Solving Stackelberg Prediction Game with Least Squares Loss via Spherically Constrained Least Squares Reformulation

Jiali Wang [1] **Wen Huang** [2] **Rujun Jiang** [1] **Xudong Li** [1] **Alex L. Wang** [3]

Abstract

The Stackelberg prediction game (SPG) is popular in characterizing strategic interactions between a learner and an attacker. As an important special case, the SPG with least squares loss (SPG-LS) has recently received much research attention. Although initially formulated as a difficult bi-level optimization problem, SPG-LS admits tractable reformulations which can be polynomially globally solved by semidefinite programming or second order cone programming. However, all the available approaches are not well-suited for handling large-scale datasets, especially those with huge numbers of features. In this paper, we explore an alternative reformulation of the SPG-LS. By a novel nonlinear change of variables, we rewrite the SPG-LS as a spherically constrained least squares (SCLS) problem. Theoretically, we show that an ϵ optimal solution to the SCLS (and the SPG-LS) can be achieved in $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$ floating-point operations, where N is the number of nonzero entries in the data matrix. Practically, we apply two well-known methods for solving this new reformulation, i.e., the Krylov subspace method and the Riemannian trust region method. Both algorithms are factorization free so that they are suitable for solving large scale problems. Numerical results on both synthetic and real-world datasets indicate that the SPG-LS, equipped with the SCLS reformulation, can be solved orders of magnitude faster than the state of the art.

1. Introduction

The big data era has led to an explosion in the availability of data from which to make decisions. It is thus indispensable to use machine learning techniques to gain deep insights from massive data. In practice, many classic data analytic approaches start by splitting available data into the training and test sets. Then, learning algorithms are fed with the training set and are expected to produce results which generalize well to the test set. However, this paradigm only works under the key implicit assumption that the available data in both training and test sets are independently and identically distributed, which, unfortunately, is not always the truth in practice. For example, in the context of email spam filtering, an attacker often adversarially generates spam emails based on his knowledge of the spam filter implemented by the email service provider (Brückner & Scheffer, 2011; Zhou et al., 2019). In addition to malicious attacks, sometimes the data providers may manipulate data for their own interests. For instance, health insurance policy holders may decide to modify self-reported data to reduce their premiums. On the other hand, the insurers (the "defenders" in this scenario) aim to select a good price model for the true data despite only seeing the modified data.

In fact, these scenarios can be modeled by the Stackelberg prediction game (SPG) (Brückner & Scheffer, 2011; Shokri et al., 2012; Zhou & Kantarcioglu, 2016; Wahab et al., 2016; Zhou et al., 2019; Bishop et al., 2020) which characterizes the interactions between two players, a learner (or, a leader) and a data provider (or, a follower). In this setting, the learner makes the first move by selecting a learning model. Then the data provider, with full knowledge of the learner's model, is allowed to modify its data. The learner's goal is to minimize its own loss function under the assumption that the training data has been optimally modified from the data provider's perspective. From the above description, we see that the SPG model concerns two levels of optimization problems: The follower optimally manipulates its data and the leader makes its optimal decision taking into account the data manipulation. Formally, it is often formulated as a hierarchical mathematical problem or a bi-level optimization problem, which is generally NP-hard even in the simplest case with linear constraints and objectives (Jeroslow, 1985).

[1]School of Data Science, Fudan University, China [2]School of Mathematical Sciences, Xiamen University, China [3]School of Computer Science, Carnegie Mellon University, USA. Correspondence to: Rujun Jiang <rjjiang@fudan.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

To overcome this issue, Bishop et al. (2020) take the first step to focus on a subclass of SPGs that can be reformulated as fractional programs. Specifically, they assume that all the loss functions of the leader and the follower are least squares, and that a quadratic regularizer is added to the follower's loss to penalise its manipulation of the data. This assumption eventually turns the bi-level optimization problem into a single-level fractional optimization task which is proven to be polynomially globally solvable. Since no other assumption is made about the learner and data provider, this subclass of SPG, termed as the SPG-LS, is general enough to be applied in wide fields. However, the bisection algorithm proposed in Bishop et al. (2020) involves solving several tens of semidefinite programs (SDPs) which are computationally prohibitive in practice. Later, Wang et al. (2021b) improves over Bishop et al. (2020) by showing that the SPG-LS can be globally solved via solving only a single SDP with almost the same size as the ones in Bishop et al. (2020). Furthermore, this single SDP can be reduced to a second order cone program (SOCP). It is shown in Wang et al. (2021b) that the SOCP approach for solving SPG-LS can be over 20,000+ times faster than the bisection method proposed in Bishop et al. (2020). Yet, the SOCP method is still not well-suited for solving large-scale SPG-LS. Indeed, the spectral decomposition in the SOCP reformulation process is time-consuming when the future dimension is high. This inevitably reduces the practical applicability of the SOCP approach for the SPG-LS.

In this paper, we present a novel reformulation to resolve the above mentioned issues of the SOCP method. Specifically, a nonlinear change of variables is proposed to reformulate the SPG-LS as a spherically constrained least squares (SCLS) problem. Then, we prove that an optimal solution to the SPG-LS can be recovered easily from any optimal solution to the SCLS under a mild assumption. The SCLS can be seen as an equality constrained version of the trust region subproblem (Conn et al., 2000), which admits a large amount of existing research on practical algorithms and theoretical complexity analysis. Based on this, we show that an ϵ optimal solution[1] of the SCLS and thus SPG-LS can be solved in $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$ flops, where N denotes the number of nonzero entries of the data matrix and $\tilde{\mathcal{O}}(\cdot)$ hides the logarithmic factors. This means there exits a linear time algorithm for finding an ϵ optimal solution of the SPG-LS. Moreover, we demonstrate the empirical efficiency of our SCLS reformulation when matrix factorization free methods like the Krylov subspace method (Gould et al., 1999; Zhang & Shen, 2018) and the Riemannian trust region Newton (RTRNewton) method (Absil et al., 2007) are used as solvers.

[1] We say $\bar{\mathbf{x}}$ is an ϵ optimal solution for an optimization problem $\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$, if $\bar{\mathbf{x}} \in \mathcal{X}$ and $f(\bar{\mathbf{x}}) \leq \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) + \epsilon$.

We summarise our contributions as follows:

- We derive an SCLS reformulation for the SPG-LS that avoids spectral decomposition steps (which are expensive when the involved data matrices are large). Moreover, we show that an optimal solution to the SPG-LS can be recovered from any optimal solution to the SCLS reformulation under a mild condition.

- Based on the reformulation, we show that an ϵ optimal solution for the SCLS can be found using $\tilde{\mathcal{O}}(1/\sqrt{\epsilon})$ matrix vector products. In other words, an ϵ solution can be obtained in running time $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$, where N is the number of nonzeros in the data matrix. Moreover, we show that an ϵ optimal solution of SCLS can be used to recover an ϵ optimal solution for the original SPG-LS.

- Two practically efficient algorithms, which are factorization free, are adopted to solve the SCLS reformulation. We show that the SCLS approach significantly outperforms the SOCP approach with experiments on both real and synthetic data sets.

2. Preliminaries

In this section, we elaborate on the SPG-LS problem adopting the same terminology as in Bishop et al. (2020); Wang et al. (2021b). To have a better understanding of our reformulation, a brief review of methods in Wang et al. (2021b), which is the fastest existing method for solving the SPG-LS, will also be provided.

We assume that the learner has access to m sample tuples $S = \{(\mathbf{x}_i, y_i, z_i)\}_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^n$ is input data with n features, y_i and z_i are the true output label of $\mathbf{x}_i$ and the label that the data provider would like to achieve, respectively. These samples are assumed to follow some fixed but unknown distribution $\mathcal{D}$. The learner aims at training a linear predictor $\mathbf{w} \in \mathbb{R}^n$ to best estimate the true output label y_i given the fake data. Meanwhile, the data provider, with full knowledge of the learner's predictive model $\mathbf{w}$, selects its own strategy (i.e., the modified data $\hat{\mathbf{x}}_i$) to make the corresponding prediction $\mathbf{w}^T\hat{\mathbf{x}}_i$ close to the desired label z_i. Note that there is also a regularizer, $\gamma > 0$, to control the deviation from the original data $\mathbf{x}_i$. This hyper-parameter adjusts the trade-off between data manipulation and closeness to the aimed target.

The problem can be modeled as a Stackelberg prediction game (Brückner & Scheffer, 2011; Bishop et al., 2020). On the one hand, each data provider aims to minimize its own loss function with a regularizer that penalizes the manipulation of the data by solving the following optimization

problem:

$$\mathbf{x}_i^* = \underset{\hat{\mathbf{x}}_i}{\operatorname{argmin}} \left\|\mathbf{w}^T\hat{\mathbf{x}}_i - z_i\right\|^2 + \gamma\left\|\mathbf{x}_i - \hat{\mathbf{x}}_i\right\|_2^2 \quad i \in [m],$$

where $\mathbf{w}$ is the learner's model parameter that is known to the data provider. On the other hand, the learner seeks to minimize the least squares loss with the modified data $\{\mathbf{x}_i^*\}_{i=1}^m$:

$$\mathbf{w}^* \in \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^m \left\|\mathbf{w}^T\mathbf{x}_i^* - y_i\right\|^2,$$

To find the Stackelberg equilibrium of the two players, we focus on the following bi-level optimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{w}} & \|X^*\mathrm{w} - \mathbf{y}\|^2 \\ \text{s.t.} & X^* = \underset{\hat{X}}{\operatorname{argmin}} \|\hat{X}\mathbf{w} - \mathbf{z}\|^2 + \gamma\|\hat{X} - X\|_F^2, \end{array} \tag{1}$$

where the i-th row of $X \in \mathbb{R}^{m\times n}$ is the input sample $\mathbf{x}_i$ and the i-th entries of $\mathbf{y}, \mathbf{z} \in \mathbb{R}^m$ are labels y_i and z_i, respectively.

In the following section, we have a quick review of single SDP and SOCP methods in Wang et al. (2021b).

2.1. SDP Reformulation

By using the Sherman-Morrison formula (Sherman & Morrison, 1950), the SPG-LS can be rewritten as a quadratic fractional program (Bishop et al., 2020)

$$\inf_{\mathbf{w}} \left\| \frac{\frac{1}{\gamma}\mathbf{z}\mathbf{w}^T\mathbf{w} + X\mathbf{w}}{1 + \frac{1}{\gamma}\mathbf{w}^T\mathbf{w}} - \mathbf{y} \right\|^2. \tag{2}$$

Introducing an augmented variable $\alpha = \mathbf{w}^T\mathbf{w}/\gamma$, we have the following quadratic fractional programming (QFP) reformulation.

$$\begin{array}{ll} \inf_{\mathbf{w},\,\alpha} & v(\mathbf{w},\alpha) \triangleq \left\| \frac{\alpha\mathbf{z} + X\mathbf{w}}{1+\alpha} - \mathbf{y} \right\|^2 \\ \text{s.t.} & \mathbf{w}^T\mathbf{w} = \gamma\alpha. \end{array} \tag{3}$$

Lemma 2.1 (Theorem 3.3 in Wang et al. (2021b)). *Problem (3) is equivalent to the following SDP*

$$\begin{array}{ll} \sup_{\mu,\lambda} & \mu \\ \text{s.t.} & A - \mu B + \lambda C \succeq 0, \end{array} \tag{4}$$

where $A = \begin{pmatrix} X^TX & X^T(\mathbf{z}-\mathbf{y}) & -X^T\mathbf{y} \\ (\mathbf{z}-\mathbf{y})^TX & \|\mathbf{z}-\mathbf{y}\|^2 & -(\mathbf{z}-\mathbf{y})^T\mathbf{y} \\ -\mathbf{y}^TX & -\mathbf{y}^T(\mathbf{z}-\mathbf{y}) & \mathbf{y}^T\mathbf{y} \end{pmatrix}, B = \begin{pmatrix} \mathcal{O}_n & & \\ & 1 & 1 \\ & 1 & 1 \end{pmatrix}$ *and* $C = \begin{pmatrix} \frac{I_n}{\gamma} & & \\ & 0 & -\frac{1}{2} \\ & -\frac{1}{2} & 0 \end{pmatrix}$. *Here $\mathcal{O}_n$ denotes a $n \times n$ matrix with all entries being zeros and I_n denotes the $n \times n$ identity matrix.*

2.2. SOCP Reformulation

Wang et al. (2021b) further constructed an invertible matrix V such that A, B and C are simultaneously congruent to arrow matrices via the change of variables associated to V, i.e.,

$$\tilde{A} := V^TAV = \begin{pmatrix} D & \mathbf{b} \\ \mathbf{b}^T & c \end{pmatrix},$$

where $D = \mathrm{Diag}(d_1, \ldots, d_{n+1}) \in \mathbb{R}^{(n+1)\times(n+1)}$, $\mathbf{b} \in \mathbb{R}^{n+1}$ and $c \in \mathbb{R}$, and

$$\tilde{B} := VBV = \begin{pmatrix} \mathcal{O}_{n+1} & \\ & 4 \end{pmatrix} \text{ and } \tilde{C} = V^TCV = \begin{pmatrix} \frac{1}{\gamma}I_{n+1} & \\ & -1 \end{pmatrix}.$$

Therefore the linear matrix inequality (LMI) constraint in (4) is equivalent to $\tilde{A} - \mu\tilde{B} + \lambda\tilde{C} \succeq 0$. Using the generalized Schur complement, we further obtain an SOCP reformulation as follows.

Lemma 2.2 (Theorem 4.1 in Wang et al. (2021b)). *With the same notation in this section, problem (4) is equivalent to the following SOCP problem*

$$\begin{array}{ll} \sup_{\mu,\lambda,\mathbf{s}} & \mu \\ \text{s.t.} & d_i + \frac{\lambda}{\gamma} \geq 0,\ i \in [n+1], \\ & c - 4\mu - \lambda - \sum_{i=1}^{n+1} s_i \geq 0, \\ & s_i(d_i + \frac{\lambda}{\gamma}) \geq b_i^2,\ s_i \geq 0,\ i \in [n+1]. \end{array} \tag{5}$$

To obtain the SOCP reformulation, we need a spectral decomposition to a matrix of order $(n+1)\times(n+1)$ (Wang et al., 2021b), which is expensive when the dimension is high and may lead to inaccurate solutions when the matrix is ill-conditioned. To amend this issue, we obtain in the next section a factorization free method based on a novel reformulation of problem (2).

3. Main Results

In this section, we show that using a nonlinear change of variables, we can rewrite (3) as a least squares problem over the unit sphere. This is the key observation of our paper.

Before presenting our main results, we first make a blanket assumption on the nonemptiness of the optimal solution set of (2).

Assumption 3.1. Assume that the optimal solution set of (2) (or equivalently, (3)) is nonempty.

Our main result is that under Assumption 3.1, the QFP (3) can be reformulated as a spherical constrained least squares (SCLS) problem

$$\begin{array}{ll} \min\limits_{\tilde{\mathbf{w}},\,\tilde{\alpha}} & \tilde{v}(\tilde{\mathbf{w}},\tilde{\alpha}) \triangleq \left\| \frac{\tilde{\alpha}}{2}\mathbf{z} + \frac{\sqrt{\gamma}}{2}X\tilde{\mathbf{w}} - (\mathbf{y} - \frac{\mathbf{z}}{2}) \right\|^2 \\ \text{s.t.} & \tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + \tilde{\alpha}^2 = 1. \end{array} \tag{6}$$

A formal statement is deferred to Theorem 3.4.

Before presenting our main results, we first introduce two lemmas that show how, given a feasible solution in (3), we can construct a feasible solution with the same objective value in (6) and vice versa (up to a minor achievability issue).

Lemma 3.2. *Suppose* $(\mathbf{w}, \alpha)$ *is a feasible solution of* (3). *Then* $(\tilde{\mathbf{w}}, \tilde{\alpha})$, *defined as*

$$\tilde{\mathbf{w}} := \frac{2}{\sqrt{\gamma}(\alpha+1)}\mathbf{w} \quad \textit{and} \quad \tilde{\alpha} := \frac{\alpha-1}{\alpha+1}, \tag{7}$$

is feasible to (6) *and* $v(\mathbf{w}, \alpha) = \tilde{v}(\tilde{\mathbf{w}}, \tilde{\alpha})$.

Proof. We first note that $(\tilde{\mathbf{w}}, \tilde{\alpha})$ are well-defined as $\alpha \geq 0$ by the feasibility of $(\mathbf{w}, \alpha)$ in (3). Next, we check feasibility of $(\tilde{\mathbf{w}}, \tilde{\alpha})$ in (6):

$$\begin{aligned}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + \tilde{\alpha}^2 &= \frac{4}{\gamma(\alpha+1)^2}\mathbf{w}^T\mathbf{w} + \frac{(\alpha-1)^2}{(\alpha+1)^2} \\ &= \frac{4\alpha + (\alpha-1)^2}{(\alpha+1)^2} = 1.\end{aligned}$$

Here, the second equality follows from the fact that $\mathbf{w}^T\mathbf{w} = \gamma\alpha$. Finally, we see that

$$\begin{aligned}&\tilde{v}(\tilde{\mathbf{w}}, \tilde{\alpha}) \\ &= \left\| \frac{\tilde{\alpha}}{2}\mathbf{z} + \frac{\sqrt{\gamma}}{2}X\tilde{\mathbf{w}} - (\mathbf{y} - \frac{\mathbf{z}}{2}) \right\|^2 \\ &= \left\| \frac{\alpha-1}{2(\alpha+1)}\mathbf{z} + \frac{\sqrt{\gamma}}{2}X\frac{2}{\sqrt{\gamma}(\alpha+1)}\mathbf{w} - (\mathbf{y} - \frac{\mathbf{z}}{2}) \right\|^2 \\ &= \left\| \frac{\alpha\mathbf{z} + X\mathbf{w}}{\alpha+1} - \mathbf{y} \right\|^2 = v(\mathbf{w}, \alpha).\end{aligned}$$

This completes the proof. □

Lemma 3.3. *Suppose* $(\tilde{\mathbf{w}}, \tilde{\alpha})$ *is feasible to* (6) *with* $\tilde{\alpha} \neq 1$. *Then* $(\mathbf{w}, \alpha)$, *defined as*

$$\mathbf{w} := \frac{\sqrt{\gamma}}{1-\tilde{\alpha}}\tilde{\mathbf{w}} \quad \textit{and} \quad \alpha := \frac{1+\tilde{\alpha}}{1-\tilde{\alpha}}, \tag{8}$$

is feasible to (3) *and* $\tilde{v}(\tilde{\mathbf{w}}, \tilde{\alpha}) = v(\mathbf{w}, \alpha)$.

Proof. We first note that $(\mathbf{w}, \alpha)$ are well-defined as $\tilde{\alpha} \neq 1$. Next, we check feasibility of $(\mathbf{w}, \alpha)$ in (2):

$$\mathbf{w}^T\mathbf{w} = \frac{\gamma}{(1-\tilde{\alpha})^2}\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} = \frac{\gamma(1-\tilde{\alpha}^2)}{(1-\tilde{\alpha})^2} = \frac{\gamma(1+\tilde{\alpha})}{1-\tilde{\alpha}} = \gamma\alpha.$$

Here, the second equality follows from the fact that $\tilde{\mathbf{w}}^T\tilde{\mathbf{w}} + \tilde{\alpha}^2 = 1$. Finally, we check the objective value of $(\mathbf{w}, \alpha)$:

$$\begin{aligned}&\left\| \tfrac{\alpha\mathbf{z}+X\mathbf{w}}{1+\alpha} - \mathbf{y} \right\|^2 = \left\| \tfrac{(1+\tilde{\alpha})\mathbf{z}+\sqrt{\gamma}X\tilde{\mathbf{w}}}{(1-\tilde{\alpha})+(1+\tilde{\alpha})} - \mathbf{y} \right\|^2 \\ =\ &\left\| \tfrac{\tilde{\alpha}}{2}\mathbf{z} + \tfrac{\sqrt{\gamma}}{2}X\tilde{\mathbf{w}} - (\mathbf{y} - \mathbf{z}/2) \right\|^2.\end{aligned}$$

□

Let v^* and $\tilde{v}^*$ be the optimal values of (3) and (6), respectively. Now we are ready to present our main results.

Theorem 3.4. *Given Assumption 3.1, then there exists an optimal solution* $(\tilde{\mathbf{w}}, \tilde{\alpha})$ *to* (6) *with* $\tilde{\alpha} \neq 1$. *Moreover,* $(\mathbf{w}, \alpha)$, *defined by* (8), *is an optimal solution to* (3) *and* $v^* = v(\mathbf{w}, \alpha) = \tilde{v}(\tilde{\mathbf{w}}, \tilde{\alpha}) = \tilde{v}^*$.

Proof. Since the feasible region of (6) is compact and $\tilde{v}$ is continuous, it follows from the well-known Weierstrass theorem that there exists at least one optimal solution to (6).

Note that if $(\tilde{\mathbf{w}}, \tilde{\alpha})$ with $\tilde{\alpha} = 1$ is a feasible solution in (6), we must have $\tilde{\mathbf{w}} = \mathbf{0}$. Now we claim that $(\mathbf{0}, 1)$ cannot be the *unique* optimal solution to (6). Suppose on the contrary that $(\mathbf{0}, 1)$ is the unique optimal solution to (6). Let $(\mathbf{w}^*, \alpha^*)$ be any optimal solution to (3). Then, from Lemma 3.2, we have

$$v(\mathbf{w}^*, \alpha^*) = \tilde{v}(\tilde{\mathbf{w}}^*, \tilde{\alpha}^*), \tag{9}$$

where $\tilde{\mathbf{w}}^* := \frac{2}{\sqrt{\gamma}(\alpha^*+1)}\mathbf{w}^*$, $\tilde{\alpha}^* := \frac{\alpha^*-1}{\alpha^*+1} < 1$, and $(\tilde{\mathbf{w}}^*, \tilde{\alpha}^*)$ is feasible to (6). Since $(\mathbf{0}, 1)$ is the unique optimal solution to (6), it holds that

$$\tilde{v}(\tilde{\mathbf{w}}^*, \tilde{\alpha}^*) > \tilde{v}(\mathbf{0}, 1) = \|\mathbf{z} - \mathbf{y}\|^2. \tag{10}$$

On the other hand, for any $\mathbf{w} \neq 0$ and $t > 0$, the pair $(t\mathbf{w}, t^2\mathbf{w}^T\mathbf{w}/\gamma)$ is clearly feasible to (3) with objective value

$$\begin{aligned}&v(t\mathbf{w}, t^2\mathbf{w}^T\mathbf{w}/\gamma) \\ =\ &\left\| \tfrac{t^2\mathbf{w}^T\mathbf{w}/\gamma}{1+t^2\mathbf{w}^T\mathbf{w}/\gamma}\mathbf{z} + \tfrac{t}{1+t^2\mathbf{w}^T\mathbf{w}/\gamma}X\mathbf{w} - \mathbf{y} \right\|^2 \\ \to\ &\|\mathbf{z} - \mathbf{y}\|^2, \text{ as } t \to \infty.\end{aligned} \tag{11}$$

Consequently, for sufficiently large t, we must have from (9), (10) and (11) that

$$v(\mathbf{w}^*, \alpha^*) > v(t\mathbf{w}, t^2\mathbf{w}^T\mathbf{w}/\gamma),$$

which contradicts the optimality of $(\mathbf{w}^*, \alpha^*)$ to (3).

The above claim shows that there exists an optimal solution $(\tilde{\mathbf{w}}, \tilde{\alpha})$ to (6) with $\tilde{\alpha} \neq 1$. Then, Lemma 3.3 yields $\tilde{v}(\tilde{\mathbf{w}}, \tilde{\alpha}) = \tilde{v}^* = v(\mathbf{w}, \alpha) \geq v^*$ with $(\mathbf{w}, \alpha)$ defined in (8). Similarly, under Assumption 3.1, from Lemma 3.2, we see that $v^* \geq \tilde{v}^*$. Thus, $v^* = \tilde{v}^*$. The proof is completed. □

We remark that the other direction of above theorem also holds. That is, under Assumption 3.1, there exists an optimal solution $(\mathbf{w}, \alpha)$ to (3), and, furthermore, $(\tilde{\mathbf{w}}, \tilde{\alpha})$, defined by (7), is optimal to (6). We also remark that using the relationship (8) and the equivalence of (2) and (3), an ϵ optimal solution of the SCLS can be used to recover an ϵ optimal solution of the SPG-LS.

Letting $\hat{L} = \begin{pmatrix} \frac{\sqrt{\gamma}}{2}X & \frac{\mathbf{z}}{2} \end{pmatrix}$ and $\mathbf{r} = \begin{pmatrix} \tilde{\mathbf{w}} \\ \tilde{\alpha} \end{pmatrix}$, we can rewrite problem (6) in a more compact form

$$\min_{\mathbf{r}} \ q(\mathbf{r}) \quad \text{s.t. } \mathbf{r}^T\mathbf{r} = 1, \tag{12}$$

where $q(\mathbf{r})$ is a least squares

$$q(\mathbf{r}) = \|\hat{L}\mathbf{r} - (\mathbf{y} - \mathbf{z}/2)\|_2^2 = \mathbf{r}^T H\mathbf{r} + 2\mathbf{g}^T\mathbf{r} + p \quad (13)$$

with $H = \hat{L}^T\hat{L}$, $\mathbf{g} = \hat{L}^T(\mathbf{z}/2-\mathbf{y})$, $p = (\mathbf{z}/2-\mathbf{y})^T(\mathbf{z}/2-\mathbf{y})$.

In the following of this paper, we focus on solving (12). Problem (12) is closed related to the well-known trust region subproblem (TRS), where the sphere constraint is replaced by a unit ball constraint. There exist various methods for solving (12) from the literature on TRS (Moré & Sorensen, 1983; Gould et al., 1999; Conn et al., 2000; Hazan & Koren, 2016; Ho-Nguyen & Kilinc-Karzan, 2017; Zhang et al., 2017; 2018; Zhang & Shen, 2018; Carmon & Duchi, 2018), or the generalized trust region subproblem (GTRS) (Moré, 1993; Pong & Wolkowicz, 2014; Jiang & Li, 2019; 2020; Wang & Kılınç-Karzan, 2020; Wang et al., 2021a), which minimizes a (possible nonconvex) quadratic function over a (possible nonconvex) quadratic inequality or equality constraint. Note that the TRS differs from the SCLS in the constraint, and the GTRS contains the SCLS as a special case.

4. Complexity and Algorithms

In this section, we first show that in theory there exists a linear time algorithm to find an ϵ optimal solution for the SPG-LS. After that, we introduce two practically efficient algorithms to solve (12) (and thus recover a solution for the SPG-LS).

We point out that the linear time algorithms for the TRS (Hazan & Koren, 2016; Ho-Nguyen & Kilinc-Karzan, 2017) can be adapted to design a linear time algorithm with complexity $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$ for the SCLS to achieve an ϵ optimal solution, and the linear time algorithms for the GTRS (Jiang & Li, 2020; Wang & Kılınç-Karzan, 2020; Wang et al., 2021a) indicate that the SCLS, as a special case of the GTRS, can also be solved in linear time $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$. Here N denotes the number of nonzero entries in the data matrix, and the logarithm in the runtime comes from the probability of success in Lanczos type methods for finding the smallest eigenvalue of a matrix. Once we obtain a solution $\tilde{\mathbf{r}}$ such that $\|\tilde{\mathbf{r}}\| = 1$ and $q(\tilde{\mathbf{r}}) \le q(\mathbf{r}^*) + \epsilon$, where $\mathbf{r}^*$ is an optimal solution of (12), we can set $\begin{pmatrix}\tilde{\mathbf{w}}\\ \tilde{\alpha}\end{pmatrix} = \tilde{\mathbf{r}}$ and $(\mathbf{w}, \alpha)$ as in (8). Then $\mathbf{w}$ is an ϵ optimal solution to (2) because $v(\mathbf{w}, \alpha) = \tilde{v}(\tilde{\mathbf{w}}, \tilde{\alpha}) = q(\mathbf{r})$ and $q(\mathbf{r}^*) = v^*$ for the same reasoning . Thus, one can obtain an ϵ optimal solution to SPG-LS in runtime $\tilde{\mathcal{O}}\left(N/\sqrt{\epsilon}\right)$ as the main cost is in solving the SCLS (12).

However, in practice the computation of even an approximate minimum eigenvalue may be expensive. Instead, we will introduce two highly efficient algorithms to solve (6) without computing approximate eigenvalues. One is the Krylov subspace method (adapted to the spherically constrained case) proposed in Gould et al. (1999), and the other is the Riemannian trust region Newton (RTRNewton) method proposed in Absil et al. (2007).

4.1. The Krylov Subspace Method

The simplest idea of the Krylov subspace method (Section 5 in Gould et al. (1999)) solves a sequence of smaller dimensional problems in the same form of (12). Specifically, define $(k+1)$st Krylov subspace

$$\mathcal{K}_k := \{\mathbf{g}, H\mathbf{g}, H^2\mathbf{g}, \ldots, H^k\mathbf{g}\}.$$

Let $Q_k = [\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_k] \in \mathbb{R}^{(n+1)\times(k+1)}$ be an orthonormal basis produced by the generalized Lanczos process. Then assuming $\dim \mathcal{K}_k = k+1$, we have that $Q_k^T H Q_k$ is a tridiagonal matrix with $Q_k^T Q_k = I_{k+1}$. Each iteration of the Krylov subspace method solves the following subproblem (adapted to the spherical constrained case)

$$\min_{r\in\mathcal{K}_k, \|\mathbf{r}\|=1} \mathbf{r}^T H\mathbf{r} + 2\mathbf{g}^T\mathbf{r} + p. \quad (14)$$

Gould et al. (1999) proved that the above subproblem can be solved efficiently in $\mathcal{O}(k)$ flops, if we use a safeguarded Newton's method, where the most expensive cost is k matrix-vector products for $H^t\mathbf{g}$, with $t \in [k]$. We remark that though Gould et al. (1999) considered the case $\|\mathbf{r}\| \le 1$, the two cases $\|\mathbf{r}\| \le 1$ and $\|\mathbf{r}\| = 1$ are essentially the same if the inequality in the TRS is active, which occur if $\|(H - \lambda_{\min} I)^\dagger \mathbf{g}\| > 1$[2]. Here $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse of a matrix $(\cdot)$.

To achieve better practical performance, Gould et al. (1999) proposed the generalized Lanczos trust-region (GLTR) method, which is an efficient implementation of the above Krylov subspace method. Based on an efficient nested restarting strategy, Zhang & Shen (2018) further proposed a nested Lanczos method for TRS (LTRSR), which is an improvement for GLTR.

The convergence behavior of the Krylov subspace method is also well analyzed in the literature. The optimality condition of problem (12) is characterized as follows (adapted from Chapter 7 in Conn et al. (2000))

$$\begin{aligned}(H + \lambda^* I)\mathbf{r}^* &= -\mathbf{g},\\ H + \lambda^* I &\succeq 0,\\ \|\mathbf{r}^*\| &= 1,\end{aligned} \quad (15)$$

where λ^* is the corresponding Lagrangian multiplier. It is shown that there always exists an optimal solution $\mathbf{r}^*$ and a unique Lagrangian multiplier λ^*, because different λ^*s yield different values for $\|\mathbf{r}^*\|$, contradicting $\|\mathbf{r}^*\| = 1$.

[2] This can be easily derived from the proof of Proposition 4.1.

Define

$$\kappa = \frac{\lambda_{\max} + \lambda^*}{\lambda_{\min} + \lambda^*}, \tag{16}$$

for $\lambda^* \geq -\lambda_{\min}$ and use the convention $\frac{1}{0} = \infty$. Here κ is regarded as the condition number of (12) (Carmon & Duchi, 2018). Zhang & Shen (2018) and Carmon & Duchi (2018) demonstrated that when $\kappa < \infty$, the Krylov subspace method satisfies

$$f(\mathbf{r}_k) - f(\mathbf{r}^*) \leq \mathcal{O}(\exp(-k/\sqrt{\kappa})),$$

where $\mathbf{r}_k$ is an optimal solution of (14), and $\mathbf{r}^*$ is an optimal solution for (12). Carmon & Duchi (2018) further proved that for all cases including $\lambda^* = -\lambda_{\min}$, a variant of the Krylov subspace method where $\mathbf{g}$ is perturbed with random vector will output a solution $\mathbf{r}_k$ satisfies

$$f(\mathbf{r}_k) - f(\mathbf{r}^*) \leq \tilde{\mathcal{O}}(1/k^2).$$

for the SCLS (adapted from their analysis for the TRS). This indeed gives another $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$ time algorithm for solving the SPG-LS up to ϵ tolerance; see also Wang et al. (2021a) for extensions of this idea to the GTRS.

Next we relate the existing convergence results with problem (12).

Proposition 4.1. *If* $\|(H - \lambda_{\min} I)^\dagger \mathbf{g}\| > 1$, *we must have* $\lambda^* > -\lambda_{\min}$ *and thus*

$$f(\mathbf{r}_k) - f(\mathbf{r}^*) \leq \mathcal{O}(\exp(-k/\sqrt{\kappa})),$$

for κ *defined in* (16).

Proof. Note that if $\lambda^* = -\lambda_{\min}$, then the first equation in (15) implies that $\mathbf{r}^* = -(H + \lambda^* I)^\dagger \mathbf{g}$ and thus the assumption in the proposition implies $\|\mathbf{r}^*\| > 1$. However, this violates the constraint $\|\mathbf{r}^*\| = 1$. Therefore we must have $\lambda^* > -\lambda_{\min}$ and thus $\kappa < \infty$. □

In fact, we checked the data in the experiments in Section 5 and found that $\|\bar{\mathbf{r}}\| > 1$ always holds in real-world datasets and our synthetic datasets.

4.2. The Riemannian Trust Region Newton (RTRNewton) Method

The feasible set in Problem (12) forms a unit sphere $\mathcal{S}^n = \{\mathbf{r} \in \mathbb{R}^{n+1} : \mathbf{r}^T\mathbf{r} = 1\}$. When $\mathcal{S}^n$ is endowed with the Euclidean metric $\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{v}^T\mathbf{u}$, the unit sphere is a Riemannian manifold (Absil et al., 2008). Therefore, the RTRNewton method proposed in Absil et al. (2007) can be used. The RTRNewton method for Problem (6) is summarized in Algorithm 1.

Algorithm 1 relies on the notion of Riemannian gradient, Riemannian Hessian, and retraction. We refer to Absil et al. (2008) for their rigorous definitions. Here, we give the Riemannian gradient, the Riemannian Hessian for Problem (12) and the used retraction.

Algorithm 1 A Riemannian Trust Region Newton Method

Require: Initial iterate $\mathbf{r}_0$, real numbers $\overline{\Delta} > 0$, $\Delta_0 \in (0, \overline{\Delta})$, $c \in (0, 0.25)$, $\tau_1 \in (0, 1)$, and $\tau_2 > 1$;

1: **for** $k = 0, 1, 2, \ldots$ **do**

2: Obtain $s_k \in \mathbb{R}^d$ by (approximately) solving

$$\mathbf{s}_k \approx \operatorname*{argmin}_{\|\mathbf{s}\|_2 \leq \Delta_k} m_k(\mathbf{s}), \tag{17}$$

where $m_k(\mathbf{s}) = q(\mathbf{r}_k) + \mathbf{s}^T \operatorname{grad} q(\mathbf{r}_k) + \frac{1}{2}\mathbf{s}^T \operatorname{Hess} q(\mathbf{r}_k)[\mathbf{s}]$, grad q denotes the Riemannian gradient of q in (18), and Hess q denotes the Riemannian Hessian of q in (19);

3: Set $\rho_k \leftarrow \frac{q(\mathbf{r}_k) - q(R_{\mathbf{r}_k}(\mathbf{s}_k))}{m_k(0) - m_k(\mathbf{s}_k)}$, where R denotes the retraction in (20);

4: **if** $\rho_k > c$ **then**

5: $\mathbf{r}_{k+1} \leftarrow R_{\mathbf{r}_k}(\mathbf{s}_k)$;

6: **else**

7: $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k$;

8: **end if**

9: **if** $\rho_k > \frac{3}{4}$ **then**

10: If $\|\mathbf{s}_k\| \geq 0.8\Delta_k$, then $\Delta_{k+1} \leftarrow \min(\tau_2\Delta_k, \overline{\Delta})$; otherwise $\Delta_{k+1} \leftarrow \Delta_k$;

11: **else if** $\rho_k < 0.1$ **then**

12: $\Delta_{k+1} \leftarrow \tau_1\Delta_k$;

13: **else**

14: $\Delta_{k+1} \leftarrow \Delta_k$;

15: **end if**

16: **end for**

The Riemannian gradient of q is given by

$$\operatorname{grad} q(\mathbf{r}) = P_{\mathrm{T}_{\mathbf{r}}\mathcal{S}^n} \nabla q(\mathbf{r}) = (I - \mathbf{r}\mathbf{r}^T)(2H\mathbf{r} + 2\mathbf{g}), \tag{18}$$

where $P_{\mathrm{T}_{\mathbf{r}}\mathcal{S}^n}$ denotes the orthogonal projection onto the tangent space at $\mathbf{r}$ with $\mathrm{T}_{\mathbf{r}}\mathcal{S}^n = \{\mathbf{s} : \mathbf{s}^T\mathbf{r} = 0\}$, and $\nabla q(\mathbf{r})$ denotes the Euclidean gradient of q, i.e., $\nabla q(\mathbf{r}) = 2H\mathbf{r} + 2\mathbf{g}$. The action of the Riemannian Hessian of q at $\mathbf{r}$ along $\mathbf{v} \in \mathrm{T}_{\mathbf{r}}\mathcal{S}^n$ is given by

$$\begin{aligned} \operatorname{Hess} q(\mathbf{r})[\mathbf{v}] =& P_{\mathrm{T}_{\mathbf{r}}\mathcal{S}^n}(\nabla^2 q(\mathbf{r})\mathbf{v} - \mathbf{v}\mathbf{r}^T \nabla q(\mathbf{r})), \\ =& (I - \mathbf{r}\mathbf{r}^T)(2H\mathbf{v} - 2\mathbf{v}\mathbf{r}^T(H\mathbf{r} + \mathbf{g})), \end{aligned} \tag{19}$$

where $\nabla^2 q(\mathbf{r}) = 2H$ denotes the Euclidean Hessian of q at $\mathbf{r}$. The retraction R that we use is given by

$$R_{\mathbf{r}}(\mathbf{v}) = \frac{\mathbf{r} + \mathbf{v}}{\|\mathbf{r} + \mathbf{v}\|}, \tag{20}$$

where $\mathbf{r} \in \mathcal{S}^n$ and $\mathbf{v} \in \mathrm{T}_{\mathbf{r}}\mathcal{S}^n$.

The subproblem (17) is approximately solved by the truncated conjugate gradient method. We use the implementations in ROPTLIB (Huang et al., 2018).

The global convergence and local superlinear convergence rate of RTRNewton have been established by Theorem 7.4.4 and Theorem 7.4.11 of Absil et al. (2008). We state the results in the theorem below.

Theorem 4.2. *Let $\{\mathbf{r}_k\}$ be a sequence of iterates generated by Algorithm 1. It follows that*

$$\lim_{k\to\infty} \operatorname{grad} q(\mathbf{r}_k) = 0.$$

Suppose $\mathbf{r}^*$ is a nondegenerate local minimizer of q, i.e., $\operatorname{grad} q(\mathbf{r}^*) = 0$ and $\operatorname{Hess} q(\mathbf{r}^*)$ is positive definite. Then there exists $c > 0$ such that for all sequence $\{\mathbf{r}_k\}$ generated by Algorithm 1 converging to $\mathbf{r}^*$, there exists $K > 0$ such that for all $k > K$,

$$\operatorname{dist}(\mathbf{r}_{k+1}, \mathbf{r}^*) \le c \operatorname{dist}(\mathbf{r}_k, \mathbf{r}^*)^2. \tag{21}$$

The proof for the theorem is deferred to Appendix A. The global convergence rate has also been established where the iteration complexity is $\mathcal{O}\left(\epsilon_g^{-2}\right)$ for $\|\operatorname{grad} q(x)\| \le \epsilon_g$. We refer interested readers to Theorem 3.9 of Boumal et al. (2019).

4.3. Time complexity comparisons

In this section, we give a theoretical worst case time complexity of different methods for solving the SPG-LS. First we point out that the RTRNewton cannot be guaranteed to converge to the *global* minimum of SPG-LS. In the worst case, the RTRNewton needs to solve $\mathcal{O}\left(\epsilon_g^{-2}\right)$ many trust region subproblems. This means the time complexity is much worse than the Krylov subspace method (Carmon & Duchi, 2018) studied in Section 4.1.

Next we compare the time complexity of the Krylov subspace method and the SOCP method. In the case of dealing with a sparse data matrix, the time complexity of the Krylov subspace method is $\tilde{\mathcal{O}}\left(N/\sqrt{\epsilon}\right)$, where N is the number of nonzero entries in the data matrix X. Here, we use the fact that the cost of the matrix-vector product in the Krylov subspace method is $\mathcal{O}(N)$ as we can compute $H\mathbf{r}$ by $\hat{L}^T(\hat{L}\mathbf{r})$ for any given $\mathbf{r} \in \mathbb{R}^n$. If $\kappa < \infty$, then the complexity can be further improved to $\mathcal{O}(N\log(1/\epsilon))$. In the dense case with $m = \mathcal{O}(n)$, the complexity is $\tilde{\mathcal{O}}\left(N/\sqrt{\epsilon}\right)$ and can be improved to $\mathcal{O}(n^2\log(1/\epsilon)$ if $\kappa < \infty$. Next we consider the time complexity for the SOCP method, which consists of the time complexity of formulating the matrix A, the spectral decomposition and the IPM for solving the SOCP. Since the spectral decomposition and the IPM can not benefit much from the data sparsity, we do not distinguish the sparse and dense cases for the SOCP method. Particularly, the cost of formulating the matrix A is lower bounded by $\mathcal{O}(N)$ and upper bounded by $\mathcal{O}(n^2)$ and the spectral decomposition takes $O\left(n^\omega\right)$ flops for some ω satisfying $2 < \omega < 3$ (Demmel et al., 2007). Meanwhile, the iteration complexity for solving the SOCP reformulation is $\mathcal{O}(\sqrt{n}\log(1/\epsilon))$ according to Monteiro & Tsuchiya (2000). As per iteration in cost in the IPM is $\mathcal{O}(n)$, the total cost of the IPM is $\mathcal{O}\left(n^{\frac{3}{2}}\log(1/\epsilon)\right)$. Therefore the worst case complexity of the SOCP method is $\mathcal{O}\left(n^w + n^{\frac{3}{2}}\log(1/\epsilon)\right)$.

Theortically, it is hard to compare the Krylov subspace method and the SOCP method as the result depends on κ, N, w and ϵ. In practice, it usually holds that $\kappa < \infty$ and the spectral decomposition step in the SOCP methods usually costs $\mathcal{O}(n^3)$. In fact, our experiments show that the spectral decomposition step often needs more time than the IPM for the SOCP. Thus, the Krylov subspace method, which can effectively utilize the data sparsity, is much faster than the SOCP approach especially for solving large-scale problems.

5. Experiment Results

In this section, we present numerical results on both synthetic and real-world datasets to verify the superiority of our proposed reformulation in terms of computational costs. We refer a nested Lanczos Method for TRS (LTRSR) (Zhang & Shen, 2018) to perform the GLTR method, and use the implementation of Riemannian trust-region Newton (RTRNewton) (Absil et al., 2007) from Riemannian Manifold Optimization Library (ROPTLIB) (Huang et al., 2018). Similar as the setting in Wang et al. (2021b), we compare the running time of above two methods with the SDP and SOCP approaches in Wang et al. (2021b), averaged over 10 trials, to evaluate the performance of our new reformulation. All the four methods solve the SDP, SOCP or the SCLS reformulations to their default precision and the solutions to the SPG-LS are recovered accordingly.

All simulations are implemented using MATLAB R2021b on a PC running Windows 10 Intel(R) Xeon(R) E5-2650 v4 CPU (2.2GHz) and 64GB RAM. We report the results of two real datasets and six synthetic datasets and defer other results to the supplementary material. In all following tests, the parameter γ is set as 0.1.

5.1. Real-world Dataset

We first compare four methods on the red wine dataset (Cortez et al., 2009), which consist of 1599 instances each with 11 features. The output label is a physiochemical measurement ranged from 0 to 10, where a higher score means that the corresponding wine sample has better quality. Wine producers would like to manipulate the data to fool the learner to predict a higher score when the raw label is

smaller than some threshold t. We consider the case that there are two kinds of providers $\mathcal{A}_{\text{modest}}$ and $\mathcal{A}_{\text{severe}}$, where the details of the manipulation are set the same as Wang et al. (2021b).

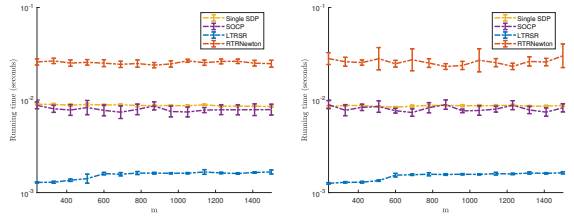

Figure 1. Comparison of four different algorithms on the red wine dataset. The left and right plots correspond to $\mathcal{A}_{\text{modest}}$ and $\mathcal{A}_{\text{severe}}$, respectively.

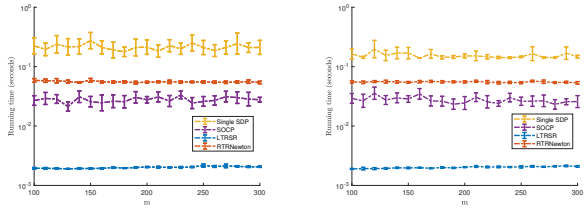

Figure 2. Comparison of four different algorithms on the residential building dataset. The left and right plots correspond to $\mathcal{A}_{\text{modest}}$ and $\mathcal{A}_{\text{severe}}$, respectively.

We also compare four methods on the residental building dataset[3] from the UCI data repository (Dua & Graff, 2017) as well. The building dataset includes 372 samples with 107 features. Each feature reflects information of a certain session such as project date, physical and financial elements. The output label is the sale prices to real estate single-family residential apartments. We consider a scenario where the seller wants to manipulate the price to higher level. As a buyer, our task is to predict the fair price under fake data. We still consider two types of sellers: $\mathcal{A}_{\text{modest}}$ with $\delta = 20$ and $\mathcal{A}_{\text{severe}}$ with $\delta = 40$.

The computational time for both datasets is reported in Figures 1 and 2. It show that LTRSR method outperforms others and follows by the SOCP and RTRNewton. One can also observe that RTRNewton is even more expensive than the SDP approach in Figure 1. The main reason is that in the red wine dataset, the number of features n is quite small ($n = 11$). Thus, the spectral decomposition step, as well as the iterations of the interior point method, in the SOCP approach is cheap.

We then report the relative errors of objective values (MSEs) of the SOCP method and our methods in Table 1 for red wine and residental building datasets. Indeed, all the methods have a very high accuracy as the relative errors are only up to `3.37e-5`. More MSE comparisons can be found in Appendix 2.1.

[3] https://archive.ics.uci.edu/ml/datasets/Residential+Building+Data+Set

Table 1. Relative error of objective values

Dataset	$(f_{SOCP} - f_{LTRSR})/\lvert f_{SOCP}\rvert$			$(f_{SOCP} - f_{RTRNew})/\lvert f_{SOCP}\rvert$		
	AVG	MIN	MAX	AVG	MIN	MAX
Wine Modest	6.17E-10	3.94E-12	4.23E-09	-8.26E-10	-4.66E-09	3.62E-09
Wine Severe	1.32E-10	3.39E-12	1.84E-09	-8.30E-11	-4.07E-10	1.80E-09
Build Modest	1.49E-07	1.93E-09	5.91E-07	-2.19E-05	-3.37E-05	-1.28E-05
Build Severe	3.02E-08	4.02E-10	1.25E-07	-1.96E-06	-3.06E-06	-1.14E-06

5.2. Synthetic Dataset

From the previous subsection, we also see that SOCP, LTRSR and RTRNewton are much faster than the SDP approach. To have a comprehensive comparison on wall-clock time among SOCP, LTRSR and RTRNewton, we test these methods on synthetic experiments.

5.2.1. Dense Data

Table 2. Time (seconds) on synthetic data without sparsity

m	n	SOCP (eig time)	RTRNew	LTRSR	Ratio
		$m = 2n$			
2000	1000	0.585 (0.064)	0.743	**0.034**	17
4000	2000	1.957 (0.317)	2.459	**0.177**	11
8000	4000	10.693 (2.758)	9.269	**0.931**	11
12000	6000	29.304 (9.444)	18.824	**2.120**	14
16000	8000	58.561 (21.634)	40.711	**3.982**	15
20000	10000	114.376 (49.754)	59.768	**6.099**	19
		$m = n$			
1000	1000	0.454 (0.065)	0.594	**0.017**	27
2000	2000	2.104 (0.325)	2.600	**0.097**	22
4000	4000	10.795 (2.698)	6.958	**0.478**	23
6000	6000	28.391 (9.481)	17.835	**1.083**	26
8000	8000	55.263 (21.555)	35.510	**2.011**	27
10000	10000	97.383 (40.091)	58.009	**3.065**	32
		$m = 0.5n$			
500	1000	0.536 (0.059)	0.523	**0.022**	24
1000	2000	1.748 (0.309)	1.432	**0.057**	31
2000	4000	9.928 (2.526)	6.932	**0.251**	40
3000	6000	27.230 (8.848)	19.179	**0.532**	51
4000	8000	54.174 (20.258)	28.953	**0.928**	58
5000	10000	94.548 (37.771)	52.756	**1.558**	61

We first conduct experiments on dense synthetic dataset. To

have better validation of the effectiveness of our proposed reformulation, we use the same artificial dataset in Wang et al. (2021b), which employs `make_regression` function in scikit-learn (Pedregosa et al., 2011) with setting the noise as 0.1 and other parameters as default.

Table 2 summarises the comparison of time on different scales with $m = ln,\ l \in \{0.5, 1, 2\}$. Here, "SOCP" represents total time needed for the SOCP approach (including "eig time"), "eig time" represents the spectral decomposition time in the SOCP approach, "RTRNew" represents the RTRNewton method, "LTRSR" represents the LTRSR method, "Ratio" represents the ratio of times of SOCP method and LTRSR.

From Table 2, we find that in large scale setting, the two methods RTRNewton and LTRSR are more efficient. LTRSR is of orders faster than the other two methods. From the "Ratio" values we also see that the time cost of SOCP approach is several tens times of that of LTRSR. We also observe that the spectral decomposition time in formulating SOCP is expensive and takes about 40% of total time. Indeed, the spectral decomposition time becomes much larger as n increases, which is also evidenced from our experiments for the sparse data setting in Table 3 below.

Table 3. Time (seconds) on synthetic data with sparsity

sparsity = 0.01					
m	n	SOCP (eig time)	RTRNew	LTRSR	Ratio
5000	10000	71.601 (39.432)	13.124	**0.225**	318
7500	15000	217.529 (120.456)	26.551	**0.534**	407
10000	20000	513.751 (288.490)	47.411	**1.049**	490
12500	25000	941.394 (539.619)	69.421	**1.606**	586
15000	30000	1539.443 (865.813)	113.223	**2.416**	637
sparsity = 0.001					
m	n	SOCP (eig time)	RTRNew	LTRSR	Ratio
5000	10000	61.587 (45.253)	1.416	**0.028**	2200
7500	15000	153.075 (117.389)	2.379	**0.053**	2888
10000	20000	335.956 (259.671)	5.453	**0.113**	2973
12500	25000	638.175 (491.391)	7.715	**0.168**	3799
15000	30000	1082.261 (832.413)	12.090	**0.235**	4605
sparsity = 0.0001					
m	n	SOCP (eig time)	RTRNew	LTRSR	Ratio
5000	10000	49.869 (45.462)	0.391	**0.009**	5541
7500	15000	141.507 (134.525)	0.716	**0.014**	10108
10000	20000	310.991 (289.447)	0.979	**0.020**	15550
12500	25000	587.124 (540.314)	1.301	**0.030**	19571
15000	30000	991.070 (912.015)	2.171	**0.037**	26786

5.2.2. Sparse Data

To further show the efficacy of our proposed reformulation, we conduct experiments on synthetic data with high feature dimension and various sparsity. We apply the `sprandn` function in MATLAB to obtain the data matrix $X \in \mathbb{R}^{m\times n}$, whose i-th row is input vector $\{\mathbf{x}_i\}_{i=1}^m$. The noise measurements $\{\xi_i\}_{i=1}^m$ i.i.d from the uniform distribution $[0, 0.5]$. Then the output label $\{y_i\}_{i=1}^m$ via $y_i = \mathbf{x}_i^T\beta + \xi_i$. Following Wang et al. (2021b), we set the fake output label as $z_i = \max\{y_i, y_{0.25}\}$.

Table 3 summarises time comparisons on synthetic datasets with different sparsity and different dimension for $m = 0.5n$. From these tables, we find that LTRSR and RTRNewton perform much better than the dense case, and their superiority over the SOCP approach becomes larger. This is mainly because both methods are the matrix free methods that require only matrix vector products in each iteration. However, the SOCP do not benefit from sparsity as well as the other two methods. We find that, by comparing the "eig time" for different instances with the same dimension but different sparsity, the "eig time" dominates the time of SOCP approach as the spectral decomposition cannot utilize the sparsity of the data. From the "Ratio" values, we find that the outperformance of LTRSR grows considerably when the sparsity and problem size increase. In the case of $(m, n) = (15000, 30000)$ and sparsity = 0.0001, LTRSR takes up to 26,000+ times faster than the SOCP approach. Moreover, our LTRSR takes less than 0.05 second for all the instances with sparsity = 0.0001. More reports on relative errors of all methods are reported in Appendix 2.2.

6. Conclusion

We propose an SCLS reformulation for the SPG-LS and show its optimal solution can be used to recover an optimal solution of the SPG-LS. We further show that an ϵ optimal solution of the SPG-LS can be also recovered from an ϵ optimal solution of the SCLS. Moreover, such an ϵ optimal solution obtained in runtime $\tilde{\mathcal{O}}(N/\sqrt{\epsilon})$. We also introduce two practical efficient methods, LTRSR and RTRNewton, for solving the SCLS. Experiments show that the SCLS approach is much faster than the existing best approach. In particular, the performance of the LTRSR dominates both RTRNewton and SOCP methods.

Acknowledgements

Wen Huang is partly supported by the Fundamental Research Funds for the Central Universities 20720190060 and NSFC 12001455. Rujun Jiang is partly supported by NSFC 12171100 and 72161160340, and Natural Science Foundation of Shanghai 22ZR1405100. Xudong Li is partly supported by the "Chenguang Program" by Shanghai Education Development Foundation and Shanghai Municipal Education Commission 19CG02, and the Shanghai Science and Technology Program 21JC1400600.

References

Absil, P.-A., Baker, C. G., and Gallivan, K. A. Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, Princeton, NJ, 2008. ISBN 9780691132983.

Bishop, N., Tran-Thanh, L., and Gerding, E. Optimal learning from verified training data. In *Advances in Neural Information Processing Systems 33*, pp. 9520–9529. NeurIPS, 2020.

Boumal, N., Absil, P.-A., and Cartis, C. Global rates of convergence for nonconvex optimization on manifolds. *IMA Journal of Numerical Analysis*, 39(1):1–33, 2019.

Brückner, M. and Scheffer, T. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 547–555, 2011.

Carmon, Y. and Duchi, J. C. Analysis of krylov subspace solutions of regularized nonconvex quadratic problems. In *Advances in Neural Information Processing Systems*, pp. 10728–10738, 2018.

Conn, A. R., Gould, N. I., and Toint, P. L. *Trust region methods*. SIAM, 2000.

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4): 547–553, 2009.

Demmel, J., Dumitriu, I., and Holtz, O. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Gould, N. I., Lucidi, S., Roma, M., and Toint, P. L. Solving the trust-region subproblem using the lanczos method. *SIAM Journal on Optimization*, 9(2):504–525, 1999.

Hazan, E. and Koren, T. A linear-time algorithm for trust region problems. *Mathematical Programming*, 158(1): 363–381, 2016.

Ho-Nguyen, N. and Kilinc-Karzan, F. A second-order cone based approach for solving the trust-region subproblem and its variants. *SIAM Journal on Optimization*, 27(3): 1485–1512, 2017.

Huang, W., Absil, P.-A., Gallivan, K. A., and Hand, P. Roptlib: an object-oriented c++ library for optimization on riemannian manifolds. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):1–21, 2018.

Jeroslow, R. G. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2):146–164, 1985.

Jiang, R. and Li, D. Novel reformulations and efficient algorithms for the generalized trust region subproblem. *SIAM Journal on Optimization*, 29(2):1603–1633, 2019.

Jiang, R. and Li, D. A linear-time algorithm for generalized trust region subproblems. *SIAM Journal on Optimization*, 30(1):915–932, 2020.

Monteiro, R. D. and Tsuchiya, T. Polynomial convergence of primal-dual algorithms for the second-order cone program based on the mz-family of directions. *Mathematical programming*, 88(1):61–83, 2000.

Moré, J. J. Generalizations of the trust region problem. *Optimization methods and Software*, 2(3-4):189–209, 1993.

Moré, J. J. and Sorensen, D. C. Computing a trust region step. *SIAM Journal on scientific and statistical computing*, 4(3):553–572, 1983.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Pong, T. K. and Wolkowicz, H. The generalized trust region subproblem. *Computational optimization and applications*, 58(2):273–322, 2014.

Sherman, J. and Morrison, W. J. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21 (1):124–127, 1950.

Shokri, R., Theodorakopoulos, G., Troncoso, C., Hubaux, J.-P., and Le Boudec, J.-Y. Protecting location privacy: optimal strategy against localization attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 617–627, 2012.

Wahab, O. A., Bentahar, J., Otrok, H., and Mourad, A. A stackelberg game for distributed formation of business-driven services communities. *Expert Systems with Applications*, 45:359–372, 2016.

Wang, A. L. and Kılınç-Karzan, F. The generalized trust region subproblem: solution complexity and convex hull results. *Mathematical Programming*, pp. 1–42, 2020.

Wang, A. L., Lu, Y., and Kilinc-Karzan, F. Implicit regularity and linear convergence rates for the generalized trust-region subproblem. *arXiv preprint arXiv:2112.13821*, 2021a.

Wang, J., Chen, H., Jiang, R., Li, X., and Li, Z. Fast algorithms for stackelberg prediction game with least squares loss. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10708–10716. PMLR, 2021b.

Zhang, L.-H. and Shen, C. A nested lanczos method for the trust-region subproblem. *SIAM Journal on Scientific Computing*, 40(4):A2005–A2032, 2018.

Zhang, L.-H., Shen, C., and Li, R.-C. On the generalized lanczos trust-region method. *SIAM Journal on Optimization*, 27(3):2110–2142, 2017.

Zhang, L.-H., Yang, W. H., Shen, C., and Feng, J. Error bounds of the lanczos approach for the trust-region subproblem. *Frontiers of Mathematics in China*, 13(2): 459–481, 2018.

Zhou, Y. and Kantarcioglu, M. Modeling adversarial learning as nested stackelberg games. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 350–362. Springer, 2016.

Zhou, Y., Kantarcioglu, M., and Xi, B. A survey of game theoretic approach for adversarial machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1259, 2019.

Appendix

A. Deferred proof for Theorem 4.2

Proof. We only need to verify the assumptions of Theorem 7.4.4 and Theorem 7.4.11 of Absil et al. (2008).

By definition, the function q is bounded below. Since the unit sphere is a compact manifold, we have that the Riemannian Hessian $\mathrm{Hess}\, q(\mathbf{r})$ is bounded above for any $\mathbf{r} \in \mathcal{S}^n$, that $q \circ R$ is radially Lipschitz continuously differentiable, and that q is Lipschitz continuously differentiable by Proposition 7.4.5 and Corollary 7.4.6 of Absil et al. (2008). The implementation in ROPTLIB for the subproblem (17) is Algorithm 11 of Absil et al. (2008) with $\theta = 1$. Therefore, the Cauchy decrease inequality is satisfied. It follows that all the assumptions of Theorem 7.4.4 are satisfied.

Since Retraction (20) is second order, the assumption of (7.36) in Theorem 7.4.11 of Absil et al. (2008) holds with $\beta_{\mathcal{H}} = 0$. Since the function q is C^∞ and the manifold is compact, the assumption of (7.37) in Theorem 7.4.11 of Absil et al. (2008) holds. The local quadratic convergence rate in (21) thus follows. □

B. Additional Experiments

In this section, we provide additional numerical results to further show the efficiency of our proposed reformulation.

1. Real-world Dataset

We demonstrate the speed of our methods on two other real-world datasets, the insurance dataset[4] and the blogfeedback dataset[5]. Similar as the setting in previous section, we compare the wall-clock time of RTRNewton (Absil et al., 2007) and LTRSR (Zhang & Shen, 2018) approaches with the SDP and SOCP methods proposed in Wang et al. (2021b).

We still apply four methods on the insurance dataset that has 1,338 instances with 7 features. Each feature shows information on certain aspects such as age, sex, bmi and region. The output labels are individual medical costs by buying health insurance. For model accuracy, we transform categorical features such as sex into a one-hot vector. We assume that the individuals incline to modify self-related data to reduce their insurance costs. Formally, the individual's desired outcome can be defined as $z_i = \max\{y_i + \delta,\ 0\}$. We have two types of individual: $\mathcal{A}_{\text{modest}}$ with $\delta = -100$ and $\mathcal{A}_{\text{severe}}$ with $\delta = -300$. All the hyperparameters are the same as those in Wang et al. (2021b). As an insurer, our goal is to select a good price model to predict the insurance costs as true as possible.

To further illustrate the effectiveness of our new reformulation, we compare four methods on the blogfeedback dataset. The blog dataset contains 52,397 samples each with 281 features processed from raw feedback materials on the Internet. Each feature represents information of a certain session. The output label is the number of comments. As a learner, our task is to predict the future comments of blog. Similarly, we assume that the true output label $\mathbf{y}$ would be modified by data providers in order to achieve the goal of increasing blog comments. For example, public media intend to manipulate data to add the blog comments and enhance its news popularity. Formally, we define the altered label as $z_i = y_i + \delta$. We still have two types of data providers: $\mathcal{A}_{\text{modest}}$ with $\delta = 5$ and $\mathcal{A}_{\text{severe}}$ with $\delta = 10$.

The wall-clock time comparison can be found in Figure 3. Similar to the previous cases, LTRSR outperforms other approaches. However, as the dimension in this problem is too small, the comparison of the other three methods does not match their performance for large scale problems.

2. Synthetic Dataset

To further demonstrate the efficiency and adaptiveness to high dimension problems of our proposed reformulation, we conduct experiments on more synthetic datasets with different hyperparameters and various dimensions and sparsity.

[4] https://www.kaggle.com/mirichoi0218/insurance
[5] https://archive.ics.uci.edu/ml/datasets/BlogFeedback

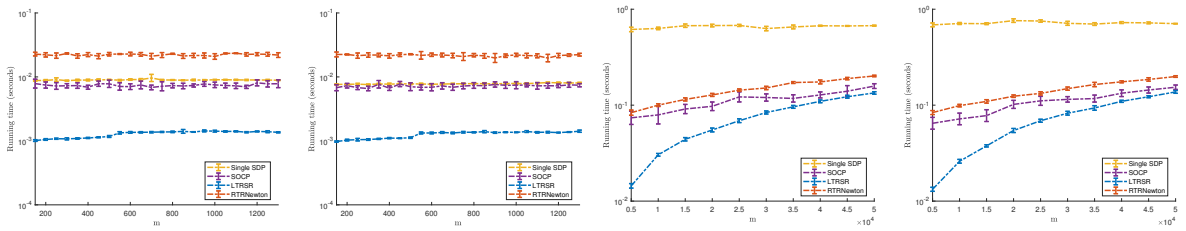

Figure 3. Performance comparison between different algorithms on the insurance and blog dataset. The left two plots correspond to the wall-clock time comparison of insurance dataset generated by $\mathcal{A}_{\text{modest}}$ and $\mathcal{A}_{\text{severe}}$, whilst the right two plots correspond to the wall-clock time comparison of blog dataset generated by $\mathcal{A}_{\text{modest}}$ and $\mathcal{A}_{\text{severe}}$.

2.1. Dense Data

We first perform experiments with $\gamma = 0.01$ on synthetic datasets without sparsity to show the superiority of our reformulation under different hyperparameter. All the other settings are the same as in Section 5.2.1.

Table 4 shows the comparison of wall-clock time on different scales with $m = ln,\ l \in \{0.5, 1, 2\}$, which is similar to the case of $\gamma = 0.1$. We observed that both SCLS approaches are faster than the SOCP approach. Moreover, all the LTRSR is less than 6 seconds while the SOCP can take up to about 110 seconds.

Table 4. Time (seconds) on synthetic data without sparsity, $\gamma = 0.01$

n	$m = 2n$				$m = n$				$m = 0.5n$			
	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio
1000	0.619 (0.087)	0.712	**0.031**	20	0.564 (0.086)	0.704	**0.020**	28	0.466 (0.078)	0.649	**0.012**	39
2000	2.244 (0.419)	2.519	**0.138**	16	1.900 (0.471)	1.828	**0.098**	19	2.438 (0.401)	2.092	**0.060**	41
4000	12.123 (3.448)	5.179	**0.956**	13	11.597 (3.539)	6.789	**0.499**	23	11.645 (3.212)	6.778	**0.249**	47
6000	33.093 (11.903)	15.857	**2.135**	16	31.262 (11.691)	18.617	**1.053**	30	32.812 (11.008)	11.070	**0.561**	58
8000	66.816 (27.466)	38.501	**3.768**	18	63.983 (27.512)	34.655	**1.984**	32	59.725 (25.061)	27.201	**0.961**	62
10000	118.044 (49.477)	59.551	**5.916**	20	109.516 (50.018)	54.060	**3.048**	36	104.251 (47.261)	39.611	**1.529**	68

2.2. Sparse Data

We proceed to perform experiments on large-scale sparse dataset of various scales $m = ln,\ l \in \{1, 2, 3\}$ with different sparsity. All the other settings are the same as in Section 5.2.2.

From Table 5, we observed the great superiority of our SCLS reformulation since all the LTRSR and RTRNewton method faster than SOCP method. LTRSR is of several orders faster than the SOCP approach, especially when the sparsity and dimension grow. The spectral time of decomposition in formulating SOCP is quite expensive as the problem size grows. In the case $(m, n) = (90000, 30000)$, sparsity $= 0.01$, the decomposition time is up about 1000 seconds, while the LTRSR method only takes about 22 seconds to solve the problem.

C. Relative Error

This section reports relative errors of function values and MSEs between SOCP and our methods for all our experiments in Tables 6, 7, 8 and 9.

1. Real-world Dataset

In Tables 6 and 7, we show the relative errors of MSEs on training sets and test sets, respectively. Here, abbreviations "Insur" represents insurance dataset, "Build" represents building dataset, "f" represents the function value of related methods and "M" represents its MSE. In Table 6, We observe that LTRSR is more accurate than SOCP as its relative error preserves positive. Indeed, all the methods have a very high accuracy as the relative errors are only up to to `3.37e-5`. Table 7 shows that all methods have similar test accuracy as the relative errors of the test MSEs of all the methods are up to `5.27e-4`.

Table 5. Time (seconds) on synthetic data with different sparsity

$m = n$

n	sparsity = 0.01				sparsity = 0.001				sparsity = 0.0001			
	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio
10000	84.547 (40.602)	21.947	**0.487**	174	56.101 (40.322)	1.892	**0.061**	920	41.854 (38.038)	0.499	**0.012**	3488
15000	254.85 (126.365)	50.797	**1.142**	223	160.486 (125.064)	5.166	**0.131**	1225	130.071 (118.582)	0.540	**0.018**	7226
20000	550.191 (281.018)	107.358	**2.088**	264	355.746 (278.687)	10.932	**0.222**	1603	299.496 (265.987)	1.075	**0.029**	10327
25000	1090.231 (581.289)	159.592	**3.502**	311	728.118 (560.429)	15.543	**0.357**	2040	591.172 (509.912)	1.285	**0.038**	15557
30000	1726.133 (963.081)	248.325	**5.393**	320	1240.319 (979.66)	19.947	**0.528**	2349	1040.441 (888.321)	2.323	**0.056**	18579

$m = 2n$

n	sparsity = 0.01				sparsity = 0.001				sparsity = 0.0001			
	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio
10000	108.861 (48.901)	50.508	**1.096**	99	64.521 (48.604)	5.765	**0.119**	542	45.599 (39.977)	0.326	**0.016**	2850
15000	316.424 (151.406)	117.217	**2.609**	121	173.231 (132.098)	12.230	**0.257**	674	148.608 (127.759)	0.947	**0.028**	5307
20000	643.464 (324.073)	219.622	**5.056**	127	379.145 (289.701)	23.199	**0.457**	830	338.119 (283.882)	1.241	**0.048**	7044
25000	1149.663 (605.581)	395.202	**7.818**	147	720.837 (561.652)	43.718	**0.725**	994	684.654 (563.812)	2.254	**0.061**	11224
30000	2026.088 (1080.883)	598.468	**12.022**	169	1217.779 (937.93)	45.074	**1.100**	1107	1106.028 (899.099)	3.869	**0.083**	13326

$m = 3n$

n	sparsity = 0.01				sparsity = 0.001				sparsity = 0.0001			
	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio	SOCP (eig time)	RTRNew	LTRSR	Ratio
10000	113.978 (51.388)	76.183	**1.592**	72	56.434 (41.871)	8.557	**0.171**	330	46.739 (39.697)	0.446	**0.022**	2125
15000	298.498 (143.174)	200.29	**3.932**	76	171.902 (129.377)	22.335	**0.396**	434	150.666 (123.946)	0.774	**0.037**	4072
20000	596.182 (288.772)	356.849	**7.750**	77	417.155 (312.681)	42.438	**0.718**	581	339.114 (274.737)	1.713	**0.061**	5559
25000	1152.428 (606.097)	614.192	**13.592**	85	782.197 (587.333)	43.646	**1.151**	680	641.319 (518.51)	2.678	**0.085**	7545
30000	1949.035 (1039.845)	948.449	**21.749**	90	1298.734 (971.861)	93.219	**1.698**	765	1085.053 (876.409)	5.526	**0.119**	9118

Table 6. Relative error of objective values on training sets

Dataset	$(f_{SOCP} - f_{LTRSR})/\lvert f_{SOCP}\rvert$			$(f_{SOCP} - f_{RTRNew})/\lvert f_{SOCP}\rvert$			**Dataset**	$(f_{SOCP} - f_{LTRSR})/\lvert f_{SOCP}\rvert$			$(f_{SOCP} - f_{RTRNew})/\lvert f_{SOCP}\rvert$		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**		**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
Wine Modest	6.17E-10	3.94E-12	4.23E-09	-8.26E-10	-4.66E-09	3.62E-09	**Insur Modest**	1.01E-05	1.40E-06	3.31E-05	1.01E-05	1.40E-06	3.31E-05
Wine Severe	1.32E-10	3.39E-12	1.84E-09	-8.30E-11	-4.07E-10	1.80E-09	**Insur Severe**	2.57E-06	4.90E-07	9.56E-06	2.57E-06	4.90E-07	9.56E-06
Build Modest	1.49E-07	1.93E-09	5.91E-07	-2.19E-05	-3.37E-05	-1.28E-05	**Blog Modest**	8.07E-09	3.33E-10	3.82E-08	8.07E-09	3.33E-10	3.82E-08
Build Severe	3.02E-08	4.02E-10	1.25E-07	-1.96E-06	-3.06E-06	-1.14E-06	**Blog Severe**	3.55E-08	2.80E-10	2.24E-07	3.55E-08	2.80E-10	2.24E-07

Table 7. Relative error of MSEs on test sets

Dataset	$(M_{SOCP} - M_{LTRSR})/\lvert M_{SOCP}\rvert$			$(M_{SOCP} - M_{RTRNew})/\lvert M_{SOCP}\rvert$			**Dataset**	$(M_{SOCP} - M_{LTRSR})/\lvert M_{SOCP}\rvert$			$(M_{SOCP} - M_{RTRNew})/\lvert M_{SOCP}\rvert$		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**		**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
Wine Modest	1.99E-07	-9.55E-07	4.31E-06	1.74E-07	-9.54E-07	3.99E-06	**Insur Modest**	1.05E-05	7.69E-07	4.21E-05	1.02E-05	-7.09E-07	4.28E-05
Wine Severe	2.03E-07	-1.79E-07	2.57E-06	2.64E-07	-4.55E-07	2.79E-06	**Insur Severe**	2.43E-06	-3.36E-07	1.01E-05	2.45E-06	-3.27E-07	1.01E-05
Build Modest	-4.52E-06	-1.25E-04	1.23E-04	-1.25E-04	-5.27E-04	5.01E-04	**Blog Modest**	-2.59E-09	-5.66E-07	2.99E-07	1.92E-09	-5.62E-07	3.03E-07
Build Severe	2.65E-07	-2.63E-05	2.54E-05	8.27E-06	-8.02E-05	1.47E-04	**Blog Severe**	-2.06E-08	-8.12E-07	3.24E-07	-5.02E-09	-7.81E-07	3.06E-07

2. Synthetic Dataset

2.1. Dense Data

Table 8 summarises relative errors of objective values on synthetic datasets without sparsity. Comparing to the result in real-world dataset, we find that all the methods have high accuracy as the relative errors of the MSEs are up to `4.60e-5`.

Table 8. Relative error on synthetic dataset without sparsity

$\gamma = 0.1$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$			$\gamma = 0.01$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**		**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
$m = 2n$	1.01E-09	9.58E-11	3.41E-09	1.01E-09	9.58E-11	3.41E-09	$m = 2n$	1.51E-05	5.39E-08	4.60E-05	1.51E-05	5.39E-08	4.60E-05
$m = n$	2.26E-08	3.37E-11	1.34E-07	2.26E-08	3.36E-11	1.34E-07	$m = n$	2.11E-06	3.26E-07	7.40E-06	2.11E-06	3.26E-07	7.40E-06
$m = 0.5n$	8.27E-09	2.54E-12	4.82E-08	8.27E-09	2.05E-12	4.82E-08	$m = 0.5n$	2.21E-06	3.44E-07	6.87E-06	2.21E-06	3.44E-07	6.87E-06

2.2. SPARSE DATA

Table 9 summarises relative error of objective values in synthetic dataset with different sparsity. Similar to the previous cases, both of our methods have high accuracy in terms of MSEs. These consistent results further prove the validity of our SCLS reformulation.

Table 9. Relative error on synthetic dataset without sparsity

sparsity = 0.0001

$\gamma = 0.1$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$			$\gamma = 0.01$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**		**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
$m = 3n$	1.24E-09	1.48E-11	5.58E-09	1.24E-09	1.45E-11	5.58E-09	$m = 3n$	3.98E-09	5.27E-11	1.19E-08	3.97E-09	3.19E-11	1.19E-08
$m = 2n$	7.49E-11	3.53E-13	1.55E-10	7.47E-11	3.20E-13	1.54E-10	$m = 2n$	2.25E-09	2.62E-12	4.88E-09	2.24E-09	-3.51E-12	4.87E-09
$m = n$	9.10E-10	9.93E-12	2.01E-09	9.10E-10	9.73E-12	2.01E-09	$m = n$	7.95E-09	1.83E-10	2.34E-08	7.94E-09	1.16E-10	2.34E-08
$m = 0.5n$	3.06E-10	3.81E-12	6.60E-10	3.06E-10	3.70E-12	6.60E-10	$m = 0.5n$	3.07E-09	7.96E-13	6.28E-09	3.06E-09	-4.52E-12	6.28E-09

sparsity = 0.001

$\gamma = 0.1$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$			$\gamma = 0.01$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**		**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
$m = 3n$	1.36E-09	5.73E-11	2.58E-09	1.36E-09	5.64E-11	2.58E-09	$m = 3n$	1.92E-08	5.04E-10	8.89E-08	1.92E-08	5.04E-10	8.89E-08
$m = 2n$	1.66E-09	6.00E-11	5.06E-09	1.66E-09	5.91E-11	5.06E-09	$m = 2n$	5.14E-08	9.80E-12	2.05E-07	5.14E-08	9.74E-12	2.05E-07
$m = n$	5.76E-10	1.53E-09	1.13E-12	6.19E-09	1.53E-09	-2.03E-13	$m = n$	1.96E-08	1.14E-10	4.83E-08	1.96E-08	1.14E-10	4.83E-08
$m = 0.5n$	1.90E-09	1.02E-10	6.68E-09	1.90E-09	1.01E-10	6.68E-09	$m = 0.5n$	4.42E-08	4.91E-09	1.34E-07	4.42E-08	4.91E-09	1.34E-07

sparsity = 0.01

$\gamma = 0.1$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$			$\gamma = 0.01$	$(\mathbf{f_{SOCP}} - \mathbf{f_{LTRSR}})/\lvert\mathbf{f_{SOCP}}\rvert$			$(\mathbf{f_{SOCP}} - \mathbf{f_{RTRNew}})/\lvert\mathbf{f_{SOCP}}\rvert$		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**		**AVG**	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
$m = 3n$	2.19E-08	2.64E-11	1.70E-08	2.19E-08	2.63E-11	1.70E-08	$m = 3n$	1.05E-07	2.15E-09	2.51E-07	1.05E-07	2.15E-09	2.51E-07
$m = 2n$	8.23E-08	8.23E-08	3.74E-07	1.02E-07	4.80E-11	3.74E-07	$m = 2n$	6.94E-07	3.99E-10	3.10E-06	6.94E-07	3.99E-10	3.10E-06
$m = n$	1.27E-08	9.14E-10	3.37E-08	1.27E-08	9.14E-10	3.37E-08	$m = n$	2.21E-06	7.78E-09	1.00E-05	2.21E-06	7.78E-09	1.00E-05
$m = 0.5n$	1.90E-08	3.52E-12	4.50E-08	1.90E-08	3.45E-12	4.50E-08	$m = 0.5n$	6.52E-06	6.40E-08	2.95E-05	6.52E-06	6.40E-08	2.95E-05