Scalable Convex Optimization Methods for Semidefinite Programming

Thèse N° 9598

Présentée le 27 août 2019

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire de systèmes d'information et d'inférence
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Alp YURTSEVER

Acceptée sur proposition du jury

Prof. O. N. A. Svensson, président du jury
Prof. V. Cevher, directeur de thèse
Prof. J. A. Tropp, rapporteur
Prof. S. Sra, rapporteur
Prof. M. Jaggi, rapporteur

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Devranın değişmez düzeni vardır.
Çözen, çözemeyen yenilir ona.
Onu okumanın bir dili vardır.
Bilimler bilimi denilir ona.

— Hüseyin Yurtsever (2012)
Sayılar Bilimi - Bilinmezlik Deryası

To my family,
dedicated to the memory of my father, Hüseyin Yurtsever.

Acknowledgements

Before we go into the technical content, I would like to add a few words to express my appreciation to the people who supported me during my studies as a PhD candidate.

A very special gratitude goes to my PhD advisor Volkan Cevher. I am grateful to him not only for our research discussions and his technical supervision; but also for being an advisor with genuine concern for my welfare, for sharing the excitement of my accomplishments, and for being a heartfelt friend.

I spent three months at MIT as a visiting student in the fall semester of 2018. I am thankful to Suvrit Sra for hosting me during this visit, and to Volkan for arranging this research visit and making it possible in the first place. I really enjoyed my time at MIT and I am looking forward to joining there as a postdoc. I would also like to thank Joel Tropp for his guidance on our collaborated research for more than 3 years, for all our research discussions, and for inviting and hosting me at Caltech multiple times during this collaboration.

I was fortunate to have worked with so many great collaborators during my PhD. I am deeply grateful to all my collaborators. In particular, this dissertation is based on some joint works with Volkan Cevher, Olivier Fercoq, Ya-Ping Hsieh, Francesco Locatello, Suvrit Sra, Quoc Tran-Dinh, Joel Tropp, and Madeleine Udell.

I was honored to have Volkan Cevher, Martin Jaggi, Suvrit Sra, Ola Svensson, and Joel Tropp as my thesis defense committee members. I am grateful for their time and valuable comments. I would also like to thank my colleagues Ahmet Alacaoglu, Ali Kavis, Fatih Sahin, and Maria Vladarean for reading my thesis and providing feedback, and to Paul Rolland for proofreading my French in the abstract.

I am thankful to our lab members and the alumni for the collaborations, research discussions, and the friendly working environment. I am indebted to our administrative assistant Gosia Baltaian for her help on many occasions and for her friendliness.

I would also like to take this opportunity to thank my friends. My PhD years coincided with some unfortunate incidents in my personal life. Without the love and support of my friends, I would never have recovered. My life is more meaningful with the memorable times that we spent together. Finally, I acknowledge the effort of my teachers over the years, which played a critical role in shaping my academic identity and interests.

Last but certainly not least, I would like to thank my parents Çiğdem and Hüseyin, my sister Bengi, and my brother Emre for their love and constant support. I dedicate this dissertation to the memory of my father. Without his encouragement, I would never have started this journey.

Lausanne, 10 July 2019 A. Y.

Abstract

With the ever-growing data sizes along with the increasing complexity of the modern problem formulations, contemporary applications in science and engineering impose heavy computational and storage burdens on the optimization algorithms. As a result, there is a recent trend where heuristic approaches with unverifiable assumptions are overtaking more rigorous, conventional methods at the expense of robustness and reproducibility.

My recent research results show that this trend can be overturned when we jointly exploit dimensionality reduction and adaptivity in optimization at its core. I contend that even the classical convex optimization did not reach yet its limits of scalability.

Many applications in signal processing and machine learning cast a fitting problem from limited data, introducing spatial priors to be able to solve these otherwise ill-posed problems. Data is small, the solution is compact, but the search space is high in dimensions. These problems clearly suffer from the wasteful use of storage resources by the classical optimization algorithms.

This problem is prevalent. Storage is a critical bottleneck that prevents us from solving many important large-scale optimization problems. For example, semidefinite programs (SDP) often have low-rank solutions. However, SDP algorithms require us to store a matrix decision variable in the ambient dimensions.

This dissertation presents a convex optimization paradigm which makes it possible to solve trillion dimensional SDP relaxations to combinatorial decision problems on a regular personal computer. The key idea is to use an optimization procedure that performs compact updates, and to maintain only a small sketch of the decision variable. Once the iterations terminate, we can recover an approximate solution from the information stored in this sketch.

We start by formalizing this idea for a model problem with low-rank solutions. Based on the classical conditional gradient method, we propose the first convex optimization algorithm with optimal storage guarantees for this model problem. Then, we develop and describe the key ingredients to extend our results for a broader set of problems, SDP in particular.

SDP formulations are key in signal processing, machine learning, and other engineering applications. By greatly enhancing the scalability of optimization algorithms for solving SDP formulations, we can potentially unlock new results in important scientific applications.

Key words: Convex optimization, semidefinite programming, low-rank matrix optimization, primal-dual methods, conditional gradient methods, low-rank matrix sketching

Résumé

Avec la taille toujours croissante des données massives et la complexité des formulations de problèmes modernes, les applications actuelles en sciences et en ingénierie imposent de lourdes charges de calcul et de stockage. En conséquence, il existe une tendance récente où les approches heuristiques avec des hypothèses invérifiables dépassent les méthodes plus rigoureuses et conventionnelles au détriment de la robutesse et de la reproductibilité.

Mes résultats récents montrent que cette tendance peut être renversée si nous exploitons conjointement la réduction de la dimensionnalité et l'adaptivité dans l'optimisation. Je soutiens que même l'optimisation convexe classique n'a pas encore atteint ses limites.

De nombreuses applications dans le traitement du signal et l'apprentissage automatique posent un problème d'ajustage à partir de données limitées, en introduisant des informations préalables spatiales afin de pouvoir résoudre ces problèmes. Les données et la solution sont compactes, mais l'espace de recherche est de grande dimension. Clairement, ces problèmes souffrent de l'utilisation inutile des ressources de stockage par les algorithmes d'optimisation.

Ce problème est très répandue. Le stockage est un goulot d'étranglement critique qui nous empêche de résoudre de nombreux problèmes d'optimisation à grande échelle. Par exemple, les programmes semi-définies (SDP) ont souvent des solutions de rang faible. Cependant, les algorithmes SDP nous obligent à stocker une variable de décision matricielle dans les dimensions ambiantes.

Cette thèse présente un paradigme d'optimisation convexe qui permet de résoudre des instances de programmation semi-définie à grande échelle de trillions de dimensions sur un ordinateur personnel ordinaire. L'idée clé est d'utiliser un algorithme d'optimisation qui effectue des mises à jour compactes, et de ne conserver qu'un petit *sketch* de la variable de décision. Une fois la procédure terminée, nous pouvons récupérer une solution approximative à partir des informations stockées dans le sketch.

Nous commençons par la formalisation de cette idée pour un problème de modèle avec des solutions à faible rang. Sur la base de la méthode classique du gradient conditionnel, nous proposons le premier algorithme d'optimisation convexe avec des garanties de stockage optimales pour ce problème de modèle. Ensuite, nous développons et décrivons les ingrédients clés pour étendre nos résultats à d'autres modèles de problèmes, SDP en particulier.

Les formulations SDP jouent un rôle clé dans le traitement du signal, l'apprentissage automatique, et d'autres applications d'ingénierie. En améliorant l'évolutivité des méthodes d'optimisation pour la résolution des formulations SDP, nous pouvons potentiellement obtenir de nouveaux résultats dans des applications scientifiques importantes.

Résumé

Mots clés : Optimisation convexe, programmation semi-définie, optimisation matricielle de rang faible, méthodes primales-duales, méthodes de gradient conditionnel, sketch pour des matrices de rang faibles

Bibliographic Note

This dissertation is based on the following publications:

- [YTDC15a] Alp Yurtsever, Quoc Tran-Dinh, Volkan Cevher. "A Universal Primal-Dual Convex Optimization Framework." Conference on Neural Information Processing Systems (NeurIPS), 2015.
- [YHC15] Alp Yurtsever, Ya-Ping Hsieh, Volkan Cevher. "Scalable Convex Methods for Phase Retrieval." IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2015.
- [YUTC17] Alp Yurtsever, Madeleine Udell, Joel Aaron Tropp, Volkan Cevher "Sketchy Decisions: Convex Low-rank Matrix Optimization with Optimal Storage." International Conference on Artificial Intelligence and Statistics (AISTATS), 2017.
- [TYUC17b] Joel Aaron Tropp, Alp Yurtsever, Madeleine Udell, Volkan Cevher. "Practical Sketching Algorithms for Low-rank Matrix Approximation." SIAM Journal on Matrix Analysis and Applications (SIMAX), 2017.
- [TYUC17a] Joel Aaron Tropp, Alp Yurtsever, Madeleine Udell, Volkan Cevher. "Fixed-rank Approximation of a Positive-Semidefinite Matrix from Streaming Data." Conference on Neural Information Processing Systems (NeurIPS), 2017.
- [YFLC18] Alp Yurtsever, Olivier Fercoq, Francesco Locatello, Volkan Cevher. "A Conditional Gradient Framework for Composite Convex Minimization with Applications to Semidefinite Programming." International Conference on Machine Learning (ICML), 2018.
- [YFC19] Alp Yurtsever, Olivier Fercoq, Volkan Cevher. "A Conditional Gradient-Based Augmented Lagrangian Framework." International Conference on Machine Learning (ICML), 2019.
- [YSC19] Alp Yurtsever, Suvrit Sra, Volkan Cevher. "Conditional Gradient Methods via Stochastic Path-Integrated Differential Estimator." International Conference on Machine Learning (ICML), 2019.
- [TYUC19] Joel Aaron Tropp, Alp Yurtsever, Madeleine Udell, Volkan Cevher. "Streaming Low-Rank Matrix Approximation with an Application to Scientific Simulation." Accepted to SIAM Journal on Scientific Computing (SISC), 2019.

Bibliographic Note

My other publications relevant to this research, but not covered in this dissertation, are:

- [YVC16] Alp Yurtsever, Bang Cong Vu, Volkan Cevher. "Stochastic Three-Composite Convex Minimization." Conference on Neural Information Processing Systems (NeurIPS), 2016.
- [OLY+16] Gergely Odor, Yen-Huan Li, Alp Yurtsever, Ya-Ping Hsieh, Quoc Tran-Dinh, Marwa El Halabi, Volkan Cevher. "Frank-Wolfe works for non-Lipschitz Continuous Gradient Objectives: Scalable Poisson Phase Retrieval." IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2016.
- [CVY18] Volkan Cevher, Bang Cong Vu, Alp Yurtsever. "Stochastic Forward Douglas-Rachford Splitting Method for Monotone Inclusions." in Large-Scale and Distributed Optimization, Chapter 7. Springer, Cham, (2018)
- [LYC18] Kfir Levy, Alp Yurtsever, Volkan Cevher. "Online Adaptive Methods, Universality and Acceleration." Conference on Neural Information Processing Systems (NeurIPS), 2018.
- [DYC+19] Lijun Ding, Alp Yurtsever, Volkan Cevher, Joel Aaron Tropp, Madeleine Udell. "An Optimal-Storage Approach to Semidefinite Programming Using Approximate Complementarity." Manuscript (arXiv:1902.03373), 2019.
- [CVY19] Volkan Cevher, Bang Cong Vu, Alp Yurtsever. "Inertial Three-Operator Splitting Method and Applications." Manuscript (arXiv:1904.12980), 2019.
- [LYFC19] Francesco Locatello, Alp Yurtsever, Olivier Fercoq, Volkan Cevher. "Stochastic Conditional Gradient Method for Composite Convex Minimization." Manuscript (arXiv: 1901.10348), 2019.

Contents

List of Figures

List of Tables

1 Introduction

We can recognize almost every problem in science and engineering as an optimization problem. Whether it is a finance problem where we want to maximize the profit based on a financial model, a problem in physics where the aim is to identify a minimum energy state under the constraints imposed by the laws of physics, or a machine learning problem for designing an autonomous self-driving vehicle, we can model it as a mathematical optimization problem.

Convex optimization is an important subclass of mathematical optimization. One major benefit of formulating an application as a convex optimization problem is its robustness. Convex optimization provides a unified modeling framework for solving problems across diverse disciplines. To find an approximate solution, convex optimization algorithms rely on favorable geometric structures and functional properties present in the convex optimization templates, avoiding further assumptions on the problem data as much as possible.

Unfortunately, with the ever-growing data sizes along with the increasing complexity of the modern applications, contemporary problems in science and engineering impose heavy computational and storage burdens on the optimization algorithms. As a result, there is a recent trend where heuristic approaches with unverifiable assumptions on the problem data are overtaking more rigorous, conventional methods at the expense of robustness and reproducibility.

My recent research results show that this trend can be overturned, when we jointly exploit dimensionality reduction and adaptivity in optimization at its core. I contend that even the classical convex optimization did not reach yet its limits of scalability. In particular, this dissertation presents a convex optimization paradigm which makes it possible to solve some trillion dimensional semidefinite programming problems on a regular laptop.

The design of an end-product algorithm with a more detailed cost analysis, along with a well-tuned, stable and robust implementation and numerical demonstrations in cutting-edge applications are deferred to a follow-up paper.

1.1 Problem Formulation

The importance of convex optimization in machine learning has increased dramatically in the last decade. Indeed, a large class of learning formulations can be addressed by the following constrained convex minimization template:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \mathcal{A}\boldsymbol{x} \in \mathcal{K}, \quad \boldsymbol{x} \in \mathcal{X}. \tag{1.1}$$

where $\mathcal{X} \subset \mathbb{R}^p$ is a convex and compact (nonempty, bounded, closed) set, and its 0-dimensional faces (*i.e.*, its vertices) are called *atoms*. $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is a proper closed convex function, $\mathcal{A} : \mathbb{R}^p \to \mathbb{R}^d$ is a linear map, and $\mathcal{K} \subseteq \mathbb{R}^d$ is a convex set.

This template covers in particular semidefinite programming (SDP) formulations. Let us focus on the following SDP template:

$$\begin{aligned} \underset{\boldsymbol{X}}{\text{minimize}} \quad & \langle \boldsymbol{C}, \boldsymbol{X} \rangle \\ \text{subject to} \quad & \langle \boldsymbol{A}_i, \boldsymbol{X} \rangle = b_i \quad \text{for } i = 1, \dots, d \\ & \boldsymbol{X} \in \mathbb{S}^n_+ \quad \text{and} \quad \operatorname{Tr} \boldsymbol{X} = \alpha \end{aligned} \tag{1.2}$$

Here, $\mathbb{S}^n_+$ denotes the cone of $n \times n$ positive semidefinite matrices. $\boldsymbol{C}, \boldsymbol{A}_1, \boldsymbol{A}_2, \dots, \boldsymbol{A}_d \in \mathbb{S}^n$ (i.e., $n \times n$ dimensional and symmetric), and $b_1, b_2, \dots, b_d \in \mathbb{R}$. This is the standard SDP template except the $\operatorname{Tr} \boldsymbol{X} = \alpha$ constraint that we add to ensure the boundedness of the problem domain. This additional constraint is not restrictive, since we can compute (or approximate) the trace of a solution a priori, either from the problem formulation or from the data in many applications.

We will use the following standard compact notation for the affine constraints

$$\begin{aligned} \mathcal{A} &: \mathbb{S}^n_+ \to \mathbb{R}^d \quad \text{where} \quad \mathcal{A}\boldsymbol{X} = \begin{bmatrix} \langle \boldsymbol{A}_1, \boldsymbol{X} \rangle & \dots & \langle \boldsymbol{A}_d, \boldsymbol{X} \rangle \end{bmatrix}; \\ \mathcal{A}^* &: \mathbb{R}^d \to \mathbb{S}^n_+ \quad \text{where} \quad \mathcal{A}^* \boldsymbol{z} = \textstyle\sum_{i=1}^d z_i \boldsymbol{A}_i. \end{aligned} \tag{1.3}$$

Then, we can reformulate the model problem (1.2) as

$$\underset{\boldsymbol{X}}{\text{minimize}} \quad \langle \boldsymbol{C}, \boldsymbol{X} \rangle \quad \text{subject to} \quad \mathcal{A}\boldsymbol{X} = \boldsymbol{b}, \quad \underbrace{\boldsymbol{X} \in \mathbb{S}^n_+, \quad \operatorname{Tr} \boldsymbol{X} = \alpha}_{\boldsymbol{X} \in \mathcal{X}}, \tag{1.4}$$

where the vector $\boldsymbol{b} := (b_1, \dots, b_d) \in \mathbb{R}^d$ is the constraint (measurement) vector. Clearly (1.4) is an instance of (1.1). We restrict ourselves to equality constraint $\mathcal{A}\boldsymbol{X} = \boldsymbol{b}$ for notational simplicity. All results that we present in this dissertation can be extended for the affine inclusion constraint $\mathcal{A}\boldsymbol{X} \in \mathcal{K}$. Similarly, extension for $\operatorname{Tr} \boldsymbol{X} \leq \alpha$ constraint is straightforward.

In most applications, the cost matrix $\boldsymbol{C}$ and the constraint matrices $\boldsymbol{A}_i$ have low intrinsic dimensionality, *e.g.,* they are low rank, or sparse. Hence, storing and applying these matrices do not put a substantial cost on our computational budget.

1.2 Semidefinite Programming

There is a common misconception between convexity and tractability of an optimization problem. In full generality, convex optimization problems are not solvable in polynomial time, unless P = NP. There are examples of NP-hard combinatorial decision problems that can be formulated as a convex optimization problem over the copositive cone[1]. As a consequence, it is not possible to design a generic solver for the complete class of convex optimization problems, and we need to define (or identify) a problem subclass before designing (resp. applying) a convex optimization algorithm.

This identification process of an appropriate algorithmic template requires field expertise. In order to circumvent this need, disciplined convex programming introduces a set of conventions to follow when formulating optimization problems. This yields a natural hierarchy of programming templates in convex optimization. Semidefinite programming (SDP) sits somewhere on top of this hierarchy. It covers most of the other well-known templates, from linear programming to second-order cone programming, as a special case.

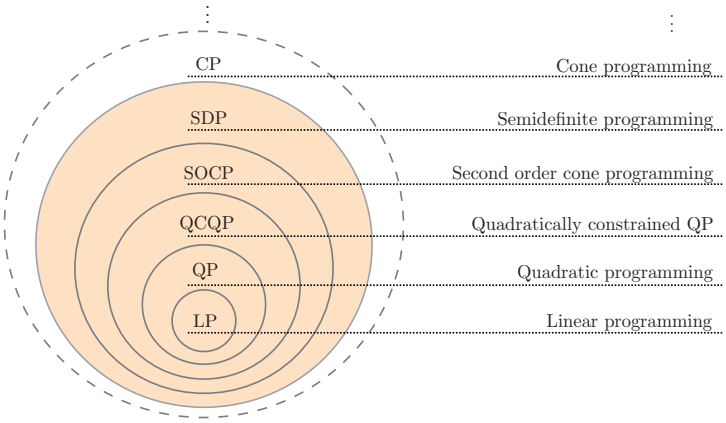


Figure 1.1 – Disciplined convex programming yield a natural hierarchy in convex optimization. SDP sits somewhere on top of this hierarchy.

Semidefinite programming relaxations are powerful for obtaining approximate solutions to countless problems in computer science [KN12, Lov03], machine learning [MNS15, SSGB07], and signal processing [Sin11, SS11, CKS15]. Many recent studies have revealed the deep and surprising phenomenon that, for many of these problems, one cannot obtain better approximations beyond the semidefinite relaxation. A famous example (see [Rag08]) in computer science is that, assuming a strengthened P≠NP conjecture, *called the Unique Games Conjecture*, it is NP-hard to beat semidefinite relaxation for a large class of combinatorial optimization problems.

[1] The definition of copositivity can be traced back to [Mot52]. For a recent survey, see [HUS10].

1.2.1 Relax and Round

Let us derive a basic SDP relaxation for a combinatorial decision problem, maximum cut problem (max-cut), in order to present some common structures in these formulations. Similar ideas arise in many other SDP applications.

Given an undirected graph $G = (V, E)$ with vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ and a set of weights $c : E \to \mathbb{R}_+$ associated to each edge on the graph, the aim is to define cut that partitions the vertices into two distinct sets, such that the sum of the weights that correspond to the edges that we cross is maximized.

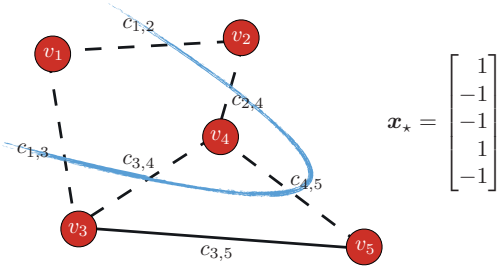


Figure 1.2 – Graph cut on a toy graph.

We can mathematically formulate this problem as a combinatorial optimization problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{maximize}} \quad \frac{1}{2}\sum_{\{i,j\}\in E} c_{i,j}(1 - x_i x_j) \quad \text{subject to} \quad x_i \in \{-1, +1\}, \quad i = 1, 2, \ldots, n. \tag{1.5}$$

This problem is NP-hard, and the difficulty arises from the quadratic terms in the objective value $x_i x_j$, and the integer valued constraint $x_i \in \{-1, +1\}$.

To deal with this difficulty, we can *lift* the problem into the matrix space: Define the matrix variable $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^*$. Then, we can equivalently formulate (1.5) as a rank constrained matrix optimization problem:

$$\underset{\boldsymbol{X}}{\text{maximize}} \quad \frac{1}{2}\sum_{\{i,j\}\in E} c_{i,j}(1 - X_{ij}) \quad \text{subject to} \quad \operatorname{diag}\boldsymbol{X} = \mathbf{1} \quad \boldsymbol{X} \in \mathbb{S}^n_+, \quad \operatorname{rank}\boldsymbol{X} = 1. \tag{1.6}$$

where the linear map $\operatorname{diag} : \mathbb{S}^n_+ \to \mathbb{R}^n$ extracts the diagonal of a matrix. This problem is still NP-hard, but we translated the difficulty into the non-convex rank constraint.

Nest step is to *relax* by dropping the rank constraint:

$$\text{maximize} \quad \frac{1}{2}\sum_{\{i,j\}\in E} c_{i,j}(1 - X_{ij}) \quad \text{subject to} \quad \operatorname{diag}\boldsymbol{X} = \mathbf{1} \quad \boldsymbol{X} \in \mathbb{S}^n_+. \tag{1.7}$$

This is the SDP relaxation of the combinatorial max-cut problem [GW95]. We can add $\operatorname{Tr}\boldsymbol{X} = n$ constraint to (1.7), which is already implied by $\operatorname{diag}\boldsymbol{X} = \mathbf{1}$.

We can solve (1.7), in polynomial time, and get an approximate solution $\boldsymbol{X}_\star$ to (1.7). However, $\boldsymbol{X}_\star$ is not rank-one in general. Hence, in order to provide an approximate solution to the original problem (1.5), we need a *rounding* step. Goemans and Williamson [GW95] describe a randomized rounding procedure which produces approximately optimal cuts with theoretical guarantees.

1.3 Arithmetics: Crisis and Opportunity

Within the last two decades, developments in machine learning applications have led to revolutionary changes in our expectations from optimization in many ways. The stupendous amount of data involved in the problem descriptions forced us to consider improving scalability of the optimization algorithms as the primary goal.

Given a fixed amount of computational resources, and a fixed target approximation accuracy, the scalability (to problem size) of an optimization procedure for a specific task can be theoretically bounded. In other words, in order to scale our algorithms to very large problems beyond some theoretical bound, we have to sacrifice from the approximation accuracy. It is for this reason that we typically consider the problem dimensions and the accuracy as variables when we derive iteration complexity of an optimization algorithm, and suppress all other data-dependent constants.

Fortunately, this compromise is theoretically well-justified in machine learning. In many machine learning applications, one minimizes a loss function that corresponds to a statistical estimator of the true model (to which we do not have access). This incurs modeling error to the overall system. As a consequence, solving the optimization problem to very high accuracy, orders beyond the estimation error, can only provide an insignificant amount of extra information about the overall system. Indeed, high accuracy solutions to these problems are not even desirable because it can cause overfitting. Accordingly, at first, first-order methods, and more recently their stochastic extensions became increasingly popular despite their slow sublinear convergence rates.

We can argue that this phenomenon is not specific to machine learning applications. Consider an SDP relaxation to a combinatorial decision problem, such as the max-cut example. We relax the original NP-hard problem as an SDP formulation by dropping the non-convex rank constraint. We approximately solve the SDP problem using an iterative optimization procedure. Then, we execute a rounding step to produce an approximate solution to the original problem. We can view this whole procedure, which consists of *relaxation*, *optimization*, and *rounding*, as a black-box model for solving the original combinatorial optimization problem, see Figure 1.3.

The goals of optimization and modeling are tightly connected. Relaxation imposes a modeling error to the overall system. Hence, solving the optimization step to very high accuracy, beyond the relaxation accuracy, do not have a significant effect on the final estimation quality. We demonstrate this phenomenon numerically for the SDP formulation of a clustering problem in Section 4.4.1. These arguments exclude a small class of examples for which the empirical study suggests that the relaxation is exact.

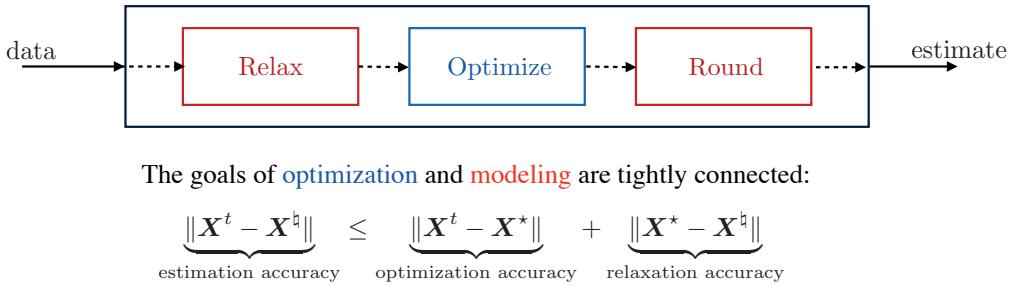


Figure 1.3 – Black-box model for solving an SDP relaxation: Relax, Optimize and Round.

1.4 Storage: Crisis and Opportunity

The majority of the semidefinite relaxation formulations relies on *lifting* the decision variable to a higher dimensional search-space. Recall the max-cut example, the original formulation (1.5) corresponds to an n-dimensional combinatorial problem, whereas the SDP relaxation (1.7) lies in an $(n \times n)$-dimensional space.

Storage cost is the critical bottleneck that prevents us from solving many important problems in large-scale. This becomes evident even in a simple max-cut example, considering the size of modern social or transportation network graphs, which can easily reach to millions of nodes. Even just the decision variable requires storing a huge, millions by millions matrix!

Decision variable is a core internal variable for optimization. Any reasonable optimization algorithm must output an approximate solution to the problem. Consequently, we cannot completely discard the decision variable from our optimization procedure, and the only hope that remains is to exploit the structures of a solution for storing it in intrinsic dimensions.

Solutions to SDP problems are often low-rank, or close to low-rank. Consider an SDP relaxation to a combinatorial optimization problem, such as the max-cut example. Even though we drop the rank constraint, roughly speaking, a solution to the relaxation is informative only if it is close to the solution set of the original problem. Accordingly, the solution (to SDP) typically exhibits a fast decay in singular value spectrum, even if it is not exactly low-rank.

Moreover, many SDP problems have low-rank solutions due to the widely known Pataki-Barvinok bound, see [Bar95] and [Pat98]. Under some mild technical assumptions, SDP problem (1.4) has a solution with rank $r \leq r^{PB}$ for some $r^{PB} = \Theta(\sqrt{d})$. In other words, SDP problems provably have low-rank solutions when $d \ll n^2$. Recall that $d = n$ in max-cut.

Unfortunately the storage issue remains, even if the solution has low intrinsic dimensionality. Virtually all convex optimization methods require storing the $n \times n$ dimensional matrix decision variable in the ambient dimensions, simply because the intermediate estimates might be incompressible. In fact, this fundamental burden has been seen as a natural limit for scalability, and the researchers switched their focus to non-convex frameworks to attain scalable solutions at the expense of robustness. In 2015, while proposing the non-convex Wirtinger-Flow method for solving phase retrieval problems, Candès et al. [CLS15b] explain the reason of using a non-convex template as follows:

> *"For certain random models, some recent SDP relaxations such as PhaseLift [CESV13] are known to provide exact solutions (up to global phase) to the generalized phase retrieval problem using a near minimal number of sampling vectors [CLS15a, CSV12]. While in principle SDP based relaxations offer tractable solutions, they become computationally prohibitive as the dimension of the signal increases. Indeed, for a large number of unknowns in the tens of thousands, say, the memory requirements are far out of reach of desktop computers so that these SDP relaxations are de facto impractical."*

Our recent research results show that this trend can be overturned. In Chapter 2, we describe a storage-optimal convex optimization paradigm for solving large-scale matrix optimization problems with low-rank solutions. The key idea is to maintain only a sketch of the matrix decision variable, and to use an algorithm that is compatible with these sketches. To demonstrate the scalability of our framework, we exhibit numerical solutions to a phase-retrival imaging problem with 25′600 pixel images containing human blood cells from a working Fourier ptychograpy system [HCO$^+$15]. Our approach can scale to problems of millions of dimensions on a regular personal computer.

1.5 State of the Research in the Field

The first reliable algorithms for solving semidefinite programs were based on interior-point methods, introduced independently by Nesterov & Nemirovski [NN89, NN94] and Alizadeh [Ali91, Ali93]. Unfortunately, interior point methods are not effective for solving large-scale problems. This is simply because they typically rely on a second order optimization algorithm as an oracle to query at each iteration. In other words, each iteration of an interior point method itself is a difficult optimization problem that does not scale well.

The success in solving SDP problems led researchers to investigate new applications. In return, invention of new applications put the existing optimization frameworks under more and more pressure to accommodate larger-scale problems. As a result of the increased popularity of SDP formulations in machine learning and signal processing applications, researchers started to search for more scalable procedures for solving SDP problems. Accordingly, first-order methods are proposed as an alternative to interior point methods for large-scale applications. The majority of these algorithms are direct adaptations of some classical methods in the

literature for an SDP template, such as the proximal gradient approach and its accelerated variants [Roc70, BT09] or the alternating direction method of multipliers [BPC+11].

A substantial challenge for the first-order methods is the computational cost of the projection oracle. Recall that projection onto positive semidefinite cone might require a full eigendecomposition, which imposes $\mathcal{O}(n^3)$ arithmetic cost per-iteration. Coupled with the slow, sublinear convergence rate of first-order methods, this creates an undesirable computational burden. The conditional gradient method (CGM, also known as the Frank-Wolfe algorithm) is extremely powerful in this setup, since it avoids projection steps by leveraging the so-called linear minimization oracle that can be solved efficiently using iterative linear algebra routines such as the power method or Lanczos algorithm [FW56, Haz08, Jag13].

Hazan [Haz08] proposed using CGM for solving smooth problems over positive semidefinite cone. However, CGM is not as flexible as the proximal first-order methods, and no extant practical variant of CGM exists in the literature for solving the SDP template (1.4). To this end, [Haz08, GH11, GH16] suggest solving a sequence of feasibility problems with the least squares loss and applying binary search to the objective value. Renegar [Ren14] develops a subgradient method for linear and semidefinite programming with similar per-iteration cost as CGM. None of these methods, however, addresses the storage bottleneck.

Despite the advancements in reducing the arithmetic costs, our progress in reducing the storage footprint is minimal in the convex realm. Almost all convex optimization methods require storing the $n \times n$ dimensional matrix decision variable, which eliminates them in large-scale.

This promoted a new trend of non-convex algorithms that apply to non-convex reformulations of the SDP template. The majority of these heuristics are based on the factorization idea of Burer & Monteiro [BM03], where the matrix variable is factorized as $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{U}^*$, such that $\boldsymbol{U} \in \mathbb{R}^{n \times r}$ for some splitting rank $r \leq n$:

$$\text{minimize} \quad \langle \boldsymbol{C}, \boldsymbol{U}\boldsymbol{U}^* \rangle \quad \text{subject to} \quad \mathcal{A}(\boldsymbol{U}\boldsymbol{U}^*) = \boldsymbol{b}. \tag{1.8}$$

One can apply a wide range of optimization methods to find a stationary point of this problem with respect to the factor $\boldsymbol{U}$. See [BM03, JNS13, Bou15, BKS16, CLS15b] for some examples. The main advantage of the non-convex heuristics is the reduced size of the decision variable. Some methods also benefit from locally linear convergence rates. On the other hand, these algorithms are provable only under stringent, unverifiable and (sometimes) unrealistic statistical assumptions on the problem data.

Very recently, we have developed a new storage-optimal algorithm for solving SDP in standard form, based on the approximate complementarity principle [DYC+19]. The main idea is to solve the low-dimensional dual SDP approximately, and then to recover the primal SDP by solving a compressed SDP problem to the eigenspace with the small eigenvalues of the dual slack matrix. We do not cover this work in this dissertation.

1.6 Organization and Contributions

This dissertation presents a novel convex optimization paradigm to design scalable algorithms for solving the general SDP template (1.4). In particular, the contributions of each chapter are as follows:

- In Chapter 2, we propose a novel paradigm for designing optimization algorithms for large-scale convex problems with structured solutions. It specifically focuses on a fundamental class of convex matrix optimization problems with low-rank solutions. The two key ideas are (i) to maintain only a sketch of the decision variable and (ii) to use algorithmic templates that are compatible with these sketches. It develops and analyzes a new sketch-based algorithm for a model problem from this class. This algorithm, SketchyCGM, modifies a standard convex optimization scheme, the conditional gradient method. In contrast to nonconvex heuristics, the guarantees for SketchyCGM do not rely on statistical models for the problem data. Numerical evidence establishes the benefits of SketchyCGM.

In order to adapt SketchyCGM for solving the SDP template (1.4), we need to design compatible algorithms for (1.4). To this end, Chapters 2 and 3 develop new optimization methods for (1.1), with similar characteristics as the conditional gradient method.

- In Chapter 3, we describe a new primal-dual optimization method for the prototypical constrained convex optimization template (1.1). In contrast to existing primal-dual algorithms, our framework avoids the proximity operator of the objective function altogether. We instead leverage computationally cheaper, Fenchel-type operators, which are the main workhorses of the generalized conditional gradient type methods. Unlike standard CGM, our algorithm can also handle the affine constraints $\mathcal{A}\boldsymbol{x} \in \mathcal{K}$. Our algorithms are *universal* in the sense that they can automatically adapt to the unknown Hölder smoothness orders within the template.

- In Chapter 4, we introduce a generalized conditional gradient method, HomotopyCGM, for the constrained convex optimization template (1.1). Our approach combines smoothing (quadratic penalty) and homotopy techniques under the CGM framework, and provably achieves $\mathcal{O}(1/\sqrt{t})$ convergence rate, where t denotes the iteration counter. We demonstrate that the same rate holds if the linear subproblems are solved approximately with additive or multiplicative error.

- In Chapter 5, We extend our HomotopyCGM algorithm from quadratic penalty to an augmented Lagrangian framework. The resulting algorithm, CGAL, retains the strong theoretical guarantees of HomotopyCGM while exhibiting significantly superior empirical performance. Numerical evidence demonstrates the benefits of CGAL in various SDP applications.

SketchyCGM adopts a sketching model as a natural mechanism for reducing the dimensionality of the decision variable. Consequently, designing practical sketching algorithms for low-rank matrix approximation is a fundamental aspect of our research.

- In Chapter 6, we develop a suite of new algorithms for fixed-rank approximation from a sketch. These methods can preserve structural properties of the input matrix, such as positive-semidefiniteness. Each method is accompanied by an informative error bound. The algorithms are simple, accurate, numerically stable, and provably correct. We implement our algorithms as a MATLAB Toolbox open for public access. Computer experiments show that the proposed methods dominate alternative techniques across a wide range of examples.

Many SDP algorithms require the problem data to be readily available. However, most problems in engineering and machine learning contains some degree of randomness. Therefore, an interesting future research direction is the stochastic extension of our findings.

- In Chapter 7, we present our preliminary study for the stochastic extension, where we aim to identify and present the stochastic CGM variants with the best theoretical guarantees. We study expectation minimization and finite-sum settings, both with and without the convexity assumption. We propose a class of novel variance-reduced stochastic CGM algorithms, and improve the best known rates in the literature for various problem settings.
- Finally in Chapter 8, we summarize the main contributions of this dissertation and describe some future research directions.

1.7 Notation and Terminology

For notational simplicity, we work on the $\mathbb{R}^p/\mathbb{R}^{n\times n}$ and $\mathbb{R}^d$ spaces with the Euclidean norms. We write $\|\cdot\|$ for the Euclidean norm, $\|\cdot\|_{\mathrm{F}}$ for the Frobenius norm, and $\|\cdot\|_{S_1}$ for the Schatten 1-norm (aka the *trace norm* or the *nuclear norm*). Depending on context, $\langle\cdot,\cdot\rangle$ refers to the Euclidean or Frobenius inner product. The symbol * denotes the conjugate transpose of a vector or matrix, as well as the adjoint of a linear map. The dagger † refers to the pseudoinverse. The symbol $[\boldsymbol{X}]_r$ stands for a best rank-r Frobenius-norm approximation of the matrix $\boldsymbol{X}$. The symbol $\succcurlyeq$ denotes the semidefinite order. The function $\mathrm{dist}(\boldsymbol{z};\mathcal{K})$ returns the minimum Euclidean distance from $\boldsymbol{z}$ to a set $\mathcal{K}$, i.e., $\mathrm{dist}(\boldsymbol{z};\mathcal{K}) := \min_{\boldsymbol{u}\in\mathcal{K}} \|\boldsymbol{z}-\boldsymbol{u}\|$. Similarly, the function $\mathrm{dist}(\boldsymbol{X};\mathcal{X})$ returns the minimum Frobenius-norm distance from $\boldsymbol{X}$ to a set $\mathcal{X}$. For a convex function f, we use ∇f both for its subgradient and gradient. We denote the diameter of $\mathcal{X}$ by $D_{\mathcal{X}} = \max_{\boldsymbol{x}_1,\boldsymbol{x}_2\in\mathcal{X}} \|\boldsymbol{x}_1-\boldsymbol{x}_2\|$. We use the computer science interpretation of the order notation $\mathcal{O},\tilde{\mathcal{O}},\Omega,\Theta$.

Next, we recall some basic notions from convex analysis that will be used especially in Chapters 3 and 4 to design new optimization methods for solving (1.1).

Subdifferential. The subdifferential of a function f at a point $\boldsymbol{x}\in\mathbb{R}^p$ is defined as

$$\partial f(\boldsymbol{x}) = \left\{ \boldsymbol{u}\in\mathbb{R}^p :\ f(\boldsymbol{v}) - f(\boldsymbol{x}) \geq \langle \boldsymbol{v}-\boldsymbol{x}, \boldsymbol{u}\rangle,\ \forall\, \boldsymbol{v}\in\mathbb{R}^p \right\}. \tag{1.9}$$

Elements of the subdifferential set are called subgradients, and we denote an arbitrary subgradient of f at $\boldsymbol{x}$ by $\nabla f(\boldsymbol{x})$. If $\partial f(\boldsymbol{x})$ is a singleton, then f is a differentiable function, and we call $\nabla f(\boldsymbol{x})$ as the gradient of f at $\boldsymbol{x}$.

Lipschitz continuity. We say that a function $g : \mathbb{R}^d \to \mathbb{R}$ is L-Lipschitz continuous if it satisfies

$$|g(\boldsymbol{x}_1) - g(\boldsymbol{x}_2)| \leq L\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d.$$

Smoothness. A differentiable function $f : \mathcal{X} \to \mathbb{R}$ is L_f-smooth if its gradient ∇f is L_f-Lipschitz continuous:

$$\|\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2)\| \leq L_f\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}.$$

Proximal operator. The proximal operator of a function $g : \mathbb{R}^d \to \mathbb{R}$ is defined as follows

$$\text{prox}_g(\boldsymbol{z}) = \arg\min_{\boldsymbol{y} \in \mathbb{R}^d} \left\{ g(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{z}\|^2 \right\}. \tag{1.10}$$

Roughly speaking, the proximal operator is tractable when the computation of (1.10) is cheap. If g is the indicator function of a nonempty, closed convex subset $\mathcal{K}$, its proximity operator is the projection operator on $\mathcal{K}$.

Linear minimization oracle. As the problem dimensions become increasingly larger, the proximal tractability assumption can be restrictive. This fact increased the popularity of the conditional gradient methods (CGM, *aka* Frank-Wolfe algorithms), which instead leverage the following linear minimization oracle:

$$\text{lmo}_{\mathcal{X}}(\boldsymbol{x}) = \arg\min_{\boldsymbol{u} \in \mathcal{X}} \langle \boldsymbol{u}, \boldsymbol{x} \rangle. \tag{1.11}$$

Lagrange saddle point. Introduce a slack variable $\boldsymbol{v} \in \mathcal{K}$, and define the Lagrangian of (a reformulation of) the problem (1.1) by

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) := f(\boldsymbol{x}) + \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle. \tag{1.12}$$

For the SDP template (1.4), we define the Lagrangian as $\mathcal{L}(\boldsymbol{X}, \boldsymbol{y}) = f(\boldsymbol{X}) + \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{X} - \boldsymbol{b} \rangle$.

Then, we can formulate the primal and dual problems as follows:

$$\underbrace{\max_{\boldsymbol{y} \in \mathbb{R}^d} \min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \boldsymbol{v} \in \mathcal{K}}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}_{dual} \leq \underbrace{\min_{\substack{\boldsymbol{x} \in \mathcal{X} \\ \boldsymbol{v} \in \mathcal{K}}} \max_{\boldsymbol{y} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y})}_{primal}. \tag{1.13}$$

We assume that the strong duality holds, *i.e.*, the relation above holds with equality.

Slater's condition is a sufficient condition for strong duality. By Slater's condition, we mean

$$\operatorname{relint}(\mathcal{X} \times \mathcal{K}) \cap \{(\boldsymbol{x}, \boldsymbol{v}) \in \mathbb{R}^p \times \mathbb{R}^d : \mathcal{A}\boldsymbol{x} = \boldsymbol{v}\} \neq \emptyset, \tag{1.14}$$

where relint stands for the relative interior.

Solution set. We denote an exact solution of (1.1) by $\boldsymbol{x}_\star$ (*resp.*, solution of (1.4) by $\boldsymbol{X}_\star$), and the set of all solutions by $\mathcal{X}_\star$.

Similarly, we denote a solution of the dual problem by $y_\star$, and the solution set by $\mathcal{Y}_\star$.

Throughout this dissertation, we assume that the solution sets $\mathcal{X}_\star$ and $\mathcal{Y}_\star$ are non-empty.

ε-solution. Our goal is to approximately solve (1.1) (*resp.* (1.4)) to obtain $\boldsymbol{x}_\varepsilon$ in the following sense: Given an accuracy level $\varepsilon > 0$, a point $\boldsymbol{x}_\varepsilon \in \mathcal{X}$ is said to be an ε-solution of (1.1) if

$$f(\boldsymbol{x}_\varepsilon) - f_\star \leq \varepsilon, \text{ and } \operatorname{dist}(\mathcal{A}\boldsymbol{x}_\varepsilon, \mathcal{K}) \leq \varepsilon. \tag{1.15}$$

Here, we call $|f(\boldsymbol{x}_\varepsilon) - f_\star|$ the primal objective residual and $\operatorname{dist}(\mathcal{A}\boldsymbol{x}_\varepsilon, \mathcal{K})$ the feasibility gap. We use the same ε for the objective residual and the feasibility gap for notational simplicity. Distinct choices can be handled simply by scaling f.

Remark that $f(\boldsymbol{x}_\varepsilon) - f_\star$ can take negative values, but the following inequality characterizes the maximum super-optimality in terms of the feasibility gap:

$$f(\boldsymbol{x}_\varepsilon) - f_\star \geq -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_\varepsilon, \mathcal{K}) \quad \text{for any dual solution } \boldsymbol{y}_\star. \tag{1.16}$$

This is an algorithm-independent bound.

Proof. From the Lagrange saddle point theory, we know that the following bound holds $\forall \boldsymbol{x} \in \mathcal{X}$ and $\forall \boldsymbol{v} \in \mathcal{K}$:

$$f_\star \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}_\star) = f(\boldsymbol{x}) + \langle \boldsymbol{y}_\star, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle \leq f(\boldsymbol{x}) + \|\boldsymbol{y}_\star\| \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\| \tag{1.17}$$

where the last inequality holds due to the Cauchy-Schwarz inequality. By setting $\boldsymbol{x} = \boldsymbol{x}_\varepsilon \in \mathcal{X}$ in this inequality and optimizing with respect to $\boldsymbol{v} \in \mathcal{K}$, we get

$$f(\boldsymbol{x}_\varepsilon) - f_\star \geq -\min_{\boldsymbol{v} \in \mathcal{K}} \|\boldsymbol{y}_\star\| \|\mathcal{A}\boldsymbol{x}_\varepsilon - \boldsymbol{v}\| = -\|\boldsymbol{y}_\star\| \operatorname{dist}(\mathcal{A}\boldsymbol{x}_\varepsilon, \mathcal{K}). \tag{1.18}$$

□

2 Storage-Efficient Convex Optimization

This chapter introduces a storage optimal convex optimization paradigm. As a model example, we consider a fundamental class of convex matrix optimization problems with low-rank solutions. We argue that the main obstacle that prevents us from solving these problems at scale is the storage cost. As a proof of concept, we exhibit the first provably correct algorithm, SketchyCGM, for these problems, with optimal storage.

This chapter is based on two distinct joint works, with Ya-Ping Hsieh and Volkan Cevher [YHC15], and with Madeleine Udell, Joel A. Tropp and Volkan Cevher [YUTC17].

Introduction

We consider a fundamental class of matrix optimization problems:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \quad f(\mathcal{A}\boldsymbol{X}) \quad \text{subject to} \quad \|\boldsymbol{X}\|_{S_1} \leq \alpha. \tag{2.1}$$

▷ f is a smooth convex loss function;
▷ $\mathcal{A}:\mathbb{R}^{m\times n}\to\mathbb{R}^d$ is a given linear map;
▷ $\alpha\in\mathbb{R}_+$ is a given model parameter.

Nuclear norm constraint is a powerful proxy that promotes low-rank solutions [Faz02]. When α is well-tuned, we expect solutions to problem (2.1) to be (approximately) low-rank. Validating template (2.1) for low-rank matrix optimization problems is beyond the scope of our work. Our focus is to overcome computational challenges for solving (2.1) in large-scale.

Problem (2.1) is smooth minimization within a nuclear norm-ball of radius α, which is considerably easier to solve than the SDP template (1.4), especially in terms of the arithmetic cost. Nevertheless, this template suffers from the same storage burden, which is evident especially in the data-limited setting where $d \ll mn$. This is a natural assumption since d typically reflects the amount of data, which is typically smaller than the ambient dimensions of the problem. An example of this phenomenon is matrix completion, where the aim is to recover the complete matrix using only a small sample set of its entries.

Let us formalize the challenges that we face in storing the matrix variable.

◦ We need only d units of memory space to store measurements $\mathcal{A}\boldsymbol{X}$, or the problem data. We also assume that $\mathcal{A}$ has an intrinsic structure that makes it easy to store and access. If we think about the matrix completion example, storing indices that are associated with the given sample set is enough to implement $\mathcal{A}$.

◦ Suppose that a solution to problem (2.1) is (approximately) low-rank, denoting the rank by $r \ll \min(m, n)$. Then, we can store this solution (resp. a rank-r approximation of this solution) using only $r(m+n)$ units of memory space in factorized form.

◦ Almost all classical algorithms in the convex optimization literature for solving (2.1) requires initializing and storing a matrix decision variable in the ambient dimensions, which requires a storage unit of size $\Theta(mn)$, much larger than the storage we need for storing the data and a solution!

Our goal is to find a solution to problem (2.1) with rigorous approximation guarantees, but without ever initializing a matrix decision variable in the ambient dimensions.

Contributions

In this chapter, we design two variants of conditional gradient method (CGM) with reduced memory footprint. We can summarize our contributions as follows:

◦ We introduce ThinCGM. ThinCGM operates much like CGM, but it stores a singular value decomposition (SVD) of the matrix variable. While ThinCGM is not theoretically guaranteed to alleviate the expansion of the memory usage, it often keeps a low memory footprint in comparison with the classical CGM. The procedure is exactly the same as CGM, except the way that matrix variable is stored in the memory. Therefore, ThinCGM should be viewed as a careful implementation of CGM rather than a new algorithm.

◦ We initiate the discussion about exploring the limits of storage efficiency for convex optimization algorithms. We formalize the concept of storage-optimality, and we introduce a new paradigm to design storage-optimal convex optimization methods.

◦ We propose SketchyCGM. SketchyCGM is the first storage-optimal convex optimization method for problem (2.1). SketchyCGM maintains a small randomized linear image (a *sketch*) of the matrix variable. After the oprimization procedure is terminated, this sketch is used to extract a provably good low-rank approximation of the solution.

◦ We present computational evidence that SketchyCGM can solve large-scale instances of (2.1) that are not accessible to other convex optimization algorithms in the literature.

Most importantly, our results demonstrate that storage alone is not a reason to drop convexity.

2.1 Conditional Gradient

Conditional gradient method (CGM) plays a crucial role in designing convex optimization algorithms for solving large-scale low-rank matrix optimization problems.

CGM is proposed for the first time in the seminal work of Frank and Wolfe [FW56] for solving smooth convex optimization problem on a polytope. Following the machine learning revolution in optimization, this classical method underwent a significant increase in popularity, especially for matrix factorization problems. It is extended for the smooth convex minimization over the simplex by Clarkson [Cla10], for the spactrahedron by Hazan [Haz08], and finally for an arbitrary compact convex set by Jaggi [Jag13].

Intuition. In essence, at each iteration, CGM solves a linear program formed by the best linear underestimate of the smooth loss function at the current iterate $\boldsymbol{X}_t$. Then, a solution to the original problem is obtained as a convex combination of the solutions to these linear programs.

One can also view CGM as an instance of the dual averaging subgradient method of Nesterov [Nes09]. See Section 3.5 for the details on this relation.

2.1.1 Opportunity

CGM does not need a projection oracle. This is the characteristic feature that makes CGM an irreplaceable algorithm for sparse signal recovery. The reason is simple: The solution to a linear program over a compact set, is provably an extreme point of this set. Since CGM constructs the solution as a convex combination of these extreme points, it is automatically feasible, leaving no reason to perform a projection step.

Why is this a big deal? Because the efficiency of implementing the projection step becomes questionable in large-scale problems. For the low-rank matrix optimization problem, for instance, projection corresponds to a soft-thresholding on the singular values. Hence, it requires computing a full SVD first. On the other hand, a linear minimization over the same nuclear norm-ball requires only the top singular vectors, which is arguably an easier task. In addition, CGM is robust against noise in the linear minimization step. Consequently, we can implement it with an approximate singular vector, which can be obtained efficiently using iterative linear algebra routines such as power method.

An auxiliary benefit is the interpretability of the primal averaging sequence of CGM. CGM forms the decision variable as a convex combination of the extreme points of the problem domain. The extreme points, naturally, have low intrinsic dimensionality. For example, in our model problem (2.1), extreme points are rank-one. Therefore, CGM constructs a solution to (2.1) as the sum of a streaming rank-one matrices. This model of streaming atomic updates is key to tackle the storage burden.

2.1.2 CGM Iteration

We describe the application of CGM for solving (2.1). Choose a feasible initial estimate:

$$\boldsymbol{X}_0 \in \mathbb{R}^{m\times n} \quad \text{where} \quad \|\boldsymbol{X}_0\|_{S_1} \leq \alpha. \tag{2.2}$$

As described, at each iteration $t = 0, 1, 2, \ldots$, CGM minimizes a linear approximation:

$$\begin{aligned} \boldsymbol{H}_t &= \arg\min_{\boldsymbol{H}} \left\{ f(\mathcal{A}\boldsymbol{X}_t) + \langle \boldsymbol{H} - \boldsymbol{X}_t, \mathcal{A}^*(\nabla f(\mathcal{A}\boldsymbol{X}_t))\rangle : \|\boldsymbol{H}\|_{S_1} \leq \alpha \right\} \\ &= \arg\min_{\boldsymbol{H}} \left\{ \langle \boldsymbol{H}, \mathcal{A}^*(\nabla f(\mathcal{A}\boldsymbol{X}_t))\rangle : \|\boldsymbol{H}\|_{S_1} \leq \alpha \right\}. \end{aligned} \tag{2.3}$$

We can implement (2.3) using the following operations:

$$\boldsymbol{H}_t = -\alpha\, \boldsymbol{u}_t \boldsymbol{v}_t^* \quad \text{where} \quad (\boldsymbol{u}_t, \boldsymbol{v}_t) = \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\mathcal{A}\boldsymbol{X}_t))). \tag{2.4}$$

`MaxSingVec` returns the left and right singular vectors that correspond to the maximum singular value. Note that we can efficiently implement `MaxSingVec` using power-method or Lanczos algorithm.

Finally, we update the decision variable:

$$\boldsymbol{X}_{t+1} = (1-\eta_t)\boldsymbol{X}_t + \eta_t \boldsymbol{H}_t \quad \text{where} \quad \eta_t = 2/(t+2). \tag{2.5}$$

Stopping Rule

CGM comes with a natural and easily computable duality gap certificate that can be used to implement a simple stopping criterion.

From convexity of f, we know

$$\begin{aligned} f^\star &\geq f(\mathcal{A}\boldsymbol{X}_t) + \langle \boldsymbol{X}_\star - \boldsymbol{X}_t, \mathcal{A}^*(\nabla f(\mathcal{A}\boldsymbol{X}_t))\rangle \\ &\geq \min_{\boldsymbol{H}} \left\{ f(\mathcal{A}\boldsymbol{X}_t) + \langle \boldsymbol{H} - \boldsymbol{X}_t, \mathcal{A}^*(\nabla f(\mathcal{A}\boldsymbol{X}_t))\rangle : \|\boldsymbol{H}\|_{S_1} \leq \alpha \right\}. \end{aligned} \tag{2.6}$$

Hence, once we compute $\boldsymbol{H}_t$ in (2.4), we can use it to compute the following surrogate:

$$\langle \mathcal{A}\boldsymbol{X}_t - \mathcal{A}\boldsymbol{H}_t, \nabla f(\mathcal{A}\boldsymbol{X}_t)\rangle \geq f(\mathcal{A}\boldsymbol{X}_t) - f^\star. \tag{2.7}$$

As a consequence, given a target suboptimality parameter $\varepsilon > 0$, we can terminate CGM when

$$\langle \mathcal{A}\boldsymbol{X}_t - \mathcal{A}\boldsymbol{H}_t, \nabla f(\mathcal{A}\boldsymbol{X}_t)\rangle \leq \varepsilon. \tag{2.8}$$

This ensures that $f(\mathcal{A}\boldsymbol{X}_t) - f^\star \leq \varepsilon$.

2.2 ThinCGM

The goal is to solve a matrix optimization problem. We are not allowed to form and store a matrix explicitly in ambient dimensions. We assume that the solution is low-rank so that it can be stored efficiently. What is the most straightforward action? Trying to store the matrix variable in a factored form, for example as an SVD.

This idea, however, should be taken with care. None of the algorithmic steps should require the explicit form of the matrix variable internally. In particular, we need an algorithmic procedure to compute SVD of the new estimate after an additive rank-one update (2.5).

2.2.1 Thin SVD Updates

In this section, we overview an algorithmic procedure that computes SVD of $\boldsymbol{X}_{t+1}$ from SVD of $\boldsymbol{X}_t$, without generating $\boldsymbol{X}_t$ in the ambient dimensions. This procedure is a direct application of Brand's ThinSVD updates in [Bra06].

Let $\boldsymbol{X}$ have rank $\hat{r}$, and $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \in \mathbb{R}^{m\times\hat{r}} \times \mathbb{R}^{\hat{r}\times\hat{r}} \times \mathbb{R}^{n\times\hat{r}}$ be its SVD, so that

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*. \tag{2.9}$$

We want to compute SVD of the next estimate defined by the update rule (2.5). Let us define the internal variables

$$\boldsymbol{m} = \boldsymbol{U}^*\boldsymbol{u}, \qquad \boldsymbol{p} = \boldsymbol{u} - \boldsymbol{U}\boldsymbol{m}, \qquad \hat{\boldsymbol{p}} = \frac{\boldsymbol{p}}{\|p\|}, \tag{2.10}$$

$$\boldsymbol{n} = \boldsymbol{V}^*\boldsymbol{v}, \qquad \boldsymbol{q} = \boldsymbol{v} - \boldsymbol{V}\boldsymbol{n}, \qquad \hat{\boldsymbol{q}} = \frac{\boldsymbol{q}}{\|q\|}. \tag{2.11}$$

Then, we compute $(\hat{\boldsymbol{U}}, \hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}})$ by solving a small $(\hat{r}+1)\times(\hat{r}+1)$ dimensional SVD problem

$$\hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^* = (1-\eta)\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{0}\\ \boldsymbol{0} & 0\end{bmatrix} - \eta\alpha\begin{bmatrix}\boldsymbol{m}\\ \|p\|\end{bmatrix}\begin{bmatrix}\boldsymbol{n}\\ \|q\|\end{bmatrix}^* \tag{2.12}$$

Now we can perform the CGM update (2.5) as follows:

$$\boldsymbol{U} \leftarrow \begin{bmatrix}\boldsymbol{U}\hat{\boldsymbol{U}} & \hat{\boldsymbol{p}}\hat{\boldsymbol{U}}\end{bmatrix} \quad \text{and} \quad \boldsymbol{V} \leftarrow \begin{bmatrix}\boldsymbol{V}\hat{\boldsymbol{V}} & \hat{\boldsymbol{q}}\hat{\boldsymbol{V}}\end{bmatrix} \quad \text{and} \quad \boldsymbol{\Sigma} \leftarrow \hat{\boldsymbol{\Sigma}}, \tag{2.13}$$

without ever generating $\boldsymbol{X}_t$ in the ambient space.

Remark. This scheme can also be generalized for additive modifications of any fixed rank. See [Bra06] for more details.

2.2.2 ThinCGM Iteration

Choose a low-rank initial estimate directly in the form of an SVD. We consider $\boldsymbol{X}_0 = \boldsymbol{0}$. Then, perform the initialization:

$$\boldsymbol{U}_0 = \boldsymbol{0} \in \mathbb{R}^m, \quad \boldsymbol{V}_0 = \boldsymbol{0} \in \mathbb{R}^n, \quad \boldsymbol{\Sigma}_0 = 0 \in \mathbb{R}, \quad \text{and} \quad \boldsymbol{y}_0 = \mathcal{A}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \boldsymbol{0} \in \mathbb{R}^d. \tag{2.14}$$

Optionally, fix an $\varepsilon > 0$ for the stopping condition, in order to prescribe a target accuracy.

At each iteration $t = 0, 1, 2, \ldots$, evaluate the linear minimization following the recipe

$$\boldsymbol{h}_t = \mathcal{A}(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*) \quad \text{where} \quad (\boldsymbol{u}_t, \boldsymbol{v}_t) = \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\boldsymbol{y}_t))). \tag{2.15}$$

Set the learning rate $\eta_t = 2/(t+2)$. Update the "dual iterate" $\boldsymbol{y}_t$, and also the factors of the primal variable using the procedure described in Section 2.2.1:

$$\begin{aligned} &\boldsymbol{y}_{t+1} = (1-\eta_t)\boldsymbol{y}_t + \eta_t \boldsymbol{h}_t; \\ &(\boldsymbol{U}_{t+1}, \boldsymbol{\Sigma}_{t+1}, \boldsymbol{V}_{t+1}) = \text{SVDUPDATE}(\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t, \boldsymbol{u}_t, \alpha, \boldsymbol{v}_t, \eta_t). \end{aligned} \tag{2.16}$$

If a stopping criteria is defined, continue until the condition is satisfied:

$$\langle \boldsymbol{y}_t - \boldsymbol{h}_t, \nabla f(\boldsymbol{y}_t) \rangle \le \varepsilon. \tag{2.17}$$

See Algorithm 2.1 for a complete pseudocode.

An important detail here is that we can evaluate `MaxSingVec` in (2.15) without forming $\mathcal{A}^*(\nabla f(\boldsymbol{y}_t))$ in ambient dimensions! To emphasize how, let us consider the compact notation of $\mathcal{A}$, in the same spirit as (1.3):

$$\begin{aligned} \mathcal{A} &: \mathbb{R}^{m\times n} \to \mathbb{R}^d \quad \text{where} \quad \mathcal{A}\boldsymbol{X} = \begin{bmatrix} \langle \boldsymbol{A}_1, \boldsymbol{X} \rangle & \ldots & \langle \boldsymbol{A}_d, \boldsymbol{X} \rangle \end{bmatrix}; \\ \mathcal{A}^* &: \mathbb{R}^d \to \mathbb{R}^{m\times n} \quad \text{where} \quad \mathcal{A}^*\boldsymbol{z} = \textstyle\sum_{i=1}^d z_i \boldsymbol{A}_i. \end{aligned} \tag{2.18}$$

By assumption, $\mathcal{A}$ has low intrinsic complexity, which translates into (2.18) as each coefficient matrix $\boldsymbol{A}_i \in \mathbb{R}^{m\times n}$ has only a few degrees of freedom. For the matrix completion problem, for instance, each $\boldsymbol{A}_i$ is one-sparse.

To evaluate `MaxSingVec`, we just need to run an iterative procedure such as power method, Lanczos algorithm or randomized SVD [HMT11]. These procedures require only an oracle to compute matrix-vector multiplications involving the matrix $\mathcal{A}^*(\nabla f(\boldsymbol{y}))$:

$$\begin{aligned} \mathcal{A}^*(\nabla f(\boldsymbol{y})) &: \mathbb{R}^n \to \mathbb{R}^m \quad \text{where} \quad \mathcal{A}^*(\nabla f(\boldsymbol{y}))\boldsymbol{u} = \textstyle\sum_{i=1}^d z_i \boldsymbol{A}_i \boldsymbol{u} \\ \big(\mathcal{A}^*(\nabla f(\boldsymbol{y}))\big)^* &: \mathbb{R}^m \to \mathbb{R}^n \quad \text{where} \quad \big(\mathcal{A}^*(\nabla f(\boldsymbol{y}))\big)^* \boldsymbol{v} = \textstyle\sum_{i=1}^d z_i \boldsymbol{A}_i^* \boldsymbol{v} \end{aligned} \tag{2.19}$$

Notice that these oracles work in $\mathbb{R}^m$ and $\mathbb{R}^n$ dimensions.

Algorithm 2.1 ThinCGM for model problem (2.1)

Require: Data for (2.1); (optional) suboptimality ε

function THINCGM
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow (\mathbf{0}, \mathbf{0}, \mathbf{0})$ and $\boldsymbol{y} \leftarrow \mathbf{0}$
  for $t \leftarrow 0, 1, 2, 3, \ldots$ **do**
    $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\boldsymbol{y})))$
    $\boldsymbol{h} \leftarrow \mathcal{A}(-\alpha \boldsymbol{u}\boldsymbol{v}^*)$
    if $\langle \boldsymbol{y} - \boldsymbol{h}, \nabla f(\boldsymbol{y}) \rangle \leq \varepsilon$ **then break for**
    end if
    $\eta \leftarrow 2/(t+2)$
    $\boldsymbol{y} \leftarrow (1-\eta)\boldsymbol{y} + \eta \boldsymbol{h}$
    $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \text{SVDUPDATE}(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}, \boldsymbol{u}, \alpha, \boldsymbol{v}, \eta)$
  end for
  return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$
end function

---

function SVDUPDATE($\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}$, $\boldsymbol{u}$, α, $\boldsymbol{v}$, η)
  $\boldsymbol{m} \leftarrow \boldsymbol{U}^* \boldsymbol{u}$ and $\boldsymbol{n} \leftarrow \boldsymbol{V}^* \boldsymbol{v}$
  $\boldsymbol{p} \leftarrow \boldsymbol{u} - \boldsymbol{U}\boldsymbol{m}$ and $\boldsymbol{q} \leftarrow \boldsymbol{v} - \boldsymbol{V}\boldsymbol{n}$
  $\hat{\boldsymbol{p}} \leftarrow \boldsymbol{p}/\|\boldsymbol{p}\|$ and $\hat{\boldsymbol{q}} \leftarrow \boldsymbol{q}/\|\boldsymbol{q}\|$
  $(\hat{\boldsymbol{U}}, \hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{V}}) \leftarrow \texttt{svd}\left((1-\eta) \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} - \eta\alpha \begin{bmatrix} \boldsymbol{m} \\ \|\boldsymbol{p}\| \end{bmatrix} \begin{bmatrix} \boldsymbol{n} \\ \|\boldsymbol{q}\| \end{bmatrix}^* \right)$
  $\boldsymbol{U} \leftarrow \begin{bmatrix} \boldsymbol{U}\hat{\boldsymbol{U}} & \hat{\boldsymbol{p}}\hat{\boldsymbol{U}} \end{bmatrix}$ and $\boldsymbol{V} \leftarrow \begin{bmatrix} \boldsymbol{V}\hat{\boldsymbol{V}} & \hat{\boldsymbol{q}}\hat{\boldsymbol{V}} \end{bmatrix}$ and $\boldsymbol{\Sigma} \leftarrow \hat{\boldsymbol{\Sigma}}$
  return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$
end function

Suppose that the CGM iteration (2.2)–(2.5) generates the sequence $(\boldsymbol{X}_t : t = 0, 1, 2, \ldots)$. It is easy to verify that the ThinCGM iteration generates the sequences $(\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t : t = 0, 1, 2, \ldots)$ such that $\boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^* = \boldsymbol{X}_t$. In other words, ThinCGM is a careful implementation of the standard CGM, rather than a new method. Consequently, ThinCGM maintains the same convergence guarantees as the standard method.

While ThinCGM is not theoretically guaranteed to alleviate the expansion of the rank, it often keeps a low memory footprint in comparison with the classical CGM. The weak point of ThinCGM is that the rank of $\boldsymbol{X}_t$ can increase with t. There is no bound on the rank, beyond the fact that it is clearly less than t (recall that each step brings a rank-one update). CGM exhibits a slow, sublinear convergence rate, which means that we need to run it for many iterations before reaching an approximate solution. As a result, the peak rank of the iterates is often much larger than the rank of the solution.

Remark. After we submitted [YHC15] for review, a discussion on the idea of using thin SVD updates to implement CGM-type methods appeared also in a concurrent work by Freund et al., see Section 3.6 in [FGM17].

2.2.3 A Numerical Study on the Rank Expansion

ThinCGM is a CGM implementation for (2.1), which attempts to maintain a low-rank representation of the decision variable. The intermediate estimates, however, are not guaranteed to be low-rank, even when the sequence starts from and converges towards low-rank points.

Consider the matrix completion problem that arises in machine learning applications such as collaborative filtering [SRJ04]. Suppose that we are trying to construct an unknown, low-rank matrix $\boldsymbol{X}_\natural \in \mathbb{R}^{m\times n}$ from a small sample set E of its entries:

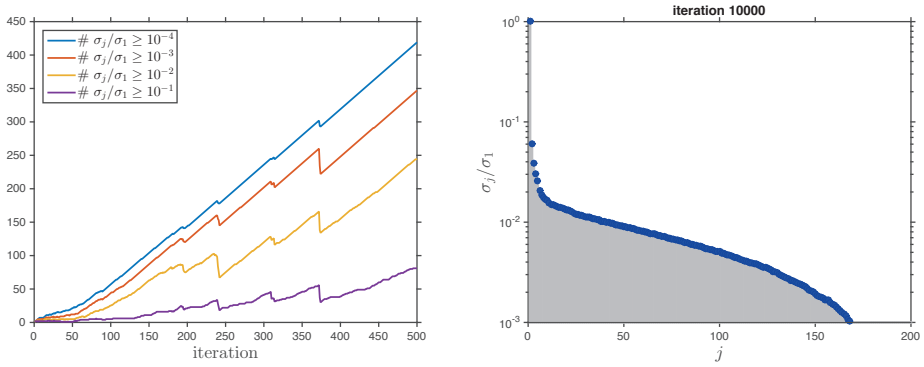
$$b_{ij} = (\boldsymbol{X}_\natural)_{ij} + \xi_{ij} \quad \text{for } (i,j) \in E, \tag{2.20}$$

where ξ_{ij} models the noise. Let us consider a Gaussian noise model, under which we can formulate the problem as an instance of (2.1) with the following setting:

- ▷ Each coefficient matrix (2.18) associated with $\mathcal{A}$ is a distinct, one-sparse $(m \times n)$ matrix.
- ▷ $\boldsymbol{b} := \mathcal{A}\boldsymbol{X}_\natural + \boldsymbol{\xi} \in \mathbb{R}^d$ is a vectorized representation of samples (2.20).
- ▷ $f(\boldsymbol{z}) := \frac{1}{2}\|\boldsymbol{z} - \boldsymbol{b}\|^2$ corresponds to negative log-likelihood estimator of the Gaussian model.

We consider MovieLens 100K dataset [HK16], which consists of 100′000 ratings (1 to 5) from 943 users in 1682 movies. We use a very basic preprocessing step, which only removes the movies that are not rated by any user, and the users that have not provided any ratings.

Figure 2.1 shows the expansion of the rank. On the left panel, we draw the evolution of ε-rank of the estimate. By ε-rank, we mean the number of singular values that exceed $\varepsilon\sigma_1$, where σ_1 is the largest singular value. On the right panel, we plot the singular singular value spectrum of the ground truth, obtained by solving the optimization problem to high accuracy. It is clear from this picture that the rank of the intermediate iterates can be much higher than the rank of the solution.



(a) The ε-rank of CGM iterates

(b) Singular value spectrum of a solution

Figure 2.1 – *Rank expansion of CGM estimates.* ThinCGM is applied to the matrix completion problem with MovieLens 100K dataset.

2.3 SketchyCGM

Our numerical study on matrix completion problem demonstrates that the rank increases steadily, even when the solution has an approximately low-rank representation with a fast decay in the singular value spectrum.

There also appears an auxiliary question in implementing ThinCGM. Due to numerical error, strictly speaking, each rank-one update increases the numerical rank. Then, we need to answer the question which singular values should we consider as noise and truncate. Without going into these details, we present a new approach with provable guarantees on the storage.

Notice that ThinCGM does not use variables $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$ at any step, except for updating the variables itself. In other words, these variables are maintained only to output a solution in the end. Instead, $\boldsymbol{y}_t$ contains all information about $\boldsymbol{X}_t$ that we need to drive CGM iterations. This observation is the important piece to create a CGM variant with optimal storage guarantees.

We propose a new approach, SketchyCGM, which completely discards the exact storage of the decision variable $\boldsymbol{X}$, or its factors. Instead, we maintain a randomized linear image (a sketch) of $\boldsymbol{X}$ which contains enough information to reconstruct a low-rank approximation of the solution once we converge.

Since the iterations are driven by the low-dimensional *dual* variable $\boldsymbol{y}$, SketchyCGM follows exactly the same solution path as the standard CGM. It does not, however, store matrix decision variable, and reconstructs it from a randomized sketch at the end of the optimization procedure. This reconstruction is provably accurate, if the solution attained by CGM is (approximately) low-rank.

Let us summarize a randomized sketching method that we introduced in [TYUC17b]. We provide a more complete overview of this approach in Chapter 6.

2.3.1 Randomized Sketch

Draw and fix two independent standard normal matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Psi}$ where

$$\boldsymbol{\Omega} \in \mathbb{R}^{n \times k} \quad \text{with} \quad k = 2r + 1 \quad \text{and} \quad \boldsymbol{\Psi} \in \mathbb{R}^{\ell \times m} \quad \text{with} \quad \ell = 4r + 3. \tag{2.21}$$

The sketch consists of two matrices $\boldsymbol{Y}$ and $\boldsymbol{W}$ that capture the range and co-range of $\boldsymbol{X}$:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega} \in \mathbb{R}^{m \times k} \quad \text{and} \quad \boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{X} \in \mathbb{R}^{\ell \times n}. \tag{2.22}$$

We can efficiently update the sketch $(\boldsymbol{Y}, \boldsymbol{W})$ to reflect a rank-one update (2.5).

The following procedure provides a rank-r approximation $\hat{\boldsymbol{X}}$ of matrix $\boldsymbol{X}$ from its sketch:

$$\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R} \quad \text{and} \quad \boldsymbol{B} = (\boldsymbol{\Psi}\boldsymbol{Q})^{\dagger}\boldsymbol{W} \quad \text{and} \quad \hat{\boldsymbol{X}} = \boldsymbol{Q}[\boldsymbol{B}]_r, \tag{2.23}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the orthogonal and triangular factors. This procedure yields a rank-r approximation $\hat{\boldsymbol{X}}$ that satisfies

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}} \le 3\sqrt{2}\|\boldsymbol{X} - [\boldsymbol{X}]_r\|_{\mathrm{F}}. \tag{2.24}$$

Similar bounds hold with high probability. See Theorem 6.2 or Theorem 4.3 in [TYUC17b] for more details.

Remark. The recommended sketch size parameters (k, ℓ) in (2.21) are chosen to balance storage against reconstruction quality.

2.3.2 SketchyCGM Iteration

Let us introduce SketchyCGM, the first storage-optimal convex optimization method that produces a low-rank approximation of a solution to (2.1) with rigorous guarantees.

Fix the target rank r. Optionally, also set a suboptimality parameter ε. Independently draw and fix standard normal test matrices $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ and $\boldsymbol{\Psi} \in \mathbb{R}^{\ell\times m}$ as in (2.21).

Initialize the iterate and the sketches:

$$\boldsymbol{y}_0 = \boldsymbol{0} \in \mathbb{R}^d \quad \text{and} \quad \boldsymbol{Y}_0 = \boldsymbol{0} \in \mathbb{R}^{m\times k} \quad \text{and} \quad \boldsymbol{W}_0 = \boldsymbol{0} \in \mathbb{R}^{\ell\times n}. \tag{2.25}$$

At each iteration $t = 0, 1, 2, \ldots$, resolve a solution to the linear minimization subproblem:

$$\boldsymbol{h}_t = \mathcal{A}(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*) \quad \text{where} \quad (\boldsymbol{u}_t, \boldsymbol{v}_t) = \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\boldsymbol{y}_t))). \tag{2.26}$$

Set the learning rate $\eta_t = 2/(t+2)$, and update the variables:

$$\begin{aligned} \boldsymbol{y}_{t+1} &= (1-\eta_t)\boldsymbol{y}_t + \eta_t \boldsymbol{h}_t; \\ \boldsymbol{Y}_{t+1} &= (1-\eta_t)\boldsymbol{Y}_t + \eta_t(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*)\boldsymbol{\Omega}; \\ \boldsymbol{W}_{t+1} &= (1-\eta_t)\boldsymbol{W}_t + \eta_t \boldsymbol{\Psi}(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*). \end{aligned} \tag{2.27}$$

If a stopping criterion is defined, continue this loop until the condition is satisfied:

$$\langle \boldsymbol{y}_t - \boldsymbol{h}_t, \nabla f(\boldsymbol{y}_t)\rangle \le \varepsilon. \tag{2.28}$$

Otherwise, continue until the maximum number of iterations is reached.

Once the iterations terminate, form a rank-r approximate solution $\hat{\boldsymbol{X}}_t$ using the procedure (2.23).

See Algorithm 2.2 for a complete pseudocode.

Algorithm 2.2 SketchyCGM for model problem (2.1)

Require: Data for (2.1); suboptimality ε; target rank r

function SKETCHYCGM
  SKETCH.INIT(m, n, r)
  $\boldsymbol{y} \leftarrow \boldsymbol{0}$
  for $t \leftarrow 0, 1, 2, 3, \ldots$ **do**
    $(\boldsymbol{u}, \boldsymbol{v}) \leftarrow \texttt{MaxSingVec}(\mathcal{A}^*(\nabla f(\boldsymbol{y})))$
    $\boldsymbol{h} \leftarrow \mathcal{A}(-\alpha \boldsymbol{u}\boldsymbol{v}^*)$
    if $\langle \boldsymbol{y} - \boldsymbol{h}, \nabla f(\boldsymbol{y}) \rangle \leq \varepsilon$ **then break for**
    end if
    $\eta \leftarrow 2/(t+2)$
    $\boldsymbol{y} \leftarrow (1-\eta)\boldsymbol{y} + \eta \boldsymbol{h}$
    SKETCH.CGMUPDATE($-\alpha \boldsymbol{u}, \boldsymbol{v}, \eta$)
  end for
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow$ SKETCH.RECONSTRUCT()
  return $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$
end function

——— **Methods for** SKETCH **object** ———

function SKETCH.INIT(m, n, r)
  $k \leftarrow 2r+1$ and $\ell \leftarrow 4r+3$
  $\boldsymbol{\Omega} \leftarrow \texttt{randn}(n, k)$ and $\boldsymbol{\Psi} \leftarrow \texttt{randn}(\ell, m)$
  $\boldsymbol{Y} \leftarrow \texttt{zeros}(m, k)$ and $\boldsymbol{W} \leftarrow \texttt{zeros}(\ell, n)$
end function

function SKETCH.CGMUPDATE($\boldsymbol{u}$, $\boldsymbol{v}$, η)
  $\boldsymbol{Y} \leftarrow (1-\eta)\boldsymbol{Y} + \eta \boldsymbol{u}(\boldsymbol{v}^* \boldsymbol{\Omega})$
  $\boldsymbol{W} \leftarrow (1-\eta)\boldsymbol{W} + \eta(\boldsymbol{\Psi}\boldsymbol{u})\boldsymbol{v}^*$
end function

function SKETCH.RECONSTRUCT()
  $\boldsymbol{Q} \leftarrow \texttt{orth}(\boldsymbol{Y})$
  $\boldsymbol{B} \leftarrow (\boldsymbol{\Psi}\boldsymbol{Q}) \backslash \boldsymbol{W}$
  $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}) \leftarrow \texttt{svds}(\boldsymbol{B}, r)$
  return $(\boldsymbol{Q}\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$
end function

The cost of storing a dual variable $\boldsymbol{y}_t$ is $\Theta(d)$, and the test matrices $(\boldsymbol{\Omega}, \boldsymbol{\Psi})$ and the sketches is $(\boldsymbol{Y}, \boldsymbol{W})$ is $\Theta(r(m+n))$. The `MaxSingVec` subroutine can be implemented with a working storage of size $\Theta(d+m+n)$. Overall, total storage cost of SketchyCGM is $\Theta(d + r(m+n))$.

Definition. We call the amount of storage that we need to specify the problem data and to write an approximate solution as the optimal storage cost. For example, the optimal storage cost of the matrix completion problem in Section 2.2.3 is $\Theta(d + r(m+n))$, where $\Theta(d)$ is required to store $\boldsymbol{b}$ and $\Theta(r(m+n))$ to write a rank-r approximate solution. If the total storage cost of an algorithm does not exceed the optimal storage cost, we say that the algorithm is storage-optimal. On this regard, SketchyCGM is a storage-optimal method!

2.3.3 Convergence Results for SketchyCGM

Suppose that the CGM iterations yield a sequence $(\boldsymbol{X}_t : t = 0, 1, \dots)$ which is converging towards a low-rank point. Then, the SketchyCGM iterations maintain the sketch of $\boldsymbol{X}_t$, from which we can recover an accurate approximation of the low-rank solution. The following theorem formalizes this intuition.

Theorem 2.1. *Suppose that the sequence* $(\boldsymbol{X}_t : t = 0, 1, \dots)$ *constructed by CGM for solving* (2.1) *is converging to a matrix* $\boldsymbol{X}_{\text{cgm}}$. *Let* $\hat{\boldsymbol{X}}_t$ *be the rank-*r *approximation of* $\boldsymbol{X}_t$ *produced by SketchyCGM. Then,*

$$\lim_{t\to\infty} \mathbb{E}\,\|\hat{\boldsymbol{X}}_t - \boldsymbol{X}_{\text{cgm}}\|_{\text{F}} \le 3\sqrt{2}\,\|\boldsymbol{X}_{\text{cgm}} - [\boldsymbol{X}_{\text{cgm}}]_r\|_{\text{F}}. \tag{2.29}$$

In particular, if $\operatorname{rank}(\boldsymbol{X}_{\text{cgm}}) \le r$, *then*

$$\mathbb{E}\,\|\hat{\boldsymbol{X}}_t - \boldsymbol{X}_{\text{cgm}}\|_{\text{F}} \to 0. \tag{2.30}$$

Corollary 2.1. Suppose that all solutions to problem (2.1) are rank-r or less. Then SketchyCGM produces a sequence of approximations that approaches to the solution set $\mathcal{X}_\star$:

$$\mathbb{E}\operatorname{dist}_{\text{F}}(\hat{\boldsymbol{X}}_t, \mathcal{X}_\star) \to 0. \tag{2.31}$$

Recall that the rank of the intermediate estimates can get much higher than the solution rank. Obviously, the information stored in the sketches might not contain enough information to recover an accurate approximation of these intermediate estimates. As a result, a convergence rate guarantee for the general problem template is not achievable.

Nevertheless, we identified a setting where it is possible to prove a bound on the convergence rate. Naturally, this requires stronger assumptions. In particular, we assume that the unique solution is stable in the sense that the objective value increases as we get far from the solution.

Theorem 2.2. *Fix* $\kappa > 0$ *and* $\nu > 0$. *Suppose the (unique) solution* $\boldsymbol{X}_\star$ *of* (2.1) *has* $\operatorname{rank}(\boldsymbol{X}_\star) \le r$, *and that it is stable in the sense that*

$$f(\mathcal{A}\boldsymbol{X}) - f(\mathcal{A}\boldsymbol{X}_\star) \ge \kappa\|\boldsymbol{X} - \boldsymbol{X}_\star\|_{\text{F}}^{\nu} \tag{2.32}$$

for all feasible $\boldsymbol{X}$. *Then we have the error bound*

$$\mathbb{E}\,\|\hat{\boldsymbol{X}}_t - \boldsymbol{X}_\star\|_{\text{F}} \le 6\left(\frac{2C\kappa^{-1}}{t+2}\right)^{1/\nu} \quad \text{for } t = 0, 1, 2, \dots \tag{2.33}$$

where C *is the curvature constant of the problem* (2.1). *See Eqn. (3) in [Jag13] for the definition of the curvature constant. We can also replace* C *by* $L_f D^2$, *where* L_f *is the smoothness constant of* f *and* D *is the domain diameter.*

We refer to the supplements of [YUTC17] for the proofs.

2.4 Extensions for Semidefinite Programming

The principal idea of SketchyCGM can be adopted to design storage-optimal convex optimization algorithms for other structured convex programs. In particular, this dissertation focuses on the SDP template (1.4). We describe the key ingredients to generalize SketchyCGM for solving a broad set of SDP formulations.

First, notice the difference of the domain constraint between the two formulations. The specific domain constraint in CGM affects the linear minimization subproblem. Consider the positive semidefinite (PSD) cone constraint with bounded trace, $\mathcal{X} = \{\boldsymbol{X} : \boldsymbol{X} \in \mathbb{S}_+^n, \operatorname{Tr} \boldsymbol{X} = \alpha\}$. In order to adapt ThinCGM and SketchyCGM to handle problems in this domain, we replace computation (2.26) with

$$\boldsymbol{h}_t = \mathcal{A}(-\alpha \boldsymbol{u}_t \boldsymbol{v}_t^*) \quad \text{where} \quad (\lambda_t, \boldsymbol{u}_t) = \texttt{MinEig}(\mathcal{A}^*(\nabla f(\boldsymbol{y}_t))). \tag{2.34}$$

`MinEig` returns the minimum eigenvalue λ_t and an associated eigenvector $\boldsymbol{u}_t$ of a conjugate symmetric matrix. Similarly, we can also handle $\mathcal{X} = \{\boldsymbol{X} : \boldsymbol{X} \in \mathbb{S}_+^n, \operatorname{Tr} \boldsymbol{X} \leq \alpha\}$ by using

$$\boldsymbol{h}_t = \begin{cases} \mathcal{A}(\alpha \boldsymbol{u}_t \boldsymbol{u}_t^*), & \lambda_t \leq 0 \\ \boldsymbol{0}, & \text{otherwise,} \end{cases} \qquad \text{where} \qquad (\lambda_t, \boldsymbol{u}_t) = \texttt{MinEig}(\mathcal{A}^*(\nabla f(\boldsymbol{y}_t))). \tag{2.35}$$

We also make small changes in the matrix sketch, in order to preserve PSD structure in the reconstruction algorithm, as described in our Algorithm 9 in [TYUC17b]. Alternatively, we can also develop more efficient sketching methods specifically for PSD matrices, see Section 6.3 and Algorithm 3 [TYUC17a].

Recall that CGM does not apply for problems with affine constraints. Consequently, we cannot directly use SketchyCGM for solving SDP in standard form. One quick remedy is to use the standard reduction approach of the SDP template to a sequence of feasibility problems by performing a binary search over the optimal value of $\langle \boldsymbol{C}, \boldsymbol{X} \rangle$, see Section 1.2 in [Gar12]. In large-scale instances, however, the binary search over the model parameters imposes additional computational cost.

Instead, we design new practical optimization algorithms that are compatible with sketching for solving SDP template (1.4). We describe these methods in the following chapters.

We say SketchyCGM is storage-optimal because the working storage is $\mathcal{O}(d + r(m+n))$, which does not exceed the storage required to store the data and a solution. We conclude this section by highlighting a recent result from Waldspurger and Waters [WW18], which demonstrates that the non-convex Burer-Monteiro factorization idea (1.8) cannot lead to storage-optimal algorithms for the standard form SDP template. In particular, they show that the factorization rank must be set in the order Pataki-Barvinok rank even for some problems with a unique rank-one solution, in order to avoid spurious stationary points in the problem formulation. Accordingly, convexity seems to be the key to achieve storage-optimality!

2.5 Numerical Experiments

We present our numerical experiments on matrix completion and phase retrieval problems, to demonstrate flexibility, scalability and robustness of our algorithms.

2.5.1 Matrix completion

Instate the matrix completion problem description from Section 2.2.3. SketchyCGM is flexible, i.e., it can solve any instance of (2.1). In Section 2.2.3, we considered a Gaussian noise model, which yields a quadratic loss function. In this section, we also consider Gauss–Laplace and Bernoulli noise models, which correspond to the huber and logistic loss, respectively.

With logistic loss, we consider a binary classification model. Only for this experiment, we binarize the data using the formula $\boldsymbol{b} \leftarrow \operatorname{sign}(\boldsymbol{b} - 3.5)$ as part of the preprocessing. For other experiments, with quadratic and huber loss, we keep the ratings intact, with no preprocessing.

For the MovieLens 100K dataset, we use the default `ub` train and test partition of the data. For each loss function, we sweep α from 3′000 to 10′000 in steps of 500. We select the value of α that provides the best test error after 10′000 iterations of CGM. A similar procedure applies to the MovieLens 10M dataset with the `rb` partition. This time, we sweep α from 50′000 to 250′000 in steps of 25′000.

We compute the test error as follows: At the end of each iteration, we execute sketch reconstruction procedure to obtain a rank-r approximate solution. Then, we evaluate the loss function for each datapoint in the test set, and compute the average loss of the test partition. We report this average value as the test error.

Figures 2.2 and 2.3 present the evolution of test error for SketchyCGM as a function of the iteration counter, for various values of the rank parameter r in SketchyCGM. For the 100K dataset, rank $r = 50$ yields test error similar with the CGM solution. For the 10M dataset, rank $r = 200$ yields equivalent performance.

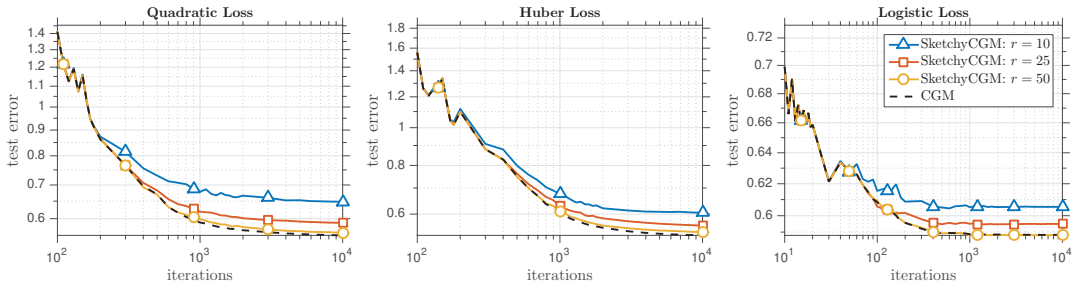


Figure 2.2 – *MovieLens 100K Dataset.* Convergence of test error for SketchyCGM with three loss functions. The parameter α is chosen by cross-validation on the final CGM solution: $\alpha = 7{,}000$ for quadratic loss, 7,500 for Huber loss, and 4,500 for logistic loss.

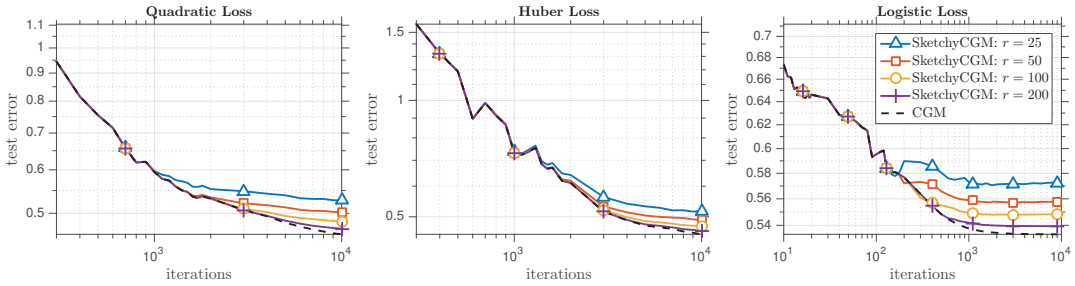


Figure 2.3 – *MovieLens 10M Dataset.* Convergence of test error for SketchyCGM with three loss functions. The parameter α is chosen by cross-validation on the final CGM solution: $\alpha = 150{,}000$ for quadratic and logistic losses, and 175,000 for Huber loss.

2.5.2 Phase Retrieval Problems

The problem of retrieving the phase of a signal from its intensity-only measurements has regained significant attention recently [CSV12, SEC+15, CZH+17]. Formally, phase retrieval problem aims to recover a signal $\boldsymbol{x}^\natural \in \mathbb{C}^n$ from d noisy quadratic measurements

$$b_i = |\langle \boldsymbol{a}_i, \boldsymbol{x}_\natural \rangle|^2 + \xi_i \quad \text{for } i = 1, \ldots, d. \tag{2.36}$$

where $\boldsymbol{a}_i \in \mathbb{C}^n$ are the given measurement vectors, and ξ_i model the unknown noise. This problem arises in many applications, including X-ray crystallography, astronomical imaging and microscopy imaging.

As an estimation problem, the nonlinear observation model (2.36) poses significant difficulties, since the standard maximum likelihood estimators yield non-convex optimization problems. Convex relaxation is useful in this setting. Let us recast the quadratic measurements (2.36) as

$$\begin{aligned} b_i &= |\langle \boldsymbol{a}_i, \boldsymbol{x}_\natural \rangle|^2 + \xi_i \\ &= \boldsymbol{a}_i^* \boldsymbol{x}_\natural \boldsymbol{x}_\natural^* \boldsymbol{a}_i + \xi_i \\ &= \mathrm{Tr}(\boldsymbol{a}_i^* \boldsymbol{x}_\natural \boldsymbol{x}_\natural^* \boldsymbol{a}_i) + \xi_i \\ &= \mathrm{Tr}(\boldsymbol{a}_i \boldsymbol{a}_i^* \boldsymbol{x}_\natural \boldsymbol{x}_\natural^*) + \xi_i \\ &= \langle \boldsymbol{A}_i, \boldsymbol{X}_\natural \rangle + \xi_i \qquad \text{where} \qquad \boldsymbol{A}_i = \boldsymbol{a}_i \boldsymbol{a}_i^*, \qquad \text{and} \qquad \boldsymbol{X}_\natural = \boldsymbol{x}_\natural \boldsymbol{x}_\natural^* \in \mathbb{S}_+^n \end{aligned} \tag{2.37}$$

This leads to the following linear observation model in the *lifted* dimensions:

$$\boldsymbol{b} = \mathcal{A}\boldsymbol{X}_\natural + \boldsymbol{\xi}, \qquad \text{where} \qquad \boldsymbol{X}_\natural = \boldsymbol{x}_\natural \boldsymbol{x}_\natural^* \in \mathbb{S}_+^n \tag{2.38}$$

As a result, data fidelity costs can be measured using convex functions, albeit in terms of matrix variables with rank constraint. We relax the problem to a convex formulation by dropping the rank constraint and obtain an SDP problem.

Scalability Test with Synthetic Phase Retrieval with Gaussian Noise

Let us first investigate the scalability of our approach on a synthetic phase retrieval setup. We consider a measurement model based on random coded diffraction patterns with octonary modulation; see [CLS15b].

We construct a synthetic data, here are the details. We draw a vector $\boldsymbol{x}_\natural \in \mathbb{C}^n$ from the standard normal distribution. Then we acquire $d = 10n$ intensity measurements (2.36), with *iid* Gaussian noise such that the SNR is 20 dB. We choose $\alpha = d^{-1}\sum_{i=1}^{d} b_i$, which gives an accurate estimate. See Section II of [YHC15] for more details on this particular choice of α.

We compare five convex optimization algorithms: the classic proximal gradient method (PGM) [Roc76]; the Auslender–Teboulle (AT) accelerated method [AT06]; the classic CGM algorithm [Jag13]; the PSD variants of ThinCGM and SketchyCGM (with rank parameter $r = 1$).

We solve the following convex optimization template

$$\underset{\boldsymbol{X}\in\mathbb{C}^{n\times n}}{\text{minimize}} \quad f(\mathcal{A}\boldsymbol{X}) \quad \text{subject to} \quad \boldsymbol{X} \in \mathbb{S}_+^n, \quad \operatorname{Tr}\boldsymbol{X} \le \alpha, \tag{2.39}$$

with quadratic loss ($f(\mathcal{A}\boldsymbol{X}) := \frac{1}{2}\|\mathcal{A}\boldsymbol{X} - \boldsymbol{b}\|^2$).

Figure 2.4(A) displays storage costs for each algorithm as the signal length n increases. We approximate memory usage by reporting the total workspace allocated by MATLAB for the algorithm. PGM, AT, and CGM have static allocations, but they use a matrix variable of size n^2. ThinCGM attempts to maintain a low-rank approximation of the decision variable, but the rank increases steadily, so the algorithm fails after $n = 10^5$. In contrast, SketchyCGM has a static memory allocation of $\Theta(n)$. It already offers the best memory footprint for $n = 10^2$, and it still works when $n = 10^6$.

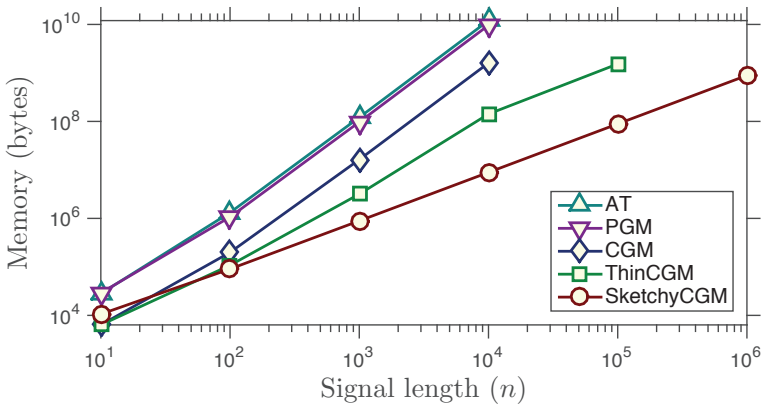


Figure 2.4 – *Memory Scaling.* Memory scaling for five convex optimization algorithms applied to a synthetic instance of the convex phase retrieval problem.

Flexibility Test with Synthetic Phase Retrieval with Poisson Noise

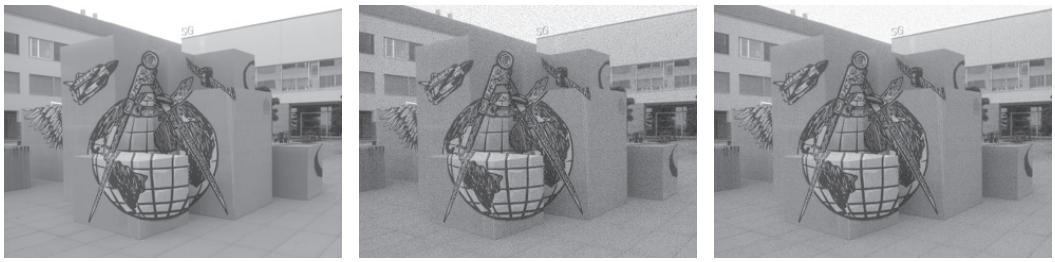
Poisson noise model can capture the physics of an imaging system better than a Gaussian noise model [SEC+15, OLY+16, CLDM18]. In this section, we demonstrate the flexibility of SketchyCGM by solving (2.39) with Poisson loss function: $f(\boldsymbol{z}) := \sum_i z_i - b_i \log(z_i)$.

Remark that Poisson loss is not a smooth function, hence the standard CGM does not directly apply. In [OLY+16], however, we show that CGM can be adapted to solve this problem with minor modifications on the step-size and initialization: We initialize the algorithm with the dual vector $\boldsymbol{y}_0 = d^{-1/2}\mathbf{1}$ and set the learning rate $\eta_t = 2/(3+t)$.

We consider a similar setup to the previous section with Gaussian noise model, but we use real images rather than random signals so that we can visualize the effects of choosing the appropriate noise model. We choose a small image $\boldsymbol{x}_\natural \in \mathbb{C}^n$ with $n = 240 \times 320 = 76'800$ pixels. We acquire $d = 20n$ measurements of the form (2.36) using the same coded diffraction model as in Gaussian noise model. Each $\xi_i \in \mathbb{R}^d$ is drawn *iid* from a Poisson distribution. We choose the mean of this distribution so that the SNR is 20 dB. Once again, we set α to the average of the measurements b_i.

We set the rank parameter $r = 1$, and run SketchyCGM for 100 iterations. To recover an approximate solution of the imaging problem, we take the top singular vector of the solution.

Figure 2.5 displays the recoveries of obtained. We compare against the recovery obtained with the Gaussian noise model. As expected, the Poisson formulation performs better.



(a) Original image (240 × 320) (b) Gaussian loss: PSNR 26.89 dB (c) Poisson loss: PSNR 32.12 dB

Figure 2.5 – *Gaussian and Poisson phase retrieval under Poisson noise.* See Section 2.5.2 for details.

Fourier Ptychography

Lifting yields an obvious curse of dimensionality, expanding the search space from n to n^2 dimensions. As a result, SDP relaxations have been thought to be impractical for phase retrieval problems of a realistic size.

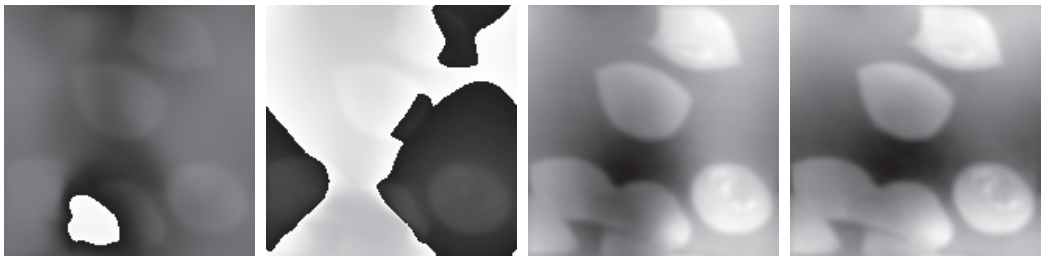
In this section, we demonstrate an application of SketchyCGM to a phase retrieval problem from a real-world microscopy imaging application. The authors of [HCO$^+$15] provided measurements of a slide containing human blood cells from a working Fourier ptychography microscopy system. See [HCO$^+$15] for the technical details of this imaging modality.

The measurement vectors $\boldsymbol{a}_i$ are obtained from windowed discrete Fourier transforms. The data consists of 29 image samples of (80×80) pixels from different illuminations. Hence, the number of measurements $d = 29 \cdot 80 \cdot 80 = 185'600$. The final image resolution is (160×160), which we treat as a vector of size $25'600$. After lifting, this yields a complex valued $(25'600 \times 25'600)$ dimensional PSD matrix problem. We consider the phase retrieval problem with Gaussian noise model (*i.e.*, with quadratic loss function), and we choose $\alpha = 1'400$.

We run SketchyCGM with rank parameter $r = 1$ for $10'000$ iterations. We use the standard basic rounding step, which takes the leading eigenvector of the solution. We compare against two non-convex approaches. The Wirtinger Flow method [CLS15b] that we implemented with the recommended parameters, and the Burer–Monteiro method [BM03] (reconstruction provided by the authors of [HCO$^+$15]).

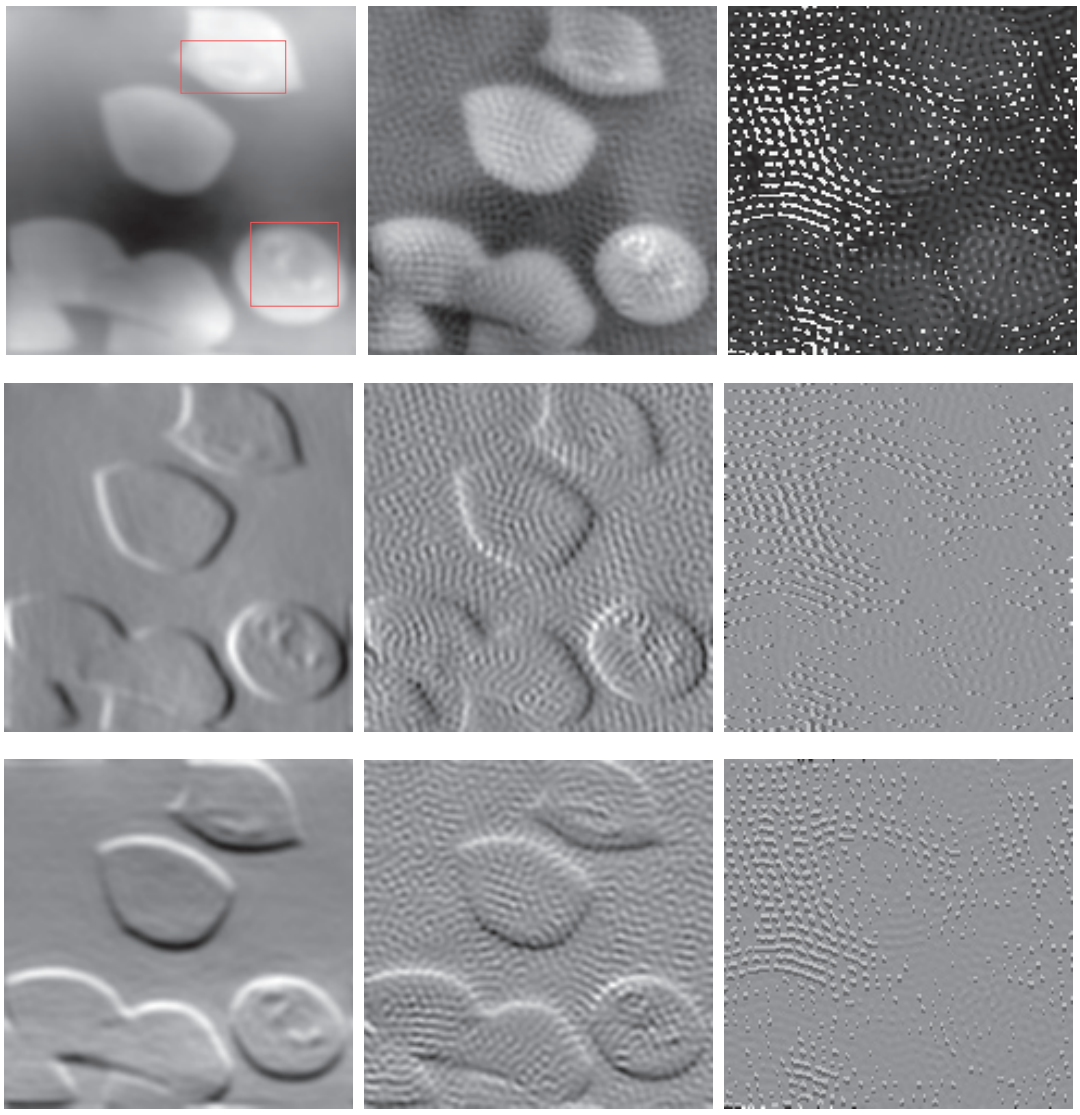
First row of Figure 2.7 displays the phase of the reconstruction obtained as a result of this procedure. Roughly, the phase indicates the thickness of the sample at a given location. Second and third rows show the phase gradient to provide an alternative view. SketchyCGM produces a superior quality image than the non-convex heuristics. These results are consistent with Figure 4 in [HCO$^+$15], which indicates 5–10 dB improvement of convex optimization over non-convex heuristics.

Figure 2.6 displays snapshots of the SketchyCGM iterates as the algorithm proceeds. We see that SketchyCGM already achieve a diagnostic quality image even after $1'000$ iterations.



(a) iteration 10 (b) iteration 100 (c) iteration 1,000 (d) iteration 10,000

Figure 2.6 – *Evolution of the Reconstruction Quality of SketchyCGM for Fourier Ptychography.* Hue indicates the complex phase of the signal.



(a) SketchyCGM (b) Burer-Monteiro[BM03, HCO+15] (c) Wirtinger Flow [CLS15b]

Figure 2.7 – *Three algorithms for Fourier ptychographic imaging via phase retrieval.* [*top*] Brightness indicates the complex phase of a pixel. Only relative differences in brightness are meaningful. Red boxes mark malaria parasites in blood cells. [*middle*] The horizontal differences of the phase maps presented. [*bottom*] The vertical differences of the phase maps.

3 Universal Primal-Dual Methods

In Chapter 2, we described a storage-optimal convex optimization paradigm for solving low-rank matrix optimization problems. Two major efficiencies of the conditional gradient method (CGM) play a crucial role in our paradigm:

- CGM executes rank one updates on the decision variable $\boldsymbol{X}$,
- Even if we discard the decision variable $\boldsymbol{X}$ from the workspace, CGM iterations can be driven by a low-dimensional "dual" variable.

In order to extend our framework for solving the SDP formulation (1.4), we need to develop algorithms for the model problem, which maintain these two characteristic features.

In this chapter, we describe our primal-dual (sub)gradient methods with these two features. Our algorithms are based on the Nesterov's universal gradient method [Nes15] applied to the non-smooth dual formulation of the model problem (1.1). We equip our algorithms with a new primal averaging mechanism to construct the primal decision variable through structured updates, which exhibits rank one updates when we use our algorithms for solving instances from the SDP formulation (1.4).

The content in this chapter is based on the joint work with Quoc Tran-Dinh and Volkan Cevher [YTDC15a].

Introduction

Processing the affine constraint ($\mathcal{A}\boldsymbol{x} \in \mathcal{K}$) in (1.4) require significant computational effort in large-scale setting. The majority of scalable numerical algorithms for constrained optimization are primal-dual-type, including decomposition, augmented Lagrangian and alternating direction methods: See [CP08, BPC$^+$11, GEB13, ST14, TDC14, LM15] and the references therein.

Primal-dual methods are a natural choice for designing storage-optimal methods. This is simply because we can use the low-dimensional dual variable to drive iterations in the dual space. Nevertheless, the majority of the primal-dual methods rely on a proximal-type (or

projection-type) oracle to recover the primal variable from the dual methods. As the problem dimensions become increasingly larger, the proximal tractability assumption can be restrictive. More importantly, the proximal operator requires full dimensional computations in the primal problem space, which eliminates the storage benefits.

Contributions

This chapter describes our universal primal-dual convex optimization algorithms.

◦ Our algorithmic approach features a gradient method and its accelerated variant that operates on the dual formulation of (1.1). Given the dual method, we then use a new averaging scheme to construct primal-iterates for the problem template (1.1). This averaging scheme results in streaming rank one updates on the primal decision variable when solving our model SDP template (1.4). Consequently, our algorithms are amenable to thin SVD updates for increased storage efficiency, or to sketching for storage optimality.

◦ Our algorithms trade-off the computational difficulty in the primal subproblems with the difficulty of optimization in the dual subproblems by avoiding the proximal mappings and using a subgradient approach instead. The dual subgradient mapping requires evaluation of a Fenchel-type oracle that we call as the *sharp operator*. Sharp operator can be viewed as a generalization of the linear minimization oracle (lmo) of CGM. In particular, when applied to the SDP template (1.4), our sharp operator can be cast as an instance of the lmo.

◦ Unlike CGM, our approach can handle non-smooth objectives and affine constraints in (1.4) efficiently. Due to the special structure of the dual problem, we can always assume that it has some Hölder smoothness (also called as weak smoothness) of order $\nu \in [0,1]$. As a result, we tailor the universal gradient algorithms of Nesterov [Nes15] for solving the dual problem. Although our algorithms do not require the knowledge of the smoothness parameters a priori, they achieve the optimal worst-case complexity guarantees in the sense of first-order black-box models [NY83], with respect to the smoothness order of the *dual problem*.

◦ We extend the convergence analysis in Nesterov's universal gradient methods [Nes15] to the primal-dual setting. We provide a rigorous analysis of the convergence rate and the worst-case complexity of these algorithms.

◦ We present an accelerated variant of our algorithm based on the combination of our analysis with the FISTA approach [BT09]. This variant requires fewer evaluations of projection onto $\mathcal{K}$ than a direct extension of the fast universal gradient method of Nesterov [Nes15].

◦ We present numerical experiments in matrix completion and quantum tomography applications to demonstrate the flexibility and scalability of the proposed approach.

3.1 Preliminaries

Let us first recall some basic notions from the primal-dual optimization.

Introduce a slack variable $\boldsymbol{v} = \mathcal{A}\boldsymbol{x} \in \mathcal{K}$, and form the Lagrangian:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) := (\boldsymbol{x}) + \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle. \tag{3.1}$$

Minimize the Lagrangian over the constraint set to obtain the dual function ϕ:

$$\begin{aligned} \phi(\boldsymbol{y}) := \min_{\boldsymbol{x}\in\mathcal{X}} \min_{\boldsymbol{v}\in\mathcal{K}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) &= \min_{\boldsymbol{x}\in\mathcal{X}} \min_{\boldsymbol{v}\in\mathcal{K}} f(\boldsymbol{x}) + \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle \\ &= \min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) + \langle \mathcal{A}^* \boldsymbol{y}, \boldsymbol{x} \rangle - \max_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{y}, \boldsymbol{v} \rangle. \end{aligned} \tag{3.2}$$

Now we can write the dual problem as

$$\underset{\boldsymbol{y}\in\mathbb{R}^d}{\text{maximize}} \quad \phi(\boldsymbol{y}). \tag{3.3}$$

This is an unconstrained concave maximization problem.

To follow the common notational convention, we define the negation of the dual function as $g(\boldsymbol{y}) := -\phi(\boldsymbol{y})$ and work with a convex minimization formulation. In the rest of this chapter, we will call g as the dual function relaxing the mathematical terminology, but recall that it is in fact the negation of it. We decompose g as

$$\begin{aligned} g(\boldsymbol{y}) = g_x(\boldsymbol{y}) + g_v(\boldsymbol{y}) \quad \text{where} \quad g_x(\boldsymbol{y}) &= -\min_{\boldsymbol{x}\in\mathcal{X}} \left[f(\boldsymbol{x}) + \langle \mathcal{A}^* \boldsymbol{y}, \boldsymbol{x} \rangle \right] \\ \text{and} \quad g_v(\boldsymbol{y}) &= \max_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{y}, \boldsymbol{v} \rangle. \end{aligned} \tag{3.4}$$

Now we can write an equivalent (negated) dual problem as

$$\underset{\boldsymbol{y}\in\mathbb{R}^d}{\text{minimize}} \quad g_x(\boldsymbol{y}) + g_v(\boldsymbol{y}). \tag{3.5}$$

This is a composite convex minimization formulation. Since g_x and g_v are generally non-smooth, FISTA and its proximal-based analysis [BT09] are not directly applicable.

3.2 Universal Primal-Dual Method

In this section, we introduce our universal primal-dual (sub)gradient method, UPD, for solving (1.1). UPD applies Nesterov's universal gradient method from [Nes15] for solving the dual problem (3.5). To recover the primal variable, we introduce an ergodic averaging mechanism.

Let us first define a Fenchel-type oracle, that we call as the *sharp operator* in the rest of this dissertation. The sharp operator plays an important role in our algorithmic design. Given an

input variable $\boldsymbol{x} \in \mathbb{R}^p$, the sharp operator returns $\boldsymbol{h} \in \mathbb{R}^p$ such that

$$\boldsymbol{h} = \text{sharp}_{f,\mathcal{X}}(\boldsymbol{x}) \quad \text{where} \quad \boldsymbol{h} \in \arg\min_{\boldsymbol{u}\in\mathcal{X}} f(\boldsymbol{u}) + \langle \boldsymbol{x}, \boldsymbol{u} \rangle. \tag{3.6}$$

We can view the sharp operator as a generalization of the lmo of CGM, as it becomes a linear minimization subproblem over the the domain $\mathcal{X}$ when f is a linear function.

Using the sharp operator, we can write a subgradient of g_x, defined in (3.5), as follows:

$$\nabla g_x(\boldsymbol{y}) = -\mathcal{A}\boldsymbol{h} \quad \text{where} \quad \boldsymbol{h} = \text{sharp}_{f,\mathcal{X}}(\mathcal{A}^*\boldsymbol{y}). \tag{3.7}$$

3.2.1 Efficiency Considerations and Hölder Smoothness

The difficulty of solving an unconstrained optimization problem depends on the problem geometry, *i.e.,* the functional properties of the objective function, such as the smoothness and the strong convexity.

For instance, when the objective function is smooth, we can use accelerated first-order methods with $\mathcal{O}(1/\sqrt{\varepsilon})$ iteration complexity, to achieve an ε-suboptimal solution. On the other hand, iteration complexity for solving non-smooth problems is incomparably worse, on the order of $\mathcal{O}(1/\varepsilon^2)$.

The classical optimization literature tackles smooth and non-smooth (with bounded subgradient) problems with distinct algorithms, specifically designed to get the optimal guarantees for the target problem. However, designing a single algorithm which works for both cases, without any prior information about the smoothness, is desirable for two practical reasons:

1. Smoothness of the problem might be unknown a priori, or it might be expensive to evaluate the smoothness parameters.
2. Locally, the problem might possess more favorable structures than the global geometry. We might expect a better empirical performance from an adaptive method in this case.

Motivated by its practical significance, there is a recent attempt to design *universal* gradient methods. Universal, in this context, means that the method can adapt to the unknown smoothness level and smoothness parameters of the problem, as defined by Nesterov in [Nes15].

Nesterov's universal gradient methods are the first to achieve this goal [Nes15]. His methods are based on an inexact line-search strategy with an additional slackness term. More recently, some line-search-free universal methods are introduced [Lev17]. These methods typically rely on an online-to-offline conversion of the AdaGrad method [DHS11]. We have also contributed to this line of research, by introducing the first line-search-free accelerated method with learning rate adaptation [LYC18].

Hölder smoothness. To close the gap between the complexity bounds for smooth and non-smooth problems, Nesterov's universal gradient methods also consider the Hölder smoothness as the intermediate class.

Let ∇g_x be a subgradient of g_x. We define the Hölder smoothness constant of order $\nu \geq 0$ as

$$M_\nu := \max_{\substack{\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathbb{R}^d \\ \boldsymbol{y}_1 \neq \boldsymbol{y}_2}} \frac{\|\nabla g_x(\boldsymbol{y}_1) - \nabla g_x(\boldsymbol{y}_2)\|}{\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|^\nu} \tag{3.8}$$

Note that the parameter M_ν explicitly depends on ν. We are interested in the case $\nu \in [0,1]$, and *especially the two extremal cases*: $\nu = 1$ corresponds to the smooth problems with Lipschitz continuous gradients, and $\nu = 0$ corresponds to the non-smooth problems with bounded subgradients.

Assumption. We assume that there exists a finite smoothness constant M defined as

$$M := \min_{0 \leq \nu \leq 1} M_\nu(g) < +\infty. \tag{3.9}$$

This is a reasonable assumption. We explain this claim with the following examples:

◦ If $\mathcal{X}$ is bounded, then $\nabla g_x(\cdot)$ is also bounded. Indeed, we have

$$\begin{aligned} \|\nabla g_x(\boldsymbol{y})\| = \|-\mathcal{A}^* \boldsymbol{h}\| \leq D_{\mathcal{X}} \|\mathcal{A}\| \quad \text{since} \quad \boldsymbol{h} &= \operatorname{sharp}_{f,\mathcal{X}}(\mathcal{A}^* \boldsymbol{y}) \\ &\in \arg\min_{\boldsymbol{u} \in \mathcal{X}} f(\boldsymbol{u}) + \langle \mathcal{A}^* \boldsymbol{y}, \boldsymbol{u} \rangle. \end{aligned} \tag{3.10}$$

Hence, we can choose $\nu = 0$ and $M = D_{\mathcal{X}} \|\mathcal{A}\|$. Our SDP template (1.4) fits into this case.

◦ If f is uniformly convex with convexity parameter $\mu_f > 0$ and degree $q \geq 2$, *i.e.,*

$$\langle \nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2), \boldsymbol{x}_1 - \boldsymbol{x}_2 \rangle \geq \mu_f \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^q \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^p, \tag{3.11}$$

then g_x satisfies (3.8) with $\nu = \frac{1}{q-1}$ and $M = \left(\mu_f^{-1} \|\mathcal{A}\|^2\right)^{\frac{1}{q-1}}$, as shown in [Nes15].

◦ In particular, if f is μ_f-strongly convex (*i.e.,* uniformly convex with degree $q = 2$), then $\nu = 1$ and g_x is smooth with parameter $M = \mu_f^{-1} \|\mathcal{A}\|^2$.

3.2.2 UPD Iteration

Set the desired accuracy $\varepsilon > 0$, and an initial step-size σ_{-1}. Initialize the primal and dual variable, and the algorithm parameters

$$\boldsymbol{x}_0 \in \mathbb{R}^p \quad \text{and} \quad \boldsymbol{y}_0 \in \mathbb{R}^d \quad \text{and} \quad S_{-1} = 0. \tag{3.12}$$

At each iteration $t = 0, 1, 2, \ldots$, compute a subgradient using the sharp operator:

$$\nabla g_x(\boldsymbol{y}_t) = -\mathcal{A}\boldsymbol{h}_t \quad \text{where} \quad \boldsymbol{h}_t = \operatorname{sharp}_{f,\mathcal{X}}(\mathcal{A}^*\boldsymbol{y}_t). \tag{3.13}$$

Then, evaluate the partial dual objective

$$g_x(\boldsymbol{y}_t) = -f(\boldsymbol{h}_t) - \langle \boldsymbol{y}_t, \mathcal{A}\boldsymbol{h}_t \rangle \tag{3.14}$$

UPD executes a line-search to find an appropriate dual step-size as follows:
For $j = -1, 0, 1, \ldots$, compute a trial point

$$\begin{aligned} \hat{\boldsymbol{y}} &= \operatorname{prox}_{\hat{\sigma} g_v}(\boldsymbol{y}_t - \hat{\sigma}\nabla g_x(\boldsymbol{y}_t)) \\ &= \boldsymbol{y}_t - \hat{\sigma}\nabla g_x(\boldsymbol{y}_t) - \hat{\sigma}\operatorname{proj}_{\mathcal{K}}(\hat{\sigma}^{-1}(\boldsymbol{y}_t - \hat{\sigma}\nabla g_x(\boldsymbol{y}_t))) \quad \text{where} \quad \hat{\sigma} = 2^j\sigma_{t-1}. \end{aligned} \tag{3.15}$$

The second line follows from the Moreau decomposition, since g_v is the support function of $\mathcal{K}$. We continue to the line-search procedure until the following condition is satisfied:

$$g_x(\hat{\boldsymbol{y}}) \le g_x(\boldsymbol{y}_t) + \langle \nabla g_x(\boldsymbol{y}_t), \hat{\boldsymbol{y}} - \boldsymbol{y}_t \rangle + \frac{1}{2\hat{\sigma}}\|\hat{\boldsymbol{y}} - \boldsymbol{y}_t\|^2 + \frac{\varepsilon}{2}. \tag{3.16}$$

This is the usual descent lemma for smooth convex minimization, except the additional slackness term $\varepsilon/2$. Once the condition is satisfied, we set $\boldsymbol{y}_{t+1} = \hat{\boldsymbol{y}}$ and $\sigma_t = \hat{\sigma}$.

It only remains to update the primal variable. First, update the accumulated sum of the old step-sizes, S_t. Then, compute the primal step-size η_t as follows:

$$\eta_t = \sigma_t / S_t \quad \text{where} \quad S_t = S_{t-1} + \sigma_t. \tag{3.17}$$

Finally, perform CGM-like update on the primal variable

$$\boldsymbol{x}_{t+1} = (1 - \eta_t)\boldsymbol{x}_t + \eta_t \boldsymbol{h}_t. \tag{3.18}$$

3.2.3 Guarantees

Theorem 3.1. *Suppose that we apply UPD for solving the constrained minimization problem* (1.1). *Assume that strong duality holds. Further assume that the objective function f has Hölder continuous (sub)gradients of some order $\nu \in [0, 1]$, i.e., the assumption* (3.9) *holds. Then, the sequence $(\boldsymbol{x}_t : t = 0, 1, \ldots)$ generated by UPD satisfies*

$$f(\boldsymbol{x}_t) - f_\star \ge -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \tag{3.19}$$

$$f(\boldsymbol{x}_t) - f_\star \le \frac{\hat{M}_\varepsilon \|\boldsymbol{y}_0\|^2}{t+1} + \frac{\varepsilon}{2} \tag{3.20}$$

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \le \frac{4\hat{M}_\varepsilon}{t+1}\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \sqrt{\frac{2\varepsilon\hat{M}_\varepsilon}{t+1}} \tag{3.21}$$

where $\boldsymbol{y}_\star$ denotes any solution of the dual problem, and

$$\hat{M}_\varepsilon := \left[\frac{1-\nu}{1+\nu}\frac{1}{\varepsilon}\right]^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}}, \qquad \text{for a given target accuracy } \varepsilon > 0. \tag{3.22}$$

Corollary 3.1 (Iteration complexity)**.** Instate the assumptions of Theorem 3.1. Then, UPD provably achieves an ε-solution of (1.1) after at most t_ε iterations, where

$$t_\varepsilon = \mathcal{O}\left(\left(\frac{M_\nu}{\varepsilon}\right)^{\frac{2}{1+\nu}}\right). \tag{3.23}$$

This matches the iteration complexity of the non-accelerated first-order methods, which is optimal for $\nu = 0$, but not for $\nu \in (0,1]$. See [NY83].

Corollary 3.2 (Oracle complexity)**.** Instate the assumptions of Theorem 3.1. Then, the total number of oracle queries N_t after t iterations of UPD is bounded as:

$$N_t \le 2(t+1) + 1 + \log_2(\sigma_{-1}) + \inf_{\nu\in[0,1]}\left\{\frac{1-\nu}{1+\nu}\log_2\left(\frac{1-\nu}{1+\nu}\frac{1}{\varepsilon}\right) + \frac{2}{1+\nu}\log_2(M_\nu)\right\}. \tag{3.24}$$

As a consequence, $N_t \approx 2t$ in the long run (as $t \to \infty$), *i.e.*, UPD requires approximately 2 oracle queries per iteration on the average.

3.2.4 Application to the SDP template

We can use UPD for solving instances from the SDP template (1.4), by setting

$$f(\boldsymbol{X}) = \langle \boldsymbol{C}, \boldsymbol{X}\rangle \quad \text{and} \quad \mathcal{X} = \{\boldsymbol{X} : \boldsymbol{X} \in \mathbb{S}_+^n, \operatorname{Tr}\boldsymbol{X} = \alpha\} \quad \text{and} \quad \mathcal{K} = \{\boldsymbol{b}\}. \tag{3.25}$$

Then, we can formulate the sharp operator as

$$\boldsymbol{H}_t = \operatorname{lmo}_{\mathcal{X}}(\mathcal{A}^*\boldsymbol{y}_t) \in \arg\min_{\boldsymbol{X}\in\mathcal{X}} \langle \boldsymbol{C} + \mathcal{A}^*\boldsymbol{y}_t, \boldsymbol{X}\rangle. \tag{3.26}$$

This is the same form as an lmo of CGM. Consequently, we can implement it using

$$\boldsymbol{H}_t = \alpha\, \boldsymbol{u}_t\boldsymbol{u}_t^* \quad \text{where} \quad (\boldsymbol{u}_t, \lambda_t) = \texttt{MinEigVec}(\boldsymbol{C} + \mathcal{A}^*\boldsymbol{y}_t). \tag{3.27}$$

We can form the partial dual objective and the subgradient as

$$\nabla g_x(\boldsymbol{y}_t) = -\mathcal{A}\boldsymbol{H}_t \quad \text{where} \quad g_x(\boldsymbol{y}_t) = -\alpha\lambda_t. \tag{3.28}$$

Since $\operatorname{proj}_{\mathcal{K}}(\cdot) = \boldsymbol{b}$ independently of its input, we can cast the dual update rule (3.15) as

$$\hat{\boldsymbol{y}} = \boldsymbol{y}_t + \hat{\sigma}(\mathcal{A}\boldsymbol{H}_t - \boldsymbol{b}) \quad \text{where} \quad \hat{\sigma} = 2^j\sigma_{t-1}. \tag{3.29}$$

Algorithm 3.1 shows the complete pseudocode of UPD for this template.

Remark. After the first iteration, we can reuse $(\hat{\boldsymbol{u}}, \hat{\lambda})$ computed in the line-search procedure of the previous iteration to implement (3.27), in order to avoid unnecessary repetitive calls of the sharp operator. As a consequence, UPD requires $j_t + 2$ queries of the sharp operator at iterations t, also counting $j = -1$ and 0.

Algorithm 3.1 UPD for the standard SDP formulation (1.4)

Require: Data for (1.4); target accuracy $\varepsilon > 0$; initial dual variable $\boldsymbol{y} \in \mathbb{R}^d$

function UPD
  $\boldsymbol{X} \leftarrow \boldsymbol{0}$ and $S \leftarrow 0$
  for $t \leftarrow 0, 1, 2, 3, \ldots$ **do**
    $(\boldsymbol{u}, \lambda) \leftarrow \texttt{MinEigVec}(\boldsymbol{C} + \mathcal{A}^* \boldsymbol{y})$
    $\boldsymbol{H} \leftarrow \alpha \boldsymbol{u}\boldsymbol{u}^*$ and $g_x \leftarrow -\alpha\lambda$
    for $j \leftarrow -1, 0, 1, \ldots$ **do**
      $\hat{\sigma} \leftarrow 2^{-j}\sigma$
      $\hat{\boldsymbol{y}} = \boldsymbol{y} + \hat{\sigma}(\mathcal{A}\boldsymbol{H} - \boldsymbol{b})$
      $(\hat{\boldsymbol{u}}, \hat{\lambda}) \leftarrow \texttt{MinEigVec}(\boldsymbol{C} + \mathcal{A}^* \hat{\boldsymbol{y}})$
      $\hat{g}_x \leftarrow -\alpha\hat{\lambda}$
      if $\hat{g}_x \le g_x + \langle \nabla g_x(\boldsymbol{y}_t), \hat{\boldsymbol{y}} - \boldsymbol{y}_t \rangle + \frac{1}{2\hat{\sigma}} \|\hat{\boldsymbol{y}} - \boldsymbol{y}_t\|^2 + \frac{\varepsilon}{2}$ **then break for**
      end if
    end for
    $\boldsymbol{y} \leftarrow \hat{\boldsymbol{y}}$ and $\sigma \leftarrow \hat{\sigma}$
    $S \leftarrow S + \sigma$
    $\eta \leftarrow \sigma / S$
    $\boldsymbol{X} \leftarrow (1 - \eta)\boldsymbol{X} + \eta \boldsymbol{H}$
  end for
  return $\boldsymbol{X}$
end function

3.3 Accelerated Universal Primal-Dual Method

We now develop AUPD, an accelerated variant of UPD. For the accelerated scheme, we study an alternative to the fast universal gradient method of Nesterov [Nes15]. We design our accelerated variant based on FISTA scheme [BT09], in order to reduce the number of queries to the $\mathrm{proj}_{\mathcal{K}}$ oracle per iteration.

3.3.1 AUPD Iteration

Set the desired accuracy $\varepsilon > 0$, and an initial step-size σ_{-1}. Initialize the primal and the dual variables, as well as the algorithm parameters:

$$\boldsymbol{x}_0 \in \mathbb{R}^p, \qquad \boldsymbol{y}_0 = \boldsymbol{z}_0 \in \mathbb{R}^d, \qquad \tau_0 = 1, \quad \text{and} \quad S_{-1} = 0. \tag{3.30}$$

At each iteration $t = 0, 1, 2, \ldots$, compute a (sub)gradient using the sharp operator:

$$\nabla g_x(\boldsymbol{y}_t) = -\mathcal{A}\boldsymbol{h}_t \quad \text{where} \quad \boldsymbol{h}_t = \mathrm{sharp}_{f,\mathcal{X}}(\mathcal{A}^*\boldsymbol{y}_t). \tag{3.31}$$

Then, compute the partial dual objective $g_x(\boldsymbol{y}_t)$ using (3.14).

AUPD executes a line-search procedure to find an appropriate dual step-size as follows: For $j = 0, 1, 2, \ldots$, compute a trial point

$$\begin{aligned} \hat{\boldsymbol{z}} &= \mathrm{prox}_{\hat{\sigma} g_v}(\boldsymbol{y}_t - \hat{\sigma}\nabla g_x(\boldsymbol{y}_t)) \\ &= \boldsymbol{y}_t - \hat{\sigma}\nabla g_x(\boldsymbol{y}_t) - \hat{\sigma}\,\mathrm{proj}_{\mathcal{K}}(\hat{\sigma}^{-1}(\boldsymbol{y}_t - \hat{\sigma}\nabla g_x(\boldsymbol{y}_t))) \quad \text{where} \quad \hat{\sigma} = 2^{-j}\sigma_t \end{aligned} \tag{3.32}$$

until the following line-search condition holds:

$$g_x(\hat{\boldsymbol{z}}) \le g_x(\boldsymbol{y}_t) + \langle \nabla g_x(\boldsymbol{y}_t), \hat{\boldsymbol{z}} - \boldsymbol{y}_t \rangle + \frac{1}{2\hat{\sigma}}\|\hat{\boldsymbol{z}} - \boldsymbol{y}_t\|^2 + \frac{\varepsilon}{2\tau_t}. \tag{3.33}$$

Once the condition is satisfied, set $\boldsymbol{z}_{t+1} = \hat{\boldsymbol{z}}$ and $\sigma_t = \hat{\sigma}$.

Update the accumulated sum S_t of old step-sizes, then compute the primal step-size η_t as

$$\eta_t = \tau_t\sigma_t / S_t \quad \text{where} \quad S_t = S_{t-1} + \tau_t\sigma_t. \tag{3.34}$$

Update the acceleration parameter τ_t, and perform the acceleration step

$$\boldsymbol{y}_{t+1} = \boldsymbol{z}_{t+1} + \frac{\tau_t - 1}{\tau_{t+1}}(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t) \quad \text{where} \quad \tau_{t+1} = \frac{1}{2}(1 + \sqrt{1 + 4\tau_t^2}). \tag{3.35}$$

Finally, update the primal-variable

$$\boldsymbol{x}_{t+1} = (1 - \eta_t)\boldsymbol{x}_t + \eta_t\boldsymbol{h}_t. \tag{3.36}$$

Remark. We use a decaying $\frac{\varepsilon}{2\tau_t}$ slackness term in the line-search condition (3.33) of AUPD in order to prevent error accumulation, in contrast to the constant $\frac{\varepsilon}{2}$ slackness in UPD.

3.3.2 Guarantees

Theorem 3.2. *Instate the assumptions of Theorem 3.1. Then, the primal sequence* $(\boldsymbol{x}_t : t = 0, 1, \dots)$ *generated by AUPD satisfies*

$$f(\boldsymbol{x}_t) - f_\star \geq -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \tag{3.37}$$

$$f(\boldsymbol{x}_t) - f_\star \leq \frac{4\hat{M}_\varepsilon \|\boldsymbol{y}_0\|^2}{(t+2)^{\frac{1+3\nu}{1+\nu}}} + \frac{\varepsilon}{2}, \tag{3.38}$$

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \leq \frac{16\hat{M}_\varepsilon}{(t+2)^{\frac{1+3\nu}{1+\nu}}} \|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \sqrt{\frac{8\varepsilon\hat{M}_\varepsilon}{(t+2)^{\frac{1+3\nu}{1+\nu}}}} \tag{3.39}$$

where $\boldsymbol{y}_\star$ *denotes any solution to the dual problem, and* $\hat{M}_\varepsilon$ *is defined as in* (3.22).

Corollary 3.3 (Iteration complexity)**.** Instate the assumptions of Theorem 3.1. Then, AUPD provably gets an ε-solution of (1.1) after at most t_ε iterations, where

$$t_\varepsilon = \mathcal{O}\left(\left(\frac{M_\nu}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right). \tag{3.40}$$

This is the optimal iteration complexity for any smoothness order $\nu \in [0, 1]$, in the sense of first-order black box models [NY83], with respect to the smoothness of the dual problem.

Corollary 3.4 (Oracle complexity)**.** Instate the assumptions of Theorem 3.1. Then, the total number of oracle queries N_t after t iterations of AUPD is bounded as:

$$N_t \leq 2(t+1) + 1 + \log_2(\sigma_{-1}) + \inf_{\nu \in [0,1]} \left\{ \frac{1-\nu}{1+\nu} \log_2\left(\frac{1-\nu}{1+\nu}\frac{t+1}{\varepsilon}\right) + \frac{2}{1+\nu}\log_2(M_\nu) \right\}. \tag{3.41}$$

Roughly speaking, AUPD requires approximately 2 oracle queries per iteration on average on the long-run.

3.3.3 Application to the SDP template

Algorithm 3.2 shows the complete pseudocode of AUPD for the standard SDP formulation (1.4). We omit the details for brevity, which are similar to the details of UPD described in Section 3.2.4.

Remark. Unlike UPD, we cannot reuse the output of the sharp operator on the step 5 of Algorithm 3.2, because the input of `MinEigVec` are different in steps 5 and 10. As a result, AUPD requires $j_t + 2$ queries of the sharp operator at iterations t. That is, one for the step 6, and $j_t + 1$ in the line-search procedure also counting $j = 0$.

Algorithm 3.2 AUPD for the standard SDP formulation (1.4)

Require: Data for (1.4); target accuracy $\varepsilon > 0$; initial dual variable $\boldsymbol{y} \in \mathbb{R}^d$

function AUPD
  $\boldsymbol{X} \leftarrow \boldsymbol{0}$ and $\boldsymbol{z} \leftarrow \boldsymbol{y}$
  $\tau \leftarrow 1$ and $S \leftarrow 0$
  for $t \leftarrow 0, 1, 2, 3, \ldots$ **do**
    $(\boldsymbol{u}, \lambda) \leftarrow \texttt{MinEigVec}(\boldsymbol{C} + \mathcal{A}^* \boldsymbol{y})$
    $\boldsymbol{H} \leftarrow \alpha \boldsymbol{u}\boldsymbol{u}^*$ and $g_x \leftarrow -\alpha\lambda$
    for $j \leftarrow 0, 1, \ldots$ **do**
      $\hat{\sigma} \leftarrow 2^{-j}\sigma$
      $\hat{\boldsymbol{z}} = \boldsymbol{y} + \hat{\sigma}(\mathcal{A}\boldsymbol{H} - \boldsymbol{b})$
      $(\hat{\boldsymbol{u}}, \hat{\lambda}) \leftarrow \texttt{MinEigVec}(\boldsymbol{C} + \mathcal{A}^* \hat{\boldsymbol{z}})$
      $\hat{g}_x \leftarrow -\alpha\hat{\lambda}$
      if $\hat{g}_x \le g_x + \langle \nabla g_x(\boldsymbol{y}_t), \hat{\boldsymbol{z}} - \boldsymbol{y}_t \rangle + \frac{1}{2\hat{\sigma}} \|\hat{\boldsymbol{z}} - \boldsymbol{y}_t\|^2 + \frac{\varepsilon}{2\tau}$ **then break for**
      end if
    end for
    $\sigma \leftarrow \hat{\sigma}$
    $\tau^+ \leftarrow 0.5(1 + \sqrt{1 + 4\tau^2})$
    $\boldsymbol{y} \leftarrow \hat{\boldsymbol{z}} + \frac{\tau - 1}{\tau^+}(\hat{\boldsymbol{z}} - \boldsymbol{z})$
    $\boldsymbol{z} \leftarrow \hat{\boldsymbol{z}}$
    $S \leftarrow S + \tau\sigma$
    $\eta \leftarrow \tau\sigma / S$
    $\boldsymbol{X} \leftarrow (1 - \eta)\boldsymbol{X} + \eta\boldsymbol{H}$
    $\tau \leftarrow \tau^+$
  end for
  return $\boldsymbol{X}$
end function

3.4 Numerical Experiments

In this section, we illustrate the scalability and flexibility of our primal-dual framework, by providing numerical experiments in quantum tomography and matrix completion problems.

3.4.1 Quantum tomography with Pauli operators

We consider the quantum tomography problem which aims to extract information from a physical quantum system. A q-qubit quantum system is mathematically characterized by its density matrix, which is a complex $n \times n$ positive semidefinite Hermitian matrix $\boldsymbol{X}_\natural$, where $n = 2^q$.

We can provably deduce the quantum state by performing compressive linear measurements based on the Pauli operators [GLF+10]: $\boldsymbol{b} = \mathcal{A}\boldsymbol{X} \in \mathbb{C}^d$. While the size of the density matrix grows exponentially in q, a significantly fewer compressive measurements, in the order of

$d = \mathcal{O}(n \log n)$, suffices to recover a pure state q-qubit density matrix by solving the following convex optimization problem:

$$\underset{\boldsymbol{X} \in \mathbb{C}^{n \times n}}{\text{minimize}} \ \frac{1}{2}\|\mathcal{A}\boldsymbol{X} - \boldsymbol{b}\|^2 \quad \text{subject to} \quad \boldsymbol{X} \in \mathbb{S}_+^n, \quad \operatorname{Tr} \boldsymbol{X} = 1 \tag{3.42}$$

Since the objective function is smooth, we can also apply CGM to this problem. Since the formulation is tuning-free, this problem provides an ideal scalability test for comparing our framework against CGM. We remain within the noiseless setting, so that we can verify the performance of the algorithms with respect to the ground-truth in large-scale. The behavior of the algorithms are similar under polarization and additive Gaussian noise.

Here are the details of our test setup. We generate a random pure quantum state (a rank-one $\boldsymbol{X}_\natural = \boldsymbol{x}_\natural \boldsymbol{x}_\natural^*$), and we take $d = 2n \log n$ random Pauli measurements. For $q = 14$ qubits system, this corresponds to a $268'435'456$ dimensional problem with $n = 317'983$ measurements.

To apply our algorithms, we recast (3.42) into (1.1) by introducing a slack variable $\boldsymbol{u} = \mathcal{A}\boldsymbol{X} - \boldsymbol{b}$:

$$\underset{\boldsymbol{u},\, \boldsymbol{X}}{\text{minimize}} \ \frac{1}{2}\|\boldsymbol{u}\|^2 \quad \text{subject to} \quad \mathcal{A}\boldsymbol{X} - \boldsymbol{u} = \boldsymbol{b}, \quad \boldsymbol{X} \in \mathbb{S}_+^n, \quad \operatorname{Tr} \boldsymbol{X} = 1 \tag{3.43}$$

We set $\varepsilon = 2 \times 10^{-4}$ for our methods and we set a wall-time of 2×10^4 seconds for all methods. We performed the experiments in MATLAB, using a computational resource of 16 CPUs of 2.40 GHz and 512 GB memory space.

Computing the sharp operator requires one query of `MinEigVec`, that we implement by using the MATLAB's built-in `eigs` function.

Figure 3.1 illustrates iteration and time complexities. UPD, with an average of 1.978 line-search steps per iteration, exhibits a similar empirical performance to CGM with standard step-size. AUPD, with an average of 1.057 line-search steps, exhibits a superior performance.

The line-search variant of CGM improves over the standard approach. We also implement a similar line-search strategy in the primal-step for our methods. In this heuristic, we choose η_t greedily such that it minimizes the objective function. We observe a significant improvement in the practical performance of our methods with this line-search variant. Note however, the improvement due to line-search is typically less significant in more realistic noisy problem setups, both for CGM and our methods.

3.4.2 Matrix completion with MovieLens dataset

To demonstrate the flexibility of our algorithms, we consider the popular matrix completion problem. In matrix completion, we seek to estimate a low-rank matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ from its subsampled entries $\boldsymbol{b} \in \mathbb{R}^d$, where $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^d$ samples d entries from the matrix that it is applied to.

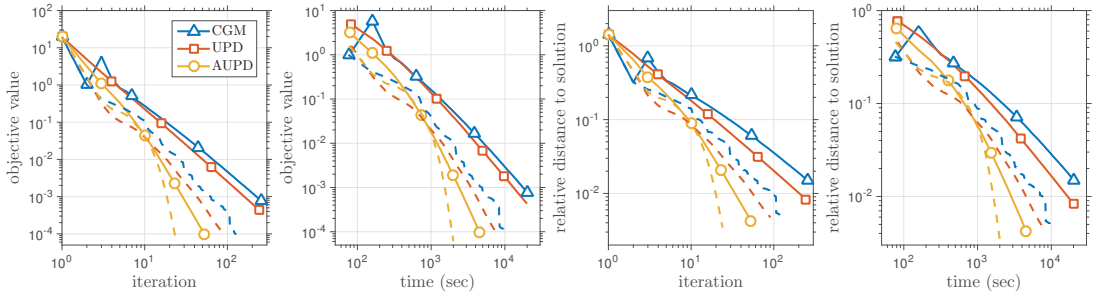


Figure 3.1 – The convergence behavior of algorithms for the $q = 14$ qubits quantum tomography problem. [Solid lines] correspond to the theoretical averaging scheme. [Dashed lines] correspond to the line-search variants (in the primal step).

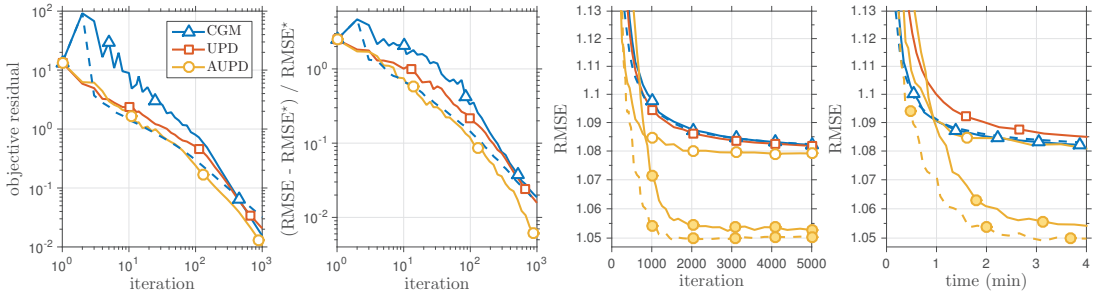


Figure 3.2 – The performance of the algorithms for the matrix completion problems. [Dashed lines] correspond to the line-search variants (in primal steps). [Empty] and the [Filled] markers correspond to the formulation (3.44) and (3.45) respectively.

Convex formulations involving the nuclear norm have been shown to be effective in estimating low-rank matrices from limited number of measurements [CR12]. For instance, we can solve the following least-squares formulation with nuclear norm constraint:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \ \frac{1}{2}\|\mathcal{A}\boldsymbol{X}-\boldsymbol{b}\|^2 \quad \text{subject to} \quad \|\boldsymbol{X}\|_{S_1} \leq \alpha. \tag{3.44}$$

We can solve (3.44) with CGM. In this formulation, α is a tuning parameter.

Unlike CGM, we can apply our algorithms to the following parameter-free formulation:

$$\underset{\boldsymbol{X}\in\mathbb{R}^{m\times n}}{\text{minimize}} \ \frac{1}{d}\|\boldsymbol{X}\|_{S_1}^2 \quad \text{subject to} \quad \mathcal{A}\boldsymbol{X}=\boldsymbol{b} \tag{3.45}$$

While the nonsmooth objective of (3.45) lets us drop the tuning parameter, it clearly burdens the computational efficiency of the convex optimization algorithms.

We solve the matrix completion problem with MovieLens 100K dataset. We use our algorithms for solving (3.44) and (3.45). In comparison, we use CGM for solving (3.44). Recall that CGM does not apply to (3.45).

For this experiment, we do not perform any pre-processing, and we use the default `ub` test and training data partition. We set the target accuracy $\varepsilon = 10^{-3}$, and we choose the tuning parameter $\alpha = 9975/2$ as in [JS10].

The sharp operator requires the computation of the top singular vectors. We use `lansvd` function from PROPACK[1] to implement the sharp operator.

The first two plots in Figure 3.2 draw the performance of the algorithms for solving (3.44). Our metrics are the normalized objective residual and the root mean squared error (test RMSE). Other two plots in Figure 3.2 compare the performance of the formulations (3.44) and (3.45) which are represented by empty and filled markers respectively. The dashed line for AUPD corresponds to the line-search heuristic, where we use greedy η_t that minimizes the feasibility gap (which is equivalent to the training RMSE).

3.5 CGM is Dual Averaging Subgradient Method

This section reviews a simple relation between CGM and the dual averaging subgradient method of Nesterov [Nes09]. We presented this relation in the appendix of [YTDC15b], which is an initial version of [YTDC15a]. For a more detailed study of the duality between the subgradient methods and CGM, we refer to [Bac15].

Consider the CGM template

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{x} \in \mathcal{X}, \tag{3.46}$$

with the smooth and convex objective function $f : \mathbb{R}^p \to \mathbb{R}$, and convex a compact domain $\mathcal{X}$.

Let us reformulate this problem by introducing a slack variable $\boldsymbol{u} = \boldsymbol{x}$:

$$\underset{\boldsymbol{u},\boldsymbol{x}}{\text{minimize}} \quad f(\boldsymbol{u}) \quad \text{subject to} \quad \boldsymbol{u} = \boldsymbol{x}, \quad \boldsymbol{x} \in \mathcal{X}. \tag{3.47}$$

The dual averaging subgradient method is perhaps one of the most efficient variants in the class of subgradient methods for solving (3.47). In order to apply this method, we first derive the dual function:

$$\begin{aligned} \phi(\boldsymbol{y}) &:= \min_{\boldsymbol{u} \in \mathbb{R}^p} \left\{ f(\boldsymbol{u}) - \langle \boldsymbol{u}, \boldsymbol{y} \rangle \right\} + \min_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{y}, \boldsymbol{x} \rangle \\ &= -f^*(\boldsymbol{y}) + \min_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{y}, \boldsymbol{x} \rangle \end{aligned} \tag{3.48}$$

where f^* is the Fenchel conjugate of f.

[1] R.M.Larsen. PROPACK Software for large and sparse SVD calculations. Available from "http://sun.stanford.edu/~rmunk/PROPACK/"

We can characterize the dual averaging subgradient method for solving the formulation (3.48) with the following two-step iterative update scheme:

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \eta_t \nabla\phi(\boldsymbol{y}_t) \\ \boldsymbol{y}_{t+1} &= \arg\max_{\boldsymbol{y}} \left\{ \langle \boldsymbol{y}, \boldsymbol{x}_{t+1} \rangle - \beta_t \psi(\boldsymbol{y}) \right\} \end{aligned} \tag{3.49}$$

Here, β_t is a non-increasing sequence of positive numbers, and $\psi : \mathbb{R}^p \to \mathbb{R}$ is a strongly convex function (also called as proximity function) to be chosen.

Let us choose $\beta_t = 1$ for all t, and $\psi = f^*$. Recall that f is smooth, hence f^* is strongly convex due to the Baillon-Haddad theorem [BC11].

Then, from the identity of the second step, $\boldsymbol{y}_{t+1} = \arg\max_{\boldsymbol{y}} \left\{ \langle \boldsymbol{y}, \boldsymbol{x}_{t+1} \rangle - \beta_t f^*(\boldsymbol{y}) \right\}$, we have

$$\boldsymbol{x}_{t+1} \in \partial f^*(\boldsymbol{y}_{t+1}), \quad \text{or equivalently,} \quad \boldsymbol{y}_{t+1} = \nabla f(\boldsymbol{x}_{t+1}). \tag{3.50}$$

This equivalence follows from the standard properties of the Fenchel duality.

$\nabla\phi(\boldsymbol{y}_t)$ in (3.49) denotes any subgradient of ϕ at $\boldsymbol{y}_t$. We can compute a subgradient by

$$\nabla\phi(\boldsymbol{y}_t) = \boldsymbol{h}_t - \boldsymbol{u}_t, \quad \text{where} \quad \boldsymbol{h}_t := \mathrm{lmo}_{\mathcal{X}}(\boldsymbol{y}_t), \quad \forall \boldsymbol{u}_t \in \partial f^*(\boldsymbol{y}_t). \tag{3.51}$$

In particular, we can choose $\boldsymbol{u}_t := \boldsymbol{x}_t$, since $\boldsymbol{x}_t \in \partial f^*(\boldsymbol{y}_t)$, as shown in (3.50).

Considering these choices, we can formulate an instance of (3.52) as follows:

$$\begin{aligned} \boldsymbol{x}_{t+1} &= \boldsymbol{x}_t + \eta_t(\boldsymbol{h}_t - \boldsymbol{x}_t) \quad \text{where} \quad \boldsymbol{h}_t = \mathrm{lmo}_{\mathcal{X}}(\boldsymbol{y}_{t+1}) \\ \boldsymbol{y}_{t+1} &= \nabla f(\boldsymbol{x}_{t+1}) \end{aligned} \tag{3.52}$$

This is nothing but a simple rearrangement of the standard CGM for solving (3.46).

3.6 Appendix: Proofs

The following Lemma that we borrow from [Nes15] provides the key properties for constructing universal gradient algorithms. We refer to [Nes15] for the proof of this lemma.

Lemma 3.3 (Lemma 2 of [Nes15])**.** Let g_x be Hölder smooth of order ν for some $\nu \in [0,1]$. Then, for any $\delta > 0$ and

$$M \geq \hat{M}_\delta := \left[\frac{1-\nu}{1+\nu}\frac{1}{\delta}\right]^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}}, \tag{3.53}$$

the following statement holds:

$$g_x(\boldsymbol{z}) \leq \underbrace{g_x(\boldsymbol{y}) + \langle \nabla g_x(\boldsymbol{y}), \boldsymbol{z}-\boldsymbol{y}\rangle + \frac{M}{2}\|\boldsymbol{z}-\boldsymbol{y}\|^2}_{Q_M(\boldsymbol{z},\boldsymbol{y})} + \frac{\delta}{2}, \qquad \forall \boldsymbol{y},\boldsymbol{z} \in \mathbb{R}^d. \tag{3.54}$$

This lemma introduces an approximate quadratic upper bound for g_x with the additional slackness term. The bound depends on the choice of the slackness parameter δ and the smoothness parameter ν. If we know in advance that $\nu = 1$, then we can set M to the smoothness constant, and we can drop the slackness term $\delta/2$.

UPD and AUPD are based on the proximal (sub)gradient step (3.15), which comes with the following estimation guarantee.

Lemma 3.4. Suppose that $\boldsymbol{y}^+$, which is defined by

$$\boldsymbol{y}^+ := \operatorname{prox}_{\sigma g_\nu}(\boldsymbol{y} - \sigma \nabla g_x(\boldsymbol{y})) \tag{3.55}$$

for some $\sigma > 0$ satisfies

$$g_x(\boldsymbol{y}^+) \leq g_x(\boldsymbol{y}) + \langle \nabla g_x(\boldsymbol{y}), \boldsymbol{y}^+ - \boldsymbol{y}\rangle + \frac{1}{2\sigma}\|\boldsymbol{y}^+ - \boldsymbol{y}\|^2 + \frac{\delta}{2} \quad \text{for some} \quad \delta \in \mathbb{R}. \tag{3.56}$$

Then, the following inequality holds $\forall \boldsymbol{z} \in \mathbb{R}^d$:

$$g(\boldsymbol{y}^+) \leq g_x(\boldsymbol{y}) + \langle \nabla g_x(\boldsymbol{y}), \boldsymbol{z}-\boldsymbol{y}\rangle + g_\nu(\boldsymbol{z}) + \frac{1}{2\sigma}\left(\|\boldsymbol{z}-\boldsymbol{y}\|^2 - \|\boldsymbol{z}-\boldsymbol{y}^+\|^2\right) + \frac{\delta}{2}. \tag{3.57}$$

Proof. Start with the optimality condition of (3.55)

$$0 \in \nabla g_x(\boldsymbol{y}) + \frac{1}{\sigma}(\boldsymbol{y}^+ - \boldsymbol{y}) + \partial g_\nu(\boldsymbol{y}^+) \implies \boldsymbol{y} - \boldsymbol{y}^+ = \sigma(\nabla g_x(\boldsymbol{y}) + \nabla g_\nu(\boldsymbol{y}^+)), \tag{3.58}$$

where we denote a subgradient of g_ν at $\boldsymbol{y}^+$ by $\nabla g_\nu(\boldsymbol{y}^+) \in \partial g_\nu(\boldsymbol{y}^+)$.

The rest of the proof is as follows:

$$\begin{aligned}
&\frac{1}{2\sigma}\left(\|\boldsymbol{z}-\boldsymbol{y}^+\|^2-\|\boldsymbol{z}-\boldsymbol{y}\|^2\right)\\
&\quad=\frac{1}{\sigma}\langle \boldsymbol{y}-\boldsymbol{y}^+,\boldsymbol{z}-\boldsymbol{y}^+\rangle-\frac{1}{2\sigma}\|\boldsymbol{y}-\boldsymbol{y}^+\|^2 &(3.59)\\
&\quad=\langle \nabla g_x(\boldsymbol{y})+\nabla g_v(\boldsymbol{y}^+),\boldsymbol{z}-\boldsymbol{y}^+\rangle-\frac{1}{2\sigma}\|\boldsymbol{y}-\boldsymbol{y}^+\|^2 &(3.60)\\
&\quad=-\langle \nabla g_x(\boldsymbol{y}),\boldsymbol{y}^+-\boldsymbol{y}\rangle+\frac{1}{2\sigma}\|\boldsymbol{y}^+-\boldsymbol{y}\|^2+\langle \nabla g_v(\boldsymbol{y}^+),\boldsymbol{z}-\boldsymbol{y}^+\rangle+\langle \nabla g_x(\boldsymbol{y}),\boldsymbol{z}-\boldsymbol{y}\rangle &(3.61)\\
&\quad\le g_x(\boldsymbol{y})-g_x(\boldsymbol{y}^+)+\frac{\delta}{2}+\langle \nabla g_v(\boldsymbol{y}^+),\boldsymbol{z}-\boldsymbol{y}^+\rangle+\langle \nabla g_x(\boldsymbol{y}),\boldsymbol{z}-\boldsymbol{y}\rangle &(3.62)\\
&\quad\le g_x(\boldsymbol{y})-g_x(\boldsymbol{y}^+)+\frac{\delta}{2}+g_v(\boldsymbol{z})-g_v(\boldsymbol{y}^+)+\langle \nabla g_x(\boldsymbol{y}),\boldsymbol{z}-\boldsymbol{y}\rangle &(3.63)\\
&\quad=g_x(\boldsymbol{y})+\langle \nabla g_x(\boldsymbol{y}),\boldsymbol{z}-\boldsymbol{y}\rangle+g_v(\boldsymbol{z})+\frac{\delta}{2}-g(\boldsymbol{y}^+) &(3.64)
\end{aligned}$$

(3.60) follows from (3.58), (3.62) uses the line-search condition (3.56), and (3.63) is based on the convexity of g_v. □

Clearly, (3.56) holds if $\sigma \le 1/\hat{M}_\delta$ due to Lemma 3.3, where $\hat{M}_\delta$ is defined by (3.53). If ν and M_ν are known, we can directly set $\sigma = 1/\hat{M}_\delta$, as it provably satisfies the line-search condition (3.56). In general, however, we do not know ν and M_ν a priori. In this case, we determine the step-size σ by using a line-search procedure.

The following lemma guarantees that the line-search terminates after a finite number of trials.

Lemma 3.5. The line-search procedure in UPD terminates after at most

$$j_t = \left\lfloor \log_2(\hat{M}_\varepsilon \sigma_{t-1}) \right\rfloor + 1 \text{ iterations.} \tag{3.65}$$

Similarly, the line-search procedure in AUPD terminates after at most

$$j_t = \left\lfloor \log_2\left(\frac{t+1}{\varepsilon}\right) + \log_2\left(M_\nu^{\frac{2}{1+\nu}}\sigma_{t-1}\right) \right\rfloor + 1 \text{ iterations.} \tag{3.66}$$

Proof. First recall that M_ν is finite due to the the Hölder smoothness assumption.

In UPD, we choose the slackness parameter $\delta = \varepsilon$, where ε is the target accuracy. For a given $\varepsilon > 0$, $\hat{M}_\varepsilon$ is also finite. Moreover, our line-search condition is provably satisfied when $\sigma \le 1/\hat{M}_\varepsilon$. Since $\hat{\sigma} = 2^{-j}\sigma_{t-1}$, the line-search procedure is terminated after at most $j = \lfloor \log_2(\hat{M}_\varepsilon \sigma_{t-1}) \rfloor + 1$ trials.

In AUPD, we choose the slackness parameter $\delta = \varepsilon/\tau_t$. Remark that

$$\tau_{t+1} = \frac{1}{2}\left(1+\sqrt{1+4\tau_t^2}\right) \le \frac{1}{2}\left(1+(1+2\tau_t)\right) = \tau_t + 1, \qquad \text{and} \qquad \tau_0 = 1. \tag{3.67}$$

Hence, by induction, we can show $\tau_t \le t+1$. Using this bound, we get

$$\hat{M}_{\varepsilon/\tau_t} = \left[\frac{1-\nu}{1+\nu}\frac{\tau_t}{\varepsilon}\right]^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}} \le \left[\frac{\tau_t}{\varepsilon}\right]^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}} \le \left[\frac{t+1}{\varepsilon}\right]^{\frac{1-\nu}{1+\nu}} M_\nu^{\frac{2}{1+\nu}}. \tag{3.68}$$

The line-search condition is provably satisfied if $\sigma \le 1/\hat{M}_{\varepsilon/\tau_t}$. As a result, our line-search procedure is terminated after at most $j = \lfloor \log_2(\frac{t+1}{\varepsilon}) + \log_2(M_\nu^{\frac{2}{1+\nu}}\sigma_{t-1})\rfloor + 1$ trials. We can show this simply by combining (3.68) with the fact that $\hat{\sigma} = 2^{-j}\sigma_{t-1}$. □

Convergence analysis of UPD

We first provide the convergence guarantee of the dual function in Theorem 3.6. Then, we prove the convergence rate and the iteration complexity guarantees of UPD.

Theorem 3.6. *Let $(\boldsymbol{y}_t : t = 0,1,2,\dots)$ be a sequence generated by UPD. Then,*

$$g(\bar{\boldsymbol{y}}_t) - g(\boldsymbol{y}) \le \bar{g}_t - g(\boldsymbol{y}) \le \frac{\hat{M}_\varepsilon}{t+1}\|\boldsymbol{y}_0 - \boldsymbol{y}\|^2 + \frac{\varepsilon}{2} \qquad \forall \boldsymbol{y} \in \mathbb{R}^d, \tag{3.69}$$

where the two averaging sequences $(\bar{\boldsymbol{y}}_t : t = 0,1,2,\dots)$ and $(\bar{g}_t : t = 0,1,2,\dots)$ are defined as

$$\bar{\boldsymbol{y}}_t := \frac{1}{S_t}\sum_{i=0}^{t}\sigma_i \boldsymbol{y}_{i+1} \quad \textit{and} \quad \bar{g}_t := \frac{1}{S_t}\sum_{i=0}^{t}\sigma_i g(\boldsymbol{y}_{i+1}). \tag{3.70}$$

In particular, by choosing $\boldsymbol{y} = \boldsymbol{y}_\star$ in (3.69)*, we get*

$$g(\bar{\boldsymbol{y}}_t) - g(\boldsymbol{y}_\star) \le \frac{\hat{M}_\varepsilon}{t+1}\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|^2 + \frac{\varepsilon}{2}. \tag{3.71}$$

Proof. The following inequality follows directly from Lemma 3.4:

$$g(\boldsymbol{y}_{i+1}) \le g(\boldsymbol{y}) + \frac{\varepsilon}{2} + \frac{1}{2\sigma_i}\left[\|\boldsymbol{y} - \boldsymbol{y}_i\|^2 - \|\boldsymbol{y} - \boldsymbol{y}_{i+1}\|^2\right], \qquad \forall \boldsymbol{y} \in \mathbb{R}^d. \tag{3.72}$$

Taking the weighted sum of this inequality over i, we get

$$\bar{g}_t \le g(\boldsymbol{y}) + \frac{\varepsilon}{2} + \frac{1}{2S_t}\left[\|\boldsymbol{y} - \boldsymbol{y}_0\|^2 - \|\boldsymbol{y} - \boldsymbol{y}_{t+1}\|^2\right], \qquad \forall \boldsymbol{y} \in \mathbb{R}^d. \tag{3.73}$$

Lemma 3.3 proves that the line-search condition (3.56) is satisfied if $\sigma \le 1/\hat{M}_\varepsilon$, where $\hat{M}_\varepsilon$ defined as in (3.22). Considering the algorithmic design of our line-search procedure, this ensures that $\sigma_i \ge 1/(2\hat{M}_\varepsilon)$. As a consequence, we have

$$S_t = \sum_{i=0}^{t}\sigma_i \ge \sum_{i=0}^{t}\frac{1}{2\hat{M}_\varepsilon} = \frac{t+1}{2\hat{M}_\varepsilon}. \tag{3.74}$$

Substituting (3.74) into (3.73), we obtain (3.69). Since g is convex, $g(\bar{\boldsymbol{y}}_t) \le \bar{g}_t$. □

Proof of Theorem 3.1

We use the following identities to relate the convergence in the dual problem to the primal:

$$\begin{aligned} g_x(\boldsymbol{y}_i) &= -f(\boldsymbol{h}_i) - \langle \boldsymbol{y}_i, \mathcal{A}\boldsymbol{h}_i \rangle \\ \nabla g_x(\boldsymbol{y}_i) &= -\boldsymbol{h}_i \\ g(\boldsymbol{y}_i) \geq g_\star := g(\boldsymbol{y}_\star) &= -f_\star, \qquad \text{for } i = 0, 1, 2, \ldots \end{aligned} \tag{3.75}$$

Substituting these expressions into Lemma 3.4, we get the following estimate

$$f(\boldsymbol{h}_i) - f_\star \leq -\langle \mathcal{A}\boldsymbol{h}_i, \boldsymbol{y} \rangle + g_v(\boldsymbol{y}) + \frac{\varepsilon}{2} + \frac{1}{2\sigma_i}\left(\|\boldsymbol{y} - \boldsymbol{y}_i\|^2 - \|\boldsymbol{y} - \boldsymbol{y}_{i+1}\|^2\right), \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \tag{3.76}$$

Taking the weighted sum of this inequality over i, and using the convexity of f, we get

$$f(\boldsymbol{x}_t) - f_\star \leq -\langle \mathcal{A}\boldsymbol{x}_t, \boldsymbol{y} \rangle + g_v(\boldsymbol{y}) + \frac{\varepsilon}{2} + \frac{1}{2S_t}\left(\|\boldsymbol{y} - \boldsymbol{y}_0\|^2 - \|\boldsymbol{y} - \boldsymbol{y}_{k+1}\|^2\right), \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \tag{3.77}$$

Choosing $\boldsymbol{y} = \boldsymbol{0}$ (note that $g_v(\boldsymbol{0}) = 0$), we get (3.20)

$$f(\boldsymbol{x}_t) - f_\star \leq \frac{\varepsilon}{2} + \frac{\|\boldsymbol{y}_0\|^2}{2S_t} \leq \frac{\varepsilon}{2} + \frac{\hat{M}_\varepsilon \|\boldsymbol{y}_0\|^2}{t+1}. \tag{3.78}$$

(3.19) follows from the Lagrange saddle point formulation:

$$f_\star \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y}_\star) = f(\boldsymbol{x}) + \langle \boldsymbol{y}_\star, \mathcal{A}\boldsymbol{x} - \boldsymbol{r} \rangle \leq f(\boldsymbol{x}) + \|\boldsymbol{y}_\star\| \|\mathcal{A}\boldsymbol{x} - \boldsymbol{r}\|, \quad \forall \boldsymbol{r} \in \mathcal{K} \text{ and } \forall \boldsymbol{x} \in \mathcal{X}. \tag{3.79}$$

The last inequality follows from the Cauchy-Schwarz inequality. We get (3.19) by setting $\boldsymbol{x} = \boldsymbol{x}_t$.

Next, we prove (3.38). Start from the following saddle point formulation:

$$f_\star \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y}_\star) = f(\boldsymbol{x}) + \langle \boldsymbol{y}_\star, \mathcal{A}\boldsymbol{x} - \boldsymbol{b} - \boldsymbol{r} \rangle, \quad \forall \boldsymbol{r} \in \mathcal{K} \text{ and } \forall \boldsymbol{x} \in \mathcal{X}. \tag{3.80}$$

Set $\boldsymbol{x} = \boldsymbol{x}_t$ in this inequality, and substitute it into (3.77):

$$\min_{\boldsymbol{r} \in \mathcal{K}} \left\{ \langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}, \boldsymbol{y} - \boldsymbol{y}_\star \rangle - \frac{1}{2S_t}\left[\|\boldsymbol{y} - \boldsymbol{y}_0\|^2 - \|\boldsymbol{y} - \boldsymbol{y}_{t+1}\|^2\right] \right\} \leq \frac{\varepsilon}{2}, \qquad \forall \boldsymbol{y} \in \mathbb{R}^d \tag{3.81}$$

$$\Longrightarrow \min_{\boldsymbol{r} \in \mathcal{K}} \left\{ \langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}, \boldsymbol{y} - \boldsymbol{y}_\star \rangle - \frac{1}{2S_t}\|\boldsymbol{y} - \boldsymbol{y}_0\|^2 \right\} \leq \frac{\varepsilon}{2}, \qquad \forall \boldsymbol{y} \in \mathbb{R}^d \tag{3.82}$$

$$\Longrightarrow \max_{\boldsymbol{y} \in \mathbb{R}^d} \min_{\boldsymbol{r} \in \mathcal{K}} \left\{ \langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}, \boldsymbol{y} - \boldsymbol{y}_\star \rangle - \frac{1}{2S_t}\|\boldsymbol{y} - \boldsymbol{y}_0\|^2 \right\} \leq \frac{\varepsilon}{2} \tag{3.83}$$

$$\Longleftrightarrow \min_{\boldsymbol{r} \in \mathcal{K}} \max_{\boldsymbol{y} \in \mathbb{R}^d} \left\{ \langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}, \boldsymbol{y} - \boldsymbol{y}_\star \rangle - \frac{1}{2S_t}\|\boldsymbol{y} - \boldsymbol{y}_0\|^2 \right\} \leq \frac{\varepsilon}{2} \tag{3.84}$$

$$\Longleftrightarrow \min_{\boldsymbol{r} \in \mathcal{K}} \left\{ \langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}, \boldsymbol{y}_0 - \boldsymbol{y}_\star \rangle + \frac{S_t}{2}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}\|^2 \right\} \leq \frac{\varepsilon}{2}, \tag{3.85}$$

(3.81) follows the definition of g_v, (3.82) removes a positive term on the left-hand-side, (3.83)

maximizes over the free variable $\boldsymbol{y}$, (3.84) holds due to the Sion's minimax theorem [Sio58], finally (3.85) analytically solves the inner maximization subproblem.

Denote by

$$\bar{\boldsymbol{r}} \in \arg\min_{\boldsymbol{r}\in\mathcal{K}} \left\{ \langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{r},\, \boldsymbol{y}_0 - \boldsymbol{y}_\star \rangle + \frac{S_t}{2} \|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}\|^2 \right\} \tag{3.86}$$

Then, we can rewrite (3.85) as

$$\langle \mathcal{A}\boldsymbol{x}_t - \bar{\boldsymbol{r}},\, \boldsymbol{y}_0 - \boldsymbol{y}_\star \rangle + \frac{S_t}{2} \|\mathcal{A}\boldsymbol{x}_t - \bar{\boldsymbol{r}}\|^2 \leq \frac{\varepsilon}{2}. \tag{3.87}$$

Using Cauchy-Schwarz inequality, this implies

$$-\|\mathcal{A}\boldsymbol{x}_t - \bar{\boldsymbol{r}}\| \cdot \|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \frac{S_t}{2} \|\mathcal{A}\boldsymbol{x}_t - \bar{\boldsymbol{r}}\|^2 \leq \frac{\varepsilon}{2}. \tag{3.88}$$

Solving this second order inequality for $\|\mathcal{A}\boldsymbol{x}_t - \bar{\boldsymbol{r}}\|$, we get

$$\begin{aligned} \|\mathcal{A}\boldsymbol{x}_t - \bar{\boldsymbol{r}}\| &\leq \frac{1}{S_t}\left(\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \sqrt{\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|^2 + \varepsilon S_t} \right) \\ &\leq \frac{1}{S_t}\left(2\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \sqrt{\varepsilon S_t} \right). \end{aligned} \tag{3.89}$$

By definition of the distance function,

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \leq \|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{r}\|, \quad \forall \boldsymbol{r} \in \mathcal{K}. \tag{3.90}$$

Finally by substituting $S_t \geq \frac{t+1}{2\hat{M}_\varepsilon}$, we get (3.38).

Proof of Corollary 3.1

To provably get an ε-solution, we need to guarantee both $\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \leq \varepsilon$ and $f(\boldsymbol{x}_t) - f_\star \leq \varepsilon$.

If we combine these conditions with the bounds from Theorem 3.1, we obtain the following conditions on t_ε:

$$\frac{4\hat{M}_\varepsilon}{t_\varepsilon + 1} \|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \sqrt{\frac{2\varepsilon\hat{M}_\varepsilon}{t_\varepsilon + 1}} \leq \varepsilon, \quad \text{and} \quad \frac{\hat{M}_\varepsilon \|\boldsymbol{y}_0\|^2}{t_\varepsilon + 1} + \frac{\varepsilon}{2} \leq \varepsilon. \tag{3.91}$$

For the objective residual, we can get the bound on t_ε by a simple rearrangement

$$t_\varepsilon + 1 \geq \frac{2\hat{M}_\varepsilon \|\boldsymbol{y}_0\|^2}{\varepsilon}. \tag{3.92}$$

For the feasibility gap, we need to solve a second order inequality with respect to $\frac{1}{\sqrt{t_\varepsilon+1}}$

$$\frac{1}{\sqrt{t_\varepsilon+1}} \le \frac{-\sqrt{2\varepsilon\hat{M}_\varepsilon} + \sqrt{2\varepsilon\hat{M}_\varepsilon + 16\varepsilon\hat{M}_\varepsilon\|y_0 - y_\star\|}}{8\hat{M}_\varepsilon\|y_0 - y_\star\|}. \tag{3.93}$$

Rearranging and simplifying, we get

$$t_\varepsilon + 1 \ge \frac{(8\hat{M}_\varepsilon\|y_0 - y_\star\|)^2}{4\varepsilon\hat{M}_\varepsilon + 16\varepsilon\hat{M}_\varepsilon\|y_0 - y_\star\| - \sqrt{16\varepsilon^2\hat{M}_\varepsilon^2 + 128\varepsilon^2\hat{M}_\varepsilon^2\|y_0 - y_\star\|}} \tag{3.94}$$

$$= \frac{(8\|y_0 - y_\star\|)^2}{4 + 16\|y_0 - y_\star\| - \sqrt{16 + 128\|y_0 - y_\star\|}} \frac{\hat{M}_\varepsilon}{\varepsilon} \tag{3.95}$$

Both (3.92) and (3.95) imply that $t_\varepsilon = \mathcal{O}\left(\frac{\hat{M}_\varepsilon}{\varepsilon}\right)$. Using the definition (3.22) of $\hat{M}_\varepsilon$, we can get the following order representation in terms of ε:

$$t_\varepsilon = \mathcal{O}\left(\inf_{\nu\in[0,1]} \left(\frac{1-\nu}{1+\nu}\right)^{\frac{1-\nu}{1+\nu}} \left(\frac{M_\nu}{\varepsilon}\right)^{\frac{2}{1+\nu}}\right) = \mathcal{O}\left(\inf_{\nu\in[0,1]} \left(\frac{M_\nu}{\varepsilon}\right)^{\frac{2}{1+\nu}}\right). \tag{3.96}$$

Proof of Corollary 3.2

Similar to the analysis in [Nes15], we estimate the total number of oracle quires of UPD.

The total number of oracle quires up to the iteration t is given by the formula

$$N_t = \sum_{i=0}^{t} (j_i + 2). \tag{3.97}$$

We designed the line-search procedure such that $\sigma_i = 2^{j_i}\sigma_{i-1}$. Consequently,

$$\begin{aligned} N_t &= \sum_{i=0}^{t} (\log_2(\sigma_i/\sigma_{i-1}) + 2) \\ &= \sum_{i=0}^{t} (\log_2(\sigma_i) - \log_2(\sigma_{i-1}) + 2) \\ &= 2(t+1) + \log_2(\sigma_{-1}) - \log_2(\sigma_t). \end{aligned} \tag{3.98}$$

Recall that $\sigma_t \ge 1/(2\hat{M}_\varepsilon)$, hence we get

$$\begin{aligned} N_t &\le 2(t+1) + 1 + \log_2(\sigma_{-1}) + \log_2(\hat{M}_\varepsilon) \\ &= 2(t+1) + 1 + \log_2(\sigma_{-1}) + \inf_{\nu\in[0,1]} \left\{\frac{1-\nu}{1+\nu}\log_2\left(\frac{1-\nu}{1+\nu}\frac{1}{\varepsilon}\right) + \frac{2}{1+\nu}\log_2(M_\nu)\right\}. \end{aligned} \tag{3.99}$$

The second line simply follows from the definition (3.22) of $\hat{M}_\varepsilon$.

Convergence analysis of AUPD

Consider the following dual update scheme of $\boldsymbol{z}_{t+1}$ and $\boldsymbol{w}_{t+1}$ from $\boldsymbol{y}_t$ and $\boldsymbol{w}_t$:

$$\begin{cases} \boldsymbol{y}_t & = (1-\tau_t)\boldsymbol{z}_t + \bar{\tau}_t \boldsymbol{w}_t \\ \boldsymbol{z}_{t+1} & = \mathrm{prox}_{\sigma_t g_v}\big(\boldsymbol{y}_t - \sigma_t \nabla g_x(\boldsymbol{y}_t)\big) \\ \boldsymbol{w}_{t+1} & = \boldsymbol{w}_t - \frac{1}{\bar{\tau}_t}\big(\boldsymbol{y}_t - \boldsymbol{z}_{t+1}\big), \end{cases} \tag{3.100}$$

where $\boldsymbol{w}_0 = \boldsymbol{z}_0$, $\bar{\tau}_0 = 1$ and

$$\bar{\tau}_t^2 = \bar{\tau}_{t-1}^2(1-\bar{\tau}_t). \tag{3.101}$$

The parameter σ_t is determined based on the following line-search condition:

$$g(\boldsymbol{z}_{t+1}) \le g(\boldsymbol{y}_t) + \langle \nabla g(\boldsymbol{y}_t), \boldsymbol{z}_{t+1} - \boldsymbol{y}_t\rangle + \frac{1}{2\sigma_t}\|\boldsymbol{z}_{t+1} - \boldsymbol{y}_t\|^2 + \frac{\varepsilon\bar{\tau}_t}{2}, \tag{3.102}$$

with $\sigma_t \le \sigma_{t-1}$ for $t \ge 0$.

Next, we show that (3.100) is equivalent to AUPD update scheme in the dual.

Lemma 3.7. The scheme (3.100) can be restated as follows:

$$\begin{cases} \boldsymbol{z}_{t+1} & = \mathrm{prox}_{\sigma_t g_v}(\boldsymbol{y}_t - \sigma_t \nabla g_x(\boldsymbol{y}_t)) \\ \tau_{t+1} & = \frac{1}{2}\big(1 + \sqrt{1+4\tau_t^2}\big) \\ \boldsymbol{y}_{t+1} & = \boldsymbol{z}_{t+1} + \frac{\tau_t - 1}{\tau_{t+1}}(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t), \end{cases} \tag{3.103}$$

where $\boldsymbol{y}_0 = \boldsymbol{z}_0$ and $\tau_0 = 1$, and σ_t is determined based on the line-search.

This dual scheme is in the form of FISTA [BT09], except for the line-search procedure.

Proof. Set $\tau_t = \bar{\tau}_t^{-1}$, then $t_0 = \tau_0^{-1} = 1$. From (3.100), we have

$$\boldsymbol{w}_t - \boldsymbol{w}_{t+1} = \frac{1}{\bar{\tau}_t}(\boldsymbol{y}_t - \boldsymbol{z}_{t+1}) = \tau_t(\boldsymbol{y}_t - \boldsymbol{z}_{t+1}). \tag{3.104}$$

We also have $\boldsymbol{y}_t = (1-\bar{\tau}_t)\boldsymbol{z}_t + \bar{\tau}_t\boldsymbol{w}_t$, which leads to

$$\boldsymbol{w}_t = \frac{1}{\bar{\tau}_t}(\boldsymbol{y}_t - (1-\bar{\tau}_t)\boldsymbol{z}_t) = \tau_t(\boldsymbol{y}_t - (1-\tau_t^{-1})\boldsymbol{z}_t). \tag{3.105}$$

Combining these expressions, we get

$$\begin{aligned} \tau_t(\boldsymbol{y}_t - \boldsymbol{z}_{t+1}) &= \boldsymbol{w}_t - \boldsymbol{w}_{t+1} \\ &= \tau_t(\boldsymbol{y}_t - (1-\tau_t^{-1})\boldsymbol{z}_t) - \tau_{t+1}(\boldsymbol{y}_{t+1} - (1-\tau_{t+1}^{-1})\boldsymbol{z}_{t+1}). \end{aligned} \tag{3.106}$$

Then, we simplify as follows:

$$\begin{aligned}\tau_{t+1}\boldsymbol{y}_{t+1} &= \tau_t\boldsymbol{z}_{t+1} + \tau_{t+1}(1-\tau_{t+1}^{-1})\boldsymbol{z}_{t+1} - \tau_t(1-\tau_t^{-1})\boldsymbol{z}_t \\ &= (\tau_t + \tau_{t+1} - 1)\boldsymbol{z}_{t+1} - (\tau_t - 1)\boldsymbol{z}_t.\end{aligned} \tag{3.107}$$

Hence $\boldsymbol{y}_{t+1} = \boldsymbol{z}_{t+1} + \frac{\tau_t - 1}{\tau_{t+1}}(\boldsymbol{z}_{t+1} - \boldsymbol{z}_t)$, which is the third step of (3.103).

Next, from the condition (3.101), we have $\tau_{t+1}^2 - \tau_{t+1} - \tau_t^2 = 0$. Hence, $\tau_{t+1} = \frac{1}{2}\big(1 + \sqrt{1+4\tau_t^2}\big)$, which is exactly the second step of (3.103). □

Proof of Theorem 3.2

Start from Lemma 3.4:

$$\begin{aligned} g(\boldsymbol{z}_{t+1}) &\le \big(g_x(\boldsymbol{y}_t) + \langle \nabla g_x(\boldsymbol{y}_t), \boldsymbol{y} - \boldsymbol{y}_t\rangle + g_v(\boldsymbol{z})\big) + \frac{\varepsilon}{2\tau_t} \\ &\qquad + \frac{1}{\sigma_t}\langle \boldsymbol{y}_t - \boldsymbol{z}_{t+1}, \boldsymbol{y}_t - \boldsymbol{y}\rangle - \frac{1}{2\sigma_t}\|\boldsymbol{z}_{t+1} - \boldsymbol{y}_t\|^2, \quad \forall \boldsymbol{y} \in \mathbb{R}^d \qquad (3.108) \\ &\le g(\boldsymbol{z}) + \frac{\varepsilon}{2\tau_t} + \frac{1}{\sigma_t}\langle \boldsymbol{y}_t - \boldsymbol{z}_{t+1}, \boldsymbol{y}_t - \boldsymbol{y}\rangle - \frac{1}{2\sigma_t}\|\boldsymbol{z}_{t+1} - \boldsymbol{y}_t\|^2, \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \qquad (3.109)\end{aligned}$$

Next, subtract $g_\star$ from (3.108) to get

$$\begin{aligned} g(\boldsymbol{z}_{t+1}) - g_\star &\le \big(g_x(\boldsymbol{y}_t) + \langle \nabla g_x(\boldsymbol{y}_t), \boldsymbol{y} - \boldsymbol{y}_t\rangle + g_v(\boldsymbol{y}) - g_\star\big) + \frac{\varepsilon}{2\tau_t} \\ &\qquad + \frac{1}{\sigma_t}\langle \boldsymbol{y}_t - \boldsymbol{z}_{t+1}, \boldsymbol{y}_t - \boldsymbol{y}\rangle - \frac{1}{2\sigma_t}\|\boldsymbol{z}_{t+1} - \boldsymbol{y}_t\|^2, \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \qquad (3.110)\end{aligned}$$

Set $\boldsymbol{y} = \boldsymbol{z}_t$ in (3.109), and subtract $g_\star$ from the both sides:

$$g(\boldsymbol{z}_{t+1}) - g_\star \le g(\boldsymbol{z}_t) - g_\star + \frac{\varepsilon}{2\tau_t} - \frac{1}{2\sigma_t}\|\boldsymbol{z}_{t+1} - \boldsymbol{y}_t\|^2 + \frac{1}{\sigma_t}\langle \boldsymbol{y}_t - \boldsymbol{z}_{t+1}, \boldsymbol{y}_t - \boldsymbol{z}_t\rangle. \tag{3.111}$$

Multiply (3.110) by τ_t^{-1}, (3.111) by $(1-\tau_t^{-1})$, and sum them up

$$\begin{aligned} g(\boldsymbol{z}_{t+1}) - g_\star &\le (1-\tau_t^{-1})\big(g(\boldsymbol{z}_t) - g_\star\big) + \frac{1}{2\sigma_t\tau_t^2}\big(\|\boldsymbol{w}_t - \boldsymbol{y}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{y}\|^2\big) + \frac{\varepsilon}{2\tau_t} \\ &\qquad + \frac{1}{\tau_t}\big(g_x(\boldsymbol{y}_t) + \langle \nabla g_x(\boldsymbol{y}_t), \boldsymbol{y} - \boldsymbol{y}_t\rangle + g_v(\boldsymbol{y}) - g_\star\big), \qquad \forall \boldsymbol{y} \in \mathbb{R}^d. \qquad (3.112)\end{aligned}$$

Multiply (3.112) by $\sigma_t\tau_t^2$, and sum over t to get

$$\begin{aligned} \sum_{i=0}^{t} \sigma_i\tau_i^2(g(\boldsymbol{z}_{i+1}) - g_\star) &\le \sum_{i=0}^{y}\Big[\sigma_i\tau_i^2(1-\tau_i^{-1})\big(g(\boldsymbol{z}_i) - g_\star\big) + \frac{1}{2}\big(\|\boldsymbol{w}_i - \boldsymbol{y}\|^2 - \|\boldsymbol{w}_{i+1} - \boldsymbol{y}\|^2\big) \\ &\qquad + \frac{\sigma_i\tau_i\varepsilon}{2} + \sigma_i\tau_i\big(g(\boldsymbol{y}_i) + \langle \nabla g_x(\boldsymbol{y}_i), \boldsymbol{y} - \boldsymbol{y}_i\rangle + g_v(\boldsymbol{y}) - g_\star\big)\Big], \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \qquad (3.113)\end{aligned}$$

By design of the update rules and the line-search procedure (since $\sigma_t \le \sigma_{t-1}$), we have

$$\tau_0 = 1 \quad \text{and} \quad \sigma_t \tau_t^2 (1 - \tau_t^{-1}) \le \sigma_{t-1} \tau_{t-1}^2 \quad \text{for } t = 1, 2, \ldots \tag{3.114}$$

Substitute (3.114) into (3.113) to get

$$\begin{aligned} \sum_{i=0}^{t} \sigma_i \tau_i^2 (g(\boldsymbol{z}_{i+1}) - g_\star) \le \frac{1}{2}\big(\|\boldsymbol{w}_0 - \boldsymbol{y}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{y}\|^2\big) + \sum_{i=1}^{t} \left[\sigma_{i-1} \tau_{i-1}^2 \big(g(\boldsymbol{z}_i) - g_\star\big)\right] \\ + \sum_{i=0}^{t} \left[\sigma_i \tau_i \left(g_x(\boldsymbol{y}_i) + \langle \nabla g_x(\boldsymbol{y}_i), \boldsymbol{y} - \boldsymbol{y}_i \rangle + g_v(\boldsymbol{y}) - g_\star + \frac{\varepsilon}{2}\right)\right], \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \end{aligned} \tag{3.115}$$

This implies

$$\begin{aligned} \frac{\sigma_t \tau_t^2}{S_t} \big(g(\boldsymbol{z}_{t+1}) - g_\star\big) \le \frac{1}{S_t} \sum_{i=0}^{t} \left[\sigma_t \tau_t \Big(g_x(\boldsymbol{y}_i) + \langle \nabla g_x(\boldsymbol{y}_i), \boldsymbol{y} - \boldsymbol{y}_i \rangle + g_v(\boldsymbol{y}) - g_\star\Big)\right] \\ + \frac{1}{2S_t}\left[\|\boldsymbol{w}_0 - \boldsymbol{y}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{y}\|^2\right] + \frac{\varepsilon}{2}, \quad \forall \boldsymbol{y} \in \mathbb{R}^d. \end{aligned} \tag{3.116}$$

Note that $g(\boldsymbol{z}_{t+1}) - g_\star \ge 0$, hence by rearranging we get

$$-\frac{1}{S_t} \sum_{i=0}^{t} \sigma_i \tau_i \Big(g_x(\boldsymbol{y}_i) + \langle \nabla g_x(\boldsymbol{y}_i), \boldsymbol{y} - \boldsymbol{y}_i \rangle + g_v(\boldsymbol{y}) - g_\star\Big) \le \frac{1}{2S_t}\big(\|\boldsymbol{w}_0 - \boldsymbol{y}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{y}\|^2\big) + \frac{\varepsilon}{2}. \tag{3.117}$$

Now, we use the following identities to map this estimate for the primal variable

$$\begin{aligned} g(\boldsymbol{y}_i) &= -f(\boldsymbol{h}_i) - \langle \boldsymbol{y}_i, \mathcal{A}\boldsymbol{h}_i \rangle \\ \nabla g_x(\boldsymbol{y}_i) &= -\mathcal{A}\boldsymbol{h}_i, \\ g_\star &= -f_\star. \end{aligned} \tag{3.118}$$

Then, using Jensen's inequality and convexity of f we get

$$f(\boldsymbol{x}_t) - f_\star \le -\langle \mathcal{A}\boldsymbol{x}_t, \boldsymbol{y} \rangle + g_v(\boldsymbol{y}) + \frac{\varepsilon}{2} + \frac{1}{2S_t}\big(\|\boldsymbol{w}_0 - \boldsymbol{y}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{y}\|^2\big), \quad \forall \boldsymbol{y} \in \mathbb{R}^d \tag{3.119}$$

$$\le \frac{\varepsilon}{2} + \frac{\|\boldsymbol{y}_0\|^2}{2\hat{S}_k}, \tag{3.120}$$

where (3.120) follows by setting $\boldsymbol{y} = \boldsymbol{0}$ in (3.119).

We can reformulate the update rule of τ_t in (3.103) as

$$\tau_t = \tau_t^2 - \tau_{t-1}^2, \quad \text{for } t = 1, 2, \ldots \tag{3.121}$$

Also note that

$$\frac{t+2}{2} \le \tau_t < t + 2, \quad \text{for } t = 0, 1, 2, \ldots \tag{3.122}$$

Using (3.121), (3.122), and the fact that

$$\sigma_0 \geq \sigma_i \geq \sigma_t \geq \frac{1}{2\hat{M}_{\varepsilon/\tau_t}} = \frac{1}{2\tau_t^{\frac{1-\nu}{1+\nu}}\hat{M}_\varepsilon} \geq \frac{1}{2(t+2)^{\frac{1-\nu}{1+\nu}}\hat{M}_\varepsilon}, \tag{3.123}$$

one can show

$$\begin{aligned} S_t = \sum_{i=0}^{t} \sigma_i \tau_i &\geq \sum_{i=0}^{t} \frac{\tau_i}{2\hat{M}_{\varepsilon/\tau_t}} = \frac{1}{2\hat{M}_{\varepsilon/\tau_t}}\left[1 + \sum_{i=1}^{t}\left(\tau_i^2 - \tau_{i-1}^2\right)\right] \\ &\geq \frac{\tau_t^2}{2(t+2)^{\frac{1-\nu}{1+\nu}}\hat{M}_\varepsilon} \geq \frac{(t+2)^{\frac{1+3\nu}{1+\nu}}}{8\hat{M}_\varepsilon}. \end{aligned} \tag{3.124}$$

We get (3.38) by substituting (3.124) into (3.120).

Inequality (3.37) follows the saddle point formulation (3.79) by setting $\boldsymbol{x} = \boldsymbol{x}_t$.

Finally, the proof for (3.39) follows similarly to the proof in Theorem 3.1. One can show

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \leq \frac{2\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|}{S_t} + \sqrt{\frac{\varepsilon}{S_t}}, \tag{3.125}$$

and we complete the proof by substituting (3.124) into this estimate.

Proof of Corollary 3.3

Using the bounds from Theorem 3.2, we obtain the following conditions on t_ε:

$$\frac{16\hat{M}_\varepsilon}{(t_\varepsilon+2)^{\frac{1+3\nu}{1+\nu}}}\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| + \sqrt{\frac{8\varepsilon\hat{M}_\varepsilon}{(t_\varepsilon+2)^{\frac{1+3\nu}{1+\nu}}}} \leq \varepsilon, \quad \text{and} \quad \frac{4\hat{M}_\varepsilon\|\boldsymbol{y}_0\|^2}{(t_\varepsilon+2)^{\frac{1+3\nu}{1+\nu}}} + \frac{\varepsilon}{2} \leq \varepsilon. \tag{3.126}$$

For the objective residual, we can get the bound on t_ε by a simple rearrangement

$$t_\varepsilon + 2 \geq \left(\frac{8\hat{M}_\varepsilon\|\boldsymbol{y}_0\|^2}{\varepsilon}\right)^{\frac{1+\nu}{1+3\nu}}. \tag{3.127}$$

For the feasibility gap, we solve a second order inequality with respect to $\frac{1}{\sqrt{(t_\varepsilon+2)^{\frac{1+3\nu}{1+\nu}}}}$

$$\frac{1}{\sqrt{(t_\varepsilon+2)^{\frac{1+3\nu}{1+\nu}}}} \leq \frac{-\sqrt{8\varepsilon\hat{M}_\varepsilon} + \sqrt{8\varepsilon\hat{M}_\varepsilon + 64\varepsilon\hat{M}_\varepsilon\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|}}{32\hat{M}_\varepsilon\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|}. \tag{3.128}$$

Rearranging and simplifying, we get

$$t_\varepsilon + 2 \geq \left(\frac{(2^5\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|)^2}{2^4 + 2^6\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\| - \sqrt{2^8 + 2^{11}\|\boldsymbol{y}_0 - \boldsymbol{y}_\star\|}}\frac{\hat{M}_\varepsilon}{\varepsilon}\right)^{\frac{1+\nu}{1+3\nu}} \tag{3.129}$$

Using the definition (3.22) of $\hat{M}_\varepsilon$, together with the bounds (3.127) and (3.129), we can show

$$t_\varepsilon = \mathcal{O}\left(\inf_{\nu\in[0,1]}\left(\frac{1-\nu}{1+\nu}\right)^{\frac{1-\nu}{1+3\nu}}\left(\frac{M_\nu}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right) = \mathcal{O}\left(\inf_{\nu\in[0,1]}\left(\frac{M_\nu}{\varepsilon}\right)^{\frac{2}{1+3\nu}}\right). \tag{3.130}$$

Proof of Corollary 3.4

Similar to the proof of Corollary 3.2, the total number of oracle quires up to the iteration t is

$$N_t = \sum_{i=0}^{t}(j_i+2) = 2(t+1) + \log_2(\sigma_{-1}) - \log_2(\sigma_t). \tag{3.131}$$

Recall that $\sigma_t \geq 1/(2\hat{M}_{\varepsilon/\tau_t})$. Using the definition of $\hat{M}_{\varepsilon/\tau_t}$, and (3.122), we get

$$\begin{aligned} N_t &\leq 2(t+1) + 1 + \log_2(\sigma_{-1}) + \inf_{\nu\in[0,1]}\left\{\frac{1-\nu}{1+\nu}\log_2\left(\frac{1-\nu}{1+\nu}\frac{\tau_t}{\varepsilon}\right) + \frac{2}{1+\nu}\log_2(M_\nu)\right\} \\ &\leq 2(t+1) + 1 + \log_2(\sigma_{-1}) + \inf_{\nu\in[0,1]}\left\{\frac{1-\nu}{1+\nu}\log_2\left(\frac{1-\nu}{1+\nu}\frac{t+2}{\varepsilon}\right) + \frac{2}{1+\nu}\log_2(M_\nu)\right\}. \end{aligned} \tag{3.132}$$

4 CGM for Composite Problems

CGM features significantly reduced computational costs (in comparison with the projected gradient methods) in key machine learning applications. As we discussed in Chapter 2, it also exhibits structured updates on the decision variable which are amenable for further storage benefits via sketching. Unfortunately, standard CGM cannot handle affine inclusion constraint $\mathcal{A}\boldsymbol{x} \in \mathcal{K}$ efficiently, which restricts its applicability to the key machine learning applications as well as the SDP formulations.

To this end, we describe a generalized CGM approach for a composite convex minimization template with broad applications. In particular, the proposed method applies to our model problems (1.1) and (1.4).

This chapter is based on the joint work with Francesco Locatello, Olivier Fercoq and Volkan Cevher [YFLC18].

Introduction

In this chapter, we consider the following composite convex minimization formulation:

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad F(\boldsymbol{x}) := f(\boldsymbol{x}) + g(\mathcal{A}\boldsymbol{x}), \tag{4.1}$$

Again, $f : \mathcal{X} \to \mathbb{R}$ is a smooth proper closed and convex function, and $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a proper closed convex function which is possibly non-smooth. $\mathcal{A} : \mathbb{R}^p \to \mathbb{R}^d$ is a given linear map. This template covers affine inclusion contraint $\mathcal{A}\boldsymbol{x} \in \mathcal{K}$ as a special case, since we can choose g as the indicator function of $\mathcal{K}$.

By using the powerful operator-splitting framework, many problems that belong to the template (4.1) can be solved efficiently. These methods make use of the gradient of the smooth part f, along with the prox-operator of the non-smooth part g and the projection onto the domain $\mathcal{X}$.

CGM-type algorithms (also called as projection-free methods) are desirable when the projection onto the problem domain is a computational bottleneck. A classical example is the positive semi-definite cone, for which the projection requires computing a full eigendecomposition.

Unfortunately, the standard CGM cannot handle the non-smooth term in (4.1). When the non-smooth part is an indicator function, one could take the linear minimization oracle (lmo) with respect to the intersection between $\mathcal{X}$ and the set represented by $g(\mathcal{A}\boldsymbol{x})$. Unfortunately, even the lmo itself can be a difficult optimization problem in this case.

To this end, we propose a novel CGM framework for solving the composite problem (4.1) with rigorous convergence guarantees. Our approach retains the simplicity of projection-free methods, but it allows us to disentangle the complexity of the lmo from the non-smooth component $g(\mathcal{A}\boldsymbol{x})$.

Our method combines the ideas of smoothing [Nes05] and homotopy under the CGM framework. Lan [Lan14] proposed a similar approach for non-smooth problems, which is also extended for the conditional gradient sliding framework in their follow-up papers [LZ16] and [LPZZ17]. Their analysis, however, is restricted with a Lipschitz continuity assumption on the non-smooth part. As a consequence, their methods and analyses do not apply to the problems with affine constraints. See Sections 4.3.5 and 4.3.6.

Contributions

Our main contributions in this chapter are as follows:

◦ We introduce a simple, easy to implement CGM framework for solving the instances from composite minimization template (4.1).

◦ When the non-smooth part is a Lipschitz-continuous regularizer, we prove that the proposed method, HCGM, achieves the optimal $\mathcal{O}(1/\sqrt{t})$ convergence rate. When the non-smooth term is an indicator function, *i.e.,* when the problem has affine constraints, we prove that the proposed method gets $\mathcal{O}(1/\sqrt{t})$ convergence rate, both in the objective residual and the feasibility gap.

◦ We analyze the convergence guarantees of the proposed method for the case where the linear minimization oracle is noisy, with additive and/or multiplicative errors. We show that our approach is robust.

◦ We present key instances of our framework, including the non-smooth minimization, minimization with affine inclusion constraints, and convex splitting. We provide a summary of the related work for each case.

◦ We present empirical evidence supporting our findings.

4.1 Preliminaries

Our method is based on the idea of combining smoothing and homotopy under the CGM framework. Hence, we first review the smoothing approach of Nesterov [Nes05], which plays a crucial role in our algorithmic design.

4.1.1 Nesterov Smoothing

Nesterov smoothing exploits an important class of non-smooth functions $\psi(\boldsymbol{x})$ that can be written in the following *max-form*:

$$\psi(\boldsymbol{x}) = \max_{\boldsymbol{u}\in\mathcal{U}} \left\{ \langle \mathcal{A}\boldsymbol{x}, \boldsymbol{u}\rangle - \hat{\phi}(\boldsymbol{u}) \right\}, \tag{4.2}$$

for some convex and compact set $\mathcal{U} \subset \mathbb{R}^d$ and a convex function $\hat{\phi} : \mathcal{U} \to \mathbb{R}$.

Let us consider a prox-function on $\mathcal{U}$ (*i.e.,* a strongly convex continuous function), and denote it by $\delta(\boldsymbol{u})$. In this work, we use the Euclidean prox-function $\delta(\boldsymbol{u}) = \frac{1}{2}\|\boldsymbol{u}\|^2$.

Smooth approximation $\psi_\beta(x)$ with smoothness parameter $\beta > 0$ is defined as

$$\psi_\beta(\boldsymbol{x}) = \max_{\boldsymbol{u}\in\mathcal{U}} \left\{ \langle \mathcal{A}\boldsymbol{x}, \boldsymbol{u}\rangle - \hat{\phi}(\boldsymbol{u}) - \beta\delta(\boldsymbol{u}) \right\}. \tag{4.3}$$

Then, ψ_β is well defined, differentiable, convex and smooth. Moreover, it uniformly approximates ψ, in the sense that it satisfies the following envelop property:

$$\psi_\beta(\boldsymbol{x}) \le \psi(\boldsymbol{x}) \le \psi_\beta(\boldsymbol{x}) + \beta D_{\mathcal{U}} \qquad (\forall \boldsymbol{x} \in \mathcal{X}), \tag{4.4}$$

where $D_{\mathcal{U}} = \max_{\boldsymbol{u}\in\mathcal{U}} \delta(\boldsymbol{u})$. See Theorem 1 in [Nes05] for the proof and more details.

If we assume $g(\mathcal{A}\cdot)$ is Lipschitz continuous, then we can write it in the max form by choosing $\psi(\boldsymbol{x}) = g(\mathcal{A}\boldsymbol{x})$ and $\hat{\phi}(\boldsymbol{u}) = g^*(\boldsymbol{u})$. Here, g^* is the Fenchel conjugate of g, defined as

$$g^*(\boldsymbol{u}) = \max_{\boldsymbol{z}} \left\{ \langle \boldsymbol{u}, \boldsymbol{z}\rangle - g(\boldsymbol{z}) \right\}. \tag{4.5}$$

Since g is convex and lower semicontinuous, Fenchel duality holds, and we have

$$g(\mathcal{A}\boldsymbol{x}) = g^{**}(\mathcal{A}\boldsymbol{x}). \tag{4.6}$$

Moreover, Lipschitz continuity assumption on g imposes the boundedness of the dual domain. See Lemma 5 in [DFTJ16] for a proof of this well-known result in the convex analysis.

4.1.2 Quadratic Penalty

Quadratic penalty approach is an effective proxy for handling the affine constraint $\mathcal{A}\boldsymbol{x} \in \mathcal{K}$. It works by replacing the constraint with a penalty function which favors the feasibility of the solution. We consider the squared Euclidean distance as the penalty function: $\frac{1}{2\beta}\operatorname{dist}^2(\mathcal{A}\boldsymbol{x}, \mathcal{K})$, where $\beta > 0$ is the so-called penalty parameter. Surprisingly, quadratic penalty approach is structurally equivalent to a *de facto* instance of the Nesterov smoothing.

Let us start by writing the Fenchel conjugate of the indicator function $\iota_{\mathcal{K}}(\cdot)$,

$$\iota^*_{\mathcal{K}}(\boldsymbol{z}) = \max_{\boldsymbol{v}\in\mathcal{K}} \langle \boldsymbol{v}, \boldsymbol{z}\rangle, \qquad \text{where} \qquad \iota_{\mathcal{K}}(\boldsymbol{y}) = \begin{cases} 0 & \text{if } \boldsymbol{y} \in \mathcal{K} \\ +\infty & \text{otherwise.} \end{cases} \tag{4.7}$$

Then, we can write the affine constraint in the max form by choosing $\hat{\phi}(\boldsymbol{z}) = \iota^*_{\mathcal{K}}(\boldsymbol{z})$, and using the following relation:

$$\begin{aligned} \iota_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}) &= \max_{\boldsymbol{z}} \left\{ \min_{\boldsymbol{v}\in\mathcal{K}} \langle \mathcal{A}\boldsymbol{x} - \boldsymbol{v}, \boldsymbol{z}\rangle \right\} \\ &= \max_{\boldsymbol{z}} \left\{ \langle \mathcal{A}\boldsymbol{x}, \boldsymbol{z}\rangle - \iota^*_{\mathcal{K}}(\boldsymbol{z}) \right\}. \end{aligned} \tag{4.8}$$

Choosing the standard Euclidean prox-function, we get the following "smooth approximation":

$$\iota_{\mathcal{K}\beta}(\mathcal{A}\boldsymbol{x}) = \max_{\boldsymbol{z}} \left\{ \min_{\boldsymbol{v}\in\mathcal{K}} \langle \mathcal{A}\boldsymbol{x} - \boldsymbol{v}, \boldsymbol{z}\rangle - \frac{\beta}{2}\|\boldsymbol{z}\|^2 \right\} \tag{4.9}$$

$$= \min_{\boldsymbol{v}\in\mathcal{K}} \max_{\boldsymbol{z}} \left\{ \langle \mathcal{A}\boldsymbol{x} - \boldsymbol{v}, \boldsymbol{z}\rangle - \frac{\beta}{2}\|\boldsymbol{z}\|^2 \right\} \tag{4.10}$$

$$= \frac{1}{2\beta}\operatorname{dist}^2(\mathcal{A}\boldsymbol{x}, \mathcal{K}), \tag{4.11}$$

where the inversion of min and max holds due to the Sion's minimax theorem [Sio58].

In summary, we can obtain the quadratic penalty form by applying Nesterov smoothing procedure to the indicator of an affine constraint. However, quadratic penalty does not serve as a uniform approximation, because the dual domain is unbounded and the envelop property (4.4) does not hold. Consequently, the common analysis techniques for smoothing does not apply for quadratic penalty methods.

Nevertheless, we exploit this structural similarity to design algorithms that work for both cases; composite problems with smoothing-friendly non-smooth regularizers and problems with affine constraints.

4.2 Homotopy CGM

Consider the following auxiliary optimization problem:

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad F_\beta(\boldsymbol{x}) := f(\boldsymbol{x}) + g_\beta(\mathcal{A}\boldsymbol{x}) \tag{4.12}$$

where g_β is the smooth approximation of g:

$$g_\beta(\mathcal{A}\boldsymbol{x}) = \max_{\boldsymbol{u}\in\mathbb{R}^d}\left\{\langle \mathcal{A}\boldsymbol{x}, \boldsymbol{u}\rangle - g^*(\boldsymbol{u}) - \frac{\beta}{2}\|\boldsymbol{u}\|^2\right\}. \tag{4.13}$$

Note that F_β is a smooth and convex function.

Solutions of (4.12) and (4.1) do not coincide for any particular β. Nevertheless, as $\beta \to 0$, any sequence of solutions of (4.12) converges to a solution of (4.1).

Based on these observations, we design HCGM as follows: At iteration t, we take a conditional gradient step with respect to the smooth approximation F_{β_t}. Then, we choose the next penalty parameter β_{t+1} slightly smaller than β_t to guide the iterates toward the solution of the original problem (4.1).

4.2.1 HCGM Iteration

Let $\beta_0 > 0$ be an initial smoothing parameter. Begin with an arbitrary choice $\boldsymbol{x}_0 \in \mathcal{X}$ for the decision variable. At each iteration $t = 0, 1, 2, \ldots$, we compute the gradient $\nabla F_{\beta_t}(\boldsymbol{x}_t)$;

$$\nabla F_{\beta_t}(\boldsymbol{x}_t) = \nabla f(\boldsymbol{x}_t) + \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \quad \text{where} \quad \beta_t = \frac{\beta_0}{\sqrt{t+2}}. \tag{4.14}$$

The argument of the maximization subproblem (4.13) can be written as $\operatorname{prox}_{\beta^{-1}g^*}(\beta^{-1}\mathcal{A}\boldsymbol{x})$. Hence, we can compute the gradient of g_{β_t} by using

$$\begin{aligned}\nabla g_\beta(\mathcal{A}\boldsymbol{x}_t) &= \mathcal{A}^* \operatorname{prox}_{\beta^{-1}g^*}(\beta^{-1}\mathcal{A}\boldsymbol{x}_t) \\ &= \beta^{-1}\mathcal{A}^*\left(\mathcal{A}\boldsymbol{x}_t - \operatorname{prox}_{\beta g}(\mathcal{A}\boldsymbol{x}_t)\right),\end{aligned} \tag{4.15}$$

where the second line follows from the well-known Moreau decomposition.

Next, we evaluate the linear minimization oracle

$$\boldsymbol{h}_t = \operatorname{lmo}_{\mathcal{X}}\big(\nabla F_{\beta_t}(\boldsymbol{x}_t)\big). \tag{4.16}$$

Finally, we update the matrix variable as follows:

$$\boldsymbol{x}_{t+1} = (1-\eta_t)\boldsymbol{x}_t + \eta_t \boldsymbol{h}_t, \quad \text{where} \quad \eta_t = 2/(t+2). \tag{4.17}$$

4.2.2 Stopping Criterion

We can design a stopping criterion for HCGM. If $g : \mathbb{R}^d \to \mathbb{R}$ is a Lipschitz continuous function, then we can use the following condition to stop our algorithm:

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{h}_t \rangle \leq \varepsilon, \qquad \text{which ensures} \qquad F(\boldsymbol{x}_t) - F_\star \leq \varepsilon. \tag{4.18}$$

If g is an indicator function, then we can use the following criterion to guarantee an ε-solution in the sense (1.15):

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{x}_t - \boldsymbol{h}_t \rangle \leq \varepsilon \quad \text{and} \quad \|\mathcal{A}\boldsymbol{x}_t - \text{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_t)\| \leq \varepsilon. \tag{4.19}$$

We can also choose different target accuracies for the objective residual and the feasibility gap.

4.2.3 Guarantees

Theorem 4.1. *Sequence $(\boldsymbol{x}_t : t = 1, 2, \dots)$ generated by HCGM satisfies*

$$F_{\beta_{t-1}}(\boldsymbol{x}_t) - f_\star \leq 2D_{\mathcal{X}} \left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0 \sqrt{t+1}} \right). \tag{4.20}$$

Theorem 4.1 does not directly guarantee the convergence of $\boldsymbol{x}_t$ to a solution of (1.1), since the bound is on smoothed gap $F_{\beta_{t-1}}(\boldsymbol{x}_t) - f_\star$. To relate $F_{\beta_{t-1}}$ back to F, one usually assumes Lipschitz continuity. This well known perspective leads us to Theorem 4.2, which is a direct extension of Theorem 4 in [Lan14] for composite functions.

Theorem 4.2. *Assume that $g : \mathbb{R}^d \to \mathbb{R}$ is L_g-Lipschitz continuous. Then, the sequence $(\boldsymbol{x}_t : t = 1, 2, \dots)$ generated by HCGM satisfies the following convergence bound:*

$$F(\boldsymbol{x}_t) - F_\star \leq 2D_{\mathcal{X}}^2 \left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0 \sqrt{t+1}} \right) + \frac{\beta_0 L_g^2}{2\sqrt{t+1}}. \tag{4.21}$$

Furthermore, if the constants $D_{\mathcal{X}}$, $\|\mathcal{A}\|$ and L_g are known or easy to approximate, we can choose $\beta_0 = 2D_{\mathcal{X}}\|\mathcal{A}\|/L_g$ to get the following convergence rate:

$$F(\boldsymbol{x}_t) - F_\star \leq \frac{2D_{\mathcal{X}}^2 L_f}{t+1} + \frac{2D_{\mathcal{X}}\|\mathcal{A}\| L_g}{\sqrt{t+1}}. \tag{4.22}$$

Lipschitz continuity assumption in Theorem 4.2 leaves many important applications out (see Sections 4.3.5 and 4.3.6). In Theorem 4.3, we take a step further and characterize the convergence when the non-smooth part is an indicator function.

Theorem 4.3. *Sequence* $(\boldsymbol{x}_t : t = 1,2,\dots)$ *generated by HCGM satisfies:*

$$f(\boldsymbol{x}_t) - f_\star \geq -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \tag{4.23}$$

$$f(\boldsymbol{x}_t) - f_\star \leq 2D_{\mathcal{X}} \left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0\sqrt{t+1}} \right) \tag{4.24}$$

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \leq \frac{2}{\sqrt{t+1}} \left(\beta_0\|\boldsymbol{y}_\star\| + D_{\mathcal{X}} \sqrt{\frac{\beta_0 L_f}{\sqrt{t+1}} + \|\mathcal{A}\|^2} \right) \tag{4.25}$$

Remark. $\boldsymbol{x}_t$ is not guaranteed to be a feasible point, since the condition $\mathcal{A}\boldsymbol{x}_t \in \mathcal{K}$ is not guaranteed. Nevertheless, it converges towards the feasible set, and the sequence $(\boldsymbol{x}_t : t = 0,1,2,\dots)$ is asymptotically feasible.

Remark. Similar to the classical CGM, we can consider a line-search variant of HCGM (which replaces the step size by $\eta_t = \min_{\eta\in[0,1]} F_{\beta_t}((1-\eta)\boldsymbol{x}_t + \eta\boldsymbol{h}_t)$), and a fully corrective variant (which replaces the last step by $\boldsymbol{x}_{t+1} = \arg\min_{\boldsymbol{x}\in\operatorname{conv}(\boldsymbol{h}_1,\dots,\boldsymbol{h}_t)} F_{\beta_t}(\boldsymbol{x})$). All results that we present in this chapter remain valid for these two design variants.

4.2.4 Convergence with Inexact Oracles

Finding an exact solution of lmo can be expensive in practice. On the other hand, approximate solutions can be much more efficient. For instance, exact computation of `MinEigVec` that we use in the SDP examples is costly, while we can use Lanczos method for an approximate computation.

Different notions of inexact lmo are already explored for CGM and greedy optimization frameworks, see [LJJSP13, LKTJ17, LTRJ17]. We revisit the notion of additive and multiplicative errors which we adapt here for our setting.

Inexact Oracle with Additive Error

At iteration t, for the given gradient direction $\nabla F_{\beta_t}(\boldsymbol{x}_t)$, we assume that the approximate lmo returns an element $\tilde{\boldsymbol{h}}_t \in \mathcal{X}$ such that:

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t \rangle \leq \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{h}_t \rangle + \delta \frac{\eta_t}{2} D_{\mathcal{X}}^2 \left(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t} \right) \quad \text{for some } \delta > 0, \tag{4.26}$$

where $\boldsymbol{h}_t$ denotes the exact solution of the lmo. As for the classical CGM, we require the accuracy of lmo to increase [Jag13] as the algorithm progresses.

Replacing the exact lmo with the approximate oracles of the form (4.26) in HCGM, we get the following convergence guarantees:

Theorem 4.4. *Sequence* $(\boldsymbol{x}_t : t = 1,2,\dots)$ *generated by HCGM with approximate lmo* (4.26) *satisfies:*

$$F_{\beta_{t-1}}(\boldsymbol{x}_t) - f_\star \le 2D_{\mathcal{X}}^2\left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0\sqrt{t+1}}\right)(1+\delta). \tag{4.27}$$

Theorem 4.5. *Assume that* g *is* L_g*-Lipschitz continuous. Then, the sequence* $\boldsymbol{x}_t$ *generated by HCGM with approximate lmo* (4.26) *satisfies:*

$$F(\boldsymbol{x}_t) - F_\star \le 2D_{\mathcal{X}}^2\left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0\sqrt{t+1}}\right)(1+\delta) + \frac{\beta_0 L_g^2}{2\sqrt{t+1}}. \tag{4.28}$$

Theorem 4.6. *Sequence* $(\boldsymbol{x}_t : t = 1,2,\dots)$ *generated by HCGM with approximate lmo* (4.26) *satisfies:*

$$f(\boldsymbol{x}_t) - f_\star \ge -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \tag{4.29}$$

$$f(\boldsymbol{x}_t) - f_\star \le 2D_{\mathcal{X}}^2\left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0\sqrt{t+1}}\right)(1+\delta) \tag{4.30}$$

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \le \frac{2}{\sqrt{t+1}}\left(\beta_0\|\boldsymbol{y}_\star\| + D_{\mathcal{X}}\sqrt{\left(\frac{\beta_0 L_f}{\sqrt{t+1}} + \|\mathcal{A}\|^2\right)(1+\delta)}\right). \tag{4.31}$$

Inexact Oracle with Multiplicative Error

We consider the multiplicative inexact oracle. Given a gradient direction $\nabla F_{\beta_t}(\boldsymbol{x}_t)$, we assume that the approximate lmo returns an element $\tilde{\boldsymbol{h}}_t \in \mathcal{X}$ such that:

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t\rangle \le \delta \cdot \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{h}_t - \boldsymbol{x}_t\rangle \quad \text{for some } \delta \in (0,1], \tag{4.32}$$

where $\boldsymbol{h}_t$ denotes the exact solution of the lmo. Replacing the exact lmo with the approximate oracles of the form (4.32) in HCGM, we get the following convergence guarantees:

Theorem 4.7. *Sequence* $(\boldsymbol{x}_t : t = 1,2,\dots)$ *generated by HCGM with approximate lmo* (4.32)*, and modifying* $\eta_t = \frac{2}{\delta t+2}$ *and* $\beta_t = \frac{\beta_0}{\sqrt{\delta(t+1)+1}}$ *satisfies:*

$$F_{\beta_{t-1}}(\boldsymbol{x}_{t+1}) - f_\star \le \frac{2}{\delta}\left(\frac{D_{\mathcal{X}}^2 L_f + \delta\mathcal{E}}{\delta(t+1)+2} + \frac{D_{\mathcal{X}}^2\|\mathcal{A}\|^2}{\beta_0\sqrt{\delta(t+1)+2}}\right) \quad \textit{where} \quad \mathcal{E} = F(\boldsymbol{x}_0) - f_\star. \tag{4.33}$$

Theorem 4.8. *Assume that* g *is* L_g*-Lipschitz continuous. Then, the sequence* $(\boldsymbol{x}_t : t = 1,2,\dots)$ *generated by HCGM with approximate lmo* (4.32)*, and modifying* $\eta_t = \frac{2}{\delta t+2}$ *and* $\beta_t = \frac{\beta_0}{\sqrt{\delta(t+1)+1}}$ *satisfies:*

$$F(\boldsymbol{x}_t) - F_\star \le \frac{2}{\delta}\left(\frac{D_{\mathcal{X}}^2 L_f + \delta\mathcal{E}}{\delta(t+1)+1} + \frac{D_{\mathcal{X}}^2\|\mathcal{A}\|^2}{\beta_0\sqrt{\delta(t+1)+1}}\right) + \frac{\beta_0 L_g^2}{2\sqrt{\delta(t+1)+1}}, \quad \textit{where} \quad \mathcal{E} = F(\boldsymbol{x}_0) - F_\star.$$

Theorem 4.9. *Sequence* $(\boldsymbol{x}_t : t = 1,2,\dots)$ *generated by HCGM with approximate lmo* (4.32)*, and modifying* $\eta_t = \frac{2}{\delta t+2}$ *and* $\beta_t = \frac{\beta_0}{\sqrt{\delta(t+1)+1}}$ *satisfies:*

$$f(\boldsymbol{x}_t) - f_\star \geq -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \tag{4.34}$$

$$f(\boldsymbol{x}_t) - f_\star \leq \frac{2}{\delta}\left(\frac{D_{\mathcal{X}}^2 L_f + \delta\mathcal{E}}{\delta(t+1)+1} + \frac{D_{\mathcal{X}}^2\|\mathcal{A}\|^2}{\beta_0\sqrt{\delta(t+1)+1}}\right) \tag{4.35}$$

$$\operatorname{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \leq \frac{2\beta_0}{\sqrt{\delta(t+1)+1}}\left(\|\boldsymbol{y}_\star\| + \sqrt{\frac{D_{\mathcal{X}}^2 C_0 + \delta\mathcal{E}}{\beta_0\delta}}\right) \tag{4.36}$$

where $\mathcal{E} = F(\boldsymbol{x}_0) - f_\star$ *and* $C_0 = L_f + \|\mathcal{A}\|^2/\beta_0$.

4.2.5 Application to the SDP template

We can use HCGM for solving instances from the SDP template (1.4), by choosing

$$f(\boldsymbol{X}) = \langle \boldsymbol{C}, \boldsymbol{X}\rangle, \quad \text{and} \quad \mathcal{X} = \{\boldsymbol{X} : \boldsymbol{X} \in \mathbb{S}_+^n, \operatorname{Tr}\boldsymbol{X} = \alpha\}, \quad \text{and} \quad \mathcal{K} = \{\boldsymbol{b}\}. \tag{4.37}$$

We can compute the gradient of the smooth-objective F_{β_t} at $\boldsymbol{X}_t$ by

$$\nabla F_{\beta_t}(\boldsymbol{X}_t) = \boldsymbol{C} + \beta_t^{-1}\mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b}) \qquad \text{where} \qquad \beta_t = \frac{\beta_0}{\sqrt{t+2}}. \tag{4.38}$$

Then, we can evaluate $\operatorname{lmo}_{\mathcal{X}}$ by using `MinEigVec` which returns the minimum eigenvalue eigenvector pair:

$$\boldsymbol{H}_t = \alpha\boldsymbol{u}_t\boldsymbol{u}_t^* \quad \text{where} \quad (\boldsymbol{u}_t, \lambda_t) = \texttt{MinEigVec}(\boldsymbol{C} + \beta_t^{-1}\mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})) \tag{4.39}$$

Algorithm 4.1 shows the complete pseudocode of HCGM for this template.

Algorithm 4.1 HCGM for the standard SDP formulation (1.4)

Require: initial state $\boldsymbol{X}$; initial penalty parameter β_0; (optional) target accuracy ε

function HCGM
  for $t \leftarrow 0,1,2,\dots$ **do**
    $\beta \leftarrow \beta_0/\sqrt{t+2}$
    $(\boldsymbol{u}, \lambda) \leftarrow \texttt{MinEigVec}\big(\boldsymbol{C} + \beta^{-1}\mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})\big)$
    $\boldsymbol{H} \leftarrow \alpha\boldsymbol{u}\boldsymbol{u}^*$
    if $\langle \boldsymbol{X} - \boldsymbol{H}, \boldsymbol{C} + \beta^{-1}\mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})\rangle \leq \varepsilon$ **and** $\|\mathcal{A}\boldsymbol{X} - b\| \leq \varepsilon$ **then break for**
    end if
    $\eta \leftarrow 2/(t+2)$
    $\boldsymbol{X} \leftarrow (1-\eta)\boldsymbol{X} + \eta\boldsymbol{H}$
  end for
  return $\boldsymbol{X}$
end function

4.3 Applications & Related Work

CGM is proposed for the first time in the seminal work of Frank and Wolfe [FW56] for solving smooth convex optimization on a polytope. It is then progressively generalized for more general settings in [LP66, DH78, Dun79, Dun80]. Nevertheless, with the introduction of the fast gradient methods with $\mathcal{O}(1/t^2)$ rate by Nesterov [Nes83], the development of CGM-type methods entered into a stagnation period.

The recent developments in machine learning applications with vast data brought the scalability of the first order methods under scrutiny. As a result, there has been a renewed interest in CGM in the last decade. We compare our framework with the recent developments of CGM literature in different camps of problem templates below.

4.3.1 Smooth Problems

CGM is extended for the smooth convex minimization over a simplex by Clarkson [Cla10], for a spactrahedron by Hazan [Haz08], and for an arbitrary compact convex set by Jaggi [Jag13].

When applied to smooth problems, HCGM is equivalent to the classical CGM, and it recovers the known optimal $\mathcal{O}(1/t)$ convergence rate. We refer to [Jag13] for a review on the applications of the smooth template.

We need to mention that Nesterov [Nes17] relaxes the smoothness assumption, by showing that CGM also converges for weakly-smooth objectives, *i.e.*, objectives with Hölder continuous gradients of order $\nu \in (0,1]$.

4.3.2 Regularized Problems

CGM for composite problems is considered by Nesterov [Nes17] and Xu [Xu17]. A similar but slightly different template, where $\mathcal{X}$ and g are assumed to be a closed convex cone and a norm respectively, is also studied by Harchaoui et al. [HJN15]. However, these works are based on the resolvents of a modified oracle instead of the standard lmo,

$$\arg\min_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, \boldsymbol{v}\rangle + g(\mathcal{A}\boldsymbol{x}), \tag{4.40}$$

which can be expensive to compute unless $\mathcal{X} \equiv \mathbb{R}^p$, or $g = 0$.

HCGM applies to the problem template (4.1) by leveraging the prox of the regularizer and the lmo of the domain separately. This allows us to consider additional sparsity, group sparsity and structured sparsity promoting regularizations, elastic-net regularization, total variation regularization and many others under the CGM framework.

The semi-proximal mirror-prox method [HH15] is also based on the smoothing idea, but the motivation of this method is fundamentally different than ours. This method considers the regularizers for which the prox is difficult to compute, but can be approximated via CGM.

4.3.3 Non-Smooth Problems

Template (4.1) covers the non-smooth convex minimization template as a special case:

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad g(\mathcal{A}\boldsymbol{x}). \tag{4.41}$$

Unfortunately, the classical CGM provably cannot handle the non-smooth minimization, as shown by the following counter-example from [Nes17].

Example. Let $\mathcal{X}$ be the unit Euclidean norm ball in $\mathbb{R}^2$, and $g(\boldsymbol{x}) = \max\{x_{(1)}, x_{(2)}\}$. Clearly, $\boldsymbol{x}_\star = [-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}]^*$. Choose an initial point $\boldsymbol{x}_0 \neq \boldsymbol{x}_\star$. We can use an oracle that returns a subgradient $\nabla f(\boldsymbol{x}) \in \{\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]\}$ at any point $\boldsymbol{x} \in \mathcal{X}$. Therefore, lmo returns $\left[\begin{smallmatrix}-1\\0\end{smallmatrix}\right]$ or $\left[\begin{smallmatrix}0\\-1\end{smallmatrix}\right]$ at each iteration, and $\boldsymbol{x}_t$ belongs to the convex hull of $\{\boldsymbol{x}_0, \left[\begin{smallmatrix}-1\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\-1\end{smallmatrix}\right]\}$ which does not contain the solution.

Our framework escapes such issues by leveraging the prox of the objective function g. In the example above, prox_g corresponds to the projection onto the simplex. Often times the cost of prox_g is negligible in comparison to the cost of $\text{lmo}_{\mathcal{X}}$.

Assume that $g : \mathbb{R}^d \to \mathbb{R}$ is L_g-Lipschitz continuous. As a consequence of Theorem 4.2, HCGM for solving (4.41) by choosing $\beta_0 = 2D_{\mathcal{X}}\|\mathcal{A}\|/L_g$ satisfies

$$g(A\boldsymbol{x}_t) - g_\star \leq \frac{2D_{\mathcal{X}}\|\mathcal{A}\|L_g}{\sqrt{t+1}}. \tag{4.42}$$

We recover the method proposed by Lan [Lan14] in this specific setting. He also proves that this rate is optimal for any algorithm that approximates the solution of (4.41) as a convex combination of lmo outputs.

We extend the analysis in this setting for inexact oracles. In contrast to the smooth case, where the additive error should decrease by $\mathcal{O}(1/t)$ rate, definition (4.26) implies that we can preserve the convergence rate in the non-smooth case as long as the additive error is $\mathcal{O}(1/\sqrt{t})$.

4.3.4 Minimax Problems

We consider the minimax problems of the following form:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \max_{\boldsymbol{y}\in\mathcal{Y}} \quad \mathcal{L}(\mathcal{A}\boldsymbol{x}, \boldsymbol{y}) \tag{4.43}$$

where $\mathcal{L}$ is a smooth convex-concave function, *i.e.*, $\mathcal{L}(\cdot, \boldsymbol{y})$ is convex $\forall \boldsymbol{y} \in \mathcal{Y}$ and $\mathcal{L}(\mathcal{A}\boldsymbol{x}, \cdot)$ is concave $\forall \boldsymbol{x} \in \mathcal{X}$. Note that this formulation is a special instance of (4.41) with $g(\mathcal{A}\boldsymbol{x}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \mathcal{L}(\mathcal{A}\boldsymbol{x}, \boldsymbol{y})$. Consequently, we can apply HCGM if prox_g is tractable.

When $\mathcal{Y}$ admits an efficient projection oracle, prox_g is also efficient for bilinear saddle point problems $\mathcal{L}(\mathcal{A}\boldsymbol{x}, \boldsymbol{y}) = \langle \mathcal{A}\boldsymbol{x}, \boldsymbol{y} \rangle$. By Moreau decomposition, we have

$$\text{prox}_g(\mathcal{A}\boldsymbol{x}_t) = \mathcal{A}\boldsymbol{x}_t - \text{proj}_{\mathcal{Y}}(\mathcal{A}\boldsymbol{x}_t), \tag{4.44}$$

hence $\nabla F_{\beta_t}(\boldsymbol{x}_t)$ takes the form

$$\nabla F_{\beta_t}(\boldsymbol{x}_t) = \nabla f(\boldsymbol{x}_t) + \frac{1}{\beta_t}\mathcal{A}^* \text{proj}_{\mathcal{Y}}(\mathcal{A}\boldsymbol{x}_t). \tag{4.45}$$

Gidel et al. [GJLJ17] propose a CGM variant for the smooth convex-concave saddle point problems. This method processes both $\boldsymbol{x}$ and $\boldsymbol{y}$ via the lmo, and hence it also requires $\mathcal{Y}$ to be bounded. Our method, on the other hand, is more suitable when $\text{proj}_{\mathcal{Y}}$ is easy.

Bilinear saddle point problem covers the maximum margin estimation of structured output models [TLJJ06] and minimax games [VNM44]. In particular, it also covers an important semidefinite programming formulation [GH16], where $\mathcal{X}$ is a spactrahedron and $\mathcal{Y}$ is the simplex. Our framework fits perfectly here since the projection onto the simplex can be computed efficiently. We defer the extension of our framework with the entropy Bregman smoothing for future.

Note that the CGM is applicable also for the variational inequality problems beyond (4.1), see [Ham84], [JN16] and [CJN17].

4.3.5 Problems with Affine Constraints

Our algorithms apply to smooth convex minimization problems with affine constraints over a convex compact set:

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad f(\boldsymbol{x}) \quad \text{subject to} \quad \mathcal{A}\boldsymbol{x} \in \mathcal{K}, \tag{4.46}$$

by setting $g(\mathcal{A}\boldsymbol{x})$ in (4.1) as indicator function of set $\mathcal{K}$.

Since the prox operator of the indicator function is the projection, $\nabla F_{\beta_t}(\boldsymbol{x}_t)$ becomes

$$\nabla F_{\beta_t}(\boldsymbol{x}_t) = \nabla f(\boldsymbol{x}_t) + \frac{1}{\beta_t}\mathcal{A}^*\big(\mathcal{A}\boldsymbol{x}_t - \text{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_t)\big). \tag{4.47}$$

We implicitly assume that $\text{proj}_{\mathcal{K}}$ is tractable. We can use a splitting scheme when it is computationally more advantageous to use $\text{lmo}_{\mathcal{K}}$ instead. See Section 4.3.6 for the details.

This template covers the standard semidefinite programming in particular. Applications include clustering [PW07], optimal power-flow [LL12], sparse PCA [dGJL07], kernel learning [LCG+04], blind deconvolution [ARR14], community detection [BBV16], *etc.* Besides machine learning applications, this formulation has a crucial role in the convex relaxation of combinatorial problems.

A significant example is the problems over the doubly nonnegative cone (*i.e.*, the intersection of the positive semidefinite cone and the positive orthant) with a bounded trace norm [YM10]. Note that the lmo over this domain can be costly since the lmo can require full dimensional updates [HJL96, LTRJ17]. Our framework can handle these problems ensuring the positive semidefiniteness by $\mathrm{lmo}_{\mathcal{X}}$, and can still ensure the convergence to the first orthant via $\mathrm{proj}_{\mathcal{K}}$.

4.3.6 Minimization via Splitting

We can take advantage of splitting since we can handle affine constraints. This lets us to disentangle the complexity of the constraints. Consider the following optimization template:

$$\begin{aligned} &\underset{\boldsymbol{x}\in\mathcal{X}_1\cap\mathcal{X}_2}{\text{minimize}} && f(\boldsymbol{x}) + g(\mathcal{A}_1\boldsymbol{x}) \\ &\text{subject to} && \mathcal{A}_2\boldsymbol{x}\in\mathcal{K}, \quad \mathcal{A}_3\boldsymbol{x}\in\mathcal{S} \end{aligned} \tag{4.48}$$

where $\mathcal{X}_1, \mathcal{X}_2 \subset \mathbb{R}^p$ are two convex compact sets, $\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$ are known linear maps.

Suppose that
· $\mathrm{lmo}_{\mathcal{X}_1}$ and $\mathrm{lmo}_{\mathcal{X}_2}$ are easy to compute, but not $\mathrm{lmo}_{\mathcal{X}_1\cap\mathcal{X}_2}$
· prox_g is easy to compute
· $\mathcal{K}$ is a simple convex set and $\mathrm{proj}_{\mathcal{K}}$ is efficient
· $\mathcal{S}$ is a convex compact set with an efficient lmo.

We can reformulate this problem introducing slack variables $\boldsymbol{\xi}\in\mathcal{X}_2$ and $\boldsymbol{\zeta}\in\mathcal{S}$ as follows:

$$\begin{aligned} &\underset{\substack{\boldsymbol{x}\in\mathcal{X}_1\ \boldsymbol{\xi}\in\mathcal{X}_2\\ \boldsymbol{\zeta}\in\mathcal{S}}}{\text{minimize}} && f(\boldsymbol{x}) + g(\mathcal{A}_1\boldsymbol{x}) \\ &\text{subject to} && \mathcal{A}_2\boldsymbol{x}\in\mathcal{K}, \quad \mathcal{A}_3\boldsymbol{x}=\boldsymbol{\zeta}, \quad \boldsymbol{x}=\boldsymbol{\xi}. \end{aligned} \tag{4.49}$$

This formulation is in the form of (4.1) with respect to the variable $(\boldsymbol{x},\boldsymbol{\xi},\boldsymbol{\zeta})\in\mathcal{X}_1\times\mathcal{X}_2\times\mathcal{S}$. It is easy to verify that HCGM leverages $\mathrm{lmo}_{\mathcal{X}_1}$, $\mathrm{lmo}_{\mathcal{X}_2}$, $\mathrm{lmo}_{\mathcal{S}}$, prox_g and $\mathrm{proj}_{\mathcal{K}}$ separately. This approach can be generalized for an arbitrary finite number of non-smooth terms in a straightforward way.

4.4 Numerical Experiments

This section presents numerical experiments supporting our theoretical findings in clustering and robust PCA examples.

4.4.1 Clustering the MNIST dataset

We consider the model-free k-means clustering SDP formulation of Peng and Wei [PW07]:

$$\text{minimize} \quad \langle \boldsymbol{C}, \boldsymbol{X} \rangle \quad \text{subject to} \quad \underbrace{\boldsymbol{X}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{X} \geq 0}_{\mathcal{A}\boldsymbol{X} \in \mathcal{K}}, \quad \underbrace{\boldsymbol{X} \in \mathbb{S}^n_+, \quad \operatorname{Tr} \boldsymbol{X} = k}_{\boldsymbol{X} \in \mathcal{X}}. \tag{4.50}$$

where $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ is the Euclidean distance matrix.

We use the setup described in [MVW17] and made available online by the authors. We can briefly describe this setup as follows: First, meaningful features from MNIST dataset[1] which consists of 28×28 grayscale images that can be stacked as 784×1 vectors, are extracted using a one-layer neural network. This gives us a weight matrix $\boldsymbol{W} \in \mathbb{R}^{784 \times 10}$ and a bias vector $\boldsymbol{b} \in \mathbb{R}^{10}$. Then, the trained neural network is applied to the first 1000 elements of the test set, which gives the probability vectors for these 1000 test points, where each entry represents the probability of being each digit.

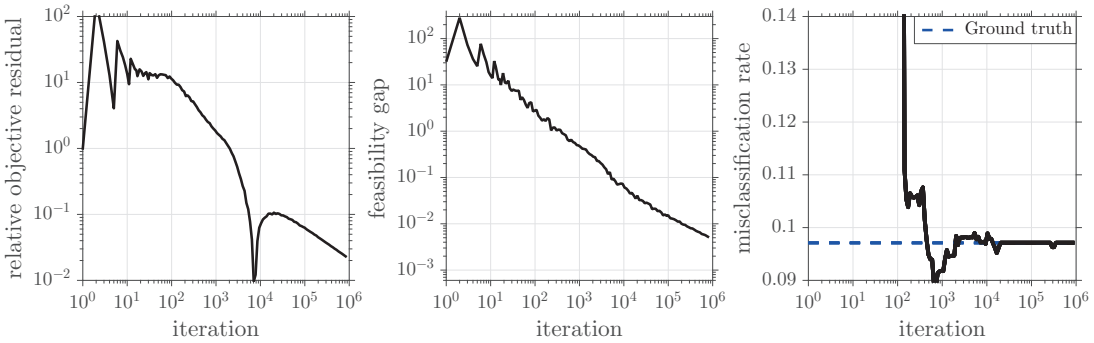


Figure 4.1 – Clustering MNIST: Empirical performance of our method. Blue dashed line on the misclassification plot represents the value reported in [MVW17].

[MVW17] runs a relax-and-round algorithm which solves (4.50) by SDPNAL+ [YST15] followed by a rounding scheme (see Section 5 of [MVW17] for details), and compares the results against MATLAB's built-in k-means++ implementation. Relax-and-round method is reported to achieve a misclassification rate of 0.0971. This rate matches with the all-time best rate for k-means++ after 100 different runs with random initializations.

For this experiment, we solve (4.50) by using HCGM. Then, we cluster data using the same rounding scheme as [MVW17]. We initialize our method from $\mathbf{0}$ and choose $\beta_0 = 1$. We implement the lmo using the built-in MATLAB function `eigs` with tolerance 10^{-9}.

[1] LeCun & Cortes. MNIST handwritten digit database. Available at "http://yann.lecun.com/exdb/mnist/"

We present the results of this experiment in Figure 4.1. We observe empirical $\mathcal{O}(1/\sqrt{t})$ rate both in the objective residual and the feasibility gap. Surprisingly, the method attains the best misclassification rate around 1000 iterations, achieving 0.0914. This improves the value reported in [MVW17] by 5.8%.

This example supports the claims that a low-to-medium accuracy solution is enough for many important SDP formulations, where we need a rounding step. In this example, the low-accuracy solution generalizes as well as the optimal point, even better.

4.4.2 Robust PCA

Suppose that we are given a large matrix that can be decomposed as the summation of a low-rank and a sparse (in some representation) matrix. Robust PCA aims to recover these components accurately, and it has many applications in machine learning and data science, such as collaborative filtering, system identification, genotype imputation, etc. Here, we focus on an image decomposition problem so that we can visualize the decomposition error results.

Our setting is similar to the setup described in [ZS18]. We consider a scaled grayscale photograph with pattern from [LMWY13], and we assume that we only have access to an occluded image. Moreover, the image is contaminated by salt and pepper noise of density 1/10. We seek to approximate the original from this noisy image.

This is essentially a matrix completion problem, and most of the scalable techniques rely on the Gaussian noise model. Note however the corresponding least-squares formulation is a good model against outliers:

$$\underset{\|\boldsymbol{X}\|_{S_1}\leq\rho}{\text{minimize}} \quad \frac{1}{2}\|\mathcal{A}\boldsymbol{X}-\boldsymbol{b}\|^2 \quad \text{subject to} \quad 0\leq\boldsymbol{X}\leq 1, \tag{4.51}$$

where $\mathcal{A}:\mathbb{R}^{p\times p}\to\mathbb{R}^d$ is the sampling operator.

Our framework also covers the following least absolute deviations formulation :

$$\underset{\|\boldsymbol{X}\|_{S_1}\leq\rho}{\text{minimize}} \quad \|\mathcal{A}\boldsymbol{X}-\boldsymbol{b}\|_1 \quad \text{subject to} \quad 0\leq\boldsymbol{X}\leq 1, \tag{4.52}$$

We solve both formulations with HCGM, starting from all zero matrix, running 1000 iterations, and assuming that we know the true nuclear norm of the original image. We choose $\beta_0 = 1$ in both cases.

This experiment demonstrates the implications of the flexibility of our framework in a simple machine learning setup. We compile the results in Figure 4.2, where the non-smooth formulation recovers a better approximation with 5dB higher peak signal to noise ratio (PSNR) and 0.27 higher structural similarity index (SSIM). Evaluation of PSNR and SSIM vs iteration counter are shown in Figure 4.3.

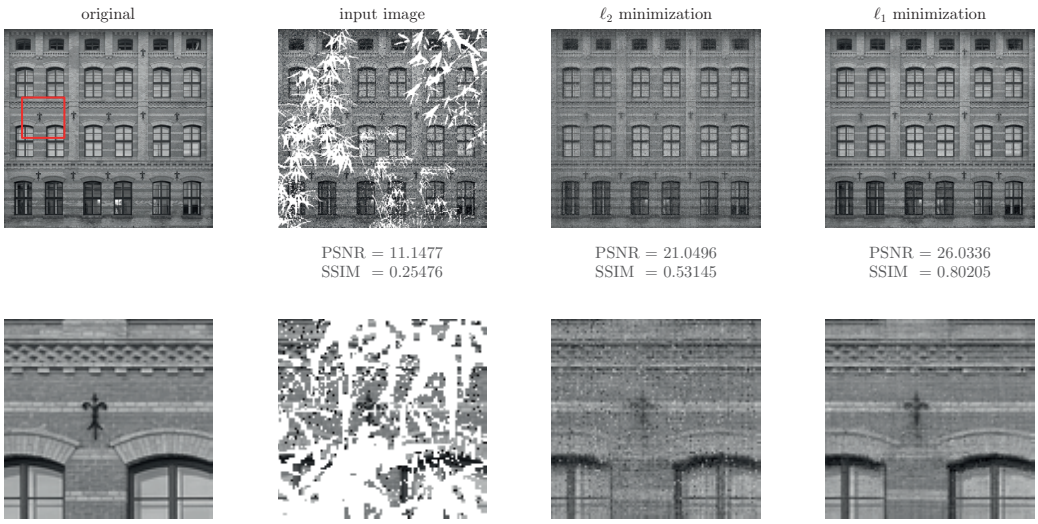


Figure 4.2 – Image inpainting from noisy test image (493 × 517): Robust PCA recovers a better approximation with 5dB higher PSNR.

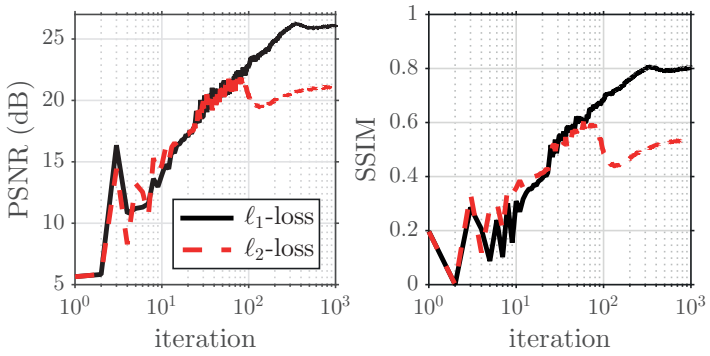


Figure 4.3 – PSNR and SSIM vs iteration counter for formulations with ℓ_1 and ℓ_2 loss.

4.5 Appendix: Proofs

Preliminaries

The following properties of smoothing are key to derive the convergence rate of our algorithms.

Let $g : \mathbb{R}^d \to \mathbb{R} \cap \{+\infty\}$ be a a proper, closed and convex function, and denote its smooth approximation by

$$g_\beta(\boldsymbol{z}) = \max_{\boldsymbol{y}\in\mathbb{R}^d} \langle \boldsymbol{z}, \boldsymbol{y}\rangle - g^*(\boldsymbol{y}) - \frac{\beta}{2}\|\boldsymbol{y}\|^2 \tag{4.53}$$

where g^* represents the Fenchel conjugate of g and $\beta > 0$ is the smoothing parameter. Then, g_β is convex and $\frac{1}{\beta}$-smooth. Let us denote the unique maximizer of this concave problem by

$$\begin{aligned}
\boldsymbol{y}^*_\beta(\boldsymbol{z}) &= \arg\max_{\boldsymbol{y}\in\mathbb{R}^d} \langle \boldsymbol{z}, \boldsymbol{y}\rangle - g^*(\boldsymbol{y}) - \frac{\beta}{2}\|\boldsymbol{y}\|^2 && (4.54)\\
&= \arg\min_{\boldsymbol{y}\in\mathbb{R}^d} \frac{1}{\beta} g^*(\boldsymbol{y}) - \frac{1}{\beta}\langle \boldsymbol{z}, \boldsymbol{y}\rangle + \frac{1}{2}\|\boldsymbol{y}\|^2 + \frac{1}{2}\|\frac{1}{\beta}\boldsymbol{z}\|^2 && (4.55)\\
&= \arg\min_{\boldsymbol{y}\in\mathbb{R}^d} \frac{1}{\beta} g^*(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{y} - \frac{1}{\beta}\boldsymbol{z}\|^2 && (4.56)\\
&= \mathrm{prox}_{\beta^{-1}g^*}(\beta^{-1}\boldsymbol{z}) = \frac{1}{\beta}\big(\boldsymbol{z} - \mathrm{prox}_{\beta g}(\boldsymbol{z})\big) && (4.57)
\end{aligned}$$

where the last equality is known as the Moreau decomposition. Then, the followings hold for $\forall \boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{R}^d$ and $\forall \beta, \gamma > 0$

$$g_\beta(\boldsymbol{z}_1) \geq g_\beta(\boldsymbol{z}_2) + \langle \nabla g_\beta(\boldsymbol{z}_2), \boldsymbol{z}_1 - \boldsymbol{z}_2\rangle + \frac{\beta}{2}\|\boldsymbol{y}^*_\beta(\boldsymbol{z}_2) - \boldsymbol{y}^*_\beta(\boldsymbol{z}_1)\|^2 \tag{4.58}$$

$$g(\boldsymbol{z}_1) \geq g_\beta(\boldsymbol{z}_2) + \langle \nabla g_\beta(\boldsymbol{z}_2), \boldsymbol{z}_1 - \boldsymbol{z}_2\rangle + \frac{\beta}{2}\|\boldsymbol{y}^*_\beta(\boldsymbol{z}_2)\|^2 \tag{4.59}$$

$$g_\beta(\boldsymbol{z}_1) \leq g_\gamma(\boldsymbol{z}_1) + \frac{\gamma - \beta}{2}\|\boldsymbol{y}^*_\beta(\boldsymbol{z}_1)\|^2 \tag{4.60}$$

Proofs can be found in Lemma 10 from [TDFC18].

Suppose that g is L_g-Lipschitz continuous. Then, for any $\beta > 0$ and any $\boldsymbol{z} \in \mathbb{R}^d$, the following bound holds:

$$g_\beta(\boldsymbol{z}) \leq g(\boldsymbol{z}) \leq g_\beta(\boldsymbol{z}) + \frac{\beta}{2}L_g^2 \tag{4.61}$$

Proof follows from equation (2.7) in [Nes05] with a remark on the duality between Lipshitzness and bounded support (see Lemma 5 in [DFTJ16] for a statement of this well-known duality).

Convergence analysis

We skip the proofs of Theorems 4.1 to 4.3 since we can get these results as a special case by setting $\delta = 0$ in Theorems 4.4 to 4.6.

Proof of Theorem 4.4

First, we use smoothness of F_{β_t} to upper bound the progress. Note that F_{β_t} is $(L_f + \|\mathcal{A}\|^2/\beta_t)$-smooth:

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) \leq F_{\beta_t}(\boldsymbol{x}_t) + \eta_t \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle + \frac{\eta_t^2}{2} \|\tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t\|^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}) \tag{4.62}$$

$$\leq F_{\beta_t}(\boldsymbol{x}_t) + \eta_t \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}), \tag{4.63}$$

where $\tilde{\boldsymbol{h}}_t$ denotes the atom selected by the inexact lmo, and the second inequality follows since $\tilde{\boldsymbol{h}}_t \in \mathcal{X}$.

By definition of inexact oracle (4.26), we have

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle \leq \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{h}_t - \boldsymbol{x}_t \rangle + \delta \frac{\eta_t}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}) \tag{4.64}$$

$$\leq \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle + \delta \frac{\eta_t}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}) \tag{4.65}$$

$$= \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle + \langle \mathcal{A}^* \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle + \delta \frac{\eta_t}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}), \tag{4.66}$$

where the second line follows since $\boldsymbol{h}_t$ is a solution of $\min_{\boldsymbol{x} \in \mathcal{X}} \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), x \rangle$.

Now, convexity of f ensures $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle \leq f(\boldsymbol{x}_\star) - f(\boldsymbol{x}_t)$. Using property (4.59), we have

$$\langle \mathcal{A}^* \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle = \langle \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \mathcal{A}\boldsymbol{x}_\star - \mathcal{A}\boldsymbol{x}_t \rangle \tag{4.67}$$

$$\leq g(\mathcal{A}\boldsymbol{x}_\star) - g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t) - \frac{\beta_t}{2} \|\boldsymbol{y}_{\beta_t}^*(\mathcal{A}\boldsymbol{x}_t)\|^2. \tag{4.68}$$

Putting these altogether, we get the following bound

$$\begin{aligned} F_{\beta_t}(\boldsymbol{x}_{t+1}) &\leq F_{\beta_t}(\boldsymbol{x}_t) + \eta_t \left(f(\boldsymbol{x}_\star) - f(\boldsymbol{x}_t) + g(\mathcal{A}\boldsymbol{x}_\star) - g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t) - \frac{\beta_t}{2} \|\nabla \boldsymbol{y}_{\beta_t}^*(\mathcal{A}\boldsymbol{x}_t)\|^2 \right) \\ &\quad + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta) \\ &= (1-\eta_t) F_{\beta_t}(\boldsymbol{x}_t) + \eta_t F_\star - \frac{\eta_t \beta_t}{2} \|\nabla \boldsymbol{y}_{\beta_t}^*(\mathcal{A}\boldsymbol{x}_t)\|^2 + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta). \end{aligned} \tag{4.69}$$

Now, using (4.60), we get

$$F_{\beta_t}(\boldsymbol{x}_t) = f(\boldsymbol{x}_t) + g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t) \tag{4.70}$$

$$\leq f(\boldsymbol{x}_t) + g_{\beta_{t-1}}(\mathcal{A}\boldsymbol{x}_t) + \frac{\beta_{t-1}-\beta_t}{2}\|\boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2 \tag{4.71}$$

$$= F_{\beta_{t-1}}(\boldsymbol{x}_t) + \frac{\beta_{t-1}-\beta_t}{2}\|\boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2. \tag{4.72}$$

We combine this with (4.69) and subtract $F_\star$ from both sides to get

$$\begin{aligned} F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star \leq (1-\eta_t)\big(F_{\beta_{t-1}}(\boldsymbol{x}_t) - F_\star\big) + \frac{\eta_t^2}{2}D^2_{\mathcal{X}}(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta) \\ + \big((1-\eta_t)(\beta_{t-1}-\beta_t) - \eta_t\beta_t\big)\frac{1}{2}\|\boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2. \end{aligned} \tag{4.73}$$

Let us choose η_t and β_t in a way to vanish the last term. By choosing $\eta_t = \frac{2}{t+2}$ and $\beta_t = \frac{\beta_0}{\sqrt{t+2}}$ for $t \geq 0$ with some $\beta_0 > 0$, we get $(1-\eta_t)(\beta_{t-1}-\beta_t) - \eta_t\beta_t < 0$. Hence, we end up with

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star \leq (1-\eta_t)\big(F_{\beta_{t-1}}(\boldsymbol{x}_t) - F_\star\big) + \frac{\eta_t^2}{2}D^2_{\mathcal{X}}(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta). \tag{4.74}$$

By recursively applying this inequality, we get

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star$$

$$\leq \prod_{j=0}^{t}(1-\eta_j)\Big(F_{\beta_{j-1}}(\boldsymbol{x}_t) - F_\star\Big) + \frac{1}{2}D^2_{\mathcal{X}}(1+\delta)\sum_{\ell=0}^{t}\eta_\ell^2\prod_{j=\ell}^{t}(1-\eta_j)(L_f + \frac{\|\mathcal{A}\|^2}{\beta_\ell}) \tag{4.75}$$

$$\leq \prod_{j=0}^{t}(1-\eta_j)\Big(F_{\beta_{j-1}}(\boldsymbol{x}_t) - F_\star\Big) + \frac{1}{2}D^2_{\mathcal{X}}(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta)\sum_{\ell=0}^{t}\eta_\ell^2\prod_{j=\ell}^{t}(1-\eta_j) \tag{4.76}$$

$$= \frac{1}{2}D^2_{\mathcal{X}}(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta)\sum_{\ell=0}^{t}\eta_\ell^2\prod_{j=\ell}^{t}(1-\eta_j), \tag{4.77}$$

where the second inequality follows since $\beta_t \leq \beta_j$ for any positive integer $j \leq t$, and the last line since $\eta_0 = 1$.

Now, we use the following relation

$$\sum_{\ell=0}^{t}\eta_\ell^2\prod_{j=\ell}^{t}(1-\eta_j) = \sum_{\ell=0}^{t}\frac{4}{(\ell+2)^2}\prod_{j=\ell}^{t}\frac{j}{j+2} = \sum_{\ell=0}^{t}\frac{4}{(\ell+2)^2}\frac{\ell(\ell+1)}{(t+1)(t+2)} \leq \frac{4}{t+2}, \tag{4.78}$$

which yields the first result of Theorem 4.4 as

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star \leq \frac{2}{t+2}D^2_{\mathcal{X}}(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})(1+\delta) = 2D^2_{\mathcal{X}}(\frac{L_f}{t+2} + \frac{\|\mathcal{A}\|^2}{\beta_0\sqrt{t+2}})(1+\delta). \tag{4.79}$$

Proof of Theorem 4.5

Now, we further assume that $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is L_g-Lipschitz continuous. From (4.61), we get

$$g(\mathcal{A}\boldsymbol{x}_{t+1}) \leq g_{\beta_t}(\mathcal{A}\boldsymbol{x}_{t+1}) + \frac{\beta_t L_g^2}{2} = g_{\beta_t}(\mathcal{A}\boldsymbol{x}_{t+1}) + \frac{\beta_0 L_g^2}{2\sqrt{t+2}}. \tag{4.80}$$

We complete the proof by adding $f(\boldsymbol{x}_{t+1}) - F_\star$ to both sides:

$$F(\boldsymbol{x}_{t+1}) - F_\star \leq F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star + \frac{\beta_0 L_g^2}{2\sqrt{t+2}}. \tag{4.81}$$

Proof of Theorem 4.6

From the Lagrange saddle point theory, we know that the following bound holds $\forall\, \boldsymbol{x} \in \mathcal{X}$ and $\forall\, \boldsymbol{v} \in \mathcal{K}$:

$$f_\star \leq \mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}_\star) = f(\boldsymbol{x}) + \langle \boldsymbol{y}_\star, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle \leq f(\boldsymbol{x}) + \|\boldsymbol{y}_\star\| \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\|, \tag{4.82}$$

Since $\boldsymbol{x}_{t+1} \in \mathcal{X}$, we get

$$f(\boldsymbol{x}_{t+1}) - f_\star \geq -\min_{\boldsymbol{v} \in \mathcal{K}} \|\boldsymbol{y}_\star\| \|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}\| = -\|\boldsymbol{y}_\star\| \mathrm{dist}(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}). \tag{4.83}$$

This proves the first bound in Theorem 4.6.

The second bound directly follows by Theorem 4.4 as

$$f(\boldsymbol{x}_{t+1}) - f_\star \leq f(\boldsymbol{x}_{t+1}) - f_\star + \frac{1}{2\beta_t} \mathrm{dist}^2(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}) := F_{\beta_t}(\boldsymbol{x}_{t+1}) - f_\star \tag{4.84}$$

$$\leq 2D_{\mathcal{X}}^2 \left(\frac{L_f}{t+2} + \frac{\|\mathcal{A}\|^2}{\beta_0 \sqrt{t+2}}\right)(1+\delta). \tag{4.85}$$

Denote by $\phi_t := 2D_{\mathcal{X}}^2 \left(\frac{L_f}{t} + \frac{\|\mathcal{A}\|^2}{\beta_0 \sqrt{t}}\right)(1+\delta)$. Then, we combine this with (4.83), and we get

$$-\|\boldsymbol{y}_\star\| \mathrm{dist}(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}) + \frac{1}{2\beta_t} \mathrm{dist}^2(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}) \leq \phi_{t+2}. \tag{4.86}$$

This is a second order inequality in terms of $\mathrm{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K})$. Solving this inequality, we get

$$\mathrm{dist}(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}) \leq \beta_t \left(\|\boldsymbol{y}_\star\| + \sqrt{\|\boldsymbol{y}_\star\|^2 + 2\frac{\phi_{t+2}}{\beta_t}} \right) \tag{4.87}$$

$$\leq \frac{2}{\sqrt{t+2}} \left(\beta_0 \|\boldsymbol{y}_\star\| + D_{\mathcal{X}} \sqrt{\left(\frac{L_f \beta_0}{\sqrt{t+2}} + \|\mathcal{A}\|^2 \right)(1+\delta)} \right). \tag{4.88}$$

Proof of Theorem 4.7

Let us define the multiplicative error δ of the LMO:

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle \leq \delta \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{h}_t - \boldsymbol{x}_t \rangle \tag{4.89}$$

For the proof we assume that $\boldsymbol{x}_0$ is feasible. First, we use the smoothness of F_{β_t} to upper bound the progress. Note that F_{β_t} is $(L_f + \|\mathcal{A}\|^2/\beta_t)$-smooth.

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) \leq F_{\beta_t}(\boldsymbol{x}_t) + \eta_t \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle + \frac{\eta_t^2}{2} \|\tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t\|^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}) \tag{4.90}$$

$$\leq F_{\beta_t}(\boldsymbol{x}_t) + \eta_t \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}), \tag{4.91}$$

where $\tilde{\boldsymbol{h}}_t$ denotes the atom selected by the inexact linear minimization oracle, and the second inequality follows since $\tilde{\boldsymbol{h}}_t \in \mathcal{X}$. By definition of inexact oracle (4.89), we have

$$\langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \tilde{\boldsymbol{h}}_t - \boldsymbol{x}_t \rangle \leq \delta \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{h}_t - \boldsymbol{x}_t \rangle \tag{4.92}$$

$$\leq \delta \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle \tag{4.93}$$

$$= \delta \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle + \delta \langle \mathcal{A}^* \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle, \tag{4.94}$$

where the second line follows since $\boldsymbol{h}_t$ is a solution of $\min_{\boldsymbol{x} \in \mathcal{X}} \langle \nabla F_{\beta_t}(\boldsymbol{x}_t), \boldsymbol{x} \rangle$.

Now, convexity of f ensures $\langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle \leq f(\boldsymbol{x}_\star) - f(\boldsymbol{x}_t)$. Using property (4.59), we have

$$\langle \mathcal{A}^* \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t \rangle = \langle \nabla g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t), \mathcal{A}\boldsymbol{x}_\star - \mathcal{A}\boldsymbol{x}_t \rangle \tag{4.95}$$

$$\leq g(\mathcal{A}\boldsymbol{x}_\star) - g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t) - \frac{\beta_t}{2} \|\boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2. \tag{4.96}$$

Putting these altogether, we get the following bound

$$\begin{aligned} F_{\beta_t}(\boldsymbol{x}_{t+1}) &\leq F_{\beta_t}(\boldsymbol{x}_t) + \eta_t \delta \left(f(\boldsymbol{x}_\star) - f(\boldsymbol{x}_t) + g(\mathcal{A}\boldsymbol{x}_\star) - g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t) - \frac{\beta_t}{2} \|\nabla \boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2 \right) \\ &\quad + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}) \\ &= (1 - \delta\eta_t) F_{\beta_t}(\boldsymbol{x}_t) + \delta\eta_t F_\star - \frac{\delta\eta_t\beta_t}{2} \|\nabla \boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2 + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}). \end{aligned} \tag{4.97}$$

Now, using (4.60), we get

$$F_{\beta_t}(\boldsymbol{x}_t) = f(\boldsymbol{x}_t) + g_{\beta_t}(\mathcal{A}\boldsymbol{x}_t) \tag{4.98}$$

$$\leq f(\boldsymbol{x}_t) + g_{\beta_{t-1}}(\mathcal{A}\boldsymbol{x}_t) + \frac{\beta_{t-1} - \beta_t}{2} \|\boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2 \tag{4.99}$$

$$= F_{\beta_{t-1}}(\boldsymbol{x}_t) + \frac{\beta_{t-1} - \beta_t}{2} \|\boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2. \tag{4.100}$$

We combine this with (4.97) and subtract $F_\star$ from both sides to get

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star \leq (1-\delta\eta_t)\big(F_{\beta_{t-1}}(\boldsymbol{x}_t) - F_\star\big) + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}) \tag{4.101}$$

$$+ \big((1-\delta\eta_t)(\beta_{t-1}-\beta_t) - \delta\eta_t\beta_t\big)\frac{1}{2}\|\nabla \boldsymbol{y}^*_{\beta_t}(\mathcal{A}\boldsymbol{x}_t)\|^2. \tag{4.102}$$

By choosing $\eta_t = \frac{2}{\delta t+2}$ and $\beta_t = \frac{\beta_0}{\sqrt{\delta(t+1)+1}}$ for some $\beta_0 > 0$, we get $(1-\delta\eta_t)(\beta_{t-1}-\beta_t) - \delta\eta_t\beta_t < 0$ for any $t \geq 0$ (with the convention $\beta_{-1} = \beta_0$, hence we end up with

$$F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star \leq (1-\delta\eta_t)\big(F_{\beta_{t-1}}(\boldsymbol{x}_t) - F_\star\big) + \frac{\eta_t^2}{2} D_{\mathcal{X}}^2 (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t}). \tag{4.103}$$

Let us call for simplicity $C := D_{\mathcal{X}}^2(L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})$, and $\mathcal{E}_{t+1} := F_{\beta_t}(\boldsymbol{x}_{t+1}) - F_\star$ Therefore, we have

$$\mathcal{E}_{t+1} \leq (1-\delta\eta_t)\mathcal{E}_t + \frac{\eta_t^2}{2} C \tag{4.104}$$

We now show by induction that:

$$E_t \leq 2\frac{\frac{1}{\delta}C + E_1}{\delta t + 2} \tag{4.105}$$

The base case $t = 0$ is trivial as $C > 0$. Call for simplicity $K := \delta t + 2$. Note that $K \geq 2$. Under this notation we can write $\eta_t = \frac{2}{\delta t+2} = \frac{2}{K}$ For the induction step, we add a positive term ($\mathcal{E}_0$ is positive as $\boldsymbol{x}_0$ is assumed feasible) to (4.104) and use the induction hypothesis:

$$E_{t+1} \leq (1-\delta\eta_t)E_t + \frac{\eta_t^2}{2}C + 2\delta\frac{E_1}{K^2} \tag{4.106}$$

$$\leq (1-\delta\frac{2}{K})E_t + \frac{2}{K^2}C + 2\delta\frac{E_1}{K^2} \tag{4.107}$$

$$\leq (1-\delta\frac{2}{K})2\frac{\frac{1}{\delta}C + E_1}{K} + \frac{2}{K^2}C + 2\delta\frac{E_1}{K^2} \tag{4.108}$$

$$= (1-\delta\frac{2}{K})2\frac{\frac{1}{\delta}C + E_1}{K} + 2\delta\left(\frac{\frac{1}{\delta}C}{K^2} + \frac{E_1}{K^2}\right) \tag{4.109}$$

$$= 2\frac{\frac{1}{\delta}C + E_1}{K}\left(1 - \delta\frac{2}{K} + \frac{\delta}{K}\right) \tag{4.110}$$

$$= 2\frac{\frac{1}{\delta}C + E_1}{K}\left(1 - \frac{\delta}{K}\right) \leq 2\frac{\frac{1}{\delta}C + E_1}{K+\delta} \tag{4.111}$$

noting that $K + \delta = \delta(t+1) + 2$ concludes the proof.

Proof of Theorems 4.8 and 4.9 follows similarly to the proofs of Theorems 4.5 and 4.6.

5 CGM via Augmented Lagrangian

In the previous chapter, we introduced a CGM extension with quadratic penalty (HCGM) for solving problems from template (1.4). HCGM is easy to implement with simple and interpretable steps. Unfortunately, many times, it performs poor in practice despite its strong theoretical guarantees.

To this end, we extend HCGM from quadratic penalty to an augmented Lagrangian framework in this chapter. The call the new variant as conditional gradient augmented Lagrangian framework (CGAL). CGAL retains similar guarantees as HCGM but it performs better in practice. We demonstrate the empirical superiority of CGAL against HCGM, as well as some other projection-free algorithms in the literature, in various numerical experiments.

This chapter is based on the joint work with Olivier Fercoq and Volkan Cevher [YFC19].

Introduction

As we have shown in Theorem 4.6, convergence rate of CGM is bounded above by $\mathcal{O}(1/\sqrt{t})$, both in objective residual and feasibility gap. However, in various numerical experiments, we observed that the method exhibits this "worst-case" rate in practice even when we are solving an arguably simple problem instance. Unfortunately, this undesirable behavior strains the practical impact of the method for solving problems at scale. In this chapter, we introduce CGAL, an extension HCGM from quadratic penalty to an augmented Lagrangian framework, in the spirit of [Ber76]. To address this issue, we introduce an extension of HCGM, from quadratic penalty to an augmented Lagrangian framework in the spirit of [Ber76].

Contributions

∘ We introduce a new algorithm for solving the problem template (1.1), based on a combination of CGM iterations and an augmented Lagrangian formulation. We show that the proposed method attains $\mathcal{O}(1/\sqrt{t})$ convergence rate both in objective residual and feasibility gap.

∘ We identify an implementable step-size rule for the dual updates thanks to the simplicity of our analysis. The cost of evaluating the proposed step-size is negligible. It retains the desirable theoretical convergence rates of HCGM while significantly enhancing the practical performance.

∘ We demonstrate the empirical superiority of CGAL in various SDP examples, in comparison with HCGM and some other CGM-type methods from the literature.

5.1 Augmented Lagrangian Penalty

First, we present a short overview of the augmented Lagrangian penalty and describe its relationship with the Nesterov smoothing and quadratic penalty techniques.

Similar to the quadratic penalty technique, we replace the affine constraint with a "feasibility-promoting" continuous penalty function in the augmented Lagrangian framework. Likewise, this penalty function is parametrized by the penalty parameter $\beta > 0$. However, in contrast with the quadratic penalty, we now introduce a "dual variable" $\boldsymbol{y} \in \mathbb{R}^d$ (also called as Lagrange multiplier). We can write the augmented Lagrangian penalty function as follows:

$$\min_{\boldsymbol{v}\in\mathcal{K}} \left\{ \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle + \frac{1}{2\beta} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\|^2 \right\}. \tag{5.1}$$

We can also view this function as a shifted quadratic penalty, since

$$\begin{aligned} &\arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \min_{\boldsymbol{v}\in\mathcal{K}} \left\{ \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle + \frac{1}{2\beta} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\|^2 \right\} && (5.2)\\ &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \min_{\boldsymbol{v}\in\mathcal{K}} \frac{1}{2\beta} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v} + \beta\boldsymbol{y}\|^2 && (5.3)\\ &= \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \frac{1}{2\beta} \operatorname{dist}^2(\mathcal{A}\boldsymbol{x} + \beta\boldsymbol{y}, \mathcal{K}). && (5.4) \end{aligned}$$

Recall our discussion in Section 4.1 on the similarity between the quadratic penalty and Nesterov smoothing techniques. It is not surprising that we can also relate the augmented Lagrangian with Nesterov smoothing. The derivation follows similarly as in Section 4.1.2 for the quadratic penalty, but instead of the simple Euclidean prox-function $\delta(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{v}\|^2$, this time we use a shifted prox-function $\delta(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{v} - \boldsymbol{y}\|^2$:

$$\begin{aligned} \iota_{\mathcal{K}\beta}(\mathcal{A}\boldsymbol{x}) &= \max_{\boldsymbol{z}} \left\{ \min_{\boldsymbol{v}\in\mathcal{K}} \langle \mathcal{A}\boldsymbol{x} - \boldsymbol{v}, \boldsymbol{z} \rangle - \frac{\beta}{2} \|\boldsymbol{z} - \boldsymbol{y}\|^2 \right\} && (5.5)\\ &= \min_{\boldsymbol{v}\in\mathcal{K}} \max_{\boldsymbol{z}} \left\{ \langle \mathcal{A}\boldsymbol{x} - \boldsymbol{v}, \boldsymbol{z} \rangle - \frac{\beta}{2} \|\boldsymbol{z} - \boldsymbol{y}\|^2 \right\} && (5.6)\\ &= \min_{\boldsymbol{v}\in\mathcal{K}} \left\{ \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle + \frac{1}{2\beta} \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\|^2 \right\}. && (5.7) \end{aligned}$$

Sion's minimax theorem [Sio58] justifies the inversion of the order of min and max.

In conclusion, we can say that the augmented Lagrangian formulation is structurally equivalent to a *de facto* instance of the Nesterov smoothing technique, applied to the indicator function of $\mathcal{K}$, and with a shifted Euclidean prox-function. Moreover, the dual variable corresponds to the center point of this prox-function.

5.2 Conditional Gradient Augmented Lagrangian Method

Consider the augmented Lagrangian of (1.1):

$$\begin{aligned}\mathcal{L}_\beta(\boldsymbol{x},\boldsymbol{y}) :&= f(\boldsymbol{x}) + \min_{\boldsymbol{v}\in\mathcal{K}}\left\{\langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x}-\boldsymbol{v}\rangle + \frac{1}{2\beta}\|\mathcal{A}\boldsymbol{x}-\boldsymbol{v}\|^2\right\}\\ &= f(\boldsymbol{x}) - \frac{\beta}{2}\|\boldsymbol{y}\|^2 + \frac{1}{2\beta}\operatorname{dist}^2(\mathcal{A}\boldsymbol{x}+\beta\boldsymbol{y},\mathcal{K}).\end{aligned} \tag{5.8}$$

Clearly $\mathcal{L}_\beta(\boldsymbol{x},\boldsymbol{y})$ is a smooth and convex function with respect to $\boldsymbol{x}$.

One CGAL iteration is composed of three basic steps:

- ▷ Primal step (conditional gradient step on $\boldsymbol{x}$),
- ▷ Penalty parameter update (decrement of β),
- ▷ Dual step (gradient step on $\boldsymbol{y}$).

5.2.1 CGAL Iteration

Primal step. CGAL is characterized by a conditional gradient step on the primal variable with respect to $\mathcal{L}_\beta(\cdot,\boldsymbol{y})$. At iteration $t = 0,1,\ldots$, we evaluate

$$\nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_t}(\boldsymbol{x}_t,\boldsymbol{y}_t) = \nabla f(\boldsymbol{x}_t) + \mathcal{A}^*\boldsymbol{y}_t + \beta_t^{-1}\mathcal{A}^*\big(\mathcal{A}\boldsymbol{x}_t - \operatorname{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_t+\beta\boldsymbol{y}_t)\big). \tag{5.9}$$

Then, we compute the linear minimization oracle

$$\boldsymbol{h}_t = \operatorname{lmo}_{\mathcal{X}}\big(\nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_t}(\boldsymbol{x}_t,\boldsymbol{y}_t)\big). \tag{5.10}$$

Finally, we form the next iterate by a generic conditional gradient step

$$\boldsymbol{x}_{t+1} = (1-\eta_t)\boldsymbol{x}_t + \eta_t\boldsymbol{h}_t, \quad \text{where} \quad \eta_t = 2/(t+2). \tag{5.11}$$

Penalty parameter update. We decrease the penalty parameter in CGAL at a controlled rate:

$$\beta_t = \frac{\beta_0}{\sqrt{t+2}} \quad \text{for some} \quad \beta_0 > 0. \tag{5.12}$$

Dual step. After computing $\boldsymbol{x}_{t+1}$ and updating the penalty parameter, we update the dual variable as follows:

$$\begin{aligned}\boldsymbol{y}_{t+1} &= \boldsymbol{y}_t + \sigma_{t+1}\nabla_{\boldsymbol{y}}\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{y}_t) \\ &= \boldsymbol{y}_t + \sigma_{t+1}\big(\mathcal{A}\boldsymbol{x}_{t+1} - \mathrm{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_{t+1} + \beta_{t+1}\boldsymbol{y}_t)\big).\end{aligned} \tag{5.13}$$

The choice of dual step-size σ_{t+1} has a significant effect on the performance of CGAL. We identify a performant choice as follows: First, define $\hat{\sigma}_{t+1}$ as

$$\hat{\sigma}_{t+1} := \min\left(\frac{1}{\beta_0}, \frac{(\eta_t\|\mathcal{A}\|D_{\mathcal{X}})^2}{2\beta_{t+1}\|\mathcal{A}\boldsymbol{x}_{t+1} - \mathrm{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_{t+1} + \beta_{t+1}\boldsymbol{y}_t)\|^2}\right), \tag{5.14}$$

Then, the recipe for σ_{t+1} is the following:

$$\sigma_{t+1} = \begin{cases} \hat{\sigma}_{t+1} & \text{if } \|\boldsymbol{y}_t + \hat{\sigma}(\mathcal{A}\boldsymbol{x}_{t+1} - \mathrm{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_{t+1} + \beta_{t+1}\boldsymbol{y}_t)\| \le D_{\mathcal{Y}} \\ 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

Here, $D_{\mathcal{Y}} > 0$ is the dual search-space parameter to be input by the user.

We emphasize that computing σ_{t+1} does not require an iterative line-search procedure. Instead, we can evaluate it by simple vector operations, hence *the computational cost of finding σ_{t+1} is negligible* in most applications.

5.2.2 Guarantees

Theorem 5.1. *Sequence $(\boldsymbol{x}_t : t = 1, 2, \dots)$ generated by CGAL satisfies:*

$$f(\boldsymbol{x}_t) - f_\star \ge -\|\boldsymbol{y}_\star\| \cdot \mathrm{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \tag{5.16}$$

$$f(\boldsymbol{x}_t) - f_\star \le 4D_{\mathcal{X}}^2\left(\frac{L_f}{t+1} + \frac{\|\mathcal{A}\|^2}{\beta_0\sqrt{t+1}} + \frac{2\|\mathcal{A}\|^2}{\beta_0(t+1)^{1.5}}\right) + \frac{\beta_0 D_{\mathcal{Y}}^2}{2\sqrt{t+1}} \tag{5.17}$$

$$\mathrm{dist}(\mathcal{A}\boldsymbol{x}_t, \mathcal{K}) \le \frac{1}{\sqrt{t+1}}\left(\beta_0\big(D_{\mathcal{Y}} + 2\|\boldsymbol{y}_t - \boldsymbol{y}_\star\|\big) + D_{\mathcal{X}}\sqrt{\frac{8L_f\beta_0}{\sqrt{t+1}} + \big(8 + \frac{1}{t+1}\big)\|\mathcal{A}\|^2}\right). \tag{5.18}$$

Considering these inequalities, we suggest choosing $D_{\mathcal{Y}}$ proportional to $D_{\mathcal{X}}\|\mathcal{A}\|/\beta_0$.

Remark. Similar to HCGM, we can extend CGAL for composite problems with Lipschitz continuous regularizers. In this case, dual update of CGAL corresponds to updating the center point of the prox-function. We omit the details.

Remark. We omit design variants of CGAL with line-search and fully corrective updates.

5.2.3 Application to the SDP template

Similar to HCGM, we can apply CGAL for solving instances of the SDP template (1.4). We can compute the directional derivative of the augmented Lagrangian function by

$$\nabla_{\boldsymbol{X}} \mathcal{L}_{\beta_t}(\boldsymbol{X}_t, \boldsymbol{y}_t) = \boldsymbol{C} + \mathcal{A}^* \boldsymbol{y}_t + \beta_t^{-1} \mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b}). \tag{5.19}$$

Then, we can evaluate the linear minimization oracle by using `MinEigVec`:

$$\boldsymbol{H}_t = \alpha \boldsymbol{u}_t \boldsymbol{u}_t^* \quad \text{where} \quad (\boldsymbol{u}_t, \beta_t^{-1}) = \texttt{MinEigVec}\big(\boldsymbol{C} + \mathcal{A}^* \boldsymbol{y}_t + \beta_t^{-1} \mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})\big). \tag{5.20}$$

Finally, since $\text{proj}_{\mathcal{K}}(\cdot) = \boldsymbol{b}$ for all inputs, the dual update of CGAL takes the following form:

$$\boldsymbol{y}_{t+1} = \boldsymbol{y}_t + \sigma_{t+1}(\mathcal{A}\boldsymbol{X}_{t+1} - \boldsymbol{b}), \tag{5.21}$$

where the dual step size is chosen as

$$\sigma_{t+1} = \begin{cases} \hat{\sigma}_{t+1} & \text{if } \|\boldsymbol{y}_t + \hat{\sigma}(\mathcal{A}\boldsymbol{X}_{t+1} - \boldsymbol{b})\| \le D_{\mathcal{Y}} \\ 0 & \text{otherwise} \end{cases} \quad \text{where} \quad \hat{\sigma}_{t+1} = \min\left(\frac{1}{\beta_0}, \frac{(\eta_t \|\mathcal{A}\| D_{\mathcal{X}})^2}{2\beta_{t+1}\|\mathcal{A}\boldsymbol{X}_{t+1} - \boldsymbol{b}\|^2}\right) \tag{5.22}$$

for some $D_{\mathcal{Y}} > 0$ to be specified by the user. Algorithm 5.1 shows the complete pseudocode.

Algorithm 5.1 CGAL for the standard SDP formulation (1.4)

Require: Data for (1.4); initial penalty parameter β_0; dual search-space parameter $D_{\mathcal{Y}}$

function CGAL
  $\boldsymbol{X} \leftarrow \boldsymbol{0}$ and $\boldsymbol{y} \leftarrow \boldsymbol{0}$
  for $t \leftarrow 0, 1, 2, \ldots$ **do**
    $\beta \leftarrow \beta_0 / \sqrt{t+2}$
    $(\boldsymbol{u}, \lambda) \leftarrow \texttt{MinEigVec}\big(\boldsymbol{C} + \mathcal{A}^* \boldsymbol{y} + \beta^{-1} \mathcal{A}^*(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})\big)$
    $\boldsymbol{H} \leftarrow \alpha \boldsymbol{u}\boldsymbol{u}^*$
    $\eta \leftarrow 2/(t+2)$
    $\boldsymbol{X} \leftarrow (1-\eta)\boldsymbol{X} + \eta \boldsymbol{H}$
    $\beta_+ \leftarrow \beta_0 / \sqrt{t+3}$
    $\sigma \leftarrow \text{DUALSTEPSIZE}(\boldsymbol{X}, \boldsymbol{y}, \beta_0, \beta_+, \eta)$
    $\boldsymbol{y} \leftarrow (1-\eta)\boldsymbol{y} + \sigma(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})$
  end for
  return $\boldsymbol{X}$
end function

---

function DUALSTEPSIZE($\boldsymbol{X}, \boldsymbol{y}, \beta_0, \beta_+, \eta$)
  $\sigma \leftarrow \min(\frac{(\eta\|\mathcal{A}\| D_{\mathcal{X}})^2}{2\beta_+ \|\mathcal{A}\boldsymbol{X} - \boldsymbol{b}\|^2}, 1/\beta_0)$
  if $\|\boldsymbol{y} + \sigma(\mathcal{A}\boldsymbol{X} - \boldsymbol{b})\| > D_{\mathcal{Y}}$ **then** $\sigma \leftarrow 0$ ▷ No dual step if $\|\boldsymbol{y}\|$ is growing too much
  end if
  return σ
end function

5.3 Numerical Experiments

In this section, we compare the empirical performance of CGAL against some existing methods in the literature in various SDP examples. Let us first provide a brief overview of the baseline methods that we will use in the comparisons.

Baseline methods

∘ **HCGM** that we introduced in [YFLC18] and described in Chapter 4, is the predecessor of CGAL. It has $\mathcal{O}(1/\sqrt{t})$ rate guarantee both in the objective residual and feasibility gap.

∘ **UPD** and **AUPD**, the universal primal-dual methods that we introduced in [YTDC15a] and described in Chapter 3: These methods are based on a primal-dual (sub)gradient formulation. Remark that the main template of UPD and AUPD is different from the template of CGAL. For example, they do not require the smoothness of f. Instead, they assume Hölder smoothness in the dual formulation. Nevertheless, both methods approaches work for the model SDP formulation (1.4). UPD and AUPD do not directly work with lmo. However, their sharp-operator can be cast as an instance of lmo for the SDP template (1.4).

These methods adopt an inexact line-search strategy by Nesterov [Nes15], which requires the target accuracy ε as an input parameter. Moreover, they are guaranteed to converge only up to $\varepsilon/2$ accuracy, *i.e.*, $f(\boldsymbol{x}) - f_\star \leq \mathcal{O}(1/\sqrt{t}) + \varepsilon/2$.

∘ **FW-AL** is a CGM-based augmented Lagrangian framework [GPLJ18]. Similar to CGAL, this method is characterized by one CGM step on $\mathcal{L}_\beta(\cdot, \boldsymbol{y}_t)$ followed by one dual gradient ascent step on $\mathcal{L}_\beta(\boldsymbol{x}_{t+1}, \cdot)$. In contrast to CGAL, the penalty parameter of the FW-AL is kept constant throughout the optimization process. Originally, FW-AL is proposed for $\mathcal{A}\boldsymbol{x} = \mathbf{0}$ type of constraints, but it can be applied to $\mathcal{A}\boldsymbol{x} = \boldsymbol{b}$ case by using a simple product space technique. The analysis of the FW-AL relies on the error bounds (see Theorem 1 in [GPLJ18] for the conditions, and [BNPS17] for more details about error bounds). Their dual step-size σ_{t+1} depends on the error bound constant α, as $\sigma_{t+1} = \frac{2\sigma_0}{t+2}$ with $\sigma_0 \leq \min\{\frac{2}{\lambda}, \frac{\alpha^2}{2\delta}\}$. As a results, σ_0 comes into algorithm as a tuning parameter, and the method has guarantees only if σ_0 is chosen small enough.

∘ **IAL** is an inexact augmented Lagrangian method, where the Lagrangian subproblems are approximately solved by using CGM up to a prescribed accuracy [LLM19]. With theoretical analysis, the error tolerance for this subproblem is determined as $\varepsilon_t = \varepsilon_0/t$ for some $\varepsilon_0 > 0$. IAL has a double-loop structure, where each outer iteration (or epoch) executes multiple iterations CGM, followed with a dual step.

IAL provably generates an ε-solution after $\mathcal{O}(1/\varepsilon^2)$ outer iterations, by choosing the penalty parameter β appropriately (proportional to $\sqrt{\varepsilon}$). This method, however, requires multiple calls of lmo at each iteration. The number of lmo calls to achieve the prescribed accuracy in the subproblem is bounded by $\mathcal{O}(1/\varepsilon_t)$ (see Theorem 2.2 in [LLM19]). Hence, the overall lmo complexity of this method becomes $\mathcal{O}(1/\varepsilon^4)$.

5.3.1 Max-cut

Maximum cut is an NP-Hard combinatorial problem from computer science. Denoting the graph Laplacian matrix by $\boldsymbol{L}$, an SDP relaxation of this problem can be formulated as [GW95]:

$$\text{minimize} \quad -\tfrac{1}{4}\langle \boldsymbol{L}, \boldsymbol{X}\rangle \quad \text{subject to} \quad \operatorname{diag}(\boldsymbol{X}) = \mathbf{1}, \quad \operatorname{Tr}\boldsymbol{X} = n, \quad \boldsymbol{X} \in \mathbb{S}^n_+. \tag{5.23}$$

Tuning all baseline methods requires substantial computational effort, especially because some of these methods have multiple tuning parameters. To this end, we first consider a small scale max-cut instance where we compare all baseline methods. In this setup we use GD97_b dataset[1], which corresponds to a 47×47 dimensional problem. In Figure 5.1, we present the performance of each method with the best parameter choice obtained after an extensive search. We also provide the performance of each algorithm at all trials at the end of this chapter in Figures 5.5 and 5.6.

Next, we consider a medium scale experiment, where we compare CGAL, HCGM and UPD for max-cut with G1 (800×800) and G40 (2000×2000) datasets[2]. We compile the results in Figure 5.2. Observe that HCGM converges with $\mathcal{O}(1/\sqrt{t})$ (which is the worst case bound) while CGAL achieves a faster rate.

5.3.2 k-means Clustering

Consider the SDP formulation of the model-free k-means clustering problem [PW07]:

$$\text{minimize} \quad \langle \boldsymbol{C}, \boldsymbol{X}\rangle \quad \text{subject to} \quad \underbrace{\boldsymbol{X}\mathbf{1} = \mathbf{1}, \quad \boldsymbol{X} \geq 0}_{\mathcal{A}\boldsymbol{X}\in\mathcal{K}}, \quad \underbrace{\boldsymbol{X} \in \mathbb{S}^n_+, \quad \operatorname{Tr}\boldsymbol{X} = k}_{\boldsymbol{X}\in\mathcal{X}}. \tag{5.24}$$

$\boldsymbol{C}$ is the Euclidean distance matrix and k is the number of clusters. We denote the vector of ones by $\mathbf{1}$, hence $\boldsymbol{X}\mathbf{1} = \mathbf{1}$ and $\boldsymbol{X} \geq 0$ together implies that each row of $\boldsymbol{X}$ is on the unit simplex. The same applies for columns of $\boldsymbol{X}$ due to symmetry.

We use the problem setup from [MVW17] that we have also considered in Section 4.4.1. This setup contains a 1000×1000 dimensional dataset generated by sampling and preprocessing the MNIST dataset[3] using a one-layer neural network. Further details on this setup and the dataset can be found in [MVW17]. In Figure 5.3, we observe once again that CGAL outperforms HCGM, achieving $\mathcal{O}(1/t)$ empirical convergence rate.

[1] V. Batagelj and A. Mrvar. Pajek datasets. Available at "http://vlado.fmf.uni-lj.si/pub/networks/data/"
[2] Y. Ye. Gset random graphs. Available at "https://www.cise.ufl.edu/research/sparse/matrices/gset/"
[3] LeCun & Cortes. MNIST handwritten digit database. Available at "http://yann.lecun.com/exdb/mnist/"

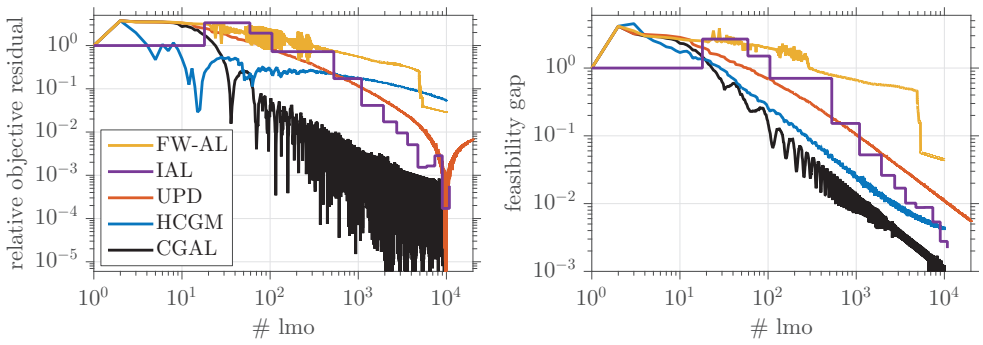


Figure 5.1 – Empirical performance of various methods for solving max-cut problem.

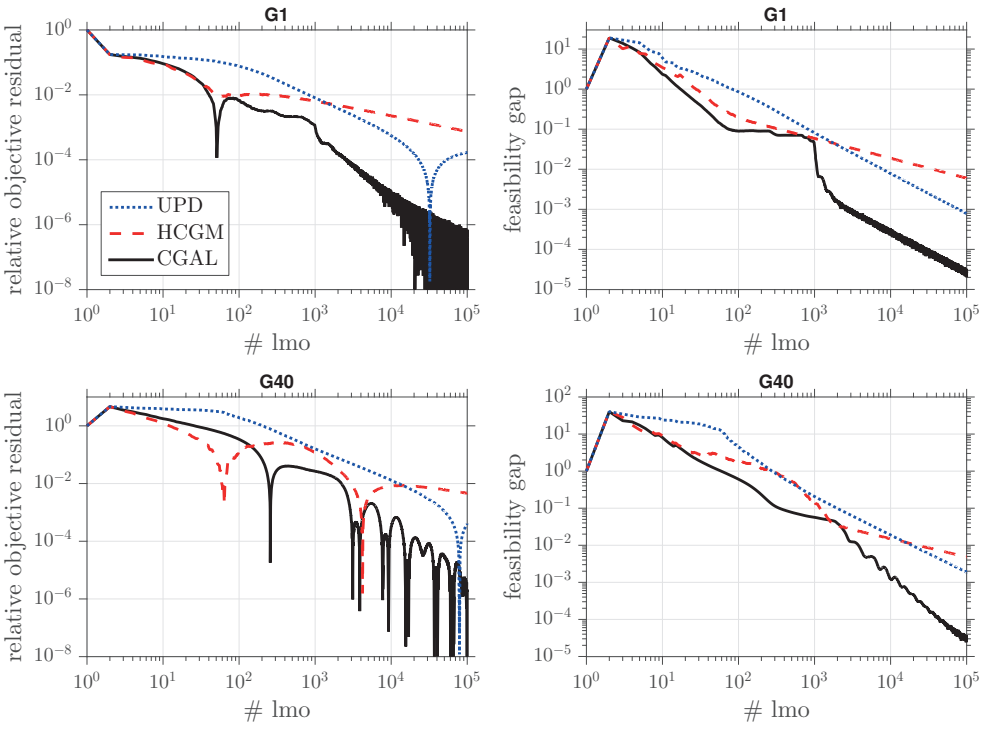


Figure 5.2 – Empirical comparison of UPD, HCGM and CGAL for max-cut problem.

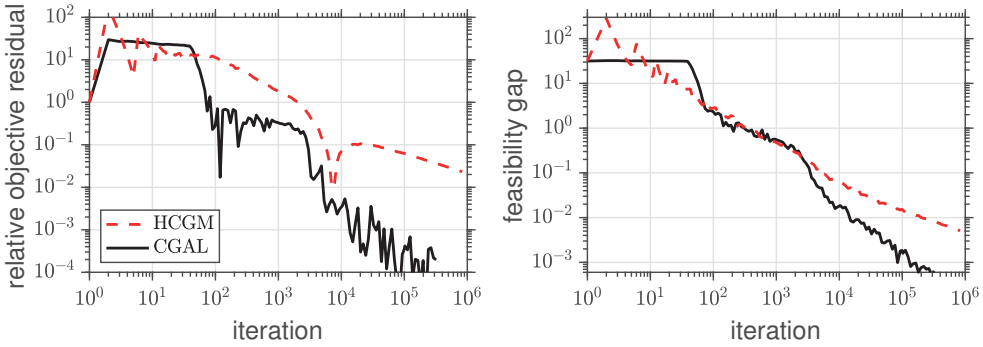


Figure 5.3 – Comparison of CGAL and HCGM for clustering SDP.

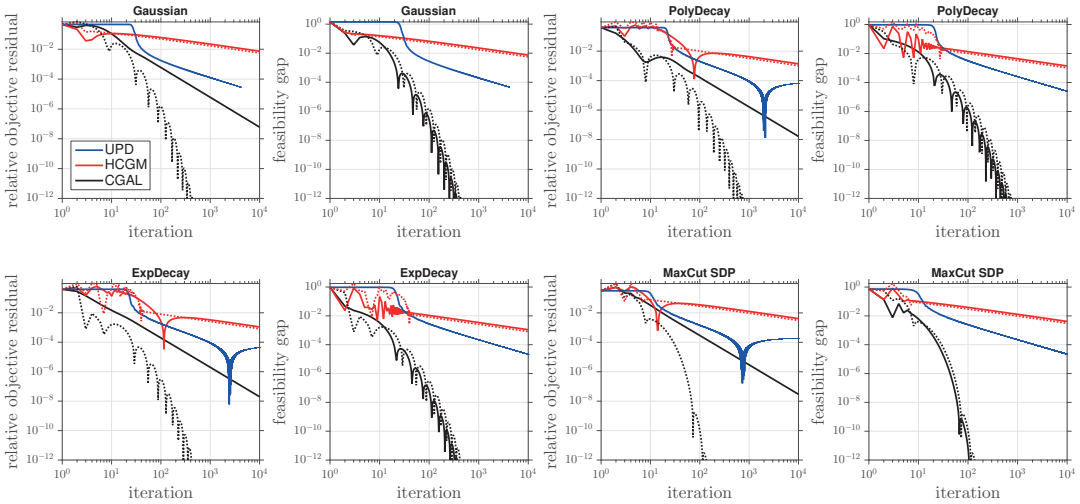


Figure 5.4 – Empirical comparison of CGAL, HCGM and UPD for solving generalized eigenvector problem with 4 different synthetic setups. Dotted lines present objective residual and feasibility gap of the atoms chosen by linear minimization oracle ($\boldsymbol{h}_t$).

5.3.3 Generalized Eigenvector Problem

Consider the SDP relaxation of the generalized eigenvector problem [BVB18]:

$$\text{maximize} \quad \langle \boldsymbol{C}, \boldsymbol{X} \rangle \quad \text{subject to} \quad \langle \boldsymbol{D}, \boldsymbol{X} \rangle = 1, \quad \boldsymbol{X} \in \mathbb{S}^n_+, \quad \operatorname{Tr} \boldsymbol{X} \leq \alpha. \tag{5.25}$$

where $\boldsymbol{C}$ and $\boldsymbol{C}$ are symmetric matrices and $\alpha > 0$ is a model parameter. We synthetically generate $\boldsymbol{D}$ with *iid* Gaussian entries and consider 4 different cases for $\boldsymbol{C}$:

- `Gaussian` - $\boldsymbol{C}$ generated by taking symmetric part of $10^3 \times 10^3$ *iid* Gaussian matrix
- `PolyDecay` - $\boldsymbol{C}$ generated by randomly rotating $\operatorname{diag}(1^{-i}, 2^{-i}, \ldots, 1000^{-i})$ $(i = 1)$
- `ExpDecay` - $\boldsymbol{C}$ generated by randomly rotating $\operatorname{diag}(10^{-i}, 10^{-2i}, \ldots, 10^{-1000i})$ $(i = 0.025)$
- `MaxCut SDP` - $\boldsymbol{C}$ is a solution of the max-cut SDP with G40 dataset (2000×2000)

This problem highlights an important observation that partially explains the reason why CGAL outperforms HCGM. First, note that this problem has a rank-1 solution by design if we set α right. In this scenario, if the problem formulation is well-conditioned, we could expect lmo to pick this solution (or some close points). CGAL updates the dual variable (which corresponds to the center point of a quadratic penalty) and adapts better to the geometry. In Figure 5.4, we provide an empirical evidence of this adaptation: Dotted lines correspond to extreme points chosen by lmo. These points ($\boldsymbol{h}_t$) converge (quickly with linear rates) to a solution for CGAL. On the ohter hand, we do not observe the same behavior for HCGM, because the condition number grows too much and lmo fails to pick good atoms. We omit lmo outputs of UPD in the figure because it does not converge to a solution.

Observations & Concluding Remarks

CGAL outperformed the alternative methods in all of our experiments. **HCGM** performed with the guaranteed "worst-case" rate in our experiments.

UPD and **AUPD** requires the target suboptimality level ε as an input parameter, and these methods have convergence guarantees only up to this target accuracy. Indeed, we can observe the saturation effect in the objective residual in our numerical experiments.

In our experiments, we observed similar performance from UPD and AUPD when solving max-cut problems. Remark that both methods have similar guarantees for this problem, since the dual subgradient is Hölder continuous or order $\nu = 0$. Surprisingly, AUPD did not exhibit the saturation in practice. We can explain this as follows: Both UPD and AUPD are based on the inexact line-search procedure, but UPD lets a constant $\frac{\varepsilon}{2}$ slackness term in the line-search condition, while AUPD has a more stringent slackness term $\approx \frac{\varepsilon}{2t}$. The is required from the theoretical perspective, in order to prevent error accumulation due to acceleration. However, the actual error accumulation might be much less than the worst-case scenario. As a result, by decreasing the slackness term we prevent the saturation.

Remark that, we can also design a non-saturating variant of UPD, by decreasing the slackness term at a controlled rate. This rate can be characterized from the theory. Note however, as we decrease it, we would also need more accurate evaluations of the sharp operator. In large-scale problems, where we use inexact oracles, this causes stability problems in the line-search procedure. Under noise, our line-search condition might become ill-defined, and the method can get stuck in the line-search procedure.

FW-AL iterations are very similar to CGAL, but the empirical behavior of the algorithm is different. The analysis of FW-AL relies on the error bound parameter, which is proved to be positive (assuming that Slater's condition holds). However, this parameter is typically unknown a priori, and we argue that it can be arbitrarily small. Hence, it might be difficult to tune in practice FW-AL for some problems.

IAL performed better in practice, in comparison with the $\mathcal{O}(1/\varepsilon^4)$ lmo complexity bound. Nevertheless, the method is arguably difficult to tune, as it involves multiple tuning parameters. IAL has a double-loop structure, and only the outer iterates are informative. This results in stair-like plots.

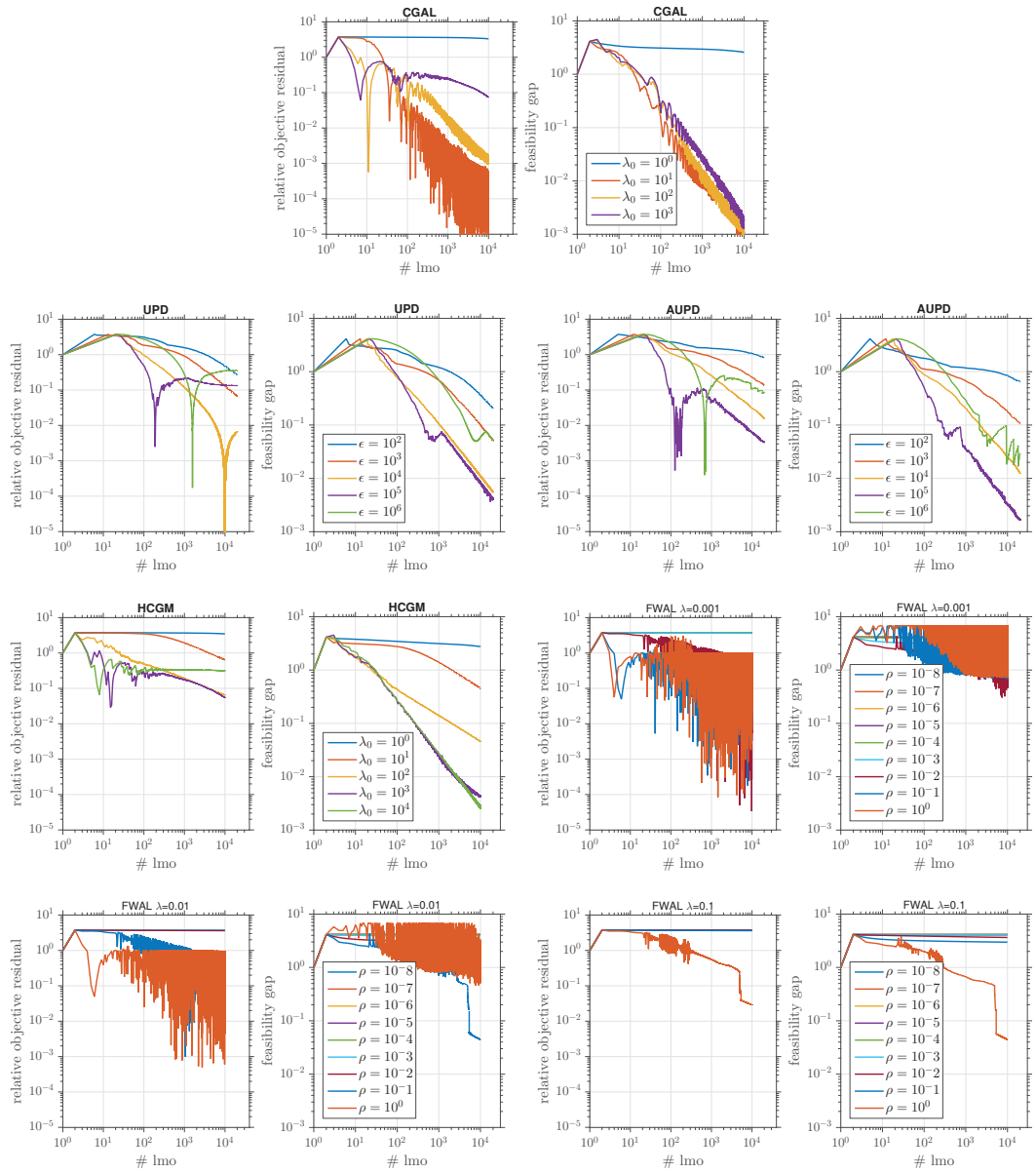


Figure 5.5 – Comparison of projection-free methods for max-cut SDP [Part 1].

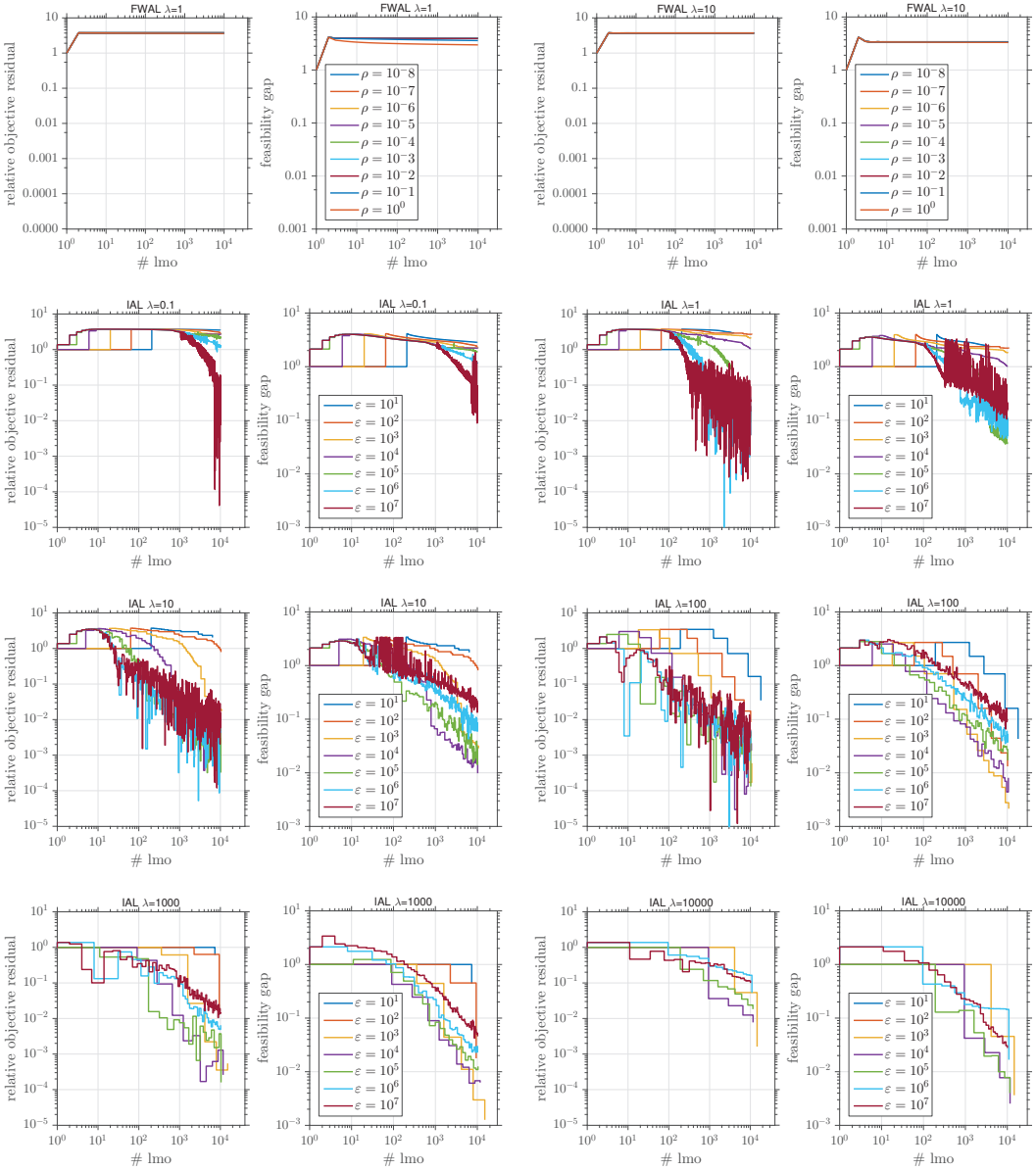


Figure 5.6 – Comparison of the projection-free methods for max-cut SDP [Part 2].

5.4 Appendix: Proofs

We formulate the augmented Lagrangian function with three variables, including the slack variable $\boldsymbol{v} = \mathcal{A}\boldsymbol{x} \in \mathcal{K}$, as follows:

$$\mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) := f(\boldsymbol{x}) + \langle \boldsymbol{y}, \mathcal{A}\boldsymbol{x} - \boldsymbol{v}\rangle + \frac{1}{2\beta}\|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\|^2, \tag{5.26}$$

where $\boldsymbol{y} \in \mathbb{R}^d$ is the Lagrange multiplier and $\beta > 0$ is the penalty parameter.

We can write the directional derivatives of the augmented Lagrangian function as

$$\nabla_{\boldsymbol{x}}\mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) = \nabla f(\boldsymbol{x}) + \mathcal{A}^*\boldsymbol{y} + \beta^{-1}\mathcal{A}^*(\mathcal{A}\boldsymbol{x} - \boldsymbol{v}) \tag{5.27}$$

$$\nabla_{\boldsymbol{v}}\mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) = -\boldsymbol{y} - \beta^{-1}(\mathcal{A}\boldsymbol{x} - \boldsymbol{v}) \tag{5.28}$$

$$\nabla_{\boldsymbol{y}}\mathcal{L}_\beta(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}) = \mathcal{A}\boldsymbol{x} - \boldsymbol{v}. \tag{5.29}$$

Clearly, L_{β_t} is $\bar{L}_t$-smooth, where we denote by $\bar{L}_t := (L_f + \frac{\|\mathcal{A}\|^2}{\beta_t})$. Now, let us define

$$\boldsymbol{v}_t := \mathrm{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_t + \beta_t\boldsymbol{y}_t) = \arg\min_{\boldsymbol{v}\in\mathcal{K}} \mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}, \boldsymbol{y}_t). \tag{5.30}$$

Then, using the Taylor expansion, we get the following estimate:

$$\begin{aligned}
\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_t, \boldsymbol{y}_t) &\le \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) + \langle \nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t), \boldsymbol{x}_{t+1} - \boldsymbol{x}_t\rangle + \frac{\bar{L}_{t+1}}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2 \\
&= \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) + \eta_t\langle \nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t), \boldsymbol{h}_t - \boldsymbol{x}_t\rangle + \eta_t^2\frac{\bar{L}_{t+1}}{2}\|\boldsymbol{h}_t - \boldsymbol{x}_t\|^2 \qquad (5.31) \\
&\le \mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) + \eta_t\langle \nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t), \boldsymbol{h}_t - \boldsymbol{x}_t\rangle + \eta_t^2\frac{\bar{L}_{t+1}}{2}D_{\mathcal{X}}^2 \\
&\quad + \frac{1}{2}(\beta_{t+1}^{-1} - \beta_t^{-1})\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 + \eta_t(\beta_{t+1}^{-1} - \beta_t^{-1})\langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t, \mathcal{A}\boldsymbol{h}_t - \mathcal{A}\boldsymbol{x}_t\rangle. \qquad (5.32)
\end{aligned}$$

Now, we can bound the inner product term as follows:

$$\begin{aligned}
&\langle \nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t), \boldsymbol{h}_t - \boldsymbol{x}_t\rangle \\
&\quad \le \langle \nabla_{\boldsymbol{x}}\mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t\rangle \qquad (5.33) \\
&\quad = \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t\rangle + \langle \boldsymbol{y}_t + \beta_t^{-1}(\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t), \mathcal{A}\boldsymbol{x}_\star - \mathcal{A}\boldsymbol{x}_t + \boldsymbol{v}_t - \boldsymbol{v}_t\rangle \qquad (5.34) \\
&\quad = \langle \nabla f(\boldsymbol{x}_t), \boldsymbol{x}_\star - \boldsymbol{x}_t\rangle - \beta_t^{-1}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 - \langle \boldsymbol{y}_t, \mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\rangle \\
&\qquad\qquad + \langle \boldsymbol{y}_t + \beta_t^{-1}(\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t), \mathcal{A}\boldsymbol{x}_\star - \boldsymbol{v}_t\rangle \qquad (5.35) \\
&\quad \le f_\star - f(\boldsymbol{x}_t) - \beta_t^{-1}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 - \langle \boldsymbol{y}_t, \mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\rangle \\
&\qquad\qquad + \langle \boldsymbol{y}_t + \beta_t^{-1}(\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t), \mathcal{A}\boldsymbol{x}_\star - \boldsymbol{v}_t\rangle \qquad (5.36) \\
&\quad = \mathcal{L}_\star - \mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) - \frac{1}{2\beta_t}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 + \langle \boldsymbol{y}_t + \beta_t^{-1}(\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t), \mathcal{A}\boldsymbol{x}_\star - \boldsymbol{v}_t\rangle \qquad (5.37) \\
&\quad \le \mathcal{L}_\star - \mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) - \frac{1}{2\beta_t}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2. \qquad (5.38)
\end{aligned}$$

The first inequality holds because $\boldsymbol{h}_t$ is the solution of the linear minimization oracle. The second inequality simply follows from the convexity of f. Finally, the last inequality comes from the optimality condition. We also use the strong duality assumption to ensure $f_\star = \mathcal{L}_\star$.

By definition of $\boldsymbol{v}_t$ in (5.30), the following holds:

$$\langle \boldsymbol{y}_t + \beta_t^{-1}(\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t),\, \boldsymbol{v} - \boldsymbol{v}_t\rangle = -\langle \nabla_{\boldsymbol{v}} \mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t),\, \boldsymbol{v} - \boldsymbol{v}_t\rangle \le 0, \qquad \forall\, \boldsymbol{v} \in \mathcal{K}, \tag{5.39}$$

and in particular for $\boldsymbol{v} = \mathcal{A}\boldsymbol{x}_\star \in \mathcal{K}$.

Substituting (5.38) back into (5.32), we get

$$\begin{aligned}\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_t, \boldsymbol{y}_t) \le (1-\eta_t)\mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) + \eta_t \mathcal{L}_\star + \left(\beta_{t+1}^{-1} - (1+\eta_t)\beta_t^{-1}\right)\frac{1}{2}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 \\ + \eta_t^2 \frac{\bar{L}_{t+1}}{2} D_{\mathcal{X}}^2 + \eta_t(\beta_{t+1}^{-1} - \beta_t^{-1})\langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t,\, \mathcal{A}\boldsymbol{h}_t - \mathcal{A}\boldsymbol{x}_t\rangle. \end{aligned} \tag{5.40}$$

We define $\bar{\boldsymbol{v}}_{t+1}$ as

$$\bar{\boldsymbol{v}}_{t+1} := \mathrm{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_{t+1} + \beta_{t+1}^{-1}\boldsymbol{y}_t) = \arg\min_{\boldsymbol{v}\in\mathcal{K}} \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}, \boldsymbol{y}_t). \tag{5.41}$$

Hence, we obtain

$$\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \bar{\boldsymbol{v}}_{t+1}, \boldsymbol{y}_t) \le \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_t, \boldsymbol{y}_t) \tag{5.42}$$

$$\le (1-\eta_t)\mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \boldsymbol{v}_t, \boldsymbol{y}_t) + \eta_t \mathcal{L}_\star + \left(\beta_{t+1}^{-1} - (1+\eta_t)\beta_t^{-1}\right)\frac{1}{2}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 + \eta_t^2 \frac{\bar{L}_{t+1}}{2} D_{\mathcal{X}}^2 + \eta_t(\beta_{t+1}^{-1} - \beta_t^{-1})\langle \mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t,\, \mathcal{A}\boldsymbol{h}_t - \mathcal{A}\boldsymbol{x}_t\rangle \tag{5.43}$$

$$\le (1-\eta_t)\mathcal{L}_{\beta_t}(\boldsymbol{x}_t, \bar{\boldsymbol{v}}_t, \boldsymbol{y}_t) + \eta_t \mathcal{L}_\star + \left(\beta_{t+1}^{-1} - (1+\eta_t)\beta_t^{-1}\right)\frac{1}{2}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 + \eta_t^2 \frac{\bar{L}_{t+1}}{2} D_{\mathcal{X}}^2 + \eta_t(\beta_{t+1}^{-1} - \beta_t^{-1})\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|\|\mathcal{A}\|D_{\mathcal{X}}, \tag{5.44}$$

where the last line follows from the Cauchy-Schwarz inequality.

Now, we use the following bound:

$$\left(\beta_{t+1}^{-1} - (1+\eta_t)\beta_t^{-1}\right)\frac{1}{2}\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|^2 + \eta_t(\beta_{t+1}^{-1} - \beta_t^{-1})\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|\|\mathcal{A}\|D_{\mathcal{X}} \le \frac{\eta_t^2(\beta_{t+1}^{-1} - \beta_t^{-1})^2}{2\left((1+\eta_t)\beta_t^{-1} - \beta_{t+1}^{-1}\right)}\|\mathcal{A}\|^2 D_{\mathcal{X}}^2, \tag{5.45}$$

which is obtained by considering the maximum value of the second order polynomial in terms of $\|\mathcal{A}\boldsymbol{x}_t - \boldsymbol{v}_t\|$.

Remark that,

$$\frac{\eta_t^2(\beta_{t+1}^{-1}-\beta_t^{-1})^2}{2\left((1+\eta_t)\beta_t^{-1}-\beta_{t+1}^{-1}\right)} \le \beta_0^{-1}\frac{\frac{2}{(t+2)^2}\left(\sqrt{t+3}-\sqrt{t+2}\right)^2}{(1+\frac{2}{t+2})\sqrt{t+2}-\sqrt{t+3}} \le \beta_0^{-1}\frac{\frac{1}{2(t+2)^3}}{(1+\frac{2}{t+2})\sqrt{t+2}-\sqrt{t+3}}$$
$$\le \beta_0^{-1}\frac{\frac{1}{2(t+2)^{2.5}}}{(1+\frac{2}{t+2})(t+2)-(t+2)} \le \frac{1}{4\beta_0(t+2)^{2.5}}. \quad (5.46)$$

Combining all these bounds and subtracting $\mathcal{L}_\star$ from both sides, we end up with

$$\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\bar{\boldsymbol{v}}_{t+1},\boldsymbol{y}_t)-\mathcal{L}_\star \le (1-\eta_t)(\mathcal{L}_{\beta_t}(\boldsymbol{x}_t,\boldsymbol{v}_t,\boldsymbol{y}_t)-\mathcal{L}_\star)+\eta_t^2\frac{\bar{L}_{t+1}}{2}D_{\mathcal{X}}^2+\frac{\|\mathcal{A}\|^2D_{\mathcal{X}}^2}{4\beta_0(t+2)^{2.5}}. \quad (5.47)$$

In order to obtain a recurrence relation, we use

$$\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\boldsymbol{v}_{t+1},\boldsymbol{y}_{t+1}) \le \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\bar{\boldsymbol{v}}_{t+1},\boldsymbol{y}_{t+1}) \quad (5.48)$$
$$= \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\bar{\boldsymbol{v}}_{t+1},\boldsymbol{y}_t)+\langle \boldsymbol{y}_{t+1}-\boldsymbol{y}_t,\mathcal{A}\boldsymbol{x}_{t+1}-\bar{\boldsymbol{v}}_{t+1}\rangle \quad (5.49)$$
$$= \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\bar{\boldsymbol{v}}_{t+1},\boldsymbol{y}_t)+\sigma_{t+1}\|\mathcal{A}\boldsymbol{x}_{t+1}-\bar{\boldsymbol{v}}_{t+1}\|^2. \quad (5.50)$$

We choose $\sigma_{t+1}\ge 0$ such that

$$\sigma_{t+1}\le 1/\beta_0 \qquad \& \qquad \sigma_{t+1}\|\mathcal{A}\boldsymbol{x}_{t+1}-\bar{\boldsymbol{v}}_{t+1}\|^2 \le \eta_t^2\frac{\bar{L}_{t+1}}{2}D_{\mathcal{X}}^2 \qquad \& \qquad \|\boldsymbol{y}_{t+1}\|\le D_{\mathcal{Y}}. \quad (5.51)$$

Note that σ_{t+1} is well defined, in the sense that there always exists $\sigma_{t+1}\ge 0$ which satisfies all conditions. This is because $\sigma_{t+1}=0$ simultaneously satisfies them in a trivial way.

Therefore, we finally obtain

$$\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\boldsymbol{v}_{t+1},\boldsymbol{y}_{t+1})-\mathcal{L}_\star \le (1-\eta_t)(\mathcal{L}_{\beta_t}(\boldsymbol{x}_t,\boldsymbol{v}_t,\boldsymbol{y}_t)-\mathcal{L}_\star)+\eta_t^2\bar{L}_{t+1}D_{\mathcal{X}}^2+\frac{\|\mathcal{A}\|^2D_{\mathcal{X}}^2}{4\beta_0(t+2)^{2.5}}. \quad (5.52)$$

Applying this inequality recursively, we get

$$\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1},\boldsymbol{v}_{t+1},\boldsymbol{y}_{t+1})-\mathcal{L}^\star$$
$$\le \prod_{j=0}^{t}(1-\eta_j)\left(\mathcal{L}_{\beta_0}(\boldsymbol{x}_0,\boldsymbol{v}_0,\boldsymbol{y}_0)-\mathcal{L}^\star\right)+\sum_{\ell=0}^{t}\left(\eta_\ell^2\bar{L}_{\ell+1}D_{\mathcal{X}}^2+\frac{\|\mathcal{A}\|^2D_{\mathcal{X}}^2}{4\beta_0(\ell+2)^{2.5}}\right)\prod_{j=\ell}^{t}(1-\eta_j) \quad (5.53)$$
$$= \sum_{\ell=0}^{t}\left(\eta_\ell^2\bar{L}_{\ell+1}D_{\mathcal{X}}^2+\frac{\|\mathcal{A}\|^2D_{\mathcal{X}}^2}{4\beta_0(\ell+2)^{2.5}}\right)\prod_{j=\ell}^{t}(1-\eta_j) \quad (5.54)$$
$$\le \sum_{\ell=0}^{t}\left(\eta_\ell^2\bar{L}_{t+1}D_{\mathcal{X}}^2+\frac{\|\mathcal{A}\|^2D_{\mathcal{X}}^2}{4\beta_0(\ell+2)^{2.5}}\right)\prod_{j=\ell}^{t}(1-\eta_j) \quad (5.55)$$

where (5.54) follows since $\eta_0=1$, and (5.55) holds because $\bar{L}_{t+1}\ge\bar{L}_{\ell+1}$ for $\ell=1,2,\dots,t$.

By using

$$\sum_{\ell=0}^{t} \eta_\ell^2 \prod_{j=\ell}^{t} (1-\eta_j) = \sum_{\ell=0}^{t} \frac{4}{(\ell+2)^2} \prod_{j=\ell}^{t} \frac{j}{j+2} = \sum_{\ell=0}^{t} \frac{4}{(\ell+2)^2} \frac{(\ell+1)\ell}{(t+1)(t+2)} \le \frac{4}{t+2}, \quad \text{and} \tag{5.56}$$

$$\sum_{\ell=0}^{t} \frac{1}{(\ell+2)^{2.5}} \prod_{j=\ell}^{t} \frac{j}{j+2} = \sum_{\ell=0}^{t} \frac{1}{(\ell+2)^{2.5}} \frac{(\ell+1)\ell}{(t+1)(t+2)} \le \frac{2}{(t+2)^{1.5}}, \tag{5.57}$$

we get the following bound on the augmented Lagrangian:

$$\mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, \boldsymbol{y}_{t+1}) - \mathcal{L}^\star \le \frac{4}{t+2} D_{\mathcal{X}}^2 \bar{L}_{t+1} + \frac{\|\mathcal{A}\|^2 D_{\mathcal{X}}^2}{2\beta_0 (t+2)^{1.5}} \tag{5.58}$$

$$= D_{\mathcal{X}}^2 \left(\frac{4L_f}{t+2} + \frac{4\|\mathcal{A}\|^2/\beta_0}{\sqrt{t+2}} + \frac{\|\mathcal{A}\|^2/\beta_0}{2(t+2)^{1.5}} \right). \tag{5.59}$$

Next, we translate this bound to the convergence guarantees in the objective residual and feasibility gap.

Convergence in objective residual. We start from the definition of augmented Lagrangian:

$$\begin{aligned} f(\boldsymbol{x}_{t+1}) &= \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, \boldsymbol{y}_{t+1}) + \frac{\beta_{t+1}}{2} \|\boldsymbol{y}_{t+1}\|^2 - \frac{1}{2\beta_{t+1}} \operatorname{dist}^2 \left(\mathcal{A}\boldsymbol{x}_{t+1} + \beta \boldsymbol{y}_{t+1}, \mathcal{K} \right) \\ &\le \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, \boldsymbol{y}_{t+1}) + \frac{\beta_{t+1} D_{\mathcal{Y}}^2}{2}. \end{aligned} \tag{5.60}$$

Then, we subtract $f_\star$ from both sides (recall that $f_\star = \mathcal{L}_\star$ due to strong duality)

$$\begin{aligned} f(\boldsymbol{x}_{t+1}) - f_\star &\le \mathcal{L}_{\beta_{t+1}}(\boldsymbol{x}_{t+1}, \boldsymbol{v}_{t+1}, \boldsymbol{y}_{t+1}) - \mathcal{L}^\star + \frac{\beta_{t+1} D_{\mathcal{Y}}^2}{2} \\ &\le D_{\mathcal{X}}^2 \left(\frac{4L_f}{t+2} + \frac{4\|\mathcal{A}\|^2/\beta_0}{\sqrt{t+2}} + \frac{\|\mathcal{A}\|^2/\beta_0}{2(t+2)^{1.5}} \right) + \frac{D_{\mathcal{Y}}^2 \beta_0}{2\sqrt{t+2}}. \end{aligned} \tag{5.61}$$

This completes the proof of the upper bound.

Now, by using the classical Lagrange saddle point theory, we get

$$f_\star \le \mathcal{L}(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{y}_\star) = f(\boldsymbol{x}) + \langle \boldsymbol{y}_\star, \mathcal{A}\boldsymbol{x} - \boldsymbol{v} \rangle \le f(\boldsymbol{x}) + \|\boldsymbol{y}_\star\| \|\mathcal{A}\boldsymbol{x} - \boldsymbol{v}\|, \qquad \forall (\boldsymbol{x}, \boldsymbol{v}) \in \mathcal{X} \times \mathcal{K}. \tag{5.62}$$

In particular, by choosing $\boldsymbol{x} = \boldsymbol{x}_{t+1}$ and $\boldsymbol{v} = \operatorname{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_{t+1})$, and rearranging, we obtain

$$f(\boldsymbol{x}_{t+1}) - f_\star \ge -\|\boldsymbol{y}_\star\| \cdot \operatorname{dist}(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}), \tag{5.63}$$

which is our lower bound on the objective residual.

Convergence in feasibility gap. Denote by $\phi_t := D_{\mathcal{X}}^2\left(\frac{4L_f}{t} + \frac{4\|\mathcal{A}\|^2/\beta_0}{\sqrt{t}} + \frac{\|\mathcal{A}\|^2/\beta_0}{2t^{1.5}}\right)$. We combine (5.58) and (5.62) with $\boldsymbol{x} = \boldsymbol{x}_{t+1}$ and $\boldsymbol{v} = \boldsymbol{v}_{t+1}$, and use Cauchy-Schwartz inequality:

$$\begin{aligned}\langle \boldsymbol{y}_{t+1} - \boldsymbol{y}_\star, \mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\rangle + \frac{1}{2\beta_t}\|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\|^2 \le \phi_{t+2} \\ \implies -\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_\star\|\|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\| + \frac{1}{2\beta_t}\|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\|^2 \le \phi_{t+2} \quad (5.64)\end{aligned}$$

This is a second order inequality with respect to $\|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\|$. Solving this inequality we get

$$\begin{aligned}\|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\| &\le \beta_t\left(\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_\star\| + \sqrt{\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_\star\|^2 + \frac{2\phi_{t+2}}{\beta_t}}\right) &(5.65)\\ &\le \beta_t\left(2\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_\star\| + \sqrt{\frac{2\phi_{t+2}}{\beta_t}}\right) &(5.66)\\ &\le \frac{1}{\sqrt{t+2}}\left(2\beta_0\|\boldsymbol{y}_{t+1} - \boldsymbol{y}_\star\| + D_{\mathcal{X}}\sqrt{\frac{8L_f\beta_0}{\sqrt{t+2}} + \left(8 + \frac{1}{t+2}\right)\|\mathcal{A}\|^2}\right). &(5.67)\end{aligned}$$

Finally, the bound on the feasibility gap follows by

$$\begin{aligned}\operatorname{dist}(\mathcal{A}\boldsymbol{x}_{t+1}, \mathcal{K}) &= \|\mathcal{A}\boldsymbol{x}_{t+1} - \dot{\boldsymbol{v}}_{t+1}\| \\ &= \|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1} + \boldsymbol{v}_{t+1} - \dot{\boldsymbol{v}}_{t+1}\| \\ &\le \|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\| + \|\boldsymbol{v}_{t+1} - \dot{\boldsymbol{v}}_{t+1}\| \quad (5.68)\\ &\le \|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\| + \|\mathcal{A}\boldsymbol{x}_{t+1} - \mathcal{A}\boldsymbol{x}_{t+1} + \beta_{t+1}\boldsymbol{y}_{t+1}\| \\ &\le \|\mathcal{A}\boldsymbol{x}_{t+1} - \boldsymbol{v}_{t+1}\| + D_{\mathcal{Y}}\beta_{t+1},\end{aligned}$$

where $\dot{\boldsymbol{v}}_{t+1} = \operatorname{proj}_{\mathcal{K}}(\mathcal{A}\boldsymbol{x}_{t+1})$, and the fourth line follows from the non-expansiveness.

6 Low-Rank Matrix Sketching from Streaming Data

Our storage-optimal convex optimization paradigm adopts a sketching model as a mechanism for reducing the cost of storing the decision variable. Consequently, designing practical sketching algorithms for low-rank matrix approximation is a fundamental aspect of our research.

This chapter is based on the three joint works with Joel A. Tropp, Madeleine Udell, and Volkan Cevher; [TYUC17a], [TYUC17b], and [TYUC19]. Some details are taken from the preliminary versions [TYUC17c] and [TUC18].

This section mainly consists of some excerpts quoted verbatim with minor modifications from these works, with the aim of providing a brief overview of our main results. These manuscripts are written by Joel A. Tropp, and he is the first author and the primary contact for these works. The author of this dissertation contributed to this research line primarily on the implementation of software and the execution of the computer experiments. Our motivation for studying low-rank matrix approximation from streaming data comes from the specific optimization application [YUTC17].

Introduction

In Chapter 2, we introduced SketchyCGM, the first provable algorithm which produces a rank-r approximate solution to a class of SDP problems with low-rank solutions using optimal storage. CGM forms a sequence of approximate low-rank solutions via the iteration

$$\boldsymbol{X}_1 \leftarrow \mathbf{0} \quad \text{and} \quad \boldsymbol{X}_{t+1} \leftarrow (1-\eta_t)\boldsymbol{X}_t + \eta_t \boldsymbol{u}_t \boldsymbol{u}_t^*, \qquad \text{for some } \eta_t \in [0,1]. \tag{6.1}$$

SketchyCGM only maintains a random low-dimensional linear image of the evolving solution $\boldsymbol{X}_t$. When the iteration terminates, it computes a rank-r approximation from this linear image.

This chapter presents a framework for computing structured low-rank approximations of a matrix from a *sketch*, a randomized low-dimensional linear image of the matrix. Suppose that $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ (or $\mathbb{C}^{m\times n}$) is an arbitrary matrix. Let r be a target rank parameter where

$r \ll \min\{m, n\}$. The computational problem is to produce a low-rank approximation $\hat{\boldsymbol{X}}$ of $\boldsymbol{X}$ whose error is comparable to a best rank-r approximation:

$$\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}} \approx \min_{\mathrm{rank}(\boldsymbol{B}) \le r} \|\boldsymbol{X} - \boldsymbol{B}\|_{\mathrm{F}}. \tag{6.2}$$

Contributions

Our aim is to construct a random sketching map that acquires enough information to solve the low-rank matrix approximation problem (6.2). We must also design algorithms for computing a reconstruction $\hat{\boldsymbol{X}}$ from the sketch without direct access to the matrix $\boldsymbol{X}$ itself. We can summarize the main features of our approach and the contributions as follows:

∘ We design sketching maps that minimizes the size of the sketch (and the test matrices) to achieve approximations that satisfy (6.2).

∘ We propose novel reconstruction algorithms that are *reliable*, *numerically stable*, and *efficient* in terms of storage and computation.

∘ We present a systematic mechanism to maintain structural properties of the input matrix in our approximations, such as symmetry or positive-semi-definiteness.

∘ We present explicit *a priori* error bounds for the algorithms so that we can determine a sketch size that suffices to achieve a specific approximation goal.

∘ Our precise error bounds with respect to the tail energy of the input matrix offers concrete guidance on the best sketch-size parameters. (details excluded in this dissertation, see Section 4.5 in [TYUC17b] and Section 5.4 in [TYUC19])

∘ We document numerical experiments on real and synthetic data to demonstrate that our methods dominate existing techniques. Our codes are available online[1] as a MATLAB toolbox.

∘ Our toolkit includes an *a posteriori* error estimator for validating the quality of the approximation. This estimator also provides a principled mechanism for selecting the precise rank of the final approximation. (details excluded in this dissertation, see Section 6 in [TYUC19])

∘ We demonstrate an application of our approach as a tool to compress large-scale scientific data from several scientific simulations and measurement processes. (details excluded in this dissertation, see Section 7 in [TYUC19])

This chapter overviews only the key features of our results. We omit all proofs and technical details in this short summary. We refer to the original papers for more details.

[1] Available at "https://github.com/alpyurtsever/SKETCH"

Preliminaries

The methods apply for the real field ($\mathbb{F} = \mathbb{R}$) and for the complex field ($\mathbb{F} = \mathbb{C}$). We introduce a parameter α that reflect the field over which we are working:

$$\alpha := \alpha(\mathbb{F}) := \begin{cases} 1, & \mathbb{F} = \mathbb{R} \\ 0, & \mathbb{F} = \mathbb{C} \end{cases} \tag{6.3}$$

Let us also introduce a function that we use to simplify many of our error bounds:

$$\omega(u, v) := \frac{u}{v - u - \alpha} \quad \text{for integers that satisfy } v > u + \alpha > \alpha. \tag{6.4}$$

Observe that the function $\omega(u, \cdot)$ is decreasing, with range $(0, u]$.

Standard normal matrix. A matrix $\boldsymbol{G} \in \mathbb{R}^{m\times n}$ has the real standard normal distribution if the entries form an independent family of standard normal random variables (i.e., Gaussian with mean zero and variance one). A matrix $\boldsymbol{G} \in \mathbb{C}^{m\times n}$ has the complex standard normal distribution if it has the form $\boldsymbol{G} = \boldsymbol{G}_1 + \mathrm{i}\boldsymbol{G}_2$ where $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ are independent, real standard normal matrices. Standard normal matrices are also known as Gaussian matrices.

Throughout, we present the analysis of the case where the linear dimension reduction maps (test matrices) are statistically independent and follow the standard normal distribution. We describe other potential distributions for the test matrices in Section 3.9 of [TYUC17b] and Section 3 of [TYUC19].

6.1 Two Component Sketch

This section presents a brief summary of our approach introduced in [TYUC17b].

6.1.1 The Sketch

To form the sketch of the target matrix, we independently draw and fix two test matrices:

$$\boldsymbol{\Omega} \in \mathbb{F}^{n\times k} \quad \text{and} \quad \boldsymbol{\Psi} \in \mathbb{F}^{\ell\times m}. \tag{6.5}$$

The sketch of the target matrix $\boldsymbol{X} \in \mathbb{F}^{m\times n}$ consists of two matrices:

$$\boldsymbol{Y} := \boldsymbol{X}\boldsymbol{\Omega} \in \mathbb{F}^{m\times k} \quad \text{and} \quad \boldsymbol{W} := \boldsymbol{\Psi}\boldsymbol{X} \in \mathbb{F}^{\ell\times n}. \tag{6.6}$$

The matrix $\boldsymbol{Y}$ collects information about the action of $\boldsymbol{X}$, while the matrix $\boldsymbol{W}$ collects information about the action of $\boldsymbol{X}^*$.

6.1.2 The Basic Reconstruction Algorithm

The first step in the reconstruction is to compute an orthobasis $\boldsymbol{Q}$ for the range of $\boldsymbol{Y}$ by means of an orthogonal–triangular factorization:

$$\boldsymbol{Y} =: \boldsymbol{Q}\boldsymbol{R} \quad \text{where} \quad \boldsymbol{Q} \in \mathbb{F}^{m \times k}. \tag{6.7}$$

The second step uses the co-range sketch $\boldsymbol{W}$ to form the matrix

$$\boldsymbol{B} := (\boldsymbol{\Psi}\boldsymbol{Q})^{\dagger}\boldsymbol{W} \in \mathbb{F}^{k \times n}. \tag{6.8}$$

Then, we report the rank-k approximation

$$\hat{\boldsymbol{X}} := \boldsymbol{Q}\boldsymbol{B} \in \mathbb{F}^{m \times n} \quad \text{where} \quad \boldsymbol{Q} \in \mathbb{F}^{m \times k} \quad \text{and} \quad \boldsymbol{B} \in \mathbb{F}^{k \times n}. \tag{6.9}$$

Intuition. To motivate the algorithm, we recall a familiar heuristic from randomized linear algebra (see Section 1 in [HMT11]), which states

$$\boldsymbol{X} \approx \boldsymbol{Q}\boldsymbol{Q}^{*}\boldsymbol{X}. \tag{6.10}$$

Although we would like to form the rank-k approximation $\boldsymbol{Q}(\boldsymbol{Q}^{*}\boldsymbol{X})$, we cannot compute the factor $\boldsymbol{Q}^{*}\boldsymbol{X}$ without revisiting the target matrix $\boldsymbol{X}$. Instead, we exploit the information in the co-range sketch $\boldsymbol{W} = \boldsymbol{\Psi}\boldsymbol{X}$. Notice that

$$\boldsymbol{W} = \boldsymbol{\Psi}(\boldsymbol{Q}\boldsymbol{Q}^{*}\boldsymbol{X}) + \boldsymbol{\Psi}(\boldsymbol{X} - \boldsymbol{Q}\boldsymbol{Q}^{*}\boldsymbol{X}) \approx (\boldsymbol{\Psi}\boldsymbol{Q})(\boldsymbol{Q}^{*}\boldsymbol{X}). \tag{6.11}$$

The heuristic (6.10) justifies dropping the second term. Multiplying on the left by the pseudoinverse $(\boldsymbol{\Psi}\boldsymbol{Q})^{\dagger}$, we arrive at the relation

$$\boldsymbol{B} = (\boldsymbol{\Psi}\boldsymbol{Q})^{\dagger}\boldsymbol{W} \approx \boldsymbol{Q}^{*}\boldsymbol{X}. \tag{6.12}$$

These considerations suggest that

$$\hat{\boldsymbol{X}} = \boldsymbol{Q}\boldsymbol{B} \approx \boldsymbol{Q}\boldsymbol{Q}^{*}\boldsymbol{X} \approx \boldsymbol{X}. \tag{6.13}$$

This explanation is inspired by the discussion in Section 5.5 in [HMT11].

Theorem 6.1. *Assume the sketch size parameters satisfy $k > r + \alpha$ and $\ell > k + \alpha$. Draw random test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n \times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell \times m}$ independently from the standard normal distribution. Then the rank-k approximation $\hat{\boldsymbol{X}}$ obtained from formula* (6.9) *satisfies*

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}}^{2} \leq \big(1 + \omega(r,k)\big) \cdot \big(1 + \omega(k,\ell)\big) \cdot \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_{r}\|_{\mathrm{F}}^{2}. \tag{6.14}$$

In particular, the selection $k = 2r + \alpha$ and $\ell = 2k + \alpha$ yields the error bound

$$\left(\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}}^2\right)^{1/2} \leq 2 \cdot \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}}. \tag{6.15}$$

We recommend this parameter choice because it achieves a good tradeoff between the size of the sketch and the accuracy of the reconstruction.

Similar bounds hold with high probability.[2]

Remark. We can extend the reconstruction error bound in Theorem 6.1, reformulating the bound with respect to tail energy instead of the optimal rank-r error $\|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}}$. This provides a good description of the actual behavior. As a consequence, we can offer a concrete guidance on the sketch size parameters based on the spectral decay of the input matrix. See Theorem 4.3 and Section 4.5 in [TYUC17b] for more details.

6.1.3 The Fixed-Rank Reconstruction Algorithm

Suppose that we wish to compute a rank-r approximation of the target matrix $\boldsymbol{X} \in \mathbb{F}^{m\times n}$. First, we form an initial rank-k approximation $\hat{\boldsymbol{X}} := \boldsymbol{QB}$ using the procedure (6.9). Then we obtain a rank-r approximation $[\![\hat{\boldsymbol{X}}]\!]_r$ of the target matrix by replacing $\hat{\boldsymbol{X}}$ with its best rank-r approximation in Frobenius norm:

$$[\![\hat{\boldsymbol{X}}]\!]_r = [\![\boldsymbol{QB}]\!]_r = (\boldsymbol{QU})[\![\boldsymbol{\Sigma}]\!]_r \boldsymbol{V}^* = \boldsymbol{Q}[\![\boldsymbol{B}]\!]_r, \quad \text{where} \quad \boldsymbol{B} = \boldsymbol{U\Sigma V}^* \text{ is an SVD of } \boldsymbol{B}. \tag{6.16}$$

Theorem 6.2. *Assume the sketch size parameters satisfy $k > r + \alpha$ and $\ell > k + \alpha$. Draw random test matrices $\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}$ and $\boldsymbol{\Psi} \in \mathbb{F}^{\ell\times m}$ independently from the standard normal distribution. Then the rank-r approximation $[\![\hat{\boldsymbol{X}}]\!]_r$ obtained from the formula eq.* (6.16) *satisfies*

$$\mathbb{E}\,\|\boldsymbol{X} - [\![\hat{\boldsymbol{X}}]\!]_r\|_{\mathrm{F}} \leq \sqrt{1 + \omega(r,k)} \cdot \left(1 + 2\sqrt{\omega(k,\ell)}\right) \cdot \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}}. \tag{6.17}$$

In particular, the selection $k = 2r + \alpha$ and $\ell = 2k + \alpha$ yields the error bound

$$\mathbb{E}\,\|\boldsymbol{X} - [\![\hat{\boldsymbol{X}}]\!]_r\|_{\mathrm{F}} \leq 3\sqrt{2} \cdot \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}}. \tag{6.18}$$

Similar results hold with high probability.

6.1.4 Computing a PSD Approximation

Suppose that the target matrix $\boldsymbol{X}$ belongs to the closed, convex set $\mathcal{X} \subset \mathbb{F}^{m\times n}$. Let $\hat{\boldsymbol{X}}_{\mathrm{in}} \in \mathbb{F}^{m\times n}$ be an initial approximation of $\boldsymbol{X}$. We can produce a new approximation $\mathrm{proj}_{\mathcal{X}}(\hat{\boldsymbol{X}}_{\mathrm{in}})$ by projecting the initial approximation onto the constraint set. This procedure always improves the approximation quality. This result is well known in convex analysis.

[2] The expectation bounds in this and the following results also describe the typical behavior because of measure concentration effects. One can develop high-probability bounds using the methods from Section 10.3 in [HMT11].

We often encounter the problem of approximating a PSD (or conjugate symmetric) matrix. Projection onto PSD cone can be efficiently performed on the factors, avoiding forming large matrices. See Section 5 in [TYUC17b] for the details.

Last of all, consider the situation where we need to approximate a structured matrix by a structured matrix with fixed rank. The most direct approach is to replace an initial structured approximation by the nearest structured rank-r matrix. Projecting a PSD (respectively, conjugate symmetric) matrix onto the set of fixed-rank matrices preserves PSD (the conjugate symmetric property). The set of matrices with a fixed rank is not convex, however, so the previous analysis does not apply. To this end, we provide the following general bound for fixed-rank approximation with structure. This bound is typically somewhat loose.

Theorem 6.3. *Let $\boldsymbol{X} \in \mathbb{F}^{m\times n}$ be a target matrix, and let $\hat{\boldsymbol{X}}_{\text{in}} \in \mathbb{F}^{m\times n}$ be an approximation. For any rank parameter r,*

$$\|\boldsymbol{X} - [\![\hat{\boldsymbol{X}}_{\text{in}}]\!]_r\|_{\text{F}} \le \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\text{F}} + 2\|\boldsymbol{X} - \hat{\boldsymbol{X}}_{\text{in}}\|_{\text{F}}. \tag{6.19}$$

6.2 Three Component Sketch

This section presents the key features of our new sketching approach with three components [TYUC19]. The new algorithm is a hybrid of the methods from [Upa16] and our previous approach with two components. The new algorithm performs better than its predecessors.

6.2.1 The Sketch

Independently, draw and fix four randomized linear dimension reduction maps:

$$\begin{aligned} &\boldsymbol{\Upsilon} \in \mathbb{F}^{k\times m} \quad \text{and} \quad \boldsymbol{\Omega} \in \mathbb{F}^{k\times n}; \\ &\boldsymbol{\Phi} \in \mathbb{F}^{s\times m} \quad \text{and} \quad \boldsymbol{\Psi} \in \mathbb{F}^{s\times n}, \end{aligned} \tag{6.20}$$

such that $r_0 \le k \le s \le \min\{m, n\}$, where r_0 is the target rank.

The sketch itself consists of three matrices:

$$\boldsymbol{W} := \boldsymbol{\Upsilon}\boldsymbol{X} \in \mathbb{F}^{k\times n}, \quad \boldsymbol{Y} := \boldsymbol{X}\boldsymbol{\Omega}^* \in \mathbb{F}^{m\times k}, \quad \text{and} \quad \boldsymbol{Z} := \boldsymbol{\Phi}\boldsymbol{X}\boldsymbol{\Psi}^* \in \mathbb{F}^{s\times s}. \tag{6.21}$$

The first two matrices $(\boldsymbol{W}, \boldsymbol{Y})$ capture the co-range and the range of $\boldsymbol{X}$. The core sketch $(\boldsymbol{Z})$ contains fresh information that improves our estimates of the singular values and singular vectors of $\boldsymbol{X}$; it is responsible for the superior performance of the new method.

6.2.2 Computing Truncated Low-Rank Approximations

The first two components $(\boldsymbol{W}, \boldsymbol{Y})$ of the sketch are used to estimate the co-range and the range of the matrix $\boldsymbol{X}$. Compute thin QR factorizations:

$$\begin{aligned} \boldsymbol{W}^* &=: \boldsymbol{P}\boldsymbol{R}_1 \quad \text{where} \quad \boldsymbol{P} \in \mathbb{F}^{n\times k}; \\ \boldsymbol{Y} &=: \boldsymbol{Q}\boldsymbol{R}_2 \quad \text{where} \quad \boldsymbol{Q} \in \mathbb{F}^{m\times k}. \end{aligned} \tag{6.22}$$

The third sketch $\boldsymbol{Z}$ is used to compute the core approximation $\boldsymbol{B}$, which describes how $\boldsymbol{X}$ acts between range$(\boldsymbol{P})$ and range$(\boldsymbol{Q})$:

$$\boldsymbol{B} := (\boldsymbol{\Phi}\boldsymbol{Q})^{\dagger}\boldsymbol{Z}((\boldsymbol{\Psi}\boldsymbol{P})^{\dagger})^* \in \mathbb{F}^{k\times k}. \tag{6.23}$$

This step is implemented by solving a family of least-squares problems. Next, form a rank-k approximation $\hat{\boldsymbol{X}}$ of the input matrix $\boldsymbol{X}$ via

$$\hat{\boldsymbol{X}} := \boldsymbol{Q}\boldsymbol{B}\boldsymbol{P}^* \tag{6.24}$$

We refer to $\hat{\boldsymbol{X}}$ as the "initial" approximation. The initial approximation can contain spurious information. To produce an approximation that is fully reliable, we must truncate the rank of the initial approximation (6.24). For a truncation parameter r, we construct a rank-r approximation by replacing $\hat{\boldsymbol{X}}$ with its best rank-r approximationin Frobenius norm:

$$[\![\hat{\boldsymbol{X}}]\!]_r = \boldsymbol{Q}[\![\boldsymbol{B}]\!]_r\boldsymbol{P}^*. \tag{6.25}$$

We refer to $[\![\hat{\boldsymbol{X}}]\!]_r$ as a "truncated" approximation.

Remark. We can form other structured approximations of $\boldsymbol{X}$ by projecting $\hat{\boldsymbol{X}}$ onto a set of structured matrices as in Section 6.1.4. We omit the details.

6.2.3 Analysis of Initial and Truncated Approximations

Theorem 6.4 (Initial Approximation: Error Bound)**.** *Let $\boldsymbol{X} \in \mathbb{F}^{m\times n}$ be an arbitrary input matrix. Assume the sketch size parameters satisfy $s \geq 2k + \alpha$. Draw independent Gaussian dimension reduction maps $(\boldsymbol{\Upsilon}, \boldsymbol{\Omega}, \boldsymbol{\Phi}, \boldsymbol{\Psi})$, as in* (6.20)*. Extract a sketch* (6.21) *of the input matrix. Then the rank-k approximation $\hat{\boldsymbol{X}}$, constructed in* (6.24)*, satisfies the error bound*

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}}^2 \leq \big(1 + \omega(k,s)\big)\cdot\big(1 + 2\omega(r,k)\big)\cdot\|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}}^2. \tag{6.26}$$

In particular, $k = 4r_0 + \alpha$ and $s = 2k + \alpha$ yields

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{\mathrm{F}}^2 \leq \tfrac{10}{3}\cdot\|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}}^2. \tag{6.27}$$

Similar bounds hold with high probability.

Remark. In Theorem 6.4, we have imposed the condition $s \geq 2k + \alpha$ because theoretical analysis and empirical work both suggest that the restriction is useful in practice. The approximation (6.24) only requires that $k \leq s$.

Remark. We can extend the reconstruction error bound in Theorem 6.4, reformulating the bound with respect to the tail energy, in order to provide a concrete guidance on the sketch size parameters based on the spectral decay of the input matrix. See Theorem 5.1 and Section 5.4 in [TYUC19] for more details.

Corollary 6.1 (Truncated Approximation: Error Bound)**.** Instate the assumptions of Theorem 6.4. Then the rank-r approximation $[\![\hat{\boldsymbol{X}}]\!]_r$ satisfies the error bound

$$\mathbb{E}\,\|\boldsymbol{X} - [\![\hat{\boldsymbol{X}}]\!]_r\|_{\mathrm{F}} \leq \left(1 + 2\sqrt{\big(1+\omega(k,s)\big)\cdot\big(1+2\omega(r,k)\big)}\right)\cdot\|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{\mathrm{F}} \tag{6.28}$$

Similar results hold with high probability.

This statement is an immediate consequence of Theorem 6.4 and Theorem 6.3.

6.3 Nyström Sketch for PSD Matrices

This section presents a brief summary of our Nystöm Sketch for fixed-rank approximation of a positive-semidefinite (PSD) matrix in [TYUC17a].

6.3.1 The Sketch

Suppose that $\boldsymbol{X} \in \mathbb{F}^{n\times n}$ is a PSD matrix. Fix a sketch size parameter k in the range $r \leq k \leq n$. Independent from $\boldsymbol{X}$, we draw and fix a random test matrix

$$\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}. \tag{6.29}$$

The sketch of the matrix $\boldsymbol{X}$ takes the form

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}. \tag{6.30}$$

To find a good approximation, we must set the sketch size k larger than r. But storage costs and computation also increase with k. One of our main contributions is to clarify the role of k.

Intuition. The Nyström method is a general technique for low-rank PSD matrix approximation. Various instantiations appear in the papers [WS00, Pla05, FBCM04, DM05, HMT11, Git11, CD13, Git13, GM16, LLS+17].

Given the test matrix $\boldsymbol{\Omega}$ and the sketch $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega}$, the Nyström method constructs a rank-k PSD approximation of the PSD matrix $\boldsymbol{X}$ via

$$\hat{\boldsymbol{X}}^{\mathrm{nys}} = \boldsymbol{Y}(\boldsymbol{\Omega}^*\boldsymbol{Y})^\dagger \boldsymbol{Y}^*. \tag{6.31}$$

In most work on the Nyström method, the test matrix $\boldsymbol{\Omega}$ depends adaptively on $\boldsymbol{X}$, so these approaches are not valid in the streaming setting. Gittens's framework [Git11, Git13, GM16] covers the streaming case.

Fixed-Rank Nyström Approximation: Prior Art. To construct a Nyström approximation with exact rank r from a sketch of size k, the standard approach is to truncate the center

$$\hat{\boldsymbol{X}}_r^{\mathrm{nysfix}} = \boldsymbol{Y}(\llbracket\boldsymbol{\Omega}^*\boldsymbol{Y}\rrbracket_r)^\dagger \boldsymbol{Y}^*. \tag{6.32}$$

The truncated Nyström approximation (6.32) appears in the many papers, including [Pla05, DM05, CD13, GM13]. We have found that the truncation method (6.32) performs poorly in the present setting. This observation motivated us to search for more effective techniques.

Fixed-Rank Nyström Approximation: Proposal. Our purpose is to develop, analyze, and evaluate a new approach for fixed-rank approximation of a PSD matrix under the streaming model. We propose a more intuitive rank-r approximation:

$$\hat{\boldsymbol{X}}_r = \llbracket\hat{\boldsymbol{X}}^{\mathrm{nys}}\rrbracket_r. \tag{6.33}$$

That is, we report a best rank-r approximation of the full Nyström approximation (6.31).

6.3.2 Fixed-Rank Nyström Approximation

We outline a numerically stable and very accurate implementation of (6.33), based on an idea from [Tyg14, LLS+17]. In Section 6.4, we show that a poor implementation may produce an approximation with 100% error!

Fix a small parameter $\nu > 0$. Instead of approximating the PSD matrix $\boldsymbol{X}$ directly, we approximate the shifted matrix $\boldsymbol{X}_\nu = \boldsymbol{X} + \nu\mathbf{I}$ and then remove the shift. Here are the steps:

1. Construct the shifted sketch $\boldsymbol{Y}_\nu = \boldsymbol{Y} + \nu\boldsymbol{\Omega}$.
2. Form the matrix $\boldsymbol{B} = \boldsymbol{\Omega}^*\boldsymbol{Y}_\nu$.
3. Compute a Cholesky decomposition $\boldsymbol{B} = \boldsymbol{C}\boldsymbol{C}^*$.
4. Compute $\boldsymbol{E} = \boldsymbol{Y}_\nu\boldsymbol{C}^{-1}$ by back-substitution.
5. Compute the (thin) singular value decomposition $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$.
6. Form $\hat{\boldsymbol{X}}_r = \boldsymbol{U}\llbracket\boldsymbol{\Sigma}^2 - \nu\mathbf{I}\rrbracket_r\boldsymbol{U}^*$.

Related, but distinct, methods were proposed by Williams & Seeger [WS00] and analyzed in Gittens's thesis [Git13].

Theorem 6.5. *Assume* $1 \le r < k \le n$. *Let* $\boldsymbol{X} \in \mathbb{F}^{n\times n}$ *be a PSD matrix. Draw a test matrix* $\boldsymbol{\Omega} \in \mathbb{F}^{n\times k}$ *from the Gaussian distribution, and form the sketch* $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega}$. *Then the approximation* $\hat{\boldsymbol{X}}_r$ *given by* (6.31) *and* (6.33) *satisfies*

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}_r\|_{S_1} \le \big(1 + \omega(r,k)\big) \cdot \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{S_1}; \tag{6.34}$$

$$\mathbb{E}\,\|\boldsymbol{X} - \hat{\boldsymbol{X}}_r\|_{S_\infty} \le \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{S_\infty} + \omega(r,k) \cdot \|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_{S_1}. \tag{6.35}$$

Similar results hold with high probability.

6.4 Numerical Experiments

This section presents computer experiments that are designed to evaluate the performance of the proposed sketching algorithms for low-rank matrix approximation. We include comparisons with alternative methods from the literature.

6.4.1 Sketching and Reconstruction Methods in our Experiments

We use the following algorithms in our numerical experiments:

∘ The [HMT11] method in Section 5.5, Remark 5.4 of [HMT11] is a simplification of the method from Section 5.2 of Woolfe et al. [WLRT08], and they perform similarly. There are two sketches, and the sketch size depends on one parameter k. The total storage cost $T = k(m+n)$.

∘ The [TYUC17b] method is our two component sketch, implemented as described in Algorithm 7 of [TYUC17b]. It involves two sketches, controlled by two parameters k, ℓ. The total storage cost $T = km + \ell n$.

∘ The [Upa16] method in Section 3.3 of [Upa16] simplifies a complicated approach from Theorem 12 in Boutsidis et al. [BWZ16]. This algorithm involves three sketches, controlled by two parameters k, s. The total storage cost $T = k(m+n) + s^2$.

∘ The [TYUC2019] method in is our three component sketching approach, implemented as described in Algorithm 4.4 of [TYUC19]. It uses three sketches, controlled by two parameters k, s. The total storage cost $T = k(m+n) + s^2$.

For our tests on fixed-rank reconstruction of a PSD in Section 6.4.5, we use:

∘ The standard Nyström approximation (6.32), and the proposed approach (6.33), (implemented as described in Algorithm 3 of [TYUC17a]). These methods involve one sketch, controlled by a single parameters k. The total storage cost $T = kn$.

∘ The [TYUC17b] method is our two component sketch, implemented for PSD matrices as described in Algorithm 9 of [TYUC17b]. It involves two sketches, controlled by two parameters k, ℓ. The total storage cost $T = (k+\ell)n$.

6.4.2 Experimental Setup

Fix an input matrix $\boldsymbol{X} \in \mathbb{F}^{n\times n}$ and a truncation rank r. Select sketch size parameters. For each trial, draw dimension reduction maps and form the sketch of the input matrix. Compute a rank-r approximation $\hat{\boldsymbol{X}}_{\text{out}}$ using a specified reconstruction algorithm. The approximation error is calculated relative to the best rank-r approximation error in Schatten p-norm:

$$S_p \text{ relative error} \quad = \quad \frac{\|\boldsymbol{X} - \hat{\boldsymbol{X}}_{\text{out}}\|_p}{\|\boldsymbol{X} - [\![\boldsymbol{X}]\!]_r\|_p} - 1. \tag{6.36}$$

We perform 20 independent trials and report the average error.

6.4.3 Classes of Input Matrices

We consider several different types of synthetic and real input matrices. See Figure 6.1 for a plot of the spectra of these input matrices.

Synthetic Examples. We work over the complex field $\mathbb{C}$. The matrix dimensions $m = n = 10^3$, and we introduce an effective rank parameter R. We compute an approximation with truncation rank $r = 10$.

∘ **Low-rank + noise:** Let $\xi \geq 0$ be a signal-to-noise parameter. These matrices take the form

$$\boldsymbol{X} = \operatorname{diag}(\underbrace{1,\dots,1}_{R},0,\dots,0) + \xi n^{-1}\boldsymbol{C} \in \mathbb{C}^{n\times n}, \tag{6.37}$$

where $\boldsymbol{C} = \boldsymbol{G}\boldsymbol{G}^*$ for a standard normal matrix $\boldsymbol{G} \in \mathbb{F}^{n\times n}$.
`LowRankLowNoise` ($\xi = 10^{-4}$), `LowRankMedNoise` ($\xi = 10^{-2}$), `LowRankHiNoise` ($\xi = 10^{-1}$)

∘ **Polynomial decay:** For a decay parameter $p > 0$, consider matrices

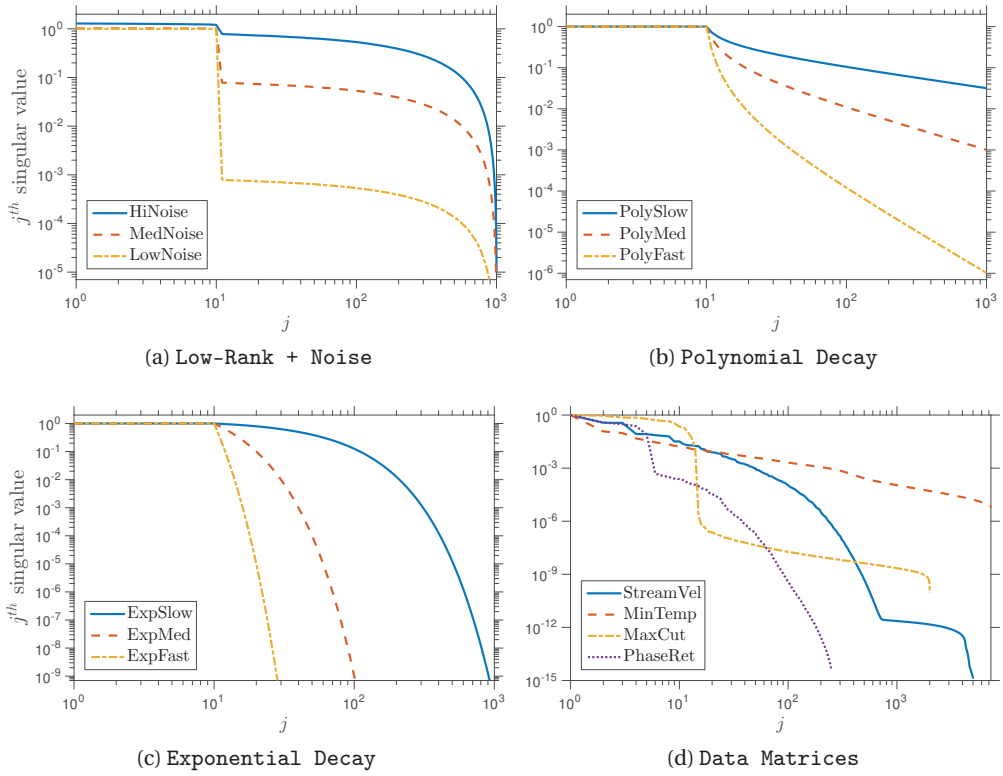
$$\boldsymbol{X} = \operatorname{diag}(\underbrace{1,\dots,1}_{R},2^{-p},3^{-p},\dots,(n-R+1)^{-p}) \in \mathbb{C}^{n\times n}. \tag{6.38}$$

`PolyDecaySlow` ($p = 0.5$), `PolyDecayMed` ($p = 1$), `PolyDecayFast` ($p = 2$).

∘ **Exponential decay:** For a decay parameter $q > 0$, consider matrices

$$\boldsymbol{X} = \operatorname{diag}(\underbrace{1,\dots,1}_{R},10^{-q},10^{-2q},\dots,10^{-(n-R)q}) \in \mathbb{C}^{n\times n}. \tag{6.39}$$

`ExpDecaySlow` ($q = 0.01$), `ExpDecayMed` ($q = 0.1$), `ExpDecayFast` ($q = 0.5$)



(a) `Low-Rank + Noise` (b) `Polynomial Decay`

(c) `Exponential Decay` (d) `Data Matrices`

Figure 6.1 – Singular value spectrum for each of the synthetic and real data classes.

Application Examples. We present some low-rank data matrices that arise in applications.

∘ **Navier–Stokes:** This data is courtesy of Beverley McKeon and Sean Symon. The real $m \times n$ matrix `StreamVel` contains streamwise velocities at $m = 10,738$ points for each of $n = 5,001$ time instants.

∘ **Weather:** This data is courtesy of William North. The real $m \times n$ matrix `MinTemp` contains the minimum temperature recorded at $m = 19,264$ stations on each of $n = 7,305$ days, across the northeastern United States during the years 1981–2016.

We also consider matrices that arise in our optimization problems:

∘ **MaxCut**: This is a real psd matrix with $m = n = 2,000$ that gives a high-accuracy solution to the max-cut SDP for a sparse graph [GW95].

∘ **PhaseRetrieval**: This is a complex psd matrix with $m = n = 25,921$ that gives a low-accuracy solution to a phase retrieval SDP [HCO+15].

6.4.4 Comparison of Formulas for General Low-Rank Matrices

Figure 6.2 presents the results of the following experiment. For synthetic matrices with effective rank $R = 10$ and truncation rank $r = 10$, we compare the relative error eq. (6.36) achieved by each of the four algorithms as a function of storage. We use Gaussian dimension reduction maps in these experiments.

Figure 6.3 contains the results of the following experiment. For each of the four algorithms, we display the relative error eq. (6.36) as a function of storage. We use sparse dimension reduction maps, see Section 3.3 in [TYUC19] for the details.

We plot the oracle error[3] attained by each method. Since the oracle error is not achievable in practice, we also chart the performance of each method at the *a priori* parameter selection.

The numerical work here supports the superiority of the proposed methods.

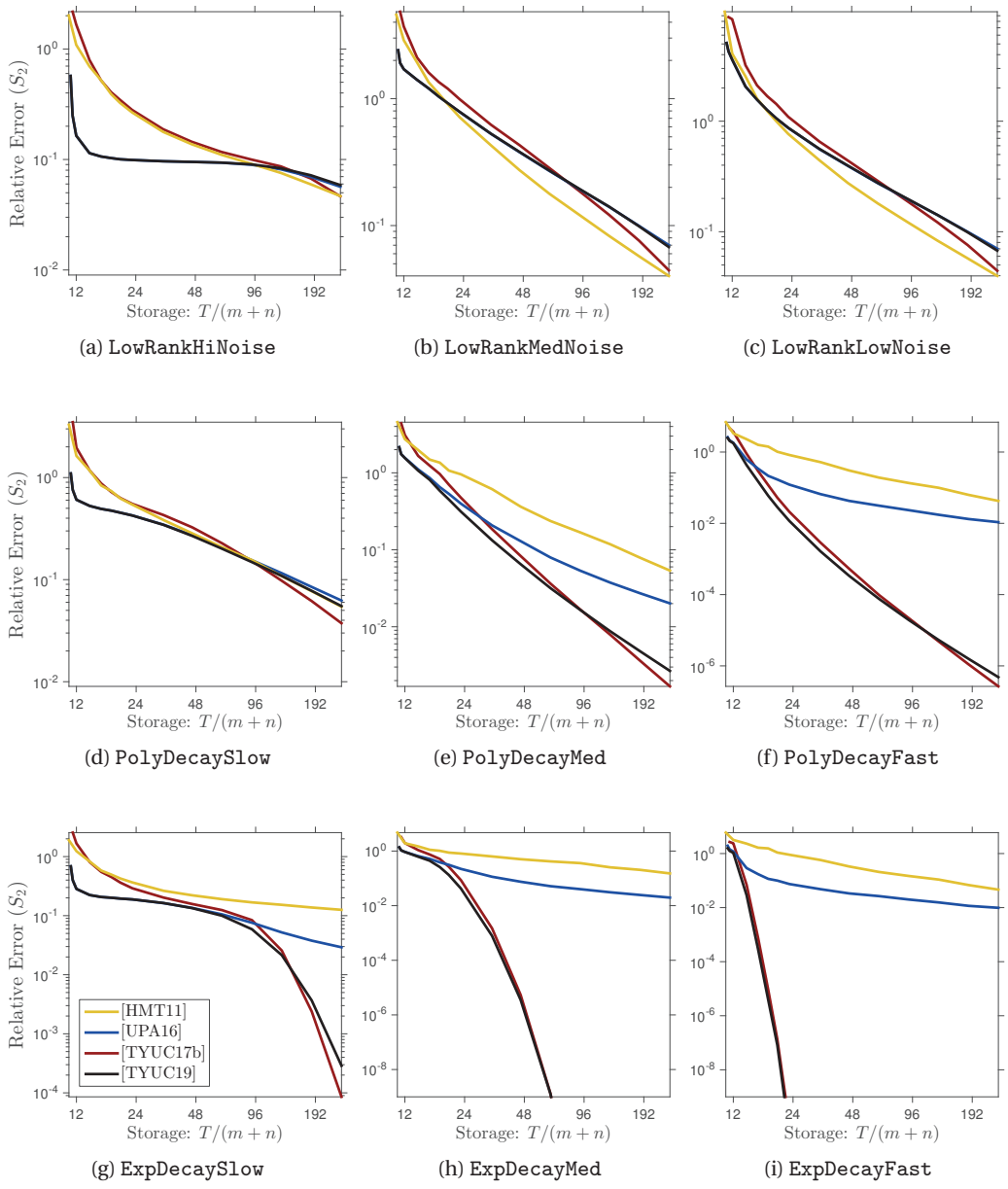
6.4.5 Comparison of Formulas for Low-Rank PSD Matrices

Figures 6.4 and 6.5 display the performance of the three fixed-rank psd approximation methods. The vertical axis is the Schatten 1-norm relative error (6.36). The variable T is the storage required for the sketch only. For the Nyström-based approximations (6.32) and (6.33), we have the correspondence $T/n = k$. For Algorithm 9 in [TYUC17b], we have $T/n = k + \ell$.

Figure 6.6 gives evidence about the numerical challenges involved in implementing Nyström approximations, such as (6.33). Our implementation is based on the Nyström approximation routine `eigenn` released by Tygert [Tyg14] to accompany the paper [LLS+17]. We compare with another implementation strategy described in the text of the same paper, Eqn. (13) in [LLS+17]. It is surprising to discover very different levels of precision in two implementations designed by professional numerical analysts.

The experiments demonstrate that the proposed method (6.33) has a significant benefit over the alternatives for input matrices that admit a good low-rank approximation.

[3] To make fair comparisons among algorithms, we can fix the storage budget and identify the parameter choices that minimize the (average) relative error incurred over the repeated trials. We refer to the minimum as the oracle error for an algorithm. The oracle error is not attainable in practice.

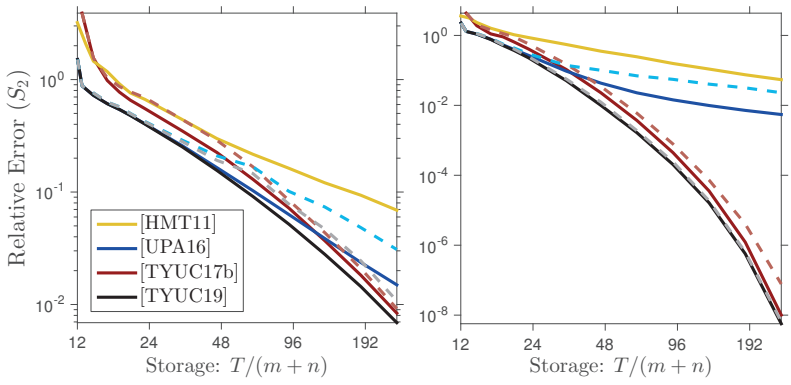


(a) `LowRankHiNoise` (b) `LowRankMedNoise` (c) `LowRankLowNoise`

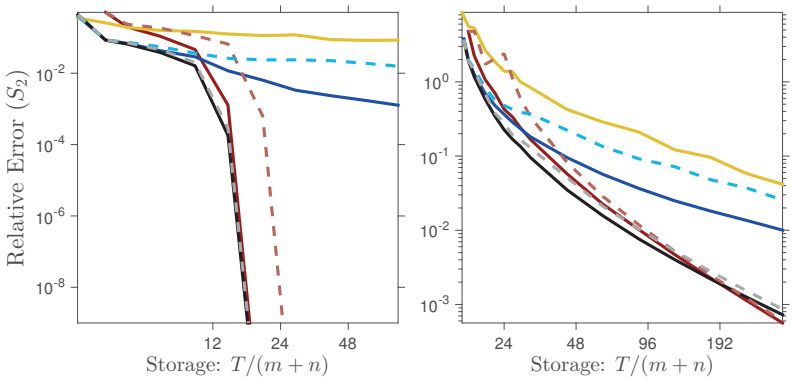
(d) `PolyDecaySlow` (e) `PolyDecayMed` (f) `PolyDecayFast`

(g) `ExpDecaySlow` (h) `ExpDecayMed` (i) `ExpDecayFast`

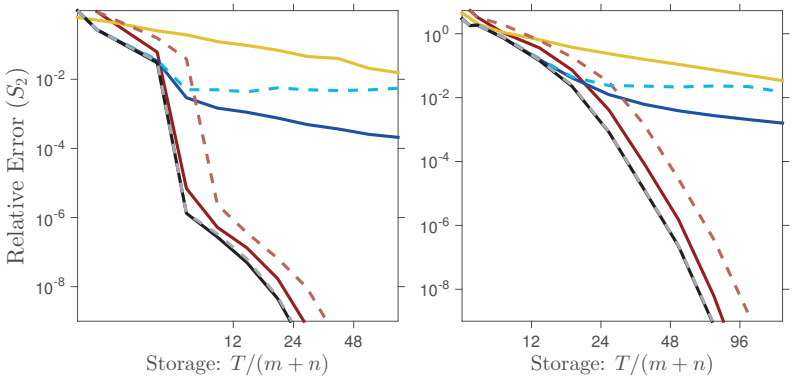
Figure 6.2 – [Comparison of reconstruction formulas: Synthetic examples.] (Gaussian maps, effective rank $R = 10$, approximation rank $r = 10$, Schatten 2-norm.) We compare the oracle error achieved by different fixed-rank approximation methods.



(a) `MinTemp` ($r = 10$) (b) `StreamVel` ($r = 10$)



(c) `MaxCut` ($r = 1$) (d) `MaxCut` ($r = 14$)



(e) `PhaseRetrieval` ($r = 1$) (f) `PhaseRetrieval` ($r = 5$)

Figure 6.3 – [Comparison of reconstruction formulas: Real data examples.] We compare the relative error achieved by different fixed-rank approximation methods. [Solid lines] are oracle errors; [dashed lines] are errors with "natural" parameter choices. (There is no dashed line for [HMT11].) See Section 5.4 in [TYUC19] for the theoretical guidance on sketch-size parameters.

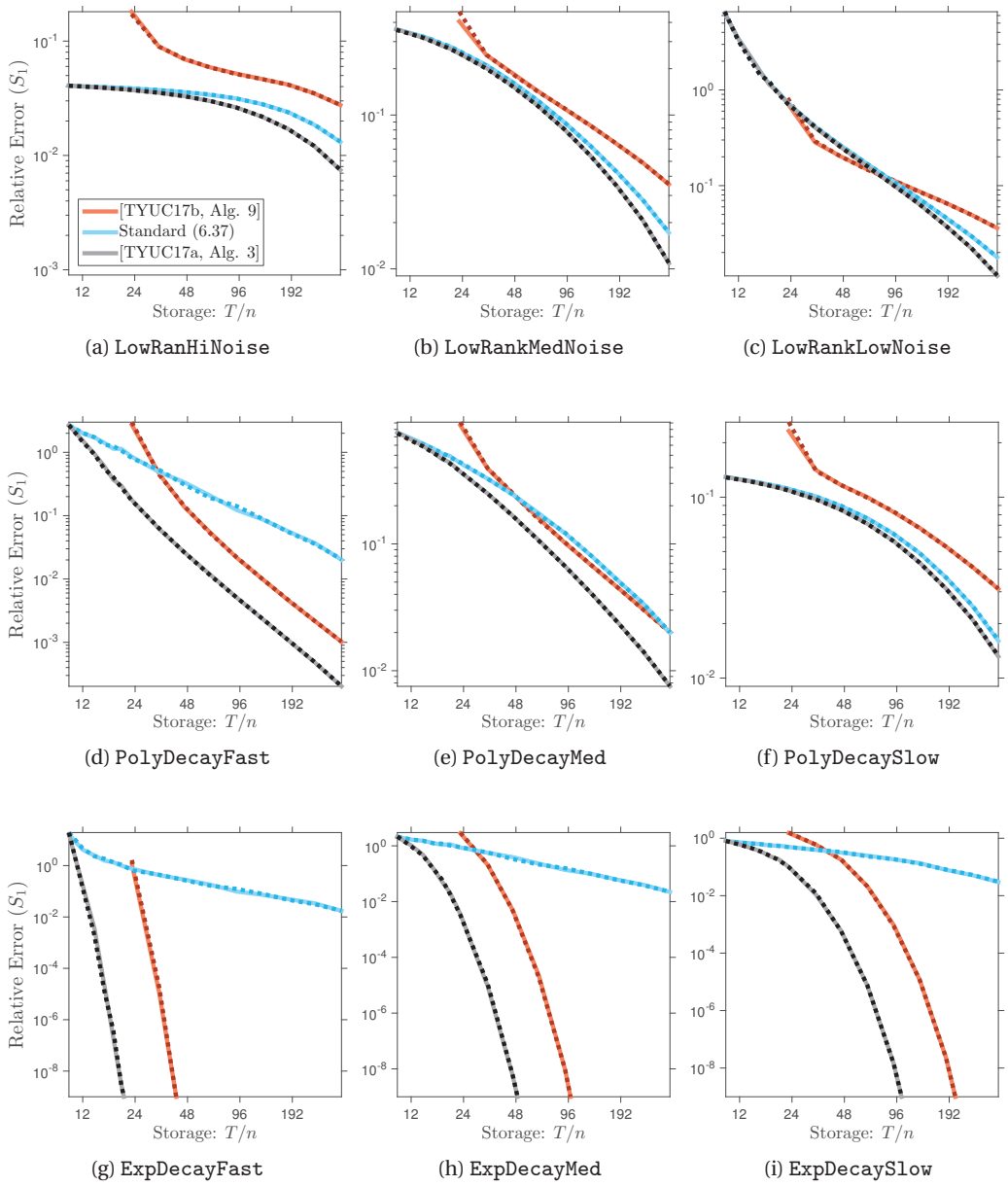
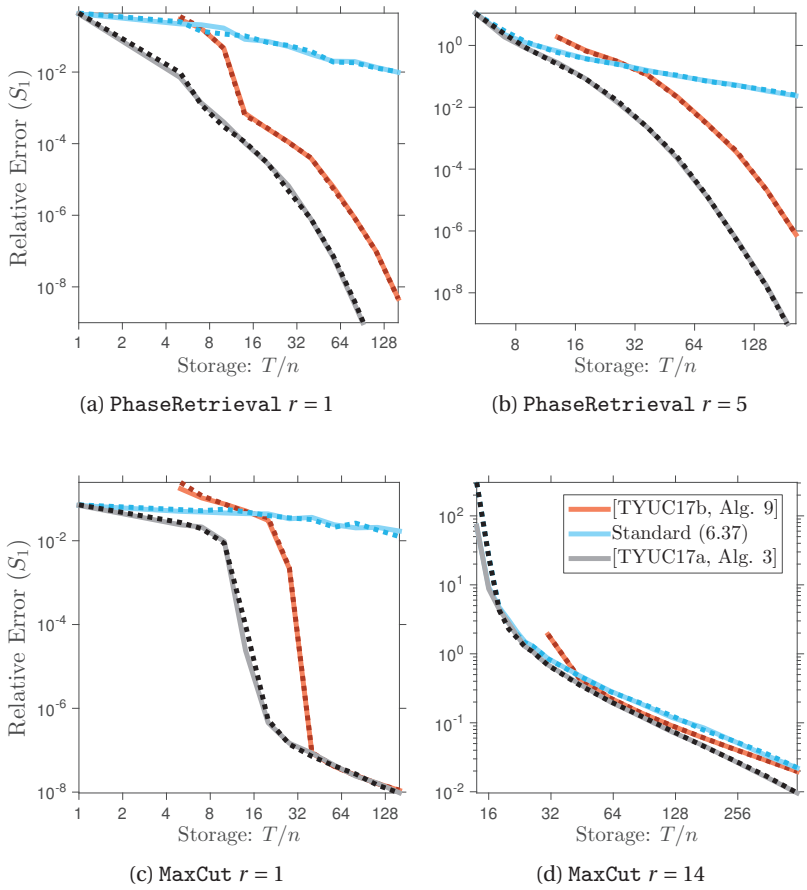


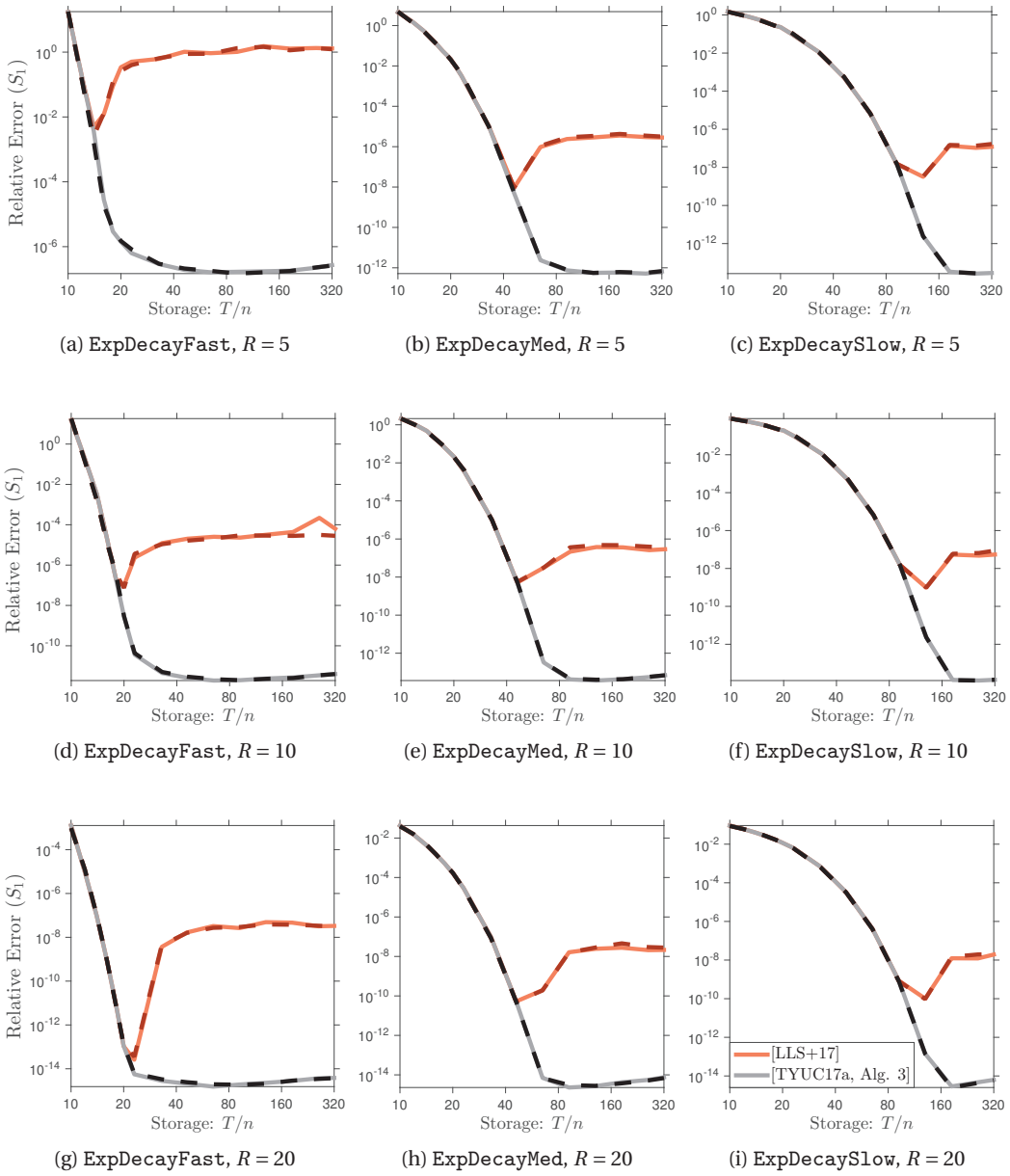
Figure 6.4 – [Comparison of fixed-rank PSD reconstruction formulas: Synthetic examples.] The data series show the performance of three algorithms for rank-r PSD approximation with $r = 10$. [Solid lines] are generated from the Gaussian sketch; [dashed lines] are from the scrambled subsampled randomized Fourier transform (SSRFT) sketch. See Section 3 in [TYUC19] for a discussion on the different randomized linear dimension reduction maps.



(a) `PhaseRetrieval` $r = 1$ (b) `PhaseRetrieval` $r = 5$

(c) `MaxCut` $r = 1$ (d) `MaxCut` $r = 14$

Figure 6.5 – [Comparison of fixed-rank PSD reconstruction formulas: Real data examples.] The data series show the performance of three algorithms for rank-r psd approximation. [Solid lines] are generated from the Gaussian sketch; [dashed lines] are from the scrambled subsampled randomized Fourier transform (SSRFT) sketch. See Section 3 in [TYUC19] for a discussion on different randomized linear dimension reduction maps.



(a) `ExpDecayFast`, $R = 5$ (b) `ExpDecayMed`, $R = 5$ (c) `ExpDecaySlow`, $R = 5$

(d) `ExpDecayFast`, $R = 10$ (e) `ExpDecayMed`, $R = 10$ (f) `ExpDecaySlow`, $R = 10$

(g) `ExpDecayFast`, $R = 20$ (h) `ExpDecayMed`, $R = 20$ (i) `ExpDecaySlow`, $R = 20$

Figure 6.6 – [Numerical stability in implementing Nyström approximation formulas.] The series are generated by two implementations of the fixed-rank PSD approximation (6.33). We compare our implementation with another approach [LLS+17] proposed at Eqn. (13) in [LLS$^+$17]. (Approximation Rank $r = 10$) [Solid lines] are generated from the Gaussian sketch; [dashed lines] are from the scrambled subsampled randomized Fourier transform (SSRFT) sketch. See Section 3 in [TYUC19] for a discussion on different randomized linear dimension reduction maps.

7 CGM with Stochastic Path-Integrated Differential Estimator

So far, we have discussed the problems where data is readily available. Unfortunately, most problems in engineering and machine learning contain randomness to some degree. For instance, the data might be distributed over time and space without being accessible in its entirety at any given time instance.

Extending our results to the stochastic setting is an important piece of future work. This chapter aims to identify and present stochastic CGM variants with the best oracle complexity guarantees as a preliminary study for this direction. The work was done jointly with Suvrit Sra and Volkan Cevher [YSC19].

Introduction

We study two problem settings in this chapter: The stochastic expectation minimization template, and the so-called finite-sum setting which is an important special case of the former.

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad F(\boldsymbol{x}) := \begin{cases} \mathbb{E}_{\boldsymbol{\xi}} f(\boldsymbol{x},\boldsymbol{\xi}) & \text{(expectation)} \\ \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} f_i(\boldsymbol{x}) & \text{(finite-sum)} \end{cases} \tag{7.1}$$

▷ $\mathcal{X} \subset \mathbb{R}^p$ is the convex and compact domain;
▷ F, f and f_i are differentiable (and *possibly non-convex*);
▷ $\boldsymbol{\xi} \sim \mathcal{P}$ is a random variable, supported on $\Xi \subseteq \mathbb{R}^q$.

The expectation objective covers a large number of applications in machine learning and statistics. The finite-sum template frequently arises in M-estimation and empirical risk minimization problems. Template (7.1) can be solved by using the well-known projected stochastic gradient descent method (SGD). At each iteration, SGD takes a stochastic gradient step followed by a projection to ensure feasibility of the new point. However, in many applications, projection onto $\mathcal{X}$ can impose a computational bottleneck (*e.g.,* projection onto the nuclear

norm-ball may require a full singular value decomposition), or it can be even intractable (*e.g.,* dual structural SVMs [LJJSP13]).

There are many applications for stochastic conditional gradient methods, both in convex and non-convex settings. This includes low-rank matrix and tensor factorizations, structured sparse matrix estimation, dictionary learning applications, multi-class classification (considered as a motivating example in [HL16]), constrained deep learning problems (*e.g.,* [RDLS18] presents an application in computer vision), and many more.

To this end, we focus on the theoretical complexity of CGM-type algorithms for stochastic and finite-sum setting, for both convex and non-convex objectives. In the following sections, we identify and present the tightest results known so far.

Contributions

∘ We propose a class of novel variance-reduced stochastic optimization algorithms, based on the recent *stochastic path-integrated differential estimator* technique (SPIDER) [FLLZ18]. By combining SPIDER with CGM, we introduce SPIDER-FW (FW stands for Frank-Wolfe).

∘ We extend our framework with the conditional gradient sliding (CGS) technique [LZ16], and propose SPIDER-CGS. This method further reduces the stochastic first-order oracle complexity of SPIDER-FW in various settings.

∘ For the non-convex setting, we present a new and compact proof for CGS along with its extension to the stochastic setting. To the best of our knowledge, our analysis is the first to provide convergence results for CGS in terms of the FW-gap (FW-gap is a widely used measure for the convergence of CGM-type methods, see Section 7.2). Although CGS does not seem to provide any improvement upon FWin this setup, we use the proof technique to extend SPIDER-CGS for the non-convex settings

∘ Finally, we use a learning rate schedule which depends on the global iteration counter, which in turn enables us to provide induction-free proofs. This approach differs from the majority of variance reduced FW methods, as they rely on an induction with respect to the outer loop counter, along with a sufficient improvement condition for each epoch.

Notation. We use the notation $[n] = \{1, 2, \dots, n\}$.

7.1 Related Works

CGM literature in the stochastic setting is much younger compared to the projection-based stochastic gradient methods. We can trace it back to a variant for online learning proposed by Hazan & Kale [HK12]. More recently, Hazan & Luo [HL16] introduced stochastic FW methods with and without variance reduction for finite-sum problems. Very recently, Mokhtari et al. [MHK18] proposed an alternative scheme for the expectation minimization setting.

CGM variants for non-convex stochastic learning are relatively understudied, with the most of the known results being due to Reddi et al. [RSPS16]. We discuss more details on the theoretical aspects of all these CGM variants in Section 7.5.

7.1.1 Conditional Gradient Sliding

Lan & Zhou [LZ16] have recently developed the conditional gradient sliding (CGS) method, which uses a modified version of the Nesterov's acceleration scheme [Nes83] where the projection subproblems are solved inexactly using CGM. In other words, CGS establishes the convergence of an inexact version of the accelerated gradient method. CGS has superior first-order oracle complexity compared to CGM, although they have the same *lmo* complexity. We discuss more details and variants of CGS in Section 7.5.

7.1.2 Stochastic Path-Integrated Differential Estimator

There has been an extensive research on the variance reduced stochastic optimization methods, in order to address the needs of big data applications in machine learning. Therefore, various variance reduction techniques are proposed in the last few years such as SAG [RSB12], SVRG [JZ13], SAGA [DBLJ14], and more recently SARAH [NLST17], SNVRG [ZXG18], and SPIDER [FLLZ18].

Among these approaches, SARAH and SPIDER are closely related, since they use the same sequential update rule for the gradient estimator $\boldsymbol{v}^k$:

$$\boldsymbol{v}^k = \nabla f_{\mathcal{S}}(\boldsymbol{x}^k) - \nabla f_{\mathcal{S}}(\boldsymbol{x}^{k-1}) + \boldsymbol{v}^{k-1}. \tag{7.2}$$

However, SARAH uses this estimator in the classical gradient descent template, while SPIDER adopts a normalized gradient approach. As a consequence, the results and the analyses differ.

As described by Wang et al. [WJZ+18], the original SPIDER framework has a restrictive step-size (proportional with the target accuracy ε), which makes the algorithm impractical despite its theoretical appeal. Fortunately, this problem does not translate into the CGM-type analysis.

7.2 Preliminaries

As we consider a new template in this chapter, we need to renew some of the definitions we made in Section 1.7, such as the solution set and ε-solution.

Solution. We denote a solution of problem (7.1) by $\boldsymbol{x}_\star$. Similarly, $F_\star$ respectively represents the optimal value.

$$\boldsymbol{x}_\star \in \arg\min_{\boldsymbol{x}\in\mathcal{X}} F(\boldsymbol{x}) \quad \text{and} \quad F_\star = F(\boldsymbol{x}_\star). \tag{7.3}$$

Measure of non-stationarity. For unconstrained non-convex problems, the typical measure of non-stationarity is the gradient norm, because $\|\nabla f(\boldsymbol{x})\| \to 0$ as $\boldsymbol{x}$ converges to a stationary point. However, this measure cannot be used for constrained problems, because $\|\nabla f(\boldsymbol{x})\|$ might not converge to 0 when we approach a solution on the boundary.

Instead, we will use a quantity called FW-gap, which naturally appears in the analysis of CGM-type algorithms.

$$\mathcal{G}(\boldsymbol{x}) := \max_{\boldsymbol{u}\in\mathcal{X}} \langle \boldsymbol{u}-\boldsymbol{x}, -\nabla F(\boldsymbol{x})\rangle. \tag{7.4}$$

The FW-gap is always non-negative, and it gets 0 if and only if we are looking at a stationary point or a solution. As a consequence, FW-gap is a meaningful measure of non-stationarity, and it is widely used for CGM-type algorithms, see [LJ16], [RSPS16] and the references therein.

ε-solution. Due to the fundamental difference in the measure of non-stationarity, we use different definitions of approximate solutions for convex and non-convex problems:
▷ If F is *convex*, we say $\boldsymbol{x}_\varepsilon \in \mathcal{X}$ is an ε-solution if

$$F(\boldsymbol{x}_\varepsilon) - F_\star \le \varepsilon. \tag{7.5}$$

▷ If F is *non-convex*, we say that a random variable $\boldsymbol{x}_\varepsilon$ chosen uniformly from a finite set of points $\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^t\}$ is an ε-solution if

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}_\varepsilon)] \le \varepsilon. \tag{7.6}$$

It is common to provide convergence guarantees in expectation for a randomly chosen iterate in the non-convex setting. See [RSPS16].

Oracle models. We adopt the following black-box oracle model from [RSPS16], to establish a ground for comparing the convergence speed of different algorithms:

- Stochastic first-order oracle (*sfo*)
  For a stochastic function $\mathbb{E}_{\boldsymbol{\xi}} f(\cdot, \boldsymbol{\xi})$ with $\boldsymbol{\xi} \sim \mathcal{P}$, *sfo* returns a pair $(f(\boldsymbol{x}, \boldsymbol{\xi}'), \nabla f(\boldsymbol{x}, \boldsymbol{\xi}'))$ where $\boldsymbol{\xi}'$ is an *iid* sample from $\mathcal{P}$. [NY83]
- Incremental first-order oracle (*ifo*)
  For a finite-sum, *ifo* takes an index $i \in [n]$ and returns $(f_i(\boldsymbol{x}), \nabla f_i(\boldsymbol{x}))$. [AB15]
- Linear minimization oracle (*lmo*)
  Well-known oracle of FW-type methods.

Assumptions (finite-sum). For the finite-sum setting, we assume that $f_i(\boldsymbol{x})$ has an averaged L-Lipschitz gradient:

$$\mathbb{E}\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\|^2 \le L^2\|\boldsymbol{x}-\boldsymbol{y}\|^2, \qquad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^2. \tag{7.7}$$

This also implies F is L-smooth, since

$$\|\nabla F(\boldsymbol{x}) - \nabla F(\boldsymbol{y})\|^2 = \|\mathbb{E}(\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y}))\|^2 \leq \mathbb{E}\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\|^2 \leq L^2\|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{7.8}$$

Assumptions (expectation). For the expectation minimization, we assume that $\nabla f(\boldsymbol{x}, \boldsymbol{\xi})$ is an unbiased estimate of the gradient:

$$\mathbb{E}\nabla f(\boldsymbol{x}, \boldsymbol{\xi}) = \nabla F(\boldsymbol{x}). \tag{7.9}$$

We also assume that the variance is bounded:

$$\mathbb{E}\|\nabla f(\boldsymbol{x}, \boldsymbol{\xi}) - \nabla F(\boldsymbol{x})\|^2 \leq \sigma^2 < \infty, \qquad \forall \boldsymbol{\xi} \in \Xi,\ \forall \boldsymbol{x} \in \mathcal{X}. \tag{7.10}$$

And finally, we assume an averaged L-Lipschitz gradient condition, *i.e.,* the following condition holds $\forall \boldsymbol{\xi} \in \Xi$:

$$\mathbb{E}\|\nabla f(\boldsymbol{x}, \boldsymbol{\xi}) - \nabla f(\boldsymbol{y}, \boldsymbol{\xi})\|^2 \leq L^2\|\boldsymbol{x} - \boldsymbol{y}\|^2, \qquad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^2. \tag{7.11}$$

Similar to the finite-sum, this implies the smoothness of F.

Assumptions (non-convex). Let us denote the initial point by $\bar{\boldsymbol{x}}^1$. Initial suboptimality $F(\bar{\boldsymbol{x}}^1) - F_\star$ arisess in the convergence bounds for the non-convex setting. For notational convenience, we denote an upper bound on this term by $\mathcal{E}$:

$$F(\bar{\boldsymbol{x}}^1) - F_\star \leq \mathcal{E} \tag{7.12}$$

Suppose that $F_\star$ is finite, then there exists a finite $\mathcal{E}$ which satisfies this bound. This is a direct consequence of the smoothness of F and the boundedness of domain.

All these assumptions are mild and frequently used in the analysis of stochastic methods and CGM-type algorithms.

7.3 SPIDER Frank-Wolfe

This section presents SPIDER-FW algorithm and its convergence guarantees under various problem settings.

Our methods have a double loop structure, hence the iterates and the parameters have two different iteration counters t and k, such as $\boldsymbol{x}^{t,k}$. For notational brevity, we drop the first counter when there is no ambiguity, such as $\boldsymbol{x}^k$. Throughout, $s_{t,k}$ denotes the global iteration counter, indexed by the k^{th} iteration of the t^{th} epoch. In our pseudocodes, *draw samples* means *iid* samples for expectation minimization, and uniform selection with replacement in the finite-sum setting.

Algorithm 7.1 SPIDER Frank-Wolfe

Input: $\bar{\boldsymbol{x}}^1 \in \mathcal{X}$
for $t = 1, 2, \ldots, T$ **do**
  Set $\boldsymbol{x}^1 = \bar{\boldsymbol{x}}^t$
  Draw Q_t samples $\mathcal{Q}_t$
  Compute $\boldsymbol{v}^1 = \nabla f_{\mathcal{Q}_t}(\boldsymbol{x}^1)$
  Compute $\boldsymbol{w}^1 = \mathrm{lmo}_{\mathcal{X}}(\boldsymbol{v}^1)$
  Update $\boldsymbol{x}^2 = \boldsymbol{x}^1 + \eta_{t,1}(\boldsymbol{w}^1 - \boldsymbol{x}^1)$
  for $k = 2, 3, \ldots, K_t$ **do**
    Draw $S_{t,k}$ samples $\mathcal{S}_{t,k}$
    Compute $\boldsymbol{v}^k = \nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{x}^k) - \nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{x}^{k-1}) + \boldsymbol{v}^{k-1}$
    Compute $\boldsymbol{w}^k \in \mathrm{lmo}_{\mathcal{X}}(\boldsymbol{v}^k)$
    Update $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \eta_{t,k}(\boldsymbol{w}^k - \boldsymbol{x}^k)$
  end for
  Set $\bar{\boldsymbol{x}}^{t+1} = \boldsymbol{x}^{K_t+1}$
end for

7.3.1 Convex Finite-Sum

We consider SPIDER-FW with

$$K_t = 2^{t-1} \quad \text{for} \quad t = 1, 2, \ldots, T.$$

We choose the sampling parameters and the learning rate parameter

$$S_{t,k} = K_t \qquad \mathcal{Q}_t = [n] \qquad \eta_{t,k} = \frac{2}{s_{t,k}+1} \quad \text{where} \quad s_{t,k} = K_t + k - 1.$$

Theorem 7.1. *Consider the convex finite-sum optimization template, and suppose that the assumptions in Section 7.2 for this template hold. Then, estimate $\boldsymbol{x}^{t,k}$ of SPIDER-FW with the parameter setting described above satisfies*

$$\mathbb{E}[F(\boldsymbol{x}^{t,k})] - F_\star = \mathcal{O}\left(\frac{LD^2}{s_{t,k}}\right) \tag{7.13}$$

Corollary 7.1. The *ifo* and *lmo* complexities of SPIDER-FW for achieving ε-solution in this setting are as follows:

$$\begin{aligned} \#(ifo) &= \mathcal{O}\left(n \ln(\tfrac{LD^2}{\varepsilon}) + \tfrac{L^2D^4}{\varepsilon^2}\right) \\ \#(lmo) &= \mathcal{O}(\tfrac{LD^2}{\varepsilon}) \end{aligned} \tag{7.14}$$

7.3.2 Convex Expectation Minimization

We consider SPIDER-FW with

$$K_t = 2^{t-1} \text{ for } t = 1, 2, \ldots, T.$$

We choose the sampling parameters and the learning rate parameter as

$$S_{t,k} = K_t \qquad Q_t = \lceil \frac{\sigma^2 K_t^2}{5L^2 D^2} \rceil \qquad \eta_{t,k} = \frac{2}{s_{t,k}+1} \text{ where } s_{t,k} = K_t + k - 1.$$

Theorem 7.2. *Consider the convex expectation minimization template, and suppose that the assumptions in Section 7.2 for this template hold. Then, estimate $\boldsymbol{x}^{t,k}$ of SPIDER-FW with the parameter setting described above satisfies*

$$\mathbb{E}[F(\boldsymbol{x}^{t,k})] - F_\star = \mathcal{O}\left(\frac{LD^2}{s_{t,k}}\right) \tag{7.15}$$

Corollary 7.2. The *sfo* and *lmo* complexities of SPIDER-FW for achieving ε-solution in this setting are as follows:

$$\begin{aligned} \#(sfo) &= \mathcal{O}\left(\tfrac{\sigma^2 D^2 + L^2 D^4}{\varepsilon^2}\right) \\ \#(lmo) &= \mathcal{O}\left(\tfrac{LD^2}{\varepsilon}\right) \end{aligned} \tag{7.16}$$

SPIDER-FW has the same asymptotic oracle complexities as SCGS [LZ16] in this setting.

7.3.3 Non-convex Finite-Sum

We consider SPIDER-FW with

$$K_t = K = \lceil \sqrt{n} \rceil.$$

We choose the parameters and the learning rate parameter

$$S_{t,k} = S = \lceil \sqrt{n} \rceil \qquad Q_t = [n] \qquad \eta_{t,k} = \eta = \frac{1}{\sqrt{s_{T,K}}} \text{ where } s_{T,K} = TK.$$

Theorem 7.3. *Consider the non-convex finite-sum template, and suppose that the assumptions in Section 7.2 for this template hold. Denote by $\boldsymbol{x}^{out}$ an iterate $\boldsymbol{x}^{t,k}$ of SPIDER-FW chosen uniformly random over all (t,k) pairs up to (T,K). Then, the following bound on the FW-gap holds:*

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{out})] = \mathcal{O}\left(\frac{\mathcal{E} + LD^2}{\sqrt{s_{T,K}}}\right) \tag{7.17}$$

Although it is impractical to store all estimates until the final iteration, all stochastic methods for the non-convex setting shown in Table 7.1 have this type convergence guarantees (see [RSPS16]). More stringently, [LJ16] shows convergence of the non-convex FW in terms of the running best iterate. In the stochastic setting, however, we cannot keep track of the best estimate, simply because we cannot measure the FW-gap.

Corollary 7.3. The *ifo* and *lmo* complexities of SPIDER-FW for achieving ε-solution in the non-convex finite-sum setting are as follows:

$$\begin{aligned} \#(ifo) &= \mathcal{O}\left(\tfrac{\sqrt{n}}{\varepsilon^2}(\mathcal{E}^2+L^2D^4)\right) \\ \#(lmo) &= \mathcal{O}\left(\tfrac{1}{\varepsilon^2}(\mathcal{E}^2+L^2D^4)\right) \end{aligned} \tag{7.18}$$

SPIDER-FW has better *ifo* complexity than state-of-the-art in the non-convex finite-sum setting, and it improves the dependence on sample size n. See Table 7.1 for comparison.

7.3.4 Non-convex Expectation Minimization

We consider SPIDER-FW with

$$K_t = K = \lceil \sigma/\varepsilon \rceil.$$

We choose the parameters and the learning rate parameter

$$S_{t,k} = S = \lceil \sigma/\varepsilon \rceil \qquad Q_t = Q = \lceil 4(\sigma/\varepsilon)^2 \rceil \qquad \eta_{t,k} = \eta = \frac{1}{\sqrt{s_{T,K}}} \text{ where } s_{T,K} = TK.$$

Theorem 7.4. *Consider the non-convex expectation minimization template, and suppose that the assumptions in Section 7.2 for this template hold. Denote by $\boldsymbol{x}^{out}$ an iterate $\boldsymbol{x}^{t,k}$ of SPIDER-FW chosen uniformly random over all (t,k) pairs up to (T,K). Then, the following bound holds:*

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{out})] = \mathcal{O}\left(\frac{\mathcal{E}+LD^2}{\sqrt{s_{T,K}}}\right) + \frac{\varepsilon}{2}. \tag{7.19}$$

Corollary 7.4. The *sfo* and *lmo* complexities of SPIDER-FW for achieving ε-solution in this setting are as follows:

$$\begin{aligned} \#(sfo) &= \mathcal{O}\left(\tfrac{\sigma}{\varepsilon^3}(\mathcal{E}^2+L^2D^4)\right) \\ \#(lmo) &= \mathcal{O}\left(\tfrac{1}{\varepsilon^2}(\mathcal{E}^2+L^2D^4)\right) \end{aligned} \tag{7.20}$$

Once again, SPIDER-FW enjoys superior *sfo* complexity while maintaining the same *lmo* complexity as its competitors. SVRF was the state-of-the-art with $\mathcal{O}(\varepsilon^{-10/3})$, see [RSPS16].

7.4 SPIDER Conditional Gradient Sliding

This section presents SPIDER-CGS (as shown in Algorithm 7.2) and its convergence guarantees for various settings.

7.4.1 Convex Finite-Sum

We consider SPIDER-CGS with

$$K_t = \lceil 2^{t/2} \rceil \text{ for } t = 1, 2, \ldots, T.$$

We choose the sampling parameters and the learning rate parameter as

$$S_{t,k} = 9K_t s_{t,K_t}^2 \qquad \mathcal{Q}_t = [n] \qquad \gamma_{t,k} = \frac{3}{s_{t,k}+2} \text{ where } s_{t,k} = \sum_{\tau=1}^{t-1} K_\tau + k.$$

Furthermore, we choose the CndG subsolver parameters as

$$\beta_{t,k} = \frac{3}{2} L\gamma_{t,k} \qquad \alpha_{t,k} = \frac{2LD^2}{(s_{t,k}+1)^2}.$$

Theorem 7.5. *Consider the convex finite-sum template, and suppose that the assumptions in Section 7.2 for this template hold. Then, estimate $\boldsymbol{y}^{t,k}$ of SPIDER-CGS with the parameter setting described above satisfies*

$$\mathbb{E}[F(\boldsymbol{y}^{t,k}) - F_\star] = \mathcal{O}\left(\frac{LD^2}{s_{t,k}^2}\right) \tag{7.21}$$

Corollary 7.5. The *ifo* and *lmo* complexities of SPIDER-CGS for achieving ε-solution in this template are as follows:

$$\begin{aligned} \#(ifo) &= \mathcal{O}\left(n \ln\left(\tfrac{LD^2}{\varepsilon}\right) + \tfrac{L^2D^4}{\varepsilon^2}\right) \\ \#(lmo) &= \mathcal{O}(\tfrac{LD^2}{\varepsilon}) \end{aligned} \tag{7.22}$$

Algorithm 7.2 SPIDER Conditional Gradient Sliding

Input: $\bar{\boldsymbol{x}}^1 = \bar{\boldsymbol{y}}^1 \in \mathcal{X}$
for $t = 1, 2, \ldots, T$ **do**
  Set $\boldsymbol{x}^1 = \bar{\boldsymbol{x}}^t$ and $\boldsymbol{y}^1 = \bar{\boldsymbol{y}}^t$
  Update $\boldsymbol{z}^1 = \boldsymbol{y}^1 + \gamma_{t,1}(\boldsymbol{x}^1 - \boldsymbol{y}^1)$
  Draw Q_t samples $\mathcal{Q}_t$
  Compute $\boldsymbol{v}^1 = \nabla f_{\mathcal{Q}_t}(\boldsymbol{z}^1)$
  $\boldsymbol{x}^2 = \textsc{CndG}(\boldsymbol{x}^1, \boldsymbol{v}^1, \alpha_{t,1}, \beta_{t,1})$
  Update $\boldsymbol{y}^2 = \boldsymbol{y}^1 + \gamma_{t,1}(\boldsymbol{x}^2 - \boldsymbol{y}^1)$
  for $k = 2, 3, \ldots, K_t$ **do**
    Update $\boldsymbol{z}^k = \boldsymbol{y}^k + \gamma_{t,k}(\boldsymbol{x}^k - \boldsymbol{y}^k)$
    Draw $S_{t,k}$ samples $\mathcal{S}_{t,k}$
    Compute $\boldsymbol{v}^k = \nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{z}^k) - \nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{z}^{k-1}) + \boldsymbol{v}^{k-1}$
    $\boldsymbol{x}^{k+1} = \textsc{CndG}(\boldsymbol{x}^k, \boldsymbol{v}^k, \alpha_{t,k}, \beta_{t,k})$
    Update $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \gamma_{t,k}(\boldsymbol{x}^{k+1} - \boldsymbol{y}^k)$
  end for
  Set $\bar{\boldsymbol{x}}^{t+1} = \boldsymbol{x}^{K_t+1}$ and $\bar{\boldsymbol{y}}^{t+1} = \boldsymbol{y}^{K_t+1}$
end for

---

function $\boldsymbol{u}^+ = \textsc{CndG}(\boldsymbol{u}, \boldsymbol{v}, \alpha, \beta)$
  Set $\boldsymbol{u}^1 = \boldsymbol{u}$
  for $k = 1, 2, \ldots$ **do**
    Compute $\boldsymbol{w}^k = \operatorname{lmo}_{\mathcal{X}}(\boldsymbol{v} + \beta(\boldsymbol{u}^k - \boldsymbol{u}))$
    Evaluate $\zeta_k = \langle \boldsymbol{v} + \beta(\boldsymbol{u}^k - \boldsymbol{u}), \boldsymbol{u}^k - \boldsymbol{w}^k \rangle$
    if $\zeta_k \le \alpha$ **then**
      break
    end if
    Set $\theta_k = \min\{1, \zeta_k / (\beta \|\boldsymbol{w}^k - \boldsymbol{u}^k\|^2)\}$
    Update $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \theta_k(\boldsymbol{w}^k - \boldsymbol{u}^k)$
  end for
  Set $\boldsymbol{u}^+ = \boldsymbol{u}^k$
end function

7.4.2 Convex Expectation Minimization

We consider SPIDER-CGS with

$$K_t = \lceil 2^{t/2} \rceil \text{ for } t = 1, 2, \ldots, T.$$

We choose the sampling parameters and the learning rate parameter as

$$S_{t,k} = 9K_t s_{t,K_t}^2 \qquad Q_t = \lceil \frac{\sigma^2 s_{t,K_t}^4}{L^2 D^2} \rceil \qquad \gamma_{t,k} = \frac{2}{s_{t,k}+1} \text{ where } s_{t,k} = \sum_{\tau=1}^{t-1} K_\tau + k.$$

Furthermore, we choose the CndG subsolver parameters as

$$\beta_{t,k} = \frac{3}{2} L\gamma_{t,k} \qquad \alpha_{t,k} = \frac{2LD^2}{(s_{t,k}+1)^2}.$$

Theorem 7.6. *Consider the convex expectation minimization template, and suppose that the assumptions in Section 7.2 for this template hold. Then, estimate $\mathbf{y}^{t,k}$ of SPIDER-CGS with the parameter setting described above satisfies*

$$\mathbb{E}[F(\mathbf{y}^{t,k}) - F_\star] = \mathcal{O}\left(\frac{LD^2}{s_{t,k}^2}\right) \tag{7.23}$$

Corollary 7.6. The *sfo* and *lmo* complexities of SPIDER-CGS for achieving ε-solution in convex expectation minimization problems are as follows:

$$\begin{aligned} \#(sfo) &= \mathcal{O}\left(\tfrac{\sigma^2 D^2 + L^2 D^4}{\varepsilon^2}\right) \\ \#(lmo) &= \mathcal{O}(\tfrac{LD^2}{\varepsilon}) \end{aligned} \tag{7.24}$$

7.4.3 Non-convex Finite-Sum

We consider SPIDER-CGS with

$$K_t = K = \lceil \sqrt{n} \rceil.$$

We choose the sampling parameters and the learning rate parameter as

$$S_{t,k} = K \qquad \mathcal{Q}_t = [n] \qquad \gamma_{t,k} = \gamma = \frac{1}{\sqrt{s_{T,K}}} \text{ where } s_{T,K} = TK.$$

Furthermore, we choose the CndG subsolver parameters as

$$\beta_{t,k} = \frac{3}{2} L\gamma \qquad \alpha_{t,k} = LD^2\gamma.$$

Theorem 7.7. *Consider the non-convex finite-sum template, and suppose that the assumptions in Section 7.2 for this template hold. Denote by* $\boldsymbol{y}^{out}$ *an iterate* $\boldsymbol{y}^{t,k}$ *of SPIDER-CGS chosen uniformly random over all* (t,k) *pairs up to* (T,K)*. Then, the following bound on the FW-gap holds:*

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{out})] = \mathcal{O}\left(\frac{\mathcal{E} + LD^2}{\sqrt{s_{T,K}}}\right) \tag{7.25}$$

Corollary 7.7. The *ifo* and *lmo* complexities of SPIDER-CGS for achieving ε-solution in non-convex finite-sum are

$$\begin{aligned} \#(ifo) &= \mathcal{O}\left(\tfrac{\sqrt{n}}{\varepsilon^2}(\mathcal{E}^2 + L^2D^4)\right) \\ \#(lmo) &= \mathcal{O}\left(\tfrac{1}{\varepsilon^2}(\mathcal{E}^2 + L^2D^4)\right) \end{aligned} \tag{7.26}$$

7.4.4 Non-convex Expectation Minimization

We consider SPIDER-CGS with

$$K_t = K = \lceil \sigma/\varepsilon \rceil.$$

We choose the sampling parameters and the learning rate parameter as

$$S_{t,k} = K \qquad Q_t = \lceil 4(\sigma/\varepsilon)^2 \rceil \qquad \gamma_{t,k} = \gamma = \frac{1}{\sqrt{s_{T,K}}} \text{ where } s_{T,K} = TK.$$

Furthermore, we choose the CndG subsolver parameters as

$$\beta_{t,k} = \frac{3}{2}L\gamma \qquad \alpha_{t,k} = LD^2\gamma.$$

Theorem 7.8. *Consider the non-convex expectation minimization template, and suppose that the assumptions in Section 7.2 for this template hold. Denote by* $\boldsymbol{y}^{out}$ *an iterate* $\boldsymbol{y}^{t,k}$ *of SPIDER-CGS chosen uniformly random over all* (t,k) *pairs up to* (T,K)*. Then, the following bound holds:*

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{out})] = \mathcal{O}\left(\frac{\mathcal{E} + LD^2}{\sqrt{s_{T,K}}}\right) + \frac{\varepsilon}{2} \tag{7.27}$$

Corollary 7.8. The *sfo* and *lmo* complexities of SPIDER-CGS for achieving ε-solution in non-convex expectation minimization problems are as follows:

$$\begin{aligned} \#(sfo) &= \mathcal{O}\left(\tfrac{\sigma}{\varepsilon^3}(\mathcal{E}^2 + L^2D^4)\right) \\ \#(lmo) &= \mathcal{O}\left(\tfrac{1}{\varepsilon^2}(\mathcal{E}^2 + L^2D^4)\right) \end{aligned} \tag{7.28}$$

	convex				non-convex			
	finite-sum		expectation		finite-sum		expectation	
	(*ifo*)	(*lmo*)	(*sfo*)	(*lmo*)	(*ifo*)	(*lmo*)	(*sfo*)	(*lmo*)
FW	$\mathcal{O}(n\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-1})$	-	-	$\mathcal{O}(n\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-2})$	-	-
CGS	$\mathcal{O}(n\varepsilon^{-1/2})$	$\mathcal{O}(\varepsilon^{-1})$	-	-	$\mathcal{O}(n\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-2})$	-	-
SFW	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-4})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-4})$	$\mathcal{O}(\varepsilon^{-2})$
SFW-1	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-3})$	-	-	-	-
Online-FW	$\mathcal{O}(\varepsilon^{-4})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-4})$	$\mathcal{O}(\varepsilon^{-2})$	-	-	-	-
SCGS	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-4})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-4})$	$\mathcal{O}(\varepsilon^{-2})$
SVRF / SVFW	$\mathcal{O}(n\ln(\varepsilon^{-1})+\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	-	-	$\mathcal{O}(n+n^{2/3}\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-10/3})$	$\mathcal{O}(\varepsilon^{-2})$
STORC†	$\mathcal{O}(n\ln(\varepsilon^{-1})+\varepsilon^{-3/2})$	$\mathcal{O}(\varepsilon^{-1})$	-	-	-	-	-	-
SPIDER-FW	$\mathcal{O}(n\ln(\varepsilon^{-1})+\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(n^{1/2}\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-2})$
SPIDER-CGS	$\mathcal{O}(n\ln(\varepsilon^{-1})+\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-1})$	$\mathcal{O}(n^{1/2}\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-2})$	$\mathcal{O}(\varepsilon^{-3})$	$\mathcal{O}(\varepsilon^{-2})$

Table 7.1 – Comparison of conditional gradient methods for stochastic optimization. Our contribution is highlighted with red font. See Section 7.5 for more details.
FW [FW56, Jag13] , CGS [LZ16] , SFW [HL16, RSPS16] , SFW-1 [MHK18] , Online-FW [HK12] , SCGS [LZ16] , SVRF / SVFW [HL16, RSPS16] , STORC [HL16]

7.5 Comparison & Discussion

Next, we present an extensive comparison of the theoretical aspects of the CGM-type algorithms. We compile a summary of this comparison in Table 7.1.

7.5.1 Convex Optimization Camp

Batch setting. CGM achieves an ε-solution after $\mathcal{O}(1/\varepsilon)$ iterations. This iteration complexity is optimal for a large class of methods that construct the decision variable through convex combination of *lmo* outputs [Lan14]. CGS, on the other side, enjoys $\mathcal{O}(1/\sqrt{\varepsilon})$ first order oracle complexity while maintaining the optimal $\mathcal{O}(1/\varepsilon)$ *lmo* complexity, by reusing the same gradients over multiple iterations [LZ16].

Stochastic setting. Hazan & Kale [HK12] propose Online-FW for an online-learning setting. As mentioned later by Hazan & Luo [HL16], these results can be translated to the stochastic template, by means of a standard online to stochastic conversion approach. This conversion yields $\mathcal{O}(1/\varepsilon^4)$ *sfo* and $\mathcal{O}(1/\varepsilon^2)$ *lmo* complexities.

Algorithm 7.3 presents the most straightforward extension of CGM for the stochastic setting. This method, SFW, converges with $\mathcal{O}(1/k)$ rate when the sample size $S_k = \Theta(k^2)$ [HL16]. As a result, it provably achieves an ε-solution after $\mathcal{O}(1/\varepsilon^3)$ *sfo* and $\mathcal{O}(1/\varepsilon)$ *lmo* calls.

Lan & Zhou introduce an extension of CGS for the stochastic setting (see Section 3 in [LZ16]). This variant, SCGS, achieves $\mathcal{O}(1/\varepsilon^2)$ *sfo* complexity, while maintaining the optimal $\mathcal{O}(1/\varepsilon)$ *lmo* complexity. Under strong convexity, the *lmo* complexity becomes $\mathcal{O}(1/\varepsilon)$.

Algorithm 7.3 Stochastic Frank-Wolfe

1: **Input:** $\boldsymbol{x}^1 \in \mathcal{X}$
2: **for** $k = 1, 2, \ldots, K$ **do**
3: Draw S_k samples $\mathcal{S}_k$
4: Compute $\boldsymbol{w}^k = \mathrm{lmo}_{\mathcal{X}}(\nabla f_{\mathcal{S}_k}(\boldsymbol{x}^k))$
5: Update $\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + \eta_k(\boldsymbol{w}^k - \boldsymbol{x}^k)$
6: **end for**

Hazan & Luo introduce the stochastic variance reduced Frank-Wolfe method, SVRF [HL16]. This method adopts the variance reduction techniques from [JZ13] and [MZJ13]. SVRF is designed specifically for the finite-sum setting, and it requires $\mathcal{O}(n\ln(1/\varepsilon))$ calls of the full gradient oracle, besides $\mathcal{O}(1/\varepsilon^2)$ *ifo* and $\mathcal{O}(1/\varepsilon)$ *lmo* calls, to get an ε-solution.

In order to improve the *ifo* complexity of the SVRF, Hazan & Luo [HL16] also design a variant based on the CGS. This variant, STORC, also requires $\mathcal{O}(n\ln(1/\varepsilon))$ calls to the full gradient oracle and $\mathcal{O}(1/\varepsilon)$ calls to *lmo*, but it enjoys a reduced number of *ifo* calls, at the order of $\mathcal{O}(1/\varepsilon^{1.5})$. Compared to the SVRF, however, STORC requires an additional assumption on the Lipschitz continuity of F. STORC gets better guarantees under strong convexity assumption, however we omit the details.

Lu & Freund [LF18] propose a stochastic CGM variant with $\mathcal{O}(1/\epsilon)$ *lmo* and $\mathcal{O}(n+1/\epsilon)$ *ifo* complexity for the convex finite-sum setting. However, the proposed method relies on a special structure of the objective function, where f_i are univariate functions of the fitted value $\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle$ for some given data sample $\boldsymbol{a}_i$.

All stochastic FW variants we discussed up to know are based on an increasing mini-batch size. Very recently, Mokhtari et al. [MHK18] have proposed an alternative scheme (SFW-1) for expectation minimization setting, which requires a single *sfo* at each iteration. Nevertheless, SFW-1 has an arguably worse computational complexity compared to SFW, with its $\mathcal{O}(1/\varepsilon^3)$ calls to *sfo* and *lmo*. We emphasize the applications of SFW-1 in submodular maximization, but this is beyond the scope of this dissertation.

For the convex finite-sum setting, SPIDER-FW and SPIDER-CGS share the same complexities as SVRF.

7.5.2 Non-convex Optimization Camp

Batch setting. CGM converges to a stationary point, see Section 2.2 in [Ber99] for details. This early result guarantees an asymptotical convergence, but does not provide any information about the convergence rate. To our knowledge, the first convergence rate for CGM in the non-convex setting is shown by Yu et al. [YZS14]. More recently, Lacoste-Julien has presented a non-asymptotic $\mathcal{O}(1/\sqrt{k})$ rate in FW-gap [LJ16].

Stochastic setting. The majority of the results known in the literature for this setting are derived by Reddi et al. in [RSPS16]. The basic stochastic CGM extension, SFW, provably gets an ε-solution after $\mathcal{O}(1/\varepsilon^4)$ *sfo* and $\mathcal{O}(1/\varepsilon^2)$ *lmo* calls. Equipped with a variance reduction technique, SVRF achieves this accuracy with $\mathcal{O}(1/\varepsilon^{10/3})$ *sfo* and $\mathcal{O}(1/\varepsilon^2)$ *lmo* complexity. In the finite-sum setting, the *ifo* complexity gets $\mathcal{O}(n + n^{2/3}/\varepsilon^2)$. We omit the details of the SAGAFW approach, also present in [RSPS16], due to a technical flaw in its analysis (while telescoping Eq.(14) in page 1249 in [RSPS16]).

Qu et al. [QLX18] show the convergence rate for special instances of CGS and SCGS in the non-convex setting. However, they consider a different convergence criterion based on a proximal gradient mapping rather than the conventional FW-gap. Consequently, their results are incomparable with the rest of the literature. For the fact that we are running a *projection-free* method, the FW-gap is a more natural choice than the projection/proximal gradient norm.

We provide a parameter setting and a compact proof for CGS and SCGS in the supplemental material. Note however this setting simply gets the same guarantees as FW and SFW respectively. Whether CGS can provide improved oracle complexities compared to FW in the non-convex setting is an open problem.

For the non-convex setting, SPIDER-FW and SPIDER-CGS have the same oracle complexities, superior to SVRF (which is the state-of-the-art to our knowledge) for finite-sum and expectation minimization problems.

7.5.3 Results from Concurrent Research

By the time we prepared our manuscript [YSC19], the idea of combining SPIDER with CGM analysis was not explored yet. However, there are a few concurrent works that appeared online while our paper was under review. In this section, we discuss these works.

The recent work by Shen et al. [SFZ+19] is very closely related to our approach. They propose a class of methods based on CGM and various variance reduction techniques for the non-convex finite-sum setting, including the SPIDER-FW. Besides, they also propose extensions that use second-order approximations. Finally, they provide simulation studies to compare empirical performance of different variants. We refer to this paper for a numerical comparison.

Hassani et al. [HKMS19] introduce a novel variance reduced CGM method, but their work focuses primarily on the submodular maximization. Accordingly, they consider a more general expectation minimization template (the so-called non-oblivious setting) where the probability distribution depends on the decision variable $\boldsymbol{x}$ and may change during the optimization procedure. Therefore, the proposed method requires some further assumptions and modifications involving computations with the Hessian approximation. Finally, Zhang et al. [ZCM+19] consider a stochastic CGM approach with SPIDER in the distributed and quantized settings.

Concluding Remarks

We have proposed two novel FW-type methods based on the idea of blending the recent variance reduction technique SPIDER into FW and CGS frameworks. We have shown that the resulting methods enjoy superior oracle complexities in various convex and non-convex optimization templates. Extension of our framework for the strongly convex case is left open. Developing a well-tuned implementation, including one that incorporates parallel optimization, is an important piece of future work.

Appendix: Proofs

Preliminaries

This section presents some known results from the existing literature, key to our analysis, for the sake of completeness.

The following Lemma from [FLLZ18] provides an error bound for the estimator v^k obtained by the SPIDER approach.

Lemma 7.9 (Lemma 1 from [FLLZ18], more specifically Eqn.(A.3) in its supplements)**.** Suppose that $\mathcal{S}_{t,k}$ is a subset that samples $S_{t,k}$ *iid* realizations from the distribution $\mathcal{P}$. Let the stochastic estimator $\nabla f_{\mathcal{S}_{t,k}}$ satisfy the averaged L-Lipschitz gradients condition from Section 7.2. Set the estimator $\boldsymbol{v}^k$ as

$$\boldsymbol{v}^k = \nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{x}^k) - \nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{x}^{k-1}) + \boldsymbol{v}^{k-1}. \tag{7.29}$$

Then, the following bound holds:

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \frac{L^2}{S_{t,k}}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2 + \|\nabla F(\boldsymbol{x}^{k-1}) - \boldsymbol{v}^{k-1}\|^2. \tag{7.30}$$

The next Lemma draws a well-known bound on the variance in terms of the mini-batch size.

Lemma 7.10 (Eqn.(3.5) from [LZ16], or Lemma 2 from [RSPS16])**.** Suppose that $\mathcal{S}_{t,k}$ is a subset that samples $S_{t,k}$ *iid* realizations from the distribution $\mathcal{P}$. Let the stochastic estimator $\nabla f_{\mathcal{S}_{t,k}}$ satisfy the bounded variance condition from Section 7.2. Then, the following bound holds:

$$\mathbb{E}\|\nabla f_{\mathcal{S}_{t,k}}(\boldsymbol{x}) - \nabla F(\boldsymbol{x})\|^2 \leq \frac{\sigma^2}{S_{t,k}} \qquad \forall \boldsymbol{x} \in \mathcal{X}. \tag{7.31}$$

Finally, we recall the convergence guarantees for the CndG procedure of CGS-type methods.

Lemma 7.11 (Similar to Theorem 2.2 part (c) from [LZ16], or more generally Theorem 2 from [Jag13])**.** Remark that CndG procedure simply applies CGM (with exact line-search) for the following projection subproblem:

$$\underset{\boldsymbol{x}\in\mathcal{X}}{\text{minimize}} \quad \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{u} + \frac{1}{\beta}\boldsymbol{v}\|^2. \tag{7.32}$$

The objective function in this subproblem is β-smooth, hence CGM requires at most $\mathcal{O}(\frac{4\beta D^2}{\alpha})$ iterations to satisfy the convergence criterion.

We refer to the associated references for the proofs.

Non-convex Conditional Gradient Sliding

In this section, we prove convergence of a CGS instance, and derive its oracle complexities in the non-convex settings. We also extend our results for the stochastic variant SCGS.

Proof of convergence for non-convex CGS

Algorithm 7.4 Conditional Gradient Sliding

1: **Input:** $\boldsymbol{x}^1 \in \mathcal{X}$
2: **Set:** $\alpha = \gamma LD^2, \quad \beta = \gamma L/2, \quad \gamma = 1/\sqrt{K}$
3: **for** $k = 1, 2, \ldots, K$ **do**
4: Update $\boldsymbol{z}^k = \boldsymbol{y}^k + \gamma(\boldsymbol{x}^k - \boldsymbol{y}^k)$
5: $\boldsymbol{x}^{k+1} = \mathrm{CndG}(\boldsymbol{x}^k, \nabla F(\boldsymbol{z}^k), \alpha, \beta)$
6: Update $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \gamma(\boldsymbol{x}^{k+1} - \boldsymbol{y}^k)$
7: **end for**

Theorem 7.12. *Consider the CGS algorithm with the parameters as described in Algorithm 7.4 (in the batch setting). Denote by $\boldsymbol{y}^{out}$ a random iterate $\boldsymbol{y}^k$ drawn uniformly random over all iterates of the CGS. Then, the following bound holds:*

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{out})] = \frac{\mathcal{E} + 3LD^2}{\sqrt{K}} \tag{7.33}$$

Corollary 7.9. The *ifo* and *lmo* complexities of CGS for achieving an ε-solution in the non-convex minimization setting are

$$\begin{aligned} \#(ifo) &= \mathcal{O}\left((\mathcal{E}^2 + L^2D^4)\frac{n}{\varepsilon^2}\right) \\ \#(lmo) &= \mathcal{O}\left((\mathcal{E}^2 + L^2D^4)\frac{1}{\varepsilon^2}\right) \end{aligned} \tag{7.34}$$

Proof. We start by the Taylor expansion and smoothness:

$$\begin{aligned} F(\boldsymbol{y}^{k+1}) &\le F(\boldsymbol{y}^k) + \langle \nabla F(\boldsymbol{y}^k), \boldsymbol{y}^{k+1} - \boldsymbol{y}^k \rangle + \frac{L}{2}\|\boldsymbol{y}^{k+1} - \boldsymbol{y}^k\|^2 \\ &= F(\boldsymbol{y}^k) + \gamma\langle \nabla F(\boldsymbol{y}^k), \boldsymbol{x}^{k+1} - \boldsymbol{y}^k \rangle + \gamma^2\frac{L}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{y}^k\|^2 \\ &\le F(\boldsymbol{y}^k) + \gamma\langle \nabla F(\boldsymbol{y}^k), \boldsymbol{x}^{k+1} - \boldsymbol{y}^k \rangle + \gamma^2\frac{L}{2}D^2 \\ &= F(\boldsymbol{y}^k) + \gamma\langle \nabla F(\boldsymbol{y}^k), \boldsymbol{w}_\star^k - \boldsymbol{y}^k \rangle + \gamma\langle \nabla F(\boldsymbol{y}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle + \gamma^2\frac{L}{2}D^2 \\ &= F(\boldsymbol{y}^k) - \gamma\mathcal{G}(\boldsymbol{y}^k) + \gamma\langle \nabla F(\boldsymbol{y}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle + \gamma^2\frac{L}{2}D^2 \end{aligned} \tag{7.35}$$

where we define $\boldsymbol{w}_\star^k = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, -\nabla F(\boldsymbol{y}^k) \rangle$.

We can equivalently write this inequality as

$$F(\boldsymbol{y}^{k+1}) \leq F(\boldsymbol{y}^k) - \gamma \mathcal{G}(\boldsymbol{y}^k) + \gamma \langle \nabla F(\boldsymbol{y}^k) - \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle + \gamma \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle + \gamma^2 \frac{L}{2} D^2 \quad (7.36)$$

Focus on the last inner-product term

$$\begin{aligned} \gamma \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle &= \gamma \langle \nabla F(\boldsymbol{z}^k) + \beta(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle - \gamma\beta \langle \boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle \\ &\leq \gamma\alpha - \gamma\beta \langle \boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle \\ &\leq \gamma\alpha + \gamma\beta D^2 \end{aligned} \quad (7.37)$$

where the first inequality follows from the role of α in CndG, and the second one from the Cauchy-Schwarz inequality.

Combining these two inequalities, we obtain

$$\begin{aligned} F(\boldsymbol{y}^{k+1}) &\leq F(\boldsymbol{y}^k) - \gamma \mathcal{G}(\boldsymbol{y}^k) + \gamma \langle \nabla F(\boldsymbol{y}^k) - \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k \rangle + \gamma\alpha + \gamma\beta D^2 + \gamma^2 \frac{L}{2} D^2 \\ &\leq F(\boldsymbol{y}^k) - \gamma \mathcal{G}(\boldsymbol{y}^k) + \gamma D \|\nabla F(\boldsymbol{y}^k) - \nabla F(\boldsymbol{z}^k)\| + \gamma\alpha + \gamma\beta D^2 + \gamma^2 \frac{L}{2} D^2 \\ &\leq F(\boldsymbol{y}^k) - \gamma \mathcal{G}(\boldsymbol{y}^k) + \gamma^2 L D \|\boldsymbol{x}^k - \boldsymbol{y}^k\| + \gamma\alpha + \gamma\beta D^2 + \gamma^2 \frac{L}{2} D^2 \\ &\leq F(\boldsymbol{y}^k) - \frac{1}{\sqrt{K}} \mathcal{G}(\boldsymbol{y}^k) + \frac{3LD^2}{K} \end{aligned} \quad (7.38)$$

Taking expectation of both sides, rearranging, and summing over all iterations, we obtain

$$\frac{1}{\sqrt{K}} \sum_{k=1}^{K} \mathbb{E}[\mathcal{G}(\boldsymbol{y}^k)] \leq F(\bar{\boldsymbol{x}}^1) - \mathbb{E}[F(\boldsymbol{y}^K)] + \sum_{k=1}^{K} \frac{3LD^2}{K} \leq F(\bar{\boldsymbol{x}}^1) - F(\boldsymbol{x}_\star) + 3LD^2 \quad (7.39)$$

Hence, by definition of $\boldsymbol{y}^{\text{out}}$, we get

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{\text{out}})] \leq \frac{F(\bar{\boldsymbol{x}}^1) - F(\boldsymbol{x}_\star)}{\sqrt{K}} + \frac{3LD^2}{\sqrt{K}} = \frac{\mathcal{E} + 3LD^2}{\sqrt{K}} \quad (7.40)$$

This completes the convergence rate proof.

To get ε-solution, we set the number of iterations K_ε such that

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{\text{out}})] \leq \frac{\mathcal{E} + 3LD^2}{\sqrt{K_\varepsilon}} \leq \varepsilon. \quad (7.41)$$

Hence, we can calculate the *lmo* complexity by using Lemma 7.11 as

$$\#(lmo) = \mathcal{O}\left(K_\varepsilon \frac{4\beta D^2}{\alpha}\right) = \mathcal{O}\left((\mathcal{E}^2 + L^2 D^4)\frac{1}{\varepsilon^2}\right) \tag{7.42}$$

which completes the proof. □

Proof of convergence for the non-convex SCGS

Algorithm 7.5 Stochastic Conditional Gradient Sliding

1: **Input:** $\boldsymbol{x}^1 \in \mathcal{X}$
2: **Set:** $\alpha = \gamma L D^2, \quad \beta = \gamma L/2, \quad \gamma = 1/\sqrt{K}$
3: **for** $k = 1, 2, \ldots, K$ **do**
4: Update $\boldsymbol{z}^k = \boldsymbol{y}^k + \gamma(\boldsymbol{x}^k - \boldsymbol{y}^k)$
5: Draw K samples $\mathcal{S}_k$
6: $\boldsymbol{x}^{k+1} = \mathrm{CndG}(\boldsymbol{x}^k, \nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k), \alpha, \beta)$
7: Update $\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \gamma(\boldsymbol{x}^{k+1} - \boldsymbol{y}^k)$
8: **end for**

Theorem 7.13. *Consider the SCGS algorithm with the parameters as described in Algorithm 7.5. Assume that the conditions for the expectation minimization from Section 7.2 hold. Denote by $\boldsymbol{y}^{out}$ a random iterate $\boldsymbol{y}^k$ drawn uniformly random over all iterates of the SCGS. Then, the following bound holds:*

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{out})] = \frac{\mathcal{E} + \sigma D + 3LD^2}{\sqrt{K}} \tag{7.43}$$

Corollary 7.10. The *sfo* and *lmo* complexities of SCGS for achieving ε-solution in the non-convex minimization setting are

$$\#(sfo) = \mathcal{O}\left((\mathcal{E} + \sigma D + LD^2)^4 \frac{1}{\varepsilon^4}\right) \qquad \text{and} \qquad \#(lmo) = \mathcal{O}\left((\mathcal{E} + \sigma D + LD^2)^2 \frac{1}{\varepsilon^2}\right) \tag{7.44}$$

Proof. From (7.35), and similar to (7.36) and (7.37), we can show

$$F(\boldsymbol{y}^{k+1}) \leq F(\boldsymbol{y}^k) - \gamma\mathcal{G}(\boldsymbol{y}^k) + \gamma\langle \nabla F(\boldsymbol{y}^k) - \nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle + \gamma\alpha + \gamma\beta D^2 + \gamma^2 \frac{L}{2} D^2 \tag{7.45}$$

where $\boldsymbol{w}_\star^k = \arg\max_{\boldsymbol{x}\in\mathcal{X}} \langle \boldsymbol{x}, -\nabla F(\boldsymbol{y}^k)\rangle$.

Focusing on the inner-product term, we get the following bound:

$$\begin{aligned}\gamma\langle\nabla F(\boldsymbol{y}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k),\,\boldsymbol{x}^{k+1}-\boldsymbol{w}_\star^k\rangle &\le \gamma\|\nabla F(\boldsymbol{y}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|\|\boldsymbol{x}^{k+1}-\boldsymbol{w}_\star^k\|\\ &\le \gamma D\|\nabla F(\boldsymbol{y}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|\\ &\le \gamma D\|\nabla F(\boldsymbol{y}^k)-\nabla F(\boldsymbol{z}^k)\|+\gamma D\|\nabla F(\boldsymbol{z}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|\\ &\le \gamma LD\|\boldsymbol{y}^k-\boldsymbol{z}^k\|+\gamma D\|\nabla F(\boldsymbol{z}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|\\ &= \gamma^2 LD\|\boldsymbol{x}^k-\boldsymbol{y}^k\|+\gamma D\|\nabla F(\boldsymbol{z}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|\\ &\le \gamma^2 LD^2+\gamma D\|\nabla F(\boldsymbol{z}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|\end{aligned} \tag{7.46}$$

Substituting back and taking expectations, we obtain

$$\begin{aligned}\mathbb{E}[F(\boldsymbol{y}^{k+1})] &\le \mathbb{E}[F(\boldsymbol{y}^k)]-\frac{1}{\sqrt{K}}\mathbb{E}[\mathcal{G}(\boldsymbol{y}^k)]+\frac{D}{\sqrt{K}}\mathbb{E}\|\nabla F(\boldsymbol{z}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|+3\gamma^2LD^2\\ &= \mathbb{E}[F(\boldsymbol{y}^k)]-\frac{1}{\sqrt{K}}\mathbb{E}[\mathcal{G}(\boldsymbol{y}^k)]+\frac{D}{\sqrt{K}}\mathbb{E}\|\nabla F(\boldsymbol{z}^k)-\nabla f_{\mathcal{S}_k}(\boldsymbol{z}^k)\|+\frac{3LD^2}{K}\end{aligned} \tag{7.47}$$

Now, we use Lemma 7.10 with the Jensen's inequality to obtain

$$\mathbb{E}[F(\boldsymbol{y}^{k+1})] \le \mathbb{E}[F(\boldsymbol{y}^k)]-\frac{1}{\sqrt{K}}\mathbb{E}[\mathcal{G}(\boldsymbol{y}^k)]+\frac{\sigma D+3LD^2}{K} \tag{7.48}$$

From here, we follow the same steps as in the proof of CGS and get (7.43).

Then, to achieve an ε-solution, we can calculate *sfo* complexity as

$$\#(sfo)=\sum_{k=1}^{K_\varepsilon}K_\varepsilon=K_\varepsilon^2=\mathcal{O}\left((\mathcal{E}+\sigma D+LD^2)^4\frac{1}{\varepsilon^4}\right) \tag{7.49}$$

Finally, *lmo* complexity can be found using Lemma 7.11

$$\#(lmo)=\mathcal{O}\left(K_\varepsilon\frac{4\beta D^2}{\alpha}\right)=\mathcal{O}\left((\mathcal{E}+\sigma D+LD^2)^2\frac{1}{\varepsilon^2}\right) \tag{7.50}$$

This completes the proof. □

Proofs for SPIDER-FW

Lemma 7.14. Suppose that the assumptions listed in Section 7.2 hold. Then, for $k=1,\dots,K_t$, we have the following bounds:

Convex finite-sum	$\mathbb{E}\|\nabla F(\boldsymbol{x}^k)-\boldsymbol{v}^k\|\le 2LD/K_t$	(7.51)
Convex expectation	$\mathbb{E}\|\nabla F(\boldsymbol{x}^k)-\boldsymbol{v}^k\|\le 3LD/K_t$	(7.52)
Non-convex finite-sum	$\mathbb{E}\|\nabla F(\boldsymbol{x}^k)-\boldsymbol{v}^k\|\le LD/\sqrt{TK}$	(7.53)
Non-convex expectation	$\mathbb{E}\|\nabla F(\boldsymbol{x}^k)-\boldsymbol{v}^k\|\le LD/\sqrt{TK}+\varepsilon/2$	(7.54)

Proof. From Lemma 7.9, we have the following inequality for $k = 2, 3, \ldots, K_t$:

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \frac{L^2}{S_{t,k}}\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2 + \|\nabla F(\boldsymbol{x}^{k-1}) - \boldsymbol{v}^{k-1}\|^2. \tag{7.55}$$

By definition, $\|\boldsymbol{x}^{k-1} - \boldsymbol{x}^k\|^2 = \|\eta_{t,k-1}(\boldsymbol{w}^{k-1} - \boldsymbol{x}^{k-1})\|^2 \leq \eta_{t,k-1}^2 D^2$. Hence, we get

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \frac{\eta_{t,k-1}^2 L^2 D^2}{S_{t,k}} + \|\nabla F(\boldsymbol{x}^{k-1}) - \boldsymbol{v}^{k-1}\|^2. \tag{7.56}$$

Convex finite-sum:
We take the telescopic sum of (7.56) from $i = 2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \sum_{i=2}^{k} \frac{\eta_{t,i-1}^2 L^2 D^2}{S_{t,i}} + \underbrace{\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2}_{0\text{, since we take full batch}} \leq \frac{4L^2D^2}{K_t}\sum_{i=2}^{k}\frac{1}{(s_{t,i-1}+1)^2} \tag{7.57}$$

By definition of $s_{t,k}$, for any $i \geq 2$ we have

$$s_{t,i-1} + 1 = K_t + i - 1 \geq K_t. \tag{7.58}$$

Hence, we get

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \frac{4L^2D^2}{K_t^3}\sum_{i=2}^{k} 1 \leq \frac{4L^2D^2}{K_t^3}k \leq \frac{4L^2D^2}{K_t^2}. \tag{7.59}$$

By using Jensen's inequality, we get (7.51).

Convex expectation minimization:
We take the telescopic sum of (7.56) from $i = 2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \frac{4L^2D^2}{K_t^2} + \|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2 \leq \frac{9L^2D^2}{K_t^2}, \tag{7.60}$$

where the bound on $\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2$ follows from Lemma 7.10 as

$$\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2 \leq \frac{\sigma^2}{Q_t} = \frac{5L^2D^2}{K_t^2}. \tag{7.61}$$

We get (7.52) by using Jensen's inequality.

Non-convex finite-sum:
We take the telescopic sum of (7.56) from $i = 2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \sum_{i=2}^{k} \frac{\eta_{t,i}^2 L^2 D^2}{S_{t,i}} + \underbrace{\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2}_{0, \text{ since we take full batch}} \leq \frac{L^2 D^2}{TK^2} \sum_{i=2}^{k} 1 \leq \frac{L^2 D^2}{TK} \tag{7.62}$$

We get (7.53) by using Jensen's inequality.

Non-convex expectation minimization:
We take the telescopic sum of (7.56) from $i = 2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\|^2 \leq \frac{L^2 D^2}{TK} + \|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2 \leq \frac{L^2 D^2}{TK} + \frac{\varepsilon^2}{4} \tag{7.63}$$

where the bound on $\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2$ follows from Lemma 7.10 as

$$\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2 \leq \frac{\sigma^2}{Q_t} \leq \frac{\varepsilon^2}{4} \tag{7.64}$$

We get (7.54) by using Jensen's inequality. □

Proof for Theorem 7.1 and Corollary 7.1

We start by the Taylor expansion and smoothness:

$$\begin{aligned} F(\boldsymbol{x}^{k+1}) &\leq F(\boldsymbol{x}^k) + \langle \nabla F(\boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \rangle + \frac{L}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \\ &\leq F(\boldsymbol{x}^k) + \eta_{t,k}\langle \nabla F(\boldsymbol{x}^k), \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta_{t,k}^2 \frac{L}{2} D^2 \\ &= F(\boldsymbol{x}^k) + \eta_{t,k}\langle \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta_{t,k}\langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta_{t,k}^2 \frac{L}{2} D^2 \end{aligned} \tag{7.65}$$

By definition of $\boldsymbol{w}^k$, we have

$$\langle \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle = \min_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{v}^k, \boldsymbol{x} - \boldsymbol{x}^k \rangle \leq \langle \boldsymbol{v}^k, \boldsymbol{x}_\star - \boldsymbol{x}^k \rangle \tag{7.66}$$

Substituting this inequality back, and rearranging, we get

$$\begin{aligned} F(\boldsymbol{x}^{k+1}) &\leq F(\boldsymbol{x}^k) + \eta_{t,k}\langle \boldsymbol{v}^k, \boldsymbol{x}_\star - \boldsymbol{x}^k \rangle + \eta_{t,k}\langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta_{t,k}^2 \frac{L}{2} D^2 \\ &= F(\boldsymbol{x}^k) + \eta_{t,k}\langle \nabla F(\boldsymbol{x}^k), \boldsymbol{x}_\star - \boldsymbol{x}^k \rangle + \eta_{t,k}\langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}_\star \rangle + \eta_{t,k}^2 \frac{L}{2} D^2 \end{aligned} \tag{7.67}$$

From convexity of F, we know

$$\langle \nabla F(\boldsymbol{x}^k), \boldsymbol{x}_\star - \boldsymbol{x}^k \rangle \leq F_\star - F(\boldsymbol{x}^k) \tag{7.68}$$

and by using Cauchy-Schwarz, we have

$$\langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}_\star \rangle \leq \|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\| \|\boldsymbol{w}^k - \boldsymbol{x}_\star\| \leq \|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\| D \tag{7.69}$$

Putting (7.68) and (7.69) back into (7.67), and subtracting $F_\star$ from both sides, we obtain:

$$F(\boldsymbol{x}^{k+1}) - F_\star \leq (1-\eta_{t,k})(F(\boldsymbol{x}^k) - F_\star) + \eta_{t,k} D \|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\| + \eta_{t,k}^2 \frac{L}{2} D^2 \tag{7.70}$$

Then, we take the expectation of both sides and use (7.51) to get

$$\begin{aligned} \mathbb{E}[F(\boldsymbol{x}^{k+1})] - F_\star &\leq (1-\eta_{t,k})(\mathbb{E}[F(\boldsymbol{x}^k)] - F_\star) + \eta_{t,k} D \mathbb{E}\|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\| + \eta_{t,k}^2 \frac{L}{2} D^2 \\ &\leq (1-\eta_{t,k})(\mathbb{E}[F(\boldsymbol{x}^k)] - F_\star) + \eta_{t,k} \frac{2LD^2}{K_t} + \eta_{t,k}^2 \frac{L}{2} D^2 \end{aligned} \tag{7.71}$$

Telescopic sum of this inequality over (t,k) pairs gives

$$\begin{aligned} \mathbb{E}[F(\boldsymbol{x}^{k+1})] - F_\star \leq \sum_{(\tau,i)} \left(\eta_{\tau,i} \frac{2LD^2}{K_\tau} + \eta_{\tau,i}^2 \frac{L}{2} D^2 \right) \prod_{(\tau',j)=(\tau,i)}^{(t,k)} (1-\eta_{\tau',j}) \\ + (\mathbb{E}[F(\boldsymbol{x}^{1,1})] - F_\star) \prod_{(\tau,i)} (1-\eta_{\tau,i}) \end{aligned} \tag{7.72}$$

The last term vanishes due to 0 factor ($\eta_{1,1} = 1$). Remark that

$$\prod_{(\tau',j)=(\tau,i)}^{(t,k)} (1-\eta_{\tau',j}) = \prod_{r=i}^{K_\tau} \frac{s_{\tau,r}-1}{s_{\tau,r}+1} \prod_{\tau'=\tau+1}^{t-1} \prod_{j=1}^{K_{\tau'}} \frac{s_{\tau',j}-1}{s_{\tau',j}+1} \prod_{\ell=1}^{k} \frac{s_{t,\ell}-1}{s_{t,\ell}+1} = \frac{(s_{\tau,i}-1)s_{\tau,i}}{s_{t,k}(s_{t,k}+1)} \tag{7.73}$$

Combining these, we get

$$\mathbb{E}[F(\boldsymbol{x}^{k+1})] - F_\star \leq \sum_{(\tau,i)} \left(\eta_{\tau,i} \frac{2LD^2}{K_\tau} + \eta_{\tau,i}^2 \frac{L}{2} D^2 \right) \frac{(s_{\tau,i}-1)s_{\tau,i}}{s_{t,k}(s_{t,k}+1)}. \tag{7.74}$$

We focus on the individual terms:

$$\sum_{(\tau,i)} \eta_{\tau,i} \frac{2LD^2}{K_\tau} \frac{(s_{\tau,i}-1)s_{\tau,i}}{s_{t,k}(s_{t,k}+1)} \leq \frac{8LD^2}{s_{t,k}(s_{t,k}+1)} \sum_{(\tau,i)} 1 \leq \frac{8LD^2}{s_{t,k}+1} \tag{7.75}$$

$$\sum_{(\tau,i)} \eta_{\tau,i}^2 \frac{L}{2} D^2 \frac{(s_{\tau,i}-1)s_{\tau,i}}{s_{t,k}(s_{t,k}+1)} \leq \frac{2LD^2}{s_{t,k}(s_{t,k}+1)} \sum_{(\tau,i)} 1 \leq \frac{2LD^2}{K_t+k} \tag{7.76}$$

We proved the convergence rate:

$$\mathbb{E}[F(\boldsymbol{x}^{k+1})] - F_\star \leq \frac{10LD^2}{s_{t,k}+1} \tag{7.77}$$

To get ε-solution, we set the number of outer iterations T_ε such that

$$\mathbb{E}[F(\tilde{\boldsymbol{x}}^{T_\varepsilon})] - F_\star \leq \frac{10LD^2}{K_{T_\varepsilon}} \leq \varepsilon. \tag{7.78}$$

Then, it is sufficiant to choose

$$T_\varepsilon = \log_2\left(\frac{10LD^2}{\varepsilon}\right) + 1. \tag{7.79}$$

Then, to achieve an ε-solution, we can calculate the *ifo* complexity as

$$\begin{aligned} \#(ifo) = \sum_{t=1}^{T_\varepsilon}\left(Q_t + \sum_{k=2}^{K_t} S_{t,k}\right) &= \sum_{t=1}^{T_\varepsilon}\left(n + \sum_{k=2}^{K_t} 2^{t-1}\right) \\ &\leq \sum_{t=1}^{T_\varepsilon}\left(n + 2^{2(t-1)}\right) \\ &= \mathcal{O}\left(nT_\varepsilon + 2^{2T_\varepsilon}\right) = \mathcal{O}\left(n\ln\left(\frac{LD^2}{\varepsilon}\right) + \frac{L^2D^4}{\varepsilon^2}\right) \end{aligned} \tag{7.80}$$

and the *lmo* complexity as

$$\#(lmo) = \sum_{t=1}^{T_\varepsilon} K_t \leq 2K_{T_\varepsilon} = 2^{T_\varepsilon} = \mathcal{O}\left(\frac{LD^2}{\varepsilon}\right). \tag{7.81}$$

Proof for Theorem 7.2 and Corollary 7.2

Proof is similar to that for finifte-sum setting, but we use (7.52) instead of (7.51) at (7.71), hence the constants change:

$$\mathbb{E}[F(\boldsymbol{x}^{k+1})] - F_\star \leq \frac{14LD^2}{K_t + k}. \tag{7.82}$$

To get ε-solution, we set the number of outer iterations T_ε as

$$T_\varepsilon = \log_2\left(\frac{14LD^2}{\varepsilon}\right) + 1. \tag{7.83}$$

Then, to achieve $(1-\varepsilon)$ accuracy, we can calculate *sfo* complexity as

$$\#(sfo) = \sum_{t=1}^{T_\varepsilon}\left(Q_t + \sum_{k=2}^{K_t} S_{t,k}\right) \leq \sum_{t=1}^{T_\varepsilon}\left(\lceil\frac{\sigma^2K_t^2}{5L^2D^2}\rceil + K_t^2\right) = \mathcal{O}\left(\frac{\sigma^2D^2}{\varepsilon^2} + \frac{L^2D^4}{\varepsilon^2}\right). \tag{7.84}$$

The *lmo* complexity is same as the finite-sum case.

Proof for Theorem 7.3 and Corollary 7.3

We start by the Taylor expansion and smoothness:

$$\begin{aligned}
F(\boldsymbol{x}^{k+1}) &\le F(\boldsymbol{x}^k) + \langle \nabla F(\boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \rangle + \frac{L}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \\
&\le F(\boldsymbol{x}^k) + \eta \langle \nabla F(\boldsymbol{x}^k), \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta^2 \frac{L}{2} D^2 \\
&= F(\boldsymbol{x}^k) + \eta \langle \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta \langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta^2 \frac{L}{2} D^2 \\
&\le F(\boldsymbol{x}^k) + \eta \langle \boldsymbol{v}^k, \boldsymbol{w}_\star^k - \boldsymbol{x}^k \rangle + \eta \langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{x}^k \rangle + \eta^2 \frac{L}{2} D^2 \\
&= F(\boldsymbol{x}^k) + \eta \langle \nabla F(\boldsymbol{x}^k), \boldsymbol{w}_\star^k - \boldsymbol{x}^k \rangle + \eta \langle \nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k, \boldsymbol{w}^k - \boldsymbol{w}_\star^k \rangle + \eta^2 \frac{L}{2} D^2 \\
&\le F(\boldsymbol{x}^k) - \eta \mathcal{G}(\boldsymbol{x}^k) + \eta D \|\nabla F(\boldsymbol{x}^k) - \boldsymbol{v}^k\| + \eta^2 \frac{L}{2} D^2
\end{aligned} \tag{7.85}$$

where $\boldsymbol{w}_\star^k = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \langle \boldsymbol{x}, -\nabla F(\boldsymbol{x}^k) \rangle$. Taking expectation of both sides and using (7.53),

$$\mathbb{E}[F(\boldsymbol{x}^{k+1})] \le \mathbb{E}[F(\boldsymbol{x}^k)] - \frac{1}{\sqrt{TK}} \mathbb{E}[\mathcal{G}(\boldsymbol{x}^k)] + \frac{3LD^2}{2TK} \tag{7.86}$$

Rearranging, and summing over all (t,k) pairs up to (T,K), we obtain

$$\frac{1}{\sqrt{TK}} \sum_{(\tau,i)}^{(T,K)} \mathbb{E}[\mathcal{G}(\boldsymbol{x}^{\tau,i})] \le F(\bar{\boldsymbol{x}}^1) - \mathbb{E}[F(\boldsymbol{x}^{T,K})] + \sum_{(\tau,i)}^{(T,K)} \frac{3LD^2}{2TK} \le F(\bar{\boldsymbol{x}}^1) - F(\boldsymbol{x}_\star) + \frac{3LD^2}{2}. \tag{7.87}$$

Hence, by definition of $\boldsymbol{x}^{\text{out}}$, we have

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{\text{out}})] \le \frac{F(\bar{\boldsymbol{x}}^1) - F(\boldsymbol{x}_\star)}{\sqrt{TK}} + \frac{3LD^2}{2\sqrt{TK}} = \frac{2\mathcal{E} + 3LD^2}{2\sqrt{TK}} \tag{7.88}$$

This completes the convergence rate proof.

To get ε-solution, we set the number of outer iterations T_ε such that

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{\text{out}})] \le \frac{2\mathcal{E} + 3LD^2}{2\sqrt{T_\varepsilon K}} \le \varepsilon. \tag{7.89}$$

Hence, it suffices to choose

$$T_\varepsilon = \left(\frac{2\mathcal{E} + 3LD^2}{2\varepsilon\sqrt{K}} \right)^2 = \frac{(2\mathcal{E} + 3LD^2)^2}{4\varepsilon^2 K}. \tag{7.90}$$

Then, to achieve and ε-solution, we can calculate *ifo* complexity as

$$\begin{aligned}\#(ifo) = \sum_{t=1}^{T_\varepsilon}\left(Q_t + \sum_{k=2}^{K} S_{t,k}\right) = \sum_{t=1}^{T_\varepsilon}\left(n + \sum_{k=2}^{K} K\right) \leq (n+K^2)T_\varepsilon \leq \sqrt{n}\frac{3(2\mathcal{E}+3LD^2)^2}{4\varepsilon^2} \\ = \mathcal{O}\left((\mathcal{E}^2 + L^2D^4)\frac{\sqrt{n}}{\varepsilon^2}\right)\end{aligned} \tag{7.91}$$

and the *lmo* complexity as

$$\#(lmo) = \sum_{t=1}^{T_\varepsilon} K = KT_\varepsilon = \frac{(2\mathcal{E}+3LD^2)^2}{4\varepsilon^2} = \mathcal{O}\left((\mathcal{E}^2 + L^2D^4)\frac{1}{\varepsilon^2}\right) \tag{7.92}$$

Proof for Theorem 7.4 and Corollary 7.4

Proof is similar to that for non-convex finifte-sum setting, but we use (7.54) instead of (7.53) at (7.85), and we get

$$\mathbb{E}[F(\boldsymbol{x}^{k+1})] \leq \mathbb{E}[F(\boldsymbol{x}^k)] - \frac{1}{\sqrt{TK}}\mathbb{E}[\mathcal{G}(\boldsymbol{x}^k)] + \frac{3LD^2}{2TK} + \frac{\varepsilon}{2\sqrt{TK}} \tag{7.93}$$

Rearranging, and summing over all (t,k) pairs up to (T,K), we get

$$\begin{aligned}\frac{1}{\sqrt{TK}}\sum_{(\tau,i)}^{(T,K)}\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{\tau,i})] &\leq F(\bar{\boldsymbol{x}}^1) - \mathbb{E}[F(\boldsymbol{x}^{T,K})] + \sum_{(\tau,i)}^{(T,K)}\left(\frac{3LD^2}{2TK} + \frac{\varepsilon}{2\sqrt{TK}}\right) \\ &\leq F(\bar{\boldsymbol{x}}^1) - F(\boldsymbol{x}_\star) + \frac{3LD^2}{2} + \frac{\varepsilon\sqrt{TK}}{2}.\end{aligned} \tag{7.94}$$

Hence, by definition of $\boldsymbol{x}^{\text{out}}$, we have

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{\text{out}})] \leq \frac{F(\bar{\boldsymbol{x}}^1) - F(\boldsymbol{x}_\star)}{\sqrt{TK}} + \frac{3LD^2}{2\sqrt{TK}} + \frac{\varepsilon}{2} = \frac{2\mathcal{E}+3LD^2}{2\sqrt{TK}} + \frac{\varepsilon}{2} \tag{7.95}$$

This completes the convergence proof.

To get ε-solution, we set the number of outer iterations T_ε such that

$$\mathbb{E}[\mathcal{G}(\boldsymbol{x}^{\text{out}})] \leq \frac{2\mathcal{E}+3LD^2}{2\sqrt{T_\varepsilon K}} + \frac{\varepsilon}{2} \leq \varepsilon. \tag{7.96}$$

Then, to achieve an ε-solution, we can calculate *sfo* complexity as

$$\begin{aligned}\#(sfo) = \sum_{t=1}^{T_\varepsilon}\left(Q_t + \sum_{k=2}^{K} S_{t,k}\right) = \sum_{t=1}^{T_\varepsilon}\left(n + \sum_{k=2}^{K} K\right) \leq (\lceil 4(\sigma/\varepsilon)^2\rceil + (\lceil\sigma/\varepsilon\rceil)^2)T_\varepsilon \\ = \mathcal{O}\left((\mathcal{E}^2 + L^2D^4)\frac{\sigma}{\varepsilon^3}\right)\end{aligned} \tag{7.97}$$

Finally, *lmo* complexity is same as the non-convex finite-sum case.

Proofs for SPIDER-CGS

Lemma 7.15. Suppose that the assumptions listed in Section 7.2 hold. Then, for $k = 1, \dots, K_t$, we have the following bounds:

Convex finite-sum	$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \le 2\sqrt{2}LD/(s_{t,k}+1)^2$	(7.98)
Convex expectation	$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \le 3LD/(s_{t,k}+1)^2$	(7.99)
Non-convex finite-sum	$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \le 2LD/\sqrt{TK}$	(7.100)
Non-convex expectation	$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \le 2LD/\sqrt{TK} + \varepsilon/2$	(7.101)

Proof. From Lemma 7.9, we have the following inequality for all $k = 2, \dots, K_t$:

$$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 \le \frac{L^2}{S_{t,k}}\|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 + \|\nabla F(\boldsymbol{z}^{k-1}) - \boldsymbol{v}^{k-1}\|^2. \tag{7.102}$$

By definition,

$$\begin{aligned}
\boldsymbol{z}^k &= \boldsymbol{y}^k + \gamma_{t,k}(\boldsymbol{x}^k - \boldsymbol{y}^k) \\
&= \boldsymbol{y}^{k-1} + \gamma_{t,k-1}(\boldsymbol{x}^k - \boldsymbol{y}^{k-1}) + \gamma_{t,k}(\boldsymbol{x}^k - \boldsymbol{y}^k) \\
\& \quad \boldsymbol{z}^{k-1} &= \boldsymbol{y}^{k-1} + \gamma_{t,k-1}(\boldsymbol{x}^{k-1} - \boldsymbol{y}^{k-1}) \\
\implies \|\boldsymbol{z}^k - \boldsymbol{z}^{k-1}\|^2 &= \|\gamma_{t,k-1}(\boldsymbol{x}^k - \boldsymbol{x}^{k-1}) + \gamma_{t,k}(\boldsymbol{x}^k - \boldsymbol{y}^k)\|^2 \\
&= \gamma_{t,k-1}^2\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2 + \gamma_{t,k}^2\|\boldsymbol{x}^k - \boldsymbol{y}^k\|^2 + 2\gamma_{t,k-1}\gamma_{t,k}\langle \boldsymbol{x}^k - \boldsymbol{y}^k, \boldsymbol{x}^k - \boldsymbol{x}^{k-1}\rangle \\
&\le \gamma_{t,k-1}^2\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|^2 + \gamma_{t,k-1}^2\|\boldsymbol{x}^k - \boldsymbol{y}^k\|^2 + 2\gamma_{t,k-1}^2\|\boldsymbol{x}^k - \boldsymbol{y}^k\|\,\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\| \\
&\le 4\gamma_{t,k-1}^2 D^2
\end{aligned} \tag{7.103}$$

Substituting into (7.102), we get

$$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 \le \frac{4\gamma_{t,k-1}^2 L^2 D^2}{S_{t,k}} + \|\nabla F(\boldsymbol{z}^{k-1}) - \boldsymbol{v}^{k-1}\|^2. \tag{7.104}$$

Convex finite-sum:
We take the telescopic sum of (7.104) from $i = 2$ to k

$$\begin{aligned}
\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 &\le \sum_{i=2}^{k} \frac{4\gamma_{t,i-1}^2 L^2 D^2}{S_{t,i}} + \underbrace{\|\nabla F(\boldsymbol{z}^1) - \boldsymbol{v}^1\|^2}_{0, \text{ since we take full batch}} \\
&= \frac{4L^2D^2}{9K_t(s_{t,K_t}+1)^2}\sum_{i=2}^{k}\frac{9}{(s_{t,i-1}+2)^2} = \frac{4L^2D^2}{K_t(s_{t,K_t}+1)^2}\sum_{i=2}^{k}\frac{1}{(s_{t,i}+1)^2}
\end{aligned} \tag{7.105}$$

Clearly, $s_{t,i}+1 \geq \frac{s_{t,K_t}+1}{\sqrt{2}}$ for all $2 \leq i \leq K_t$. Hence, we get

$$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 \leq \frac{8L^2D^2}{K_t(s_{t,K_t}+1)^2} \sum_{i=2}^{k} \frac{1}{(s_{t,K_t}+1)^2} \leq \frac{8L^2D^2}{(s_{t,k}+1)^4} \tag{7.106}$$

We get (7.98) by using Jensen's inequality.

Convex expectation minimization:
We take the telescopic sum of (7.104) from $i=2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 \leq \frac{8L^2D^2}{(s_{t,k}+1)^4} + \|\nabla F(\boldsymbol{z}^1) - \boldsymbol{v}^1\|^2 \leq \frac{8L^2D^2}{(s_{t,k}+1)^4} + \frac{\sigma^2}{Q_t} \leq \frac{9L^2D^2}{(s_{t,k}+1)^4} \tag{7.107}$$

where the bound on $\|\nabla F(\boldsymbol{x}^1) - \boldsymbol{v}^1\|^2$ follows from Lemma 7.10.

We get (7.99) by using Jensen's inequality.

Non-convex finite-sum:
We take the telescopic sum of (7.104) from $i=2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 \leq \sum_{i=2}^{k} \frac{4\gamma^2L^2D^2}{S} + \underbrace{\|\nabla F(\boldsymbol{z}^1) - \boldsymbol{v}^1\|^2}_{0,\text{ since we take full batch}} \leq \sum_{i=2}^{k} \frac{4L^2D^2}{TK^2} \leq \frac{4L^2D^2}{TK} \tag{7.108}$$

We get (7.100) by using Jensen's inequality.

Non-convex expectation minimization:
We take the telescopic sum of (7.104) from $i=2$ to k

$$\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|^2 \leq \sum_{i=2}^{k} \frac{4\gamma^2L^2D^2}{S} + \|\nabla F(\boldsymbol{z}^1) - \boldsymbol{v}^1\|^2 \leq \frac{4L^2D^2}{TK} + \frac{\varepsilon^2}{4} \tag{7.109}$$

where the bound on $\|\nabla F(\boldsymbol{z}^1) - \boldsymbol{v}^1\|^2$ follows from Lemma 7.10. We get (7.101) by using Jensen's inequality. □

Proof for Theorem 7.5 and Corollary 7.5

We start by Taylor expansion and smoothness:

$$\begin{aligned}
F(\boldsymbol{y}^{k+1}) &\leq F(\boldsymbol{z}^k) + \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{y}^{k+1} - \boldsymbol{z}^k \rangle + \frac{L}{2}\|\boldsymbol{y}^{k+1} - \boldsymbol{z}^k\|^2 \\
&= F(\boldsymbol{z}^k) + \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{y}^k - \boldsymbol{z}^k \rangle + \gamma_{t,k}\langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle \\
&\qquad + \gamma_{t,k}\langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}_\star - \boldsymbol{y}^k \rangle + \frac{L\gamma_{t,k}^2}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \qquad (7.110) \\
&= (1-\gamma_{t,k})(F(\boldsymbol{z}^k) + \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{y}^k - \boldsymbol{z}^k \rangle) + \gamma_{t,k}(F(\boldsymbol{z}^k) + \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}_\star - \boldsymbol{z}^k \rangle) \\
&\qquad + \gamma_{t,k}\langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle + \frac{L\gamma_{t,k}^2}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2
\end{aligned}$$

From the convexity of F, we have

$$F(\boldsymbol{y}^k) \geq F(\boldsymbol{z}^k) + \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{y}^k - \boldsymbol{z}^k \rangle \qquad \text{and} \qquad F_\star \geq F(\boldsymbol{z}^k) + \langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}_\star - \boldsymbol{z}^k \rangle \qquad (7.111)$$

Hence, we get

$$\begin{aligned}
F(\boldsymbol{y}^{k+1}) &\leq (1-\gamma_{t,k})F(\boldsymbol{y}^k) + \gamma_{t,k}F_\star + \gamma_{t,k}\langle \nabla F(\boldsymbol{z}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}^k \rangle + \frac{L\gamma_{t,k}^2}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \\
&= (1-\gamma_{t,k})F(\boldsymbol{y}^k) + \gamma_{t,k}F_\star + \gamma_{t,k}\langle \boldsymbol{v}^k + \beta_{t,k}(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle \\
&\qquad - \gamma_{t,k}\beta_{t,k}\langle \boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle + \gamma_{t,k}\langle \nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle \\
&\qquad + \frac{L\gamma_{t,k}^2}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \\
&\leq (1-\gamma_{t,k})F(\boldsymbol{y}^k) + \gamma_{t,k}F_\star + \gamma_{t,k}\alpha_{t,k} - \gamma_{t,k}\beta_{t,k}\langle \boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle \\
&\qquad + \gamma_{t,k}\langle \nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle + \frac{L\gamma_{t,k}^2}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 \qquad (7.112) \\
&= (1-\gamma_{t,k})F(\boldsymbol{y}^k) + \gamma_{t,k}F_\star + \gamma_{t,k}\alpha_{t,k} + \frac{\beta_{t,k}\gamma_{t,k}}{2}\left(\|\boldsymbol{x}^k - \boldsymbol{x}_\star\|^2 - \|\boldsymbol{x}^{k+1} - \boldsymbol{x}_\star\|^2\right) \\
&\qquad + \gamma_{t,k}\left(\frac{L\gamma_{t,k} - \beta_{t,k}}{2}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|^2 + \langle \nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{x}_\star \rangle\right) \\
&\leq (1-\gamma_{t,k})F(\boldsymbol{y}^k) - \gamma_{t,k}F_\star + \gamma_{t,k}\alpha_{t,k} + \frac{\beta_{t,k}\gamma_{t,k}}{2}\left(\|\boldsymbol{x}^k - \boldsymbol{x}_\star\|^2 - \|\boldsymbol{x}^{k+1} - \boldsymbol{x}_\star\|^2\right) \\
&\qquad + \gamma_{t,k}D\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|
\end{aligned}$$

where the second inequality follows from the definition of $\alpha_{t,k}$, and the last inequality from the fact that $\beta_{t,k} \geq L\gamma_{t,k}$ together with Cauchy-Schwarz inequality. Then, we subtract $F_\star$ from both sides, take the expectation of both sides, and compute the telescopic sum over (t,k). The first term vanishes due to the $(1-\gamma_{1,1}) = 0$ factor.

Denoting by $\mathcal{E}_{t,k} := \mathbb{E}\|\boldsymbol{x}^k - \boldsymbol{x}_\star\|^2$, and $\mathcal{E}_{t,k+} := \mathbb{E}\|\boldsymbol{x}^{k+1} - \boldsymbol{x}_\star\|^2$, we get

$$\begin{aligned}&\mathbb{E}[F(\boldsymbol{y}^{k+1}) - F_\star] \\ &\quad\leq \sum_{(\tau,i)} \left[\gamma_{\tau,i}\alpha_{\tau,i} + \frac{\beta_{\tau,i}\gamma_{\tau,i}}{2}\left(\mathcal{E}_{\tau,i} - \mathcal{E}_{\tau,i+}\right) + \gamma_{t,k} D\mathbb{E}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|\right] \prod_{(\tau',j)=(\tau,i)}^{(t,k)} (1-\gamma_{\tau',j}).\end{aligned} \tag{7.113}$$

Remark that

$$\prod_{(\tau',j)=(\tau,i)}^{(t,k)} (1-\gamma_{\tau',j}) = \prod_{r=i}^{K_\tau} \frac{s_{\tau,r}-1}{s_{\tau,r}+2} \prod_{\tau'=\tau+1}^{t-1} \prod_{j=1}^{K_{\tau'}} \frac{s_{\tau',j}-1}{s_{\tau',j}+2} \prod_{\ell=1}^{k} \frac{s_{t,\ell}-1}{s_{t,\ell}+2} = \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \tag{7.114}$$

Now we focus on the individual terms

$$\sum_{(\tau,i)} \gamma_{\tau,i}\alpha_{\tau,i} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \leq \sum_{(\tau,i)} \frac{3}{s_{\tau,i}+2} \frac{2LD^2}{s_{\tau,i}^2} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \tag{7.115}$$

Now, show that

$$\begin{aligned}&\sum_{(\tau,i)} \frac{\beta_{\tau,i}\gamma_{\tau,i}}{2}\left(\mathcal{E}_{\tau,i} - \mathcal{E}_{\tau,i+}\right) \prod_{(\tau',j)=(\tau,i)}^{(t,k)} (1-\gamma_{\tau',j}) \\ &\qquad= \sum_{(\tau,i)} \frac{27L}{4} \frac{1}{(s_{\tau,i}+2)^2} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \left(\mathcal{E}_{\tau,i} - \mathcal{E}_{\tau,i+}\right) \\ &\qquad= \frac{27L}{4s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \sum_{(\tau,i)} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{(s_{\tau,i}+2)^2} \left(\mathcal{E}_{\tau,i} - \mathcal{E}_{\tau,i+}\right)\end{aligned} \tag{7.116}$$

Remark that

$$\begin{aligned}&\sum_{s_{\tau,i}=1}^{s_{t,k}} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{(s_{\tau,i}+2)^2}\mathcal{E}_{\tau,i} - \sum_{s_{\tau,i}=1}^{s_{t,k}} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{(s_{\tau,i}+2)^2}\mathcal{E}_{\tau,i+} \\ &\qquad= \sum_{s_{\tau,i}=1}^{s_{t,k}-1} \frac{s_{\tau,i}(s_{\tau,i}+1)(s_{\tau,i}+2)}{(s_{\tau,i}+3)^2}\mathcal{E}_{\tau,i+} - \sum_{s_{\tau,i}=1}^{s_{t,k}} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{(s_{\tau,i}+2)^2}\mathcal{E}_{\tau,i+} \\ &\qquad\leq \sum_{s_{\tau,i}=1}^{s_{t,k}} \frac{s_{\tau,i}(s_{\tau,i}+1)(s_{\tau,i}+2)}{(s_{\tau,i}+3)^2}\mathcal{E}_{\tau,i+} - \sum_{s_{\tau,i}=1}^{s_{t,k}} \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{(s_{\tau,i}+2)^2}\mathcal{E}_{\tau,i+} \\ &\qquad\leq D^2 \sum_{s_{\tau,i}=1}^{s_{t,k}} \left(\frac{s_{\tau,i}(s_{\tau,i}+1)(s_{\tau,i}+2)}{(s_{\tau,i}+3)^2} - \frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{(s_{\tau,i}+2)^2}\right) \\ &\qquad\leq D^2 s_{t,k}\end{aligned} \tag{7.117}$$

Hence, we get

$$\sum_{(\tau,i)} \frac{\beta_{\tau,i}\gamma_{\tau,i}}{2}\left(\mathcal{E}_{\tau,i}-\mathcal{E}_{\tau,i+}\right)\prod_{(\tau',j)=(\tau,i)}^{(t,k)}(1-\gamma_{\tau',j}) \leq \frac{27LD^2}{4(s_{t,k}+1)(s_{t,k}+2)} \tag{7.118}$$

Finally, we focus on the last term:

$$\begin{aligned}\sum_{(\tau,i)} \gamma_{\tau,i} D\mathbb{E}\|\nabla F(\boldsymbol{z}^{\tau,i})-\boldsymbol{v}^{\tau,i}\|\frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \\ \leq LD^2\sum_{s_{\tau,i}=1}^{s_{t,k}}\frac{6\sqrt{2}}{(s_{\tau,i}+2)(s_{\tau,i}+1)^2}\frac{(s_{\tau,i}-1)s_{\tau,i}(s_{\tau,i}+1)}{s_{t,k}(s_{t,k}+1)(s_{t,k}+2)} \\ \leq \frac{6\sqrt{2}LD^2}{(s_{t,k}+1)(s_{t,k}+2)}\end{aligned} \tag{7.119}$$

Combining these bounds, we obtain

$$\mathbb{E}[F(\boldsymbol{y}^{k+1})-F_\star] = \mathcal{O}\left(\frac{LD^2}{(s_{t,k}+1)(s_{t,k}+2)}\right) \tag{7.120}$$

Easy to verify by induction that $K_t \leq s_{t,k} \leq 4K_t$. Hence, $s_{t,k} = \Theta(K_t) = \Theta(2^{t/2})$.

Hence, $\mathbb{E}[F(\boldsymbol{y}^{k+1})-F_\star] = \mathcal{O}(LD^2 2^{-t})$. Therefore, to get ϵ-solution, we set T_ϵ as

$$T_\epsilon = \Theta\left(\log_2\left(\frac{LD^2}{\epsilon}\right)\right) \tag{7.121}$$

Then, to achieve this accuracy, we can calculate *ifo* complexity as

$$\begin{aligned}\#(ifo) = \sum_{t=1}^{T_\epsilon}\left(Q_t+\sum_{k=2}^{K_t}S_{t,k}\right) = \mathcal{O}\left(\sum_{t=1}^{T_\epsilon}\left(n+\sum_{k=2}^{K_t}2^{3t/2}\right)\right) = \mathcal{O}\left(nT_\epsilon+\sum_{t=1}^{T_\epsilon}2^{2t}\right) \\ = \mathcal{O}\left(nT_\epsilon+2^{2T_\epsilon}\right) \\ = \mathcal{O}\left(n\ln\left(\frac{LD^2}{\epsilon}\right)+\frac{L^2D^4}{\epsilon^2}\right)\end{aligned} \tag{7.122}$$

Finally, we can find *lmo* complexity by using Lemma 7.11:

$$\#(lmo) = \mathcal{O}\left(\sum_{t=1}^{T_\epsilon}\sum_{k=1}^{K_t}\frac{4\beta_{t,k}D^2}{\alpha_{t,k}}\right) = \mathcal{O}\left(\sum_{t=1}^{T_\epsilon}K_t^2\right) = \mathcal{O}\left(2^{T_\epsilon}\right) = \mathcal{O}\left(\frac{LD^2}{\epsilon}\right) \tag{7.123}$$

Proof for Theorem 7.6 and Corollary 7.6

Similar to the proof of finifte-sum setting, but we use (7.99) instead of (7.98), hence the constants at (7.119) change.

To get ϵ-solution, we can calculate *sfo* complexity as

$$\begin{aligned}\#(sfo) = \sum_{t=1}^{T_\epsilon}\left(Q_t + \sum_{k=2}^{K_t} S_{t,k}\right) = \mathcal{O}\left(\sum_{t=1}^{T_\epsilon}\left(\frac{\sigma^2 K_t^4}{L^2D^2} + \sum_{k=2}^{K_t} 2^{3t/2}\right)\right) = \mathcal{O}\left(\left(\frac{\sigma^2}{L^2D^2}+1\right)\sum_{t=1}^{T_\epsilon} 2^{2t}\right) \\ = \mathcal{O}\left(\left(\tfrac{\sigma^2}{L^2D^2}+1\right)2^{T_\epsilon}\right) = \mathcal{O}\left(\frac{\sigma^2D^2 + L^2D^4}{\epsilon^2}\right) \quad (7.124)\end{aligned}$$

The *lmo* complexity is same as the convex finite-sum case.

Proof for Theorem 7.7 and Corollary 7.7

We start by (7.35), and rearrange it to obtain

$$F(\boldsymbol{y}^{k+1}) \le F(\boldsymbol{y}^k) - \gamma\mathcal{G}(\boldsymbol{y}^k) + \gamma\langle\nabla F(\boldsymbol{y}^k) - \boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle + \gamma\langle\boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle + \gamma^2\frac{L}{2}D^2 \quad (7.125)$$

where $\boldsymbol{w}_\star^k = \arg\max_{\boldsymbol{x}\in\mathcal{X}}\langle\boldsymbol{x}, -\nabla F(\boldsymbol{y}^k)\rangle$.

Focus on the last inner-product term

$$\begin{aligned}\gamma\langle\boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle &= \gamma\langle\boldsymbol{v}^k + \beta(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle - \gamma\beta\langle\boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle \\ &\le \gamma\alpha - \gamma\beta\langle\boldsymbol{x}^{k+1} - \boldsymbol{x}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle \\ &\le \gamma\alpha + \gamma\beta D^2\end{aligned} \quad (7.126)$$

The first inequality follows from the role of α in CndG, and the second from Cauchy-Schwarz.

Now we use

$$\begin{aligned}\gamma\langle\nabla F(\boldsymbol{y}^k) - \boldsymbol{v}^k, \boldsymbol{x}^{k+1} - \boldsymbol{w}_\star^k\rangle &\le \gamma D\|\nabla F(\boldsymbol{y}^k) - \boldsymbol{v}^k\| \\ &\le \gamma D\|\nabla F(\boldsymbol{y}^k) - \nabla F(\boldsymbol{z}^k)\| + \gamma D\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \\ &\le \gamma LD\|\boldsymbol{y}^k - \boldsymbol{z}^k\| + \gamma D\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \\ &= \gamma^2 LD\|\boldsymbol{x}^k - \boldsymbol{y}^k\| + \gamma D\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| \\ &= \gamma^2 LD^2 + \gamma D\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\|\end{aligned} \quad (7.127)$$

Combining these bounds, we obtain

$$\begin{aligned}F(\boldsymbol{y}^{k+1}) &\le F(\boldsymbol{y}^k) - \gamma\mathcal{G}(\boldsymbol{y}^k) + \gamma D\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| + \gamma\alpha + \gamma\beta D^2 + \frac{3}{2}\gamma^2 LD^2 \\ &= F(\boldsymbol{y}^k) - \frac{1}{\sqrt{TK}}\mathcal{G}(\boldsymbol{y}^k) + \frac{D}{\sqrt{TK}}\|\nabla F(\boldsymbol{z}^k) - \boldsymbol{v}^k\| + \frac{4LD^2}{TK}\end{aligned} \quad (7.128)$$

Now we take expectation of both sides and use (7.100), and we obtain

$$\mathbb{E}[F(\boldsymbol{y}^{k+1})] \leq \mathbb{E}[F(\boldsymbol{y}^{k})] - \frac{1}{\sqrt{TK}}\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{k})] + \frac{6LD^2}{TK} \tag{7.129}$$

Rearranging, and summing over all (t,k) pairs up to (T,K), we get

$$\frac{1}{\sqrt{TK}}\sum_{(\tau,i)}\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{\tau,i})] \leq F(\bar{\boldsymbol{x}}^{1}) - F(\boldsymbol{x}_{\star}) + 6LD^2 \tag{7.130}$$

Hence, by definition of $\boldsymbol{y}^{\text{out}}$, we get

$$\mathbb{E}[\mathcal{G}(\boldsymbol{y}^{\text{out}})] \leq \frac{\mathcal{E} + 6LD^2}{\sqrt{TK}} \tag{7.131}$$

This completes the convergence rate proof.

Proof for the *ifo* complexity follows similarly to the one for SPIDER-FW.

To show *lmo* complexity, we use

$$\#(lmo) = \mathcal{O}\left(\sum_{t=1}^{T_\varepsilon}\sum_{k=1}^{K}\frac{4\beta D^2}{\alpha}\right) = \mathcal{O}\left(\sum_{t=1}^{T_\varepsilon}\sum_{k=1}^{K}6\right) = \mathcal{O}\left(6KT_\varepsilon\right) = \mathcal{O}\left((\mathcal{E}^2 + L^2D^4)\frac{1}{\varepsilon^2}\right) \tag{7.132}$$

Proof for Theorem 7.8 and Corollary 7.8

Follows similarly as in the non-convex finite-sum case, but we use (7.101) instead of (7.100), hence we have additional $\varepsilon/2$ term on the right-hand-side of (7.131). *sfo* complexity follows similarly as in SPIDER-FW. *lmo* complexity is the same as the finite-sum case.

8 Conclusion & Future Directions

This chapter presents an overview of our main contributions along with some future directions.

In this dissertation, we considered a fundamental class of constrained convex optimization problem (1.1) with broad applications. In particular, we focused on an SDP formulation (1.4), which can be written as a special instance of this problem template. As we described in Chapter 1, storing the matrix decision variable is the critical bottleneck that prevents us from solving many important applications of this SDP formulation at scale.

• In Chapter 2, we introduced a novel convex optimization paradigm for designing algorithms with optimal storage cost, which constructs an approximate solution with rigorous guarantees to large-scale problems with structured solutions. The two key ingredients are to maintain only a sketch of the large decision variable and to use algorithmic templates that are compatible with these sketches. A natural target for this approach is the class of matrix optimization problems with low-rank solutions. We presented the first storage-optimal algorithm for this problem class, SketchyCGM, which modifies the standard conditional gradient method (CGM) to work with sketches. In spite of recent enthusiasm for nonconvex heuristics, our research justifies that the convex optimization did not reach yet its limits of scalability and that it still remains as a viable option at large-scale.

Based on this paradigm, our recent research is focused on designing a practical convex optimization solver for a broad set of semidefinite programming (SDP) instances. Some main ingredients for this design are presented in the subsequent sections. The detailed analysis and implementation of the resulting method for this specific template, with potent numerical demonstrations in cutting-edge applications, is left for future work.

Our paradigm is beyond low-rank optimization. As an immediate future direction, we may consider convex optimization problems with other types of structured solutions. There are different dimensionality reduction techniques to store and retrieve structured objects. One example in this line is designing a storage-optimal convex optimization algorithm for problems with sparse or group-sparse solutions, with potential applications in optimal transport.

• In Chapter 3, we proposed and studied new primal-dual optimization algorithms for solving a generic constrained convex optimization template, which covers our model SDP formulation (1.4) as a special instance. The proposed methods, UPD and its accelerated variant AUPD, have several key advantages compared to the existing primal-dual methods in the literature. As opposed to the smoothing (of the dual problem), we preserve the original structure of the dual problem. Then, we apply a (sub)gradient method to this dual problem, and recover the primal variable as a weighted sum of the solutions to the primal subproblem. This primal averaging sequence is crucial for our storage-optimal optimization paradigm. Moreover, our algorithms are adaptive to the unknown smoothness level (Hölder-smoothness order) of the dual problem, as our analysis leads to optimal iteration complexity guarantees in the sense of first-order black-box models [NY83].

Unfortunately, these algorithms have some substantial limitations at the current stage. Our algorithms borrow a line-search strategy introduced by Nesterov [Nes15] for his universal gradient methods in the unconstrained setting, to achieve adaptivity in the dual problem. Unfortunately, this line-search procedure, hence also our algorithms explicitly depend on the target accuracy input ε, and the methods only guarantee convergence towards an ε-suboptimal set. As a consequence, it is difficult to tune these algorithms without a rough knowledge of the optimal value. Also note that oracle errors can cause these algorithms to get stuck in the line-search procedure.

The literature on adaptive optimization has recently evolved, following the increased interest in stochastic and online learning settings. AdaGrad and its variants attempt to adapt the learning rate using an accumulation of the gradient norms. These developments in the stochastic setting can be translated to the deterministic setting via some online to offline conversion mechanisms to design first-order algorithms that adapt to the smoothness, see [CBCG04, Lev17]. In this line, we introduced an accelerated adaptive method [LYC18]. Our method, AcceleGrad, can be viewed as an accelerated variant of AdaGrad in the deterministic setting. In contrast to AdaGrad, AcceleGrad achieves optimal rates for both smooth and non-smooth problems. It also has convergence guarantees in the stochastic setting.

These results, so far, remained in the unconstrained optimization setting. An important future direction with significant potential impact is extending these results to the constrained setting. We might expect the resulting algorithms to maintain the key benefits of our universal primal-dual convex optimization framework, while addressing most of its seeming limitations. In various problem instances, we observed significant empirical improvement with a heuristic primal-dual variant of AcceleGrad, against UPD and AUPD.

• In Chapter 4, we presented an extension of CGM for solving a generic composite convex minimization template over a convex and compact domain. Our method (HCGM) combines the ideas of smoothing (of the primal problem) and homotopy for the smoothness parameter under the standard CGM template. By means of set-indicator functions, HCGM can also be applied to the problems with affine constraints, and in particular to our model SDP formu-

lation (1.4). In this case, we can view HCGM as a quadratic penalty method. We proved that the method achieves an optimal $\mathcal{O}(1/\sqrt{t})$ convergence rate, which holds even when the linear minimization oracles are noisy with additive or multiplicative errors.

The numerical evaluation of HCGM for the clustering SDP (in Section 4.4.1) also reveals an important phenomenon that supports our claims about trading accuracy for scalability. It is clear from Figure 4.1 that the overall error after rounding (the misclassification rate in this case) saturates after a low-to-medium accuracy level (that we achieved after ~1000 iterations in this example), and solving the optimization problem beyond this accuracy does not lead to any further useful information. This observation can be generalized for the majority of the SDP formulations obtained as the relaxation of a combinatorial decision problem, excluding the examples where the empirical study suggests that the relaxation is exact.

• The slow empirical convergence of HCGM strains its practical impact in solving large-scale SDP instances. In Chapter 5, we address this issue by introducing a natural extension of HCGM, going from quadratic penalty to an augmented Lagrangian formulation. The new method, CGAL, retains the strong theoretical guarantees of HCGM, but it exhibits significantly superior empirical performance. This improvement is due to the dual updates, which can be viewed as shifting the center point of the quadratic penalty in a favorable direction adaptively. In a selection of numerical test setups with SDP formulations, we observed empirical $\mathcal{O}(1/t)$ convergence rate for CGAL, compared against $\mathcal{O}(1/\sqrt{t})$ rate of HCGM.

An immediate future direction is finding a theoretical explanation for this performance improvement, beyond an educated intuition. Remark that we empirically observed $\mathcal{O}(1/t)$ convergence rate for CGAL in various numerical experiments. It might be possible to prove this rate, maybe under more stringent assumptions such as a quadratic growth condition.

HCGM and CGAL preserve the key features of the standard CGM, such as the cheap linear minimization oracles and the structured updates on the decision variable. As a result, they are also amenable to sketching for storage-optimal extensions. Considering its superior empirical performance; CGAL, in particular, might be the key ingredient for the next phase of large-scale SDP solvers.

• Our storage-optimal optimization paradigm is based on a sketching model to truncate the storage costs. Hence, designing practical sketching algorithms for low-rank approximation is a fundamental aspect of our research. In Chapter 6, we summarized our contributions.

Based on the specific role of sketching in our approach, we focus on minimizing the sketch size. Moreover, our sketching algorithms are specifically intended for environments where the data matrix is presented as a stream of linear updates. Our aim is to address questions that arise when using sketching algorithms in practice, that had been often neglected in the literature. Unlike the majority of the existing works, our algorithms are accompanied by informative error bounds, which provide guidance when we set the sketch-size parameters. In addition, we also address the numerical stability problems.

We designed a benchmark for numerical evaluation of streaming low-rank matrix approximation, with synthetic and real datasets. We presented extensive numerical experiments to compare our methods and the other methods in the literature, in order to identify methods that produce the best approximations in practice, under various scenarios. We implemented our methods as well as the other methods that we consider in these experiments, treating the sketch as an abstract data type using ideas from object-oriented programming. Beyond optimization, we also presented concrete applications on "on-the-fly compression" of large-scale scientific data. We omit most of these details in this dissertation for brevity, we refer to our original manuscripts for these details. We published our code online, which is available at "https://github.com/alpyurtsever/SKETCH" as a software toolbox for MATLAB. This toolbox also includes various dimensionality reduction maps, optional *a posteriori* error estimator that allows the user to validate the quality of the reconstructed matrix, as well as the codes to regenerate the results of our numerical experiments.

See the follow-up paper [SGL+19] of our collaborators on an algorithm for low-rank Tucker approximation of a tensor from streaming data as an extension in this line of research.

• Lastly, in Chapter 7, we studied the theoretical complexity of the CGM variants for stochastic expectation minimization and finite-sum formulations, in order to identify and present the tightest results known so far. In this chapter, we also introduced a class of novel variance-reduced stochastic CGM variants, based on the recent stochastic path-integrated differential estimator (SPIDER) technique [FLLZ18].

An interesting future direction is to extend these algorithms for solving problems with affine constraints, in a similar way to HCGM or CGAL. In this line, we have already taken an initial step in our recent work [LYFC19], by combining our homotopy smoothing approach with a stochastic CGM framework by Mokhtari et al. [MHK18], and demonstrated the application of the proposed method for solving some basic stochastic SDP problems. While the guaranteed rate of our method is not very fast, this is the first stochastic CGM variant for solving problems with affine constraints. We can expect to achieve better rates using our SPIDER-CGM approach as the base method. Developing a well-tuned implementation is another important piece of future work.

SDP formulations are key in signal processing, machine learning, and other engineering applications. Most aspects of my research are immediately applicable in addition to their long term promise. By greatly enhancing the scalability of optimization methods for solving SDP problems, we can potentially unlock new results in important scientific applications.

Bibliography

[AB15] A. Agarwal and L. Bottou. A lower bound for the optimization of finite sums. In *Proc. 32nd Int. Conf. Machine Learning*, 2015.

[Ali91] F. Alizadeh. *Combinatorial optimization with interior point methods and semi-definite matrices*. PhD thesis, University of Minnesota, 1991.

[Ali93] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1993.

[ARR14] A. Ahmed, B. Recht, and J. Romberg. Blind deconvolution using convex programming. *IEEE Trans. on Inf. Theory*, 60(3):1711–1732, 2014.

[AT06] A. Auslender and M. Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM J. Optim.*, 16(3):697–725, 2006.

[Bac15] F. Bach. Duality between subgradient and conditional gradient methods. *SIAM J. Optim.*, 25(1):115–129, 2015.

[Bar95] A. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13(2):189–202, 1995.

[BBV16] A.S. Bandeira, N. Boumal, and V. Voroninski. On the low-rank approach for semidefinite programs arising in synchronization and community detection. In *29th Annual Conference on Learning Theory*, 2016.

[BC11] Heinz H. Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator thoery in Hilbert spaces*. Springer, New York, NY, 2011.

[Ber76] D. P. Bertsekas. On penalty and multiplier methods for constrained minimization. *SIAM J. Control Optim.*, 14(2):216–235, 1976.

[Ber99] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2nd edition, 1999.

[BKS16] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. *J. Mach. Learn. Res.*, 49:1–53, 2016.

[BM03] S. Burer and R. D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Math. Program.*, 95(2, Ser. B):329–357, 2003.

[BNPS17] J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Math. Program.*, 165(2):471–507, 2017.

[Bou15] N. Boumal. A riemannian low-rank method for optimization over semidefinite matrices with block-diagonal constraints. arXiv:1506.00575v2, 2015.

[BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Machine Learning*, 3(1):1–122, 2011.

[Bra06] M. Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra Appl.*, 415(1):20–30, 2006.

[BT09] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

[BVB18] N. Boumal, V. Voroninski, and A.S. Bandeira. Deterministic guarantees for Burer–Monteiro factorizations of smooth semidefinite programs. arXiv:1804.02008v1, 2018.

[BWZ16] C. Boutsidis, D. Woodruff, and P. Zhong. Optimal principal component analysis in distributed and streaming models. In *Proc. 48th ACM Symp. Theory of Computing*, 2016.

[CBCG04] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Trans. on Inf. Theory*, 50(9):2050–2057, 2004.

[CD13] J. Chiu and L. Demanet. Sublinear randomized algorithms for skeleton decompositions. *SIAM J. Matrix Anal. Appl.*, 34(3):1361–1383, 2013.

[CESV13] E. J. Candès, Y. C. Eldar, T. Strohmer, and V. Voroninski. Phase retrieval via matrix completion. *SIAM J. Imaging Sciences*, 6(1):199–225, 2013.

[CJN17] B. Cox, A. Juditsky, and A. Nemirovski. Decomposition techniques for bilinear saddle point problems and variational inequalities with affine monotone operators. *J. Optim. Theory Appl.*, 2(402–435), 2017.

[CKS15] K. N. Chaudhury, Y. Khoo, and A. Singer. Global registration of multiple point clouds using semidefinite programming. *SIAM J. Optim.*, 25(1):468–501, 2015.

[Cla10] K. L. Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms*, 6(4), 2010.

[CLDM18] H. Chang, Y. Lou, Y. Duan, and S. Marchesini. Total variation–based phase retrieval for poisson noise removal. *SIAM J. Imaging Sciences*, 11(1):24–55, 2018.

[CLS15a] E. J. Candès, X. Li, and M. Soltanolkotabi. Phase retrieval from coded diffraction patterns. *Applied and Computational Harmonic Analysis*, 39(2):277–299, 2015.

[CLS15b] E. J. Candès, X. Li, and M. Soltanolkotabi. Phase retrieval via Wirtinger Flow: Theory and algorithms. *IEEE Trans. Inform. Theory*, 61(4):1985–2007, 2015.

[CP08] P. L. Combettes and J.-C. Pesquet. A proximal decomposition method for solving convex variational inverse problems. *Inverse Problems*, 24(6):065014, 2008.

[CR12] E. Candès and B. Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

[CSV12] E. J. Candès, T. Strohmer, and V. Voroninski. Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming. *Communications on Pure and Applied Mathematics*, 2012.

[CVY18] V. Cevher, B. C. Vu, and A. Yurtsever. Stochastic forward Douglas-Rachford splitting method for monotone inclusions. In P. Giselsson and A. Rantzer, editors, *Large–Scale and Distributed Optimization*, chapter 7, pages 149–179. Springer International Publishing, 2018.

[CVY19] V. Cevher, B. C. Vu, and A. Yurtsever. Inertial three-operator splitting method and applications. arXiv:1904.12980v1, 2019.

[CZH+17] R. Chandra, Z. Zhong, J. Hontz, V. McCulloch, C. Studer, and T. Goldstein. Phasepack: A phase retrieval library. *Asilomar Conference on Signals, Systems, and Computers*, 2017.

[DBLJ14] A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems 28*, 2014.

[DFTJ16] C. Dünner, S. Forte, M. Takác, and M. Jaggi. Primal–dual rates and certificates. In *Proc. 33rd Int. Conf. Machine Learning*, 2016.

[dGJL07] A. d'Aspremont, L.E. Ghaoui, M.I. Jordan, and G.R.G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Review*, 49(3):434–448, 2007.

[DH78] J. Dunn and S. Harshbarger. Conditional gradient algorithms with open loop step size rules. *J. Math. Anal. Appl.*, 62(2):432–444, 1978.

[DHS11] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011.

[DM05] P. Drineas and M. W. Mahoney. On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *J. Mach. Learn. Res.*, 6:2153–2175, 2005.

[Dun79] J. Dunn. Rates of convergence for conditional gradient algorithms near singular and nonsinglular extremals. *SIAM J. Control Optim.*, 17(2):187–211, 1979.

[Dun80] J. Dunn. Convergence rates for conditional gradient sequences generated by implicit step length rules. *SIAM J. Control Optim.*, 18(5):473–487, 1980.

[DYC+19] L. Ding, A. Yurtsever, V. Cevher, J. A. Tropp, and M. Udell. An optimal-storage approach to semidefinite programming using approximate complementarity. arXiv:1902.03373v1, 2019.

[Faz02] M. Fazel. *Matrix rank minimization with applications*. PhD thesis, Stanford Univ., 2002.

[FBCM04] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):214–225, Jan. 2004.

[FGM17] R. M. Freund, P. Grigas, and R. Mazumder. An extended Frank–Wolfe method with "in-face" directions and its application to low-rank matrix completion. *SIAM J. Optim.*, 27(1):319–346, 2017.

[FLLZ18] C. Fang, C. J. Li, Z. Lin, and T. Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. In *Advances in Neural Information Processing Systems 32*, 2018.

[FW56] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[Gar12] D. Garber. *Approximating Semidefinite Programs in Sublinear Time*. PhD thesis, Israel Institute of Technology, 2012.

[GEB13] T. Goldstein, E. Esser, and R. Baraniuk. Adaptive primal-dual hybrid gradient methods for saddle point problems. arXiv:1305.0546v1, 2013.

[GH11] D. Garber and E. Hazan. Approximating semidefinite programs in sublinear time. In *Advances in Neural Information Processing Systems 25*, 2011.

[GH16] D. Garber and E. Hazan. Sublinear time algorithms for approximate semidefinite programming. *Math. Program., Ser. A*, (158):329–361, 2016.

[Git11] A. Gittens. The spectral norm error of the naïve Nyström extension. arXiv:1110.5305, 2011.

[Git13] A. Gittens. *Topics in Randomized Numerical Linear Algebra*. PhD thesis, California Institute of Technology, 2013.

[GJLJ17] G. Gidel, T. Jebara, and S. Lacoste-Julien. Frank-Wolfe algorithms for saddle point problems. In *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, 2017.

[GLF+10] D. Gross, Y.-K. Liu, S. T. Flammia, S. Becker, and J. Eisert. Quantum state tomography via compressed sensing. *Phys. Rev. Lett.*, 105, 2010.

[GM13] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. arXiv:1303.1849, 2013.

[GM16] A. Gittens and M. W. Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *J. Mach. Learn. Res.*, 17:Paper No. 117, 65, 2016.

[GPLJ18] G. Gidel, F. Pedregosa, and S. Lacoste-Julien. Frank-Wolfe splitting via augmented Lagrangian method. In *Proc. 21st Int. Conf. Artificial Intelligence and Statistics*, 2018.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[Ham84] H. J. Hammond. *Solving asymmetric variational inequality problems and systems of equations with generalized nonlinear programming algorithms*. PhD thesis, MIT, 1984.

[Haz08] E. Hazan. Sparse approximate solutions to semidefinite programs. In *Proc. 8th Latin American Conf. Theoretical Informatics*, pages 306–316, 2008.

[HCO+15] R. Horstmeyer, R. Y. Chen, X. Ou, B. Ames, J. A. Tropp, and C. Yang. Solving ptychography with a convex relaxation. *New J. Physics*, 17(5):053044, 2015.

[HH15] N. He and Z. Harchaoui. Semi–proximal mirror–prox for nonsmooth composite minimization. In *Advances in Neural Information Processing Systems 28*, 2015.

[HJL96] C. L. Hamilton-Jester and C.-K. Li. Extreme vectors of doubly nonnegative matrices. *Rocky Mountain J. Math.*, 26(4):1371–1383, 1996.

[HJN15] Z. Harchaoui, A. Juditsky, and A. Nemirovski. Conditional gradient algorithms for norm–regularized smooth convex optimization. *Math. Program., Ser. A*, (152):75–112, 2015.

[HK12] E. Hazan and S. Kale. Projection–free online learning. In *Proc. 29th Int. Conf. Machine Learning*, 2012.

[HK16] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm Trans. Interative Interlligent Systems*, 5(4):19, 2016.

[HKMS19] H. Hassani, A. Karbasi, A. Mokhtari, and Z. Shen. Stochastic conditional gradient++. arXiv:1902.06992, 2019.

[HL16] E. Hazan and H. Luo. Variance-reduced and projection-free stochastic optimization. In *Proc.* 33*rd Int. Conf. Machine Learning*, 2016.

[HMT11] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288, 2011.

[HUS10] J.-B. Hiriart-Urruty and A. Seeger. A variational approach to copositive matrices. *SIAM Review*, 52(4):593–629, 20010.

[Jag13] M. Jaggi. Revisiting Frank–Wolfe: Projection–free sparse convex optimization. In *Proc.* 30*th Int. Conf. Machine Learning*, 2013.

[JN16] A. Juditsky and A. Nemirovski. Solving variational inequalities with monotone operators on domains given by Linear Minimization Oracles. *Math. Program.*, 156(1-2):221–256, 2016.

[JNS13] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proc.* 45*th Ann. ACM Symp. Theory of Computing*, 2013.

[JS10] M. Jaggi and M. Sulovský. A Simple Algorithm for Nuclear Norm Regularized Problems. In *Proc.* 27*th Int. Conf. Machine Learning*, 2010.

[JZ13] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, 2013.

[KN12] S. Khot and A. Naor. Grothendieck-type inequalities in combinatorial optimization. *Communications on Pure and Applied Mathematics*, 65(7):992–1035, 2012.

[Lan14] G. Lan. The complexity of large–scale convex programming under a linear optimization oracle. arXiv:1309.5550v2, 2014.

[LCG+04] G.R.G. Lanckriet, N. Cristianini, L.E. Ghaoui, P. Bartlett, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, 5:27–72, 2004.

[Lev17] K. Y. Levy. Online to offline conversions, universality and adaptive minibatch sizes. In *Advances in Neural Information Processing Systems*, pages 1612–1621, 2017.

[LF18] H. Lu and R. M. Freund. Generalized stochastic Frank-wolfe algorithm with stochastic" substitute"gradient for structured convex optimization. arXiv:1807.07680, 2018.

[LJ16] S. Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. arXiv:1607.00345, 2016.

[LJJSP13] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proc. 30th Int. Conf. Machine Learning*, 2013.

[LKTJ17] F. Locatello, R. Khanna, M. Tschannen, and M. Jaggi. A unified optimization view on generalized matching pursuit and Frank-Wolfe. In *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, 2017.

[LL12] J. Lavaei and H.L. Low. Zero duality gap in optimal power flow problem. *IEEE Trans. on Power Syst.*, 27(1):92–107, February 2012.

[LLM19] Y.-F. Liu, X. Liu, and S. Ma. On the non-ergodic convergence rate of an inexact augmented Lagrangian framework for composite convex programming. *Mathematics of Operations Research*, 44(2):632–650, 2019.

[LLS+17] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert. Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Trans. Math. Softw.*, 43(3):28:1–28:14, Jan. 2017.

[LM15] G. Lan and R. D. C. Monteiro. Iteration-Complexity of First-Order Augmented Lagrangian Methods for Convex Programming. *Math. Program.*, pages 1–37, 2015.

[LMWY13] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Trans. Pattern Anal. Mach. Intell*, 35(1):208–230, 2013.

[Lov03] L. Lovász. Semidefinite programs and combinatorial optimization. In *Recent advances in algorithms and combinatorics*, pages 137–194. Springer, 2003.

[LP66] E. Levitin and B. Polyak. Constrained minimization methods. *USSR Comput. Math. & Math. Phys.*, 6(5):1–50, 1966.

[LPZZ17] G. Lan, S. Pokutta, Y. Zhou, and D. Zink. Conditional accelerated lazy stochastic gradient descent. In *Proc. 34th Int. Conf. Machine Learning*, 2017.

[LTRJ17] F. Locatello, M. Tschannen, G. Rätsch, and M. Jaggi. Greedy algorithms for cone constrained optimization with convergence guarantees. In *Advances in Neural Information Processing Systems 30*, 2017.

[LYC18] K. Y. Levy, A. Yurtsever, and V. Cevher. Online adaptive methods, universality and acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[LYFC19] F. Locatello, A. Yurtsever, O. Fercoq, and V. Cevher. Stochastic conditional gradient method for composite convex minimization. arXiv:1901.10348v2, 2019.

[LZ16] G. Lan and Y. Zhou. Conditional gradient sliding for convex optimization. *SIAM J. Optim.*, 26(2):1379–1409, 2016.

[MHK18] A. Mokhtari, H. Hassani, and A. Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. arXiv:1804.09554, 2018.

[MNS15] E. Mossel, J. Neeman, and A. Sly. Consistency thresholds for the planted bisection model. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 69–75. ACM, 2015.

[Mot52] T. S. Motzkin. Copositive quadratic forms. Technical report, National Bureau of Standards Report No. 1818, 1952.

[MVW17] D. G. Mixon, S. Villar, and R. Ward. Clustering subgaussian mixtures by semidefinite programming. *Information and Inference: A Journal of the IMA*, 6(4):389–415, 2017.

[MZJ13] M. Mahdavi, L. Zhang, and R. Jin. Mixed optimization for smooth functions. In *Advances in Neural Information Processing Systems 26*, 2013.

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[Nes05] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103:127–152, 2005.

[Nes09] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Math. Program.*, 120(1, Ser. B):221–259, 2009.

[Nes15] Y. Nesterov. Universal gradient methods for convex optimization problems. *Math. Program.*, 152(1-2):381–404, 2015.

[Nes17] Y. Nesterov. Complexity bounds for primal-dual methods minimizing the model of objective function. *Math. Program.*, 2017.

[NLST17] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takác. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proc. 34th Int. Conf. Machine Learning*, 2017.

[NN89] Y. Nesterov and A. Nemirovski. *Self-concordant functions and polynomial time methods in convex programming.* USSR Academy of Sciences, Central Economic & Mathematic Institute, 1989.

[NN94] Y. Nesterov and A. Nemirovski. *Interior-point polynomial algorithms in convex programming.* Society for Industrial and Applied Mathematics, 1994.

[NY83] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley & Sons, 1983.

[OLY+16] G. Odor, Y.-H. Li, A. Yurtsever, Y.-P. Hsieh, Q. Tran-Dinh, M. El Halabi, and V. Cevher. Frank-Wolfe works for non-lipschitz continuous gradient objectives: Scalable poisson phase retrieval. In 41*st IEEE Int. Conf. Acoustics, Speech & Signal Processing*, 2016.

[Pat98] G. Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of Operations Research*, 23(2):338–358, 1998.

[Pla05] J. C. Platt. FastMap, MetricMap, and Landmark MDS are all Nyström algorithms. In *Proc. 10th Int. Workshop Artificial Intelligence and Statistics*, 2005.

[PW07] J. Peng and Y. Wei. Approximating K–means–type clustering via semidefinite programming. *SIAM J. Optim.*, 18(1):186–205, 2007.

[QLX18] C. Qu, Y. Li, and H. Xu. Non-convex conditional gradient sliding. In *Proc.* 35*th Int. Conf. Machine Learning*, 2018.

[Rag08] P. Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the* 40*th Annual ACM Symposium on Theory of Computing*, pages 245–254, 2008.

[RDLS18] S. N. Ravi, T. Dinh, V. S. R. Lokhande, and V. Singh. Constrained deep learning using conditional gradient and applications in computer vision. arXiv:1803.06453v1, 2018.

[Ren14] J. Renegar. Efficient first-order methods for linear programming and semidefinite programming. arXiv:1409.5832, 2014.

[Roc70] R. T. Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, 1970.

[Roc76] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM J. Control Optim.*, 14(5):877–898, 1976.

[RSB12] N. L. Roux, M. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 26*, 2012.

[RSPS16] S. J. Reddi, S. Sra, B. Póczos, and A. Smola. Stochastic Frank-Wolfe methods for nonconvex optimization. In 54*th Annual Allerton Conf. Communication, Control, and Computing*, 2016.

[SEC+15] Y. Shechtman, Y. C. Eldar, O. Cohen, H. N. Chapman, J. Miao, and M. Segev. Phase retrieval with application to optical imaging. *IEEE Signal Process. Mag.*, 32(3):87–109, 2015.

[SFZ+19] Z. Shen, C. Fang, P. Zhao, J. Huang, and H. Qian. Complexities in projection-free stochastic non-convex minimization. In *Proc. 22nd Int. Conf. Artificial Intelligence and Statistics*, 2019.

[SGL+19] Y. Sun, Y. Guo, C. Luo, J. A. Tropp, and M. Udell. Low-rank Tucker approximation of a tensor from streaming data. arXiv:1904.10951v1, 2019.

[Sin11] A. Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36, 2011.

[Sio58] M. Sion. On general minimax theorems. *Pacific J. Math.*, 8(1):171–176, 1958.

[SRJ04] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorizations. In *Adv. Neural Information Processing Systems 17*, Vancouver, Dec. 2004.

[SS11] A. Singer and Y. Shkolnisky. Three-dimensional structure determination from common lines in cryo-EM by eigenvectors and semidefinite programming. *SIAM J. Imaging Sciences*, 4(2):543–572, 2011.

[SSGB07] L. Song, A. Smola, A. Gretton, and K. M. Borgwardt. A dependence maximization view of clustering. In *Proc. 20th Int. Conf. Machine Learning*, 2007.

[ST14] R. Shefi and M. Teboulle. Rate of Convergence Analysis of Decomposition Methods Based on the Proximal Method of Multipliers for Convex Minimization. *SIAM J. Optim.*, 24(1):269–297, 2014.

[TDC14] Q. Tran-Dinh and V. Cevher. Constrained convex minimization via model-based excessive gap. In *Advances in Neural Information Processing Systems 27*, 2014.

[TDFC18] Q. Tran-Dinh, O. Fercoq, and V. Cevher. A smooth primal-dual optimization framework for nonsmooth composite convex minimization. *SIAM J. Optim.*, 28(1):96–134, 2018.

[TLJJ06] B. Taskar, S. Lacoste-Julien, and M.I. Jordan. Structured prediction, dual extragradient and Bregman projections. *J. Mach. Learn. Res.*, 2006.

[Tyg14] M. Tygert. Beta versions of Matlab routines for principal component analysis. Available at http://tygert.com/software.html, 2014.

[TYUC17a] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. In *Advances in Neural Information Processing Systems 30*, 2017.

[TYUC17b] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Practical sketching algorithms for low-rank matrix approximation. *SIAM J. Matrix Anal. Appl.*, 38(4):1454–1485, 2017.

[TYUC17c] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Randomized single-view algorithms for low-rank matrix approximation. ACM Report 2017-01, Caltech, Jan. 2017.

[TYUC18] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. More practical sketching algorithms for low-rank matrix approximation. ACM Report 2018-01, Caltech, Oct. 2018.

[TYUC19] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Streaming low–rank matrix approximation with an application to scientific simulation. arXiv:1902.08651, 2019.

[Upa16] J. Upadhyay. Fast and space-optimal low-rank factorization in the streaming model with application in differential privacy. arXiv:1604.01429v3, 2016.

[VNM44] J. Von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton press, 1944.

[WJZ+18] K. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh. SpiderBoost: A class of faster variance-reduced algorithms for nonconvex optimization. arXiv:1810.10690, 2018.

[WLRT08] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. A fast randomized algorithm for the approximation of matrices. *Appl. Comput. Harmon. Anal.*, 25(3):335–366, 2008.

[WS00] C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, 2000.

[WW18] I. Waldspurger and A. Waters. Rank optimality for the Burer-Monteiro factorization. arXiv:1812.03046v1, 2018.

[Xu17] H.-K. Xu. Convergence analysis of the Frank–Wolfe algorithm and its generalization in Banach spaces. arXiv:1710.07367v1, 2017.

[YFC19] A. Yurtsever, O. Fercoq, and V. Cevher. A conditional gradient-based augmented lagrangian framework. In *Proc.* 36*th Int. Conf. Machine Learning*, 2019.

[YFLC18] A. Yurtsever, O. Fercoq, F. Locatello, and V. Cevher. A conditional gradient framework for composite convex minimization with applications to semidefinite programming. In *Proc.* 35*th Int. Conf. Machine Learning*, 2018.

[YHC15] A. Yurtsever, Y.-P. Hsieh, and V. Cevher. Scalable convex methods for phase retrieval. In *6th IEEE Intl. Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, 2015.

[YM10] A. Yoshise and Y. Matsukawa. On optimization over the doubly nonnegative cone. In *IEEE International Symposium on Computer-Aided Control System Design*, 2010.

[YSC19] A. Yurtsever, S. Sra, and V. Cevher. Conditional gradient methods via stochastic path-integrated differential estimator. In *Proc. 36th Int. Conf. Machine Learning*, 2019.

[YST15] L. Yang, D. Sun, and K-.C. Toh. SDPNAL+: A majorized semismooth Newton–CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Math. Program. Comput.*, 7(3):331–366, 2015.

[YTDC15a] A. Yurtsever, Q. Tran-Dinh, and V. Cevher. A universal primal-dual convex optimization framework. In *Advances in Neural Information Processing Systems 28*, 2015.

[YTDC15b] A. Yurtsever, Q. Tran-Dinh, and V. Cevher. Universal primal-dual proximal-gradient methods. arXiv:1502.03123v1, 2015.

[YUTC17] A. Yurtsever, M. Udell, J.A. Tropp, and V. Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Proc. 20th Int. Conf. Artificial Intelligence and Statistics*, 2017.

[YVC16] A. Yurtsever, B. C. Vu, and V. Cevher. Stochastic three-composite convex minimization. In *Advances in Neural Information Processing Systems 29*, 2016.

[YZS14] Y. Yu, X. Zhang, and D. Schuurmans. Generalized conditional gradient for sparse estimation. arXiv:1410.4828v1, 2014.

[ZCM+19] M. Zhang, L. Chen, A. Mokhtari, H. Hassani, and A. Karbasi. Quantized Frank-Wolfe: Communication-efficient distributed optimization. arXiv:1902.06332, 2019.

[ZS18] W.-J. Zeng and H. C. So. Outlier–robust matrix completion via ℓ_p-minimization. *IEEE Trans. on Sig. Process*, 66(5):1125–1140, 2018.

[ZXG18] D. Zhou, P. Xu, and Q. Gu. Stochastic nested variance reduction for nonconvex optimization. In *Advances in Neural Information Processing Systems 32*, 2018.

Alp Yurtsever
EPFL STI IEL LIONS
ELD 244 (Bâtiment ELD)
Station 11, 1015 Lausanne
Switzerland

Website: alpyurtsever.github.io
E-mail : alpyurtsever@gmail.com

Education

Sep 2013 – June 2019	**PhD** École Polytechnique Fédérale de Lausanne, Lausanne Computer and Communication Sciences (EDIC) Laboratory of Information and Inference Systems (LIONS) Thesis topic : Scalable convex optimization methods for semidefinite programming Thesis supervisor : Prof. Volkan Cevher
Sep 2018 – Dec 2018	**Visitor** Massachusetts Institute of Technology, Cambridge Institute for Data, Systems and Society (IDSS) Laboratory for Information and Decision Systems (LIDS) Hosting faculty : Prof. Suvrit Sra
Sep 2009 – June 2013	**BSc** Middle East Technical University, Ankara Electrical and Electronics Engineering Physics (Double Major)

Awards and Fellowships

- SNF Early PostDoc.Mobility
  The fellowship includes a grant towards living costs, awarded by Swiss National Science Foundation for 18 months to successful early-career postdocs who wish to enhance their scientific profile by working at a research institution abroad.
- IEEE CAMSAP 2015 student paper award
  Three best student papers of the workshop among the papers by graduate or undergraduate students are identified by the committee, in Cancun, Mexico, in December 2015.
- EDIC fellowship *One year fellowship program that is granted for the promising applicants with strong academic record, by the doctoral school of IC department at EPFL.*

Service to the community

Reviewing activities

OXFORD Academic: IMA Journal of Numerical Analysis (IMAJNA)
IEEE Transactions on Signal and Information Processing over Networks (SIPN)
IEEE Transactions on Signal Processing (T-SP)
Journal of Selected Topics in Signal Processing (J-STSP)
International Journal of Control, Automation and Systems (IJCAS)
Operations Research Letters
International Conference on Machine Learning (ICML)
Neural Information Processing Systems (NeurIPS)

**Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.**