博士論文 (要約)

Applications of Accelerated Proximal Gradient Methods to Binary Classification and Polynomial Optimization

(加速近接勾配法の2値判別と多項式最適化への応用)

ITO Naoki

伊藤 直紀

Department of Mathematical Informatics
Graduate School of Information Science and Technology
The University of Tokyo
Bunkyo-ku, Tokyo, 113-8656, Japan

December 1, 2017

Doctoral Dissertation
submitted to
Department of Mathematical Informatics
Graduate School of Information Science and Technology, The University of Tokyo
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy in the field of Mathematical Informatics

Abstract

This thesis concerns with large scale optimization methods for binary classification and polynomial optimization. We develop efficient optimization algorithms based on the accelerated proximal gradient (APG) method and appropriate formulations for the algorithms.

First, we present a unified binary classification method. We develop a unified formulation for various binary classification models (including support vector machines (SVMs), logistic regression, and Fisher's descriminant analisys) and a fast APG (FAPG) algorithm for the formulation. Our FAPG is developed by devising various acceleration techniques such as backtracking line search and adaptive restarting strategy. We also give a theoretical convergence guarantee for the proposed FAPG method. Numerical experiments show that our algorithm is stable and highly competitive to specialized algorithms designed for specific models, e.g., sequential minimal optimization and stochastic coordinate descent for SVM.

Next, we improve the doubly nonnegative (DNN) relaxation method for a class of polynomial optimization problems (POPs) proposed by Kim, Kojima, and Toh. In order to approximate the optimal value of a POP by its lower bound, they proposed a DNN relaxation problem and solved it by the bisection and projection (BP) method. The BP method reduces the problem to a dual optimization problem having a single variable and applies the bisection method to the dual; The feasibility of a given point is determined by the APG method. In this thesis, we propose new DNN relaxation problems which give better lower bounds than theirs and can still be solved by the BP method efficiently. We further improved the BP method by developing an adaptive restarting APG and a new criterion for checking feasibility. Numerical experiments demonstrate the advantage of our BP method over SDPNAL+ which is the state-of-the-art solver for DNN optimization problems, especially for very large and sparse POPs.

Finally, by focusing on a class of combinatorial quadratic optimization problems (QOPs), which are special cases of POPs, we examine how the difference in formulations of QOPs can affect on the numerical computation of conic relaxation methods. The binary and complementarity conditions of the combinatorial optimization problems can be expressed in several ways, each of which results in different conic relaxations. For the completely positive (CPP), DNN, and semidefinite programming (SDP) relaxations of the combinatorial QOPs, we prove the equivalences and differences among the relaxations by investigating the feasible regions obtained from different representations of the combinatorial condition. We also theoretically study the issue of the primal and dual nondegeneracy, the existence of an interior solution and the size of the relaxations, as a result of different representations of the combinatorial condition. These characteristics of the conic relaxations affect the numerical efficiency and stability of the algorithms.

Acknowledgment

I would like to show my greatest appreciation to Professor Akiko Takeda. I have studied under her supervision since I was undergraduate, while she is not my official supervisor now due to her transfers. She generously provided me a lot of opportunities necessary for carrying out research. Without her professional guidance and enthusiastic encouragement, this work would not have been possible.

I am also deeply grateful to Professor Satoru Iwata. He provided me the great opportunity to join his laboratory's seminar at the university of Tokyo since I was in Keio university. His unremmiting guidance and comments improved the quality of my research and presentation significantly.

My gratitude extends to the other members of my thesis committee: Professor Hiroshi Hirai, Professor Yoshihiro Kanno, and Professor Taiji Suzuki. Their constructive comments greatly help me to gain more insight, cultivate a better understanding, and deepen this study.

I also would like to express my gratitude to collaborators. My sincere gratitude goes to Professor Kim-Chuan Toh. All of this work highly benefited from many discussions we had. Moreover, I have learned a lot about developing sophisticated numerical software under his guidance. It is my great honor to have an opportunity of doing research with him. The results of Chapter 4 and 5 originate in ideas given by Professor Masakazu Kojima and Professor Sunyoung Kim. I greatly appreciate to them for countless enlightening discussion.

My thanks also go to the previous and present members of the laboratory. In particular, I am indebted to Professor Kunihiko Sadakane, Professor Yuji Nakatsukasa, and Professor Shinichi Tanigawa for their constructive comments and warm supports. I also thank to Tasuku Soma, Yutaro Yamaguchi, and Yu Yokoi. We shared a lot of time working at laboratory from morning to midnight. Getting tired, we went to eat just large amount of meat with our friends: Mirai Tanaka, Takayuki Okuno, Kei Kimura, Yasushi Kawase, and Takanori Maehara. It made my Ph.D. experience unique. Besides the academic community, I am blessed with many supportive friends. I especially thank to Shota Tsujino, Shachimal Seida, and Anna Ogawa for their warm and continuous encouragement.

I also appreciate the financial support by Japan society for the promotion of science (JSPS) and Japan science and technology (JST) agency CREST.

Finally I would like to express my deepest gratitude and appreciation to my family for their unconditional love, understanding, support, and encouragement throughout my study.

Contents

1. Introduction

1.1. Background

An optimization problem is a problem of computing the infimum (or supremum) of a real-valued objective function over a set which is typically defined by equality and inequality constraints. The infimum is called the optimal value. A point in the set is called feasible solution. If there exists a feasible solution at which the objective function has the optimal value, then it is called an optimal solution.

A first-order method is an iterative method of searching the optimal value and an optimal solution using the value of the function and its gradient information. Examples include the steepest descent method [Cauchy, 1847], the gradient projection method (e.g., [Goldstein, 1964; Levitin and Polyak, 1966]), the conditional gradient method [Frank and Wolfe, 1956], and the proximal gradient method (e.g., [Bruck, 1975; Passty, 1979; Fukushima and Mine, 1981]). Compared to the second-order method such as the Newton method and the interior point method (e.g., [Karmarkar, 1984; Tanabe, 1987; Kojima et al., 1989; Nesterov and Nemiroskii, 1994]) which also uses the Hessian matrix information of the function, the first-order method tends to have less memory requirement and computational burden per iteration. The recent growing importance of big data optimization, e.g., in the fields of machine learning and image processing, has motivated researchers to develop the first-order method.

A historical breakthrough in the development of the first-order method is the acceleration technique for the steepest descent method proposed by Nesterov [1983]. It contains a characteristic update rule which adds *momentum* to the sequence of solutions. The convergence rate of the accelerated steepest descent method coincides with the lower complexity bounds of the first-order method proven by Nemirovsky and Yudin [1983]. In that sense, the accelerated method is often called an *optimal* method. While Nesterov's subsequent studies have also proposed various acceleration techniques, Beck and Teboulle [2009] recently developed the accelerated proximal gradient (APG) method by extending the idea of Nesterov in 1983. Through the application of the APG method to image deblurring problem, the study proved the practical efficiency of the acceleration techniques and triggered extensive studies on accelerated first-order methods.

Although the accelerated first-order methods have the optimal convergence rate in theory, it is very important to design various techniques for further speeding up the practical convergence. Backtracking line search [Beck and Teboulle, 2009; Scheinberg et al., 2014] for adjusting stepsize and restarting strategies [O'Donoghue and Candès, 2015; Nesterov, 2013; Lin and Xiao, 2015; Su et al., 2014] for controlling the momentum has been proposed for the purpose. The stepsize is an important parameter for first-order methods. It is advantageous to take a large stepsize whenever possible. The

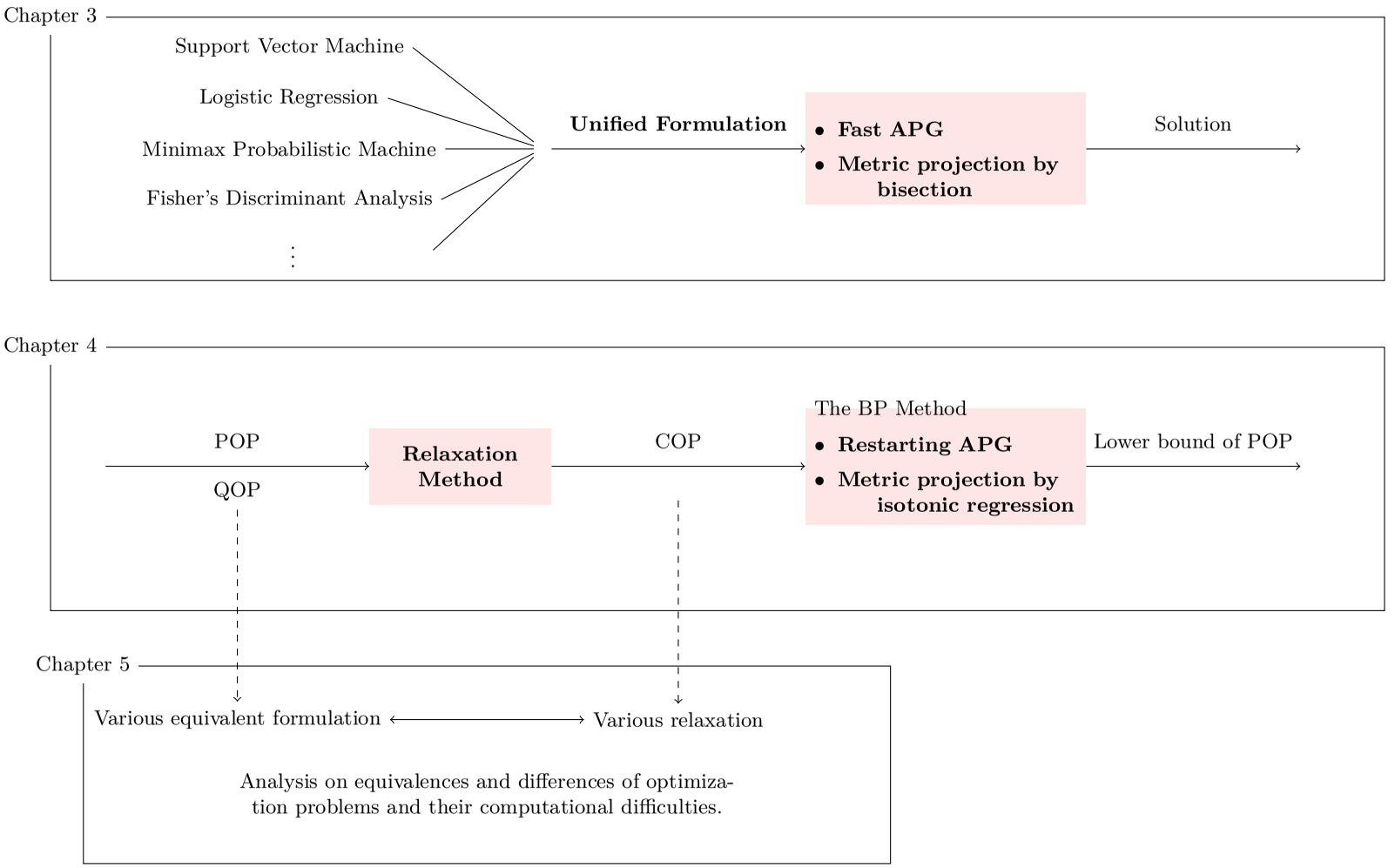
backtracking proposed by Beck and Teboulle [2009] uses a large stepsize at first and decrease the stepsize when a certain condition is violated. It is designed so that the stepsize is non-increasing, which may be conservative, but later improved by Scheinberg et al. [2014] so that the stepsize can be increase. The advantage of the backtracking line search is that it can improve the practical performance while maintaining the optimal convergence rate. The adaptive restarting technique O'Donoghue and Candès [2015] is known as one of the most effective speed-up strategies. It prevents the overshooting of the sequence of solutions which is caused by the momentum. Although Beck and Teboulle [2009] provide a heuristic convergence analysis for their restarting method when the objective function is a strongly convex quadratic function, the convergence for a general convex objective function has been unknown. There are also several other restarting schemes [Nesterov, 2013; Lin and Xiao, 2015; Su et al., 2014] which have guaranteed convergence in the case that the objective function is strongly convex. To the best of our knowledge, however, none of the restarting strategies have convergence guarantee for a general non-strongly convex function.

1.2. Our Contributions

In this thesis, we develop a practically efficient algorithm based on the APG method for large-scale optimization. Our basic idea is to employ and combine various acceleration techniques such as backtracking line search [Beck and Teboulle, 2009; Scheinberg et al., 2014] and adaptive restarting strategy [O'Donoghue and Candès, 2015]. One of the theoretical contributions in this thesis is to show that in fact APG with the adaptive restarting strategy [O'Donoghue and Candès, 2015] can have convergence guarantee for general convex functions by a simple modificaiton. To the best of our knowledge, it is the first convergence results of restarting APG for non-strongly convex functions.

In order to make use of the APG method, it is necessary to formulate a given problem in an appropriate form. Moreover, depending on the structure of the problem formulation, some speeding-up strategies may be invalid. Hence it is necessary to design an algorithm according to the problem formulation. In this thesis, we focus on large-scale optimization problems which appear in binary classification and polynomial optimization, and develop new formulations and algorithms based on the APG method for them.

Figure 1.1.: The diagram of our contributions.



The contributions of this thesis are illustrated in Figure 1 and summarized in the followings.

Unified Binary Classification Method based on APG: The first contribution is to propose a unified binary classification method. Binary classification is the problem of predicting the class a given sample belongs to and is one of the most important problems in machine learning. To achieve a good prediction performance, it is important to find a suitable model for a given dataset. However, it is often time consuming and impractical for practitioners to try various classification models because each model employs a different formulation and algorithm. For example, there has been proposed several highly efficient algorithms [Platt, 1998; Hsieh et al., 2008] for the support vector machines (SVMs) [Cortes and Vapnik, 1995; Schölkopf et al., 2000]. However, it would be difficult to apply the algorithms to other models, and hence practitioners have to implement another algorithm or even develop a new algorithm for the models. The difficulty can be mitigated if we have a unified formulation and an efficient universal algorithmic framework for various classification models to expedite the comparison of performance of different models for a given dataset.

In this thesis, we present a unified formulation of various classification models (including the support vector machines (SVMs) [Cortes and Vapnik, 1995; Schölkopf et al., 2000], Fisher's discriminant analysis [Fisher, 1936], minimax probability machine [Nath and Bhattacharyya, 2007], logistic regression [Cox, 1958], distance weighted discrimination [Marron et al., 2007]) and develop a general optimization algorithm based on the APG method for the formulation. We design various techniques based on backtracking line search [Beck and Teboulle, 2009; Scheinberg et al., 2014] and adaptive restarting strategy [O'Donoghue and Candès, 2015] in order to speed up the practical convergence of our method. We also give a theoretical convergence guarantee for the proposed fast APG algorithm. Numerical experiments show that our algorithm is stable and highly competitive to specialized algorithms designed for specific models, e.g., sequential minimal optimization (SMO) [Platt, 1998; Chang and Lin, 2011] and stochastic coordinate descent [Fan et al., 2008] for SVM.

Doubly Nonnegative Relaxation Method based on APG for Polynomial Optimization: The second contribution of this thesis is to develop a doubly nonnegative (DNN) relaxation method for a class of polynomial optimization problems (POPs) over the nonnegative orthant. POP is a problem of minimizing a polynomial objective function under polynomial equality and inequality constraints. It is known as NP-hard in general and considered to be difficult to obtain the optimal value efficiently and accurately by numerical optimization methods. In order to approximate the optimal value by its lower bound, various convex conic relaxation methods have been proposed. The semidefinite programming (SDP) relaxation has been studied extensively and proved to be very successful in solving various POPs. Recently, the doubly nonnegative (DNN) relaxation has attracted attention as it gives a better lower bound compared with the SDP relaxation. The computational burden of the DNN problem, however, is expensive

since it consists of quadratic number of nonnegative inequality constraints with respect to the size of the variable matrix. Numerical optimization methods to mitigate this difficulty by exploiting the structure of the DNN problem have been proposed, for instance, SDPNAL+ [Yang et al., 2015] and the bisection and projection (BP) method [Kim et al., 2016a; Arima et al., 2017].

The BP method is a first-order method of solving a class of conic optimization problems (COPs) with two simple cone constraints. It reduces the problem to a dual optimization problem having a single variable and applies the bisection to the dual; the feasibility of a given point is determined by the APG method. For binary and box constrained POPs, Kim et al. [2016b] proposed a DNN relaxation formulation that fits in the framework of the BP method. Their formulation is very simple, and hence there is still room for improvement in the approximation accuracy of the DNN relaxation. In this thesis, we propose a method to construct a tighter DNN relaxation problem which still fits in the framework of the BP method. We also provide a method to compute a good upper bound of the trace of the matrix variable, which is important for stabilizing the BP method. Furthermore, the APG method for checking feasibility is improved by the adaptive restarting strategy [O'Donoghue and Candès, 2015]. The theoretical convergence of the restarting APG follows from the analysis of unified binary classification method. This theoretical guarantee is very important in the application of checking feasibility since it requires high solution reliability and accuracy.

Effects of Difference in Conic Relaxation Formulations on Algorithms: As a third contribution, by focusing on a class of combinatorial quadratic optimization problems (QOPs), which are POPs with polynomials of degree 2, we examine how the difference in formulation of QOPs can affect on the numerical algorithm of a conic relaxation method. Various conic relaxations of QOPs with nonnegative variables for combinatorial optimization problems, such as the binary integer quadratic problem, quadratic assignment problem (QAP), and maximum stable set problem have been proposed over the years. The binary and complementarity conditions of the combinatorial optimization problems can be expressed in several ways, each of which results in different conic relaxations. For the completely positive (CPP), DNN, and SDP relaxations of the combinatorial optimization problems, we prove the equivalences and differences among the relaxations by investigating the feasible regions obtained from different representations of the combinatorial condition, a generalization of the binary and complementarity condition. We also theoretically study the issue of the primal and dual nondegeneracy, the existence of an interior solution and the size of the relaxations, as a result of different representations of the combinatorial condition. These characteristics of the conic relaxations affect the numerical efficiency and stability of the solver used to solve them. We demonstrate the theoretical results with numerical results on QAP instances solved by SDPT3 [Tütüncü et al., 2003], SDPNAL+ [Yang et al., 2015], and the BP method based on the APG method.

Finally, we mention the reason of using the APG method in this thesis rather than the

stochastic gradient descent (SGD) methods which are developed in order to solve even larger scale optimization problems, especially in the field of machine learning. SGD approximates the gradient of an objective function by a computationally inexpensive random variable whose expectation coincides with the gradient. In order to design an efficient random variable, SGD in machine learning assumes that the objective function is splittable, i.e., expressed by the sum of differentiable loss functions for each sample. Our unified binary classification method, however, includes binary classification models that do not satisfy this assumption. On the other hand, since APG can solve even a problem whose objective function is unsplittable, it can be used for unified binary classification method and also problems other than machine learning such as polynomial optimization. In other words, the methods proposed in this thesis build on the extensive generality of APG.

The rest of this thesis is organized as follows. In Chapter 2, we define notations and symbols. We also introduce the APG method and existing techniques to improve the practical performance. Chapter 3 presents our first main result: a unified binary classification method. In Chapter 4, we provide a DNN relaxation method for a class of POPs as the second main result. As the third main result, we examine how the difference in formulation of QOPs can affect on the numerical computation of a conic relaxation method in Chapter 5. Finally, Chapter 6 concludes this thesis.

Credit. Chapter 3 is based on the joint work [Ito et al., 2017b] with Akiko Takeda and Kim-Chuan Toh. Chapters 4 and 5 are based on joint works with Sunyoung Kim, Masakazu Kojima, Akiko Takeda, and Kim-Chuan Toh. An earlier version of Chapter 5 appears in the preprint [Ito et al., 2017a].

2. Preliminaries

2.1. Notation and Symbols

Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space. Let $\mathbb{Z}^n \subseteq \mathbb{R}^n$ be the set of n-dimensional integer vectors. Let $\mathbb{Z}^n_+$ be the set of nonnegative vectors in $\mathbb{Z}^n$. We assume that each $\boldsymbol{x} \in \mathbb{R}^n$ is a column vector of element x_i $(1 \leq i \leq n)$. $\boldsymbol{e} \in \mathbb{R}^n$ denotes the all-one vector, and $\boldsymbol{e}_i \in \mathbb{R}^n$ denotes the vector whose i-th element is 1 and all others are 0 $(i = 1, 2, \ldots, n)$. Let $\mathbb{R}^{m\times n}$ be the linear space of $m \times n$ real matrices. The (i, j)-th element of $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is denoted by X_{ij}. For given $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{Y} \in \mathbb{R}^{p\times q}$, the Kronecker product of them is defined by

$$\boldsymbol{X} \otimes \boldsymbol{Y} = \begin{pmatrix} X_{11}\boldsymbol{Y} & X_{12}\boldsymbol{Y} & \cdots & X_{1n}\boldsymbol{Y} \\ X_{21}\boldsymbol{Y} & X_{22}\boldsymbol{Y} & \cdots & X_{2n}\boldsymbol{Y} \\ \vdots & \vdots & \ddots & \vdots \\ X_{m1}\boldsymbol{Y} & X_{m2}\boldsymbol{Y} & \cdots & X_{mn}\boldsymbol{Y} \end{pmatrix} \in \mathbb{R}^{mp\times nq}.$$

Let $\mathbb{S}^n \subseteq \mathbb{R}^{n\times n}$ denote the linear space of $n \times n$ real symmetric matrices. $\boldsymbol{I} \in \mathbb{S}^n$ denotes the identity matrix. $\boldsymbol{E} \in \mathbb{S}^n$ denotes the square matrix whose elements are all one. $\boldsymbol{x}^T$ and $\boldsymbol{X}^T$ denote the transpose of $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{X} \in \mathbb{R}^{m\times n}$, respectively.

We define the inner product $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$ on $\mathbb{R}^n$ by $\boldsymbol{x}^T\boldsymbol{y}$ and the inner product $\langle \boldsymbol{X}, \boldsymbol{Y}\rangle$ on $\mathbb{S}^n$ by $\mathrm{trace}(\boldsymbol{X}^T\boldsymbol{Y})$ $(= \sum_{i=1}^n \sum_{j=1}^n X_{ij}Y_{ij})$. $\|\cdot\|$ denotes the norm induced from the inner product $\langle \cdot, \cdot\rangle$, i.e., $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle}$. $\Pi_{\mathbb{K}}(\boldsymbol{x})$ denotes a metric projection of a point $\boldsymbol{x}$ onto a set $\mathbb{K}$, i.e., $\Pi_{\mathbb{K}}(\boldsymbol{x}) = \mathrm{argmin}_{\boldsymbol{z}}\{\|\boldsymbol{x} - \boldsymbol{z}\| \mid \boldsymbol{z} \in \mathbb{K}\}$. For $p \geq 1$, we also introduce ℓ_p-norm $\|\cdot\|_p$ on $\mathbb{R}^n$ which is defined by

$$\|\boldsymbol{x}\|_p = \Big(\sum_{i=1}^n |x_i|^p\Big)^{1/p} \quad (\boldsymbol{x} \in \mathbb{R}^n).$$

In addition, the ℓ_∞-norm $\|\boldsymbol{x}\|_\infty$ is defined by $\max\{|x_i| \mid i = 1, 2, \ldots, n\}$.

For a finite set S, $|S|$ denotes its cardinality. For given sets S and T, $S + T = \{s + t \mid s \in S,\ t \in T\}$ denotes the Minkowski sum of them.

2.2. The Accelerated Proximal Gradient (APG) Method

In this section, we introduce the accelerated proximal gradient (APG) method [Beck and Teboulle, 2009] which is designed for the following problem:

$$F^* = \min_{\boldsymbol{\alpha}\in\mathbb{R}^d} F(\boldsymbol{\alpha}) \quad \text{where} \quad F(\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}) + g(\boldsymbol{\alpha}). \tag{2.1}$$

Note that one can express a minimization problem constrained over a set S in the form of (2.1) by setting $g(\cdot) = \delta_S(\cdot)$, where

$$\delta_S(\boldsymbol{\alpha}) = \begin{cases} 0 & (\boldsymbol{\alpha} \in S) \\ +\infty & (\boldsymbol{\alpha} \notin S) \end{cases}$$

is the indicator function of the set S.

To apply the APG method to (2.1), we need to assume the following conditions:

1. $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is a proper, closed, and convex function which is possibly nonsmooth. Its effective domain $\mathrm{dom}(g) = \{\boldsymbol{\alpha} \in \mathbb{R}^d \mid g(\boldsymbol{\alpha}) < +\infty\}$ is closed and convex.

2. $f : \mathbb{R}^d \to \mathbb{R}$ is a proper, closed, convex, and continuously differentiable function, and its gradient $\nabla f(\cdot)$ is Lipschitz continuous on $\mathbb{R}^d$,[1] i.e., there exists a constant $L > 0$ such that

$$\|\nabla f(\boldsymbol{\alpha}) - \nabla f(\boldsymbol{\beta})\| \leq L\|\boldsymbol{\alpha} - \boldsymbol{\beta}\| \quad \forall \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^d. \tag{2.2}$$

   The minimum value of such L is referred to as the Lipschitz constant L_f of $\nabla f(\cdot)$.

3. The problem (2.1) is solvable, i.e., the optimal value F^* is finite and an optimal solution $\boldsymbol{\alpha}^*$ exists.

Let $L \geq L_f$. We define an approximate function $Q_L : \mathbb{R}^d \to \mathbb{R}$ of $f(\boldsymbol{\alpha})$ around $\boldsymbol{\beta}$ and a mapping $T_L(\boldsymbol{\alpha}) : \mathbb{R}^d \to \mathbb{R}^d$ as follows:

$$\begin{aligned} Q_L(\boldsymbol{\alpha}; \boldsymbol{\beta}) &= f(\boldsymbol{\beta}) + \langle \nabla f(\boldsymbol{\beta}), \boldsymbol{\alpha} - \boldsymbol{\beta} \rangle + \frac{L}{2}\|\boldsymbol{\alpha} - \boldsymbol{\beta}\|^2 + g(\boldsymbol{\alpha}) \\ T_L(\boldsymbol{\beta}) &= \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^d} Q_L(\boldsymbol{\alpha}; \boldsymbol{\beta}). \end{aligned}$$

The basic proximal gradient (PG) method generates a sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ by

$$\begin{aligned} \boldsymbol{\alpha}^{k+1} = T_L(\boldsymbol{\alpha}^k) &= \operatorname*{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^d} \Big\{ g(\boldsymbol{\alpha}) + \frac{L}{2}\Big\|\boldsymbol{\alpha} - \Big(\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k)\Big)\Big\|^2 \Big\} \\ &= \mathrm{prox}_{g,L}\Big(\boldsymbol{\alpha}^k - \frac{1}{L}\nabla f(\boldsymbol{\alpha}^k)\Big), \end{aligned}$$

where $\mathrm{prox}_{g,L}(\bar{\boldsymbol{\alpha}}) := \mathrm{argmin}_{\boldsymbol{\alpha} \in \mathbb{R}^d} \Big\{ g(\boldsymbol{\alpha}) + \frac{L}{2}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|^2 \Big\}$ is the proximal operator of $g(\cdot)$. If $g(\cdot) = \delta_S(\cdot)$, then $\mathrm{prox}_{g,L_k}(\cdot) = \Pi_S(\cdot)$. In this case, the above PG method coincides with the gradient projection method. If $g(\cdot) = \|\cdot\|_1$, then $\big(\mathrm{prox}_{g,L}(\boldsymbol{\alpha})\big)_i = \mathrm{sign}(\alpha_i)\max\{0, |\alpha_i| - L\}$ $(i = 1, 2, \ldots, d)$ which is known as the soft-thresholding operator. Other analytical computations of the proximal operators of various $g(\cdot)$ can be found in [Parikh and Boyd, 2014, Section 6].

[1] It is sufficient if $\nabla f(\cdot)$ is Lipschitz continuous on a neighborhood of $\mathrm{dom}(g)$: the convex hull of $\mathrm{dom}(g) \cup \{\boldsymbol{\beta}^k \mid k = 1, 2, \ldots\}$, where $\boldsymbol{\beta}^k$ $(k = 1, 2, \ldots)$ are points generated at Step 3 of the APG method. Obviously, however, it is not possible to know the points a priori.

It is known that the PG method has the iteration complexity such that $F(\boldsymbol{\alpha}^k) - F^* \leq O(1/k)$, where $\boldsymbol{\alpha}^*$ is an optimal solution of (2.1). The APG method [Beck and Teboulle, 2009], which is also known as *FISTA*, is an acceleration of the PG method. It generates two sequences $\{\boldsymbol{\beta}^k\}_{k=1}^{\infty}$ and $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$. For an arbitrary initial point $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0 \in \mathbb{R}^d$ and $t_1 = 1$, the APG method solves (2.1) through the following steps ($k = 1, 2, \ldots$):

The Accelerated Proximal Gradient Method [Beck and Teboulle, 2009]

Step 1. Compute

$$\boldsymbol{\alpha}^k \leftarrow T_L(\boldsymbol{\beta}^k) = \mathrm{prox}_{g,L}\Big(\boldsymbol{\beta}^k - \frac{1}{L}\nabla f(\boldsymbol{\beta}^k)\Big).$$

Step 2. Compute $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$.

Step 3. Compute $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$.

For the APG method, the iteration complexity result such that

$$F(\boldsymbol{\alpha}^k) - F^* \leq \frac{2L_f\|\boldsymbol{\alpha}^0 - \boldsymbol{\alpha}^*\|^2}{(k+1)^2}$$

is known (see [Beck and Teboulle, 2009, Theorem 4.4]). The second term $\frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ in Step 3 can be seen as the *momentum* of the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$. It enlarges the moving distance of the sequences $\{\boldsymbol{\alpha}\}_{k=0}^{\infty}, \{\boldsymbol{\beta}\}_{k=1}^{\infty}$ which may lead them closer to the optimum $\boldsymbol{\alpha}^*$ more quickly. While various other APG methods (e.g, [Nesterov, 2013; Monteiro et al., 2016; Su et al., 2014]) are proposed, the above APG (namely, FISTA) is used in many applications because it is simpler to implement.

It is known that $\boldsymbol{\alpha}^*$ is an optimal solution of (2.1) if and only if $\boldsymbol{\alpha}^* = T_L(\boldsymbol{\alpha}^*)$. More specifically, the necessary and sufficient optimality condition for $\boldsymbol{\alpha}^*$ to be an optimal solution of (2.1) is

$$\exists \gamma_\alpha \in \partial g(\boldsymbol{\alpha}^*) \quad \text{s.t.} \quad \langle \nabla f(\boldsymbol{\alpha}^*) + \gamma_\alpha,\, \boldsymbol{\alpha} - \boldsymbol{\alpha}^* \rangle \geq 0, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^d. \tag{2.3}$$

On the other hand, from the definition of $T_L(\boldsymbol{\beta})$, we have

$$\exists \gamma_\beta \in \partial g(T_L(\boldsymbol{\beta})) \quad \text{s.t.} \quad \langle \nabla f(T_L(\boldsymbol{\beta})) + L(T_L(\boldsymbol{\beta}) - \boldsymbol{\beta}) + \gamma_\beta,\, \boldsymbol{\alpha} - T_L(\boldsymbol{\beta}) \rangle \geq 0, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^d, \tag{2.4}$$

for any $\boldsymbol{\beta} \in \mathbb{R}^d$. The term $L(T_L(\boldsymbol{\beta}) - \boldsymbol{\beta})$ in (2.4) can be seen as the residual of the optimality condition (2.3). Thus it would be a natural criterion to terminate the APG if $L\|T_L(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\| < \epsilon$ with a small constant $\epsilon > 0$.

Despite having a strong iteration complexity result, the APG method may still not be efficient enough for practical purpose. In the following, we describe several well-known strategies to make the APG method practically efficient.

2.2.1. Backtracking Strategy

We assume that L is greater than or equals to the Lipschitz constant L_f of $\nabla f(\boldsymbol{\alpha})$. However it is advantageous to use a smaller value for L whenever possible since the constant L plays the role of a step size as in a gradient descent method; fixing L to be the Lipschitz constant L_f is usually too conservative (see Table 3.5 in Section 3.4). Thus we adopt the following backtracking strategy [Beck and Teboulle, 2009] after Step 1 with arbitrary given small constants $\eta_u > 1$ and $L_0 > 0$:

'bt': While

$$F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k), \tag{2.5}$$

update $L_k \leftarrow \eta_u L_k$ and $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k)$. Set $L_{k+1} \leftarrow L_k$.

We note that the inequality $F(\boldsymbol{\alpha}^k) \leq Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ is satisfied if $L_k \geq L_f$, i.e., it is a weaker condition than (2.2). 'bt' ensures that the complexity result $F(\boldsymbol{\alpha}^k) - F^* \leq O(1/k^2)$ of APG still holds.

2.2.2. Decreasing Strategy for the Estimates L_k of the Lipschitz Constant L_f

Beck and Teboulle [2009] designed the backtracking strategy 'bt' so that the values of L_k is non-decreasing. In fact, the convergence analysis of [Beck and Teboulle, 2009] requires the value of L_k to be non-decreasing. However, it is advantageous to decrease the value of L_k whenever possible since the constant $\frac{1}{L_k}$ gives a larger step size.

To allow L_k to decrease, Scheinberg et al. [2014] modifies the APG method so that $\{t_k\}_{k=1}^{\infty}$ satisfies $t_k/L_k \geq t_{k+1}(t_{k+1}-1)/L_{k+1}$ $(\forall k \geq 1)$ and the sequences $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ and $\{\boldsymbol{\beta}^k\}_{k=1}^{\infty}$ are generated along with $\{t_k\}_{k=1}^{\infty}$. To be specific, let us introduce the following simple step to decrease the value of L_k, where $\eta_d > 1$ is a given constant.

'dec': Set $L_{k+1} \leftarrow L_k/\eta_d$.

The modified APG of [Scheinberg et al., 2014] can be described as in Algorithm 2.1. It differs from the original APG in that the updates of t_k and $\boldsymbol{\beta}^k$ are added to 'bt' and Step 2 is modified. The convergence of Algorithm 2.1 is shown as follows.

Proposition 2.2.1 (from Scheinberg et al., 2014)**.** *Let S^* be the set of optimal solutions of* (2.1)*. For any $\boldsymbol{\alpha}^* \in S^*$, the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ generated by Algorithm 2.1 satisfies the following inequality:*

$$F(\boldsymbol{\alpha}^k) - F^* \leq \frac{2\eta_u L_f \|\boldsymbol{\alpha}^0 - \boldsymbol{\alpha}^*\|^2}{k^2}, \quad \forall k \geq 1.$$

Algorithm 2.1 An Accelerated Proximal Gradient Method with Non-Monotonic Step Size

Input: f, ∇f, g, $\text{prox}_{g,L}$, $\epsilon > 0$, $L_1 = L_0 > 0$, $\eta_u > 1$, $\eta_d > 1$, $k_{max} > 0$, $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0$
Output: $\boldsymbol{\alpha}^k$
Initialize: $t_1 \leftarrow 1,\ t_0 \leftarrow 0$
for $k = 1, \ldots, k_{max}$ **do**
  $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # Step 1
  while $F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ **do**
    $L_k \leftarrow \eta_u L_k$ # 'bt'
    $t_k \leftarrow \frac{1+\sqrt{1+4(L_k/L_{k-1})t_{k-1}^2}}{2}$
    $\boldsymbol{\beta}^k \leftarrow \boldsymbol{\alpha}^{k-1} + \frac{t_{k-1}-1}{t_k}(\boldsymbol{\alpha}^{k-1} - \boldsymbol{\alpha}^{k-2})$
    $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # 'bt'
  end while
  if $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\| < \epsilon$ **then**
    break
  end if
  $L_{k+1} \leftarrow L_k/\eta_d$ # 'dec'
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4(L_{k+1}/L_k)t_k^2}}{2}$ # Step 2'
  $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ # Step 3
end for

2.2.3. Restarting Strategy

The value $\frac{t_k-1}{t_{k+1}} \in [0, 1)$ in Step 3 determines the amount of *momentum* in $\frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$. When the value $\frac{t_k-1}{t_{k+1}}$ is close to 1, i.e., the momentum is high, the sequences of solutions $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}$ and $\{\boldsymbol{\beta}^k\}_{k=1}^{\infty}$ would overshoot and oscillate around the optimal solution $\boldsymbol{\alpha}^*$. In order to avoid the oscillation and further speed up the convergence, O'Donoghue and Candès [2015] introduced an adaptive restarting strategy:

're': If $\nabla f(\boldsymbol{\beta}^k)^\top(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}) + g(\boldsymbol{\alpha}^k) - g(\boldsymbol{\alpha}^{k-1}) > 0$,
then update $t_{k+1} \leftarrow 1$, $t_k \leftarrow 0$, $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^{k-1}$, and $\boldsymbol{\alpha}^k \leftarrow \boldsymbol{\alpha}^{k-1}$.

Roughly, the APG method resets the momentum back to zero and restarts from the previous point $\boldsymbol{\alpha}^{k-1}$ if the direction of motion $\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}$ seems to cause the (approximated) objective value to increase, which may be a sign of overshooting. Note that the computational cost of 're' is inexpensive since $\nabla f(\boldsymbol{\beta}^k)$ has already been computed at Step 1. O'Donoghue and Candès [2015] also provided a heuristic convergence analysis for their restarting scheme ('re') when f is a strongly convex (i.e., there exists a constant $\mu > 0$ such that $f(\boldsymbol{\alpha}) - \frac{\mu}{2}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_2^2$ is convex) quadratic function and $g = 0$. However, the convergence of 're' for a general convex objective function is unknown.

There are several other restarting schemes [Nesterov, 2013; Lin and Xiao, 2015; Su et al., 2014]. They have guaranteed convergence in the case that f is strongly convex.

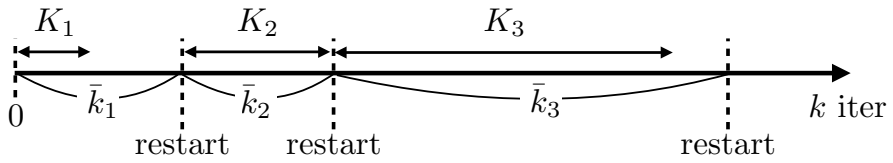


Figure 2.1.: Illustration of Maintaining Top-Speed Strategy ('mt'). A prohibition period of restart for K_i iteration are imposed after the i-th restart occurs. If the condition $\nabla f(\boldsymbol{\beta}^k)^\top(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}) < 0$ is satisfied after the period passed, then the $(i+1)$-th restart occurs. The next prohibition period is doubled, i.e., $K_{i+1} = 2K_i$. $\bar{k}_i (\geq K_i)$ denotes the number of iteration taken between the $(i-1)$-th and i-th restart, which will be used to convergence analysis in Section 3.3.2.

Nesterov [2013] proposed a restarting method, which has asymptotic linear convergence if g is strongly convex. Lin and Xiao [2015] showed that a similar restart technique can achieve the same convergence rate as Nesterov's scheme if f is strongly convex. Su et al. [2014] modeled the APG method as a second-order ordinary differential equation (ODE). They provided a *speed restarting* framework that ensures a linear convergence of the ODE with respect to time in the case that f is strongly convex and $g = 0$. To the best of our knowledge, however, none of the restarting schemes have convergence guarantees for a general non-strongly convex function.

2.2.4. Maintaining Top-Speed Strategy

The restarting strategy cancels out high momentum and prevents overshooting. In the neighborhood of an optimum, however, maintaining high momentum may be effective rather than restarting APG (see Figures 3.5 and 3.7 in Section 3.4), because large overshooting may not occur. Thus we put a prohibition period of restart for K_i iteration after the i-th restart occurs, where K_i increases as $K_i = 2K_{i-1}$ $(i = 2, 3, \ldots)$ and $K_1 \geq 2$. See Figure 2.1 for illustration. Precisely, letting $k = k_{re}$ be the iteration count at which the last restart occurs, we introduce the following step:

'mt': If $k \leq k_{re} + K_i$, then skip 're'. If restart occurs, then update $k_{re} \leftarrow k$, $K_{i+1} \leftarrow 2K_i$, and $i \leftarrow i + 1$.

While a similar strategy is taken in the experiment of [Monteiro et al., 2016], its convergence analysis is not provided. One of our contributions is to show that in fact the modification 'mt' of the restarting strategy can ensure the convergence rate of $O\big((\log k/k)^2\big)$ under a mild assumption. We will elaborate it in Section 3.3.2.

3. A Unified Optimization Method for Binary Classification

3.1. Overview

Binary classification is one of the most important problems in machine learning. Among the wide variety of binary classification models which have been proposed to date, the most popular ones include support vector machines (SVMs) [Cortes and Vapnik, 1995; Schölkopf et al., 2000] and logistic regression [Cox, 1958]. To achieve a good prediction performance, it is often important for the user to find a suitable model for a given dataset. However, the task of finding a suitable model is often time consuming and tedious as different classification models generally employ different formulations and algorithms. Moreover, the user might have to change not only the optimization algorithms but also solvers/software in order to solve different models. The goal of this chapter is to present a unified formulation for various classification models and also to design a fast universal algorithmic framework for solving different models. By doing so, one can simplify and speed up the process of finding the best classification model for a given dataset. We can also compare various classification methods in terms of computation time and prediction performance in the same platform.

In this chapter, we first propose a unified classification model which can express various models including C-SVM [Cortes and Vapnik, 1995], ν-SVM [Schölkopf et al., 2000], ℓ_2-SVM, logistic regression [Cox, 1958], MM-FDA, MM-MPM [Nath and Bhattacharyya, 2007], distance weighted discrimination [Marron et al., 2007]. The unified model is first formulated as an unconstrained ℓ_2-regularized loss minimization problem, which is further transformed into the problem of minimizing a convex objective function over a simple feasible region such as the intersection of a box and a hyperplane, truncated simplex, unit ball, and so on. Taking different loss functions (correspondingly different feasible regions in the transformed problem) will lead to different classification models. For example, when the feasible region is given by the intersection of the unit hypercube and a hyperplane, the unified formulation coincides with the well-known C-SVM or logistic regression, and when the region is given by a truncated simplex, the problem is the same as the ν-SVM.

It is commonly acknowledged that there is "no free lunch" in supervised learning in the sense that no single algorithm can outperform all other algorithms in all cases. Therefore, there can be no single "best" software for binary classification. However, by taking advantage of the above-mentioned unified formulation, we can design an efficient algorithm which is applicable to the various existing models mentioned in the last paragraph. Our proposed algorithm is based on the accelerated proximal gradient (APG)

method [Beck and Teboulle, 2009; Nesterov, 2005] and during the algorithm, only the procedure for computing the projection onto the associated feasible region differs for different models. In other words, by changing the computation of the projection, our algorithm can be applied to arbitrary classification models (i.e., arbitrary feasible region) without changing the optimization algorithmic framework. The great advantage of our algorithm is that most existing models have simple feasible regions, which make the computation of the projection easy and efficient.

The APG method is known to be one of the most efficient first-order methods theoretically. In order to make our APG based method practically efficient, we further employ various techniques such as backtracking line search [Beck and Teboulle, 2009; Scheinberg et al., 2014] and adaptive restarting strategies [O'Donoghue and Candès, 2015] to speed up the convergence of the algorithm. Our method incorporates several speed-up strategies while guaranteeing its convergence even for a general non-strongly convex function. Since our method is a further improvement of the standard APG method, we will call it as fast accelerated proximal gradient (FAPG) method. Our FAPG method improves several other restarting schemes [Nesterov, 2013; Lin and Xiao, 2015; Su et al., 2014] which have guaranteed convergence in the case when the objective function is strongly convex. To make our method even more efficient in practice, we simplify some steps in the implementation of our FAPG algorithm, though we can no longer ensure its theoretical convergence. To summarize, while our method has extensive generality, numerical experiments show that it performs stably and is highly competitive to specialized algorithms designed for specific classification models. Indeed, our method solved SVMs with a linear kernel substantially faster than LIBSVM [Chang and Lin, 2011] which implemented the SMO [Platt, 1998] and SeDuMi [Sturm, 1999] which implemented an interior-point method. Moreover, our FAPG method often run faster than the highly optimized LIBLINEAR [Fan et al., 2008] especially for large-scale datasets with feature dimension $n > 2000$. The FAPG method can be applied not only to the unified classification model but also to the general convex composite optimization problem such as ℓ_1-regularized classification models, which are solved faster than LIBLINEAR in most cases. It may be better to think of a stochastic variant of our method for further improvement and we leave it as a future research topic. We focus here on the deterministic one as the first trial to provide an efficient unified algorithm which is applicable to all well-known existing classification models.

The rest of this chapter is organized as follows. Section 3.2 presents a unified formulation of binary classification models. In Section 3.3, we provide solution methods for the unified formulation. We develop efficient algorithms for computing projections which are used in the APG method. Then we design our FAPG method combined with various techniques to speed-up its practical convergence. The iteration complexity of $O((\log k/k)^2)$ of our algorithm is also established, where k is the iteration counter. Numerical experiments are presented in Section 3.4.

3.2. A Unified Binary Classification Model

3.2.1. The Primal and Dual Formulations

Let $\mathcal{X} \subset \mathbb{R}^n$ be the input domain and $\{+1, -1\}$ be the set of the binary labels. Suppose that we have samples,

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \mathcal{X} \times \{+1, -1\}.$$

Define $M := \{1, \ldots, m\}$, $M_+ := \{i \in M \mid y_i = +1\}$, and $M_- := \{i \in M \mid y_i = -1\}$. Let $m_+ = |M_+|$ and $m_- = |M_-|$.

We compute $(\boldsymbol{w}, b)$ for a decision function $h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} - b$ using these samples and use $h(\boldsymbol{x})$ to predict the label for a new input point $\hat{\boldsymbol{x}} \in \mathcal{X}$. If $h(\hat{\boldsymbol{x}})$ is positive (resp. negative), then the label of $\hat{\boldsymbol{x}}$ is predicted to be $+1$ (resp. -1). Here we focus on linear learning models by using linear functions $h(\boldsymbol{x})$, but the discussions in this paper can be directly applied to non-linear kernel models [Schölkopf and Smola, 2002] using nonlinear maps $\phi(\boldsymbol{x})$ mapping $\boldsymbol{x}$ from the original space to a high dimensional space.

There are various binary classification models to compute $(\boldsymbol{w}, b)$ such as the support vector machines (SVMs), e.g. [Cortes and Vapnik, 1995; Schölkopf et al., 2000; Schölkopf and Smola, 2002], the margin maximized minimax probability machine (MM-MPM) [Nath and Bhattacharyya, 2007], the model based on Fisher's discriminant analysis (MM-FDA) [Bhattacharyya, 2004; Takeda et al., 2013], and the logistic regression [Cox, 1958].

Many binary classification models have the following formulation which consists of the sum of loss of each sample and a regularization term:

$$\min_{\boldsymbol{w}, b} \sum_{i=1}^{m} \ell\big(y_i(\boldsymbol{w}^\top \boldsymbol{x}_i - b)\big) + \frac{1}{C}\|\boldsymbol{w}\|_p^p, \tag{3.1}$$

where $\ell : \mathbb{R} \to \mathbb{R}$ is a proper, closed, and convex function; $C > 0$ is a parameter; and $\|\cdot\|_p$ is the p-norm with $p \in [1, \infty]$.

In this chapter, we focus on the following ℓ_2-regularized loss minimization problem:

$$\min_{\boldsymbol{w}, b} \mathcal{L}(\widetilde{\boldsymbol{A}}^\top \boldsymbol{w} - \boldsymbol{a}b) + \frac{1}{2C}\|\boldsymbol{w}\|_2^2, \tag{3.2}$$

where $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$ is a proper, closed, and convex function; $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{n \times l}$, $\boldsymbol{a} \in \mathbb{R}^l$, and $C > 0$ is a parameter. We will later show examples such that

- $l = m, \quad \widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{X}} := [y_1\boldsymbol{x}_1, y_2\boldsymbol{x}_2, \ldots, y_m\boldsymbol{x}_m], \quad \boldsymbol{a} = \boldsymbol{y}, \quad$ and
- $l = n, \quad \widetilde{\boldsymbol{A}} = \boldsymbol{I}, \quad \boldsymbol{a} = \boldsymbol{0}.$

Our algorithm to be described later can also be applied to a more general loss-regularized model with $p \in (1, \infty)$:

$$\min_{\boldsymbol{w}, b} \{\mathcal{L}(\widetilde{\boldsymbol{A}}^\top \boldsymbol{w} - \boldsymbol{a}b) \mid \|\boldsymbol{w}\|_p \le \lambda\}, \tag{3.3}$$

where $\lambda > 0$ is a parameter. We note that the condition $p \in (1, \infty)$ is required for our unified classification algorithm because it requires the smoothness of the dual norm of $\|\cdot\|_p$. However, if $p \in \{1, \infty\}$ and $\mathcal{L}$ is smooth, the APG method can be applied directly to the primal problem (3.3). See Appendix A.2 for ℓ_1-regularized problems, i.e., the case of $p = 1$.

The dual problems of (3.2) is given by

$$\min_{\boldsymbol{\alpha}} \Big\{ \mathcal{L}^*(-\boldsymbol{\alpha}) + \frac{C}{2} \|\widetilde{\boldsymbol{A}}\boldsymbol{\alpha}\|_2^2 \mid \boldsymbol{\alpha}^\top \boldsymbol{a} = 0 \Big\}, \tag{3.4}$$

where $\mathcal{L}^*(\boldsymbol{\alpha}) = \sup_{\boldsymbol{z}} \{\boldsymbol{\alpha}^\top \boldsymbol{z} - \mathcal{L}(\boldsymbol{z})\}$ is the convex conjugate of $\mathcal{L}$.

Since the problem (3.3) can be transformed as follows:

$$\begin{aligned}
&\min_{\boldsymbol{w}, b} \{\mathcal{L}(\widetilde{\boldsymbol{A}}^\top \boldsymbol{w} - b\boldsymbol{a}) \mid \|\boldsymbol{w}\|_p \le \lambda\} \\
&= \min_{\boldsymbol{w}, b, \boldsymbol{z}} \{\mathcal{L}(\boldsymbol{z}) \mid \|\boldsymbol{w}\|_p \le \lambda,\ \boldsymbol{z} = \widetilde{\boldsymbol{A}}^\top \boldsymbol{w} - b\boldsymbol{a}\} \\
&= \min_{\|\boldsymbol{w}\|_p \le \lambda, b, \boldsymbol{z}} \max_{\boldsymbol{\alpha}} \{\mathcal{L}(\boldsymbol{z}) + \boldsymbol{\alpha}^\top (\boldsymbol{z} - \widetilde{\boldsymbol{A}}^\top \boldsymbol{w} + b\boldsymbol{a})\},
\end{aligned}$$

its dual problem is derived as follows:

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \min_{\|\boldsymbol{w}\|_p \le \lambda, b, \boldsymbol{z}} \{\mathcal{L}(\boldsymbol{z}) + \boldsymbol{\alpha}^\top (\boldsymbol{z} - \widetilde{\boldsymbol{A}}^\top \boldsymbol{w} + b\boldsymbol{a})\} \\
&= -\min_{\boldsymbol{\alpha}} \Big\{ \max_{\boldsymbol{z}} \{ -\boldsymbol{\alpha}^\top \boldsymbol{z} - \mathcal{L}(\boldsymbol{z})\} + \max_{\|\boldsymbol{w}\|_p \le \lambda} \{\boldsymbol{\alpha}^\top \widetilde{\boldsymbol{A}}^\top \boldsymbol{w}\} + \max_{b} \{\boldsymbol{\alpha}^\top \boldsymbol{a} b\} \Big\} \\
&= -\min_{\boldsymbol{\alpha}} \{\mathcal{L}^*(-\boldsymbol{\alpha}) + \lambda \|\widetilde{\boldsymbol{A}}\boldsymbol{\alpha}\|_p^* \mid \boldsymbol{\alpha}^\top \boldsymbol{a} = 0\}.
\end{aligned} \tag{3.5}$$

where $\|\boldsymbol{z}\|_p^* = \sup_{\|\boldsymbol{w}\|_p \le 1} \{\boldsymbol{z}^\top \boldsymbol{w}\}$ is the dual norm of $\|\cdot\|_p$. It is known that $\|\cdot\|_p^* = \|\cdot\|_q$, where $q = p/(p-1)$. We note that the norms $\|\cdot\|_p$ and $\|\cdot\|_q^*$ are smooth for $p \in (1, \infty)$. The problem (3.5) can be seen as the Fenchel dual of (3.3).

As shown later, in all classification models, $\mathcal{L}^*$ is the sum of a smooth function and an indicator function of a simple set S for which the projection Π_S can be computed efficiently. Thus we can apply the APG method as an acceleration of the gradient projection method to the dual problems (3.4) and (3.5). By this construction, we can ensure the fast convergence rate of $F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^*) \le O(1/k^2)$ for various classification models in a unified way. In other word, the APG method can obtain a solution with desired accuracy $\epsilon > 0$ in $O(1/\sqrt{\epsilon})$ iteration.

There is an existing work [Zhou et al., 2010] which applies the APG method to the primal C-SVM, i.e., (3.2) with the hinge loss function. However, their method requires $O(1/\epsilon)$ iteration to obtain a solution with the desired accuracy $\epsilon > 0$ because it approximates the hinge loss by a smoothed function and making the approximation tighter requires extra iterations [Nesterov, 2005].

3.2.2. Relation to Existing Binary Classification Models

In this section, we show some specific examples of $\mathcal{L}$ and $\mathcal{L}^*$.

3.2.2.1. C-SVM

Let $\widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{X}}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (3.2). Let $\ell(z) = \max\{0, 1 - z\}$ be the hinge loss and $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^m \ell(z_i)$. Then the primal problem (3.2) is known as the standard C-SVM [Cortes and Vapnik, 1995]. Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^m \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} -\alpha & \text{if } \alpha \in [0,1] \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (3.4) can be reduced to

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2}\|\widetilde{\boldsymbol{X}}\boldsymbol{\alpha}\|_2^2 - \boldsymbol{\alpha}^\top \boldsymbol{e} + \delta_{S_C}(\boldsymbol{\alpha}), \tag{3.6}$$

where $S_C = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{0} \le \boldsymbol{\alpha} \le \boldsymbol{e}\}$.

3.2.2.2. ℓ_2-SVM

Let $\widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{X}}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (3.2). Let $\ell(z) = (\max\{0, 1 - z\})^2$ be the truncated squared loss and $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^m \ell(z_i)$. Then the primal problem (3.2) is known as the ℓ_2-SVM. Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^m \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} \frac{\alpha^2}{4} & \text{if } \alpha \ge 0 \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (3.4) can be reduced to

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2}\|\widetilde{\boldsymbol{X}}\boldsymbol{\alpha}\|_2^2 + \frac{\|\boldsymbol{\alpha}\|_2^2}{4} + \delta_{S_{\ell_2}}(\boldsymbol{\alpha}), \tag{3.7}$$

where $S_{\ell_2} = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{\alpha} \ge \boldsymbol{0}\}$.

3.2.2.3. Logistic regression

Let $\widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{X}}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (3.2). Let $\ell(z) = \log(1 + \exp(-z))$ be the logistic loss and $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^m \ell(z_i)$. Then the primal problem (3.2) is the logistic regression [Cox, 1958]. Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^m \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} \alpha\log(\alpha) + (1-\alpha)\log(1-\alpha) & (0 < \alpha < 1) \\ 0 & (\alpha = 0, 1) \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (3.4) can be reduced to

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2}\|\widetilde{\boldsymbol{X}}\boldsymbol{\alpha}\|_2^2 + \sum_{i:\alpha_i > 0} \alpha_i \log(\alpha_i) + \sum_{i:\alpha_i < 1} (1-\alpha_i)\log(1-\alpha_i) + \delta_{S_{\mathrm{LR}}}(\boldsymbol{\alpha}), \tag{3.8}$$

where $S_{\mathrm{LR}} = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{0} \le \boldsymbol{\alpha} \le \boldsymbol{e}\}$.

We should note that the gradient of the second and third terms in (3.8) is not Lipschitz continuous on $S_{\rm LR}$ and not defined at $\alpha_i = 0, 1$ $(i \in M)$. However, since it is known that its optimal solution $\boldsymbol{\alpha}^*$ satisfies $\mathbf{0} < \boldsymbol{\alpha}^* < \boldsymbol{e}$ [Yu et al., 2011], we can instead solve (3.8) by replacing $S_{\rm LR}$ by $S_{\rm LR}(\xi) := \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi \boldsymbol{e} \leq \boldsymbol{\alpha} \leq (1-\xi)\boldsymbol{e}\}$, where $\xi > 0$ is a small constant. The second and third terms in (3.8) are differentiable over $S_{\rm LR}(\xi)$. We will later show numerically that the resulting decision hyperplane is robust to a small deviation of ξ (see Table 3.6 in Section 3.4).

3.2.2.4. ν-SVM

Let $\widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{X}}$ and $\boldsymbol{a} = \boldsymbol{y}$ in (3.2). Let

$$\mathcal{L}(\boldsymbol{z}) = \min_{\rho} \Big\{ -\rho + \frac{1}{m\nu} \sum_{i=1}^{m} \max\{\rho - z_i, 0\} \Big\},$$

where $\nu \in (0, 1]$ is a given parameter. The function $\mathcal{L}(\boldsymbol{z})$ is the expected value of the upper ν-tail distribution, which is known as the conditional value-at-risk (CVaR) [Rockafellar and Uryasev, 2000]. Gotoh and Takeda [2005] pointed out that minimizing CVaR is equivalent to ν-SVM [Schölkopf et al., 2000], which is also known to be equivalent to C-SVM.

Since

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \begin{cases} 0 & \Big(\boldsymbol{e}^\top \boldsymbol{\alpha} = 1 \text{ and } \boldsymbol{\alpha} \in \Big[0, \frac{1}{m\nu}\Big]^m\Big) \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem is given by

$$\min_{\boldsymbol{\alpha}} \ \frac{C}{2} \|\widetilde{\boldsymbol{X}} \boldsymbol{\alpha}\|_2^2 + \delta_{S_\nu}(\boldsymbol{\alpha}), \tag{3.9}$$

where $S_\nu = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \boldsymbol{\alpha}^\top \boldsymbol{e} = 1,\ \mathbf{0} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu} \boldsymbol{e}\}$. Note that any positive value for C does not change the optimal solution of (3.9), and therefore, we can set $C = 1$. There exists a valid range $(\nu_{min}, \nu_{max}] \subseteq (0, 1]$ for ν, where ν_{min} is the infimum of $\nu > 0$ such that $\widetilde{\boldsymbol{X}} \boldsymbol{\alpha}^* \neq \mathbf{0}$, and $\nu_{max} := \frac{2 \max\{m_+, m_-\}}{m}$ is the maximum of $\nu \leq 1$ for which $S_\nu \neq \phi$. Since the parameter ν takes a value in the finite range $(\nu_{min}, \nu_{max}]$, choosing the parameter value for ν-SVM is often easier than that for C-SVM.

3.2.2.5. Distance Weighted Discrimination (DWD)

Let $\widetilde{\boldsymbol{A}} = \widetilde{\boldsymbol{X}}$, $\boldsymbol{a} = \boldsymbol{y}$, $\lambda = 1$, and $p = 2$ in (3.3). For a positive parameter ν, let $\ell(z)$ be the function defined by

$$\begin{aligned} \ell(z) &= \min_{\rho} \Big\{ \frac{1}{\rho} + \frac{1}{m\nu}(\rho - z) \mid \rho \geq z,\ \rho \geq \sqrt{m\nu} \Big\} \\ &= \begin{cases} \frac{1}{z} & \text{if } z \geq \sqrt{m\nu} \\ \frac{2\sqrt{m\nu} - z}{m\nu} & \text{otherwise} \end{cases}. \end{aligned}$$

Consider $\mathcal{L}(\boldsymbol{z}) = \sum_{i=1}^{m} \ell(z_i)$. This loss function is used in the distance weighted discrimination (DWD) [Marron et al., 2007] which is proposed as an alternative to ν-SVM. Since we have

$$\mathcal{L}^*(-\boldsymbol{\alpha}) = \sum_{i=1}^{m} \ell^*(-\alpha_i), \quad \text{where} \quad \ell^*(-\alpha) = \begin{cases} -2\sqrt{\alpha} & (0 \leq \alpha \leq \frac{1}{m\nu}) \\ +\infty & \text{otherwise,} \end{cases}$$

the dual problem (3.4) can be equivalently reduced to

$$\min_{\boldsymbol{\alpha}} \Big\{ \|\widetilde{\boldsymbol{X}}\boldsymbol{\alpha}\|_2 - 2\sum_{i=1}^{m} \sqrt{\alpha_i} + \delta_{S_{\mathrm{DWD}}}(\boldsymbol{\alpha}) \Big\}, \tag{3.10}$$

where $S_{\mathrm{DWD}} = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \mathbf{0} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu}\boldsymbol{e}\}$.

As in the case of the logistic regression, the second term in (3.10) is not differentiable at $\alpha_i = 0$ ($i \in M$). However, since it is known that there exists an optimal solution $\boldsymbol{\alpha}^*$ such that $\alpha_i^* > 0$ ($i \in M$), we can solve (3.10) by replacing S_{DWD} by $S_{\mathrm{DWD}}(\xi) = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi\boldsymbol{e} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu}\boldsymbol{e}\}$ where $\xi > 0$ is a small constant. The second term in (3.10) is differentiable over $S_{\mathrm{DWD}}(\xi)$.

3.2.2.6. Extended Fisher's discriminant analysis (MM-FDA)

The well-known Fisher's discriminant analysis (FDA) uses a decision function to maximize the ratio of the variance between the classes to the variance within the classes. Let $\bar{\boldsymbol{x}}_o$ and Σ_o, $o \in \{+, -\}$, be the mean vectors and the positive definite covariance matrices of $\boldsymbol{x}_i$, $i \in M_o$, respectively. Then FDA is formulated as follows:

$$\max_{\boldsymbol{w}} \ \frac{\big(\boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-)\big)^2}{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}}.$$

Its optimal solution is given by

$$\boldsymbol{w}^* = (\Sigma_+ + \Sigma_-)^{-1}(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-).$$

Let $\widetilde{\boldsymbol{A}} = \boldsymbol{I}$ and $\boldsymbol{a} = \mathbf{0}$ in (3.2). Here we consider the mean-standard deviation type of risk corresponding to FDA:

$$\mathcal{L}(\boldsymbol{w}) = -\boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + \kappa\sqrt{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}},$$

where $\kappa > 0$ is a parameter. Since

$$\begin{aligned} \mathcal{L}^*(-\boldsymbol{\alpha}) &= \sup_{\boldsymbol{w}} \Big\{ -\boldsymbol{\alpha}^\top\boldsymbol{w} + \boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) - \kappa\sqrt{\boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)\boldsymbol{w}} \Big\} \\ &= \sup_{\boldsymbol{w}} \min_{\|\boldsymbol{\mu}\|_2 \leq \kappa} \Big\{ -\boldsymbol{\alpha}^\top\boldsymbol{w} + \boldsymbol{w}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + \boldsymbol{w}^\top(\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{\mu} \Big\} \\ &= \min_{\boldsymbol{\mu}} \{\delta_{S_{\mathrm{FDA}}}(\boldsymbol{\mu}) \mid \boldsymbol{\alpha} = (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{\mu}\}, \end{aligned}$$

where $S_{\mathrm{FDA}} = \{\boldsymbol{\mu} \in \mathbb{R}^n \mid \|\boldsymbol{\mu}\|_2 \leq \kappa\}$, the dual problem (3.4) can be reduced to

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu}} \Big\{\delta_{S_{\mathrm{FDA}}}(\boldsymbol{\mu}) + \frac{C}{2}\|\boldsymbol{\alpha}\|_2^2 \mid \boldsymbol{\alpha} = (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{\mu}\Big\},$$

which is equivalent to

$$\min_{\boldsymbol{\mu}} \{\delta_{S_{\mathrm{FDA}}}(\boldsymbol{\mu}) + \big\|(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{\mu}\big\|_2^2\}. \tag{3.11}$$

Takeda et al. [2013] showed that

$$\kappa_{\max} := \left| \begin{array}{ll} \inf_{\boldsymbol{\mu},\kappa} & \kappa \\ \text{s.t.} & \min_{\boldsymbol{\mu}} \{\big\|(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\boldsymbol{\mu}\big\|_2 + \delta_{S_{\mathrm{FDA}}}(\boldsymbol{\mu})\} = 0 \end{array} \right.$$

is equivalent to FDA. Hence (3.11) can be seen as an extension of FDA. We will refer it as MM-FDA in this thesis.

3.2.2.7. Maximum Margin Minimax Probability Machine (MM-MPM)

Let $\widetilde{\boldsymbol{A}} = \boldsymbol{I}$ and $\boldsymbol{a} = \boldsymbol{0}$ in (3.2). We consider a class-wise mean-standard deviation type of risk:

$$\mathcal{L}(\boldsymbol{w}) = \Big(-\boldsymbol{w}^\top \bar{\boldsymbol{x}}_+ + \kappa\sqrt{\boldsymbol{w}^\top \Sigma_+ \boldsymbol{w}}\Big) - \Big(-\boldsymbol{w}^\top \bar{\boldsymbol{x}}_- + \kappa\sqrt{\boldsymbol{w}^\top \Sigma_- \boldsymbol{w}}\Big).$$

Similar to MM-FDA, we have

$$\begin{aligned} \mathcal{L}^*(\boldsymbol{w}) &= \sup_{\boldsymbol{w}} \Big\{ -\boldsymbol{\alpha}^\top \boldsymbol{w} + \Big(\boldsymbol{w}^\top \bar{\boldsymbol{x}}_+ - \kappa\sqrt{\boldsymbol{w}^\top \Sigma_+ \boldsymbol{w}}\Big) - \Big(\boldsymbol{w}^\top \bar{\boldsymbol{x}}_- - \kappa\sqrt{\boldsymbol{w}^\top \Sigma_- \boldsymbol{w}}\Big)\Big\} \\ &= \sup_{\boldsymbol{w}} \min_{\substack{\|\boldsymbol{\mu}_+\|_2 \leq \kappa \\ \|\boldsymbol{\mu}_-\|_2 \leq \kappa}} \Big\{ -\boldsymbol{\alpha}^\top \boldsymbol{w} + (\boldsymbol{w}^\top \bar{\boldsymbol{x}}_+ + \boldsymbol{w}^\top \Sigma_+^{1/2}\boldsymbol{\mu}_+) - (\boldsymbol{w}^\top \bar{\boldsymbol{x}}_- + \boldsymbol{w}^\top \Sigma_-^{1/2}\boldsymbol{\mu}_-)\Big\} \\ &= \min_{\boldsymbol{\mu}_+,\boldsymbol{\mu}_-} \{\delta_{S_{\mathrm{MPM}}}(\boldsymbol{\mu}_+, \boldsymbol{\mu}_-) \mid \boldsymbol{\alpha} = (\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\boldsymbol{\mu}_+) - (\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\boldsymbol{\mu}_-)\}, \end{aligned}$$

where $S_{\mathrm{MPM}} = \{(\boldsymbol{\mu}_+, \boldsymbol{\mu}_-) \in \mathbb{R}^n \times \mathbb{R}^n \mid \|\boldsymbol{\mu}_+\|_2 \leq \kappa,\ \|\boldsymbol{\mu}_-\|_2 \leq \kappa\}$. Thus the dual problem (3.4) can be reduced to

$$\min_{\boldsymbol{\alpha},\boldsymbol{\mu}_+,\boldsymbol{\mu}_-} \Big\{\delta_{S_{\mathrm{MPM}}}(\boldsymbol{\mu}_+, \boldsymbol{\mu}_-) + \frac{C}{2}\|\boldsymbol{\alpha}\|_2^2 \mid \boldsymbol{\alpha} = (\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\boldsymbol{\mu}_+) - (\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\boldsymbol{\mu}_-)\Big\},$$

which is equivalent to

$$\min_{\boldsymbol{\mu}_+,\boldsymbol{\mu}_-} \{\delta_{S_{\mathrm{MPM}}}(\boldsymbol{\mu}_+, \boldsymbol{\mu}_-) + \big\|(\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\boldsymbol{\mu}_+) - (\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\boldsymbol{\mu}_-)\big\|_2^2\}. \tag{3.12}$$

The last problem (3.12) is equivalent to the dual of the maximum margin minimax probability machine (MM-MPM) [Nath and Bhattacharyya, 2007].

3.3. Algorithms for Unified Binary Classification

In this section, we first provide an efficient vector projection computation in order to apply the APG method to the unified formulation of the binary classification models. After that, we develop a fast APG (FAPG) method and show its convergence.

3.3.1. Vector Projection Computation

When applying the APG method to a constrained optimization problem over S, the projection Π_S onto S appears at Step 1. We need to change the computation of the projection Π_S depending on the definition of S. In this section, we provide efficient projection computations for S_C, S_{ℓ_2}, $S_{\mathrm{LR}}(\xi)$, S_ν, $S_{\mathrm{DWD}}(\xi)$, S_{MPM}, and S_{FDA}.

3.3.1.1. A Bisection Method for Projection

Many models shown in Section 3.2 have a linear equality and box constraints. Here we consider the set $S = \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{e} = r,\ l \leq \alpha_i \leq u\ (\forall i \in M)\}$ and provide an efficient algorithm for computing the projection $\Pi_S(\bar{\boldsymbol{\alpha}})$:

$$\min_{\boldsymbol{\alpha}} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \mid \boldsymbol{\alpha}^\top \boldsymbol{e} = r, \quad l \leq \alpha_i \leq u \ \ (i \in M) \Big\}. \tag{3.13}$$

We assume that (3.13) is feasible. One way to solve (3.13) is to use breakpoint search algorithms (e.g., [Helgason et al., 1980; Kiwiel, 2008; Duchi et al., 2008]). They are exact algorithms and have linear time complexity of $O(m)$. Helgason et al. [1980] and Kiwiel [2008] developed the breakpoint search algorithms for the continuous quadratic knapsack problem (CQKP) which involves the projection (3.13) as a special case. By integrating the breakpoint search algorithm and the red-black tree data structure, Duchi et al. [2008] developed an efficient update algorithm of the gradient projection method when the gradient is sparse.

In this section, we provide a numerical algorithm for (3.13). Although its worst case complexity is $O\Big(m \log\big(\frac{\bar{\alpha}_{\max} - \bar{\alpha}_{\min}}{\epsilon'}\big)\Big)$, where $\bar{\alpha}_{\max} = \max\{\bar{\alpha}_i \mid i \in M\}$ and $\bar{\alpha}_{\min} = \min\{\bar{\alpha}_i \mid i \in M\}$, it often requires less time to compute a solution $\hat{\boldsymbol{\alpha}}$ such that $\|\hat{\boldsymbol{\alpha}} - \Pi_S(\bar{\boldsymbol{\alpha}})\|_\infty < \epsilon' = u$ in practice, where $u \approx 2.22 \times 10^{-16}$ is the IEEE 754 double precision. Note that the error u can occur even when using the breakpoint search algorithm because u is the supremum of the relative error due to rounding in the floating point number system. Thus, it is sufficient to choose $\epsilon' = u$ as the stopping tolerance for our algorithm in practice. [1]

It is known that the projection (3.13) can be reduced to finding the root of an one dimensional monotone function.

[1] Another practical way is to decrease ϵ', i.e., to improve the accuracy of the projection progressively, as APG iterates. The APG method with the inexact computation [Jiang et al., 2012] also share the same iteration complexity $O(1/k^2)$ as the exact counterpart.

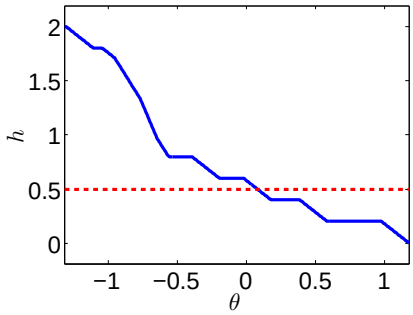


Figure 3.1.: Illustration of the function $h(\theta) := \sum_{i \in M} \alpha_i(\theta)$. The function h is piecewise linear.

Lemma 3.3.1 (e.g. [Helgason et al., 1980])**.** *Suppose that the problem* (3.13) *is feasible. Let*

$$\alpha_i(\theta) = \min\Big\{ \max\{l, \bar{\alpha}_i - \theta\}, u\Big\}, \quad i \in M$$

and let

$$h(\theta) = \sum_{i \in M} \alpha_i(\theta).$$

The following statements hold:

1. *$h(\theta)$ is a continuous, non-increasing, and piecewise linear function which has breakpoints at $\bar{\alpha}_i - u$ and $\bar{\alpha}_i - l$ $(i \in M)$.*
2. *There exists $\theta^* \in (\bar{\alpha}_{\min} - \frac{r}{m}, \bar{\alpha}_{\max} - \frac{r}{m})$ such that $h(\theta^*) = r$.*
3. *Let $\hat{\alpha}_i = \alpha_i(\theta^*)$, $i \in M$. Then $\hat{\boldsymbol{\alpha}}$ is an optimal solution of* (3.13)*.*

An example of $h(\theta)$ is illustrated in Figure 3.1. To solve $h(\theta) = r$, the breakpoint search uses the binary search to find two adjacent breakpoints that contain a solution θ^* between them. Then the exact solution θ^* can be obtained by linear interpolation of the two breakpoints.

Instead of binary search, we employ the bisection method to solve the equation $h(\theta) = r$.

The Bisection Algorithm for Projection

Step 0. Set $\theta^u \leftarrow \bar{\alpha}_{\max} - \frac{r}{m}$ and $\theta^l \leftarrow \bar{\alpha}_{\min} - \frac{r}{m}$.

Step 1. Set $\hat{\theta} \leftarrow (\theta^u + \theta^l)/2$

Step 2. Compute $h(\hat{\theta})$.

Step 3. If $h(\hat{\theta}) = r$, then output $\boldsymbol{\alpha}$ with $\alpha_i \leftarrow \alpha_i(\hat{\theta})$. Else if $h(\hat{\theta}) < r$, then set $\theta^u \leftarrow \hat{\theta}$. Else if $h(\hat{\theta}) > r$, then set $\theta^l \leftarrow \hat{\theta}$.

Step 4. If $|\theta^u - \theta^l| < \epsilon'$, then output $\boldsymbol{\alpha}$ with $\alpha_i \leftarrow \alpha_i(\hat{\theta})$. Else, go to Step 1.

All steps require at most $O(m)$ operations and the maximum number of iteration is $\lceil \log(\frac{\bar{\alpha}_{\max} - \bar{\alpha}_{\min}}{\epsilon'}) \rceil$. Thus, the computational complexity of the bisection algorithm is at most $O\big(m \log(\frac{\bar{\alpha}_{\max} - \bar{\alpha}_{\min}}{\epsilon'})\big)$.

As the bisection method narrows the interval (θ^l, θ^u), some of the terms $\alpha_i(\theta)$, $i \in M$, can be set to l, $\bar{\alpha}_i - \theta$, or u for $\theta \in (\theta^l, \theta^u)$. By exploiting the property of $\alpha_i(\theta)$, we can refine the computation of $h(\hat{\theta})$ in Step 3 by avoiding recalculation of certain sums. Moreover, we can often obtain an exact solution. Let us divide M into the following three disjoint sets for given θ^l and θ^u satisfying $\theta^l < \theta^u$:

$$\begin{aligned} U &:= \{i \in M \mid \bar{\alpha}_i - \theta^u \geq u \quad (\text{i.e., } \alpha_i(\theta) = u, \quad \forall \theta \in [\theta^l, \theta^u])\} \\ L &:= \{i \in M \mid \bar{\alpha}_i - \theta^l \leq l \quad (\text{i.e., } \alpha_i(\theta) = l, \quad \forall \theta \in [\theta^l, \theta^u])\} \\ I &:= M \backslash (U \cup L). \end{aligned}$$

If $\bar{\alpha}_i - \theta^u \geq l$ and $\bar{\alpha}_i - \theta^l \leq u$ for all $i \in I$, then

$$\theta = \Big(|U|u + |L|l + \sum_{i \in I} \alpha_i - r\Big)/|I| \tag{3.14}$$

is an exact solution. If $I \neq \phi$, then we also divide I into the following three disjoint sets for given $\hat{\theta} \in (\theta^l, \theta^u)$:

$$\begin{aligned} I^U &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \geq u, \quad (\text{i.e., } \alpha_i(\theta) = u \quad \forall \theta \in [\theta^l, \hat{\theta}])\} \\ I^L &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \leq l, \quad (\text{i.e., } \alpha_i(\theta) = l \quad \forall \theta \in [\hat{\theta}, \theta^u])\} \\ I^C &= I \backslash (I^U \cup I^L) \end{aligned}$$

Then we have

$$h(\hat{\theta}) = \sum_{i \in M} \alpha_i(\hat{\theta}) = \underbrace{|U|u}_{s_u} + \underbrace{|I^U|u}_{\Delta s_u} + \underbrace{|L|l}_{s_l} + \underbrace{|I^L|l}_{\Delta s_l} + \underbrace{\sum_{i \in I^C} \alpha_i}_{s_c} - |I^C|\hat{\theta}.$$

By storing the value of each terms, we can reduce the number of additions. The resulting algorithm can be described as Algorithm 3.1.

Compared to the existing breakpoint search algorithms, our bisection method is advantageous since it can avoid the computation of breakpoints and the comparison between them. It is often faster than the breakpoint search algorithms in practice (see Table 3.2 in Section 3.4). In many cases, our algorithm can obtain the exact solution by (3.14).

Remark 3.3.1. *Mimicking [Kiwiel, 2008], we can divide the set M more finely and reduce the recalculation of certain sums as shown in Appendix A.1. However, Algorithm 3.1 allows for simpler implementation and runs faster on our computer systems.*

In the followings, we use Algorithm 3.1 to compute the projection onto S_C, S_{ℓ_2}, $S_{\mathrm{LR}}(\xi)$, $S_{\mathrm{DWD}}(\xi)$, and S_ν.

Algorithm 3.1 The bisection algorithm for (3.13).

INPUT: $\bar{\boldsymbol{\alpha}},\ r,\ \boldsymbol{l},\ \boldsymbol{u},\ \epsilon' > 0$ OUTPUT: $\boldsymbol{\alpha}$

INITIALIZE: $I \leftarrow M,\ s_u \leftarrow s_l \leftarrow s_c \leftarrow 0,$
$\theta^u \leftarrow \bar{\alpha}_{\max} - \frac{r}{m},\ \theta^l \leftarrow \bar{\alpha}_{\min} - \frac{r}{m}$ # Step 1

while $|\theta^u - \theta^l| > \epsilon'$ **do**

$\hat{\theta} \leftarrow \frac{\theta^u + \theta^l}{2}$ # Step 2

$\hat{u} \leftarrow u + \hat{\theta}, \quad \hat{l} \leftarrow l + \hat{\theta}$

$I^U \leftarrow \{i \in I \mid \bar{\alpha}_i \geq \hat{u}\}, \quad I^L \leftarrow \{i \in I \mid \bar{\alpha}_i \leq \hat{l}\}$ # Step 3

$I^C \leftarrow I \backslash (I^U \cup I^L)$

$\Delta s_u \leftarrow |I^U| u, \quad \Delta s_l \leftarrow |I^L| l, \quad s_c \leftarrow \sum_{i \in I^C} \alpha_i$

$val \leftarrow s_u + \Delta s_u + s_l + \Delta s_l + s_c - |I^C| \hat{\theta}$

if $val < r$ **then** # Step 4

$\theta^u \leftarrow \hat{\theta}, \quad I \leftarrow I^C \cup I^L$

$s_u \leftarrow s_u + \Delta s_u$

else if $val > r$ **then**

$\theta^l \leftarrow \hat{\theta}, \quad I \leftarrow I^C \cup I^U$

$s_l \leftarrow s_l + \Delta s_l$

else

break

end if

if $\bar{\alpha}_i - \theta^u \geq l$ and $\bar{\alpha}_i - \theta^l \leq u \quad \forall i \in I$ **then**

$\hat{\theta} \leftarrow (s_u + s_l + s_c - r)/|I|$

break

end if

end while

$\alpha_i \leftarrow \alpha_i(\hat{\theta}),\ \forall i \in M$ # Step 5

3.3.1.2. Projection for C-SVM, Logistic Regression, ℓ_2-SVM, DWD

The projection for C-SVM, logistic regression, ℓ_2-SVM, and DWD can be formulated as follows:

$$\min_{\boldsymbol{\alpha}} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \mid \boldsymbol{y}^\top \boldsymbol{\alpha} = 0, \quad l \le \alpha_i \le u \quad (i \in M) \Big\}, \tag{3.15}$$

where $(l, u) = (0, 1)$ for C-SVM, $(l, u) = (0, \infty)$ for ℓ_2-SVM, $(l, u) = (\xi, 1-\xi)$ for logistic regression, and $(l, u) = (\xi, \frac{1}{m\nu})$ for DWD. By transforming the variable $\boldsymbol{\alpha}$ and the vector $\bar{\boldsymbol{\alpha}}$ to

$$\beta_i = \begin{cases} \alpha_i & \text{if } y_i = 1 \\ -\alpha_i + l + u & \text{otherwise} \end{cases} \quad \text{and} \quad \bar{\beta}_i = \begin{cases} \bar{\alpha}_i & \text{if } y_i = 1 \\ -\bar{\alpha}_i + l + u & \text{otherwise,} \end{cases}$$

respectively, the problem (3.15) can be reduced to

$$\min_{\boldsymbol{\beta}} \Big\{ \frac{1}{2}\|\boldsymbol{\beta} - \bar{\boldsymbol{\beta}}\|_2^2 \mid \boldsymbol{\beta}^\top \boldsymbol{e} = (l+u)m_-, \quad l \le \beta_i \le u \quad (i \in M) \Big\}.$$

The last problem can be solved by Algorithm 3.1.

3.3.1.3. Projection for ν-SVM

In applying the APG method to ν-SVM, we shall be concerned with the projection $\Pi_{S_\nu}(\bar{\boldsymbol{\alpha}})$:

$$\min_{\boldsymbol{\alpha}} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha} - \bar{\boldsymbol{\alpha}}\|_2^2 \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0, \ \boldsymbol{\alpha}^\top \boldsymbol{e} = 1, \ \boldsymbol{0} \le \boldsymbol{\alpha} \le \frac{1}{m\nu}\boldsymbol{e} \Big\}. \tag{3.16}$$

Let $\boldsymbol{\alpha}_+$ and $\boldsymbol{\alpha}_-$ be the subvectors of $\boldsymbol{\alpha}$ corresponding to the label $+1$ and -1, respectively. Let $\boldsymbol{e}_+$ and $\boldsymbol{e}_-$ be subvectors of $\boldsymbol{e}$ with size m_+ and m_-. From the fact that the conditions $\boldsymbol{\alpha}^\top \boldsymbol{y} = 0$ and $\boldsymbol{\alpha}^\top \boldsymbol{e} = 1$ are equivalent to $\boldsymbol{\alpha}_o^\top \boldsymbol{e}_o = \frac{1}{2}$ $(o \in \{+, -\})$, the problem (3.16) can be decomposed into the following two problems:

$$\min_{\boldsymbol{\alpha}_o} \Big\{ \frac{1}{2}\|\boldsymbol{\alpha}_o - \bar{\boldsymbol{\alpha}}_o\|_2^2 \mid \boldsymbol{\alpha}^\top \boldsymbol{e}_o = \frac{1}{2}, \ \boldsymbol{0} \le \boldsymbol{\alpha}_o \le \frac{1}{m\nu}\boldsymbol{e}_o \Big\}, \ o \in \{+, -\}. \tag{3.17}$$

Algorithm 3.1 can be applied to solve (3.17).

3.3.1.4. Projection for MM-MPM and MM-FDA

The projection $\Pi_{S_{\mathrm{FDA}}}(\bar{\boldsymbol{\alpha}})$ of $\bar{\boldsymbol{\alpha}}$ onto $S_{\mathrm{FDA}} = \{\boldsymbol{\alpha} \in \mathbb{R}^n \mid \|\boldsymbol{\alpha}\|_2 \le \kappa\}$ is easy to compute; We have

$$\Pi_{S_{\mathrm{FDA}}}(\bar{\boldsymbol{\alpha}}) = \min\Big\{1, \ \frac{\kappa}{\|\bar{\boldsymbol{\alpha}}\|_2}\Big\}\bar{\boldsymbol{\alpha}}.$$

Similarly, the projection $\Pi_{S_{\mathrm{MPM}}}(\bar{\boldsymbol{\alpha}}_+, \bar{\boldsymbol{\alpha}}_-)$ for MM-MPM can be computed as follows:

$$\Pi_{S_{\mathrm{MPM}}}(\bar{\boldsymbol{\alpha}}_+, \bar{\boldsymbol{\alpha}}_-) = \Big(\min\Big\{1, \frac{\kappa}{\|\bar{\boldsymbol{\alpha}}_+\|_2}\Big\}\bar{\boldsymbol{\alpha}}_+, \ \min\Big\{1, \frac{\kappa}{\|\bar{\boldsymbol{\alpha}}_-\|_2}\Big\}\bar{\boldsymbol{\alpha}}_- \Big).$$

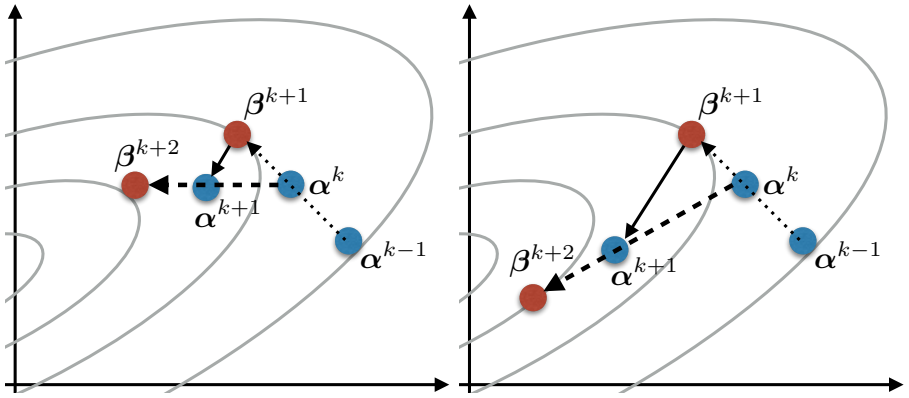


Figure 3.2.: Effects of the value of L_k to the momentum $\frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ (left: L_k is large, right: L_k is small). A smaller value of L_k gives a larger step size at Step 1 as illustrated by the solid lines. This results in high momentum at Step 3 as illustrated by the dashed lines.

3.3.2. A Fast APG (FAPG) Method with Convergence Guarantee

We have shown several strategies to speed-up the APG method in Section 2.2. While the convergence proof has been shown for the backtracking 'bt' and decreasing 'dec' strategy, the convergence for the restart 're' and maintaining top-speed 'mt' strategy is unknown. In this section, we further propose a practical stabilization strategy and develop a fast APG (FAPG) method by combining all these techniques. We then show its convergence.

3.3.2.1. A Fast APG with Stabilization

Decreasing L ('dec') and restarting strategies ('re') shown in Section 2.2 are effective to speed-up the practical convergence of the APG method, especially in early iterations and near the optimal solution, respectively (see Figures 3.5 and 3.9 in Section 3.4). However, they have opposing effects. Decreasing L_k enlarges the stepsize $\frac{1}{L_k}$, which extends $\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}$. This inherently induces high momentum (as illustrated in Figure 3.2). On the other hand, the restarting strategy cancels out high momentum. Hence combining them triggers the restart frequently and makes the APG method unstable. In order to avoid the instability, it would be necessary to reduce the η_d (rate of decreasing L_k) near the optimum. Considering that the restart would occur when the sequences approach the optimum, we take the following strategy to reduce the value of η_d with a constant $\delta \in (0, 1)$:

'st': Update $\eta_d \leftarrow \delta \cdot \eta_d + (1 - \delta) \cdot 1$ when the restart occurs.

Algorithm 3.2 is our FAPG method which combines these strategies. The difference from Algorithm 2.1 is the "if block" after Step 3.

Algorithm 3.2 A fast APG (FAPG) method with speeding-up strategies.

Input: f, ∇f, g, $\text{prox}_{g,L}$, $\epsilon > 0$, $L_1 = L_0 > 0$, $\eta_u > 1$, $\eta_d > 1$, $k_{max} > 0$, $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0 = \boldsymbol{\alpha}^{-1}$, $K_1 \geq 2$, $\delta \in (0, 1)$

Output: $\boldsymbol{\alpha}^k$

Initialize: $t_1 \leftarrow 1,\ t_0 \leftarrow 0,\ i \leftarrow 1,\ k_{re} \leftarrow 0$

for $k = 1, \ldots, k_{max}$ **do**

  $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # Step 1

  while $F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ **do**

    $L_k \leftarrow \eta_u L_k$ # 'bt'

    $t_k \leftarrow \frac{1+\sqrt{1+4(L_k/L_{k-1})t_{k-1}^2}}{2}$

    $\boldsymbol{\beta}^k \leftarrow \boldsymbol{\alpha}^{k-1} + \frac{t_{k-1}-1}{t_k}(\boldsymbol{\alpha}^{k-1} - \boldsymbol{\alpha}^{k-2})$

    $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # 'bt'

  end while

  if $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\| < \epsilon$ **then**

    break

  end if

  $L_{k+1} \leftarrow L_k/\eta_d$ # 'dec'

  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4(L_{k+1}/L_k)t_k^2}}{2}$ # Step 2'

  $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k-1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ # Step 3

  if $k > k_{re} + K_i$ and $\langle \nabla f(\boldsymbol{\beta}^k), \boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}\rangle + g(\boldsymbol{\alpha}^k) - g(\boldsymbol{\alpha}^{k-1}) > 0$ **then**

    $k_{re} \leftarrow k,\quad K_{i+1} \leftarrow 2K_i,\quad i \leftarrow i+1$ # 'mt'

    $\eta_d \leftarrow \delta \cdot \eta_d + (1-\delta)\cdot 1$ # 'st'

    $t_{k+1} \leftarrow 1,\quad t_k \leftarrow 0,\quad \boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^{k-1},\quad \boldsymbol{\alpha}^k \leftarrow \boldsymbol{\alpha}^{k-1}$ # 're'

  end if

end for

3.3.2.2. Convergence Analysis

Now we analyse the convergence rate of Algorithm 3.2. Let $\bar{k}_i$ denotes the number of iteration taken between the $(i-1)$-th and the i-th restart (see Figure 2.1 for illustration). In the followings, we denote $\boldsymbol{\alpha}^k$ as $\boldsymbol{\alpha}^{(j,k_{j+1})}$ if $k = \sum_{i=1}^{j} \bar{k}_i + k_{j+1}$ ($\bar{k}_{j+1} \geq k_{j+1} \geq 0$). In other word, $\boldsymbol{\alpha}^{(i,k_{i+1})}$ denotes a point obtained at the k_{i+1}-th iteration counting from the i-th restart. Note that $\boldsymbol{\alpha}^{(i+1,0)} = \boldsymbol{\alpha}^{(i,\bar{k}_{i+1}-1)}$ ($\forall i \geq 0$). We will frequently switch the notations $\boldsymbol{\alpha}^k$ and $\boldsymbol{\alpha}^{(i,k_{i+1})}$ for notational convenience.

The following lemma is useful to evaluate the function F.

Lemma 3.3.2 (from [Beck and Teboulle, 2009])**.** *If $F(T_L(\boldsymbol{\beta})) \leq Q_L(T_L(\boldsymbol{\beta}); \boldsymbol{\beta})$, then*

$$F(T_L(\boldsymbol{\beta})) - F(\boldsymbol{\alpha}) \leq \frac{L}{2}\{\|\boldsymbol{\beta} - \boldsymbol{\alpha}\|_2^2 - \|T_L(\boldsymbol{\beta}) - \boldsymbol{\alpha}\|_2^2\}, \quad \forall \boldsymbol{\alpha} \in \mathbb{R}^d.$$

Using Lemma 3.3.2, we obtain the following key lemma.

Lemma 3.3.3. *Let the sequence $\{\boldsymbol{\alpha}^k\}_{k=1}^{\infty} (\equiv \{\boldsymbol{\alpha}^{(i,k_{i+1})}\})$ be generated by Algorithm 3.2. The following inequality holds:*

$$F(\boldsymbol{\alpha}^k) \leq F(\boldsymbol{\alpha}^0), \quad \forall k \geq 1.$$

Moreover,

$$F(\boldsymbol{\alpha}^{(i,k_{i+1})}) \leq F(\boldsymbol{\alpha}^{(i,0)}), \quad \forall i \geq 0 \;\; \textit{and} \;\; \forall k_{i+1} \in \{0, 1, 2, \ldots, \bar{k}_{i+1}\},$$

and

$$F(\boldsymbol{\alpha}^{(i+1,0)}) \leq F(\boldsymbol{\alpha}^{(i,0)}), \quad \forall i \geq 0.$$

Proof. First, assume that the restart does not occur (i.e., the steps 're', 'st', and 'mt' are not executed) until the k-th iteration. From Lemma 3.3.2, we have

$$F(\boldsymbol{\alpha}^1) - F(\boldsymbol{\alpha}^0) \leq \frac{L_1}{2}\{\|\boldsymbol{\beta}^1 - \boldsymbol{\alpha}^0\|_2^2 - \|\boldsymbol{\alpha}^1 - \boldsymbol{\alpha}^0\|_2^2\} \;=\; -\frac{L_1}{2}\|\boldsymbol{\alpha}^1 - \boldsymbol{\alpha}^0\|_2^2 \leq 0. \tag{3.18}$$

For all $n = 1, 2, \ldots$, we also have

$$\begin{aligned} F(\boldsymbol{\alpha}^{n+1}) - F(\boldsymbol{\alpha}^n) &\leq \frac{L_{n+1}}{2}\{\|\boldsymbol{\beta}^{n+1} - \boldsymbol{\alpha}^n\|_2^2 - \|\boldsymbol{\alpha}^{n+1} - \boldsymbol{\alpha}^n\|_2^2\} \\ &= \frac{L_{n+1}}{2}\Big\{\Big(\frac{t_n - 1}{t_{n+1}}\Big)^2 \|\boldsymbol{\alpha}^n - \boldsymbol{\alpha}^{n-1}\|_2^2 - \|\boldsymbol{\alpha}^{n+1} - \boldsymbol{\alpha}^n\|_2^2\Big\}. \end{aligned}$$

Summing over $n = 1, 2, \ldots, k-1$, we obtain

$$\begin{aligned} &F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^1) \\ &\leq \frac{1}{2}\left\{L_2\Big(\frac{t_1 - 1}{t_2}\Big)^2 \|\boldsymbol{\alpha}^1 - \boldsymbol{\alpha}^0\|_2^2 + \sum_{n=2}^{k-1}\Big(L_{n+1}\Big(\frac{t_n - 1}{t_{n+1}}\Big)^2 - L_n\Big)\|\boldsymbol{\alpha}^n - \boldsymbol{\alpha}^{n-1}\|_2^2 - L_k\|\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1}\|_2^2\right\}. \end{aligned}$$

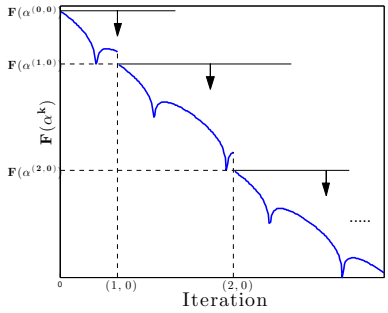


Figure 3.3.: Illustration of the sequence $\{F(\boldsymbol{\alpha}^k)\}_{k=0}^{\infty}$ generated by Algorithm 3.2. Lemma 3.3.3 ensures that the function values $F(\boldsymbol{\alpha}^{(i,0)})$ at restarted points are non-increasing and gives upper bounds of the subsequent function values.

Note that $t_1 - 1 = 0$ and $\left(L_{n+1}\left(\frac{t_n-1}{t_{n+1}}\right)^2 - L_n\right) \leq 0$ for all $n \geq 1$ because

$$\frac{t_n - 1}{t_{n+1}} = \frac{2(t_n - 1)}{1 + \sqrt{1 + 4(L_{n+1}/L_n)t_n^2}} \leq \frac{2(t_n - 1)}{\sqrt{4(L_{n+1}/L_n)t_n^2}} = \sqrt{\frac{L_n}{L_{n+1}}}\frac{t_n - 1}{t_n}.$$

Thus we have

$$F(\boldsymbol{\alpha}^k) - F(\boldsymbol{\alpha}^1) \leq 0.$$

Similarly, since the restart does not occur from $\boldsymbol{\alpha}^{(i,0)}$ to $\boldsymbol{\alpha}^{(i,\bar{k}_{i+1})}$ $(\forall i \geq 0)$, we have

$$F(\boldsymbol{\alpha}^{(i,k_{i+1})}) \leq F(\boldsymbol{\alpha}^{(i,0)}), \quad \forall k_{i+1} \in \{0, 1, \ldots, \bar{k}_{i+1}\},$$

and hence by definition,

$$F(\boldsymbol{\alpha}^{(i+1,0)}) = F(\boldsymbol{\alpha}^{(i,\bar{k}_{i+1}-1)}) \leq F(\boldsymbol{\alpha}^{(i,0)}).$$

□

Lemma 3.3.3 states that the initial function value $F(\boldsymbol{\alpha}^{(i,0)})$ of each outer iteration gives an upper bound of the subsequent function values and hence the sequence $\{F(\boldsymbol{\alpha}^{(i,0)})\}_{i=1}^{\infty}$ is non-increasing as illustrated in Figure 3.3.

Now we are ready to show the convergence result of Algorithm 3.2.

Theorem 3.3.1. *Consider the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty}(\equiv \{\boldsymbol{\alpha}^{(i,k_{i+1})}\})$ generated by Algorithm 3.2. Let S^* be the set of optimal solutions and $B := \{\boldsymbol{\alpha} \mid F(\boldsymbol{\alpha}) \leq F(\boldsymbol{\alpha}^0)\}$ be the level set. Assume that there exists a finite R such that*

$$R \geq \sup_{\boldsymbol{\alpha} \in B} \inf_{\boldsymbol{\alpha}^* \in S^*} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_2.$$

Then we have

$$F(\boldsymbol{\alpha}^k) - F^* \leq 2\eta_u L_f R^2 \Big(\frac{\log_2(k+2)}{k - \log_2(k+2)}\Big)^2, \quad \forall k \geq 3.$$

Proof. From Lemma 3.3.3, we have $\boldsymbol{\alpha}^k \in B$ for all $k \geq 1$. Let $\boldsymbol{\alpha}^k = \boldsymbol{\alpha}^{(j,k_{j+1})}$. We assume $k_{j+1} \geq 1$ without loss of generality. From Proposition 2.2.1, we have

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F^* \leq \frac{2\eta_u L_f \|\boldsymbol{\alpha}^{(j,0)} - \boldsymbol{\alpha}^*\|_2^2}{k_{j+1}^2} \quad {}^\forall\boldsymbol{\alpha}^* \in S^*.$$

Moreover, for all $1 \leq i \leq j$, Lemma 3.3.3 leads to

$$\begin{aligned} F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F^* &\leq F(\boldsymbol{\alpha}^{(j,0)}) - F^* \\ &\leq F(\boldsymbol{\alpha}^{(i,0)}) - F^* \\ &= F(\boldsymbol{\alpha}^{(i-1,\bar{k}_i-1)}) - F^*. \end{aligned}$$

From the assumption, there exists $\boldsymbol{\alpha}^{*(i)} \in S^*$ such that $\|\boldsymbol{\alpha}^{(i-1,0)} - \boldsymbol{\alpha}^{*(i)}\|_2 \leq R$. Hence, we have

$$\begin{aligned} F(\boldsymbol{\alpha}^{(i-1,\bar{k}_i-1)}) - F^* &\leq \frac{2\eta_u L_f \|\boldsymbol{\alpha}^{(i-1,0)} - \boldsymbol{\alpha}^{*(i)}\|_2^2}{(\bar{k}_i - 1)^2} \\ &\leq \frac{2\eta_u L_f R^2}{(\bar{k}_i - 1)^2}. \end{aligned}$$

Note that $\bar{k}_i \geq K_i \geq 2$. Thus we obtain

$$\begin{aligned} F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F^* &\leq \frac{2\eta_u L_f R^2}{(\max\{\bar{k}_1 - 1, \bar{k}_2 - 1, \ldots, \bar{k}_j - 1, k_{j+1}\})^2} \\ &\leq \frac{2\eta_u L_f R^2}{(\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\} - 1)^2} \end{aligned}$$

From

$$k \geq \sum_{i=1}^{j} K_i = \frac{K_1(2^j - 1)}{2 - 1} \geq 2^{j+1} - 2,$$

the number of restart j is at most $\log_2(k+2) - 1$. Hence we have

$$\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\} \geq \frac{k}{j+1} \geq \frac{k}{\log_2(k+2)}, \tag{3.19}$$

which leads to

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F^* \leq 2\eta_u L_f R^2 \Big(\frac{\log_2 k}{k - \log_2(k+2)}\Big)^2, \quad \forall k \geq 3.$$

□

The above theorem ensures the convergence of Algorithm 3.2 for C-SVM, ν-SVM, MM-MPM, MM-FDA, and ℓ_2-SVM because the level set B is bounded for any initial point $\boldsymbol{\alpha}^0 \in \mathrm{dom}(g)$. (Note that S_C, S_ν, S_{FDA}, and S_{MPM} are bounded, and the objective function of ℓ_2-SVM is strongly convex.) Similarly, the algorithm is convergent for the

logistic regression problem (3.8) if S_{LR} is replaced by $\{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi \boldsymbol{e} \leq \boldsymbol{\alpha} \leq \boldsymbol{e}\}$ for some small constant $\xi > 0$. To guarantee the convergence of Algorithm 3.2 for the distance weighted discrimination problem (3.10), we need to replace S_{DWD} by $S_{\mathrm{DWD}}(\xi) := \{\boldsymbol{\alpha} \in \mathbb{R}^m \mid \boldsymbol{\alpha}^\top \boldsymbol{y} = 0,\ \xi \boldsymbol{e} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu}\boldsymbol{e}\}$ for some small constant $\xi > 0$ and make the assumption that $\|\widetilde{\boldsymbol{X}}\boldsymbol{\alpha}\|_2 > 0$ for all $\boldsymbol{\alpha} \in S_{\mathrm{DWD}}(\xi)$.

The iteration complexity $O((\log(k)/k)^2)$ shown in Theorem 3.3.1 can be improved to $O((W(k)/k)^2)$ by finding a better lower bound of (3.19), where $W(\cdot)$ is the inverse function of $f(x) = x\exp(x)$. $W(\cdot)$ is known as Lambert's W function [Corless et al., 1996], and its properties have been studied. $W(\cdot)$ is single-valued for $z \geq 0$, but multi-valued for $z < 0$. For $z \geq 0$, $W(z)$ is monotonically increasing, but diverges slower than $\log(z)$. It is known that $W(z)$ cannot be expressed in terms of elementary functions.

Theorem 3.3.2. *Consider the sequence $\{\boldsymbol{\alpha}^k\}_{k=0}^{\infty} (\equiv \{\boldsymbol{\alpha}^{(i,k_{i+1})}\})$ generated by Algorithm 3.2. Let S^* be the set of optimal solutions and $B := \{\boldsymbol{\alpha} \mid F(\boldsymbol{\alpha}) \leq F(\boldsymbol{\alpha}^0)\}$ be the level set. Assume that there exists a finite R such that*

$$R \geq \sup_{\boldsymbol{\alpha} \in B} \inf_{\boldsymbol{\alpha}^* \in S^*} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}^*\|_2.$$

Then we have

$$F(\boldsymbol{\alpha}^k) - F^* \leq 2\eta_u L_f R^2 \Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2, \quad \forall k \geq 3.$$

Proof. Here we improve the lower bound (3.19) of

$$\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\}.$$

Since $\bar{k}_i \geq K_i \geq 2^i$ for all $i \in \{1, 2, \ldots, j\}$, we have

$$\max\{\bar{k}_1, \bar{k}_2, \ldots, \bar{k}_j, k_{j+1}\} \geq \max\Big\{\frac{k}{j+1}, 2^j\Big\}.$$

$\frac{k}{j+1}$ is monotonically decreasing and 2^j is monotonically increasing with respect to j. Thus the right-hand side is minimized at $j = \hat{j}$ such that

$$\frac{k}{\hat{j}+1} = 2^{\hat{j}}.$$

The above equation implies that

$$\begin{aligned}
&\frac{2k}{\hat{j}+1} = \exp\left((\hat{j}+1)\ln 2\right) \\
&\Rightarrow 2k\ln 2 = (\hat{j}+1)\ln 2 \exp\left((\hat{j}+1)\ln 2\right) \\
&\Rightarrow W(2k\ln 2) = (\hat{j}+1)\ln 2 \\
&\Rightarrow \hat{j} = \frac{W(2k\ln 2)}{\ln 2} - 1,
\end{aligned}$$

which yields

$$\max\Big\{\frac{k}{j+1}, 2^j\Big\} \geq \frac{k\ln 2}{W(2k\ln 2)}.$$

This leads to

$$F(\boldsymbol{\alpha}^{(j,k_{j+1})}) - F(\boldsymbol{\alpha}^*) \leq 2\eta_u L_f R^2 \Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2, \quad \forall k \geq 3.$$

□

3.3.3. A Practical FAPG

In this section, we develop a practical version of the FAPG method which reduces the computation cost of FAPG at each iterations.

3.3.3.1. Heuristic Backtracking Procedure

Although the APG method of [Scheinberg et al., 2014], which is employed in Algorithm 2.1 and modified in Algorithm 3.2, can guarantee the convergence, its principal drawback is the extra computational cost at the backtracking step. Recall that

$$Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k) = f(\boldsymbol{\beta}^k) + \langle \nabla f(\boldsymbol{\beta}^k),\, \boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\rangle + \frac{L_k}{2}\|\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\|_2^2 + g(\boldsymbol{\alpha}^k).$$

To check the condition $F(\boldsymbol{\alpha}^k) \leq Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$, we need to recompute $f(\boldsymbol{\beta}^k)$ and $\nabla f(\boldsymbol{\beta}^k)$ since $\boldsymbol{\beta}^k$ is updated at each loop of the backtracking.

On the other hand, since the original APG of [Beck and Teboulle, 2009] does not update $\boldsymbol{\beta}^k$ during the backtracking step, we can reduce the computation cost by storing the value of $f(\boldsymbol{\beta}^k)$ and $\nabla f(\boldsymbol{\beta}^k)$. Hence we follow the strategy of [Beck and Teboulle, 2009], i.e., we remove the steps for updating t_k and $\boldsymbol{\beta}^k$ subsequent to 'bt' from Algorithm 3.2 and restore Step 2 as $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$.

3.3.3.2. Skipping Extra Computations

The backtracking step ('bt') involves extra computation of the function value $F(\boldsymbol{\alpha}^k)$. Checking the termination criteria $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2 < \epsilon$ also involves the extra computation of $\nabla f(\boldsymbol{\alpha}^k)$ which is needed for $T_{L_k}(\boldsymbol{\alpha}^k)$. These computation costs can be significant (see Tables 3.3 and 3.9 in Section 3.4). Thus we compute the 'bt' and $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2$ in every 10 and 100 iterations, respectively.

Instead of checking the condition $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|_2 < \epsilon$, we check whether $L_k\|\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\|_2 < \epsilon$ at each iteration. The reason for doing so is that computing $L_k\|\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k\|_2$ is much cheaper than computing $T_{L_k}(\boldsymbol{\alpha}^k)$. It follows from (2.3) and (2.4) that if $\boldsymbol{\alpha}^k = \boldsymbol{\beta}^k$, then $\boldsymbol{\alpha}^k$ is an optimal solution. Moreover, $L_k(\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k)$ represents a residual of a sufficient optimality condition for $\boldsymbol{\alpha}^k$ (and necessary and sufficient optimality condition for $\boldsymbol{\beta}^k$).

Finally, our practical FAPG is described as in Algorithm 3.3. We note that FAPG can be applied not only to the unified formulation shown in Section 3.2, but also to the optimization problem of the form (2.1). See Appendix A.2 in which we demonstrate the performance of FAPG for ℓ_1-regularized classification models.

Algorithm 3.3 A practical FAPG method.

Input: f, ∇f, g, $\text{prox}_{g,L}$, $\epsilon > 0$, $L_1 > 0$, $\eta_u > 1$, $\eta_d > 1$ $\delta \in (0,1)$, $k_{max} > 0$, $K_1 \geq 2$, $\boldsymbol{\beta}^1 = \boldsymbol{\alpha}^0$

Output: $\boldsymbol{\alpha}^k$

Initialize: $t_1 \leftarrow 1$, $i \leftarrow 1$, $k_{re} \leftarrow 0$

for $k = 1, \ldots, k_{max}$ **do**

  $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # Step 1

  if k mod $10 == 1$ **then**

    while $F(\boldsymbol{\alpha}^k) > Q_{L_k}(\boldsymbol{\alpha}^k; \boldsymbol{\beta}^k)$ **do**

      $L_k \leftarrow \eta L_k$

      $\boldsymbol{\alpha}^k \leftarrow T_{L_k}(\boldsymbol{\beta}^k) = \text{prox}_{g,L_k}\left(\boldsymbol{\beta}^k - \frac{1}{L_k}\nabla f(\boldsymbol{\beta}^k)\right)$ # 'bt'

    end while

  end if

  if $\|L_k(\boldsymbol{\alpha}^k - \boldsymbol{\beta}^k)\| < \epsilon$ or ((k mod $100 == 1$) and ($\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\| < \epsilon$)) **then**

    break

  end if

  $L_{k+1} \leftarrow L_k/\eta_d$ # 'dec'

  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$ # Step 2

  $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^k + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1})$ # Step 3

  if $k > k_{re} + K_i$ and $\langle \nabla f(\boldsymbol{\beta}^k), \boldsymbol{\alpha}^k - \boldsymbol{\alpha}^{k-1} \rangle + g(\boldsymbol{\alpha}^k) - g(\boldsymbol{\alpha}^{k-1}) > 0$ **then**

    $k_{re} \leftarrow k$, $K_{i+1} \leftarrow 2K_i$, $i \leftarrow i + 1$. # 'mt'

    $\eta_d \leftarrow \delta \cdot \eta_d + (1 - \delta) \cdot 1$. # 'st'

    $t_{k+1} \leftarrow 1$, $\boldsymbol{\beta}^{k+1} \leftarrow \boldsymbol{\alpha}^{k-1}$, $\boldsymbol{\alpha}^k \leftarrow \boldsymbol{\alpha}^{k-1}$ # 're'

  end if

end for

3.3.4. Computation of a Primal Solution from a Dual Solution

Various classification models can be solved in a unified way by applying the vector projection method and the FAPG method to the dual formulation (9) or (10). However, a primal solution $(\boldsymbol{w}, b)$ of (7) or (8) is still required to do classification tasks. In this section, we provide a method to compute the primal solution $(\hat{\boldsymbol{w}}, \hat{b})$ which corresponds to the dual solution $\hat{\boldsymbol{\alpha}}$.

3.3.4.1. Computation of the primal vector $\boldsymbol{w}$

The primal vector $\hat{\boldsymbol{w}}$ in (3.3) corresponding to the dual solution $\hat{\boldsymbol{\alpha}}$ (and $\hat{\boldsymbol{\mu}}$ for MM-FDA and MM-MPM) can be obtained as

$$\hat{\boldsymbol{w}} = \operatorname*{argmax}_{\|\boldsymbol{w}\|_p \le 1} \boldsymbol{w}^\top \boldsymbol{z}, \quad \text{with} \quad \hat{w}_i = \frac{\operatorname{sign}(z_i)|z_i|^{q-1}}{\|\boldsymbol{z}\|_q^{q-1}} \quad \forall\, i \in M, \tag{3.20}$$

where $q = p/(p-1)$ and $\boldsymbol{z} = \widetilde{\boldsymbol{A}}\hat{\boldsymbol{\alpha}}$ (i.e., $\boldsymbol{z} = \widetilde{\boldsymbol{X}}\hat{\boldsymbol{\alpha}}$ for C-SVM, ℓ_2-SVM, ν-SVM, logistic regression, and DWD; $\boldsymbol{z} = \hat{\boldsymbol{\alpha}} = (\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_-) + (\Sigma_+ + \Sigma_-)^{1/2}\hat{\boldsymbol{\mu}}$ for MM-FDA; and $\boldsymbol{z} = \hat{\boldsymbol{\alpha}} = (\bar{\boldsymbol{x}}_+ + \Sigma_+^{1/2}\hat{\boldsymbol{\mu}}_+) - (\bar{\boldsymbol{x}}_- + \Sigma_-^{1/2}\hat{\boldsymbol{\mu}}_-)$ for MM-MPM).

3.3.4.2. Computation of the bias term b

There are some issues on computing an optimal bias term b. For C-SVM, ℓ_2-SVM, ν-SVM, DWD, and logistic regression, the bias term b is derived from the Lagrange multiplier corresponding to the constraint $\boldsymbol{\alpha}^\top \boldsymbol{y} = 0$. However, it is often difficult to compute the corresponding Lagrange multiplier. In addition, MM-FDA and MM-MPM does not provide a specific way to compute the bias term. Thus we need to estimate an appropriate value of b.

One of the efficient ways is to estimate b as the minimum solution of training error under a given $\hat{\boldsymbol{w}}$, i.e.,

$$\hat{b} \in \operatorname*{argmin}_b \ \rho(b) := \sum_{i=1}^{m} \ell(y_i(\boldsymbol{x}_i^\top \hat{\boldsymbol{w}} - b)), \quad \text{where} \quad \ell(z) = \begin{cases} 1 & (z < 0) \\ 0 & (\text{otherwise}). \end{cases} \tag{3.21}$$

Let $\zeta_i = \boldsymbol{x}_i^\top \hat{\boldsymbol{w}}$ $(i = 1, 2, \ldots, m)$. Let σ be the permutation such that $\zeta_{\sigma(1)} \le \zeta_{\sigma(2)} \le \ldots \le \zeta_{\sigma(m)}$. If $b < \zeta_{\sigma(1)}$, then $\boldsymbol{x}_i^\top \hat{\boldsymbol{w}} - b > 0$ $(\forall i \in M)$, i.e., all samples are predicted as positive $\hat{y} = +1$. Then we have m_- misclassified samples and thus $\rho(b) = m_-$. If $b \in (\zeta_{\sigma(1)}, \zeta_{\sigma(2)})$, then only $\boldsymbol{x}_{\sigma(1)}$ is predicted as negative $\hat{y} = -1$. Thus we have $\rho(b) = m_- + y_{\sigma(1)}$. Similarly, if $b \in (\zeta_{\sigma(k)}, \zeta_{\sigma(k+1)})$, then we have $\rho(b) = m_- + \sum_{i=1}^{k} y_{\sigma(i)}$. Thus, by letting $k^* \in \operatorname{argmin}_k \sum_{i=1}^{k} y_{\sigma(i)}$, an arbitrary constant $\hat{b} \in (\zeta_{\sigma(k^*)}, \zeta_{\sigma(k^*+1)})$ is an optimal solution of the problem (3.21). The minimization algorithm is as follows:

Step 1. Compute $\zeta_i = \hat{\boldsymbol{w}}^\top \boldsymbol{x}_i$ $(i = 1, 2, \ldots, m)$.

Step 2. Sort ζ_i as $\zeta_{\sigma(1)} \le \zeta_{\sigma(2)} \le \ldots \le \zeta_{\sigma(m)}$.

Step 3. Find $k^* = \mathrm{argmin}_k \sum_{i=1}^{k} y_{\sigma(i)}$.

Step 4. Compute $\hat{b} = (\zeta_{(k^*)} + \zeta_{(k^*+1)})/2$.

Its computational complexity is $O\big(m(n + \log m)\big)$.

3.3.4.3. Duality Gap

The component b obtained above is not necessarily the primal optimal solution. Moreover, although $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\mu}}$ are close to optimal, they may not be exactly optimal due to numerical error. However, we still obtain a primal solution $(\hat{\boldsymbol{w}}, \hat{b})$ with some optimality gap guarantee. Since $(\hat{\boldsymbol{w}}, \hat{b})$ and $\hat{\boldsymbol{\alpha}}$ are the primal and dual feasible solutions, respectively, we can obtain the duality gap by

$$\Big\{\mathcal{L}(\widetilde{\boldsymbol{A}}^\top \hat{\boldsymbol{w}} - \boldsymbol{a}\hat{b}) + \frac{1}{2C}\|\hat{\boldsymbol{w}}\|_2^2)\Big\} - \Big\{\mathcal{L}^*(-\hat{\boldsymbol{\alpha}}) + \frac{C}{2}\|\widetilde{\boldsymbol{A}}\hat{\boldsymbol{\alpha}}\|_2^2\Big\}$$

for the formulations of (3.2) and (3.4); and

$$\mathcal{L}(\widetilde{\boldsymbol{A}}^\top \hat{\boldsymbol{w}} - \boldsymbol{a}\hat{b}) - \Big\{\mathcal{L}^*(-\hat{\boldsymbol{\alpha}}) + \lambda\|\widetilde{\boldsymbol{A}}\hat{\boldsymbol{\alpha}}\|_p^*\Big\}$$

for the formulations of (3.3) and (3.5). Therefore, the obtained primal solution $(\hat{\boldsymbol{w}}, \hat{b})$ has a guarantee that the gap between its objective value and the optimal value is at most the duality gap.

3.4. Numerical Experiments

In this section, we demonstrate the performance of our proposed FAPG algorithm. We ran the numerical experiments on a Red Hat Enterprise Linux Server release 6.4 (Santiago) with Intel Xeon Processor E5-2680 (2.7GHz) and 64 GB of physical memory. We implemented the practical FAPG method (Algorithm 3.3) in MATLAB R2013a and the bisection method (Algorithm 3.1) in C++. The C++ code was called from MATLAB via MEX files.

We conducted the experiments using artificial datasets and benchmark datasets from LIBSVM Data [Chang and Lin, 2011]. The artificial datasets were generated as follows. Positive samples $\{\boldsymbol{x}_i \in \mathbb{R}^n \mid i \in M_+\}$ and negative samples $\{\boldsymbol{x}_i \in \mathbb{R}^n \mid i \in M_-\}$ were distributed with n-dimensional standard normal distributions $\mathcal{N}_n(\boldsymbol{0}, I_n)$ and $\mathcal{N}_n(\frac{10}{\sqrt{n}}\boldsymbol{e}, SS^\top)$, respectively, where the elements of the $n \times n$ matrix S are i.i.d. random variables following the standard normal distribution $\mathcal{N}(0, 1)$. The marginal probability of the label was assumed to be same, i.e. $P(y = +1) = P(y = -1) = \frac{1}{2}$. After generating the samples, we scaled them so that each input vector $\boldsymbol{x}_i$ $(\forall i \in M)$ was in $[-1, 1]^n$ for the purpose of computational stability, following LIBSVM [Chang and Lin, 2011]. On the other hand, we scaled the benchmark datasets, that are not scaled by Chang and Lin [2011], so that $\boldsymbol{x}_i \in [0, 1]^n$, $(\forall i \in [m])$ in order to leverage their sparsity. The details of benchmark datasets are shown in Table 3.1.

Table 3.1.: Details of datasets. We have scaled the datasets that are highlighted in boldface type.

data	m (	m_+,	m_-)	n	range	density	source
a8a	22,696 (	5,506,	17,190)	123	$[0,1]^n$	0.113	[Bache and Lichman, 2013]
a9a	32,561 (	7,841,	24,720)	123	$[0,1]^n$	0.113	[Bache and Lichman, 2013]
australian	690 (	307,	383)	14	$[-1,1]^n$	0.874	[Bache and Lichman, 2013]
breast-cancer	683 (	444,	239)	10	$[-1,1]^n$	1.000	[Bache and Lichman, 2013]
cod-rna	59,535 (	39,690,	19,845)	8	$[0,1]^n$	0.999	[Uzilov et al., 2006]
colon-cancer	62 (	40,	22)	2,000	$[0,1]^n$	0.984	[Alon et al., 1999]
covtype	581,012 (	297,711,	283,301)	54	$[0,1]^n$	0.221	[Bache and Lichman, 2013]
diabetes	768 (	500,	268)	8	$[-1,1]^n$	0.999	[Bache and Lichman, 2013]
duke	44 (	21,	23)	7,129	$[0,1]^n$	0.977	[West et al., 2001]
epsilon	400,000 (	199,823,	200,177)	2,000	$[-0.15,0.16]^n$	1.000	[Sonnenburg et al., 2008]
fourclass	862 (	307,	555)	2	$[-1,1]^n$	0.996	[Ho and Kleinberg, 1996]
german.numer	1,000 (	300,	700)	24	$[-1,1]^n$	0.958	[Bache and Lichman, 2013]
gisette	6,000 (	3,000,	3,000)	5,000	$[-1,1]^n$	0.991	[Guyon et al., 2005]
heart	270 (	120,	150)	13	$[-1,1]^n$	0.962	[Bache and Lichman, 2013]
ijcnn1	35,000 (	3,415,	31,585)	22	$[-0.93,1]^n$	0.591	[Prokhorov, 2001]
ionosphere	351 (	225,	126)	34	$[-1,1]^n$	0.884	[Bache and Lichman, 2013]
leu	38 (	11,	27)	7,129	$[0,1]^n$	0.974	[Golub et al., 1999]
liver-disorders	345 (	145,	200)	6	$[-1,1]^n$	0.991	[Bache and Lichman, 2013]
madelon	2,000 (	1,000,	1,000)	500	$[0,1]^n$	0.999	[Guyon et al., 2005]
mushrooms	8,124 (	3,916,	4,208)	112	$[0,1]^n$	0.188	[Bache and Lichman, 2013]
news20.binary	19,996 (	9,999,	9,997)	1,355,191	$[0,1]^n$	3.36e-4	[Keerthi and DeCoste, 2005]
rcv1-origin	20,242 (	10,491,	9,751)	47,236	$[0,0.87]^n$	0.002	[Lewis et al., 2004]
real-sim	72,309 (	22,238,	50,071)	20,958	$[0,1]^n$	0.002	[McCallum]
skin-nonskin	245,057 (	50,859,	194,198)	3	$[0,1]^n$	0.983	[Bache and Lichman, 2013]
sonar	208 (	97,	111)	60	$[-1,1]^n$	1.000	[Bache and Lichman, 2013]
splice	1,000 (	517,	483)	60	$[-1,1]^n$	1.000	[Bache and Lichman, 2013]
svmguide1	3,089 (	1,089,	2,000)	4	$[0,1]^n$	0.997	[Hsu et al., 2003]
svmguide3	1,243 (	947,	296)	22	$[0,1]^n$	0.805	[Hsu et al., 2003]
url	2,396,130 (	1,603,985,	792,145)	3,231,961	$[0,1]^n$	3.54e-5	[Ma et al., 2009]
w7a	24,692 (	740,	23,952)	300	$[0,1]^n$	0.039	[Platt, 1998]
w8a	49,749 (	1,479,	48,270)	300	$[0,1]^n$	0.039	[Platt, 1998]

3.4.1. Projection Algorithms

Before presenting the performance of our practical FAPG algorithm, we compared the performance of our bisection algorithm (Algorithm 3.1) against the breakpoint search algorithm [Kiwiel, 2008, Algorithm 3.1] with random pivoting. Both algorithms were implemented in C++. We generated $\mathbb{R}^n$-valued random vectors $\tilde{\boldsymbol{\alpha}}$ with uniformly distributed elements and computed the projections $P_{S_\nu}(\tilde{\boldsymbol{\alpha}})$ of $\tilde{\boldsymbol{\alpha}}$ onto S_ν, where $S_\nu := \{\boldsymbol{\alpha} \mid \boldsymbol{e}_o^\top \boldsymbol{\alpha}_o = \frac{1}{2},\ o \in \{+,-\},\ \mathbf{0} \leq \boldsymbol{\alpha} \leq \frac{1}{m\nu}\boldsymbol{e}\}$ with $\nu = 0.5$. The bisection algorithm used the accuracy of $\epsilon' \approx 2.22 \times 10^{-16}$ (i.e., IEEE 754 double precision). Table 3.2 reports the average and standard deviation of the computation times and the number of iterations of 20 trials. As we can see from Table 3.2, the bisection method was faster and more stable (in the sense that the deviations are smaller) than the breakpoint search algorithm. This explains why we have chosen to use the bisection methods in Section 3.3.1 to perform the projection steps in Algorithm 3.3.

Table 3.2.: Runtime of projection algorithms.

		msec.(#iter.)			
		Breakpoint		Bisection	
dim. n	range	ave.	std.	ave.	std.
100,000	$[0,10]^n$	8.3 (25.7)	2.0 (5.4)	4.9 (31.1)	0.6 (4.7)
100,000	$[0,1000]^n$	9.0 (27.3)	1.4 (4.1)	5.5 (27.1)	1.2 (3.2)
1,000,000	$[0,10]^n$	94.2 (22.6)	16.2 (4.3)	49.9 (32.0)	3.4 (3.1)
1,000,000	$[0,1000]^n$	99.1 (26.5)	15.6 (3.2)	53.4 (30.0)	4.4 (1.5)

3.4.2. ν-SVM

As mentioned in Section 3.2.2, the standard C-SVM (3.6) and ν-SVM (3.9) are equivalent. Here we chose ν-SVM to solve because choosing the parameter of ν-SVM is easier than that of C-SVM. We solved the ν-SVM (3.9) via our FAPG method, SeDuMi [Sturm, 1999], and LIBSVM [Chang and Lin, 2011]. SeDuMi is a general purpose optimization solver implementing an interior point method for large-scale second-order cone problems such as (3.9). LIBSVM implements the sequential minimal optimization (SMO) [Platt, 1998] which is specialized for learning ν-SVM. For reference, we also compared the FAPG method with LIBLINEAR [Fan et al., 2008] which implements a highly optimized stochastic dual coordinate descent method [Hsieh et al., 2008] for C-SVM[2] [Cortes and Vapnik, 1995] and is known to be quite an efficient method; we note that it may not be a fair comparison because LIBLINEAR omits the bias term b of C-SVM from the calculations, [3] i.e., it solves a less complex model than the ν-SVM (3.9) in order to

[2] C-SVM (with the bias term b) is known to lead to the same decision function as ν-SVM if ν and C are set properly [Schölkopf et al., 2000]. The value of C corresponding to ν can be computed by LIBSVM.

[3] Although LIBLINEAR can virtually deal with the bias term b by augmenting the dimension of the samples, the best performance of the resulting model tends to be lower than the one of ν-SVM as

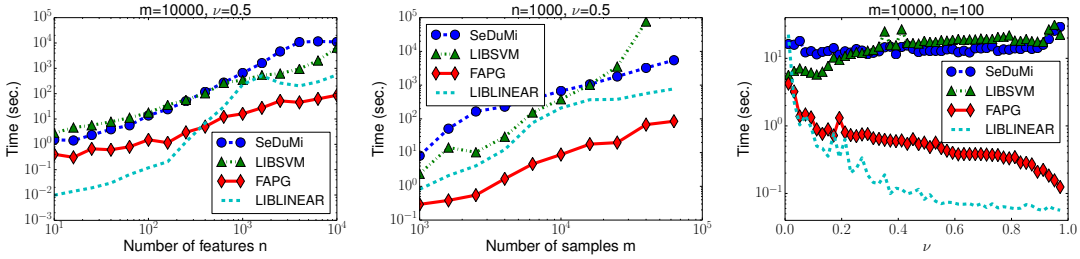


Figure 3.4.: Computation time for ν-SVM.

speed up the computation.

We terminate the algorithms if the violation of the KKT optimality condition is less than $\epsilon = 10^{-6}$. The heuristic option in LIBSVM was set to "off" in order to speed up its convergence for large datasets. In the FAPG method, (η_u, η_d, δ) were set to $(1.1, 1.1, 0.8)$. L_0 was set to the maximum value in the diagonal elements of $\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{X}}$ (i.e., the coefficient matrix of the quadratic form). The initial point $\boldsymbol{\alpha}^0$ was set to the *center* $\boldsymbol{\alpha}^c$ of S_ν, i.e. $\alpha_i^c = \frac{1}{2m_o}$, $i \in M_o$, $o \in \{+, -\}$.

3.4.2.1. Scalability

First, we compare the computation time with respect to the size of the datasets and parameters using artificial datasets. The results are shown in Figure 3.4. The left panel shows the computation time with respect to the dimension n of the features for $m = 10000$ and $\nu = 0.5$. The FAPG method has a clear advantage when the dimension n is high, say $n \geq 10^3$. The middle panel shows the computation time with respect to the number m of the samples for $n = 1000$ and $\nu = 0.5$. LIBSVM did not converge within a week for $m = 63000$. SeDuMi, LIBLINEAR, and the APG method were scalable for the increased number of samples m. The right panel illustrates the computation time with respect to the parameter ν for $m = 10000$ and $n = 100$. We may observe that the FAPG method (and LIBLINEAR) is very efficient when ν is larger than 0.1. This can be attributed to the fact that larger ν shrinks the feasible region S_ν and shorten the distance between the initial point $\boldsymbol{\alpha}^0 = \boldsymbol{\alpha}^c$ and the optimal solution $\boldsymbol{\alpha}^*$.

3.4.2.2. Runtime Breakdown

Table 3.3 shows the runtime breakdown of the FAPG method for the artificial dataset with $(m, n, \nu) = (10000, 1000, 0.5)$. The computation of the gradient $\nabla f(\boldsymbol{\alpha})$ and the function $f(\boldsymbol{\alpha})$ was the most time-consuming parts in the FAPG method. Since the computations of 'bt' (and $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$) involve the extra evaluation of $f(\boldsymbol{\alpha}^k)$ (and/or $\nabla f(\boldsymbol{\alpha}^k)$), computing them only every 100 iteration (and 10 iteration, respectively) as in Algorithm 3.3 would be effective to reduce the total runtime. Our projection algorithm

reported in [Kitamura et al., 2014].

Table 3.3.: Runtime breakdown of the FAPG method (sec.).

Function	% Time	Time	# Evals.	Time/Eval.
$\nabla f(\boldsymbol{\alpha})$	78.1%	14.909	1375	0.0108
$f(\boldsymbol{\alpha})$	10.8%	2.053	369	0.0056
$P_{S_\nu}(\boldsymbol{\alpha})$	6.9%	1.307	1453	0.0009
Total Runtime:		19.080		

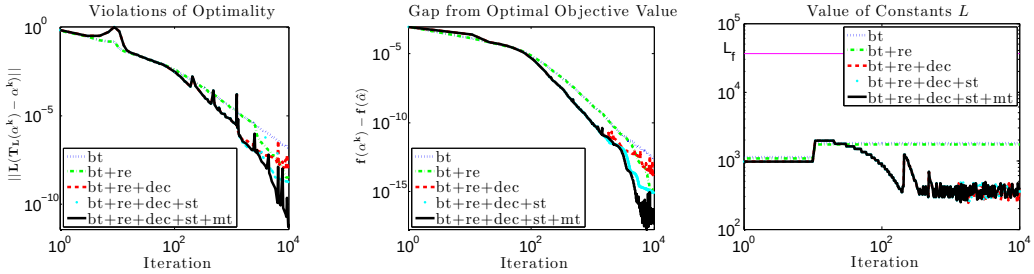


Figure 3.5.: Effect of various acceleration strategies for the APG Method.

was efficient enough in the sense that its runtime was marginal compared to the runtime of the other parts.

3.4.2.3. Effect of Each Acceleration Strategy

Figure 3.5 shows the running history of the FAPG method with various acceleration strategies for the artificial dataset with $(m, n, \nu) = (10000, 1000, 0.5)$.

The left panel depicts the violations of the optimality, i.e. the values of $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$. The FAPG method with 'bt+re' restarted at $k = 2594$, where the sharp decrease occurred. 'dec' was effective to reduce $L_k\|T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k\|$ in the early iterations. The FAPG method with 'bt+re+dec' seems to be unstable near the optimum (say $k \geq 10^3$), but the one with 'bt+re+dec+st' converged stably. Moreover, 'bt+re+dec+st+mt' obtained much more accurate solution than others. We note that several spikes of the values occurred when using 'dec' because it sometimes leads a small value of L_k that violates the condition (2.5). However, the violation was swiftly recovered in every 10 iterations by 'bt' and did not affect the speed of convergence so much.

The middle panel illustrated the gap in objective value $f(\boldsymbol{\alpha}^k) - f(\hat{\boldsymbol{\alpha}})$ where we regarded $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{10000}$ of 'bt+re+dec+st+mt' as an optimal solution. The gap behaved similar to the violations of the optimality $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$. At $k = 1000$, the objective value $f(\boldsymbol{\alpha}^k)$ reached $f(\hat{\boldsymbol{\alpha}})$ within a relative error 0.0003% though the violation of optimality was greater than $\epsilon = 10^{-6}$. Thus, a larger tolerance, say $\epsilon = 10^{-5}$, may also lead to a reasonable solution in practice.

The right panel illustrated the value of constant L_k of the FAPG at each iteration and the value of Lipschitz constant L_f; L_f is known to be the largest eigenvalue of $\widetilde{\boldsymbol{X}}^\top \widetilde{\boldsymbol{X}}$. While $L_f = 3.61 \times 10^4$, the FAPG method with 'bt+re+dec+st+mt' leads to a much smaller average value of 3.68×10^2 for L_k, while the maximum value for L_k is 1.95×10^3.

Table 3.4.: Computation time for benchmark datasets (sec.). '–' means that the algorithm did not converge with in 36000 seconds. '**' means that it had run out of memory. The best results except for LIBLINEAR are indicated by boldface. The underlined results are better than LIBLINEAR.

				Linear				RBF		
data	m	n	ν	SeDuMi	LIBSVM	FAPG (iter)	LIBLIN	SeDuMi	LIBSVM	FAPG(iter)
a8a	22,696	123	0.368	16.82	32.31	**1.70** (563)	0.06	–	**68.24**	72.88(343)
a9a	32,561	123	0.365	25.42	66.11	**4.84** (665)	0.07	–	**138.89**	170.63(390)
australian	690	14	0.348	0.34	**0.12**	0.86 (4056)	0.003	8.03	**0.049**	0.076(243)
breast-cancer	683	10	0.128	0.21	**0.004**	0.05 (253)	0.001	6.72	**0.015**	0.050(144)
cod-rna	59,535	8	0.223	6.71	53.73	**3.39** (588)	0.17	**	267.56	**(**)
colon-cancer	62	2,000	0.078	0.18	**0.01**	0.09 (210)	0.15	0.17	**0.013**	**0.013**(109)
covtype	581,012	54	0.620	288.71	16164.88	**60.75** (701)	1.55	**	–	**(**)
diabetes	768	8	0.533	0.14	**0.02**	0.06 (306)	0.003	8.53	**0.060**	0.093(296)
duke	44	7,129	0.106	0.38	**0.04**	0.11 (317)	0.30	0.09	0.041	**0.017**(120)
epsilon	400,000	2,000	0.500	–	–	**1685.33** (2643)	–	**	–	**(**)
fourclass	862	2	0.543	0.12	**0.01**	0.07 (356)	0.0004	12.34	0.069	**0.068**(150)
german.numer	1,000	24	0.525	**0.26**	**0.26**	0.29 (1107)	0.04	17.13	0.14	**0.10**(236)
gisette	6,000	5,000	0.100	4572.24	57.48	**9.31** (590)	0.41	5586.74	59.07	**4.93**(235)
heart	270	13	0.388	0.14	**0.005**	0.04 (232)	0.001	0.47	**0.008**	0.040(196)
ijcnn1	35,000	22	0.186	7.31	53.57	**5.67** (2000)	0.29	–	**73.83**	236.15(511)
ionosphere	351	34	0.202	0.25	**0.02**	0.22 (1064)	0.34	0.85	**0.012**	0.051(234)
leu	38	7,129	0.070	0.40	**0.03**	0.13 (175)	0.17	0.08	0.034	**0.013**(103)
liver-disorders	345	6	0.731	0.11	**0.01**	0.11 (736)	0.007	0.81	**0.015**	0.037(168)
madelon	2,000	500	0.603	29.15	11.57	**0.32** (510)	87.43	133.85	3.77	**0.24**(108)
mushrooms	8,124	112	0.096	8.74	1.34	**0.56** (435)	0.06	28505.46	**5.74**	19.02(661)
news20.binary	19,996	1,355,191	0.100	–	1333.26	**22.29** (321)	1.57	–	929.85	**51.86**(11)
rcv1-origin	20,242	47,236	0.097	–	587.10	**7.82** (485)	8.96	–	192.16	**97.68**(189)
real-sim	72,309	20,958	0.065	–	6351.62	**12.71** (384)	4.33	**	**1879.90**	**(**)
skin-nonskin	245,057	3	0.233	62.70	726.20	**8.02** (609)	0.09	**	**4425.59**	**(**)
sonar	208	60	0.117	0.29	**0.12**	0.33 (1922)	3.81	0.27	**0.009**	0.060(279)
splice	1,000	60	0.432	0.56	0.25	**0.11** (331)	0.02	15.27	0.18	**0.067**(120)
svmguide1	3,089	4	0.180	0.35	**0.08**	0.15 (394)	0.003	1188.65	**0.42**	1.45(323)
svmguide3	1,243	22	0.408	**0.36**	1.07	0.93 (3248)	1.70	30.17	0.23	**0.21**(430)
url	2,396,130	3,231,961	0.500	–	–	**4853.91** (2521)	–	**	**	**(**)
w7a	24,692	300	0.031	69.90	135.91	**14.13** (4280)	0.81	–	**17.89**	286.27(1208)
w8a	49,749	300	0.031	189.40	143.42	**38.69** (5960)	0.47	–	**124.55**	1423.42(2100)

3.4.2.4. Benchmark Results

We also conducted experiments using the benchmark datasets. C was set to 10. The computation time is shown in Table 3.4. When using linear kernel, the FAPG method outperformed LIBSVM and SeDuMi for large datasets where $m \geq 50000$. We indicated the smallest computation time among the three methods except for LIBLINEAR by boldface because the optimal solutions of LIBLINEAR are different from those of the other three methods (note that LIBLINEAR solves a simpler type of SVM, i.e., C-SVM without the bias term b). However, the FAPG method had an advantage over LIBLINEAR for many datasets where $n \geq 2000$.

When using the RBF kernel, SeDuMi broke down for datasets with $m \geq 10000$ as in the case of the artificial datasets. The FAPG method run out of memory for $m \geq 50000$ since it requires the $m \times m$ dense kernel matrix K to compute the gradient $\nabla f(\boldsymbol{\alpha})$. The FAPG can avoid the memory shortage by computing the elements of K on demand as LIBSVM does for large datasets. Although the current implementation has room for improvement, our practical FAPG method was still competitive with LIBSVM and had

Table 3.5.: Constants for benchmark datasets.

	Linear			RBF		
	L_k(FAPG)			L_k(FAPG)		
	ave.	max.	L_f	ave.	max.	L_f
a8a	2.99e3	1.54e4	1.43e5	7.00e1	2.27e2	2.00e4
a9a	5.59e3	3.38e4	2.05e5	1.37e2	4.99e2	2.88e4
australian	1.97e2	1.21e3	2.91e3	1.92e1	4.69e1	3.56e2
breast-cancer	9.05e1	2.27e2	1.68e4	1.21e1	2.13e1	4.52e2
cod-rna	1.05e3	5.65e3	1.24e5	–	–	–
colon-cancer	4.02e2	6.60e2	3.80e4	5.91e-1	9.09e-1	5.65e1
covtype	3.71e4	2.37e5	4.58e6	–	–	–
diabetes	5.21e1	1.53e2	1.76e3	1.64e1	4.69e1	6.34e2
duke	2.82e3	4.55e3	6.00e4	6.89e-1	9.09e-1	4.09e1
epsilon	2.09e4	1.85e5	1.40e5	–	–	–
fourclass	5.61e1	1.87e2	2.80e2	5.72e1	1.03e2	5.36e2
german.numer	2.16e2	1.03e3	8.44e3	1.95e1	4.69e1	4.50e2
gisette	1.84e4	5.71e4	2.02e7	9.17e0	2.13e1	3.61e3
heart	1.07e2	2.53e2	7.49e2	1.01e1	2.13e1	1.19e2
ijcnn1	1.20e3	1.05e4	5.89e3	1.93e2	8.84e2	3.12e4
ionosphere	2.46e2	9.83e2	2.14e3	1.19e1	2.83e1	2.24e2
leu	1.70e3	2.75e3	6.05e4	6.07e-1	9.09e-1	3.46e1
liver-disorders	1.27e1	3.85e1	1.84e3	5.04e0	9.68e0	8.23e2
madelon	9.54e1	2.87e2	2.44e5	6.54e-1	1.00e0	1.92e3
mushrooms	1.84e3	6.34e3	2.30e5	2.31e1	1.03e2	1.74e4
news20.binary	4.63e1	1.03e2	1.17e3	9.09e-1	9.09e-1	2.00e4
rcv1-origin	3.20e1	8.30e1	4.49e2	4.42e-1	9.09e-1	2.02e4
real-sim	6.75e1	2.27e2	9.21e2	–	–	–
skin-nonskin	5.08e3	2.82e4	2.19e5	–	–	–
sonar	2.27e2	1.12e3	2.68e3	4.88e0	9.68e0	1.51e2
splice	3.59e2	1.11e3	1.74e3	6.63e0	1.06e1	3.57e2
svmguide1	3.07e1	1.13e2	2.47e3	1.52e1	4.69e1	2.92e3
svmguide3	1.53e2	6.68e2	4.92e3	1.22e1	4.69e1	1.15e3
url	9.94e5	1.10e7	1.57e8	–	–	–
w7a	5.53e3	4.62e4	6.52e4	4.80e1	1.83e2	2.31e4
w8a	1.04e4	1.06e5	1.32e5	1.14e2	4.51e2	4.65e4

stable and good performance for datasets with large n.

We should remark that the number of iterations taken by the FAPG method with the RBF kernel tends to be smaller than the one with the linear kernel. However, when using the RBF kernel, the computational complexity of $\nabla f(\boldsymbol{\alpha})$ changes from $O(mn)$ to $O(m^2)$. Hence the total runtime tended to increase except for "gisette" whose n and m have the same order of magnitude.

In summary, the FAPG method showed better performance than specialized algorithms designed for learning SVM, such as LIBSVM and LIBLINEAR, in many datasets. Taking into account of the generality (i.e., applicability to other models) of FAPG, one could argue that it is a very efficient method.

Table 3.5 shows the values taken by the parameter L_k. We can see that our practical FAPG method (with backtracking strategy and decreasing strategy for L_k) keep the values of L_k to be much smaller than the Lipschitz constant L_f.

Table 3.6.: Classification accuracies with varied ξ from 10^{-2} to 10^{-5}.

	Artificial	svmguide1	mushrooms	a8a
$\xi = 10^{-2}$	99.8%	95.5%	99.4%	84.7%
$\xi = 10^{-3}$	100.0%	95.5%	100.0%	84.8%
$\xi = 10^{-4}$	100.0%	95.5%	100.0%	84.8%
$\xi = 10^{-5}$	100.0%	95.5%	100.0%	84.8%
LIBLINEAR	85.2%	90.7%	100.0%	84.7%

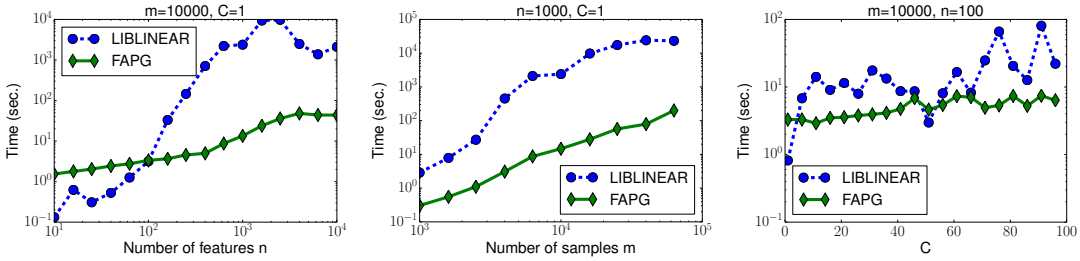


Figure 3.6.: Computation time for logistic regression.

3.4.3. Logistic Regression

We solved the dual logistic regression (3.8) via the FAPG method and LIBLINEAR [Fan et al., 2008]. LIBLINEAR implements a highly optimized stochastic dual coordinate descent method [Yu et al., 2011] whose subproblems are solved by the Newton method. Note that, as in the case of SVM, LIBLINEAR omits the bias term b from the calculations, i.e., it solves a less complex model than (3.8). We terminate both algorithms if the violation of the KKT optimality condition is less than $\epsilon = 10^{-6}$. In the FAPG method, (η_u, η_d, δ) were set to $(1.1, 1.1, 0.8)$. L_0 was set to 1. The initial point $\boldsymbol{\alpha}^0$ was set to $\xi \boldsymbol{e}$.

3.4.3.1. Sensitivity to the parameter ξ

Since the gradient ∇f is not defined at $\alpha_i = 0, 1$ $(i \in M)$, we approximately solve the problem by using the constraints $\xi \leq \alpha_i \leq 1 - \xi$ $(i \in M)$, where $\xi > 0$. Table 3.6 shows the classification accuracy for $\xi = 10^{-2}, 10^{-3}, 10^{-4}$, and 10^{-5}. For all cases, the approximated logistic regression performed better than LIBLINEAR. The classification accuracy of FAPG was identical for $\xi \leq 10^{-3}$, and better than the accuracy with $\xi = 10^{-2}$. We employ $\xi = 10^{-4}$ in the following experiments.

3.4.3.2. Scalability

First, we compare the computation time with respect to the size of the datasets and parameter using artificial datasets. The results (for the linear kernel) are shown in Figure 3.6. The left panel shows the computation time with respect to the dimension n

Table 3.7.: Runtime breakdown of the FAPG method (sec.).

Function	% Time	Time	# Evals.	Time/Eval.
$\nabla f(\boldsymbol{\alpha})$	85.4%	10.005	1452	6.89.e-3
$f(\boldsymbol{\alpha})$	12.3%	1.443	340	4.24.e-3
$P_{S_{\mathrm{LR}}}(\boldsymbol{\alpha})$	0.5%	0.058	1473	3.94.e-5
Total Runtime:		11.720		

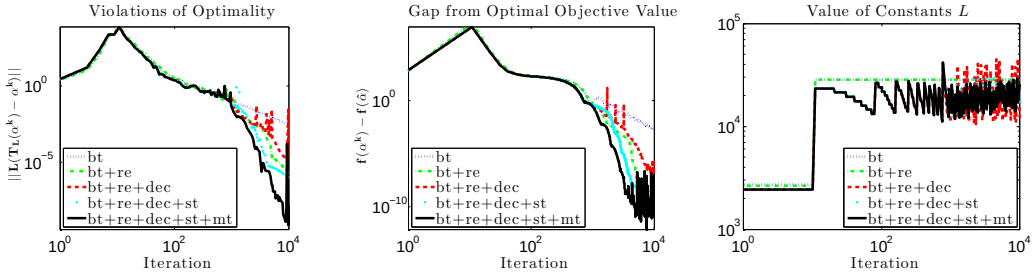


Figure 3.7.: Effect of various acceleration strategies for the FAPG Method.

of the features for $m = 10000$ and $C = 10$. The FAPG method has a clear advantage when the dimension n is high, say $n \geq 10^2$. The middle panel shows the computation time with respect to the number m of the samples for $n = 1000$ and $C = 10$. The computation time of FAPG grew slower than the one of LIBLINEAR. The right panel illustrates the computation time with respect to the parameter C for $m = 10000$ and $n = 100$. Unlike ν-SVM, the computation time of the FAPG did not change so much depending on the parameter C because it does not affect the area of feasible region. We can observe that the FAPG method is numerically more stable than LIBLINEAR with respect to the changes of the parameter C.

3.4.3.3. Runtime Breakdown

Table 3.7 shows the runtime breakdown of the FAPG method for the artificial dataset with $(m, n, C) = (10000, 1000, 10)$. The computation of the gradient $\nabla f(\boldsymbol{\alpha})$ and the function $f(\boldsymbol{\alpha})$ was the most time-consuming parts as in the case of ν-SVM. Thus skipping backtracking step and evaluation of the optimality $\|L_k(\boldsymbol{\alpha}^k - T_{L_k}(\boldsymbol{\alpha}^k))\|$ reduce the computation time significantly because it can avoid the extra computation of f and ∇f.

3.4.3.4. Effect of Each Acceleration Strategy

Figure 3.7 shows the running history of the FAPG method with various acceleration strategies for the dataset 'a8a'. C was set to 10.

The left panel depicts the violations of the optimality, i.e. the values of $\|L_k(T_{L_k}(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^k)\|$. 're' had effect on decreasing the values after $k = 762$ at which the first restart occur. 'bt+re+dec' just became unstable compared to 'bt+re' because the gradient of the logistic regression is intrinsically large near the boundary of the feasible region S_{LR}.

Table 3.8.: Computation time for benchmark datasets (sec.). '–' means that the algorithm did not converge with in 36000 seconds. '**' means that it had run out of memory. The results indicated by underline are better than LIBLINEAR.

	m	n	C	FAPG (iter)	LIBLIN
a8a	22696	123	10	19.82 (4939)	9.82
a9a	32561	123	10	35.32 (6155)	57.62
australian	690	14	10	0.45 (1738)	0.01
breast-cancer	683	10	10	0.49 (1904)	0.004
cod-rna	59535	8	10	31.40 (3718)	2.03
colon-cancer	62	2000	10	0.11 (269)	0.16
covtype	581012	54	10	2039.19 (20143)	73.74
diabetes	768	8	10	0.20 (794)	0.01
duke	44	7129	10	0.27 (327)	0.31
epsilon	400000	2000	10	27791.25 (10873)	5225.43
fourclass	862	2	10	0.21 (851)	0.001
german.numer	1000	24	10	0.56 (1741)	0.31
gisette	6000	5000	10	20.30 (1289)	0.54
heart	270	13	10	0.15 (817)	0.02
ijcnn1	35000	22	10	14.57 (2478)	0.51
ionosphere	351	34	10	0.39 (1639)	0.31
leu	38	7129	10	0.11 (148)	0.19
liver-disorders	345	6	10	0.16 (828)	0.01
madelon	2000	500	10	2.23 (910)	1035.26
mushrooms	8124	112	10	2.26 (1224)	0.09
news20.binary	19996	1355191	10	96.24 (1263)	22.15
rcv1-origin	20242	47236	10	14.41 (1293)	11.41
real-sim	72309	20958	10	142.79 (5630)	7.80
skin-nonskin	245057	3	10	247.11 (8373)	3.59
sonar	208	60	10	0.22 (1150)	0.10
splice	1000	60	10	0.66 (1689)	6.02
svmguide1	3089	4	10	0.91 (1313)	0.16
svmguide3	1243	22	10	0.49 (1263)	0.29
url	2396130	3231961	10	– (–)	–
w7a	24692	300	10	17.00 (3400)	10.69
w8a	49749	300	10	51.13 (5279)	42.77

However, 'bt+re+dec+st' could recover from the instability. It is remarkable that only 'bt+re+dec+st+mt' fell below 10^{-6} in 10000 iterations. Thus, one could argue that 'mt' had a significant effect on reducing the violation. The middle panel illustrated the gap in the objective value $f(\boldsymbol{\alpha}^k) - f(\hat{\boldsymbol{\alpha}})$ where we regarded $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{10000}$ of 'bt+re+dec+st+mt' as an optimal solution. 'bt+re+dec+st+mt' decreased the function value faster than others and found the minimum solution while oscillation occurred at the end of iteration. The right panel illustrated the value of constant L_k of the FAPG at each iteration.

3.4.3.5. Benchmark Results

We measured computation time for the benchmark datasets. The parameter C was set to 10 throughout this experiment. The experimental results are shown in Table 3.8. LIBLINEAR was efficient because it solves a less complex model (without the bias term) than (3.8). In some cases, however, FAPG solved (3.8) faster than LIBLINEAR. The computation time and iteration count of FAPG was nearly proportional to the number

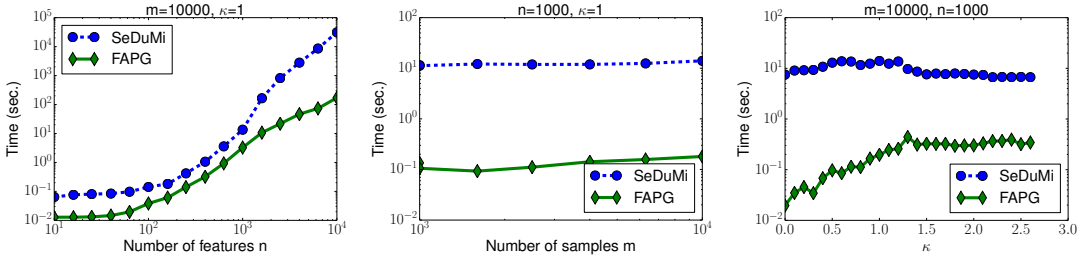


Figure 3.8.: Computation time for MM-MPM.

of samples m. On the other hand, LIBLINEAR took much longer time for 'madeon' and 'splice' although they are relatively small datasets.

Taking these results and Figure 3.6 into account, one could argue that the FAPG method exhibits stable convergence empirically.

3.4.4. MM-MPM

Next, we conducted experiments on MM-MPM (3.12). To the best of our knowledge, there are no specialized methods for MM-MPM. Thus, we compared the FAPG method only to SeDuMi [Sturm, 1999] which implements an interior point method for the large-scale second-order cone problems such as (3.12). We terminated the computations if the violation of KKT optimality is less than $\epsilon = 10^{-6}$. In the FAPG method, (η_u, η_d, δ) were set to $(1.1, 1.1, 0.8)$. The value of L_0 was set to the maximum value in the diagonal elements of $\widetilde{\Sigma}^\top \widetilde{\Sigma}$ (i.e., the coefficient matrix of the quadratic form), where $\widetilde{\Sigma} = [\Sigma_+^{1/2}, -\Sigma_-^{1/2}]$. The initial point $\boldsymbol{\mu}^0 = (\boldsymbol{\mu}_+^0, \boldsymbol{\mu}_-^0)$ was set to the origin $\mathbf{0}$.

3.4.4.1. Scalability

The computation time for the artificial datasets are shown in Figure 3.8. The left panel shows the results with respect to the number n of features for $m = 10000$ and $\kappa = 1$. The FAPG method has a clear advantage over SeDuMi for large n, say $n \geq 10^3$. The middle panel illustrates the computation time with respect to the number m of samples for $n = 2000$ and $\kappa = 1$. The computation time is nearly independent of the number m of samples because the sizes of matrices $\Sigma_o^{1/2}$ $(o \in \{+, -\})$, which are used for computing the function $f(\boldsymbol{\mu})$ and the gradient $\nabla f(\boldsymbol{\mu})$, are $n \times n$. The right panel shows the computation time with respect to the parameter κ for $m = 10000$ and $n = 2000$. We can observe that a larger value of κ leads to more computation time for the FAPG method although it is still far more efficient than SeDuMi. The effect of a larger κ on the FAPG method could be because it gives a larger feasible region S_{MPM}, which in turns leads to a larger distance between the initial point $\boldsymbol{\mu}^0$ and the optimal solution $\boldsymbol{\mu}^*$ as in the case of ν-SVM (the right panel of Figure 3.4).

Table 3.9.: Runtime breakdown of FAPG method for MM-MPM (sec.).

Function	% Time	Time	# Evals.	Time/Eval.
$\nabla f(\boldsymbol{\mu})$	75.9%	0.836	339	2.47e-3
$f(\boldsymbol{\mu})$	14.2%	0.157	119	1.32e-3
$P_{S_{\mathrm{MPM}}}(\boldsymbol{\mu})$	3.1%	0.034	383	8.88e-5
Total Runtime:		1.102		

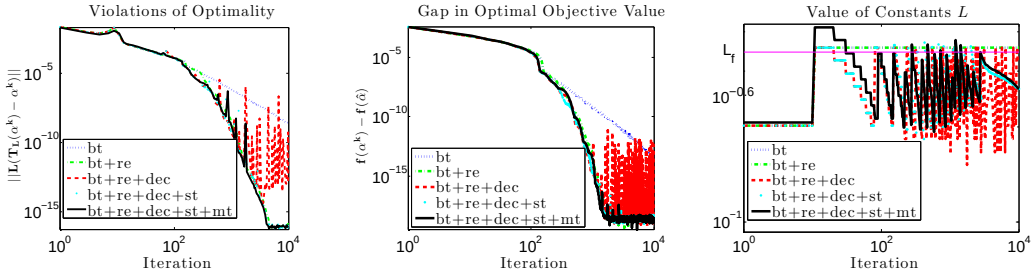


Figure 3.9.: Effect of each strategy for the APG method.

3.4.4.2. Runtime Breakdown

Table 3.9 shows the runtime breakdown of the FAPG method for the artificial dataset with $(m, n, \kappa) = (10000, 1000, 1)$. As in the case of ν-SVM (Table 3.3), the computations of the gradient $\nabla f(\boldsymbol{\mu})$ and the function value $f(\boldsymbol{\mu})$ are the most time-consuming parts. Thus, computing 'bt' and $L_k \|T_{L_k}(\boldsymbol{\mu}^k) - \boldsymbol{\mu}^k\|$ periodically, which involves the computations of $f(\boldsymbol{\mu}^k)$ and/or $\nabla f(\boldsymbol{\mu}^k)$, is effective to reduce the total runtime. The projection $P_{S_{\mathrm{MPM}}}(\boldsymbol{\mu})$ for MM-MPM shown in Section 3.3.1.4 can be computed highly efficiently.

3.4.4.3. Effect of Each Strategy

Figure 3.9 illustrates the running history of FAPG with various practical strategies for the artificial dataset with $(m, n, \kappa) = (10000, 1000, 1)$. As in the case of ν-SVM, 're' is effective in reducing the violation of optimality $L_k \|T_{L_k}(\boldsymbol{\mu}^k) - \boldsymbol{\mu}^k\|$ and the value of $f(\boldsymbol{\mu}^k) - f(\hat{\boldsymbol{\alpha}})$. 'dec' seems to make the FAPG method to be unstable, but 'st' can stabilize it. 'bt+re+dec+st' and 'bt+re+dec+st+mt' decreased $L\|T_L(\boldsymbol{\mu}^k) - \boldsymbol{\mu}^k\|$ and $f(\boldsymbol{\mu}^k) - f(\hat{\boldsymbol{\alpha}})$ slightly faster than 'bt+re'.

In the right panel, we can see that the APG method uses values smaller than L_f for L_k in most iterations, where the Lipschitz constant L_f of the gradient $\nabla f(\boldsymbol{\mu}) = \widetilde{\Sigma}^\top(\bar{\boldsymbol{x}}_+ - \bar{\boldsymbol{x}}_- + \widetilde{\Sigma}\boldsymbol{\mu})$ is known to be the largest eigenvalue of the matrix $\widetilde{\Sigma}^\top\widetilde{\Sigma}$ (recall that $\widetilde{\Sigma} = [\Sigma_+^{1/2}, -\Sigma_-^{1/2}]$ and $\boldsymbol{\mu} = (\boldsymbol{\mu}_+, \boldsymbol{\mu}_-)$).

3.4.4.4. Benchmark Results

Table 3.10 shows the computational results for the benchmark datasets. We did the experiments by setting $\kappa = \kappa_{max}/2$, but MM-MPM could not be solved for $n \geq 20000$

Table 3.10.: Computational results for MM-MPM with the linear kernel. The best results are indicated by boldface. '**' means that the algorithm could not be computed due to out of memory.

					Computation Time			Values		
								L(FAPG)		
data	m	n	κ_{max}	κ	SeDuMi	FAPG (	iter)	ave.	max.	L_f
a8a	22,696	123	9.52e-1	4.76e-1	0.664	**0.034** (	25)	1.12e0	1.21e0	1.34e0
a9a	32,561	123	9.60e-1	4.80e-1	0.166	**0.010** (	25)	1.12e0	1.21e0	1.34e0
australian	690	14	1.23e0	6.17e-1	0.054	**0.006** (	25)	1.85e0	2.00e0	2.06e0
breast-cancer	683	10	2.32e0	1.16e0	0.038	**0.005** (	30)	1.01e0	1.10e0	1.17e0
cod-rna	59,535	8	1.41e0	7.05e-1	0.068	**0.015** (	115)	1.69e-1	2.29e-1	2.34e-1
colon-cancer	62	2,000	1.85e0	9.24e-1	311.329	**0.178** (	157)	2.56e1	4.22e1	2.81e1
covtype	581,012	54	6.59e-1	3.29e-1	0.086	**0.016** (	107)	8.31e-1	1.10e0	1.07e0
diabetes	768	8	6.88e-1	3.44e-1	0.061	**0.003** (	18)	3.70e-1	3.86e-1	4.79e-1
duke	44	7,129	1.98e0	9.89e-1	13459.0	**0.660** (	291)	1.55e2	2.15e2	1.97e2
epsilon	400,000	2,000	1.16e0	5.82e-1	361.0	**1.186** (	217)	3.64e-1	6.07e-1	4.77e-1
fourclass	862	2	7.22e-1	3.61e-1	0.069	**0.003** (	12)	4.68e-1	5.47e-1	6.04e-1
german.numer	1,000	24	6.52e-1	3.26e-1	0.063	**0.005** (	33)	2.45e0	2.84e0	2.89e0
gisette	6,000	5,000	8.53e0	4.27e0	5438.2	**56.292** (	1640)	9.40e1	2.29e2	1.14e2
heart	270	13	1.10e0	5.48e-1	0.049	**0.004** (	24)	1.85e0	1.98e0	2.08e0
ijcnn1	35,000	22	9.00e-1	4.50e-1	0.068	**0.013** (	114)	4.06e-1	6.52e-1	3.04e-1
ionosphere	351	34	1.30e0	6.48e-1	0.084	**0.016** (	110)	3.49e0	4.49e0	5.15e0
leu	38	7,129	2.11e0	1.05e0	12971.377	**0.514** (	220)	1.37e2	2.68e2	1.46e2
liver-disorders	345	6	4.09e-1	2.04e-1	0.075	**0.009** (	71)	3.05e-1	4.08e-1	3.78e-1
madelon	2,000	500	6.50e-1	3.25e-1	1.964	**0.017** (	43)	2.81e-1	3.28e-1	3.31e-1
mushrooms	8,124	112	1.53e1	7.66e0	0.170	**0.166** (	840)	2.38e0	3.63e0	2.92e0
news20.binary	19,996	1,355,191	**	**	**	** (	**)	**	**	**
rcv1-origin	20,242	47,236	**	**	**	** (	**)	**	**	**
real-sim	72,309	20,958	**	**	**	** (	**)	**	**	**
skin-nonskin	245,057	3	1.63e0	8.14e-1	0.066	**0.006** (	50)	1.65e-1	1.98e-1	2.25e-1
sonar	208	60	1.29e0	6.44e-1	0.096	**0.020** (	116)	3.47e0	4.60e0	5.07e0
splice	1,000	60	1.02e0	5.09e-1	0.071	**0.009** (	39)	2.39e0	2.84e0	2.92e0
svmguide1	3,089	4	1.26e0	6.29e-1	0.070	**0.003** (	28)	1.06e-1	1.15e-1	8.61e-2
svmguide3	1,243	21	6.24e-1	3.12e-1	0.083	**0.012** (	105)	5.90e-1	8.91e-1	5.82e-1
url	2,396,130	3,231,961	**	**	**	** (	**)	**	**	**
w7a	24,692	300	1.41e0	7.03e-1	0.588	**0.036** (	112)	1.42e0	2.21e0	1.82e0
w8a	49,749	300	1.39e0	6.97e-1	0.592	**0.031** (	108)	1.45e0	2.21e0	1.88e0

Table 3.11.: Average performance of each classification model. The best results are indicated by boldface. '–' means that the cross-validation could not be done within 36000 sec. '**' means that it had run out of memory.

dataset	ν-SVM (ν)	Logistic (C)	MM-MPM (κ)	MM-FDA (κ)
a8a	84.4% (0.37)	**84.6**% (0.5)	80.7% (0.94)	84.4% (1.32)
a9a	84.7% (0.36)	**84.8**% (0.1)	80.7% (0.95)	84.7% (1.33)
australian	85.7% (0.83)	**87.7**% (0.1)	86.1% (0.25)	87.5% (1.39)
breast-cancer	97.1% (0.07)	97.4% (0.5)	**97.5**% (2.08)	97.4% (0.31)
cod-rna	**93.9**% (0.28)	**93.9**% (4.5)	93.5% (0.42)	93.7% (0.40)
colon-cancer	**88.8**% (0.29)	87.1% (6.0)	87.1% (0.18)	87.1% (1.22)
covtype	**76.3**% (0.59)	75.6% (1.0)	75.6% (0.65)	75.5% (0.92)
diabetes	**77.3**% (0.54)	**77.3**% (3.5)	74.9% (0.61)	76.8% (0.88)
duke	**88.5**% (0.02)	86.0% (5.5)	**88.5**% (0.98)	**88.5**% (1.36)
epsilon	–	–	89.6% (1.04)	**89.7**% (1.48)
fourclass	77.7% (0.68)	78.5% (0.1)	72.7% (0.57)	**78.6**% (0.10)
german.numer	76.7% (0.54)	**77.4**% (0.1)	71.9% (0.39)	77.3% (0.55)
gisette	97.4% (0.11)	97.4% (1.0)	**97.9**% (3.41)	**97.9**% (4.83)
heart	84.1% (0.40)	**84.4**% (0.1)	84.1% (0.76)	84.1% (0.15)
ijcnn1	74.7% (0.19)	91.8% (20.0)	85.9% (0.89)	**91.0**% (0.64)
ionosphere	88.3% (0.21)	**90.0**% (5.0)	86.9% (0.39)	87.8% (0.85)
leu	**94.2**% (0.02)	**94.2**% (5.0)	**94.2**% (2.10)	**94.2**% (0.21)
liver-disorders	**68.1**% (0.76)	67.9% (6.0)	63.8% (0.20)	66.4% (0.52)
madelon	59.3% (0.91)	57.7% (0.1)	59.7% (0.13)	**60.0**% (0.09)
mushrooms	**100.0**% (0.01)	**100.0**% (6.0)	**100.0**% (9.19)	**100.0**% (6.21)
news20.binary	**97.1**% (0.21)	96.5% (20.0)	**	**
rcv1-origin	97.0% (0.11)	**97.1**% (20.0)	**	**
real-sim	97.5% (0.13)	**97.6**% (15.0)	**	**
skin-nonskin	93.7% (0.32)	92.4% (6.0)	93.5% (1.61)	**93.8**% (2.10)
sonar	**79.8**% (0.40)	78.8% (0.5)	**79.8**% (0.64)	77.9% (0.91)
splice	80.9% (0.50)	80.2% (3.0)	80.6% (0.61)	**81.0**% (0.20)
svmguide1	**95.4**% (0.13)	**95.4**% (20.0)	94.4% (1.12)	91.6% (1.56)
svmguide3	**82.5**% (0.41)	82.1% (15.0)	74.3% (0.55)	81.9% (0.69)
url	–	–	**	**
w7a	**98.5**% (0.04)	**98.5**% (4.0)	96.0% (1.26)	98.2% (1.76)
w8a	98.6% (0.04)	**98.7**% (17.5)	96.1% (1.25)	98.3% (1.73)

because the sizes of the $n \times n$ matrices $\Sigma_o^{1/2}$ ($o \in \{+, -\}$) are extremely large.

The FAPG method was much faster than SeDuMi especially when the dimension is high, say $n \geq 2000$. Unlike for ν-SVM (Table 3.5), the FAPG method for MM-MPM sometimes led to larger values of L_k than the Lipschitz constant L_f. However, the average of the values of L_k is still smaller than L_f.

3.4.5. Classification Ability

Using the benchmark datasets, we compared the classification ability of classification models: ν-SVM, logistic regression, MM-MPM, and MM-FDA. Each dataset was randomly partitioned into 10 disjoint sets. We investigated the averages of the test accuracy using cross-validation over the 10 disjoint sets. We found the best parameter of the each classification model using grid search with cross-validation. The results are reported

in Table 3.11. The model that shows the best performance varies with datasets. This implies the importance of finding a suitable classification model to a dataset in order to achieve a high prediction performance. Our algorithm is useful for the purpose; it provides a unified and efficient framework for solving various classification models.

4. Conic Relaxation Methods for Polynomial Optimization Problems

4.1. Overview

Polynomial optimization problem (POP) is a problem of minimizing a polynomial objective function under polynomial equality and inequality constraints. Quadratic optimization problem (QOP) is a special case of POP where the objective function and constraints are given by quadratic functions. Since many combinatorial constraints such as binary and complementarity constraints can be expressed by using quadratic functions, various combinatorial optimization problems such as the maximum cut problem, maximum stable set problem, and quadratic assignment problem (QAP) can be formulated as QOPs. Examples of POP with higher degree polynomials include the optimal power flow problem and the sensor network localization problem. POP (and QOP) is known as NP-hard in general, and it is considered difficult to obtain the optimum value efficiently and accurately by computation methods. In order to approximate the optimum value by its lower bound, various convex conic relaxation methods have been proposed.

A conic relaxation problem is formulated as a problem of minimizing a linear function in symmetric matrix variables subject to linear equalities, linear inequalities, and a (convex) cone constraint. Important examples of cones include the positive semidefinite (PSD) cone, the doubly nonnegative (DNN) cone that is the intersection of the PSD cone and the nonnegative matrix cone, and the completely positive (CPP) cone that is the convex hull of symmetric rank-1 nonnegative matrices. The conic relaxation problems with the PSD, DNN, and CPP cone constraints are called semidefinite programming (SDP) relaxation, DNN relaxation, and CPP relaxation, respectively. The SDP relaxation has been studied extensively and proved to be very successful in solving various QOPs.

The recent theoretical findings on CPP relaxation methods for QOPs have given rise to a considerable attention in the development of the numerical method of the DNN relaxation problems; It has been revealed that for a very wide class of QOPs with nonnegative variables (for example, QOPs over a probabilistic simplex, or binary variables), there exist CPP relaxations that give the exact optimum value of QOPs [Bomze et al., 2000; Bomze and de Klerk, 2002.; de Klerk and Pasechnik, 2002; Povh and Rendl, 2007, 2009; Burer, 2009; Arima et al., 2014a]. Such CPP relaxation is often referred as 'CPP reformulation' of QOP because of the equivalence in terms of the optimum value. Although the CPP reformulation is the convex optimization problem, it is still numerically intractable since the problem of checking feasibility is NP-hard [Murty and Kabadi,

1987]. The computationally tractable DNN relaxation problem has attracted attention as a natural relaxation of the CPP reformulation. In fact, the DNN relaxation problem often gives much better lower bounds than the SDP relaxation problem.

Although the DNN problem can be naturally reduced to the standard form of the SDP problem in polynomial size, algorithms for the standard SDP such as the interior point method (IPM) suffer numerical inefficiency because that SDP consists of quadratic number of nonnegative inequality constraints with respect to the size of the variable matrix. Numerical methods to mitigate this difficulty have been proposed, for instance, SDPNAL+ [Yang et al., 2015] and the bisection and projection (BP) method [Kim et al., 2016a; Arima et al., 2017]. SDPNAL+ is based on the 2-phase augmented Lagrangian method for general DNN optimization problems; In Phase I, a reasonable initial solution is generated by a first-order method, and in Phase II, an accurate solution is obtained by the semismooth Newton method, which uses second-order derivative information. On the other hand, BP is an algorithm for a special class of conic optimization problems (COPs) including the DNN relaxation of QOPs and can be regarded as a first-order method. It reformulates the COPs to a dual problem with a single variable, and then applies the bisection method to the dual; the feasibility of a given point is checked by the accelerated proximal gradient (APG) method numerically. Both SDPNAL+ and BP use the structure of the DNN cone that it is the intersection of two simple cones, the SDP and nonnegative cones, where the metric projection onto them can be computed easily.

For general POPs beyond QOPs, there has been known theoretically powerful relaxation tool: Lasserre's hierarchy of SDP relaxations [Lasserre, 2001a,b] and its dual variant: the sum-of-square relaxation method [Parrilo, 2003], which generate a sequence of SDP relaxation problems of a given POP. The index of sequence is called *relaxation-order*. The sequence of the optimum values of the SDP relaxations monotonically converges to the optimum value of the POP under mild assumptions. As the relaxation-order increases, however, the size of SDP relaxation grows exponentially. An approach to mitigate this difficulty is exploiting the sparsity in POPs as proposed in [Waki et al., 2006]. However, solving very large scale SDP relaxations still remains very challenging problem.

For a given POP with nonnegative variables, we can think of a sequence of DNN relaxation problems by adding a nonnegative cone constraint to the (sparse) SDP relaxations of the hierarchy. However, it makes the large scale SDPs even larger. As an alternative, Kim et al. [2016b] introduced a simplified DNN relaxation problem for box and binary constrained POPs, which fits in the framework of the BP method. If all the variables in the POP are binary, their DNN relaxation is consistent with the SDP relaxations of Lasserre and Waki et al. except for the existence of the nonnegative constraints. If there is a box constrained variable, it is possibly weaker than the SDP relaxations due to the absence of the PSD constraints derived from the box constraint.[1] However, thanks to this simplification, we can solve it efficiently with the BP method. To further tighten the simplified DNN relaxation problem, they also add valid inequalities into the nonnegative

[1]That is the PSD constraints for the localizing moment matrices [Lasserre, 2001a].

cone, which results the nonnegative polyhedral cone constraints. The valid inequalities are carefully chosen not to increase the computational complexity of the metric projection onto the resulting nonnegative polyhedral cone. However, the number of valid inequalities added by their method are very limited. In particular, if all the variables are binary, no valid inequalities are added with their method. Thus, there is still room to improve their DNN relaxation formulation.

In this chapter, we provide a method for constructing a tight DNN relaxation problem, and also improves the BP method. We show that the computation of the metric projection onto the nonnegative polyhedral cone can be reduced to a problem called *isotonic regression* which has been studied extensively. There have been known efficient algorithms for cases where the polyhedral cone and isotonic regression problem have special structures as summarized in [Stout, 2014]. In order to make use of the algorithms, we propose a method to add as many valid constraints as possible into the cone while maintaining its special structure. In addition, we also propose methods to compute a good upper bound of the trace of the variable matrix over the feasible region, which is necessary for stabilizing the BP method. It is based on approximation algorithms [Iwata and Nagano, 2009; Wan et al., 2010] for submodular function minimization under a set cover constraint. Finally, based on the FAPG proposed in the previous chapter, we accelerate the process of checking feasibility in the BP method.

Furthermore, the framework of the DNN relaxation [Kim et al., 2016b] for POPs with binary and box constraints can be naturally extended to that for POPs subject to binary, box, complementarity, and polynomial equality constraints. Therefore, we first gives the generalized framework of the DNN relaxation, and then develop the methods mentioned above. The new framework is advantageous because the complementarity constraints can be handled within the nonnegative polyhedral cone constraint without changing the computational complexity of the metric projection.

This chapter is organized as follows. First, we review the BP method in Section 4.2. In Section 4.3, we provide the new framework of the DNN relaxation for POPs subject to binary, box, complementarity, and polynomial equality constraints. Then, we propose a method to add as many valid inequalities as possible while keeping the resulting DNN relaxation problem within the framework of the BP method. We also develop a method to estimate a good upper bound of the trace of the variable matrix over the feasible region, which is necessary for stabilizing the BP method. In Section 4.4, we improve the APG method used in BP by using the adaptive restarting strategy, a good initial solution, and a new termination criterion. Numerical experiments in Section 4.5 demonstrate the efficiency of the proposed method.

4.2. The Bisection and Projection Method for Conic Optimization Problems

In this section, we describe the bisection and projection (BP) method [Kim et al., 2016a; Arima et al., 2017] designed to solve a class of conic optimization problem.

4.2.1. A Class of Conic Optimization Problems (COPs)

Let $\mathcal{X}$ be a finite dimensional vector space endowed with an inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$ defined by $\|\boldsymbol{Z}\| = \sqrt{\langle \boldsymbol{Z}, \boldsymbol{Z} \rangle}$ for all $\boldsymbol{Z} \in \mathcal{X}$. For a cone $\mathbb{K} \subseteq \mathcal{X}$, let $\mathbb{K}^* = \{\boldsymbol{Y} \in \mathcal{X} \mid \langle \boldsymbol{Z}, \boldsymbol{Y} \rangle \geq 0 \text{ for all } \boldsymbol{Z} \in \mathbb{K}\}$ denote the dual cone of $\mathbb{K}$. Let $\mathrm{ri}(\mathbb{K})$ denote the relative interior of $\mathbb{K}$.

Let $\mathbb{K}_1$ and $\mathbb{K}_2$ be closed convex cones in $\mathcal{X}$ such that $\mathrm{ri}(\mathbb{K}_1) \cap \mathrm{ri}(\mathbb{K}_2) = \phi$. Then $(\mathbb{K}_1 \cap \mathbb{K}_2)^* = \mathbb{K}_1^* + \mathbb{K}_2^*$ holds. For given $\boldsymbol{Q}_0 \in \mathcal{X}$, $\boldsymbol{O} \neq \boldsymbol{H}_0 \in \mathbb{K}_1^* + \mathbb{K}_2^*$ and $\boldsymbol{Q}_i \in \mathbb{K}_1^* + \mathbb{K}_2^*$ $(i = 1, 2, \ldots, m)$, we introduce the following conic optimization problem (COP):

$$\eta^* = \min\{\langle \boldsymbol{Q}_0, \boldsymbol{Z} \rangle \mid \langle \boldsymbol{H}_0, \boldsymbol{Z} \rangle = 1,\ \langle \boldsymbol{Q}_i, \boldsymbol{Z} \rangle = 0\ (i = 1, 2, \ldots, m),\ \boldsymbol{Z} \in \mathbb{K}_1 \cap \mathbb{K}_2\}. \quad (4.1)$$

Under the condition $\boldsymbol{Z} \in \mathbb{K}_1 \cap \mathbb{K}_2$, it holds that $\langle \boldsymbol{Q}_i, \boldsymbol{Z} \rangle \geq 0$ $(i \in \{1, 2, \ldots, m\})$. Hence, by letting $\boldsymbol{H}_1 = \sum_{i=1}^m \boldsymbol{Q}_i \in \mathbb{K}_1^* + \mathbb{K}_2^*$, the COP (4.1) can be simplified as follows:

$$\eta^* = \min\{\langle \boldsymbol{Q}_0, \boldsymbol{Z} \rangle \mid \langle \boldsymbol{H}_0, \boldsymbol{Z} \rangle = 1,\ \langle \boldsymbol{H}_1, \boldsymbol{Z} \rangle = 0,\ \boldsymbol{Z} \in \mathbb{K}_1 \cap \mathbb{K}_2\}. \quad (4.2)$$

Applying the Lagrangian relaxation to the simplified COP (4.2), we obtain the following COP:

$$\eta^*(\lambda) = \min\left\{\left\langle \boldsymbol{Q}_0 + \lambda \frac{\|\boldsymbol{Q}_0\|}{\|\boldsymbol{H}_1\|} \boldsymbol{H}_1, \boldsymbol{Z} \right\rangle \;\middle|\; \langle \boldsymbol{H}_0, \boldsymbol{Z} \rangle = 1,\ \boldsymbol{Z} \in \mathbb{K}_1 \cap \mathbb{K}_2 \right\} \quad (4.3)$$

which has a single linear equality constraint. Arima et al. [2014b, Lemma 2.2] showed that $\eta^*(\lambda) \uparrow \eta^*$ as $\lambda \uparrow \infty$. Let $\boldsymbol{G}(y_0) = \boldsymbol{Q}_0 + \lambda \frac{\|\boldsymbol{Q}_0\|}{\|\boldsymbol{H}_1\|} \boldsymbol{H}_1 - y_0 \boldsymbol{H}_0$. Then, the dual of (4.3) can be described as

$$y_0^*(\lambda) = \sup\{y_0 \mid \boldsymbol{G}_\lambda(y_0) \in \mathbb{K}_1^* + \mathbb{K}_2^*\}. \quad (4.4)$$

Arima et al. [2014b, Lemma 2.3] showed the strong duality between (4.3) and (4.4) for every $\lambda \in \mathbb{R}$, i.e., $\eta^*(\lambda) = y_0^*(\lambda)$.

As shown in [Kim et al., 2016a], it holds for (4.4) that

$$\boldsymbol{G}_\lambda(y_0) \in \mathbb{K}_1^* + \mathbb{K}_2^* \text{ if } y_0 \leq y_0^*(\lambda), \quad \boldsymbol{G}_\lambda(y_0) \notin \mathbb{K}_1^* + \mathbb{K}_2^* \text{ otherwise} \quad (4.5)$$

since $\boldsymbol{H}_0 \in \mathbb{K}_1^* + \mathbb{K}_2^*$. Using the property (4.5), the approximate value of $y_0^*(\lambda)$ can be computed by the bisection method if we can check the feasibility of a given y_0. The recently proposed bisection and projection (BP) method [Kim et al., 2016a] judges the feasibility of y_0 by a numerical algorithm based on the APG method. However, since the numerical algorithm can sometimes misjudge the feasibility, there is no theoretical guarantee as to the property of the output of their BP method. To address this issue, Arima et al. [2017] improved the BP method so that it gives a valid lower bound of $y_0^*(\lambda)$ under mild conditions. In the next section, we present a numerical algorithm to check the feasibility of y_0 based on the APG method. We then introduce the BP method of [Arima et al., 2017] in Section 4.2.3.

4.2.2. The Accelerated Proximal Gradient Method for Checking Feasibility

We consider the problem to decide whether a given element $\boldsymbol{G} \in \mathcal{X}$ is in $\mathbb{K}_1^* + \mathbb{K}_2^*$. If we set $\boldsymbol{G} = \boldsymbol{G}_\lambda(y_0)$, then it is the problem to check the feasibility of y_0 in (4.4). Let us consider the following regression problem:

$$f^* = \min\Big\{\frac{1}{2}\|\boldsymbol{G} - \boldsymbol{Y}\|^2 \;\Big|\; \boldsymbol{Y} \in \mathbb{K}_1^* + \mathbb{K}_2^*\Big\} \tag{4.6}$$

$$= \min\Big\{\frac{1}{2}\|\boldsymbol{G} - (\boldsymbol{Y}_1 + \boldsymbol{Y}_2)\|^2 \;\Big|\; \boldsymbol{Y}_1 \in \mathbb{K}_1^*,\ \boldsymbol{Y}_2 \in \mathbb{K}_2^*\Big\} \tag{4.7}$$

$$= \min\Big\{f(\boldsymbol{Y}_1) := \frac{1}{2}\|\Pi_{\mathbb{K}_2}(\boldsymbol{Y}_1 - \boldsymbol{G})\|^2 \;\Big|\; \boldsymbol{Y}_1 \in \mathbb{K}_1^*\Big\}, \tag{4.8}$$

where $\boldsymbol{Y}_2 = \Pi_{\mathbb{K}_2^*}(\boldsymbol{G} - \boldsymbol{Y}_1)$. The last equality holds from Moreau's decomposition theorem [Moreau, 1962; Combettes and Reyes, 2013]: $\boldsymbol{Z} = \Pi_{\mathbb{K}}(\boldsymbol{Z}) - \Pi_{\mathbb{K}^*}(-\boldsymbol{Z})$. Obviously, $\boldsymbol{G} \in \mathbb{K}_1^* + \mathbb{K}_2^*$ if and only if $f^* = 0$.

The objective function $f(\boldsymbol{Y}_1)$ of (4.8) has the gradient $\nabla f(\boldsymbol{Y}_1) = \Pi_{\mathbb{K}_2}(\boldsymbol{Y}_1 - \boldsymbol{G})$. Since the projection operator $\Pi_{\mathbb{K}}$ onto a convex set $\mathbb{K}$ is nonexpansive [Bertsekas et al., 2003, Proposition 2.2.1], we have

$$\|\nabla f(\boldsymbol{Y}_1) - \nabla f(\boldsymbol{Y}_1')\| \le \|(\boldsymbol{Y}_1 - \boldsymbol{G}) - (\boldsymbol{Y}_1' - \boldsymbol{G})\| = \|\boldsymbol{Y}_1 - \boldsymbol{Y}_1'\| \quad (\boldsymbol{Y}_1, \boldsymbol{Y}_1' \in \mathcal{X}).$$

Therefore, the gradient $\nabla f(\boldsymbol{Y}_1)$ is Lipschitz continuous with the Lipschitz constant $L_f = 1$. The KKT conditions for (4.8) are

$$\boldsymbol{X} = \boldsymbol{G} - \boldsymbol{Y}_1 - \boldsymbol{Y}_2, \quad \langle \boldsymbol{X}, \boldsymbol{Y}_1\rangle = 0, \quad \langle \boldsymbol{X}, \boldsymbol{Y}_2\rangle = 0,$$
$$\boldsymbol{X} \in \mathbb{K}_1 \cap \mathbb{K}_2, \quad \boldsymbol{Y}_1 \in \mathbb{K}_1^*, \quad \boldsymbol{Y}_2 \in \mathbb{K}_2^*.$$

Assuming that the metric projections $\Pi_{\mathbb{K}_1}$ and $\Pi_{\mathbb{K}_2}$ onto the cones $\mathbb{K}_1$ and $\mathbb{K}_2$ can be computed easily, Kim et al. [2016a] solve (4.8) by the APG method [Beck and Teboulle, 2009] and checks whether $f^* = 0$ numerically. More precisely, they use Algorithm 4.1 to solve (4.8), where

$$g(\boldsymbol{X}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) = \max\Big\{\frac{\langle \boldsymbol{X}, \boldsymbol{Y}_1\rangle}{1 + \|\boldsymbol{X}\| + \|\boldsymbol{Y}_1\|}, \frac{\langle \boldsymbol{X}, \boldsymbol{Y}_2\rangle}{1 + \|\boldsymbol{X}\| + \|\boldsymbol{Y}_2\|}, \frac{\Pi_{\mathbb{K}_1^*}(-\boldsymbol{X})}{1 + \|\boldsymbol{X}\|}, \frac{\Pi_{\mathbb{K}_2^*}(-\boldsymbol{X})}{1 + \|\boldsymbol{X}\|}\Big\}$$

measures the violation of the KKT conditions, and judge as $\boldsymbol{G} \in (\mathbb{K}_1^* + \mathbb{K}_2^*)$ if $\|\boldsymbol{X}\| < \epsilon$, $\boldsymbol{G} \notin (\mathbb{K}_1^* + \mathbb{K}_2^*)$ otherwise. Note that $f(\boldsymbol{Y}_1^k) = \frac{1}{2}\|\boldsymbol{X}^k\|^2$. From [Beck and Teboulle, 2009, Theorem 4.4], the sublinear convergence of Algorithm 4.1

$$f(\boldsymbol{Y}_1^k) - f^* \le \frac{2L_f\|\boldsymbol{Y}_1^{\text{init}} - \boldsymbol{Y}_1^*\|^2}{(k+1)^2} \tag{4.9}$$

is ensured for any optimal solution $\boldsymbol{Y}_1^*$ of (4.8).

In order to save computation time, it is important to terminate the algorithm early and judge as $\boldsymbol{G} \notin \mathbb{K}_1^* + \mathbb{K}_2^*$ if it encounters stagnation. For the purpose, Kim et al. [2016a] employed various heuristic stopping criteria[2] for Algorithm 4.1. We omit the details here.

[2]These criteria are not presented in their paper, but implemented in their codes. We would like to thank the authors for sharing the codes.

Algorithm 4.1 The APG method for checking feasiblity.

Input: $\boldsymbol{G} \in \mathcal{X}$, $\boldsymbol{Y}_1^0 \in \mathcal{X}$, $\Pi_{\mathbb{K}_1}$, $\Pi_{\mathbb{K}_2}$, $\epsilon > 0$, $\delta > 0$, $k_{max} > 0$,
Output: $(\boldsymbol{X}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) = (\boldsymbol{X}^k, \boldsymbol{Y}_1^k, \boldsymbol{Y}_2^k)$
Initialize: $t_1 \leftarrow 1,\ L \leftarrow 1,\ \overline{\boldsymbol{Y}}_1^1 \leftarrow \boldsymbol{Y}_1^0$
for $k = 1, \ldots, k_{max}$ **do**
  $\boldsymbol{Y}_1^k \leftarrow \Pi_{\mathbb{K}_1^*}\Big(\overline{\boldsymbol{Y}}_1^k - \frac{1}{L}\Pi_{\mathbb{K}_2}(\overline{\boldsymbol{Y}}_1^k - \boldsymbol{G})\Big)$ # Step 1
  $\boldsymbol{Y}_2^{k+1} \leftarrow \Pi_{\mathbb{K}_2^*}(\boldsymbol{G} - \boldsymbol{Y}_1^k),\quad \boldsymbol{X}^k \leftarrow \boldsymbol{G} - \boldsymbol{Y}_1^k - \boldsymbol{Y}_2^k$ # KKT conditions
  if $\|\boldsymbol{X}^k\| < \epsilon$ or $g(\boldsymbol{X}^k, \boldsymbol{Y}_1^k, \boldsymbol{Y}_2^k) < \delta$ **then**
    break
  end if
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$ # Step 2
  $\overline{\boldsymbol{Y}}_1^{k+1} \leftarrow \boldsymbol{Y}_1^k + \frac{(t_k)-1}{t_{k+1}}(\boldsymbol{Y}_1^k - \boldsymbol{Y}_1^{k-1})$ # Step 3
end for

4.2.3. The Bisection and Projection Method for the COP

From the property (4.5), the approximate value of $y_0^*(\lambda)$ can be computed by the bisection method if we can check the feasibility of a given y_0. Kim et al. [2016a] proposed the bisection and projection (BP) method which uses Algorithm 4.1 to check the feasibility. Algorithm 4.1, however, sometimes misjudges an infeasible point as feasible due to $\epsilon > 0$. Conversely, it also can misjudge a feasible point as infeasible due to $\delta > 0$. Hence, there was no theoretical guarantee as to the property of the output of their BP method.

Arima et al. [2017] improved the BP method so that it generates a valid lower bound $y_0^{v\ell}$ of the optimal value $y_0^*(\lambda)$ while it also uses Algorithm 4.1 for checking the feasibility of a given y_0. Their method assumes the following two conditions for a given interior point $\boldsymbol{I}$ of $\mathbb{K}_1^*$:

(A1) We have a large enough positive number $\rho > 0$ such that $\langle \boldsymbol{I}, \boldsymbol{Z}\rangle \le \rho$ for every feasible solution $\boldsymbol{Z}$ of (4.3).

(A2) It is easy to compute $\lambda_{\min}(\boldsymbol{Z}) = \sup\{\lambda \mid \boldsymbol{Z} - \lambda \boldsymbol{I} \in \mathbb{K}_1^*\}$ for all $\boldsymbol{Z} \in \mathcal{X}$.

If $\mathbb{K}_1 = \mathbb{K}_1^* = \mathbb{S}_+^n$ and $\boldsymbol{I}$ is the identity matrix, then $\langle \boldsymbol{I}, \boldsymbol{Z}\rangle$ is the trace of $\boldsymbol{Z}$ and $\lambda_{\min}(\boldsymbol{Z})$ is the minimum eigenvalue of $\boldsymbol{Z}$. Under the first assumption (A1), the problem (4.3) is equivalent to

$$\min\Big\{\Big\langle \boldsymbol{Q}_0 + \lambda\frac{\|\boldsymbol{Q}_0\|}{\|\boldsymbol{H}_1\|}\boldsymbol{H}_1,\ \boldsymbol{Z}\Big\rangle \ \Big|\ \langle \boldsymbol{H}_0,\ \boldsymbol{Z}\rangle = 1,\ \langle \boldsymbol{I},\ \boldsymbol{Z}\rangle \le \rho,\ \boldsymbol{Z} \in \mathbb{K}_1 \cap \mathbb{K}_2\Big\}. \tag{4.10}$$

Its dual problem

$$\sup_{y_0, \boldsymbol{Y}_2, \mu}\ \{y_0 + \rho\mu \mid \boldsymbol{G}_\lambda(y_0) - \boldsymbol{Y}_2 - \mu\boldsymbol{I} \in \mathbb{K}_1^*,\ \boldsymbol{Y}_2 \in \mathbb{K}_2^*,\ \mu \le 0\} \tag{4.11}$$

is equivalent to (4.4) and has the optimal value $y_0^*(\lambda)$. Suppose that $\bar{y}_0 \in \mathbb{R}$ and $\bar{\boldsymbol{Y}}_2 \in \mathbb{K}_2^*$ is given. Let $\bar{\mu} = \min\{0, \lambda_{\min}(\boldsymbol{G}_\lambda(\bar{y}_0) - \bar{\boldsymbol{Y}}_2)\}$. Then $(y_0, \boldsymbol{Y}_1, \mu) = (\bar{y}_0, \bar{\boldsymbol{Y}}_2, \bar{\mu})$ is a feasible

solution of (4.11), and $y_0^{v\ell} = \bar{y}_0 + \rho\bar{\mu}$ gives a valid lower bound of $y_0^*(\lambda)$. If $(\bar{y}_0, \bar{\boldsymbol{Y}}_2)$ is an optimal solution of (4.4), then $y_0^{v\ell} = y_0^*(\lambda)$ because $\bar{\mu} = 0$. By incorporating the computation of the valid lower bound $y_0^{v\ell}$, Arima et al. [2017] proposed the improved BP method described in Algorithm 4.2.

Algorithm 4.2 The bisection and projection (BP) method of [Arima et al., 2017].

Input: $y_0^\ell \le y_0^u,\ tol > 0, \rho > 0, \quad \epsilon > 0, \delta > 0, \Pi_{\mathbb{K}_1}, \Pi_{\mathbb{K}_2}, k_{\max}$.
Output: $y_0^{v\ell}$
Initialize: $\hat{\boldsymbol{Y}}_1 \leftarrow \Pi_{\mathbb{K}_1^*}(\boldsymbol{G}_\lambda(\frac{y_0^\ell + y_0^u}{2}))$.
while $y_0^u - y_0^\ell > tol$ **do**
  $y_0^m \leftarrow (y_0^\ell + y_0^u)/2, \quad \hat{\boldsymbol{Y}}_1^{\text{init}} \leftarrow \hat{\boldsymbol{Y}}_1$
  $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}_1, \hat{\boldsymbol{Y}}_2) \leftarrow$ The output of Algorithm 4.1 with inputs $\big(\boldsymbol{G}_{\lambda(y_0^m)}, \hat{\boldsymbol{Y}}_1^{\text{init}}, \Pi_{\mathbb{K}_1}, \Pi_{\mathbb{K}_2}, \epsilon, \delta, k_{\max}\big)$.
  $y_0^{v\ell} \leftarrow \min\{y_0^{v\ell}, y_0^m + \rho \min\{0, \lambda_{\min}(\boldsymbol{G}_\lambda(y_0^m) - \hat{\boldsymbol{Y}}_2)\}\}$
  if $\|\hat{\boldsymbol{X}}\| < \epsilon$ **then**
    $y_0^\ell \leftarrow \max\{y_0^m, y_0^{v\ell}\}$
  else
    $y_0^u \leftarrow y_0^m, \quad y_0^\ell \leftarrow \max\{y_0^\ell, y_0^{v\ell}\}$
  end if
end while

The performance of Algorithm 4.2 highly depends on the value of ρ; using small ρ that satisfies the condition (A1) can improve the stability and efficiency (see [Arima et al., 2017]).

Remark 4.2.1. *Suppose that $\rho > 0$ is too small not to satisfy the condition (A1). Then,* (4.4) *and* (4.11) *may not be equivalent. Even in the case, $y_0^{v\ell}$ in Algorithm 4.2 still gives a valid lower bound of* (4.11) *while y_0^ℓ and y_0^u search the solution of* (4.4)*. This property of Algorithm 4.2 can be advantage if both* (4.4) *and* (4.11) *are the dual of conic relaxations of an optimization problem, where* (4.11) *is known to be tighter than* (4.4)*. In fact, Arima et al. [2017] applied Algorithm 4.2 to a DNN relaxation of the quadratic assignment problems under such settings.*

4.3. Efficient Conic Relaxations of the POPs

4.3.1. A Conic Relaxation of Polynomial Optimization Problems

In this section, we introduce a class of polynomial optimization problems (POPs) and its conic relaxation that fits into the form of (4.1). The POPs dealt in this chapter generalize the binary and box constrained POPs handled in [Kim et al., 2016b].

4.3.1.1. Polynomial Optimization Problems (POPs)

For a vector of variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ and a vector of nonnegative integers $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T \in \mathbb{Z}_+^n$, $\boldsymbol{x}^{\boldsymbol{\alpha}}$ denotes the monomial $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$. We regard every

monomial $\boldsymbol{x}^{\boldsymbol{\alpha}}$ as a function with respect to $\boldsymbol{x}$. We call $\deg(\boldsymbol{x}^{\boldsymbol{\alpha}}) = \sum_{i=1}^{n} \alpha_i$ the degree of the monomial $\boldsymbol{x}^{\boldsymbol{\alpha}}$. For a finite subset $\mathcal{F} \in \mathbb{Z}^n_+$ with cardinality $|\mathcal{F}|$, let $\mathbb{R}^{\mathcal{F}}$ denote the linear space of $|\mathcal{F}|$-dimensional real vector whose elements are indexed by the elements of $\mathcal{F}$. Each polynomial $f : \mathbb{R}^n \to \mathbb{R}$ is represented as $f(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{F}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}$ for some nonempty finite subset $\mathcal{F} \in \mathbb{Z}^n_+$ and $\boldsymbol{c} \in \mathbb{R}^{\mathcal{F}}$. We call $\mathrm{supp}(f) = \{\boldsymbol{\alpha} \in \mathcal{F} \mid c_{\boldsymbol{\alpha}} \neq 0\}$ the support of f and $\deg(f) = \max\{\deg(\boldsymbol{x}^{\boldsymbol{\alpha}}) \mid \boldsymbol{\alpha} \in \mathrm{supp}(f)\}$ the degree of f.

Let $f_i : \mathbb{R}^n \to \mathbb{R}$ $(i \in \{0, 1, 2, \ldots, m\})$ be polynomial functions. Let

$$H = \{\boldsymbol{x} \in \mathbb{R}^n \mid x_i \in [0, 1] \ (i \in I_{\mathrm{box}}), \quad x_j \in \{0, 1\} \ (j \in I_{\mathrm{bin}}), \quad \boldsymbol{x}^{\boldsymbol{\gamma}} = 0 \ (\boldsymbol{\gamma} \in \mathcal{C})\}, \tag{4.12}$$

where $\mathcal{C} \subseteq \{0, 1\}^n$, $I_{\mathrm{box}} \cup I_{\mathrm{bin}} = \{1, 2, \ldots, n\}$ and $I_{\mathrm{box}} \cap I_{\mathrm{bin}} = \phi$. We consider the following polynomial optimization problem (POP):

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ f_0(\boldsymbol{x}) \;\middle|\; (f_i(\boldsymbol{x}))^2 = 0 \ \ (i = 1, 2, \ldots, m), \quad \boldsymbol{x} \in H \right\}. \tag{4.13}$$

We mention that the POPs handled in [Kim et al., 2016b] are the special cases of (4.13) with $\mathcal{C} = \phi$ and $m = 0$. It is obvious that the constraint $(f_i(\boldsymbol{x}))^2 = 0$ is equivalent to $f_i(\boldsymbol{x}) = 0$. However, it is necessary to formulate the POP using the squared constraint $(f_i(\boldsymbol{x}))^2 = 0$ in order to generate a DNN relaxation problem that fits into the form of (4.1). Note that any polynomial inequality $g(\boldsymbol{x}) \leq 0$ can be reduced to the equality $(g(\boldsymbol{x}) - sM)^2 = 0$ by introducing a slack variable $s \in [0, 1]$, where $M \geq \inf\{g(\boldsymbol{x}) \mid \boldsymbol{x} \in H\}$.

Let us define $\boldsymbol{r} : \mathbb{Z}^n_+ \to \mathbb{Z}^n_+$ by

$$(\boldsymbol{r}(\boldsymbol{\alpha}))_i = \begin{cases} \min\{\alpha_i, 1\} & \text{if } i \in I_{\mathrm{bin}} \\ \alpha_i & \text{otherwise (i.e., } i \in I_{\mathrm{box}}) \end{cases} \tag{4.14}$$

If $\boldsymbol{x} \in H$, then $\boldsymbol{x}^{\boldsymbol{\alpha}} = \boldsymbol{x}^{\boldsymbol{r}(\boldsymbol{\alpha})}$ holds for all $\boldsymbol{\alpha} \in \mathbb{Z}^n_+$. In the succeeding discussion, we assume that $\mathrm{supp}(f_i) = \boldsymbol{r}(\mathrm{supp}(f_i))$ $(i = 0, 1, 2, \ldots, m)$ without loss of generality.

4.3.1.2. Lifting the POP with Moment Matrix

Now we lift the POP (4.13) to a problem over the space of symmetric matrix. Let

$$d = \max\left\{ \max\{\deg(f_i) \mid i = 1, 2, \ldots, m\}, \ \max\{\deg(\boldsymbol{x}^{\boldsymbol{\gamma}}) \mid \boldsymbol{\gamma} \in \mathcal{C}\}\right\}.$$

Suppose that we have $V^k \subseteq \mathbb{Z}^n_+$ $(k = 1, 2, \ldots, \ell)$ such that $\mathcal{A}^k_\omega := \{\boldsymbol{\alpha} \in \mathbb{Z}^n_+ \mid \alpha_i = 0 \ (i \notin V^k), \ \sum_{i\in V^k} \alpha_i \leq \omega\}$ satisfies the following conditions for any $\omega \geq \lceil \frac{d}{2} \rceil$:

$$\mathcal{C} \subseteq \bigcup_{k=1}^{\ell} (\mathcal{A}^k_\omega + \mathcal{A}^k_\omega), \quad \mathrm{supp}(f_0) \subseteq \bigcup_{k=1}^{\ell} (\mathcal{A}^k_\omega + \mathcal{A}^k_\omega),$$
$$\text{and} \quad \forall i \in \{1, 2, \ldots, m\}, \ \ \exists k \in \{1, 2, \ldots, \ell\} \quad \text{s.t.} \quad \mathrm{supp}(f_i) \subseteq \mathcal{A}^k_\omega. \tag{4.15}$$

We call the parameter ω *relaxation order*. Taking $\ell = 1$ and $V^1 = \{1, 2, \ldots, n\}$ is an obvious example which satisfies the condition (4.15). However, taking V^k $(k =$

$1, 2, \ldots, \ell$) as their cardinality $|V^k|$ to be small can improve the computational efficiency of the conic relaxation problem shown later. We can use the method of [Waki et al., 2006] to find such V^k ($k = 1, 2, \ldots, \ell$) from the sparsity pattern of the Hessian of polynomials f_i ($i = 0, 1, \ldots, m$).

Let $\mathbb{S}^{\mathcal{A}^k_\omega}$ denote the linear space of $|\mathcal{A}^k_\omega| \times |\mathcal{A}^k_\omega|$ real symmetric matrices whose rows and columns are indexed by the elements of $\mathcal{A}^k_\omega$. For each $k \in \{1, 2, \ldots, \ell\}$, we define the vector of monomials $\boldsymbol{x}^{\mathcal{A}^k_\omega} \in \mathbb{R}^{\mathcal{A}^k_\omega}$ by $(\boldsymbol{x}^{\mathcal{A}^k_\omega})_{\boldsymbol{\alpha}} = \boldsymbol{x}^{\boldsymbol{\alpha}}$ ($\boldsymbol{\alpha} \in \mathcal{A}^k_\omega$) and the moment matrix $\boldsymbol{x}^{\square\mathcal{A}^k_\omega} \in \mathbb{S}^{\mathcal{A}^k_\omega}$ by

$$\boldsymbol{x}^{\square\mathcal{A}^k_\omega} = \boldsymbol{x}^{\mathcal{A}^k_\omega}(\boldsymbol{x}^{\mathcal{A}^k_\omega})^T, \quad \text{i.e.,} \quad (\boldsymbol{x}^{\square\mathcal{A}^k_\omega})_{\boldsymbol{\alpha}\boldsymbol{\beta}} = \boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\beta}} \quad (\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{A}^k_\omega).$$

From the assumption (4.15) on V^k, $\boldsymbol{x}^{\square\mathcal{A}^k_\omega}$ ($k \in \{1, 2, \ldots, \ell\}$) contain all monomials in f_i for $i \in \{0, 1, \ldots, m\}$. Hence, there exists $\boldsymbol{F}_i = (\boldsymbol{F}^1_i, \boldsymbol{F}^2_i, \ldots, \boldsymbol{F}^\ell_i) \in \mathbb{S}^{\mathcal{A}_1} \times \mathbb{S}^{\mathcal{A}_2} \times \ldots \times \mathbb{S}^{\mathcal{A}_\ell}$ such that $f_i(\boldsymbol{x}) = \sum_{k=1}^{\ell} \langle \boldsymbol{F}^k_i, \boldsymbol{x}^{\square\mathcal{A}^k_\omega} \rangle$. Furthermore, for $i \in \{1, 2, \ldots, m\}$, there exists $\boldsymbol{f}_i \in \mathbb{R}^{\mathcal{A}^k_\omega}$ such that $f_i(\boldsymbol{x}) = \langle \boldsymbol{f}_i, \boldsymbol{x}^{\mathcal{A}^k_\omega} \rangle$. Hence, $(f_i(\boldsymbol{x}))^2 = \langle \boldsymbol{F}^k, \boldsymbol{x}^{\square\mathcal{A}^k_\omega} \rangle$ for some k, where $\boldsymbol{F}^k_i = \boldsymbol{f}_i \boldsymbol{f}_i^T$ is positive semidefinite. As a consequence, the POP (4.13) can be reformulated as follows:

$$\min_{\boldsymbol{Z}} \left\{ \langle \boldsymbol{F}_0, \boldsymbol{Z} \rangle \;\middle|\; \langle \boldsymbol{F}_i, \boldsymbol{Z} \rangle = 0 \ (i = 1, 2, \ldots, m), \quad \boldsymbol{Z} = (\boldsymbol{Z}^1, \boldsymbol{Z}^2, \ldots, \boldsymbol{Z}^\ell) \in M \right\}, \quad (4.16)$$

where the inner product $\langle \cdot, \cdot \rangle$ on $\mathbb{S}^{\mathcal{A}_1} \times \mathbb{S}^{\mathcal{A}_2} \times \ldots \times \mathbb{S}^{\mathcal{A}_\ell}$ is defined by $\langle \boldsymbol{A}, \boldsymbol{B} \rangle = \sum_{k=1}^{\ell} \langle \boldsymbol{A}^k, \boldsymbol{B}^k \rangle$ ($\boldsymbol{A}, \boldsymbol{B} \in \mathbb{S}^{\mathcal{A}_1} \times \mathbb{S}^{\mathcal{A}_2} \times \ldots \times \mathbb{S}^{\mathcal{A}_\ell}$), and

$$M = \{(\boldsymbol{x}^{\square\mathcal{A}^1_\omega}, \boldsymbol{x}^{\square\mathcal{A}^2_\omega}, \ldots, \boldsymbol{x}^{\square\mathcal{A}^\ell_\omega}) \in \mathbb{S}^{\mathcal{A}^1_\omega} \times \mathbb{S}^{\mathcal{A}^2_\omega} \times \ldots \times \mathbb{S}^{\mathcal{A}^\ell_\omega} \mid \boldsymbol{x} \in H\}.$$

Example 4.3.1. *Let us consider the following POP such that $n = 3$, $m = 1$, $\mathcal{C} = \{(\frac{1}{2})\}$, $I_{\rm box} = \{1\}$, and $I_{\rm bin} = \{2, 3\}$:*

$$\min_{\boldsymbol{x} \in \mathbb{R}^3} \left\{ f_0(\boldsymbol{x}) = -x_1x_2 - x_2x_3 \;\middle|\; \begin{array}{ll} (f_1(\boldsymbol{x}))^2 = (x_2 + x_3 - 1)^2 = 0, & \\ x_1x_2 = 0, \quad x_1 \in [0, 1], \quad x_2, x_3 \in \{0, 1\}. & \end{array} \right\}. \quad (4.17)$$

We give two examples of V^k and $\mathcal{A}^k_\omega$.

Case I. *Let $\ell = 1$, $V^k = \{1, 2, 3\}$, and $\omega = 1$. Then $\mathcal{A}^1_\omega = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$. We have*

$$\boldsymbol{x}^{\mathcal{A}^1_\omega} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad \text{and} \quad \boldsymbol{x}^{\square\mathcal{A}^1_\omega} = \begin{pmatrix} 1 & x_1 & x_2 & x_3 \\ x_1 & x_1^2 & x_1x_2 & x_1x_3 \\ x_2 & x_1x_2 & x_2^2 & x_2x_3 \\ x_3 & x_1x_3 & x_2x_3 & x_3^2 \end{pmatrix}.$$

If we define

$$\boldsymbol{F}^1_0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -0.5 & -0.5 \\ 0 & -0.5 & 0 & 0 \\ 0 & -0.5 & 0 & 0 \end{pmatrix}, \quad \text{and} \quad \boldsymbol{F}^1_1 = \begin{pmatrix} -1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \\ 1 \\ 1 \end{pmatrix}^T,$$

then $f_i = \langle \boldsymbol{F}^1_i, \boldsymbol{x}^{\square\mathcal{A}^1_\omega} \rangle$ holds for $i \in \{0, 1\}$.

Case II. *Let* $\ell = 2$, $V^1 = \{1,2\}$, $V^2 = \{2,3\}$, *and* $\omega = 1$. *Then* $\mathcal{A}^1_\omega = \left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}\right\}$ *and* $\mathcal{A}^2_\omega = \left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}\right\}$. *We have*

$$\boldsymbol{x}^{\mathcal{A}^1_\omega} = \begin{pmatrix}1\\x_1\\x_2\end{pmatrix} \qquad \boldsymbol{x}^{\mathcal{A}^2_\omega} = \begin{pmatrix}1\\x_2\\x_3\end{pmatrix}$$

$$\boldsymbol{x}^{\square\mathcal{A}^1_\omega} = \begin{pmatrix}1 & x_1 & x_2\\ x_1 & x_1^2 & x_1x_2\\ x_2 & x_1x_2 & x_2^2\end{pmatrix} \qquad \boldsymbol{x}^{\square\mathcal{A}^2_\omega} = \begin{pmatrix}1 & x_2 & x_3\\ x_2 & x_2^2 & x_2x_3\\ x_3 & x_2x_3 & x_3^2\end{pmatrix}.$$

If we define

$$\boldsymbol{F}_0 = (\boldsymbol{F}_0^1, \boldsymbol{F}_0^2) = \left(\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & -0.5\\ 0 & -0.5 & 0\end{pmatrix}, \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & -0.5\\ 0 & -0.5 & 0\end{pmatrix}\right),$$

$$\boldsymbol{F}_1 = (\boldsymbol{F}_1^1, \boldsymbol{F}_1^2) = \left(O, \begin{pmatrix}-1\\1\\1\end{pmatrix}\begin{pmatrix}-1\\1\\1\end{pmatrix}^T\right),$$

then $f_i(\boldsymbol{x}) = \sum_{k=1}^2 \langle \boldsymbol{F}_i^k, \boldsymbol{x}^{\square\mathcal{A}^k_\omega}\rangle$ *holds for* $i \in \{0,1\}$.

Taking $|V^k|$ *small as in Case II is advantageous in reducing the size of each block of* $\boldsymbol{F}_i$ *and speeding up the matrix computation.*

4.3.1.3. Valid Constraints and Conic Relaxation of POP

The POP (4.16) is numerically intractable in general due to the nonconvex constraint $\boldsymbol{Z} \in M$. Therefore, we will apply the convex relaxation to M in order to obtain a lower bound of the optimal value of (4.16). More precisely, we investigate valid conditions for $\boldsymbol{x}^{\square\mathcal{A}^k_\omega}$ over $\boldsymbol{x} \in H$ and translate them to convex constraints on $\boldsymbol{Z}^k$ ignoring nonconvex conditions. The most important example is that, since $\boldsymbol{x}^{\square\mathcal{A}^k_\omega}$ is a rank-1 and positive semidefinite matrix, we translate the condition to the positive semidefinite constraints on $\boldsymbol{Z}^k$:

(V1) $\boldsymbol{Z}^k \in \mathbb{S}_+^{\mathcal{A}^k_\omega} \quad (k = 1,2,\ldots,\ell)$

but ignore the rank-1 condition which is nonconvex.

Next we consider linear identities and inequalities which hold for the elements of $\boldsymbol{x}^{\square\mathcal{A}^k_\omega}$ $(k = 1,2,\ldots,\ell)$. For illustration purpose, let $x_1 \in [0,1]$ and $x_2, x_3 \in \{0,1\}$. Then, for example, $x_1x_2 = x_1x_2^2$ and $x_1 \geq x_1x_3 \geq x_1^2x_3$ hold while x_1 and x_2x_3 are not comparable in general. To describe such relations formally, we introduce an equivalence relation $\sim$ and a partial order $\succeq_{\mathrm{f}}$ on sets of the supports of monomials.

Recall the definition (4.14) of the function $\boldsymbol{r} : \mathbb{Z}_+^n \to \mathbb{Z}_+^n$. Let the binary relation $\sim$ on $\bigcup_{k=1}^\ell (\mathcal{A}^k_\omega + \mathcal{A}^k_\omega)$ be defined by $\boldsymbol{\sigma} \sim \boldsymbol{\sigma}'$ iff $\boldsymbol{r}(\boldsymbol{\sigma}) = \boldsymbol{r}(\boldsymbol{\sigma}')$. We see that $\boldsymbol{\sigma} \sim \boldsymbol{\sigma}'$ implies

$\boldsymbol{x}^{\boldsymbol{\sigma}} = \boldsymbol{x}^{\boldsymbol{\sigma}'}$ for all $\boldsymbol{x} \in H$. Then, $\sim$ is an equivalent relation since it satisfies reflexivity, symmetry, and transitivity. By letting

$$\mathcal{R} = \boldsymbol{r}(\bigcup_{k=1}^{\ell} (\mathcal{A}_\omega^k + \mathcal{A}_\omega^k)) \tag{4.18}$$

be the set of representative, the equivalence classes of $\boldsymbol{\sigma} \in \mathcal{R}$ with respect to $\sim$ can be described as

$$[\boldsymbol{\sigma}] = \Big\{ \boldsymbol{\alpha} \in \bigcup_{k=1}^{\ell} (\mathcal{A}_\omega^k + \mathcal{A}_\omega^k) \;\Big|\; \boldsymbol{r}(\boldsymbol{\alpha}) = \boldsymbol{\sigma} \Big\}.$$

For all $k, k' \in \{1, 2, \ldots, \ell\}$, we have

$$(\boldsymbol{x}^{\square \mathcal{A}_\omega^k})_{\boldsymbol{\alpha}\boldsymbol{\beta}} = \boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\beta}} = \boldsymbol{x}^{\boldsymbol{\alpha}'+\boldsymbol{\beta}'} = (\boldsymbol{x}^{\square \mathcal{A}_\omega^{k'}})_{\boldsymbol{\alpha}'\boldsymbol{\beta}'} \quad \text{if } \boldsymbol{\alpha}+\boldsymbol{\beta} \in [\boldsymbol{\sigma}] \text{ and } \boldsymbol{\alpha}'+\boldsymbol{\beta}' \in [\boldsymbol{\sigma}] \text{ for some } \boldsymbol{\sigma} \in \mathcal{R}. \tag{4.19}$$

The relation (4.19) can be simply represented by $(\boldsymbol{x}^{\square \mathcal{A}_\omega^1}, \boldsymbol{x}^{\square \mathcal{A}_\omega^2}, \ldots, \boldsymbol{x}^{\square \mathcal{A}_\omega^\ell}) = \sum_{\boldsymbol{\sigma} \in \mathcal{R}} \boldsymbol{B}_{\boldsymbol{\sigma}} \boldsymbol{x}^{\boldsymbol{\sigma}}$, where $\boldsymbol{B}_{\boldsymbol{\sigma}} = (\boldsymbol{B}_{\boldsymbol{\sigma}}^1, \boldsymbol{B}_{\boldsymbol{\sigma}}^2, \ldots, \boldsymbol{B}_{\boldsymbol{\sigma}}^\ell) \in \mathbb{S}^{\mathcal{A}_\omega^1} \times \mathbb{S}^{\mathcal{A}_\omega^2} \times \ldots \times \mathbb{S}^{\mathcal{A}_\omega^\ell}$ $(\boldsymbol{\sigma} \in \mathcal{R})$ is defined by

$$(\boldsymbol{B}_{\boldsymbol{\sigma}}^k)_{\boldsymbol{\alpha}\boldsymbol{\beta}} = \begin{cases} 1 & \text{if } (\boldsymbol{\alpha}, \boldsymbol{\beta}) \in [\boldsymbol{\sigma}] \\ 0 & \text{otherwise} \end{cases} \quad (k = 1, \ldots, \ell).$$

Note that $\{\boldsymbol{B}_{\boldsymbol{\sigma}} \mid \boldsymbol{\sigma} \in \mathcal{R}\}$ is an orthogonal (but not orthonormal) basis of linear hull of M. We translate the condition (4.19) to the constraint

(V2) $\boldsymbol{Z} = \sum_{\boldsymbol{\sigma} \in \mathcal{R}} \boldsymbol{B}_{\boldsymbol{\sigma}} \xi_{\boldsymbol{\sigma}}$

by introducing new variables $\boldsymbol{\xi} \in \mathbb{R}^{\mathcal{R}}$.

Let the binary relation $\succeq_{\mathrm{f}}$ on $\mathcal{R}$ be defined by $\boldsymbol{\sigma} \succeq_{\mathrm{f}} \boldsymbol{\sigma}'$ iff $\boldsymbol{\sigma} \leq \boldsymbol{\sigma}'$. [3] Then, $\succeq_{\mathrm{f}}$ is the partial order since it satisfies reflexivity, antisymmetry, and transitivity. $(\mathcal{R}, \succeq_{\mathrm{f}})$ is a partially ordered set. We see that $\boldsymbol{\sigma} \succeq_{\mathrm{f}} \boldsymbol{\sigma}'$ implies $\boldsymbol{x}^{\boldsymbol{\sigma}} \geq \boldsymbol{x}^{\boldsymbol{\sigma}'}$ for all $\boldsymbol{x} \in H$. We translate it to the constraints on $\boldsymbol{\xi} \in \mathbb{R}^{\mathcal{R}}$:

(V3) $\xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'}$ if $\boldsymbol{\sigma} \succeq_{\mathrm{f}} \boldsymbol{\sigma}'$.

We also have $\boldsymbol{x}^{\bar{\boldsymbol{\gamma}}} = 0$ for all $\bar{\boldsymbol{\gamma}} \in \bar{\mathcal{C}}$, where $\bar{\mathcal{C}} = \{\bar{\boldsymbol{\gamma}} \in \mathcal{R} \mid \exists \boldsymbol{\gamma} \in \mathcal{C}$ such that $\boldsymbol{\gamma} \succeq_{\mathrm{f}} \bar{\boldsymbol{\gamma}}\}$. We can translate it to

(V4) $\xi_{\bar{\boldsymbol{\gamma}}} = 0 \quad (\bar{\boldsymbol{\gamma}} \in \bar{\mathcal{C}})$

or, more simply, replace the condition (V2) by

(V2)' $\boldsymbol{Z} = \sum_{\boldsymbol{\sigma} \in \mathcal{R} \setminus \bar{\mathcal{C}}} \boldsymbol{B}_{\boldsymbol{\sigma}} \xi_{\boldsymbol{\sigma}}$.

[3] $\succeq_{\mathrm{f}}$ is used for constructing the 'full DNN relaxation' in [Kim et al., 2016b]. The subscript 'f' of $\succeq_{\mathrm{f}}$ stands for 'full'.

In the followings, we employ (V2)' instead of (V2) and (V4) and restrict our attention to the sub-partially ordered set $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm f})$. The notion of the partially ordered set plays a crucial role on constructing efficient conic relaxations of (4.16) in Section 4.3.2.

Besides the above conditions, we also know that $\boldsymbol{x}^{\Box\mathcal{A}_\omega^k}$ $(k = 1, 2, \ldots, \ell)$ are nonnegative matrices for all $\boldsymbol{x} \in H$, and $(\boldsymbol{x}^{\Box\mathcal{A}_\omega^k})_{\boldsymbol{00}} = 1$ for all $k \in \{1, \ldots, \ell\}$. We translate them to

(V5) $\xi_{\boldsymbol{\sigma}} \geq 0 \quad (\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}})$.

and

(V6) $\langle \boldsymbol{H}_0, \boldsymbol{Z}\rangle = 1$,

where $\boldsymbol{H}_0 = (\boldsymbol{H}_0^1, \boldsymbol{H}_0^2, \ldots, \boldsymbol{H}_0^\ell) \in \mathbb{S}^{\mathcal{A}_\omega^1} \times \mathbb{S}^{\mathcal{A}_\omega^2} \times \ldots \times \mathbb{S}^{\mathcal{A}_\omega^\ell}$ is defined by

$$(\boldsymbol{H}_0^k)_{\boldsymbol{\alpha\beta}} = \begin{cases} 1/\ell & \text{if } \boldsymbol{\alpha} = \boldsymbol{\beta} = \boldsymbol{0} \\ 0 & \text{otherwise.} \end{cases}$$

In consideration of the constraints (V1), (V2)', (V3), (V5), and (V6), we construct the conic relaxation problem of (4.16):

$$\zeta^{\succeq_{\rm f}} = \min_{\boldsymbol{Z}} \left\{ \langle \boldsymbol{F}_0, \boldsymbol{Z}\rangle \;\middle|\; \begin{array}{l} \langle \boldsymbol{F}_i, \boldsymbol{Z}\rangle = 0 \ (i = 1, 2, \ldots, m), \quad \langle \boldsymbol{H}_0, \boldsymbol{Z}\rangle = 1 \\ \boldsymbol{Z} \in \mathbb{K}_1 \cap \mathbb{K}_2^{\succeq_{\rm f}} \end{array} \right\}, \qquad (\mathrm{COP}^{\succeq_{\rm f}})$$

where $\mathbb{K}_1 = \mathbb{S}_+^{\mathcal{A}_\omega^1} \times \mathbb{S}_+^{\mathcal{A}_\omega^2} \times \ldots \times \mathbb{S}_+^{\mathcal{A}_\omega^\ell}$ is the Cartesian product of the positive semidefinite cones and

$$\mathbb{K}_2^{\succeq_{\rm f}} = \Big\{ \sum_{\boldsymbol{\sigma}\in\mathcal{R}\backslash\bar{\mathcal{C}}} \boldsymbol{B}_{\boldsymbol{\sigma}}\xi_{\boldsymbol{\sigma}} \;\Big|\; \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'} \ (\boldsymbol{\sigma} \succeq_{\rm f} \boldsymbol{\sigma}'), \ \ \xi_{\boldsymbol{\sigma}} \geq 0 \ (\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}) \Big\}$$

is a nonnegative polyhedral cone.

Example 4.3.2. *We show examples of $\mathcal{R}$, $\bar{\mathcal{C}}$, $[\boldsymbol{\sigma}]$, and $\boldsymbol{B}_{\boldsymbol{\sigma}}$. For Case II in Example 4.3.1, we have*

$$\mathcal{R} = \boldsymbol{r}\Big(\bigcup_{k=1}^{2} \mathcal{A}_\omega^k + \mathcal{A}_\omega^k \Big) = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \right\}$$

and $\bar{\mathcal{C}} = \left\{ \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \right\}$. Let $\boldsymbol{\sigma} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \in \mathcal{R}\backslash\bar{\mathcal{C}}$. Note that $\boldsymbol{x}^{\boldsymbol{\sigma}} = x_2$. We have $[\boldsymbol{\sigma}] = \left\{ \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \right\}$ which implies $x_2 = x_2^2$ derived from $x_2 \in \{0, 1\}$. We also have

$$(\boldsymbol{B}_{\boldsymbol{\sigma}}^1, \boldsymbol{B}_{\boldsymbol{\sigma}}^2) = \left(\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \right).$$

Note that 1 is in the place where the moment matrices have x_2 or x_2^2.

The conic relaxation problem $\mathrm{COP}^{\succeq_{\mathrm{f}}}$ fits into the form of (4.1). However, applying the BP method to $\mathrm{COP}^{\succeq_{\mathrm{f}}}$ may not efficient since the computation of the metric projection $\Pi_{\mathbb{K}_2^{\succeq_{\mathrm{f}}}}$ is not obvious. The interior point method (IPM) can be applied to solving $\mathrm{COP}^{\succeq_{\mathrm{f}}}$ by reformulating it to the standard form of the semidefinite programming problem. However, it would be computationally expensive since $\mathbb{K}_2^{\succeq_{\mathrm{f}}}$ consists of large number of inequality constraints. To cope with this difficulty, Kim et al. [2016b] proposed a further relaxation of $\mathrm{COP}^{\succeq_{\mathrm{f}}}$ by partially reducing the inequality constraints $\xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'}$ ($\boldsymbol{\sigma} \succeq_{\mathrm{f}} \boldsymbol{\sigma}'$). To describe it formally, let us introduce a *sub-partial order* $\succeq$ of $\succeq_{\mathrm{f}}$, that is, a partial order such that $\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}$ implies $\boldsymbol{\sigma} \succeq_{\mathrm{f}} \boldsymbol{\sigma}'$ for $\boldsymbol{\sigma}, \boldsymbol{\sigma}' \in \mathcal{R}\backslash\bar{\mathcal{C}}$. For a given $\succeq$, we define

$$\mathbb{K}_2^{\succeq} = \Big\{ \sum_{\boldsymbol{\sigma}\in\mathcal{R}\backslash\bar{\mathcal{C}}} \boldsymbol{B}_{\boldsymbol{\sigma}}\xi_{\boldsymbol{\sigma}} \;\Big|\; \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'} \ (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}'), \ \ \xi_{\boldsymbol{\sigma}} \geq 0 \ (\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}) \Big\}.$$

Let $\mathrm{COP}^{\succeq}$ be the problem in which $\mathbb{K}_2^{\succeq_{\mathrm{f}}}$ of $\mathrm{COP}^{\succeq_{\mathrm{f}}}$ is replaced by $\mathbb{K}_2^{\succeq}$. Let $\zeta^{\succeq}$ be the optimal value of $\mathrm{COP}^{\succeq}$. It is obvious that $\zeta^{\succeq} \leq \zeta^{\succeq_{\mathrm{f}}} \leq \eta^*$. Kim et al. [2016b] define $\succeq$ by $\succeq_{\mathrm{k}}$, where

$$\boldsymbol{\gamma} \succeq_{\mathrm{k}} \boldsymbol{\gamma}' \quad \text{iff} \quad \begin{array}{l} \forall i \in I_{\mathrm{bin}}, \ \ \gamma_i = \gamma_i' \\ \forall j \in I_{\mathrm{box}}, \ \ \exists c \geq 1, \quad \text{such that} \quad c\gamma_j = \gamma_j'. \end{array} \tag{4.20}$$

Since the structure of $\mathbb{K}_2^{\succeq_{\mathrm{k}}}$ is considerably simpler than the one of $\mathbb{K}_2^{\succeq_{\mathrm{f}}}$, the metric projection $\Pi_{\mathbb{K}_2^{\succeq_{\mathrm{k}}}}$ can be computed by the simple algorithm of [Kim et al., 2016b, Algorithm 3.3]. As a result, the conic relaxation problem $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ can be solved by the BP method efficiently. Note that the IPM for $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ is still computationally expensive since $\mathbb{K}_2^{\succeq_{\mathrm{k}}}$ consists of large number of inequality such as nonnegative constraints. We mention that if $m = 0$ and $I_{\mathrm{box}} = \phi$, then $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ is equivalent to Lasserre's hierarchy of SDP relaxations [Lasserre, 2001b] (and its dual variant: the sum-of-square relaxation method [Parrilo, 2003]) of (4.13) except for the existence of nonnegative constraints.

The definition of the sub-partial order $\succeq$ determines the tightness of the relaxation $\mathrm{COP}^{\succeq}$ and the computational complexity of $\Pi_{\mathbb{K}_2^{\succeq}}$. In Section 4.3.2, we propose several methods to construct sub-partial order $\succeq$ which leads to a stronger relaxation than $\succeq_{\mathrm{k}}$ keeping the computational complexity of $\Pi_{\mathbb{K}_2^{\succeq}}$ moderate.

4.3.2. Sub-Partial Orders and Metric Projection Computations

As we described in the previous section, it is important to find an appropriate sub-partial order $\succeq$ of $\succeq_{\mathrm{f}}$ such that the computational complexity of $\Pi_{\mathbb{K}_2^{\succeq}}$ is moderate in the application of the BP method. For the following discussion, we introduce the Hasse diagram to display the structure of a partially ordered set. For a given partial order $\succeq$ on a set S, the Hasse diagram of $(S, \succeq)$ is a directed graph $D = (V, A)$ with $V = S$ and

$$A = \{(\boldsymbol{\sigma}, \boldsymbol{\sigma}') \in S \times S \mid \boldsymbol{\sigma} \succeq \boldsymbol{\sigma}', \quad \nexists \boldsymbol{\delta} \in S\backslash\{\boldsymbol{\sigma}, \boldsymbol{\sigma}'\} \text{ s.t. } \boldsymbol{\sigma} \succeq \boldsymbol{\delta} \succeq \boldsymbol{\sigma}'\}.$$

See Figure 4.1 for example which shows the Hasse diagrams of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{f}})$ and $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{k}})$,

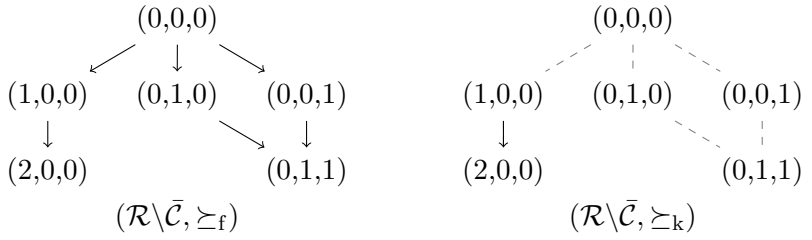


Figure 4.1.: The Hasse diagrams of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{f}})$ and $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{k}})$, where $\mathcal{R}\backslash\bar{\mathcal{C}}$ is the one given in Example 4.3.2. The Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{k}})$ has only one edge.

where $\mathcal{R}\backslash\bar{\mathcal{C}}$ is the one given in Example 4.3.2. Roughly, the number of edges in the Hasse diagram correlates with the number of inequalities and tightness of the conic relaxation problem.

4.3.2.1. Isotonic Regression

The metric projection $\Pi_{\mathbb{K}_2^{\succeq}}(\boldsymbol{G})$ of $\boldsymbol{G}$ onto $\mathbb{K}_2^{\succeq}$ is the solution of the following optimization problem:

$$\Pi_{\mathbb{K}_2^{\succeq}}(\boldsymbol{G}) = \operatorname*{argmin}_{\boldsymbol{X}} \{\|\boldsymbol{X} - \boldsymbol{G}\|^2 \mid \boldsymbol{X} \in \mathbb{K}_2^{\succeq}\}. \tag{4.21}$$

Let $w_{\boldsymbol{\sigma}} = \langle \boldsymbol{B}_{\boldsymbol{\sigma}}, \boldsymbol{B}_{\boldsymbol{\sigma}} \rangle$ and $g_{\boldsymbol{\sigma}} = \frac{1}{w_{\boldsymbol{\sigma}}} \langle \boldsymbol{G}, \boldsymbol{B}_{\boldsymbol{\sigma}} \rangle$. For $\boldsymbol{X} \in \mathbb{K}_2^{\succeq}$, it holds that

$$\begin{aligned}
\|\boldsymbol{X} - \boldsymbol{G}\|^2 &= \Big\| \sum_{\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}} \boldsymbol{B}_{\boldsymbol{\sigma}} \xi_{\boldsymbol{\sigma}} - \boldsymbol{G} \Big\|^2 \\
&= \sum_{\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}} \Big(w_{\boldsymbol{\sigma}} \xi_{\boldsymbol{\sigma}}^2 - 2\langle \boldsymbol{G}, \boldsymbol{B}_{\boldsymbol{\sigma}} \rangle \xi_{\boldsymbol{\sigma}} \Big) + \|\boldsymbol{G}\|^2 \\
&= \sum_{\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}} w_{\boldsymbol{\sigma}} (\xi_{\boldsymbol{\sigma}} - g_{\boldsymbol{\sigma}})^2 + \|\boldsymbol{G}\|^2 - \sum_{\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}} w_{\boldsymbol{\sigma}} g_{\boldsymbol{\sigma}}^2.
\end{aligned}$$

Hence, the problem (4.21) is equivalent to

$$\min_{\boldsymbol{\xi} \in \mathbb{R}^{\mathcal{R}\backslash\bar{\mathcal{C}}}} \Big\{ \sum_{\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}} w_{\boldsymbol{\sigma}} (\xi_{\boldsymbol{\sigma}} - g_{\boldsymbol{\sigma}})^2 \;\Big|\; \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'} \;\; (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}'), \quad \xi_{\boldsymbol{\sigma}} \geq 0 \;\; (\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}) \Big\}. \tag{4.22}$$

Suppose that the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq)$ has p-connected components $P_1, P_2, \ldots, P_p$. For $\boldsymbol{x} \in \mathbb{R}^{\mathcal{R}\backslash\bar{\mathcal{C}}}$, let us define $\boldsymbol{x}^i \in \mathbb{R}^{P_1}$ by $\boldsymbol{x}^i_{\boldsymbol{\sigma}} = \boldsymbol{x}_{\boldsymbol{\sigma}}$ for all $\boldsymbol{\sigma} \in P_i$ ($i \in \{1, 2, \ldots, p\}$). Then, the problem (4.22) can be devided into the following subproblems:

$$\min_{\boldsymbol{\xi}^i \in \mathbb{R}^{P_i}} \Big\{ \sum_{\boldsymbol{\sigma} \in P_i} w^i_{\boldsymbol{\sigma}} (\xi^i_{\boldsymbol{\sigma}} - g^i_{\boldsymbol{\sigma}})^2 \;\Big|\; \xi^i_{\boldsymbol{\sigma}} \geq \xi^i_{\boldsymbol{\sigma}'} \;\; (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}'), \quad \xi^i_{\boldsymbol{\sigma}} \geq 0 \;\; (\boldsymbol{\sigma} \in P_i) \Big\} \quad (i = 1, 2, \ldots, p). \tag{4.23}$$

The following Lemma gives a strategy to solve (4.23).

Lemma 4.3.1. *Let $P \subseteq \mathcal{R}\backslash\bar{\mathcal{C}}$ and the sub-partial order $\succeq$ of $\succeq_{\mathrm{f}}$ be given. Let*

$$\boldsymbol{\xi}^0 = \operatorname*{argmin}_{\boldsymbol{\xi}\in\mathbb{R}^P} \Big\{ \sum_{\boldsymbol{\sigma}\in P} w_{\boldsymbol{\sigma}}(\xi_{\boldsymbol{\sigma}} - g_{\boldsymbol{\sigma}})^2 \;\Big|\; \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'} \;\; (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}'), \quad \xi_{\boldsymbol{\sigma}} \geq 0 \;\; (\boldsymbol{\sigma} \in P) \Big\}, \tag{4.24}$$

$$\boldsymbol{\xi}^{-\infty} = \operatorname*{argmin}_{\boldsymbol{\xi}\in\mathbb{R}^P} \Big\{ \sum_{\boldsymbol{\sigma}\in P} w_{\boldsymbol{\sigma}}(\xi_{\boldsymbol{\sigma}} - g_{\boldsymbol{\sigma}})^2 \;\Big|\; \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'} \;\; (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}') \Big\}. \tag{4.25}$$

Then $\xi^0_{\boldsymbol{\sigma}} = \max\{0, \xi^{-\infty}_{\boldsymbol{\sigma}}\}$ $(\boldsymbol{\sigma} \in P)$.

Proof. Both (4.24) and (4.25) satisfy the Slater condition; there exists an interior feasible solution $\xi_{\boldsymbol{\sigma}} = \boldsymbol{x}^{\boldsymbol{\sigma}}$, where $x_1 = \ldots = x_n = 0.1$. Hence, the KKT condition is a necessary and sufficient condition for optimality. Since $\boldsymbol{\xi}^{-\infty}$ is the optimal solution of (4.25), there exists the Lagrange multipliers $\boldsymbol{\lambda}^{-\infty} \in \mathbb{R}^{(P\times P)}$ such that $(\boldsymbol{\xi}, \boldsymbol{\lambda}) = (\boldsymbol{\xi}^{-\infty}, \boldsymbol{\lambda}^{-\infty})$ satisfies the KKT conditions for (4.25):

$$\begin{cases} 2w_{\boldsymbol{\sigma}}(\xi_{\boldsymbol{\sigma}} - g_{\boldsymbol{\sigma}}) - \displaystyle\sum_{\boldsymbol{\sigma}'\in P} \lambda_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')} = 0 & (\boldsymbol{\sigma} \in P) \\ \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'}, \quad \lambda_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')} \geq 0, \quad \lambda_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')}(\xi_{\boldsymbol{\sigma}'} - \xi_{\boldsymbol{\sigma}}) = 0 & (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}') \end{cases}$$

On the other hand, the KKT conditions for (4.24) are:

$$\begin{cases} 2w_{\boldsymbol{\sigma}}(\xi_{\boldsymbol{\sigma}} - g_{\boldsymbol{\sigma}}) - \displaystyle\sum_{\boldsymbol{\sigma}'\in P} \lambda_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')} - \mu_{\boldsymbol{\sigma}} = 0 & (\boldsymbol{\sigma} \in P) \\ \xi_{\boldsymbol{\sigma}} \geq \xi_{\boldsymbol{\sigma}'}, \quad \lambda_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')} \geq 0, \quad \lambda_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')}(\xi_{\boldsymbol{\sigma}'} - \xi_{\boldsymbol{\sigma}}) = 0 & (\boldsymbol{\sigma} \succeq \boldsymbol{\sigma}') \\ \mu_{\boldsymbol{\sigma}} \geq 0, \quad \mu_{\boldsymbol{\sigma}}\xi_{\boldsymbol{\sigma}} = 0 & (\boldsymbol{\sigma} \in P) \end{cases}$$

Let $\xi^0_{\boldsymbol{\sigma}} = \max\{0, \xi^{-\infty}_{\boldsymbol{\sigma}}\}$, $\mu^0_{\boldsymbol{\sigma}} = 2w_{\boldsymbol{\sigma}} \max\{0, -\xi^{-\infty}_{\boldsymbol{\sigma}}\}$, and $\lambda^0_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')} = \lambda^{-\infty}_{(\boldsymbol{\sigma},\boldsymbol{\sigma}')}$. Since $(\boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = (\boldsymbol{\xi}^0, \boldsymbol{\lambda}^0, \boldsymbol{\mu}^0)$ satisfies the KKT conditions for (4.23), $\boldsymbol{\xi}^0$ is the optimal solution of (4.23). □

From Lemma 4.3.1, the projection onto $\mathbb{K}_2^{\succeq}$ can be constructed from the optimal solution of (4.25) with $P = P_i$. The problem (4.25) is a special case of the *isotonic regression* problem which has been studied extensively. See [Best and Chakravarti, 1990; Pardalos and Xue, 1999; Luss et al., 2010] and references there in. The fastest known exact algorithms for the isotonic regression are summarized in [Stout, 2014]. If the partial order $\succeq$ of (4.23) is given by $\succeq_f$, which is naturally defined on n-dimensional integer grid $\mathbb{Z}^n_+$, the fastest algorithm takes $O(|\mathcal{R}\backslash\bar{\mathcal{C}}|^3 \log |\mathcal{R}\backslash\bar{\mathcal{C}}|)$ time to solve (4.23). Note that the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{f}})$ is connected since every element is comparable to the origin $\mathbf{0}$. The computational complexity $O(|\mathcal{R}\backslash\bar{\mathcal{C}}|^3 \log |\mathcal{R}\backslash\bar{\mathcal{C}}|)$ of $\Pi_{\mathbb{K}_2^{\succeq_{\mathrm{f}}}}$ is much larger than the one of $\Pi_{\mathbb{K}_1}$, which requires eigenvalue decomposition, and would be a bottleneck of APG. In order to improve the computational efficiency of APG, it is necessary to choose an appropriate sub-partial order $\succeq$ of $\succeq_f$.

Here we consider two types of the partial order $\succeq$ such that the Hasse diagram of $(P_i, \succeq)$ (and $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq)$) has special structure. Assume that the Hasse diagram of $(P_i, \succeq)$ consists of a single path, i.e., every pair of elements in P_i is comparable. Such P_i is called

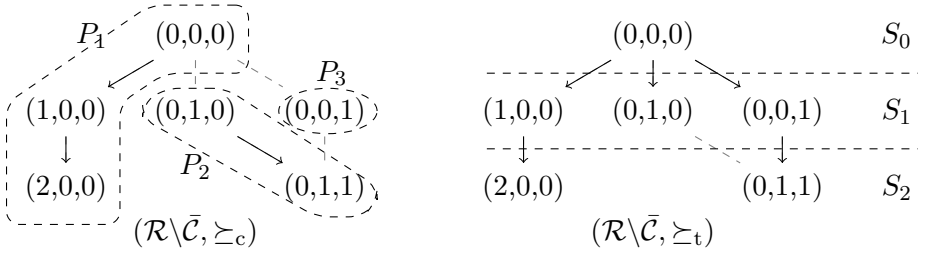


Figure 4.2.: Examples of the Hasse diagrams of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{c}})$ and $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{t}})$. The graph of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{c}})$ consists of the three vertex disjoint paths. $\{P_1, P_2, P_3\}$ is a chain decomposition of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{c}})$ with the minimum number of chains. The graph of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{c}})$ is a spanning arborescence.

a *chain* in $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq)$. The isotonic regression (4.25) with a chain $P = P_i$ can be solved by the pool-adjacent-violators algorithm [Ayer et al., 1955] in linear time $O(|P_i|)$. Next, assume that the Hasse diagram of $(P_i, \succeq)$ is a directed rooted tree in which all edges point away from the root, namely, *arborescence*. Then the isotonic regression (4.25) with $P = P_i$ can be solved in $O(|P_i| \log |P_i|)$ time by the algorithm [Pardalos and Xue, 1999] using binomial heap data structure. Since these computational complexities are less than the one of $\Pi_{\mathbb{K}_1}$, it does not deteriorate the computational efficiency of each iteration of APG. Thus, we aim to find the sub-partial order $\succeq$ where the Hasse diagram on $(P_i, \succeq)$ forms a path or arborescence, or equivalently, the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq)$ forms a set of vertex disjoint paths or arborescences.

$\succeq_{\mathrm{k}}$ is the one of the examples whose Hasse diagram is a set of vertex disjoint paths (see [Kim et al., 2016b, Lemma 4.1]). We mention that the idea of [Kim et al., 2016b, Algorithm 3.3] for computing $\Pi_{\mathbb{K}_2^{\succeq_{\mathrm{k}}}}$ is equivalent to the one of the PAV algorithm. In $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{k}})$, however, there is a few number of comparable elements, i.e., there are a few edges in the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{k}})$ (see Figure 4.1). In particular, if $I_{\mathrm{box}} = \phi$, then every pair of distinct elements in $\mathcal{R}\backslash\bar{\mathcal{C}}$ are incomparable, i.e., there are no edges in the Hasse diagram, by the definition (4.20) of $\succeq_{\mathrm{k}}$. Since the number of edges in the Hasse diagram correlates with the tightness of the conic relaxation problem, there is a possibility that $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ gives much worse lower bound than $\mathrm{COP}^{\succeq_{\mathrm{f}}}$.

In the next section, we provide methods to construct sub-partial orders whose Hasse diagrams have more edges than $\succeq_{\mathrm{k}}$ while they maintain the special structures discussed above. By using such sub-pertial orders, we can improve the tightness of the relaxation problem without increasing the computational complexity of $\Pi_{\mathbb{K}_2^{\succeq}}$ so much compared to $\Pi_{\mathbb{K}_2^{\succeq_{\mathrm{k}}}}$.

4.3.2.2. Chain Decomposition

Recall that $P \subseteq \mathcal{R}\backslash\bar{\mathcal{C}}$ is called a chain in $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{f}})$ iff every pair of elements in P is comparable with respect to $\succeq_{\mathrm{f}}$. A family of chains $\{P_1, P_2, \ldots, P_p\}$ is called a *chain decomposition* of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\mathrm{f}})$ if it is a partition of $\mathcal{R}\backslash\bar{\mathcal{C}}$. For a given chain decomposition

$\{P_1, P_2, \ldots, P_p\}$, let a sub-partial order $\succeq_{\rm c}$ of $\succeq_{\rm f}$ be defined by $\boldsymbol{\sigma} \succeq_{\rm c} \boldsymbol{\sigma'}$ for every pair $(\boldsymbol{\sigma}, \boldsymbol{\sigma'})$ in each P_i $(i = 1, 2, \ldots, p)$. Then, the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm c})$ has the connected components P_i $(i = 1, 2, \ldots, p)$ such that each P_i consists of a directed path (see Figure 4.2 for example). Then, the each subproblem (4.23) with $\succeq_{\rm c}$ can be solved in linear time $O(|P_i|)$. The number of edges in the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm c})$ is

$$\sum_{i=1}^{p}(|P_i| - 1) = |\mathcal{R}\backslash\bar{\mathcal{C}}| - p.$$

Hence, to maximize the number of edges, it is sufficient to find the chain decomposition that minimizes p.

Fulkerson [1956] showed, in his elegant proof of Dilworth's theorem[4] [Dilworth, 1950], a method to construct such chain decomposition from a maximum matching in the bipartite graph $G = (\mathcal{R}\backslash\bar{\mathcal{C}}, \mathcal{R}\backslash\bar{\mathcal{C}}, E)$, where $E = \{(u, v) \in (\mathcal{R}\backslash\bar{\mathcal{C}}) \times (\mathcal{R}\backslash\bar{\mathcal{C}}) \mid u \succeq_{\rm f} v\}$. There has been known several polynomial time algorithms for finding maximum bipartite matching (see [Ahuja et al., 1993, Section 12]). For example, using the Hopcroft-Karp algorithm [Hopcroft and Karp, 1973], we can find a maximum bipartite matching in $O(\sqrt{|\mathcal{R}\backslash\bar{\mathcal{C}}|} \cdot |E|)$ time.

4.3.2.3. Spanning Arborescence

Consider the Hasse diagram $D = (\mathcal{R}\backslash\bar{\mathcal{C}}, A)$ of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm f})$. Suppose that we have $A' \subseteq E$ such that $D' = (\mathcal{R}\backslash\bar{\mathcal{C}}, A')$ forms a set of arborescences. Let us define $\succeq_{\rm t}$ by $\boldsymbol{\sigma} \succeq_{\rm t} \boldsymbol{\sigma'}$ iff there exists a directed path from $\boldsymbol{\sigma}$ to $\boldsymbol{\sigma'}$ in D'. Then, the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm t})$ matches D'. Let $P_1, P_2, \ldots, P_p$ be the connected components in D'. The each subproblem (4.23) with $\succeq_{\rm t}$ can be solved in $O(|P_i| \log |P_i|)$ time. The number of edges in the Hasse diagram D' of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm t})$ is

$$\sum_{i=1}^{p}(|P_i| - 1) = |\mathcal{R}\backslash\bar{\mathcal{C}}| - p.$$

Hence, to maximize the number of edges, it is sufficient to minimize the number of p.

Observe that, there exists a path from $\mathbf{0}$ for any node in D since every element in $\mathcal{R}\backslash\bar{\mathcal{C}}$ is comparable with $\mathbf{0}$. Hence, fortunately, there exists a spanning arborescence in D, i.e., there exists D' such that $p = 1$. The following lemma reveals the structure of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm f})$ which is useful for finding a spanning arborescence in $D = (\mathcal{R}\backslash\bar{\mathcal{C}}, A)$.

Lemma 4.3.2. *Let $D = (\mathcal{R}\backslash\bar{\mathcal{C}}, A)$ be the Hasse diagram of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm f})$. Let $d_{\max} = \max\{\|\boldsymbol{\sigma}\|_1 \mid \boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}}\}$. Let $S_i = \{\boldsymbol{\sigma} \in \mathcal{R}\backslash\bar{\mathcal{C}} \mid \|\boldsymbol{\sigma}\|_1 = i\}$ $(i = 0, 1, \ldots, d_{\max})$. For any $d \in \{1, 2, \ldots, d_{\max}\}$ and $\boldsymbol{\sigma} \in S_d$, there exists $\boldsymbol{\sigma'} \in S_{d-1}$ such that $\boldsymbol{\sigma'} \succeq_{\rm f} \boldsymbol{\sigma}$, i.e., $(\boldsymbol{\sigma}, \boldsymbol{\sigma'}) \in A$.*

[4] Consider a subset of $\mathcal{R}\backslash\bar{\mathcal{C}}$ such that any pair of elements in the subset is not comparable with respect to $\succeq_{\rm f}$. Such subset is called an *antichain* of $(\mathcal{R}\backslash\bar{\mathcal{C}}, \succeq_{\rm f})$. Dilworth's theorem states that the minimum number of chains in any chain decomposition coincides with the maximum number of elements in any antichain. Fulkerson also showed that a method to construct an antichain with the largest number of elements.

Proof. First, observe that $\{S_0, S_1, \ldots, S_{d_{\max}}\}$ is a partition of $\mathcal{R}\backslash\bar{\mathcal{C}}$. Fix $d \in \{1, 2, \ldots, d_{\max}\}$. By the definition 4.18 of $\mathcal{R}$, we have

$$\forall \boldsymbol{\sigma} \in S_d,\ \exists \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathcal{A}^k_\omega \quad \text{such that} \quad \boldsymbol{\sigma} = \boldsymbol{r}(\boldsymbol{\alpha} + \boldsymbol{\beta}).$$

Take $i \in \{1, 2, \ldots, n\}$ such that $\sigma_i \geq 1$. Then $\boldsymbol{\sigma}' := \boldsymbol{\sigma} - \boldsymbol{e}_i$ satisfies $\|\boldsymbol{\sigma}'\|_1 = d - 1$ and $\boldsymbol{\sigma}' \succeq_{\mathrm{f}} \boldsymbol{\sigma}$. To prove $\boldsymbol{\sigma}' \in S_{d-1}$, it is sufficient to show $\boldsymbol{\sigma}' \in \mathcal{R}\backslash\bar{\mathcal{C}}$. Assume that $\beta_i \geq 1$ without loss of generality. If $i \in I_{\mathrm{box}}$, then $\boldsymbol{\sigma}' = \boldsymbol{r}(\boldsymbol{\alpha} + (\boldsymbol{\beta} - \boldsymbol{e}_i))$. If $i \in I_{\mathrm{bin}}$, then $\boldsymbol{\sigma}' = \boldsymbol{r}((\boldsymbol{\alpha} - \alpha_i \boldsymbol{e}_i) + (\boldsymbol{\beta} - \beta_i \boldsymbol{e}_i))$. Since $\boldsymbol{\alpha}$, $(\boldsymbol{\beta} - \boldsymbol{e}_i)$, $(\boldsymbol{\alpha} - \alpha_i \boldsymbol{e}_i)$, and $(\boldsymbol{\beta} - \beta_i \boldsymbol{e}_i)$ are elements of $\mathcal{A}^k_\omega$, it holds that $\boldsymbol{\sigma}' \in \mathcal{R}$. Since $\boldsymbol{\sigma} \notin \bar{\mathcal{C}}$, we have $\boldsymbol{\sigma}' \notin \bar{\mathcal{C}}$ by the definition of $\bar{\mathcal{C}}$. As a consequence, we have $\boldsymbol{\sigma}' \in S_{d-1}$. □

Based on Lemma 4.3.2, we propose Algorithm 4.3 for finding a spanning arborescence. In the output $D' = (\mathcal{R}\backslash\bar{\mathcal{C}}, A')$ of Algorithm 4.3, indegree of any vertex is at most 1. For

Algorithm 4.3 A greedy algorithm for finding a spanning arborescence.

Input: $\mathcal{R}\backslash\bar{\mathcal{C}}$
Output: $D' = (\mathcal{R}\backslash\bar{\mathcal{C}}, A')$
Initialization: Compute S_i. $d \leftarrow d_{\max}$. $A' \leftarrow \phi$.

Step 1. For each $\boldsymbol{\sigma} \in S_d$, find an element $\boldsymbol{\sigma}' \in S_{d-1}$ such that $\boldsymbol{\sigma}' \succeq_{\mathrm{f}} \boldsymbol{\sigma}$ and add $(\boldsymbol{\sigma}', \boldsymbol{\sigma})$ into A'.

Step 2. If $d = 1$, then output $D' = (\mathcal{R}\backslash\bar{\mathcal{C}}, A')$. Else, $d \leftarrow d - 1$ and go to Step 1.

all $d \in \{1, 2, \ldots, d_{\max}\}$, any vertex in S_d is connected with one of S_{d-1}. Since S_0 has a single element $\mathbf{0}$, D' is an arborescence whose root is $\mathbf{0}$.

4.3.3. An Upper Bound of the Trace of the Moment Matrix

Let $\boldsymbol{I}^k$ be the identity matrix in $\mathbb{S}^{\mathcal{A}^k_\omega}_+$ $(k \in \{1, 2, \ldots, \ell\})$ and $\boldsymbol{I} = (\boldsymbol{I}^1, \boldsymbol{I}^2, \ldots, \boldsymbol{I}^\ell) \in \mathbb{S}^{\mathcal{A}^1_\omega}_+ \times \mathbb{S}^{\mathcal{A}^2_\omega}_+ \times \ldots \times \mathbb{S}^{\mathcal{A}^\ell_\omega}_+$. As we described in Section 4.2, estimating a tight upper bound ρ of $\sup\{\langle \boldsymbol{I}, \boldsymbol{Z}\rangle \mid \boldsymbol{Z} \in U\}$ for a certain set U is important for improving the performance of the BP method (Algorithm 4.2). In this section, we assumed U to be the feasible region of (4.16) and propose computation methods for ρ. Then $\langle \boldsymbol{I}, \boldsymbol{Z}\rangle \leq \rho$ works as a valid constraint which can possibly tighten the conic relaxation $\mathrm{COP}^{\succeq}$. As mentioned in Remark 4.2.1, using such ρ is advantageous for Algorithm 4.2 although it may not satisfy the condition (A1).

Our ultimate purpose is to compute the optimal value of

$$\max_{\boldsymbol{Z}}\{\langle \boldsymbol{I}, \boldsymbol{Z}\rangle \mid \langle \boldsymbol{F}_i, \boldsymbol{Z}\rangle \ \ (i = 1, 2, \ldots, m), \quad \boldsymbol{Z} \in M\}.$$

Algorithm 4.4 The greedy algorithm for [Wan et al., 2010].

Initialize: $S \leftarrow \phi$
while $^{\exists}e \in \Omega \backslash S$ such that $f(S \cup \{e\}) - f(S) > 0$ **do**
  $s \leftarrow \mathrm{argmax}_{e \in \Omega \backslash S} \frac{f(S \cup \{e\}) - f(S)}{c(\{e\})}$
  $S \leftarrow S \cup \{s\}$
end while
Output: S

Here, we instead consider the following problem:

$$\begin{aligned} \max_{\boldsymbol{Z}} \{\langle \boldsymbol{I}, \boldsymbol{Z} \rangle \mid \boldsymbol{Z} \in M\} &= \max_{\boldsymbol{x}} \Big\{ \sum_{k=1}^{\ell} \langle \boldsymbol{I}^k, \boldsymbol{x}^{\square \mathcal{A}_\omega^k} \rangle \mid \boldsymbol{x} \in H \Big\} \\ &= \max_{\boldsymbol{x}} \Big\{ \sum_{k=1}^{\ell} \sum_{\boldsymbol{\alpha} \in \mathcal{A}_\omega^k} \boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\alpha}} \mid \boldsymbol{x} \in H \Big\}. \end{aligned} \tag{4.26}$$

Although the problem (4.26) is still numerically intractable in general, it can be reduced to submodular function minimization under a set cover constraints for which there has been known efficient approximation algorithms. By using the approximation algorithms, we can obtain the upper bound of (4.26). In Section 4.3.3.1, we introduce the framework of the submodular function minimization and approximation algorithms briefly. Then we reduce the problem (4.26) to the framework in Section 4.3.3.2.

4.3.3.1. Minimizing Submodular Function under a Set Cover

Consider a ground set Ω and a set function $f : 2^\Omega \to \mathbb{R}$. $f(\cdot)$ is called *non-decreasing* if $f(S) \leq f(T)$ for all $S \subseteq T \in 2^\Omega$. $f(\cdot)$ is called *submodular* if

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T), \quad \forall S \subseteq T \in 2^\Omega, \ \forall e \in \Omega \backslash T,$$

and *modular* if

$$f(S \cup \{e\}) - f(S) = f(T \cup \{e\}) - f(T), \quad \forall S \subseteq T \in 2^\Omega, \ \forall e \in \Omega \backslash T.$$

A set function $f : 2^\Omega \to \mathbb{R}_+$ is called *polymatroid* if it is submodular, non-decreasing, and $f(\phi) = 0$. For given subsets E_i $(i \in \Omega)$ of some ground set Θ, the function of the form $g(S) = |\bigcup_{i \in S} E_i|$ for $S \subseteq \Omega$ is called a *coverage function*. The coverage function is a typical example of the (integer-valued) polymatroid function.

For given non-decreasing submodular functions $f(\cdot)$ and $c(\cdot)$, consider the following problem

$$\min_{S \subseteq \Omega} \{c(S) \mid f(S) = f(\Omega)\}. \tag{4.27}$$

Let S^* be an optimal solution of (4.27). If $f(\cdot)$ and $c(\cdot)$ are given by integer-valued polymatroid functions, then the greedy algorithm [Wan et al., 2010] (shown in Algorithm 4.4)

produces $vh(\eta)$-approximation solution $\tilde{S}$ of (4.27) (i.e., $vh(\eta)c(S^*) \geq c(\tilde{S}) \geq c(S^*)$), where

$$v = \min_{S^*:\text{opt. sol.}} \frac{\sum_{e\in S^*} c(\{e\})}{c(S^*)}$$

is the *curvature* of the submodular cost $c(\cdot)$, $h(k) = \sum_{i=1}^{k} \frac{1}{i}$ is the k-th Harmonic number, and $\eta = \max_{e\in\Omega} f(\{e\})$. Note that a lower bound of $c(S^*)$ can be given by $c(\tilde{S})/(vh(\eta))$ while $c(\tilde{S})$ is an upper bound.

Let $c(\cdot)$ and $f(\cdot)$ of (4.27) be given by a non-decreasing submodular function and a coverage function, respectively. For such problem, Iwata and Nagano [2009] proposed an η-approximation primal-dual method based on a linear programming relaxation technique. Since their method searches not only an upper bound but also a lower bound explicitly, it would result a better lower bound than the greedy method. Each iteration of their method uses Dinkelbach's discrete Newton method whose each iteration requires to solve

$$\min_{S\in\Omega} c(S) - m(S), \tag{4.28}$$

where $m(\cdot)$ is a modular function given in the iteration. Computation time for minimizing the submodular function $c(\cdot) - m(\cdot)$ in the nested double loop can be tremendous in general. However, if $c(\cdot)$ is also given by a coverage function, then the problem (4.28) can be reduced to a maximum flow problem using the techninque of [Rhys, 1970]. There has been known various practically efficient polynomial time algorithms for solving maximum flow problems (see [Ahuja et al., 1993, Figure 7.19]). Hence, the primal-dual algorithm can be implemented efficiently.

4.3.3.2. Reducing (4.26) to (4.27)

Let us start from giving an obvious upper bound of (4.26). Fix $\boldsymbol{\alpha} \in \mathcal{A}_\omega^k$ for some $k \in \{1, 2, \ldots, \ell\}$. For $\boldsymbol{x} \in H$, we have $\boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\alpha}} \leq 1$. If $\boldsymbol{\alpha} + \boldsymbol{\alpha} \in \bar{\mathcal{C}}$ (i.e., there exists $\boldsymbol{\gamma} \in \mathcal{C}$ such that $\boldsymbol{\alpha} + \boldsymbol{\alpha} \geq \boldsymbol{\gamma}$), then $\boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\alpha}} = 0$. Thus, by letting $\mathcal{B}_\omega^k = \{\boldsymbol{\alpha} \in \mathcal{A}_\omega^k \mid \boldsymbol{\alpha} + \boldsymbol{\alpha} \notin \bar{\mathcal{C}}\}$, we have

$$\sum_{k=1}^{\ell} \sum_{\boldsymbol{\alpha}\in\mathcal{A}_\omega^k} \boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\alpha}} = \sum_{k=1}^{\ell} \sum_{\boldsymbol{\alpha}\in\mathcal{B}_\omega^k} \boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\alpha}} \leq \sum_{k=1}^{\ell} |\mathcal{B}_\omega^k|, \tag{4.29}$$

for all $\boldsymbol{x} \in H$. In order to improve the upper bound (4.29), we shall be concerned with the following problem:

$$\max_{\boldsymbol{x}} \Big\{ \sum_{k=1}^{\ell} \sum_{\boldsymbol{\alpha}\in\mathcal{B}_\omega^k} \boldsymbol{x}^{\boldsymbol{\alpha}+\boldsymbol{\alpha}} \mid \boldsymbol{x} \in H \Big\} \tag{4.30}$$

Our purpose is to obtain an upper bound (or the optimal value, if possible) of (4.30) better than $\sum_{k=1}^{\ell} |\mathcal{B}_\omega^k|$.

It is obvious that there exists an optimal solution $\boldsymbol{x}^*$ of (4.30) such that $\boldsymbol{x}^* \in \{0,1\}^n$. Thus, we can restrict our attention to $H \cap \{0,1\}^n$ instead of H. We introduce a set

variable $S_0 \subseteq \Omega := \{1, 2, \ldots, n\}$ which determines x_i to be 0 if $i \in S_0$ and 1 otherwise. Let us define

$$E_i = \{(k, \boldsymbol{\alpha}, \boldsymbol{\alpha}) \mid \boldsymbol{\alpha} \in \mathcal{B}_\omega^k,\ \alpha_i \geq 1,\ k \in \{1, 2, \ldots, \ell\}\},$$
$$F_i = \{\boldsymbol{\gamma} \mid \boldsymbol{\gamma} \in \mathcal{C},\ \gamma_i \geq 1\},$$

and the coverage functions

$$c(S_0) = \Big| \bigcup_{i \in S_0} E_i \Big| \quad \text{and} \quad f(S_0) = \Big| \bigcup_{i \in S_0} F_i \Big|.$$

Then, the function $c(\cdot)$ counts the number of zero elements in the diagonal of the moment matrices, and the function $f(\cdot)$ counts the number of satisfied complementarity constraints. Using $c(\cdot)$ and $f(\cdot)$, the problem (4.30) can be equivalently written as follows:

$$\min_{S_0 \subseteq \Omega} \{c(S_0) \mid f(S_0) = |\mathcal{C}|\}. \tag{4.31}$$

The problem (4.31) has the form of (4.27) since $f(\Omega) = |\mathcal{C}|$. If c^* is the optimal value of (4.31), then $\sum_{k=1}^{\ell} |\mathcal{B}_\omega^k| - c^*$ is the optimal value of (4.30). If $\underline{c} \geq 0$ is a lower bound of (4.31), then $\sum_{k=1}^{\ell} |\mathcal{B}_\omega^k| - \underline{c}$ gives an upper bound of (4.30). Note that the upper bound bound $\sum_{k=1}^{\ell} |\mathcal{B}_\omega^k| - \underline{c}$ is better than or equals to $\sum_{k=1}^{\ell} |\mathcal{B}_\omega^k|$.

Using the greedy algorithm (Algorithm 4.4), we obtain a $vh(\eta)$-approximation solution $\tilde{S}_0$ of (4.31). By the definition of $f(\cdot)$, we have

$$\eta = \max_{i \in \Omega} f(\{i\}) = \max_{i \in \Omega} |F_i| = \max_{i \in \Omega} \sum_{\boldsymbol{\gamma} \in \mathcal{C}} \gamma_i.$$

Let $d = \max\{\|\boldsymbol{\alpha}\|_0 \mid \boldsymbol{\alpha} \in \mathcal{A}_\omega^k,\ k \in \{1, 2, \ldots, \ell\}\}$. For any $S_0 \subseteq \Omega$, we have $\sum_{i \in S_0} c(\{i\}) \leq d \cdot c(S_0)$. This implies

$$v = \min_{S_0^* : \text{opt. sol.}} \frac{\sum_{i \in S_0^*} c(\{i\})}{c(S_0^*)} \leq d.$$

As a consequence, $\underline{c} = c(\tilde{S}_0)/(dh(\eta))$ gives a lower bound of (4.31). If we use the primal-dual algorithm [Iwata and Nagano, 2009], then we can get not only a η-approximation solution $\tilde{S}_0$ of (4.31), but also a lower bound $\underline{c}$ directly.

4.4. Improvement of the BP method

In this section, we introduce the restarting strategy into the APG method and propose a new stopping criterion for the restarting APG assuming that we know the positive value R such that $\|\boldsymbol{Y}_1^* - \boldsymbol{Y}_1^0\| \leq R$. We also provide a heuristic method for estimating R in the BP method.

4.4.1. A Restarting APG for Checking Feasibility

In Section 3.3.2, we proposed the FAPG method (Algorithm 3.2) combining various practical technique. If we apply it to (4.8), then we obtain the convergence guarantee

$$f(\boldsymbol{Y}_1^k) - f^* \leq 2\eta_u R^2 L_f \Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2, \quad \forall k \geq 3. \tag{4.32}$$

from Theorem 3.3.2.

Using this convergence guarantee, we propose a new criterion for checking whether $f^* = 0$. If $f^* = 0$, then we have

$$f(\boldsymbol{Y}_1^k) \leq 2\eta_u R^2 L_f \Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2 \tag{4.33}$$

for all $k \geq 3$. Conversely, if we observe

$$f(\boldsymbol{Y}_1^k) > 2\eta_u R^2 L_f \Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2, \tag{4.34}$$

for some $k \geq 3$, then it is an evidence of $f^* > 0$. Moreover, if $f^* > 0$, then there exists $k \geq 3$ such that (4.34) holds because $f(\boldsymbol{Y}_1^k) \geq f^* > 0$ while the right hand side converges to 0. Hence, we can use (4.34) as a criterion to detect the case $f^* > 0$ in the APG method.

When implementing FAPG, we use the stopping criterion $g(\boldsymbol{X}^k, \boldsymbol{Y}_1^k, \boldsymbol{Y}_2^k) < \delta$ as Algorithm 4.1 and (4.34). We checks the condition $g(\boldsymbol{X}, \boldsymbol{Y}_1, \boldsymbol{Y}_2) < \delta$ periodically, say in every 10 iteration, since it is time consuming due to the eigenvalue decomposition for computing $\Pi_{\mathbb{K}_1^*}$. In addition, we have found that 'bt', 'dec', and 'st' of FAPG were not effective for the problem (4.7) since they usually set L to $L_f(=1)$. Hence, we eliminate these procedures. Note that the convergence guarantee (4.32) still holds with $\eta_u = 1$ if we set $L \leftarrow L_f$. In order to make the algorithm more practical, we set the initial value of L to less than 1 and increase the value of L as $L \leftarrow \eta_r L$ ($\eta_r > 1$) if restart occurs. We have found setting L to 0.8 provides good performance for many instances. Although this modification collapses the convergence analysis of (4.32), we have observed that (4.34) still detects the case $f^* > 0$ correctly for many instances.

To summarize, we use Algorithm 4.5 for checking feasibility.

4.4.2. The BP Method with the Restarting APG

Here we consider a method for estimating R in order to use Algorithm 4.5 in the BP method. Let $y_0^\ell \leq y_0^* \leq y_0^u$ and $y_0^m = (y_0^\ell + y_0^u)/2$. Let $(\boldsymbol{Y}_1^\ell, \boldsymbol{Y}_2^\ell)$, $(\boldsymbol{Y}_1^m, \boldsymbol{Y}_2^m)$, and $(\boldsymbol{Y}_1^u, \boldsymbol{Y}_2^u)$ be the optimal solutions of (4.7) (and (4.8)) with $\boldsymbol{G} = \boldsymbol{G}_\lambda(y_0^\ell)$, $\boldsymbol{G}_\lambda(y_0^m)$, and $\boldsymbol{G}_\lambda(y_0^u)$, respectively. Then

$$\boldsymbol{Y}^\ell = \boldsymbol{Y}_1^\ell + \boldsymbol{Y}_2^\ell, \quad \boldsymbol{Y}^m = \boldsymbol{Y}_1^m + \boldsymbol{Y}_2^m, \quad \text{and} \quad \boldsymbol{Y}^u = \boldsymbol{Y}_1^u + \boldsymbol{Y}_2^u$$

are the optimal solutions of (4.6) with $\boldsymbol{G} = \boldsymbol{G}_\lambda(y_0^\ell)$, $\boldsymbol{G}_\lambda(y_0^m)$, and $\boldsymbol{G}_\lambda(y_0^u)$, respectively. Note that $\boldsymbol{G}_\lambda(y_0^\ell) = \boldsymbol{Y}^\ell$ holds.

Algorithm 4.5 A practical restarting APG for checking feasibility.

Input: $\boldsymbol{G} \in \mathcal{X}$, $\boldsymbol{Y}_1^0 \in \mathcal{X}$, $\Pi_{\mathbb{K}_1}$, $\Pi_{\mathbb{K}_2}$, $\epsilon > 0$, $\delta > 0$, $k_{max} > 0$, $\eta_r > 1$
Output: $(\boldsymbol{X}^k, \boldsymbol{Y}_1^k, \boldsymbol{Y}_2^k)$
Initialize: $t_1 \leftarrow 1$, $L_1 \leftarrow 0.8$, $\overline{\boldsymbol{Y}}_1^0 \leftarrow \boldsymbol{Y}_1^0$, $K_1 = 2$, $i = 1$, $k_{re} = 0$
for $k = 1, \ldots, k_{max}$ **do**
  $\boldsymbol{Y}_1^k \leftarrow \Pi_{\mathbb{K}_1^*}\Big(\overline{\boldsymbol{Y}}_1^k - \frac{1}{L_k}\Pi_{\mathbb{K}_2}(\overline{\boldsymbol{Y}}_1^k - \boldsymbol{G})\Big)$ # Step 1
  $\boldsymbol{Y}_2^k \leftarrow \Pi_{\mathbb{K}_2^*}(\boldsymbol{G} - \boldsymbol{Y}_1^k)$, $\boldsymbol{X}^k \leftarrow \boldsymbol{G} - \boldsymbol{Y}_1^k - \boldsymbol{Y}_2^k$ # KKT conditions
  if $\|\boldsymbol{X}^k\| < \epsilon$ or
  $\Big((k \bmod 10) == 0$ and $g(\boldsymbol{X}^k, \boldsymbol{Y}_1^k, \boldsymbol{Y}_2^k) < \delta\Big)$ or
  $\Big(f(\boldsymbol{Y}_1^k) > 2R^2L\Big(\frac{W(2k\ln 2)}{k\ln 2 - W(2k\ln 2)}\Big)^2$ and $k \geq 3\Big)$ **then**
    break
  end if
  $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$ # Step 2
  $\overline{\boldsymbol{Y}}_1^{k+1} \leftarrow \boldsymbol{Y}_1^k + \frac{(t_k)-1}{t_{k+1}}(\boldsymbol{Y}_1^k - \boldsymbol{Y}_1^{k-1})$ # Step 3
  if $\|\boldsymbol{X}^k\| - \|\boldsymbol{X}^{k-1}\| > 0$ and $k > K_i + k_{re}$ **then**
    $t^{k+1} \leftarrow 1$, $\overline{\boldsymbol{Y}}_1^{k+1} \leftarrow \boldsymbol{Y}_1^k$, $k_{re} \leftarrow k$
    $K_{i+1} \leftarrow 2K_i$, $i \leftarrow i + 1$
    $L_{k+1} \leftarrow \eta_r L_k$
  else
    $L_{k+1} \leftarrow L_k$
  end if
end for

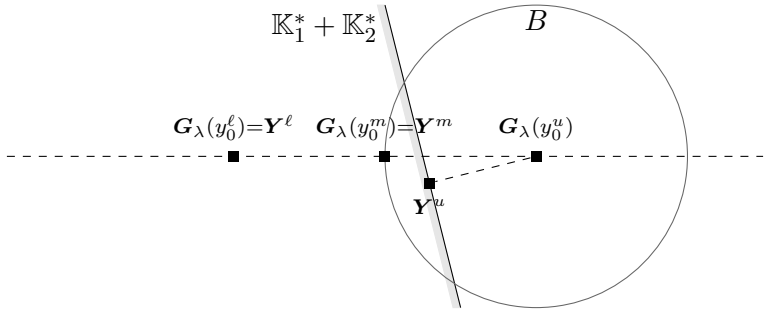


Figure 4.3.: Illustration of the optimal solutions $\boldsymbol{Y}^\ell$, $\boldsymbol{Y}^m$, and $\boldsymbol{Y}^u$.

Assume that $y_0^\ell \leq y_0^m \leq y_0^*$. Then, $\boldsymbol{G}_\lambda(y_0^m) = \boldsymbol{Y}^m$ and $f^* = 0$ holds for (4.6) with $\boldsymbol{G} = \boldsymbol{G}_\lambda(y_0^m)$. Let

$$B = \{\boldsymbol{Y} \in \mathcal{X} \mid \|\boldsymbol{Y} - \boldsymbol{G}_\lambda(y_0^u)\| \leq \|\boldsymbol{Y}^m - \boldsymbol{G}_\lambda(y_0^u)\|\}$$

be a sphere centered at $\boldsymbol{G}_\lambda(y_0^u)$. Since $\boldsymbol{Y}^u$ is the optimal solution of (4.6) with $\boldsymbol{G}_\lambda(y_0^u)$, it is closer to $\boldsymbol{G}_\lambda(y_0^u)$ than the feasible solution $\boldsymbol{Y}^m$, i.e., $\boldsymbol{Y}^u \in B$. Since $\boldsymbol{Y}^m$ lies on the line segment from $\boldsymbol{Y}^\ell$ to $\boldsymbol{G}_\lambda(y_0^u)$ and at the boundary of B, $\boldsymbol{Y}^m$ is the closest point in B from $\boldsymbol{Y}^\ell$ (see Figure 4.3). Thus, we have

$$\|\boldsymbol{Y}^m - \boldsymbol{Y}^\ell\| \leq \|\boldsymbol{Y}^u - \boldsymbol{Y}^\ell\|.$$

We also have

$$\|\boldsymbol{Y}^m - \boldsymbol{Y}^u\| \leq \|\boldsymbol{Y}^m - \boldsymbol{G}_\lambda(y_0^u)\| = \|\boldsymbol{Y}^m - \boldsymbol{Y}^\ell\|, \tag{4.35}$$

where the inequality holds from the nonexpasiveness of the projection onto a convex cone, and the equality holds by the definition of $\boldsymbol{G}_\lambda(\cdot)$. Hence, for both $\overline{\boldsymbol{Y}} = \boldsymbol{Y}^\ell$ and $\overline{\boldsymbol{Y}} = \boldsymbol{Y}^u$, we have

$$\|\overline{\boldsymbol{Y}} - \boldsymbol{Y}^m\| \leq \|\boldsymbol{Y}^\ell - \boldsymbol{Y}^u\|.$$

From the observation, it can also be expected that

$$\|\boldsymbol{Y}_1^{\text{init}} - \boldsymbol{Y}_1^m\| \leq \|\boldsymbol{Y}_1^\ell - \boldsymbol{Y}_1^u\|$$

by setting $\boldsymbol{Y}_1^{\text{init}}$ to $\boldsymbol{Y}_1^\ell$ or $\boldsymbol{Y}_1^u$. In practice, we have approximate solutions $(\hat{\boldsymbol{Y}}_1^\ell, \hat{\boldsymbol{Y}}_2^\ell)$ of $(\boldsymbol{Y}_1^\ell, \boldsymbol{Y}_2^\ell)$ and $(\hat{\boldsymbol{Y}}_1^u, \hat{\boldsymbol{Y}}_2^u)$ of $(\boldsymbol{Y}_1^u, \boldsymbol{Y}_2^u)$ computed by Algorithm 4.5. Therefore, we use $R = \gamma\|\hat{\boldsymbol{Y}}_1^\ell - \hat{\boldsymbol{Y}}_1^u\|$ for the stopping criteria (4.34), where $\gamma \geq 1$ is a parameter.

As a consequence, our BP method is described as Algorithm 4.6.

4.5. Numerical Experiments

In this section, we demonstrate the efficiency of the methods proposed in Sections 4.3 and 4.4. Specifically, we show the following results through numerical experiments.

Algorithm 4.6 The BP method with the practical restarting APG.

Input: $y_0^\ell \le y_0^u,\ tol > 0, \rho > 0, \gamma > 1, \quad \epsilon > 0, \delta > 0, \Pi_{\mathbb{K}_1}, \Pi_{\mathbb{K}_2}, k_{\max}, \eta_r$
Output: $y_0^{v\ell}$
Initialize: $\hat{\boldsymbol{Y}}_1^{\rm init} \leftarrow \Pi_{\mathbb{K}_1^*}\Big(G(\frac{y_0^\ell + y_0^u}{2})\Big)$, $R \leftarrow \infty$, $y_0^{v\ell} \leftarrow -\infty$.
$(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}_1^u, \hat{\boldsymbol{Y}}_2) \leftarrow$ The outputs of Algorithm 4.5
with the inputs $(\boldsymbol{G}_\lambda(y_0^u), \hat{\boldsymbol{Y}}_1^{\rm init}, \Pi_{\mathbb{K}_1}, \Pi_{\mathbb{K}_2}, \epsilon, \delta, k_{\max}, \eta_r, R)$.
$y_0^{v\ell} \leftarrow \min\{y_0^{v\ell}, y_0^m + \rho \min\{0, \lambda_{\min}(\boldsymbol{G}_\lambda(y_0^m) - \hat{\boldsymbol{Y}}_2)\}\}$
$y_0^\ell \leftarrow \max\{y_0^\ell, y_0^{v\ell}\}$
$(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}_1^\ell, \hat{\boldsymbol{Y}}_2) \leftarrow$ The outputs of Algorithm 4.5
with the inputs $(\boldsymbol{G}_\lambda(y_0^\ell), \hat{\boldsymbol{Y}}_1^u, \Pi_{\mathbb{K}_1}, \Pi_{\mathbb{K}_2}, \epsilon, \delta, k_{\max}, \eta_r, R)$.
while $y_0^u - y_0^\ell > tol$ **do**
  $y_0^m \leftarrow (y_0^\ell + y_0^u)/2, \quad \hat{\boldsymbol{Y}}_1^{\rm init} \leftarrow \hat{\boldsymbol{Y}}_1^u, \quad R \leftarrow \gamma \|\hat{\boldsymbol{Y}}_1^\ell - \hat{\boldsymbol{Y}}_1^u\|_F^2$.
  $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}}_1, \hat{\boldsymbol{Y}}_2) \leftarrow$ The outputs of Algorithm 4.5
  with the inputs $(\boldsymbol{G}_\lambda(y_0^m), \hat{\boldsymbol{Y}}_1^{\rm init}, \Pi_{\mathbb{K}_1}, \Pi_{\mathbb{K}_2}, \epsilon, \delta, k_{\max}, \eta_r, R)$
  $y_0^{v\ell} \leftarrow \min\{y_0^{v\ell}, y_0^m + \rho \min\{0, \lambda_{\min}(\boldsymbol{G}_\lambda(y_0^m) - \hat{\boldsymbol{Y}}_2)\}\}$
  if $\|\hat{\boldsymbol{X}}\| < \epsilon$ **then**
    $y_0^\ell \leftarrow \min\{y_0^m, y_0^{v\ell}\}$
  else
    $y_0^u \leftarrow y_0^m, \quad y_0^\ell \leftarrow \max\{y_0^\ell, y_0^{v\ell}\}$
  end if
end while

- $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ proposed in Section 4.3.2 gave better lower bounds of POPs than $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ for some instances. $\mathrm{COP}^{\succeq_{\mathrm{t}}}$ gave the same lower bounds as $\mathrm{COP}^{\succeq_{\mathrm{c}}}$.
- The approximation methods presentsed in Section 4.3.3 could find much better upper bounds ρ of the trace of the moment matrix than the obvious one (4.29). In addition, we illustrates how the better upper bound could improve the valid lower bound $y_0^{v\ell}$.
- Our BP method (Algorithm 4.6) solved the conic relaxation $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ of randomly generated POPs much faster than the original BP method of Arima et al. [2017] (Algorithm 4.2). We also compared our BP method to SDPNAL+ [Yang et al., 2015] which is a state-of-art solver for very large scale SDP problems with nonnegative constraints. Our BP method could compute solutions of $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ faster than SDPNAL+, especially for large scale and sparse POPs.

All the computations were performed in MATLAB on a Mac Pro with Intel Xeon E5 CPU (2.7 GHZ) and 64 GB memory.

4.5.1. Effect of Sub-Partial Order

In this section, we demonstrate the efficiency of conic relaxations $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ and $\mathrm{COP}^{\succeq_{\mathrm{t}}}$.Let us consider the following two instances:

$$\min_{\boldsymbol{z}} \Big\{ \sum_{\|\boldsymbol{\alpha}\|_1 \le d} \boldsymbol{z}^{\boldsymbol{\alpha}} \mid \boldsymbol{z} \in \{-1,1\}^n \Big\}, \tag{4.36}$$

$$\min_{\boldsymbol{z}} \{ \boldsymbol{z}^T \boldsymbol{E} \boldsymbol{z} \mid \boldsymbol{z} \in \{-1,1\}^n \}. \tag{4.37}$$

These instances are known to be difficult to approximate by Lasserre's hierarchy of SDP relaxations (and its dual variant: the sum-of-square (SOS) relaxation). More precisely, Sakaue et al. [2017] numerically showed that there exists n and d such that the exact optimal value of the problem (4.36) cannot be obtained by solving Lasserre's hierarchy with relaxation order $\omega < \lceil \frac{n+d-1}{2} \rceil$. Kim and Kojima [2017] showed that if d is an odd number, then Lasserre's hierarchy of (4.37) with $\omega < n/2$ gives a trivial lower bound 0 although the optimal value is 1.

We applied the affine transformation $x_i = (z_i + 1)/2$ to these problems so that $x_i \in \{0,1\}$, and constructed their conic relaxations $\mathrm{COP}^{\succeq_{\mathrm{k}}}$, $\mathrm{COP}^{\succeq_{\mathrm{c}}}$, and $\mathrm{COP}^{\succeq_{\mathrm{t}}}$. Note that since $m = 0$ and $I_{\mathrm{box}} = \phi$, $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ is equivalent to Lasserre's hierarchy of (4.36) and (4.37) except for the existence of the nonnegative constraints. The purpose here is to check whether $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ and $\mathrm{COP}^{\succeq_{\mathrm{t}}}$ can improve the lower bound obtained by $\mathrm{COP}^{\succeq_{\mathrm{k}}}$. We tested (4.36) for $n \in \{3,4,\ldots,10\}$, $d \in \{2,3,\ldots,n\}$, and $\omega \in \{\lceil \frac{d}{2} \rceil, \lceil \frac{d}{2} \rceil + 1, \ldots, \lceil \frac{n+d-1}{2} \rceil - 1\}$; and (4.37) for $n \in \{3,4,\ldots,10\}$ and $\omega \in \{\lceil \frac{d}{2} \rceil, \lceil \frac{d}{2} \rceil + 1, \ldots, \lceil \frac{n}{2} \rceil - 1\}$. In order to obtain very accurate optimal values, we translated the relaxation problems of very small instances into the standard form of SDP and solved them by SDPT3 [Tütüncü et al., 2003], which is a software implementing an interior point method, rather than by BP.

Table 4.1.: Valid lower bounds of (4.36) and (4.37).

	opt. val.	$\mathrm{COP}^{\succeq_{\mathrm{k}}}$	$\mathrm{COP}^{\succeq_{\mathrm{c}}}$	$\mathrm{COP}^{\succeq_{\mathrm{t}}}$
(4.36) $(n=4, d=2, \omega=2)$	-1	-1.3109	-1.1664	-1.1664
(4.36) $(n=6, d=2, \omega=3)$	-2	-2.3809	-2.2899	-2.2899
(4.36) $(n=8, d=2, \omega=3)$	-3	-3.4803	-3.4396	-3.4396
(4.36) $(n=10, d=2, \omega=4)$	-4	-4.4941	-4.4739	-4.4739
(4.36) $(n=10, d=2, \omega=5)$	-4	-4.4855	-4.4613	-4.4613
(4.36) $(n=8, d=4, \omega=4)$	-5	-5.8977	-5.7185	-5.7185
(4.37) $(n=7, \omega=3)$	1	0.0632	0.1446	0.1446
(4.37) $(n=9, \omega=4)$	1	0.0197	0.0651	0.0651

There were several instances that $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ gave better lower bounds than $\mathrm{COP}^{\succeq_{\mathrm{k}}}$. Table 4.1 shows the results of the instances. The lower bound obtained by $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ improved the relative error $\frac{\text{opt.val.}-\text{lower bound}}{\text{opt. val.}}$ from 4% to 46% compared to $\mathrm{COP}^{\succeq_{\mathrm{k}}}$ for these instances. On the other hand, $\mathrm{COP}^{\succeq_{\mathrm{t}}}$ gave the same lower bounds as $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ for all instances. Hence, $\mathrm{COP}^{\succeq_{\mathrm{c}}}$ would be advantageous over $\mathrm{COP}^{\succeq_{\mathrm{t}}}$ as it gives the same lower bound with less computational burden.

4.5.2. Computation of ρ and Its Effect to the BP method

4.5.2.1. Quadratic Assignment Problem (QAP)

The quadratic assignment problem (QAP) is the one of the problem such that the exact value of the trace of the moment matrix is known. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be given $r \times r$ matrices. The QAP is described as

$$\begin{aligned}\zeta^*_{\mathrm{QAP}} &= \min\left\{\langle \boldsymbol{X},\, \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T\rangle \mid \boldsymbol{X} \text{ is a permutation matrix}\right\} \\ &= \min\left\{\langle \boldsymbol{X},\, \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T\rangle \;\middle|\; \begin{array}{l} \boldsymbol{X}\boldsymbol{e} = \boldsymbol{X}^T\boldsymbol{e} = \boldsymbol{e}, \quad \boldsymbol{E} \geq \boldsymbol{X} \geq \boldsymbol{O} \\ X_{ti}X_{tj} = X_{it}X_{jt} = 0 \;\; (i,j,t \in \{1,2,\ldots,r\},\; i \neq j) \end{array} \right\},\end{aligned} \tag{4.38}$$

where the formulation (4.38) follows [Kim et al., 2016a]. We transform the problem (4.38) in the matrix variable $\boldsymbol{X} \in \mathbb{R}^{r\times r}$ to a problem in an n-dimensional column vector variable $\boldsymbol{x}$, where $n = r^2$. For every $\boldsymbol{X} = (\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^r) \in \mathbb{R}^{r\times r}$, where $\boldsymbol{x}^p$ denotes the p-th column vector of $\boldsymbol{X}$, we consider the n-dimensional vector $\boldsymbol{x} = [\boldsymbol{x}^1; \boldsymbol{x}^2; \ldots; \boldsymbol{x}^r]$ by arranging $\boldsymbol{x}^p$ $(1 \leq p \leq r)$ vertically. Let

$$\begin{aligned} J_p &= \{(p-1)r+1, (p-1)r+2, \ldots, (p-1)+r\} \;\; (1 \leq p \leq r), \\ J_{r+q} &= \{q, q+r, \ldots, q+(r-1)r\} \;\; (1 \leq q \leq r), \end{aligned}$$

and $E = \{(i,j) \mid i \in J_t, \quad j \in J_t, \quad i \neq j, \quad t \in \{1,2,\ldots,r\}\}$. Then, QAP (4.38) is equivalently formulated as follows:

$$\zeta^*_{\mathrm{QAP}} = \min\left\{ \boldsymbol{x}^T(\boldsymbol{B}\otimes\boldsymbol{A})\boldsymbol{x} \;\middle|\; \begin{array}{l} (\boldsymbol{e}^T\otimes\boldsymbol{I})\boldsymbol{x} = (\boldsymbol{I}\otimes\boldsymbol{e})\boldsymbol{x} = \boldsymbol{e}, \quad \boldsymbol{x} \in [0,1]^n \\ x_i x_j = 0 \quad (i,j) \in E \end{array} \right\}.$$

Table 4.2.: Upper bounds of $\langle \boldsymbol{I}, \boldsymbol{x}^{\square\mathcal{A}_1} \rangle$ for QAP. The approximation methods shown in Section 4.3.3 could compute much better upper bounds of $1 + r$ than the obvious value $1 + r^2$ without prior knowledge that $\boldsymbol{x}^{\square\mathcal{A}_1}$ is generated from a permutation matrix.

instance	Best $(1 + r)$	Primal-dual	Greedy	Obvious $(1 + r^2)$
tai10a	11	51	74	101
chr12a	13	73	108	145

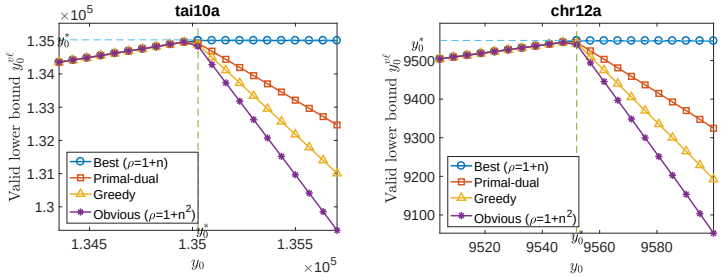


Figure 4.4.: Relation between the value of y_0 and the valid lower bound $y_0^{v\ell}$. Using a better upper bound ρ could improve the valid lower bound $y_0^{v\ell}$. The dashed line indicates the optimal value of QAP.

Let $\ell = 1$, $V^1 = \{1, 2, \ldots, n\}$, and $\omega = 1$. Then, $\boldsymbol{x}^{\mathcal{A}_1} = (1, x_1, x_2, \ldots, x_n)^T$ and $\boldsymbol{x}^{\square\mathcal{A}_1} = \boldsymbol{x}^{\mathcal{A}_1}(\boldsymbol{x}^{\mathcal{A}_1})^T$. Since $\boldsymbol{X}$ is a permutation matrix, $\boldsymbol{X}$ and $\boldsymbol{x}$ have exactly r nonzero elements with value 1. Hence, we have $\langle \boldsymbol{I}, \boldsymbol{x}^{\square\mathcal{A}_1} \rangle = 1 + r$ while the obvious upper bound given by (4.29) is $1 + r^2$.

We also can compute the upper bound of $\langle \boldsymbol{I}, \boldsymbol{x}^{\square\mathcal{A}_1} \rangle$ by the approximation methods presented in Section 4.3.3 without prior knowledge that $\boldsymbol{X}$ is a permutation matrix. Table 4.2 compares the computed upper bounds with the best value $1+r$ and the obvious value $1 + r^2$ for instances from QAPLIB [Burkard et al., 1997]. We can see that the primal-dual method [Iwata and Nagano, 2009] outperformed the greedy method [Wan et al., 2010] while they both could compute considerably better upper bounds than the obvious bounds $1 + r^2$.

In each iteration of the BP method, it fixes a value of y_0 and computes the valid lower bound $y_0^{v\ell}$ by using ρ. We regard this procedure as a function which maps a value y_0 to a valid lower bound $y_0^{v\ell}$. The function varies depending on the value of ρ. Figure 4.4 illustrates the graph of the functions $y_0 \mapsto y_0^{v\ell}$ by using each ρ shown in Table 4.2. We can see that the valid lower bounds were improved significantly by using smaller value for ρ. Especially, when $\rho = 1 + r$, a very tight valid lower bound was obtained for $y_0 \geq y_0^*(\lambda)$. From the observations, we can conclude that estimating the tight upper bound ρ of $\langle \boldsymbol{I}, \boldsymbol{x}^{\square\mathcal{A}_1} \rangle$ is very important in order to improve the performance of the BP method. The approximation methods can be used for the purpose.

Table 4.3.: Upper bounds of $\langle \boldsymbol{I}, \boldsymbol{x}^{\square\mathcal{A}_1} \rangle$ for MISP. The approximation methods shown in Section 4.3.3 could compute much better upper bounds than the obvious value $1 + |V|$.

	Primal-Dual	Greedy	$1 + \|V\|$
1dc.256	129	203	257
1et.256	130	203	257
1tc.256	130	202	257
1zc.256	129	205	257

4.5.2.2. Maximum Independent Set Problem

Let a graph $G = (V, E)$ be given, where $V = \{1, 2, \ldots, n\}$. The maximum independent set problem (MISP) is a problem to find the maximum subset V' of V such that every pair in V' is not in E (i.e., not adjacent). It can be formulated as follows:

$$\min_{\boldsymbol{x} \in \mathbb{R}^V} \Big\{ -\sum_{i \in V} x_i \;\Big|\; x_i x_j = 0 \;\; (i,j) \in E, \quad \boldsymbol{x} \in \{0,1\}^V \Big\}.$$

Let $\ell = 1$, $V^1 = \{1, 2, \ldots, n\}$, and $\omega = 1$. Then, $\boldsymbol{x}^{\mathcal{A}_1} = (1, x_1, x_2, \ldots, x_n)^T$ and $\boldsymbol{x}^{\square\mathcal{A}_1} = \boldsymbol{x}^{\mathcal{A}_1}(\boldsymbol{x}^{\mathcal{A}_1})^T$. The obvious upper bound of $\langle \boldsymbol{I}, \boldsymbol{x}^{\square\mathcal{A}_1} \rangle$ is given by $1 + |V|$. We also can compute the upper bound by the approximation methods presented in Section 4.3.3. Table 4.3 compares the computed upper bounds with the obvious value $1 + |V|$ for instances from [Sloane]. We can see that the primal-dual method [Iwata and Nagano, 2009] outperformed the greedy method [Wan et al., 2010] while they both could compute considerably better upper bounds than the obvious bounds $1 + |V|$.

4.5.3. Performance of the BP methods and SDPNAL+

In this section, we compare the performance of our BP (Algorithm 4.6) with the original BP (Algorithm 4.2) and SDPNAL+ [Yang et al., 2015] which is a state-of-the-art solver for very large scale semidefinite optimization problem with nonnegative constraints. All test problems are of the following form:

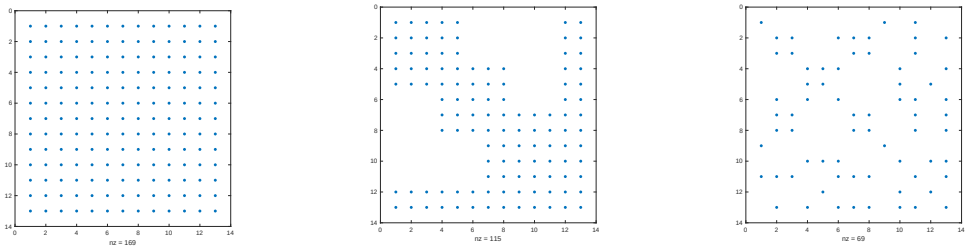
$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \{ f(\boldsymbol{x}) \mid \boldsymbol{x} \in D \cup C \}.$$

The objective function $f = \sum_{\boldsymbol{\alpha} \in \mathcal{F}} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}$ were generated as follows. For given $V^k \subseteq \{1, 2, \ldots, n\}$ $(k = 1, 2, \ldots, \ell)$ and $d \geq 1$, we defined the set of supports of f by

$$\mathcal{F} = \Big\{ \boldsymbol{\alpha} \in \mathbb{Z}^n_+ \;\Big|\; \sum_{i=1}^n \alpha_i \leq d, \quad \alpha_i = 0 \;\text{ if }\; i \notin V^k \;\; \forall k \in \{1, 2, \ldots, \ell\} \Big\}.$$

Each $c_{\boldsymbol{\alpha}}$ was taken from the uniform distribution over $[-1, 1]$. We conducted the experiments on the following 3 types of V^k:

Dense: $\ell = 1$ and $V^1 = \{1, 2, \ldots, n\}$.



Dense | Arrow $((a, b, c, \ell) = (5, 2, 2, 3))$ | Chordal $(r = 0.4)$

Figure 4.5.: Examples of sparsity pattern of the Hessian of f with $n = 13$.

Arrow: For given $\ell \geq 2$, $a \geq 3$, $b \geq 1$, and $c \geq 1$, we set

$$V^k = (\{(k-1)(a-b)\} + \{1, 2, \ldots a\}) \cup (\{(\ell-1)(a-b)+a\} + \{1, 2, \ldots, c\}).$$

Chordal: Let the number n of variables and the *radio range* $r > 0$ be given. For n points $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ taken from a uniform distribution over $[0, 1]^2$, we constructed a graph $G = (V, E)$, where $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ and $E = \{(\boldsymbol{v}_i, \boldsymbol{v}_j) \mid \|\boldsymbol{v}_i - \boldsymbol{v}_j\| \leq r\}$. We define V^k $(k = 1, 2, \ldots, \ell)$ by the maximal cliques in a chordal extension of $G = (V, E)$. The chordal extension and its maximal cliques were found by using the technique [Blair and Peyton, 1993] that is based on Choresky decomposition of the adjacency matrix of G.

Figure 4.5 illustrates the sparsity pattern of the Hessian of each f. D was set to $\{0, 1\}^n$ and $[0, 1]^n$, and C was set to $\mathbb{R}^n$ and $\{\boldsymbol{x} \mid x_i x_j = 0, \ \ (i, j) \in \mathcal{E}\}$. $\mathcal{E}$ was generated uniform randomly so that

$$\mathcal{E} \subseteq \bigcup_{k=1}^{\ell} (V^k \times V^k)$$

and $|\mathcal{E}| = 2n$. For all computation, the relaxation order ω was set to $\lceil \frac{d}{2} \rceil$.

For BPs, the parameters $(tol, \gamma, \epsilon, \delta, k_{\max}, \eta_r)$ were set to $(10^{-5}, 10, 10^{-13}, 10^{-6}, 20000, 1.1)$. For SDPNAL+, the parameter 'tol' was set to 10^{-6} and 'stopoptions' to 2 so that the solver continues to run even if it encounters some stagnations. SDPNAL+ was terminated in 20000 iterations even if the stopping criteria were not satisfied.

Table 4.4.: The computation results of the original BP (Algorithm 4.2) and the proposed BP (Algorithm 4.6). Our BP was faster than the original BP for many instances.

				$C = \mathbb{R}^n$		$C = \{\boldsymbol{x} \mid x_i x_j = 0,\ (i,j) \in \mathcal{E}\}$	
				original BP	proposed BP	original BP	proposed BP
	type	d	n	LB (sec,iter)	LB (sec,iter)	LB (sec,iter)	LB (sec,iter)
$D = \{0,1\}^n$	dense	2	200	-5.723307e2 (1.01e2, 13)	-5.723122e2 (9.58e1, 17)	-3.021572e2 (5.93e1, 12)	-3.021530e2 (7.74e1, 15)
		2	400	-1.545884e3 (6.40e2, 16)	-1.545843e3 (6.16e2, 18)	-8.793695e2 (6.05e2, 14)	-8.793639e2 (6.27e2, 17)
		3	10	-1.002771e1 (8.37e0, 13)	-1.002771e1 (6.90e0, 13)	-6.234222e0 (4.49e0, 12)	-6.234199e0 (4.14e0, 12)
		3	20	-3.144437e1 (9.23e1, 12)	-3.144428e1 (9.07e1, 12)	-1.526608e1 (5.27e1, 13)	-1.526650e1 (6.68e1, 14)
		5	10	-2.074843e1 (3.56e1, 14)	-2.074848e1 (3.38e1, 14)	-6.017181e0 (1.68e1, 13)	-6.017179e0 (1.56e1, 13)
	Arrow ($a = 10$, $b = 2$, $c = 2$)	2	324 ($\ell = 40$)	-2.226768e2 (4.51e1, 16)	-2.226722e2 (4.30e1, 16)	-1.005919e2 (5.31e1, 18)	-1.005939e2 (3.63e1, 18)
		2	644 ($\ell = 80$)	-4.437694e2 (1.21e2, 16)	-4.437684e2 (1.18e2, 16)	-1.740101e2 (9.62e1, 18)	-1.740066e2 (6.65e1, 18)
		3	244 ($\ell = 30$)	-3.307368e2 (7.47e2, 19)	-3.307385e2 (5.50e2, 20)	-9.463120e1 (3.47e2, 19)	-9.463072e1 (3.00e2, 19)
		3	484 ($\ell = 60$)	-6.612551e2 (1.29e3, 19)	-6.612425e2 (9.92e2, 19)	-1.843124e2 (7.08e2, 19)	-1.843163e2 (5.09e2, 19)
		5	52 ($\ell = 6$)	-1.628055e2 (1.83e3, 19)	-1.628054e2 (1.66e3, 21)	-1.893079e1 (8.50e2, 19)	-1.893073e1 (7.04e2, 19)
	chordal	2	400	-3.482365e2 (4.11e2, 18)	-3.482376e2 (3.24e2, 18)	-8.640456e1 (2.95e2, 19)	-8.640917e1 (3.21e2, 19)
		2	800	-1.206950e3 (1.49e3, 19)	-1.206964e3 (1.38e3, 19)	-1.973497e2 (1.06e3, 19)	-1.973530e2 (1.47e3, 19)
		3	100	-5.456033e1 (4.12e1, 17)	-5.456046e1 (2.52e1, 17)	-5.169589e1 (4.36e1, 16)	-5.169582e1 (2.91e1, 16)
		3	200	-2.065787e2 (4.17e2, 18)	-2.065780e2 (2.82e2, 18)	-7.533638e1 (1.38e2, 18)	-7.534027e1 (1.07e2, 18)
		5	100	-1.097921e2 (9.00e1, 17)	-1.097917e2 (3.64e1, 17)	-8.240211e1 (3.98e1, 17)	-8.240200e1 (2.94e1, 17)
$D = [0,1]^n$	dense	2	200	-5.606821e2 (1.04e2, 13)	-5.606566e2 (9.08e1, 19)	-3.191656e2 (1.09e2, 13)	-3.191556e2 (1.16e2, 16)
		2	400	-1.516305e3 (6.57e2, 13)	-1.516262e3 (6.57e2, 16)	-8.701231e2 (5.19e2, 15)	-8.700985e2 (5.66e2, 20)
		3	10	-2.085634e1 (1.18e1, 16)	-2.085633e1 (9.80e0, 16)	-9.052818e0 (1.15e1, 11)	-9.052849e0 (9.74e0, 11)
		3	20	-4.213167e1 (2.78e2, 13)	-4.213083e1 (2.52e2, 13)	-2.532441e1 (7.08e1, 15)	-2.532461e1 (8.25e1, 15)
		5	10	-3.604522e1 (3.48e2, 14)	-3.605142e1 (3.05e2, 15)	-1.357776e1 (1.38e2, 13)	-1.357794e1 (1.30e2, 13)
	Arrow ($a = 10$, $b = 2$, $c = 2$)	2	324 ($\ell = 40$)	-2.285440e2 (7.66e1, 18)	-2.285441e2 (5.91e1, 18)	-1.010020e2 (6.34e1, 18)	-1.010060e2 (4.55e1, 17)
		2	644 ($\ell = 80$)	-4.711960e2 (1.63e2, 18)	-4.711968e2 (1.01e2, 18)	-2.131692e2 (1.33e2, 17)	-2.131563e2 (2.65e2, 17)
		3	84 ($\ell = 10$)	-1.367252e2 (4.56e2, 18)	-1.367280e2 (3.42e2, 18)	-4.976779e1 (3.48e2, 20)	-4.976758e1 (2.28e2, 19)
		3	164 ($\ell = 20$)	-2.815741e2 (1.02e3, 19)	-2.815798e2 (9.37e2, 19)	-9.442022e1 (5.38e2, 19)	-9.442065e1 (3.71e2, 19)
		5	20 ($\ell = 2$)	-1.541230e2 (2.52e3, 19)	-1.541231e2 (2.02e3, 19)	-2.023071e1 (1.75e3, 20)	-2.023120e1 (1.56e3, 20)
	Chordal ($r = 0.1$)	2	200	-1.021505e2 (1.50e2, 18)	-1.021542e2 (9.82e1, 18)	-4.670504e1 (8.57e1, 19)	-4.670487e1 (8.45e1, 19)
		2	400	-3.486643e2 (5.09e2, 18)	-3.486785e2 (3.36e2, 18)	-8.688880e1 (6.79e2, 18)	-8.688948e1 (2.79e2, 19)
		3	100	-5.668382e1 (1.25e2, 20)	-5.669066e1 (8.11e1, 18)	-5.304546e1 (1.14e2, 19)	-5.304416e1 (8.15e1, 18)
		3	200	-2.124497e2 (1.38e3, 18)	-2.124955e2 (4.51e2, 17)	-7.678925e1 (2.92e2, 19)	-7.679217e1 (2.07e2, 19)
		5	80	-6.679337e1 (2.09e2, 17)	-6.679132e1 (1.68e2, 17)	-6.470408e1 (1.57e2, 20)	-6.470058e1 (1.21e2, 19)

Table 4.5.: The computation results of SDPNAL+ and our BP for $D = \{0,1\}^n$. The computation time in red color indicates that the solver was more than 3 times faster than the other.

obj. \ constr.			$C = \mathbb{R}^n$		$C = \{\boldsymbol{x} \mid x_i x_j = 0,\ (i,j) \in E\}$	
type	d	n	SDPNAL+ LB (sec,iter)	proposed BP LB (sec,iter)	SDPNAL+ LB (sec,iter)	proposed BP LB (sec,iter)
Dense	2	100	-2.508723e2 (1.13e1,2640)	-2.508823e2 (1.10e1,7)	-6.218346e2 (1.99e1,4344)	-6.218532e2 (1.96e1,10)
	2	200	-7.373652e2 (2.25e1,1785)	-7.373967e2 (1.14e2,15)	-2.351449e3 (4.27e1,3595)	-2.351563e3 (6.11e1,13)
	2	400	-2.106236e3 (1.26e2,2356)	-2.117211e3 (1.86e2,7)	-9.558810e3 (1.52e2,2767)	-9.560612e3 (1.91e2,10)
	2	800	-5.995385e3 (7.92e2,3039)	-5.995703e3 (3.62e3,16)	-3.603980e4 (1.16e3,5570)	-3.604408e4 (1.27e3,10)
	3	5	-1.074577e1 (2.35e0,480)	-1.074604e1 (3.27e0,13)	-3.031196e0 (1.98e0,155)	-3.031297e0 (1.95e0,15)
	3	10	-3.266364e1 (9.35e1,1882)	-3.266508e1 (8.91e1,10)	-1.196724e1 (1.57e1,475)	-1.196746e1 (4.68e1,17)
	3	20	-7.931911e1 (1.13e3,5382)	-7.932204e1 (5.04e2,14)	-2.874508e1 (2.08e2,1940)	-2.874610e1 (3.96e2,14)
	3	40	-1.323866e2 (2.60e4,20000)	-1.343181e2 (6.10e3,17)	-4.856321e1 (1.39e3,3161)	-4.856598e1 (2.60e3,17)
	5	5	-3.312959e0 (1.99e0,476)	-3.312959e0 (1.33e0,13)	-5.730156e-1 (6.29e-1,86)	-5.730639e-1 (8.26e-1,11)
	5	10	-1.506213e1 (3.42e1,643)	-1.506309e1 (1.43e1,9)	-3.031196e0 (3.37e0,155)	-3.031318e0 (1.04e1,17)
	5	20	-1.342569e2 (4.17e4,20000)	-1.340623e2 (2.67e4,19)	-2.097163e1 (2.21e3,1538)	-2.097288e1 (3.81e3,17)
Arrow ($a = 10$, $b = 2$, $c = 2$)	2	164 ($\ell = 20$)	-3.069421e2 (3.16e2,6242)	-3.069512e2 (1.81e1,13)	-1.182815e2 (3.70e2,5852)	-1.182901e2 (2.71e1,9)
	2	324 ($\ell = 40$)	-6.272282e2 (4.90e2,6990)	-6.272577e2 (3.28e1,14)	-2.722193e2 (4.17e2,1930)	-2.722325e2 (4.10e1,15)
	2	644 ($\ell = 80$)	-1.243046e3 (1.02e3,6854)	-1.243105e3 (6.72e1,13)	-5.238687e2 (1.72e3,2434)	-5.238795e2 (1.04e2,11)
	2	1284 ($\ell = 160$)	-2.528641e3 (1.21e3,1928)	-2.528718e3 (1.36e2,13)	-1.024988e3 (3.74e3,5603)	-1.025048e3 (2.45e2,17)
	3	124 ($\ell = 15$)	-2.205903e1 (1.72e1,1402)	-2.205954e1 (2.54e1,17)	-7.456768e0 (3.53e0,159)	-7.457272e0 (1.49e1,17)
	3	244 ($\ell = 30$)	-4.226726e2 (2.08e3,2818)	-4.226879e2 (4.52e2,14)	-1.203221e2 (9.11e2,2316)	-1.203248e2 (3.63e2,18)
	3	484 ($\ell = 60$)	-6.558257e2 (6.34e3,4362)	-6.558529e2 (7.38e2,16)	-1.779116e2 (1.81e3,1564)	-1.779222e2 (6.56e2,18)
	3	964 ($\ell = 120$)	-1.260095e3 (5.58e4,20000)	-1.260078e3 (1.54e3,15)	-3.390619e2 (1.14e4,4183)	-3.390689e2 (1.35e3,18)
	5	28 ($\ell = 3$)	-9.023598e1 (1.32e3,961)	-9.023791e1 (1.31e3,15)	-1.071734e1 (2.19e2,1239)	-1.071754e1 (5.40e2,18)
	5	52 ($\ell = 6$)	-1.179803e2 (7.79e3,20000)	-1.179862e2 (1.06e3,11)	-1.879566e1 (2.06e2,317)	-1.879591e1 (5.21e2,14)
	5	100 ($\ell = 12$)	-2.714688e2 (1.87e4,20000)	-2.706813e2 (4.07e3,18)	-3.955196e1 (1.02e3,777)	-3.955411e1 (1.45e3,19)
Chordal ($r = 0.1$)	2	200	-1.088644e2 (1.22e3,1383)	-1.088738e2 (8.45e1,17)	-4.758922e1 (5.63e2,1032)	-4.758970e1 (1.03e2,19)
	2	400	-3.413272e2 (4.53e3,2334)	-3.413378e2 (3.88e2,20)	-8.438539e1 (7.27e3,6479)	-8.440153e1 (2.27e2,18)
	2	800	-1.258048e3 (1.97e4,10556)	-1.258274e3 (1.10e3,18)	-2.025567e2 (2.24e4,2152)	-2.024907e2 (7.06e2,19)
	2	1600	–	-3.898720e3 (6.20e3,19)	–	
	3	50	-2.410153e1 (5.24e0,156)	-2.410148e1 (1.48e1,17)	-2.410153e1 (4.76e0,156)	-2.410199e1 (1.42e1,17)
	3	100	-5.494730e1 (5.68e1,794)	-5.494936e1 (3.18e1,16)	-4.955103e1 (4.06e1,634)	-4.955256e1 (3.38e1,17)
	3	200	-1.972756e2 (1.64e3,1115)	-1.972783e2 (4.69e2,17)	-5.848304e1 (2.81e2,634)	-5.848489e1 (1.14e2,18)
	3	300	-4.512758e2 (1.70e4,1918)	-4.512869e2 (1.33e3,18)	-9.176276e1 (2.63e3,794)	-9.176771e1 (4.79e2,19)
	5	50	-2.893852e1 (1.20e1,475)	-2.893932e1 (1.93e1,16)	-2.753925e1 (7.74e0,312)	-2.753937e1 (1.47e1,16)
	5	100	-1.091866e2 (7.06e1,797)	-1.091868e2 (3.84e1,17)	-9.448516e1 (5.04e1,796)	-9.448693e1 (3.10e1,18)
	5	200	-6.002734e2 (2.93e4,4162)	-6.002825e2 (2.18e3,18)	-1.312750e2 (1.07e3,634)	-1.312789e2 (5.08e2,18)

Table 4.6.: The computation results of SDPNAL+ and our BP for $D = [0, 1]^n$. The computation time in red color indicates that the solver was more than 3 times faster than the other.

obj. \ constr.			$C = \mathbb{R}^n$		$C = \{\boldsymbol{x} \mid x_i x_j = 0,\ (i,j) \in E\}$	
type	d	n	SDPNAL+ LB (sec,iter)	proposed BP LB (sec,iter)	SDPNAL+ LB (sec,iter)	proposed BP LB (sec,iter)
Dense	2	100	-2.13965e2 (1.11e1,2499)	-2.13974e2 (8.90e0,14)	-1.20791e2 (4.65e0,551)	-1.20793e2 (1.36e1,14)
	2	200	-6.95276e2 (4.22e1,3593)	-6.95297e2 (5.90e1,14)	-3.85332e2 (1.95e1,1250)	-3.85346e2 (6.68e1,18)
	2	400	-2.14803e3 (5.02e2,11737)	-2.14831e3 (3.08e2,13)	-1.19813e3 (1.32e2,2438)	-1.42695e3 (5.31e1,6)
	2	800	-5.74551e3 (2.87e3,15801)	-5.74591e3 (2.19e3,14)	-3.45074e3 (1.13e3,5208,-2)	-3.71652e3 (7.40e2,8)
	3	5	-1.65247e1 (8.18e1,18606)	-1.65281e1 (5.27e0,8)	-4.76195e0 (1.28e1,2290)	-4.76197e0 (1.03e1,6)
	3	10	-4.01117e1 (9.71e2,20000)	-4.01738e1 (1.62e2,9)	-1.62663e1 (3.45e2,20000)	-1.64553e1 (9.58e1,18)
	3	20	-8.28647e1 (2.81e3,20000)	-8.31069e1 (1.87e3,11)	-2.91937e1 (1.58e4,20000)	-2.92752e1 (1.47e3,13)
	3	40	-1.78469e2 (1.79e4,20000)	-1.80586e2 (7.29e3,14)	-6.85313e1 (1.24e4,20000)	-6.84317e1 (5.68e3,20)
	5	5	-4.29514e0 (7.43e1,20000)	-4.29664e0 (1.20e1,9)	-8.47532e-1 (2.99e0,395)	-8.47556e-1 (3.29e0,15)
	5	10	-2.39170e1 (7.41e2,20000)	-2.42987e1 (9.00e2,12)	-8.44910e0 (5.24e2,20000)	-8.45719e0 (9.88e1,18)
	5	20	– ($\geq$ 1.00e5,–)	-1.71141e2 (5.17e4,8)	– ($\geq$ 1.00e5,–)	-5.500295e1 (2.54e4,8)
Arrow ($a = 10$, $b = 2$, $c = 2$)	2	164 ($\ell = 20$)	-3.06945e2 (5.70e1,1828)	-3.06954e2 (1.66e1,13)	-1.19088e2 (3.27e2,9364)	-1.19094e2 (1.75e1,11)
	2	324 ($\ell = 40$)	-6.27246e2 (1.13e2,1492)	-6.27272e2 (3.00e1,14)	-2.73390e2 (3.51e2,4149)	-2.73400e2 (3.75e1,14)
	2	644 ($\ell = 80$)	-1.24338e3 (4.12e2,1991)	-1.24345e3 (5.46e1,13)	-5.26410e2 (2.00e3,7905)	-5.26423e2 (6.03e1,11)
	2	1284 ($\ell = 160$)	-2.52915e3 (1.06e3,2321)	-2.52920e3 (1.84e2,16)	-1.02894e3 (4.23e3,16949)	-1.02896e3 (1.75e2,16)
	3	44 ($\ell = 5$)	-2.55494e1 (3.10e2,19471)	-2.55560e1 (1.86e2,14)	-9.95897e0 (2.56e2,19353)	-9.96340e0 (3.56e1,14)
	3	84 ($\ell = 10$)	-6.566684e1 (4.46e3,20000)	-6.573373e1 (1.68e3,22)	-2.053815e1 (1.63e3,20000)	-2.050310e1 (4.14e2,23)
	3	164 ($\ell = 20$)	-1.426242e2 (2.20e4,20000)	-1.410596e2 (8.44e3,24)	-3.689789e1 (2.44e4,20000)	-3.658250e1 (2.23e3,22)
	3	324 ($\ell = 40$)	-5.36907e2 (1.09e4,20000)	-5.37179e2 (1.76e3,13)	-1.77313e2 (1.40e4,20000)	-1.77334e2 (2.19e3,19)
	5	12 ($\ell = 1$)	-6.493273e1 (2.22e3,20000)	-6.241569e1 (2.04e3,17)	-1.815851e1 (1.26e3,20000)	-1.817109e1 (1.20e3,17)
	5	20 ($\ell = 2$)	-1.18201e2 (8.30e3,20000)	-1.19228e2 (4.87e3,11)	-1.96787e1 (3.17e3,20000)	-1.96819e1 (1.21e3,17)
	5	36 ($\ell = 4$)	-2.02467e2 (4.10e4,20000)	-2.03008e2 (7.29e3,16)	-3.98620e1 (4.95e3,20000)	-3.99237e1 (2.89e3,19)
Chordal ($r = 0.1$)	2	100	-3.823840e1 (6.00e1,463)	-3.823985e1 (3.73e1,16)	-3.487212e1 (5.85e1,485)	-3.487320e1 (5.03e1,19)
	2	200	-1.095193e2 (1.56e3,1251)	-1.095211e2 (1.01e2,17)	-4.820353e1 (9.07e2,1535)	-4.820233e1 (1.28e2,19)
	2	400	-3.414408e2 (7.84e3,3052)	-3.414572e2 (5.30e2,18)	-8.483938e1 (1.86e4,12799)	-8.484490e1 (5.22e2,19)
	2	800	-1.258153e3 (3.15e4,5362)	-1.258374e3 (9.30e2,18)	-2.027105e2 (1.97e4,3045)	-2.026716e2 (1.06e3,19)
	3	50	-2.506908e1 (3.08e2,10762)	-2.507087e1 (3.64e1,17)	-2.506909e1 (3.05e2,10886)	-2.507043e1 (3.34e1,18)
	3	100	-5.809462e1 (1.33e3,20000)	-5.808397e1 (7.73e1,17)	-5.167946e1 (1.30e3,20000)	-5.166894e1 (9.84e1,17)
	3	200	-2.042882e2 (1.21e4,20000)	-2.043830e2 (5.99e2,17)	-6.040685e1 (1.23e4,11308)	-6.040493e1 (3.75e2,18)
	3	400	–	-9.480865e2 (3.55e4,20)	–	-1.100411e2 (2.82e4,20)
	5	40	-2.647768e1 (3.21e2,20000)	-2.648323e1 (3.06e1,18)	-2.647755e1 (3.88e2,19854)	-2.648344e1 (3.58e1,19)
	5	80	-7.909182e1 (1.48e3,20000)	-7.910323e1 (1.14e2,18)	-7.909333e1 (1.91e3,20000)	-7.910488e1 (1.32e2,18)
	5	160	-2.794243e2 (2.54e4,20000)	-2.795148e2 (2.41e3,19)	-1.141319e2 (3.31e4,20000)	-1.140274e2 (8.15e2,19)

Table 4.4 shows the results of our proposed BP (Algorithm 4.6) and the original BP (Algorithm 4.2). Compared to the original BP method, our BP could compute the valid lower bounds within 0.1% difference in less computation time. Our proposed techniques certainly improved the performance of the BP method. Next, we compared the performances of SDPNAL+ and our BP method. The results for $D = \{0,1\}^n$ and $[0,1]^n$ are shown in Tables 4.5 and 4.6, respectively. When $D = [0,1]^n$, there were some instances such that the lower bounds obtained by SDPNAL+ were 10 percent better than BP. On the other hand, when $D = \{0,1\}^n$, the lower bounds obtained by BP were comparable to and sometimes better than SDPNAL+. In both cases of D, we can see that the computation time of BP was much faster than SDPNAL+, especially for large scale and sparse polynomial optimization problems. We mention that SDPNAL+ is based on semismooth Newton method which uses second-order derivative of functions, and hence it is expected produce an accurate solution in a moderate number of iterations. Although the BP method uses only first-order derivative information, it could compute lower bounds comparable to SDPNAL+ in less computation time. We therefore conclude that the BP method is a very efficient method for solving the conic relaxation of polynomial optimization problems.

5. Equivalences and Differences in Conic Relaxations of Combinatorial Quadratic Optimization Problems

The contents in this chapter is submitted to Journal of Global Optimization and is in peer review process. By the copyright policy of the journal, it is restricted to deposit the contents in any web repository for a year from the official publication date. This chapter is supposed to be released within five years from March 22, 2017. We refer the readers to the earlier version [Ito et al., 2017a] of this chapter until the period passes.

6. Conclusion

In this thesis, based on the APG method, we developed new formulations and optimization algorithms for binary classification and the DNN relaxation of polynomial optimization problems.

In Chapter 3, we presented a unified classification model and provided a general algorithm for the model. We designed a fast accelerated proximal gradient (FAPG) method based on the original APG method in [Beck and Teboulle, 2009] to the model by devising efficient projection computations and effective heuristic acceleration strategies. Our unified algorithm makes it easy to compare various models, because we can use the same algorithmic framework for the models by only changing the computation of projections. Thus, it provides a practical and useful tool for practitioners who are looking for the best model for a given dataset. Numerical experiments demonstrate the efficiency of our algorithm for large datasets. Indeed, our method often run faster than LIBLINEAR [Fan et al., 2008] especially for large-scale datasets such that $n > 2000$.

In Chapter 4, we proposed a doubly nonnegative (DNN) relaxation method for polynomial optimization problems (POPs) subject to polynomial equalities, binary and box constraints, and complementarity conditions. We first proposed a framework of DNN relaxation problems of the POPs using the notion of equivalent relation and partial order. Then we desined sub-partial orders in order to derive simple DNN relaxation problems. With this simplification, the DNN relaxation problems fall into a framework of COPs that can be solved by the BP method [Kim et al., 2016a; Arima et al., 2017] efficiently. Moreover, we developed a method to compute a good upper bound of trace of matrix variable, which is an important parameter for stabilizing the BP method, based on the greedy approximation algorithm for the minimizing submodular cost submodular cover problem. In the BP method, feasibility of a given point is determined by the APG method. We improved the APG method by employing the adaptive restarting strategy and new stopping criteion. Numerical experiments showed that our DNN relaxation problem provided better lower bounds than the one of [Kim et al., 2016b], and the BP method with our restarting APG outperformed the original BP method and SDPNAL+.

In Chapter 5, we showed that many conic relaxations proposed for a combinatorial optimization problem are equivalent in the quality of the optimal values they provide. However, they differ in the size of the matrix variable, the number of linear equality and inequality constraints, the existence of an interior feasible solution, and the primal/dual degeneracy, which are crucial issues for the performance of a numerical method. We have proved the equivalences and differences in the SDP, DNN, and CPP relaxations of combinatorial optimization problems by examining several ways of representing the combinatorial condition. This approach has revealed the connections among the existing relaxations, and also provided new conic relaxations for the QAP. We have tested

the theoretical results with QAP instances and obtained consistent numerical results using SDPT3, SDPNAL+ and BP. For the combinatorial optimization problems that are not dealt with in this thesis, for instance, the quadratic multiple knapsack problem, maximum stable set problem and graph partitioning problem, the same approach presented in this paper can be applied to show the equivalence and difference in their conic relaxations.

We conclude this thesis by remarking a future direction of the research. Recently, large scale machine learning has motivated researchers to develop stochastic gradient descent (SGD) methods. SGD approximates the gradient $\nabla\mathcal{L}(\cdot)$ of a loss function $\mathcal{L}(\cdot)$ by a computationally inexpensive random variable whose expectation coincides with $\nabla\mathcal{L}(\cdot)$. More precisely, it assumes that the loss function is defined as the sum of loss of each sample, i.e., $\mathcal{L}(\widetilde{\boldsymbol{X}}^\top\boldsymbol{w} - \boldsymbol{y}b) = \sum_{i=1}^m \ell(y_i(\boldsymbol{w}^\top\boldsymbol{x}_i - b))$, and approximates the *full* gradient $\nabla\mathcal{L}(\widetilde{\boldsymbol{X}}^\top\boldsymbol{w} - \boldsymbol{y}b)$ by the gradient $\nabla\ell(y_i(\boldsymbol{w}^\top\boldsymbol{x}_i - b))$ of loss of a randomly chosen sample. PG and APG methods can be adapted in SGD, e.g., as [Nitanda, 2014; Defazio et al., 2014] and references there in. In particular, the stochastic dual coordinate ascent (SDCA) method [Shalev-Shwartz and Zhang, 2013, 2016] is a stochastic APG applied to the dual formulation, which is closely related to our unified classification model. While its framework would be useful to develop a stochastic variant of our FAPG presented in this paper, it cannot be directly applied to the unified classification model because ν-SVM, MM-MPM, and MM-FDA do not satisfy the assumption on $\mathcal{L}(\cdot)$: the loss function is defined as the sum of loss of each sample. The objective function of the DNN relaxation problem does not have such structure, either. It is an important future work to relax the assumption for stochastic methods and extend our APG methods to stochastic ones in order to solve even larger scale problems.

A. Appendix

A.1. A Refined Bisection Algorithm

The computational cost of $h(\hat{\theta})$ in Algorithm 3.1 can be reduced by dividing the indices set M more finely as in [Kiwiel, 2008]. Here we divide M into the following four disjoint sets for given θ^l and θ^u satisfying $\theta^l < \theta^u$:

$$\begin{aligned}
U &:= \{i \in M \mid \bar{\alpha}_i - \theta^u \geq u && \text{(i.e., } \alpha_i(\theta) = u, && \forall \theta \in [\theta^l, \theta^u])\} \\
L &:= \{i \in M \mid \bar{\alpha}_i - \theta^l \leq l && \text{(i.e., } \alpha_i(\theta) = l, && \forall \theta \in [\theta^l, \theta^u])\} \\
C &:= \{i \in M \mid \bar{\alpha}_i - \theta^l < u,\ \bar{\alpha}_i - \theta^u > l && \text{(i.e., } \alpha_i(\theta) = \bar{\alpha}_i - \theta, && \forall \theta \in [\theta^l, \theta^u])\} \\
I &:= M \backslash (U \cup C \cup L) && ([\theta^l, \theta^u] \text{ contains a breakpoint of } \alpha_i(\theta)) \ .
\end{aligned}$$

If $I = \phi$, then solving

$$\theta = \Big(|U|u + |L|l + \sum_{i \in C} \bar{\alpha}_i - r\Big)/|C|$$

obtains an exact solution. If $I \neq \phi$, then we also divide I into the following five disjoint sets for given $\hat{\theta} \in (\theta^l, \theta^u)$:

$$\begin{aligned}
I^U &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \geq u, && \text{(i.e., } \alpha_i(\theta) = u && \forall \theta \in [\theta^l, \hat{\theta}])\} \\
I^L &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} \leq l, && \text{(i.e., } \alpha_i(\theta) = l && \forall \theta \in [\hat{\theta}, \theta^u])\} \\
I^{C_u} &= \{i \in I \mid \bar{\alpha}_i - \theta^l < u,\ \bar{\alpha}_i - \hat{\theta} > l, && \text{(i.e., } \alpha_i(\theta) = \bar{\alpha}_i - \theta && \forall \theta \in [\theta^l, \hat{\theta}])\} \\
I^{C_l} &= \{i \in I \mid \bar{\alpha}_i - \hat{\theta} < u,\ \bar{\alpha}_i - \theta^u > l, && \text{(i.e., } \alpha_i(\theta) = \bar{\alpha}_i - \theta && \forall \theta \in [\hat{\theta}, \theta^u])\} \\
I^I &= I \backslash (I^U \cup I^{C_u} \cup I^{C_l} \cup I^L)
\end{aligned}$$

Then we have

$$\begin{aligned}
h(\hat{\theta}) = \sum_{i \in M} \alpha_i(\hat{\theta}) = \underbrace{|U|u}_{s_u} + \underbrace{|I^U|u}_{\Delta s_u} + \underbrace{|L|l}_{s_l} + \underbrace{|I^L|l}_{\Delta s_l} \\
+ \underbrace{\sum_{i \in I^C} \alpha_i}_{s_c} + \underbrace{\sum_{i \in I^{C_u}} \alpha_i}_{\Delta s_{c_u}} + \underbrace{\sum_{i \in I^{C_l}} \alpha_i}_{\Delta s_{c_l}} + \sum_{i \in I^I} \alpha_i - |C \cup I^{C_u} \cup I^{C_l} \cup I^I|\hat{\theta}.
\end{aligned}$$

By leveraging on the structure of h, the refined bisection method for (3.13) can be described as Algorithm 1.1.

Algorithm 1.1 Refined Bisection Algorithm for (3.13)

INPUT: $\bar{\boldsymbol{\alpha}},\ r,\ \boldsymbol{l},\ \boldsymbol{u},\ \epsilon' > 0$ OUTPUT: $\boldsymbol{\alpha}$

INITIALIZE: $I \leftarrow M,\ s_u \leftarrow s_l \leftarrow s_c \leftarrow 0,\ \theta^u \leftarrow \bar{\alpha}_{\max} - \frac{r}{m},\ \theta^l \leftarrow \bar{\alpha}_{\min} - \frac{r}{m}$ # Step 1

while $|\theta^u - \theta^l| > \epsilon'$ **do**

$\hat{\theta} \leftarrow \frac{\theta^u + \theta^l}{2}$ # Step 2

$\hat{u} \leftarrow u + \hat{\theta}, \quad \hat{l} \leftarrow l + \hat{\theta}$

$I^U \leftarrow \{i \in I \mid \bar{\alpha}_i \geq \hat{u}\}, \quad I^L \leftarrow \{i \in I \mid \bar{\alpha}_i \leq \hat{l}\}$ # Step 3

$u^u \leftarrow u + \theta^u, \quad l^l \leftarrow l + \theta^l$

$I^{C_u} \leftarrow \{i \in I \mid \bar{\alpha}_i > l^l,\ \bar{\alpha}_i < \hat{u}\}, \quad I^{C_l} \leftarrow \{i \in I \mid \bar{\alpha}_i < u^u,\ \bar{\alpha}_i > \hat{l}\}$

$I^I \leftarrow I \backslash (I^U \cup I^{C_u} \cup I^{C_l} \cup I^L)$

$\Delta s_u \leftarrow |I^U| u, \quad \Delta s_l \leftarrow |I^L| l$

$\Delta s_{c_u} \leftarrow \sum_{i \in I^{C_u}} \alpha_i, \quad \Delta s_{c_l} \leftarrow \sum_{i \in I^{C_l}} \alpha_i$

$val \leftarrow s_u + \Delta s_u + s_l + \Delta s_l + s_c + \Delta s_{c_u} + \Delta s_{c_l} + \sum_{i \in I^I} \alpha_i - |C \cup I^{C_u} \cup I^{C_l} \cup I^I| \hat{\theta}$

if $val < r$ **then** # Step 4

$\theta^u \leftarrow \hat{\theta}, \quad I \leftarrow I^I \cup I^{C_l} \cup I^L$

$s_u \leftarrow s_u + \Delta s_u, \quad s_c \leftarrow s_c + \Delta s_{c_u}$

else if $val > r$ **then**

$\theta^l \leftarrow \hat{\theta}, \quad I \leftarrow I^I \cup I^{C_u} \cup I^U$

$s_l \leftarrow s_l + \Delta s_l, \quad s_c \leftarrow s_c + \Delta s_{c_l}$

else

break

end if

if $I == \phi$ **then**

$\hat{\theta} \leftarrow (s_u + s_l + s_c - r)/|C|$

break

end if

end while

$\alpha_i \leftarrow \alpha_i(\hat{\theta}),\ \forall i \in M$ # Step 5

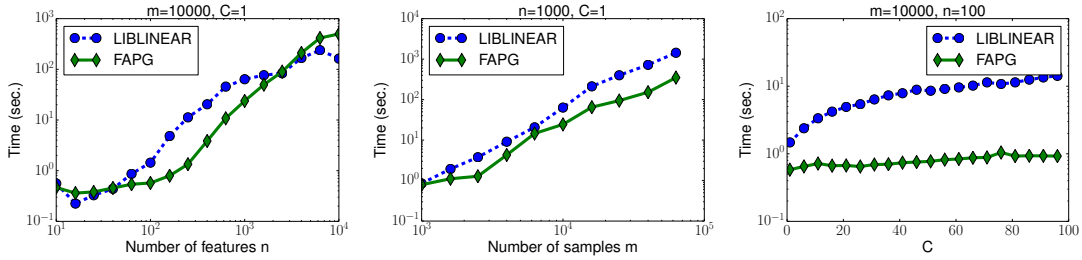


Figure A.1.: Computation Time for ℓ_1-regularized Logistic Regression (A.1).

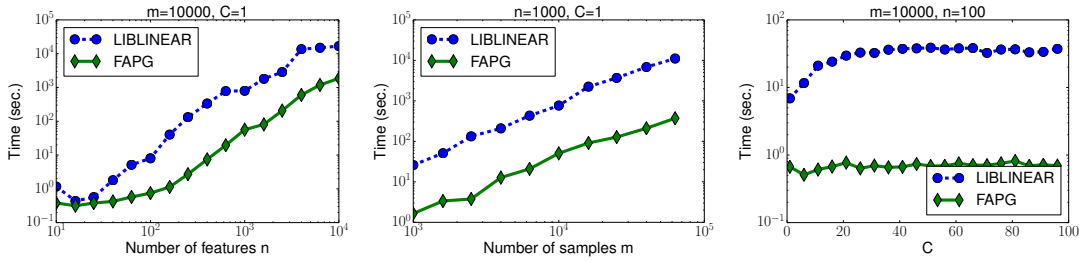


Figure A.2.: Computation Time for ℓ_1-regularized ℓ_2-loss SVM (A.2).

A.2. Applications of the FAPG to ℓ_1-Regularized Models

The ℓ_1-regularized logistic regression:

$$\min_{\boldsymbol{w}} C \sum_{i=1}^{m} \log(1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)) + \|\boldsymbol{w}\|_1 \tag{A.1}$$

and the ℓ_1-regularized ℓ_2-loss SVM:

$$\min_{\boldsymbol{w}} C \sum_{i=1}^{m} \big(\max\{0, 1 - y_i \boldsymbol{w}^\top \boldsymbol{x}_i\} \big)^2 + \|\boldsymbol{w}\|_1 \tag{A.2}$$

have the form of (2.1), where $C > 0$ is a hyperparameter. Letting $g(\cdot) = \|\cdot\|_1$, we have $\big(\mathrm{prox}_{g,L}(\boldsymbol{w})\big)_i = \mathrm{sign}(w_i) \max\{0, |w_i| - L\}$ $(i = 1, 2, \ldots, m)$ which is known as the soft-thresholding operator.

We applied the practical FAPG (Algorithm 3.3) and LIBLINEAR [Fan et al., 2008] to (A.1) and (A.2) using the artificial datasets which is generated as described in Section 3.4. We set the initial point $\boldsymbol{w}^0$ to the origin $\mathbf{0}$ and the tolerance ϵ to 10^{-6}. For all datasets, FAPG found solutions with better objective value than LIBLINEAR does.

Figures A.1 and A.2 show the computation time of FAPG and LIBLINEAR. FAPG was highly competitive to and numerically more stable than LIBLINEAR with respect to the changes of the hyperparameter C.

Bibliography

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

N. Arima, S. Kim, and M. Kojima. Simplified copositive and lagrangian relaxations for linearly constrained quadratic optimization problems in continuous and binary variables. *Pacific Journal of Optimization*, 10:437–451, 2014a.

N. Arima, S. Kim, M. Kojima, and K. C. Toh. Lagrangian-conic relaxations, part I: A unified framework and its applications to quadratic optimization problems. Research report B-475, Tokyo Institute of Technology, Department of Mathematical and Computing Sciences, Oh-Okayama, Meguro-ku, Tokyo 152-8552, January 2014b.

N. Arima, S. Kim, M. Kojima, and K. Toh. A robust Lagrangian-DNN method for a class of quadratic optimization problems. *Computational Optimization and Applications*, 66(3):453–479, 2017.

M. Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *The Annals of Mathematical Statistics*, 26(4):641–647, December 1955.

K. Bache and M. Lichman. UCI machine learning repository, 2013. `http://archive.ics.uci.edu/ml`, Accessed September 18, 2017.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2:183–202, 2009.

D. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Optimization and Computation Series. Athena Scientific, 2003.

M. J. Best and N. Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1):425–439, May 1990.

C. Bhattacharyya. Second order cone programming formulations for feature selection. *Journal of Machine Learning Research*, 5:1417–1433, 2004.

J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In L. J. W. H. George A., Gilbert J. R., editor, *Graph Theory and Sparse Matrix Computation*. Springer-Verlag, New York, 1993.

I. M. Bomze and E. de Klerk. Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *Journal of Global Optimization*, 24: 163–185, 2002.

I. M. Bomze, M. Dür, E. de Klerk, C. Roos, A. Quist, and T. Terlaky. On copositive programming and standard quadratic optimization problems. *Journal of Global Optimization*, 18:301–320, 2000.

R. E. Bruck. An iterative solution of a variational inequality for certain monotone operators in hilbert space. *Bulletin of the American Mathematical Society*, 81(5): 890–892, September 1975.

S. Burer. On the copositive representation of binary and continuous non-convex quadratic programs. *Mathematical Programming*, 120:479–495, 2009.

R. E. Burkard, S. E. Karisch, and F. Rendl. QAPLIB – a quadratic assignment problem library. *Journal of Global Optimization*, 10:391–403, 1997. The online version is available at `http://anjos.mgi.polymtl.ca/qaplib/`.

M. A. Cauchy. Methode générale pour la résolution des systémes d'equations simultanées (in French). *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 25:536–538, 1847. English translation by R. J. Pulskamp is available at Y. Yu's webpage `https://cs.uwaterloo.ca/~y328yu/classics/cauchy-en.pdf`, Accessed October 10, 2017.

C.-C. Chang and C.-J. Lin. LIBSVM : A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

P. L. Combettes and N. N. Reyes. Moreau's decomposition in banach spaces. *Mathematical Programming*, 139(1):103–114, June 2013.

R. M. Corless, G. H. Gonnet, D. E. G. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, 1996.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

E. de Klerk and D. V. Pasechnik. Approximation of the stability number of a graph via copositive programming. *SIAM Journal on Optimization*, 12(4):875–892, 2002.

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 27*, pages 1646–1654. Curran Associates, Inc., 2014.

R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the L1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279, Helsinki, Finland, 2008.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. Software available at `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

M. Fukushima and H. Mine. A generalized proximal point algorithm for certain non-convex minimization problems. *International Journal of Systems Science*, 12(8):989–1000, 1981.

D. R. Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. *Proceedings of the American Mathematical Society*, 7(4):701–702, 1956.

A. A. Goldstein. Convex programming in hilbert space. *Bulletin of the American Mathematical Society*, 70(5):709–710, 1964.

T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286(5439):531–537, 1999.

J. Gotoh and A. Takeda. A linear classification model based on conditional geometric score. *Pacific Journal of Optimization*, 1(2):277–296, 2005.

I. Guyon, S. Gunn, A. B. Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17*, pages 545–552, 2005.

K. Helgason, J. Kennington, and H. Lall. A polynomially bounded algorithm for a singly constrained quadratic program. *Mathematical Programming*, 18:338–343, 1980.

T.-K. Ho and E. M. Kleinberg. Building projectable classifiers of arbitrary complexity. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 2, pages 880–885. IEEE, 1996.

J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th International Conference on Machine learning*, pages 408–415. ACM, 2008.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, 2003.

N. Ito, S. Kim, M. Kojima, A. Takeda, and K.-C. Toh. Equivalences and differences in conic relaxations of combinatorial quadratic optimization problems, 2017a. `http://www.optimization-online.org/DB_HTML/2017/07/6134.html`.

N. Ito, A. Takeda, and K.-C. Toh. A unified formulation and fast accelerated proximal gradient method for classification. *Journal of Machine Learning Research*, 18(16): 1–49, 2017b.

S. Iwata and K. Nagano. Submodular function minimization under covering constraints. In *50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 671–680, 2009.

K. Jiang, D.-F. Sun, and K.-C. Toh. An inexact accelerated proximal gradient method for large scale linearly constrained convex SDP. *SIAM Journal on Optimization*, 22: 1042–1064, 2012.

N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, Dec 1984.

S. S. Keerthi and D. DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

S. Kim and M. Kojima. Binary quadratic optimization problems that are difficult to solve by conic relaxations. *Discrete Optimization*, 24:170–183, 2017.

S. Kim, M. Kojima, and K. C. Toh. A Lagrangian-DNN relaxation: a fast method for computing tight lower bounds for a class of quadratic optimization problems. *Mathematical Programming*, 156:161–187, 2016a.

S. Kim, M. Kojima, and K.-C. Toh. Doubly nonnegative relaxations for quadratic and polynomial optimization problems with binary and box constraints. Technical Report B-483, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro-ku, Tokyo 152-8552, July 2016b.

M. Kitamura, A. Takeda, and S. Iwata. Exact SVM training by Wolfe's minimum norm point algorithm. In *2014 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, September 2014.

K. C. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112:473–491, 2008.

M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 29–47. Springer New York, New York, NY, 1989.

J. B. Lasserre. Global optimization with polynomials and the problems of moments. *SIAM Journal on Optimization*, 11:796–817, 2001a.

J. B. Lasserre. *An explicit exact SDP relaxation for nonlinear 0-1 programs*, pages 293–303. Springer Berlin Heidelberg, June 2001b.

E. S. Levitin and B. T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, 6(5):1–50, 1966.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Q. Lin and L. Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. *Computational Optimization and Applications*, 60(3):633–674, April 2015.

R. Luss, S. Rosset, and M. Shahar. Decomposing isotonic regression for efficiently solving large problems. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1513–1521. Curran Associates, Inc., 2010.

J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 681–688. ACM, 2009.

J. S. Marron, M. J. Todd, and J. Ahn. Distance-weighted discrimination. *Journal of the American Statistical Association*, 102(480):1267–1271, 2007.

A. McCallum. SRAA: simulated/real/aviation/auto UseNet data. `http://people.cs.umass.edu/~mccallum/data.html`, Accessed September 18, 2017.

R. D. C. Monteiro, C. Ortiz, and B. F. Svaiter. An adaptive accelerated first-order method for convex optimization. *Computational Optimization and Applications*, 64: 31–73, 2016.

J. J. Moreau. Décomposition orthogonale d'un espace hilbertien selon deux cones mutuellement polaires. *Comptes Rendus de l'Académie des Sciences*, 255:238–240, 1962.

K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and non-linear programming. *Mathematical Programming*, 39:117–129, 1987.

J. S. Nath and C. Bhattacharyya. Maximum margin classifiers with specified false positive and false negative error rates. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 35–46. SIAM, 2007.

A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.

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.

Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Y. E. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 152:127–152, 2005.

Y. E. Nesterov and A. Nemiroskii. *Interior Point Methods for Convex Programming*. SIAM, Philadelphia, USA, 1994.

A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1574–1582. Curran Associates, Inc., 2014.

B. O'Donoghue and E. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15:715–732, 2015.

P. M. Pardalos and G. Xue. Algorithms for a class of isotonic regression problems. *Algorithmica*, 23(3):211–222, March 1999.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, May 2003.

G. B. Passty. Ergodic convergence to a zero of the sum of monotone operators in hilbert space. *Journal of Mathematical Analysis and Applications*, 72(2):383–390, 1979.

J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, January 1998.

J. Povh and F. Rendl. A copositive programming approach to graph partitioning. *SIAM Journal on Optimization*, 18:223–241, 2007.

J. Povh and F. Rendl. Copositive and semidefinite relaxations of the quadratic assignment problem. *Discrete Optimization*, 6:231–241, 2009.

D. Prokhorov. IJCNN 2001 neural network competition. *Slide presentation in IJCNN'01*, 2001. , Accessed September 18, 2017.

J. Rhys. A selection problem of shared fixed costs and network flows. *Management Science*, 17(3):200–207, 1970.

R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2(3):21–41, 2000.

S. Sakaue, A. Takeda, S. Kim, and N. Ito. Exact semidefinite programming relaxations with truncated moment matrix for binary polynomial optimization problems. *SIAM Journal on Optimization*, 27(1):565–582, 2017.

K. Scheinberg, D. Goldfarb, and X. Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundations of Computational Mathematics*, 14(3): 389–417, Jun 2014.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, May 2000.

S. Shalev-Shwartz and T. Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14:567–599, 2013.

S. Shalev-Shwartz and T. Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1):105–145, 2016.

N. Sloane. Challenge problems: Independent sets in graphs. `http://neilsloane.com/doc/graphs.html`, Accessed November 30, 2017.

S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge, 2008. Accessed February 6, 2016.

Q. F. Stout. Fastest isotonic regression algorithms. `http://web.eecs.umich.edu/~qstout/IsoRegAlg_140812.pdf`, 2014. Accessed August 24, 2017.

J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1–4):625–653, 1999. Software available af `http://sedumi.ie.lehigh.edu/`.

W. Su, S. Boyd, and E. J. Candès. A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, Mar 2014.

A. Takeda, H. Mitsugi, and T. Kanamori. A unified classification model based on robust optimization. *Neural computation*, 25(3):759–804, 2013.

K. Tanabe. Centered Newton method for linear programming and quadratic programming. In *the 8th Mathematical Programming Symposium*, pages 131–152, Japan, 1987.

R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical Programming*, 95:189–217, 2003.

A. V. Uzilov, J. M. Keegan, and D. H. Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):173, 2006.

H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of squares and semidefinite programming relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17:218–242, 2006.

P. Wan, D.-Z. Du, P. Pardalos, and W. Wu. Greedy approximations for minimum submodular cover with submodular cost. *Computational Optimization and Application*, 45(2):463–474, 2010.

M. West, C. Blanchette, H. Dressman, E. Huang, S. Ishida, R. Spang, H. Zuzan, J. A. Olson, J. R. Marks, and J. R. Nevins. Predicting the clinical status of human breast cancer by using gene expression profiles. *Proceedings of the National Academy of Sciences*, 98(20):11462–11467, 2001.

L. Q. Yang, D. F. Sun, and K. C. Toh. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Mathematical Programming Computation*, 7(3):331–366, 2015.

H.-F. Yu, F.-L. Huang, and C.-J. Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1):41–75, 2011.

T. Zhou, D. Tao, and X. Wu. NESVM: A fast gradient method for support vector machines. In *2010 IEEE 10th International Conference on Data Mining*, pages 679–688, December 2010.