to what is used in practice but still complicated to design. For sake of simplicity, we will only focus on the Gaussian measurement case in our discussions. This would allow us to study the algorithmic techniques used to solve the problem without getting involved in the intricacies of Fourier or coded diffraction measurements.

To formalize our problem statement, we let $\mathbf{w}^* \in \mathbb{C}^p$ be an unknown p-dimensional signal, $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top \in \mathbb{C}^{n \times p}$ be a measurement matrix. Our goal is to recover $\mathbf{w}^*$ given X and $|\mathbf{y}|$, where $\mathbf{y} = [y_1, y_2, \ldots, y_k]^\top \in \mathbb{C}^n = X\mathbf{w}^*$ and $|y_k| = \left|\mathbf{x}_i^\top \mathbf{w}^*\right|$. Our focus would be on the special case where $X_{kj} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1) + i \cdot \mathcal{N}(0,1)$ where $i^2 = -1$. We would like to recover $\mathbf{w}^*$ efficiently using only $n = \widetilde{\mathcal{O}}(p)$ measurements. We will study how gAM and gPGD-style techniques can be used to solve this problem. We will point to approaches using the relaxation technique in the bibliographic notes.

Notation: We will abuse the notation $\mathbf{x}^\top$ to denote the complex row-conjugate of a complex vector $\mathbf{x} \in \mathbb{C}^p$, something that is usually denoted by $\mathbf{x}^*$. A random vector $\mathbf{x} = \mathbf{a} + i\mathbf{b} \in \mathbb{C}^n$ will be said to be distributed according to the standard Gaussian distribution over $\mathbb{C}^n$, denoted as $\mathcal{N}_{\mathbb{C}}(\mathbf{0}, I)$ if $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are independently distributed as standard (real valued) Gaussian vectors i.e., $\mathbf{a}, \mathbf{b} \sim \mathcal{N}(\mathbf{0}, I_n)$.

10.3 Phase Retrieval via Alternating Minimization

One of the most popular approaches for solving the phase retrieval problem is a simple application of the gAM approach, first proposed by Gerchberg and Saxton [1972] more than 4 decades ago. The intuition behind the approach is routine – there exist two parameters of

Algorithm 12 Gerchberg-Saxton Alternating Minimization (GSAM)

```
Input: Measurement matrix $X \in \mathbb{C}^{n \times p}$, observed response magnitudes
    $|\mathbf{y}| \in \mathbb{R}^n_+$, desired accuracy $\epsilon$
Output: A signal $\widehat{\mathbf{w}} \in \mathbb{C}^p$
Set $T \leftarrow \log 1/\epsilon$
Partition $n$ data points into $T+1$ sets $S_0, S_1, \ldots, S_T$
$\mathbf{w}^0 \leftarrow \text{eig}\left(\frac{1}{|S_0|}\sum_{k \in S_0} |y_k|^2 \cdot \mathbf{x}_k\mathbf{x}_k^\top, 1\right)$        //Leading eigenvector
for $t = 1, 2, \ldots, T$ do
    Phase Estimation: $\phi_k = \mathbf{x}_k^\top \mathbf{w}^{t-1}/|\mathbf{x}_k^\top \mathbf{w}^{t-1}|$, for all $k \in S_t$
    Signal Estimation: $\mathbf{w}^t = \arg\min_{\mathbf{w} \in \mathbb{C}^p} \sum_{k \in S_t} ||y_k| \cdot \phi_k - \mathbf{x}_k^\top \mathbf{w}|^2$
end for
return $\mathbf{w}^T$
```

interest in the problem – the phase of the data points i.e., $y_k/\,|y_k|$, and the true signal $\mathbf{w}^*$. Just as before, we observe a two-way connection between these parameters. Given the true phase values of the points $\phi_k = y_k/|y_k|$, the signal $\mathbf{w}^*$ can be recovered simply by solving a system of linear equations: $\phi_k \cdot |y_k| = \mathbf{x}_k^\top \mathbf{w}, k = 1, \ldots, n$. On the other hand, given $\mathbf{w}^*$, estimating the phase of the points is straightforward as $\phi_k = (\mathbf{x}_k^\top \mathbf{w}^*)/|\mathbf{x}_k^\top \mathbf{w}^*|$.

Given the above, a gAM-style approach naturally arises: we alternately estimate ϕ_k and $\mathbf{w}^*$. More precisely, at the t-th step, we first estimate ϕ_k^t using $\mathbf{w}^{t-1}$ as $\phi_k^t = (\mathbf{x}_k^\top \mathbf{w}^{t-1})/|\mathbf{x}_k^\top \mathbf{w}^{t-1}|$ and then update our estimate of $\mathbf{w}$ by solving a least squares problem $\mathbf{w}^t = \arg\min_{\mathbf{w}} \sum_k |\phi_k^t|y_k| - \mathbf{x}_k^\top \mathbf{w}|^2$ over complex variables. Algorithm 12 presents the details of this *Gerchberg-Saxton Alternating Minimization* (GSAM) method. To avoid correlations, the algorithm performs these alternations on distinct sets of points at each time step. These disjoint sets can be created by sub-sampling the overall available data.

In their original work, Gerchberg and Saxton [1972] proposed to use a random vector $\mathbf{w}^0$ for initialization. However, the recent work of Netrapalli et al. [2013] demonstrated that a more careful initialization, in particular the largest eigenvector of $M = \sum_k |y_k|^2 \cdot \mathbf{x}_k\mathbf{x}_k^\top$, is beneficial. Such a spectral initialization leads to an initial solution that is already at most a (small) constant distance away from the optimal solution $\mathbf{w}^*$.

As we have seen to be the case with most gAM-style approaches, including the EM algorithm, this approximately optimal initialization is crucial to allow the alternating minimization procedure to take over and push the iterates toward the globally optimal solution.

Notions of Convergence: Before we proceed to give convergence guarantees for the GSAM algorithm, note that an exact recovery of $\mathbf{w}^*$ is impossible, since phase information is totally lost. More specifically, two signals $\mathbf{w}^*$ and $e^{i\theta} \cdot \mathbf{w}^*$ for some $\theta \in \mathbb{R}$ will generate exactly the same responses when phase information is eliminated. Thus, the best we can hope for is to recover $\mathbf{w}^*$ up to a phase shift. There are several ways of formalizing notions of convergence modulo a phase shift. We also note that complete proofs of the convergence results will be tedious and hence we will only give proof sketches for them.

10.4 A Phase Retrieval Guarantee for GSAM

While the GSAM approach has been known for decades, its convergence properties and rates were not understood until the work of Netrapalli et al. [2013] who analyzed the heuristic (with spectral initialization as described in Algorithm 12) for Gaussian measurement matrices. In particular, they proved that GSAM recovers an ϵ-optimal estimate of $\mathbf{w}^*$ in roughly $T = \log(1/\epsilon)$ iterations so long as the number of measurements satisfies $n = \widetilde{\Omega}\left(p \log^3 p/\epsilon\right)$.

This result is established by first proving a linear convergence guarantee for the GSAM procedure assuming a sufficiently nice initialization. Next, it is shown that the spectral initialization described in Algorithm 12 does indeed satisfy this condition.

Linear Convergence: For this part, it is assumed that the GSAM procedure has been initialized at $\mathbf{w}^0$ such that $\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2 \leq \frac{1}{100}$. The following result (which we state without proof) shows that the alternating procedure hereafter ensures a linear rate of convergence.

Theorem 10.1. Let $y_k = \mathbf{x}_k^\top \mathbf{w}^*$ for $k = 1, \ldots, n$ where $\mathbf{x}_k \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, I)$ and $n \geq C \cdot p \log^3(p/\epsilon)$ for a suitably large constant C. Then, if the initialization satisfies $\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2 \leq \frac{1}{100}$, then with probability at least $1 - 1/n^2$, GSAM outputs an ϵ-accurate solution $\left\|\mathbf{w}^T - \mathbf{w}^*\right\|_2 \leq \epsilon$ in no more than $T = \mathcal{O}\left(\log \frac{\|\mathbf{y}\|_2}{\epsilon}\right)$ steps.

In practice, one can use fast approximate solvers such as the conjugate gradient method to solve the least squares problem at each iteration. These take $\mathcal{O}\left(np\log(1/\epsilon)\right)$ time to solve a least squares instance. Since $n = \widetilde{\mathcal{O}}\left(p\log^3 p\right)$ samples are enough, the GSAM algorithm operates with computation time at most $\widetilde{\mathcal{O}}\left(p^2\log^3(p/\epsilon)\right)$.

Initialization: We now establish the utility of the initialization step. The proof hinges on a simple observation. Consider the random variable $Z = |y|^2 \cdot \mathbf{x}\mathbf{x}^\top$ corresponding to a randomly chosen vector $\mathbf{x} \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, I)$ and $y = \mathbf{x}^\top\mathbf{w}^*$. For sake of simplicity, let us assume that $\left\|\mathbf{w}^*\right\|_2 = 1$. Since $\mathbf{x} = [x_1, x_2, \ldots, x_p]^\top$ is a spherically symmetric vector in the complex space $\mathbb{C}^p$, the random variable $\mathbf{x}^\top\mathbf{w}^*$ has an identical distribution as the vector $e^{i\theta} \cdot \mathbf{x}^\top\mathbf{e}_1$ where $\mathbf{e}_1 = [1, 0, 0, \ldots, 0]^\top$, for any $\theta \in \mathbb{R}$. Using the above, it is easy to see that

$$\mathbb{E}\left[|y|^2 \cdot \mathbf{x}\mathbf{x}^\top\right] = \mathbb{E}\left[|x_1|^2 \cdot \mathbf{x}\mathbf{x}^\top\right] = 4 \cdot \mathbf{e}_1\mathbf{e}_1^\top + 4 \cdot I$$

Using a slightly more tedious calculation involving unitary transformations, we can extend the above to show that, in general,

$$\mathbb{E}\left[|y|^2 \cdot \mathbf{x}\mathbf{x}^\top\right] = 4 \cdot \mathbf{w}^*(\mathbf{w}^*)^\top + 4 \cdot I =: D$$

The above clearly indicates that the largest eigenvector of the matrix D is along $\mathbf{w}^*$. Now notice that the matrix whose leading eigenvector we are interested in during initialization,

$$S := \frac{1}{|S_0|}\sum_{k \in S_0} |y_k|^2 \cdot \mathbf{x}_k\mathbf{x}_k^\top,$$

is simply an empirical estimate to the expectation $\mathbb{E}\left[|y|^2 \cdot \mathbf{x}\mathbf{x}^\top\right] = D$. Indeed, we have $\mathbb{E}\left[S\right] = D$. Thus, it is reasonable to expect that the leading eigenvector of S would also be aligned to $\mathbf{w}^*$. We can make this statement precise using results from the concentration of finite sums of self-adjoint independent random matrices from [Tropp, 2012].

Theorem 10.2. The spectral initialization method (Step 3 of Algorithm 12), with probability at least $1 - 1/|S_0|^2 \leq 1 - 1/p^2$, ensures an initialization $\mathbf{w}^0$ such that $\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2 \leq c$ for any constant $c > 0$, so long as it is executed with a randomly chosen set S_0 of data points of size $|S_0| \geq C \cdot p \log p$ for a suitably large constant C depending on c.

Proof. To make the analysis simple, we will continue to assume that $\mathbf{w}^* = e^{i\theta} \cdot \mathbf{e}_1$ for some $\theta \in \mathbb{R}$. We can use Bernstein-style results for matrix concentration (for instance, see [Tropp, 2012, Theorem 1.5]) to show that for any chosen constant $c > 0$, if $n \geq C \cdot p \log p$ for a large enough constant C that depends on the constant c, then with probability at least $1 - 1/|S_0|^2$, we have

$$\|S - D\|_2 \leq c$$

Note that the norm being used above is the spectral/operator norm on matrices. Given this, it is possible to get a handle on the leading eigenvalue of S. Observe that since $\mathbf{w}^0$ is the leading eigenvector of S, and since we have assumed $\mathbf{w}^* = e^{i\theta} \cdot \mathbf{e}_1$, we have

$$\begin{aligned}
\left|\left\langle \mathbf{w}^0, S\mathbf{w}^0 \right\rangle\right| &\geq \left|\left\langle \mathbf{w}^*, S\mathbf{w}^* \right\rangle\right| \\
&\geq \left|\left\langle \mathbf{w}^*, D\mathbf{w}^* \right\rangle\right| - \left|\left\langle \mathbf{w}^*, (S - D)\mathbf{w}^* \right\rangle\right| \\
&\geq 8 - \|S - D\|_2 \left\|\mathbf{w}^*\right\|_2^2 \\
&\geq 8 - c
\end{aligned}$$

On the other hand, using the triangle inequality again, we have

$$\begin{aligned}
\left|\left\langle \mathbf{w}^0, S\mathbf{w}^0 \right\rangle\right| &= \left|\left\langle \mathbf{w}^0, (S - D)\mathbf{w}^0 \right\rangle + \left\langle \mathbf{w}^0, D\mathbf{w}^0 \right\rangle\right| \\
&\leq \|S - D\|_2 \left\|\mathbf{w}^0\right\|_2^2 + 4\left|\mathbf{w}_1^0\right|^2 + 4\left\|\mathbf{w}^0\right\|_2^2 \\
&= c + 4\left|\mathbf{w}_1^0\right|^2 + 4,
\end{aligned}$$

where in the second step we have used $D = 4 \cdot \mathbf{e}_1\mathbf{e}_1^\top + 4 \cdot I$. The two opposing inequalities on the quantity $\left|\left\langle \mathbf{w}^0, S\mathbf{w}^0 \right\rangle\right|$ give us $\left|\mathbf{w}_1^0\right|^2 \geq 1 - c/2$. Noticing that $\left|\mathbf{w}_1^0\right|^2 = \left|\left\langle \mathbf{w}^0, \mathbf{e}_1 \right\rangle\right| = \left|\left\langle \mathbf{w}^0, \mathbf{w}^* \right\rangle\right|$ then establishes

$$\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2^2 = 2(1 - \left|\left\langle \mathbf{w}^0, \mathbf{w}^* \right\rangle\right|) \leq 2(1 - \sqrt{1 - c/2}) \leq c/2$$

which finishes the proof. □

Algorithm 13 Wirtinger's Flow for Phase Retrieval (WF)

```
Input: Measurement matrix $X \in \mathbb{C}^{n \times p}$, observed response magnitudes
    $|\mathbf{y}| \in \mathbb{R}^n_+$, step size $\eta$
Output: A signal $\widehat{\mathbf{w}} \in \mathbb{C}^p$
$\mathbf{w}^0 \leftarrow= \text{eig}(\frac{1}{n}\sum_{k=1}^n |y_k|^2 \cdot \mathbf{x}_k\mathbf{x}_k^\top, 1)$    //Leading eigenvector
for $t = 1, 2, \ldots$ do
    $\mathbf{w}^t \leftarrow \mathbf{w}^{t-1} - 2\eta \cdot \sum_k (|\mathbf{x}_k^\top \mathbf{w}^{t-1}|^2 - |y_k|^2)\mathbf{x}_k\mathbf{x}_k^\top \mathbf{w}^{t-1}$
end for
return $\mathbf{w}^t$
```

10.5 Phase Retrieval via Gradient Descent

There also exists a relatively straightforward reformulation of the phase retrieval problem that allows a simple gPGD-style approach to be applied. The recent work of Candès et al. [2015] did this by reformulating the phase retrieval problem in terms of the following objective

$$f(\mathbf{w}) = \sum_{k=1}^{n} \left(|y_k|^2 - |\mathbf{x}_k^\top \mathbf{w}|^2 \right)^2,$$

and then performing gradient descent (over complex variables) on this unconstrained optimization problem. In the same work, this technique was named the *Wirtinger's flow* algorithm, presumably as a reference to the notions of Wirtinger derivatives, and shown to offer provable convergence to the global optimum, just like the Gerchberg-Saxton method, when initialization is performed using a spectral method. Algorithm 13 outlines the Wirtinger's Flow (WF) algorithm. Note that WF offers accelerated update times. Note that unlike the GSAM approach, the WF algorithm does not require sub-sampling but needs to choose a step size parameter instead.

10.6 A Phase Retrieval Guarantee for WF

Since the Wirtinger's Flow algorithm uses an initialization technique that is identical to that used by the Gerchberg-Saxton method, we can straightaway apply Theorem 10.2 to assure ourselves that with probability at least $1-1/n^2$, we have $\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2 \leq c$ for any constant $c > 0$. Starting from such an initial point, Candès et al. [2015] argue that each step of the gradient descent procedure decreases the distance to optima by at least a constant (multiplicative) factor. This allows a linear convergence result to be established for the WF procedure, similar to the GSAM approach, however, with each iteration being much less expensive, being a gradient descent step, rather than the solution to a complex-valued least squares problem.

Theorem 10.3. Let $y_k = \mathbf{x}_k^\top \mathbf{w}^*$ for $k = 1, \ldots, n$ where $\mathbf{x}_k \sim \mathcal{N}_{\mathbb{C}}(\mathbf{0}, I)$. Also, let $n \geq C \cdot p \log p$ for a suitably large constant C. Then, if the initialization satisfies $\left\|\mathbf{w}^0 - \mathbf{w}^*\right\|_2 \leq \frac{1}{100}$, then with probability at least $1 - 1/p^2$, WF outputs an ϵ-accurate solution $\left\|\mathbf{w}^T - \mathbf{w}^*\right\|_2 \leq \epsilon$ in no more than $T = \mathcal{O}\left(\log \frac{\|\mathbf{y}\|_2}{\epsilon}\right)$ steps.

Candès et al. [2015] also studied a coded diffraction pattern (CDP) model which uses measurements X that are more "practical" for X-ray crystallography style applications and based on a combination of random multiplicative perturbations of a standard Fourier measurement. For such measurements, Candès et al. [2015] provided a result similar to Theorem 10.3 but with a slightly inferior rate of convergence: the new procedure is able to guarantee an ϵ-optimal solution only after $T = \mathcal{O}\left(p \cdot \log \frac{\|\mathbf{y}\|_2}{\epsilon}\right)$ steps, i.e., a multiplicative factor of p larger than that required by the WF algorithm for Gaussian measurements.

10.7 Bibliographic Notes

The phase retrieval problem has been studied extensively in the X-ray crystallography literature where the focus is mostly on studying Fourier measurement matrices [Candès and Li, 2014]. For Gaussian measurements, initial results were obtained using a technique called *Phase-lift*, which essentially viewed the quadratic equation $|y_k|^2 = (\mathbf{x}_k^\top \mathbf{w}^*)^2$ as the linear measurement of a rank-one matrix, i.e., $|y_k|^2 = \left\langle \mathbf{x}_k \mathbf{x}_k^\top, W^* \right\rangle$ where $W^* = \mathbf{w}^*(\mathbf{w}^*)^\top$.

This rank-one constraint was then replaced by a nuclear norm constraint and the resulting problem was solved as a semi-definite program

(SDP). This technique was shown to achieve the information theoretically optimal sample complexity of $n = \Omega(p)$ (recall that the GSAM and WF techniques require $n = \Omega(p \log p)$. However, the running time of the Phase-lift algorithm is prohibitive at $O(np^2 + p^3)$.

In addition to the standard phase retrieval problem, several works [Netrapalli et al., 2013, Jaganathan et al., 2013] have also studied the sparse phase retrieval problem where the goal is to recover a sparse signal $\mathbf{w}^* \in \mathbb{C}^p$, with $\|\mathbf{w}^*\|_0 \leq s \ll p$, using only magnitude measurements $|y_k| = |\mathbf{x}_k^\top \mathbf{w}^*|$. The best known results for such problems require $n \geq s^3 \log p$ measurements for s-sparse signals. This is significantly worse than information theoretically optimal $\mathcal{O}(s \log p)$ number of measurements. However, Jaganathan et al. [2013] showed that for a phase-lift style technique, one cannot hope to solve the problem using less than $\mathcal{O}\left(s^2 \log p\right)$ measurements.

References

Alekh Agarwal, Sahand N. Negahban, and Martin J. Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40(5):2452–2482, 2012.

Alekh Agarwal, Animashree Anandkumar, Prateek Jain, and Praneeth Netrapalli. Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization. *SIAM Journal of Optimization*, 26(4):2775–2799, 2016.

Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. Finding Approximate Local Minima Faster than Gradient Descent. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2017.

Animashree Anandkumar and Rong Ge. Efficient approaches for escaping higher order saddle points in non-convex optimization. In *Proceedings of the 29th Conference on Learning Theory (COLT)*, pages 81–102, 2016.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor Decompositions for Learning Latent Variable Models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

Andreas Andresen and Vladimir Spokoiny. Convergence of an Alternating Maximization Procedure. *Journal of Machine Learning Research*, 17:1–53, 2016.

Sanjeev Arora, Rong Ge, and Ankur Moitra. New Algorithms for Learning Incoherent and Overcomplete Dictionaries. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, 2014.

Kamyar Azizzadenesheli, Alessandro Lazaric, and Anima Anandkumar. Reinforcement Learning of POMDPs using Spectral Methods. In *Proceedings of the 29th Conference on Learning Theory (COLT)*, 2016.

Sivaraman Balakrishnan, Martin J. Wainwright, and Bin Yu. Statistical Guarantees for the EM Algorithm: From Population to Sample-based Analysis. *Annals of Statistics*, 45(1):77–120, 2017.

Richard Baraniuk, Mark Davenport, Ronald DeVore, and Michael Wakin. A Simple Proof of the Restricted Isometry Property for Random Matrices. *Constructive Approximation*, 28(3):253–263, 2008.

Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 3rd edition, 2016.

Kush Bhatia, Prateek Jain, and Purushottam Kar. Robust Regression via Hard Thresholding. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

Kush Bhatia, Prateek Jain, Parameswaran Kamalaruban, and Purushottam Kar. Consistent Robust Regression. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

Srinadh Bhojanapalli and Prateek Jain. Universal Matrix Completion. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Thomas Blumensath. Sampling and Reconstructing Signals From a Union of Linear Subspaces. *IEEE Transactions on Information Theory*, 57(7): 4660–4671, 2011.

Jean Bourgain, Stephen Dilworth, Kevin Ford, Sergei Konyagin, and Denka Kutzarova. Explicit constructions of RIP matrices and related problems. *Duke Mathematical Journal*, 159(1):145–185, 2011.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Alon Brutzkus and Amir Globerson. Globally Optimal Gradient Descent for a ConvNet with Gaussian Inputs. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Sebastien Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(34):231–357, 2015.

Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A Singular Value Thresholding Algorithm for Matrix Completion. *SIAM Journal of Optimization*, 20(4):1956–1982, 2010.

Emmanuel Candès and Terence Tao. Decoding by Linear Programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

Emmanuel J. Candès. The Restricted Isometry Property and Its Implications for Compressed Sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

Emmanuel J. Candès and Xiaodong Li. Solving Quadratic Equations via PhaseLift When There Are About as Many Equations as Unknowns. *Foundations of Computational Mathematics*, 14(5):1017–1026, 2014.

Emmanuel J. Candès and Benjamin Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6): 717–772, 2009.

Emmanuel J. Candès and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56 (5):2053–2080, 2009.

Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Stable Signal Recovery from Incomplete and Inaccurate Measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

Emmanuel J. Candès, Xiaodong Li, and Mahdi Soltanolkotabi. Phase Retrieval via Wirtinger Flow: Theory and Algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

Yair Carmon, John C. Duchi, Oliver Hinder, and Aaron Sidford. "Convex Until Proven Guilty": Dimension-Free Acceleration of Gradient Descent on Non-Convex Functions. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Rick Chartrand. Exact Reconstruction of Sparse Signals via Nonconvex Minimization. *IEEE Information Processing Letters*, 14(10):707–710, 2007.

Caihua Chen and Bingsheng He. Matrix Completion via an Alternating Direction Method. *IMA Journal of Numerical Analysis*, 32(1):227–245, 2012.

Laming Chen and Yuantao Gu. Local and global optimality of LP minimization for sparse recovery. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

Yin Chen and Arnak S. Dalalyan. Fused sparsity and robust estimation for linear models with unknown variance. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, 2012.

Yudong Chen, Constantine Caramanis, and Shie Mannor. Robust Sparse Regression under Adversarial Corruption. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

Yudong Chen, Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Matrix Completion with Column Manipulation: Near-Optimal Sample-Robustness-Rank Tradeoffs. *IEEE Transactions on Information Theory*, 62(1):503–526, 2016.

Yeshwanth Cherapanamjeri, Kartik Gupta, and Prateek Jain. Nearly-optimal Robust Matrix Completion. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Anna Choromanska, Mikael Hena, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The Loss Surfaces of Multilayer Networks. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Compressed Sensing and Best k-term Approximation. *Journal of the American Mathematical Society*, 22(1):211–231, 2009.

Christophe Croux and Kristel Joossens. Robust Estimation of the Vector Autoregressive Model by a Least Trimmed Squares Procedure. In *Proceedings in Computational Statistics (COMPSTAT)*, 2008.

Yann N. Dauphin, Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 2933–2941, 2014.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

David L. Donoho. Compressed Sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

David L. Donoho, Arian Maleki, and Andrea Montanari. Message Passing Algorithms for Compressed Sensing: I. Motivation and Construction. *Proceedings of the National Academy of Sciences USA*, 106(45):18914–18919, 2009.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

Maryam Fazel, Ting Kei Pong, Defeng Sun, and Paul Tseng. Hankel matrix rank minimization with applications in system identification and realization. *SIAM Journal on Matrix Analysis and Applications*, 34(3):946–977, 2013.

Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Simon Foucart. A Note on Guaranteed Sparse Recovery via ℓ_1-minimization. *Applied and Computational Harmonic Analysis*, 29(1):97–103, 2010.

Simon Foucart. Hard Thresholding Pursuit: an Algorithm for Compressive Sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563, 2011.

Simon Foucart and Ming-Jun Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

Rosalind Franklin and Raymond G. Gosling. Evidence for 2-Chain Helix in Crystalline Structure of Sodium Deoxyribonucleate. *Nature*, 172:156–157, 1953a.

Rosalind Franklin and Raymond G. Gosling. Molecular Configuration in Sodium Thymonucleate. *Nature*, 171:740–741, 1953b.

Rahul Garg and Rohit Khandekar. Gradient Descent with Sparsification: An iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, 2009.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping From Saddle Points - Online Stochastic Gradient for Tensor Decomposition. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, pages 797–842, 2015.

Rong Ge, Jason D. Lee, and Tengyu Ma. Matrix Completion has No Spurious Local Minimum. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016.

R. W. Gerchberg and W. Owen Saxton. A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures. *Optik*, 35 (2):237–246, 1972.

Surbhi Goel and Adam Klivans. Learning Depth-Three Neural Networks in Polynomial Time. arXiv:1709.06010v1 [cs.DS], 2017.

Donald Goldfarb and Shiqian Ma. Convergence of Fixed-Point Continuation Algorithms for Matrix Rank Minimization. *Foundations of Computational Mathematics*, 11(2):183–210, 2011.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins Studies in Mathematical Sciences. The John Hopkins University Press, 3rd edition, 1996.

Rémi Gribonval, Rodolphe Jenatton, and Francis Bach. Sparse and Spurious: Dictionary Learning With Noise and Outliers. *IEEE Transaction on Information Theory*, 61(11):6298–6319, 2015.

Moritz Hardt. Understanding Alternating Minimization for Matrix Completion. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2014.

Moritz Hardt and Mary Wootters. Fast Matrix Completion Without the Condition Number. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, 2014.

Moritz Hardt, Raghu Meka, Prasad Raghavendra, and Benjamin Weitz. Computational limits for matrix completion. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, 2014.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations.* Number 143 in Monographs on Statistics and Applied Probability. The CRC Press, 2016.

Ishay Haviv and Oded Regev. The Restricted Isometry Property of Subsampled Fourier Matrices. In Bo'az Klartag and Emanuel Milman, editors, *Geometric Aspects of Functional Analysis*, volume 2169 of *Lecture Notes in Mathematics*, pages 163–179. Springer, Cham, 2017.

Peter J. Huber and Elvezio M. Ronchetti. *Robust Statistics.* Wiley Series in Probability and Statistics. John Wiley & Sons, 2nd edition, 2009.

Kishore Jaganathan, Samet Oymak, and Babak Hassibi. Sparse Phase Retrieval: Convex Algorithms and Limitations. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2013.

Prateek Jain and Praneeth Netrapalli. Fast Exact Matrix Completion with Finite Samples. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, 2015.

Prateek Jain and Ambuj Tewari. Alternating Minimization for Regression Problems with Vector-valued Outputs. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

Prateek Jain, Raghu Meka, and Inderjit Dhillon. Guaranteed Rank Minimization via Singular Value Projections. In *Proceedings of the 24th Annual Conference on Neural Information Processing Systems (NIPS)*, 2010.

Prateek Jain, Ambuj Tewari, and Inderjit S. Dhillon. Orthogonal Matching Pursuit with Replacement. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, 2011.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank Matrix Completion using Alternating Minimization. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, pages 665–674, 2013.

Prateek Jain, Ambuj Tewari, and Purushottam Kar. On Iterative Hard Thresholding Methods for High-dimensional M-Estimation. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS)*, 2014.

Ali Jalali, Christopher C Johnson, and Pradeep D Ravikumar. On Learning Discrete Graphical Models using Greedy Methods. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1935–1943, 2011.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to Escape Saddle Points Efficiently. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1724–1732, 2017.

Boris Sergeevich Kashin. The diameters of octahedra. *Uspekhi Matematicheskikh Nauk*, 30(4(184)):251–252, 1975.

Raghunandan H. Keshavan. *Efficient Algorithms for Collaborative Filtering*. Ph.D. Thesis, Stanford University, 2012.

Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh. Matrix Completion from a Few Entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for. Recommender Systems. *IEEE Computer*, 42(8):30–37, 2009.

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient Descent Only Converges to Minimizers. In *Proceedings of the 29th Conference on Learning Theory (COLT)*, pages 1246–1257, 2016.

Kiryung Lee and Yoram Bresler. ADMiRA: Atomic Decomposition for Minimum Rank Approximation. *IEEE Transactions on Information Theory*, 56 (9):4402–4416, 2010.

Yuanzhi Li and Yang Yuan. Convergence Analysis of Two-layer Neural Networks with ReLU Activation. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Amand A. Lucas. A-DNA and B-DNA: Comparing Their Historical X-ray Fiber Diffraction Images. *Journal of Chemical Education*, 85(5):737–743, 2008.

Zhi-Quan Luo and Paul Tseng. On the Convergence of the Coordinate Descent Method for Convex Differentiable Minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

Zhi-Quan Luo and Paul Tseng. Error bounds and convergence analysis of feasible descent methods: A general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

Ricardo A. Maronna, R. Douglas Martin, and Victor J. Yoha. *Robust Statistics: Theory and Methods*. John Wiley, 2006.

R. Douglas Martin and Judy Zeh. Robust Generalized M-estimates for Autoregressive Parameters: Small-sample Behavior and Applications. Technical Report 214, University of Washington, 1978.

Raghu Meka, Prateek Jain, Constantine Caramanis, and Inderjit Dhillon. Rank Minimization via Online Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Balas Kausik Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24(2):227–234, 1995.

Deanna Needell and Joel A. Tropp. CoSaMP: Iterative Signal Recovery from Incomplete and Inaccurate Samples. *Applied and Computational Harmonic Analysis*, 26:301–321, 2008.

Sahand N. Negahban, Pradeep Ravikumar, Martin J. Wainwright, and Bin Yu. A Unified Framework for High-Dimensional Analysis of M-Estimators with Decomposable Regularizers. *Statistical Science*, 27(4):538–557, 2012.

Jelani Nelson, Eric Price, and Mary Wootters. New constructions of RIP matrices with fast multiplication and fewer rows. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer-Academic, 2003.

Yurii Nesterov. Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems. *SIAM Journal of Optimization*, 22(2):341–362, 2012.

Yurii Nesterov and B.T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

Praneeth Netrapalli, Prateek Jain, and Sujay Sanghavi. Phase Retrieval using Alternating Minimization. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, 2013.

Nam H. Nguyen and Trac D. Tran. Exact recoverability from dense corrupted observations via L1 minimization. *IEEE Transactions on Information Theory*, 59(4):2036–2058, 2013a.

Nam H Nguyen and Trac D Tran. Robust Lasso With Missing and Grossly Corrupted Observations. *IEEE Transaction on Information Theory*, 59(4): 2036–2058, 2013b.

Samet Oymak, Benjamin Recht, and Mahdi Soltanolkotabi. Sharp Time-Data Tradeoffs for Linear Inverse Problems. arXiv:1507.04793 [cs.IT], 2015.

Garvesh Raskutti, Martin J. Wainwright, and Bin Yu. Restricted Eigenvalue Properties for Correlated Gaussian Designs. *Journal of Machine Learning Research*, 11:2241–2259, 2010.

Benjamin Recht. A Simpler Approach to Matrix Completion. *Journal of Machine Learning Research*, 12:3413–3430, 2011.

Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed Minimum Rank Solutions to Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3):471–501, 2010.

Sashank Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alexander J. Smola. Fast Stochastic Methods for Nonsmooth Nonconvex Optimization. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

Peter J. Rousseeuw. Least Median of Squares Regression. *Journal of the American Statistical Association*, 79(388):871–880, 1984.

Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, 1987.

Ankan Saha and Ambuj Tewari. On the Non-asymptotic Convergence of Cyclic Coordinate Descent Methods. *SIAM Journal on Optimization*, 23 (1):576–601, 2013.

Hanie Sedghi and Anima Anandkumar. Training Input-Output Recurrent Neural Networks through Spectral Methods. arXiv:1603.00954 [CS.LG], 2016.

Shai Shalev-Shwartz and Tong Zhang. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. *Journal of Machine Learning Research*, 14:567–599, 2013.

Yiyuan She and Art B. Owen. Outlier Detection Using Nonconvex Penalized Regression. *Journal of the American Statistical Association*, 106(494):626–639, 2011.

Daniel A. Spielman, Huan Wang, and John Wright. Exact Recovery of Sparsely-Used Dictionaries. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, 2012.

Suvrit Sra, Sebastian Nowozin, and Stephen J. Wright, editors. *Optimization for Machine Learning*. The MIT Press, 2011.

Norbert Stockinger and Rudolf Dutter. Robust time series analysis: A survey. *Kybernetika*, 23(7):1–3, 1987.

Ju Sun, Qing Qu, and John Wright. When Are Nonconvex Problems Not Scary? arXiv:1510.06096 [math.OC], 2015.

Ruoyu Sun and Zhi-Quan Lu. Guaranteed Matrix Completion via Non-convex Factorization. In *Proceedings of the 56th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, 2015.

Ambuj Tewari, Pradeep Ravikumar, and Inderjit S. Dhillon. Greedy Algorithms for Structurally Constrained High Dimensional Problems. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, 2011.

Joel A. Tropp. User-Friendly Tail Bounds for Sums of Random Matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

Joel A. Tropp and Anna C. Gilbert. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, Dec. 2007. ISSN 0018-9448.

Meng Wang, Weiyu Xu, and Ao Tang. On the Performance of Sparse Recovery Via ℓ_p-Minimization $(0 \leq p \leq 1)$. *IEEE Transactions on Information Theory*, 57(11):7255–7278, 2011.

Zhaoran Wang, Quanquan Gu, Yang Ning, and Han Liu. High Dimensional EM Algorithm: Statistical Optimization and Asymptotic Normality. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

Karen H.S. Wilson, Sarah E. Eckenrode, Quan-Zhen Li, Qing-Guo Ruan, Ping Yang, Jing-Da Shi, Abdoreza Davoodi-Semiromi, Richard A. McIndoe, Byron P. Croker, and Jin-Xiong She. Microarray Analysis of Gene Expression in the Kidneys of New- and Post-Onset Diabetic NOD Mice. *Diabetes*, 52 (8):2151–2159, 2003.

John Wright, Alan Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma. Robust Face Recognition via Sparse Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

Stephen J Wright and Jorge Nocedal. *Numerical Optimization*, volume 2. Springer New York, 1999.

C.-F. Jeff Wu. On the Convergence Properties of the EM Algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

Allen Y. Yang, Zihan Zhou, Arvind Ganesh Balasubramanian, S Shankar Sastry, and Yi Ma. Fast ℓ_1-Minimization Algorithms for Robust Face Recognition. *IEEE Transactions on Image Processing*, 22(8):3234–3246, 2013.

Fanny Yang, Sivaraman Balakrishnan, and Martin J. Wainwright. Statistical and computational guarantees for the Baum-Welch algorithm. In *Proceedings of the 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2015.

Xinyang Yi and Constantine Caramanis. Regularized EM Algorithms: A Unified Framework and Statistical Guarantees. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

Xinyang Yi, Constantine Caramanis, and Sujay Sanghavi. Alternating Minimization for Mixed Linear Regression. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

Ya-Xiang Yuan. Recent advances in trust region algorithms. *Mathematical Programming*, 151(1):249–281, 2015.

Tong Zhang. Adaptive Forward-Backward Greedy Algorithm for Learning Sparse Representations. *IEEE Transactions on Information Theory*, 57: 4689–4708, 2011.

Yuchen Zhang, Percy Liang, and Moses Charikar. A Hitting Time Analysis of Stochastic Gradient Langevin Dynamics. In *Proceedings of the 30th Conference on Learning Theory*, 2017.

Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery Guarantees for One-hidden-layer Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale Parallel Collaborative Filtering for the Netflix Prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM)*, 2008.