

A proximal ADMM with the Broyden family for Convex Optimization Problems

Yan Gu and Nobuo Yamashita

April 2, 2019

Abstract

Alternating direction methods of multipliers (ADMM) have been well studied and effectively used in various application fields. The classical ADMM must solve two subproblems exactly at each iteration. To overcome the difficulty of computing the exact solution of the subproblems, some proximal terms are added to the subproblems. Recently, Gu and Yamashita studied a special proximal ADMM whose regularized matrix in the proximal term is generated by the BFGS update (or limited memory BFGS) at every iteration for a structured quadratic optimization problem, and reported that the numbers of iterations were almost same as those by the exact ADMM in their numerical experiments. In this paper, we propose such a proximal ADMM for more general convex optimization problems, and extend the proximal term by the Broyden family update. We also show the convergence of the proposed method under standard assumptions.

Keywords: alternating direction method, variable metric semi-proximal method, Broyden family, convex optimization.

1 Introduction

Consider the following convex optimization problem:

$$\begin{aligned} & \text{minimize} && f(x) + g(y) \\ & \text{subject to} && Ax + By = b, \\ & && x \in \mathbb{R}^{n_1}, y \in \mathbb{R}^{n_2}, \end{aligned} \tag{1.1}$$

where $f: \mathbb{R}^{n_1} \rightarrow \mathbb{R} \cup \{\infty\}$ and $g: \mathbb{R}^{n_2} \rightarrow \mathbb{R} \cup \{\infty\}$ are proper convex functions, $A \in \mathbb{R}^{m \times n_1}$, $B \in \mathbb{R}^{m \times n_2}$ and $b \in \mathbb{R}^m$.

For solving (1.1), Gabay and Mercier [13], and Glowinski and Marrocco [14] proposed the alternating direction method of multipliers (ADMM) which generates the iterative sequence via the following recursion:

$$\begin{cases} x^{k+1} = \arg \min_x \mathcal{L}_\beta(x, y^k, \lambda^k), \end{cases} \tag{1.2a}$$

$$\begin{cases} y^{k+1} = \arg \min_y \mathcal{L}_\beta(x^{k+1}, y, \lambda^k), \end{cases} \tag{1.2b}$$

$$\begin{cases} \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b), \end{cases} \tag{1.2c}$$

where $\mathcal{L}_\beta: \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \times \mathbb{R}^m \rightarrow \mathbb{R}$ is the augmented Lagrangian function for (1.1) defined by

$$\mathcal{L}_\beta(x, y, \lambda) := f(x) + g(y) - \langle \lambda, Ax + By - b \rangle + \frac{\beta}{2} \|Ax + By - b\|^2. \tag{1.3}$$

Here $\lambda \in \mathbb{R}^m$ is multipliers associated to the equality constraints and $\beta > 0$ is a penalty parameter.

The global convergence of the ADMM (1.2a)-(1.2c) can be established under very mild conditions [3].

Since the subproblems in (1.2a)-(1.2b) may be difficult to solve exactly in many applications, Eckstein [7] and He et al. [17] have considered adding proximal terms to the subproblems. Fazel et al. [10] proposed the following semi-proximal ADMM scheme:

$$\begin{cases} x^{k+1} = \arg \min_x \mathcal{L}_\beta(x, y^k, \lambda^k) + \frac{1}{2} \|x - x^k\|_T^2, & (1.4a) \\ y^{k+1} = \arg \min_y \mathcal{L}_\beta(x^{k+1}, y, \lambda^k) + \frac{1}{2} \|y - y^k\|_S^2, & (1.4b) \\ \lambda^{k+1} = \lambda^k - \alpha\beta(Ax^{k+1} + By^{k+1} - b), & (1.4c) \end{cases}$$

where $\alpha \in (0, (1 + \sqrt{5})/2)$, $\|z\|_G = \sqrt{z^\top G z}$ for $z \in \mathbb{R}^n$ and $G \in \mathbb{R}^{n \times n}$. Fazel et al. [10] proved its the global convergence if T and S are positive semidefinite. See [5, 10, 19, 26] for more details of the semi-proximal ADMM.

He et al. [17] also proposed an inexact proximal ADMM in which the parameters β , proximal terms T and S were replaced by some bounded sequences of positive definite matrices $\{H_k\}$, $\{T_k\}$ and $\{S_k\}$, respectively. The convergence properties of such algorithms have been established in [2, 15, 22].

When we apply ADMM for solving some concrete applications, we usually assume one of the subproblems is relatively easy to solve. Therefore, we suppose that y -subproblems in (1.4b) are easily solved throughout this paper. We allow the positive semidefinite matrix T in the proximal term (1.4a) to be changed at every step as in the inexact ADMM [17]. Then T depends on k , and thus we denote it by T_k . The resulting ADMM is a variable metric semi-proximal ADMM. The iterative scheme of the variable metric semi-proximal ADMM (**VMSP-ADMM**) algorithm is given as

$$\begin{cases} x^{k+1} = \arg \min_x \mathcal{L}_\beta(x, y^k, \lambda^k) + \frac{1}{2} \|x - x^k\|_{T_k}^2, & (1.5a) \\ y^{k+1} = \arg \min_y \mathcal{L}_\beta(x^{k+1}, y, \lambda^k) + \frac{1}{2} \|y - y^k\|_S^2, & (1.5b) \\ \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b). & (1.5c) \end{cases}$$

VMSP-ADMM is also closely related to the inexact ADMM, where the subproblems in (1.2) or (1.4) to be solved approximately with certain implementable criteria [4, 6, 8, 9, 17, 27]. In this paper, we assume that (1.5a) is solved exactly, and focus on how to construct T_k .

When $f(x) = \frac{1}{2}\|x\|^2$, $B = I$ and $b = 0$, Gu and Yamashita [16] proposed to construct the proximal term T_k as $T_k = B_k - M$, where $M = \nabla_{xx}^2 \mathcal{L}_\beta(x, y, \lambda) = \beta A^\top A + I$ and B_k was a certain positive definite matrix. Note that $M \succ 0$, where $V \succeq 0$ ($V \succ 0$) means that V is symmetric and positive semidefinite (positive definite). Gu and Yamashita proposed to generate B_k via the BFGS update with respect to M at every iteration. Then they showed that $B_{k+1} \succeq M$ for all k whenever $B_k \succeq M$. Therefore $T_k \succeq 0$ if the initial matrix B_0 satisfies $B_0 \succeq M$. The variable metric semi-proximal ADMM with BFGS update (**ADM-BFGS**) for this special problem was given as

$$\begin{cases} x^{k+1} = x^k + H_k (A^\top \lambda^k - \beta A^\top y^k - M x^k), & (1.6a) \\ y^{k+1} = \arg \min_y \left\{ g(y) - \langle \lambda^k, Ax^{k+1} + y \rangle + \frac{\beta}{2} \|Ax^{k+1} + y\|^2 + \frac{1}{2} \|y - y^k\|_S^2 \right\}, & (1.6b) \\ \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + y^{k+1}), & (1.6c) \end{cases}$$

where $H_k = B_k^{-1}$. Gu and Yamashita reported that the numbers of iterations were almost same as those by the exact ADMM (1.4a) in their numerical experiments.

In this paper, we extend their method to more general convex problems. In particular, we consider the following two problems.

One is formulated as

$$\begin{aligned} \text{Problem 1:} \quad & \text{minimize} && \sum_{i=1}^N f_i(A_i x) \\ & \text{subject to} && x \in \mathbb{R}^n, \end{aligned} \quad (1.7)$$

where $f_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R} \cup \{\infty\}$, $i = 1, 2, \dots, N$ are proper convex functions, and $A_i \in \mathbb{R}^{m_i \times n_i}$, $i = 1, 2, \dots, N$.

The other problem is

$$\begin{aligned} \text{Problem 2:} \quad & \text{minimize} \quad \sum_{i=1}^N f_i(x_i) \\ & \text{subject to} \quad \sum_{i=1}^N A_i x_i = b, \\ & \quad \quad \quad x_i \in \mathbb{R}^{n_i}, \quad i = 1, 2, \dots, N, \end{aligned} \tag{1.8}$$

where $n = \sum_{i=1}^N n_i$, $f_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R} \cup \{\infty\}$, $i = 1, 2, \dots, N$ are proper convex functions, $A_i \in \mathbb{R}^{m \times n_i}$, $i = 1, 2, \dots, N$ and $b \in \mathbb{R}^m$.

Now, we apply the ADMM (1.5a)-(1.5b) to the above two convex problems. To this end, we reformulate the problems as (1.1). By introducing some auxiliary variables $y_i \in \mathbb{R}^{n_i}$ ($i = 1, 2, \dots, N$), problem (1.7) can be reformulated as

$$\begin{aligned} & \text{minimize} \quad \sum_{i=1}^N f_i(y_i) \\ & \text{subject to} \quad y_i = A_i x, \quad i = 1, 2, \dots, N \\ & \quad \quad \quad x \in \mathbb{R}^n, y_i \in \mathbb{R}^{n_i}, \quad i = 1, 2, \dots, N. \end{aligned} \tag{1.9}$$

Letting $y = (y_1^\top, y_2^\top, \dots, y_N^\top)^\top$, $f(x) \equiv 0$, $g(y) = \sum_{i=1}^N f_i(y_i)$, $A = [A_1^\top, A_2^\top, \dots, A_N^\top]^\top$, $B = -I$, $b = 0$, problem (1.9) is reduced to (1.1).

Similarly, by introducing some auxiliary variables $y_i \in \mathbb{R}^{n_i}$ ($i = 1, 2, \dots, N$), problem (1.8) can be reformulated as

$$\begin{aligned} & \text{minimize} \quad \sum_{i=1}^N f_i(y_i) \\ & \text{subject to} \quad \sum_{i=1}^N A_i x_i = b, \\ & \quad \quad \quad x_i = y_i, \quad i = 1, 2, \dots, N \\ & \quad \quad \quad x_i, y_i \in \mathbb{R}^{n_i}, \quad i = 1, 2, \dots, N. \end{aligned} \tag{1.10}$$

Letting $x = (x_1^\top, x_2^\top, \dots, x_N^\top)^\top$, $y = (y_1^\top, y_2^\top, \dots, y_N^\top)^\top$, $f(x) \equiv 0$, $g(y) = \sum_{i=1}^N f_i(y_i)$, $\tilde{A} = [A_1, A_2, \dots, A_N]$, and $A = \begin{bmatrix} \tilde{A} \\ I \end{bmatrix}$, $B = \begin{bmatrix} 0 \\ -I \end{bmatrix}$, $b = 0$, problem (1.10) is also reduced to (1.1).

The main contributions of this papers are as follows. At first, inspired by the very recent work [16] which considered the proximal ADMM with the BFGS update for structured quadratic optimization problems, we further extend the proximal ADMM with the Broyden family update for two general convex problems (1.9) and (1.10). Moreover, we show the global convergence of the proposed method. We also present some numerical results of the proposed method for solving the L1 regularized logistic regression problem.

The rest of the paper is organized as follows. We describe the construction of T_k via the Broyden family update and show the details on applying the ADMM for the above two convex problems (1.7) and (1.8) in Section 2. Section 3 discusses the global convergence of the proposed method under certain flexible conditions on the proximal matrices sequence. In Section 4, we test the L1 regularized logistic regression problem to illustrate the efficiency of the proposed method. Finally, we make some concluding remarks in Section 5.

2 Proximal ADMMs with Broyden family update for two convex optimization problems

In this section, we first explain how to construct T_k via the Broyden family update, and then present concrete algorithms for two convex optimization problems (1.7) and (1.8).

2.1 Construction of the regularized matrix T_k via the Broyden family

Throughout this paper, we assume that x -subproblems (1.5a) are unconstrained quadratic programming problem, and that the Hessian matrix of the augmented Lagrangian function (1.3) is a constant matrix defined as

$$M := \nabla_{xx}^2 \mathcal{L}_\beta(x, y, \lambda).$$

Note that M is always positive semidefinite. Note also that x -subproblems for (1.9) and (1.10) become unconstrained quadratic programming problems as seen later.

The Hessian of the objective function of x -subproblem (1.5a) is $B_k = T_k + M$. Note that if $T_k = 0$, that is, we consider the standard ADMM, then $B_k = M$. In order to avoid computing the inverse of M but still has some information on M , we consider a matrix B_k that has the following two properties:

Property (i) $B_k \succeq M$;

Property (ii) B_k has some second order information on M .

Property (i) implies that $T_k = B_k - M$ is positive semidefinite, which is required for global convergence. Property (ii) is needed for rapid convergence, since $T_k \approx 0$ as $B_k \approx M$. Gu and Yamashita [16] proposed to construct B_k via the BFGS update with respect to M at every iteration. Since the BFGS update usually constructs the inverse of B_k , let $H_k = B_k^{-1}$. Using H_k , we can easily solve the x -subproblems as in (1.6a).

When $M \succ 0$, we consider the normal BFGS update with a given $s \in \mathbb{R}^n$ and $l = Ms$. Note that $s^\top l > 0$ when $s \neq 0$. Let $s_k = x^{k+1} - x^k$, $l_k = Ms_k$. Then BFGS recursions for B_{k+1}^{BFGS} and H_{k+1}^{BFGS} are given as

$$B_{k+1}^{\text{BFGS}} = B_k + \frac{l_k l_k^\top}{l_k^\top s_k} - \frac{B_k s_k s_k^\top B_k^\top}{s_k^\top B_k s_k}, \quad (2.1)$$

$$H_{k+1}^{\text{BFGS}} = \left(I - \frac{s_k l_k^\top}{s_k^\top l_k} \right) H_k \left(I - \frac{l_k s_k^\top}{s_k^\top l_k} \right) + \frac{s_k s_k^\top}{s_k^\top l_k}. \quad (2.2)$$

Besides, the DFP updates for B_{k+1}^{DFP} and H_{k+1}^{DFP} are given as

$$B_{k+1}^{\text{DFP}} = \left(I - \frac{l_k s_k^\top}{l_k^\top s_k} \right) B_k \left(I - \frac{s_k l_k^\top}{l_k^\top s_k} \right) + \frac{l_k l_k^\top}{l_k^\top s_k} \quad (2.3)$$

$$H_{k+1}^{\text{DFP}} = H_k + \frac{s_k s_k^\top}{s_k^\top l_k} - \frac{H_k l_k l_k^\top H_k^\top}{l_k^\top H_k l_k}.$$

Since $s_k^\top l_k > 0$, all matrices B_{k+1}^{BFGS} , H_{k+1}^{BFGS} , B_{k+1}^{DFP} and H_{k+1}^{DFP} are positive definite whenever $B_k, H_k \succ 0$.

Motivated by the proximal ADMM with the BFGS update, we propose an extension of the proximal ADMM with the Broyden family update [11, 12, 23, 24], that is, a linear combination of the BFGS and DFP updates for B_{k+1} as follows:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^\top B_k}{s_k^\top B_k s_k} + \frac{l_k l_k^\top}{l_k^\top s_k} + t (s_k^\top B_k s_k) v_k v_k^\top, \quad v_k = \frac{l_k}{l_k^\top s_k} - \frac{B_k s_k}{s_k^\top B_k s_k}, \quad t \in [0, 1], \quad (2.4)$$

which is equivalent to

$$B_{k+1} = (1-t) B_{k+1}^{\text{BFGS}} + t B_{k+1}^{\text{DFP}}, \quad \text{with } t \in [0, 1]. \quad (2.5)$$

Similarly, we have the Broyden family update for H_{k+1} as follows:

$$H_{k+1} = H_k - \frac{H_k l_k l_k^\top H_k}{l_k^\top H_k l_k} + \frac{s_k s_k^\top}{s_k^\top l_k} + (1-t) (l_k^\top H_k l_k) v_k v_k^\top, \quad v_k = \frac{s_k}{s_k^\top l_k} - \frac{H_k l_k}{l_k^\top H_k l_k}, \quad t \in [0, 1], \quad (2.6)$$

which yields

$$H_{k+1} = (1-t) H_{k+1}^{\text{BFGS}} + t H_{k+1}^{\text{DFP}}, \quad \text{with } t \in [0, 1]. \quad (2.7)$$

The update scheme (2.5) becomes the pure BFGS update (2.1) when $t = 0$, and becomes the pure DFP update (2.3) when $t = 1$.

Now we consider when Property (i) holds. For the pure BFGS update, Gu and Yamashita [16] showed the following useful property.

Lemma 2.1. [16, Theorem 2.2] *Let $s \in \mathbb{R}^n$ such that $s \neq 0$, and let $l = Ms$. If $H \preceq M^{-1}$, then $H^{\text{BFGS}} \preceq M^{-1}$.*

This lemma is based on Lemma 2.1 in [16]. We modify the lemma in [16] to extend Lemma 2.1 to the Broyden family.

Lemma 2.2. *Let $l \in \mathbb{R}^n$ such that $l \neq 0$. Moreover let $s = M^{-1}l$ and $\Phi = \{z \in \mathbb{R}^n \mid \langle l, z \rangle = 0\}$. Then for any $v \in \mathbb{R}^n$, there exist $c \in \mathbb{R}$ and $z \in \Phi$ such that $v = cs + z$.*

Proof. We can easily show the lemma in a way similar to the proof of [16, Lemma 2.1]. □

We can rewrite the DFP update (2.3) as

$$B^{\text{DFP}} = B - \frac{Bsl^\top + ls^\top B}{l^\top s} + \left(1 + \frac{s^\top Bs}{l^\top s}\right) \frac{ll^\top}{l^\top s}. \quad (2.8)$$

Moreover we have

$$B^{\text{DFP}}s = l = Ms. \quad (2.9)$$

Then we can obtain the following lemma for B^{DFP} .

Lemma 2.3. *Let $l \in \mathbb{R}^n$ such that $l \neq 0$, and let $s = M^{-1}l$. If $B \succeq M$, then $B^{\text{DFP}} \succeq M$.*

Proof. Let v be an arbitrary nonzero vector in \mathbb{R}^n . Let $\Phi = \{z \in \mathbb{R}^n \mid \langle l, z \rangle = 0\}$. From Lemma 2.2 there exist $c \in \mathbb{R}$ and $z \in \Phi$ such that $v = cs + z$. It then follows from (2.9) and the definition of z that

$$\begin{aligned} v^\top B^{\text{DFP}}v &= (cs + z)^\top B^{\text{DFP}}(cs + z) \\ &= c^2 s^\top l + 2cz^\top l + z^\top B^{\text{DFP}}z \\ &= c^2 s^\top l + z^\top B^{\text{DFP}}z \\ &= c^2 s^\top Ms + z^\top B^{\text{DFP}}z. \end{aligned} \quad (2.10)$$

Since $z \in \Phi$, we have

$$z^\top \left(\frac{ls^\top}{l^\top s} B \frac{sl^\top}{l^\top s} \right) z = 0, \quad z^\top \left(\frac{ls^\top}{l^\top s} B \right) z = 0 \quad \text{and} \quad \frac{z^\top ll^\top z}{l^\top s} = 0.$$

It then follows from (2.8) that

$$z^\top B^{\text{DFP}}z = z^\top Bz - 2z^\top \left(\frac{ls^\top}{l^\top s} B \right) z + z^\top \left(\frac{ls^\top}{l^\top s} B \frac{sl^\top}{l^\top s} \right) z + \frac{z^\top ll^\top z}{l^\top s} = z^\top Bz. \quad (2.11)$$

Furthermore, equation (2.9) implies

$$s^\top Mz = l^\top z = 0.$$

Combining (2.10) and (2.11), we have

$$\begin{aligned} v^\top B^{\text{DFP}}v &= c^2 s^\top Ms + z^\top Bz \\ &\geq c^2 s^\top Ms + z^\top Mz \\ &= (cs + z)^\top M(cs + z) - 2cs^\top Mz \\ &= v^\top Mv, \end{aligned}$$

where the inequality follows from the assumption. Since v is arbitrary, we have $B^{\text{DFP}} \succeq M$. □

Using the above lemmas, we can show the following desired property for the Broyden family update.

Theorem 2.4. *Let $s, l \in \mathbb{R}^n$ such that $s, l \neq 0$, and let $l = Ms$. Suppose that B_k is updated by the Broyden family (2.5) for all $k > 0$. If $B_0 \succeq M$, then $B_k \succeq M$ for any $t \in [0, 1]$ and $k > 0$.*

Proof. Due to Lemmas 2.1 and 2.3 we show that if $(H_0)^{-1} = B_0 \succeq M$, then $B_k^{\text{BFGS}} \succeq M$ and $B_k^{\text{DFP}} \succeq M$ for all $k > 0$. It follows from (2.5) that for any $t \in [0, 1]$ and $k > 0$, $B_k \succeq M$. \square

Theorem 2.4 implies that $T_k = B_k - M \succeq 0$ when the initial matrix B_0 satisfies $B_0 \succeq M$.

When M is merely positive semidefinite, we cannot directly use the BFGS and DFP updates. In this case we may consider $M^\delta := M + \delta I$ with sufficiently small $\delta > 0$, and construct B_k by the BFGS and DFP updates for M^δ . That is, B_k^{BFGS} and B_k^{DFP} are updated by the BFGS and DFP with respect to M^δ at every iteration, respectively, and $l_\delta = M_\delta s = Ms + \delta s$. Then $T_k = B_k - M \succeq \delta I \succ 0$ is a positive definite matrix. Note that $s^\top l_\delta > 0$ when $s \neq 0$, and recursions (2.2) and (2.3) still hold. When δ is small enough, B_k is also close to M .

In the subsequent subsections, we describe the details of applying the VMSP-ADMM with the Broyden family update for two convex problems (1.7) and (1.8) given in Introduction.

2.2 VMSP-ADMM for convex problem 1

First, we consider problem 1. Let $\mathcal{L}_\beta^1(x, y_1, \dots, y_N, \lambda_1, \dots, \lambda_N)$ be the augmented Lagrangian function for (1.9) that defined by

$$\mathcal{L}_\beta^1(x, y_1, \dots, y_N, \lambda_1, \dots, \lambda_N) := \sum_{i=1}^N f_i(y_i) - \sum_{i=1}^N \langle \lambda_i, A_i x - y_i \rangle + \sum_{i=1}^N \frac{\beta}{2} \|A_i x - y_i\|^2, \quad (2.12)$$

where $\lambda_i \in \mathbb{R}^{m_i}$ ($i = 1, 2, \dots, N$) are multipliers associated to the linear constraints and $\beta > 0$ is a penalty parameter. As in Introduction, we further define $y = (y_1^\top, y_2^\top, \dots, y_N^\top)^\top$, $g(y) = \sum_{i=1}^N f_i(y_i)$, $\mathcal{A}_1 = [A_1^\top, A_2^\top, \dots, A_N^\top]^\top$, $\lambda = (\lambda_1^\top, \lambda_2^\top, \dots, \lambda_N^\top)^\top$, and $m = \sum_{i=1}^N m_i$. Then $y \in \mathbb{R}^m$, $\lambda \in \mathbb{R}^m$ and the augmented Lagrangian function is rewritten as

$$\mathcal{L}_\beta^1(x, y, \lambda) := g(y) - \langle \lambda, \mathcal{A}_1 x - y \rangle + \frac{\beta}{2} \|\mathcal{A}_1 x - y\|^2. \quad (2.13)$$

Using a proximal matrix $T_k^1 \in \mathbb{R}^{n \times n}$, the x -subproblem (1.5a) for (1.9) is written as

$$\begin{aligned} x^{k+1} &= \arg \min_x \left\{ -\langle \lambda^k, \mathcal{A}_1 x - y^k \rangle + \frac{\beta}{2} \|\mathcal{A}_1 x - y^k\|^2 + \frac{1}{2} \|x - x^k\|_{T_k^1}^2 \right\} \\ &= (\beta \mathcal{A}_1^\top \mathcal{A}_1 + T_k^1)^{-1} (\mathcal{A}_1^\top \lambda^k + \beta \mathcal{A}_1^\top y^k + T_k^1 x^k). \end{aligned} \quad (2.14)$$

Note that $\beta \mathcal{A}_1^\top \mathcal{A}_1 = \sum_{i=1}^N \beta A_i^\top A_i = \nabla_{xx}^2 \mathcal{L}_\beta^1(x, y, \lambda)$. Let M^1 be defined as

$$M^1 := \nabla_{xx}^2 \mathcal{L}_\beta^1(x, y, \lambda) = \beta \mathcal{A}_1^\top \mathcal{A}_1 \succeq 0.$$

Note that M^1 is not necessarily positive definite. Then, as written in the previous subsection, we may not apply the BFGS and DFP updates for M . Therefore we use the following perturbed matrix M^{δ_1} instead of M^1 .

$$M^{\delta_1} := M^1 + \delta_1 I \succ 0 \text{ with } \delta_1 > 0.$$

Then, we propose to construct a matrix B_k^1 by the Broyden family with $s_k = x_1^{k+1} - x_1^k$ and $l_k = M^{\delta_1} s_k$, and set

$$T_k^1 = B_k^1 - M^1.$$

Note that Theorem 2.4 implies that $T_k^1 \succ 0$ if $B_0^1 \succeq M^{\delta_1}$.

Using B_k^1 , x -subproblem (2.14) is written as

$$x^{k+1} = x^k + (B_k^1)^{-1} (\mathcal{A}_1^\top \lambda^k + \beta \mathcal{A}_1^\top y^k - M^1 x^k).$$

Based on the descriptions of the BFGS (or limited memory BFGS, abbreviated to L-BFGS) and DFP updates, we first give our algorithm for problem (1.7).

Algorithm 1: VMSP-ADMM with the Broyden family update (**ADM-BD1**) for convex Problem 1 (1.7)

Input : data matrix \mathcal{A}_1 , initial point (x^0, y^0, λ^0) , penalty parameter β, δ_1 , maxIter;
initial matrix $H_0^1 \preceq (M^{\delta_1})^{-1}$, coefficient $t_1 \in [0, 1]$, constant $\bar{k}^1 \in [1, \infty]$, stopping criterion ϵ .

Output: approximative solution (x^k, y^k, λ^k)

```

1 initialization;
2 while  $k < \text{maxIter}$  or not convergence do
3   if  $k \leq \bar{k}^1$  and  $x_k - x_{k-1} \neq 0$  then
4     | update  $H_k^1$  via (2.6) or L-BFGS;
5   else
6     |  $H_k^1 = H_{k-1}^1$ ;
7   end
8   update  $x^{k+1}$  by solving the  $x$ -subproblem:  $x^{k+1} = x^k + H_k^1 (\mathcal{A}_1^\top \lambda^k + \beta \mathcal{A}_1^\top y^k - M^1 x^k)$ ;
9   update  $y^{k+1}$  by solving the  $y$ -subproblem:
      
$$y^{k+1} = \arg \min_y \left\{ g(y) - \langle \lambda^k, \mathcal{A}_1 x^{k+1} - y \rangle + \frac{\beta}{2} \|\mathcal{A}_1 x^{k+1} - y\|^2 + \frac{1}{2} \|y - y^k\|_S^2 \right\};$$

10  update Lagrangian multipliers:  $\lambda^{k+1} = \lambda^k - \beta (\mathcal{A}_1 x^{k+1} - y^{k+1})$ .
11 end
```

Remark 2.5. Note that constant $\bar{k}^1 \in [1, \infty]$ in the algorithm means that the B_k^1 (H_k^1) updated by the BFGS (or L-BFGS) and DFP procedures will be stopped at \bar{k}^1 , that is, $B_k^1 = B_{\bar{k}^1}^1$ ($H_k^1 = H_{\bar{k}^1}^1$) for $k \geq \bar{k}^1$. Specially,

- $\bar{k}^1 < \infty$: we can show the global convergence since it is reduced to the usual proximal ADMM after \bar{k}^1 steps;
- $\bar{k}^1 = \infty$: the B_k^1 (H_k^1) are updated for all k , and the numerical experiments in [16] show it works;
- \bar{k}^1 is small: it is not efficient, since $B_{\bar{k}^1}^1$ is not close to M^{δ_1} .

2.3 VMSP-ADMM for convex problem 2

For problem 2, let $\mathcal{L}_\beta^2(x_1, \dots, x_N, y_1, \dots, y_N, \lambda, \mu_1, \dots, \mu_N)$ be the augmented Lagrangian function for (1.10) that defined by

$$\begin{aligned} \mathcal{L}_\beta^2(x_1, \dots, x_N, y_1, \dots, y_N, \lambda, \mu_1, \dots, \mu_N) := & \sum_{i=1}^N f_i(y_i) - \langle \lambda, \sum_{i=1}^N A_i x_i - b \rangle - \sum_{i=1}^N \langle \mu_i, x_i - y_i \rangle \\ & + \frac{\beta_1}{2} \left\| \sum_{i=1}^N A_i x_i - b \right\|^2 + \sum_{i=1}^N \frac{\beta_2}{2} \|x_i - y_i\|^2, \end{aligned} \quad (2.15)$$

where $\lambda \in \mathbb{R}^m$, $\mu_i \in \mathbb{R}^{n_i}$ ($i = 1, 2, \dots, N$) are multipliers associated to the linear constraints and $\beta_1, \beta_2 > 0$ are the penalty parameters, respectively.

We also define $x = (x_1^\top, x_2^\top, \dots, x_N^\top)^\top$, $y = (y_1^\top, y_2^\top, \dots, y_N^\top)^\top$, $\mathcal{A}_2 = [A_1, A_2, \dots, A_N]$, $g(y) = \sum_{i=1}^N f_i(y_i)$, and $n = \sum_{i=1}^N n_i$. Then $x, y \in \mathbb{R}^n$ and $\mathcal{A}_2 \in \mathbb{R}^{m \times n}$. Moreover let $\mu = (\mu_1^\top, \mu_2^\top, \dots, \mu_N^\top)^\top \in \mathbb{R}^n$. The augmented Lagrangian function (2.15) can be written as

$$\mathcal{L}_\beta^2(x, y, \lambda, \mu) = g(y) - \langle \lambda, \mathcal{A}_2 x - b \rangle - \langle \mu, x - y \rangle + \frac{\beta_1}{2} \|\mathcal{A}_2 x - b\|^2 + \frac{\beta_2}{2} \|x - y\|^2. \quad (2.16)$$

Similar to problem 1 in Subsection 2.2, using a proximal matrix $T_k^2 \in \mathbb{R}^{n \times n}$, the x -subproblem (1.5a) for (1.10) is written as

$$\begin{aligned} x^{k+1} &= \arg \min_x \left\{ -\langle \lambda^k, \mathcal{A}_2 x - b \rangle - \langle \mu^k, x - y^k \rangle + \frac{\beta_1}{2} \|\mathcal{A}_2 x - b\|^2 + \frac{\beta_2}{2} \|x - y^k\|^2 + \frac{1}{2} \|x - x^k\|_{T_k^2}^2 \right\} \\ &= (\beta_1 \mathcal{A}_2^\top \mathcal{A}_2 + \beta_2 I + T_k^2)^{-1} (\mathcal{A}_2^\top \lambda^k + \mu^k + \beta_1 \mathcal{A}_2^\top b + \beta_2 y^k + T_k^2 x^k). \end{aligned} \quad (2.17)$$

Let M^2 be defined as

$$M^2 := \nabla_{xx}^2 \mathcal{L}_\beta^2(x, y, \lambda, \mu) = \beta_1 \mathcal{A}_2^\top \mathcal{A}_2 + \beta_2 I.$$

Note that $M^2 \succ 0$ whenever $\beta_2 > 0$. Then, we construct a matrix B_k^2 via the Broyden family for M^2 and set:

$$T_k^2 = B_k^2 - M^2.$$

Theorem 2.4 implies that $T_k^2 \succeq 0$ if $B_k^2 \succeq M^2$. Using B_k^2 , x -subproblem (2.17) is written as

$$x^{k+1} = x^k + (B_k^2)^{-1} (\mathcal{A}_2^\top \lambda^k + \mu^k + \beta_1 \mathcal{A}_2^\top b + \beta_2 y^k - M^2 x^k).$$

The algorithm for problem (1.8) is as follows:

Algorithm 2: VMSP-ADMM with the Broyden family update (**ADM-BD2**) for convex Problem 2 (1.8)

Input : data matrix \mathcal{A}_2 , initial point $(x^0, y^0, \lambda^0, \mu^0)$, penalty parameters β_1, β_2 , maxIter;
initial matrix $H_0^2 \preceq (M^2)^{-1}$, coefficient $t_2 \in [0, 1]$, constant $\bar{k}^2 \in [1, \infty]$, stopping criterion ϵ .
Output: approximative solution $(x^k, y^k, \lambda^k, \mu^k)$

```

1 initialization;
2 while  $k < \text{maxIter}$  or not convergence do
3   if  $k \leq \bar{k}^2$  and  $x_k - x_{k-1} \neq 0$  then
4     | update  $H_k^2$  via (2.6) or L-BFGS;
5   else
6     |  $H_k^2 = H_{k-1}^2$ ;
7   end
8   update  $x^{k+1}$  by solving the  $x$ -subproblem:
       $x^{k+1} = x^k + H_k^2 (\mathcal{A}_2^\top \lambda^k + \mu^k + \beta_1 \mathcal{A}_2^\top b + \beta_2 y^k - M^2 x^k)$ ;
9   update  $y^{k+1}$  by solving the  $y$ -subproblem:
       $y^{k+1} = \arg \min_y \left\{ g(y) - \langle \mu^k, x^{k+1} - y \rangle + \frac{\beta_2}{2} \|x^{k+1} - y\|^2 + \frac{1}{2} \|y - y^k\|_S^2 \right\}$ ;
10  update Lagrangian multipliers:  $\lambda^{k+1} = \lambda^k - \beta_1 (\mathcal{A}_2 x^{k+1} - b)$ ;
11  update Lagrangian multipliers:  $\mu^{k+1} = \mu^k - \beta_2 (x^{k+1} - y^{k+1})$ .
12 end

```

The constant \bar{k}^2 in the Algorithm 2 plays the same role as \bar{k}^1 in Algorithm 1.

3 Convergence of VMSP-ADMM with Broyden family update

In this section, we first consider general convergence properties for the variable metric semi-proximal ADMM (1.5a)-(1.5c) (VMSP-ADMM). Then we discuss the convergence of VMSP-ADMM with the Broyden family update (ADM-BD1 and ADM-BD2) for problem 1 and problem 2.

3.1 Global convergence of variable metric semi-proximal ADMM

Let Ω^* be a set of (x^*, y^*, λ^*) satisfying the KKT condition of problem (1.1). Since the subdifferential mapping of the closed proper convex functions are maximal monotone [25], there exist two positive semidefinite matrices

Σ_f and Σ_g such that for all $x, \hat{x} \in \mathbb{R}^n$, $f'(x) \in \partial f(x)$, and $f'(\hat{x}) \in \partial f(\hat{x})$,

$$(x - \hat{x})^\top (f'(x) - f'(\hat{x})) \geq \|x - \hat{x}\|_{\Sigma_f}^2,$$

and for all $y, \hat{y} \in \mathbb{R}^n$, $g'(y) \in \partial g(y)$, and $g'(\hat{y}) \in \partial g(\hat{y})$,

$$(y - \hat{y})^\top (g'(y) - g'(\hat{y})) \geq \|y - \hat{y}\|_{\Sigma_g}^2.$$

Throughout this paper, we make the following assumptions.

Assumption 3.1. *The set Ω^* of KKT points is non-empty.*

Assumption 3.2. *For all $k > 0$, $T_k + \Sigma_f + \beta A^\top A$ and $S + \Sigma_g + \beta B^\top B$ are positive definite.*

Now, we begin to investigate the convergence of VMSP-ADMM. First, we assume some conditions for sequence $\{T_k\}$ to guarantee the convergence.

Condition 3.1. *For the sequence $\{T_k\}$ generated by the framework (1.5), there exist $T \succeq 0$ and a non-negative sequence $\{\gamma_k\}$ such that*

- 1 $T \preceq T_{k+1} \preceq (1 + \gamma_k)T_k$, for all k ,
- 2 $T + \Sigma_f + \beta A^\top A$ is positive definite,
- 3 $\sum_0^\infty \gamma_k < \infty$.

Now we give the main theorem of this subsection.

Theorem 3.3. *Suppose that Assumptions 3.1 and 3.2 hold. Suppose also that $\{T_k\}$ is a sequence satisfying Condition 3.1. Let $\{(x^k, y^k, \lambda^k)\}$ be generated by (1.5). Then the following statements hold:*

(a) *we have for $k \geq 1$ that*

$$\begin{aligned} & \|x^k - x^*\|_{T_k}^2 + \|y^k - y^*\|_S^2 + \beta \|B(y^k - y^*)\|^2 + \frac{1}{\beta} \|\lambda^k - \lambda^*\|^2 + \|y^k - y^{k-1}\|_S^2 \\ & - \left(\|x^{k+1} - x^*\|_{T_k}^2 + \|y^{k+1} - y^*\|_S^2 + \beta \|B(y^{k+1} - y^*)\|^2 + \frac{1}{\beta} \|\lambda^{k+1} - \lambda^*\|^2 + \|y^{k+1} - y^k\|_S^2 \right) \\ & \geq \|x^{k+1} - x^k\|_{T_k}^2 + \|y^{k+1} - y^k\|_S^2 + \beta \|B(y^{k+1} - y^k)\|^2 + \beta \|Ax^{k+1} + By^{k+1} - b\|^2 \\ & + 2\|x^{k+1} - x^*\|_{\Sigma_f}^2 + 2\|y^{k+1} - y^*\|_{\Sigma_g}^2. \end{aligned}$$

(b) *sequence $\{(x^k, y^k, \lambda^k)\}$ converges to a point $(x^*, y^*, \lambda^*) \in \Omega^*$.*

Proof. The proof can be done in a way similar to the proofs in [16, Theorem 2.3] and [10, Theorem B.1]. □

3.2 Global convergences of Algorithms 1 and 2

We establish the global convergences of Algorithms 1 and 2 as corollaries of the convergence results in Subsection 3.1. Throughout this subsection, let $H_k^{B1} = (B_k^{B1})^{-1}$, $H_k^{D1} = (B_k^{D1})^{-1}$, $H_k^{B2} = (B_k^{B2})^{-1}$ and $H_k^{D2} = (B_k^{D2})^{-1}$.

For the global convergence, we require condition 3.1 holds. In particular, we need that $0 \preceq T \preceq T_k$ for all k , that is, T_k should be positive semidefinite for all k . To see this, we first note that initial matrices H_0^1 and H_0^2 satisfy $H_0^1 \preceq (M^{\delta_1})^{-1}$ and $H_0^2 \preceq (M^2)^{-1}$. Then Theorem 2.4 implies $H_k^1 \preceq (M^{\delta_1})^{-1}$ and $H_k^2 \preceq (M^2)^{-1}$ for all k , and hence $T_k^1, T_k^2 \succeq 0$ for all k .

Therefore, Algorithms 1 and 2 are well-defined, that is, they can generate a sequence $\{x^k, y^k, \lambda^k\}$. Moreover, we get the following convergence properties based on Theorem 3.3.

Theorem 3.4. Suppose that sequences $\{T_k^1\}$ satisfies Condition 3.1. Let $\{(x^k, y^k, \lambda^k)\}$ be a sequence generated by Algorithm 1. Then sequence $\{(x^k, y^k, \lambda^k)\}$ converges to a KKT point of (1.9).

Theorem 3.5. Suppose that sequences $\{T_k^2\}$ satisfies Condition 3.1. Let $\{(x^k, y^k, \lambda^k, \mu^k)\}$ be a sequence generated by Algorithm 2. Then sequence $\{(x^k, y^k, \lambda^k, \mu^k)\}$ converges to a KKT point of (1.10).

Until now, we cannot give sufficient conditions for Condition 3.1 to hold when $\{B_k\}$ is updated by a pure Broyden family. Now we provide two amendments for sequences $\{T_k^1\}$ and $\{T_k^2\}$ with the Broyden family (2.5) to guarantee Condition 3.1.

Amendment I: Let \bar{k}^1 be finite in Algorithm 1. Then the updating of B_k^1 stops at \bar{k}^1 , and thus sequence $\{T_k^1\}$ satisfies Condition 3.1. Similarly, let \bar{k}^2 be finite in Algorithm 2. Then sequence $\{T_k^2\}$ satisfies Condition 3.1 as well.

Amendment II: Suppose that $\bar{k}^1 = \infty$ in Algorithm 1. We generate $\{B_k^1\}$ as follows:

$$B_{k+1}^1 = B_k^1 + c_k^1 (\bar{B}_{k+1}^1 - B_k^1), \quad (3.1)$$

where \bar{B}_{k+1}^1 is updated by the pure Broyden family (2.4) for $M^{\delta_1} > 0$ with $\delta_1 > 0$, and $\{c_k^1\}$ is a sequence such that $c_k^1 \in [0, 1]$, and $\sum_{k=0}^{\infty} c_k^1 < \infty$. We can easily obtain that Condition 3.1 holds for the sequence $\{T_k^1\}$ as that shown in [16].

Suppose that $\bar{k}^2 = \infty$ in Algorithm 2. We generate $\{B_k^2\}$ as follows:

$$B_{k+1}^2 = B_k^2 + c_k^2 \left(\frac{\tilde{l}_k^2 (\tilde{l}_k^2)^\top}{(\tilde{l}_k^2)^\top s_k^2} - \frac{B_k^2 s_k^2 (s_k^2)^\top (B_k^2)^\top}{(s_k^2)^\top B_k^2 s_k^2} + t_2 ((s_k^2)^\top B_k^2 s_k^2) \tilde{v}_k^2 (\tilde{v}_k^2)^\top \right), \quad (3.2)$$

where $\tilde{l}_k^2 = M^{\delta_2} s_k^2 = M^2 s_k^2 + \delta_2 s_k^2$ with $\delta_2 > 0$, $t_2 \in [0, 1]$ is a scalar parameter,

$$\tilde{v}_k^2 = \left(\frac{\tilde{l}_k^2}{(\tilde{l}_k^2)^\top s_k^2} - \frac{B_k^2 s_k^2}{(s_k^2)^\top B_k^2 s_k^2} \right),$$

and $\{c_k^2\}$ is a sequence such that $c_k^2 \in [0, 1]$, and $\sum_{k=0}^{\infty} c_k^2 < \infty$.

We can rewrite (3.2) as

$$B_{k+1}^2 = B_k^2 + c_k^2 (\bar{B}_{k+1}^2 - B_k^2), \quad (3.3)$$

$$\bar{B}_{k+1}^2 = (1 - t_2) B_{k+1}^{B2} + t_2 B_{k+1}^{D2}, \quad \text{with } t_2 \in [0, 1],$$

where B_{k+1}^{B2} and B_{k+1}^{D2} are updated by the pure BFGS and DFP updates with respect to M^{δ_2} , respectively.

As shown in [16], Condition 3.1 holds for the sequence $\{T_k^2\}$.

4 Numerical Experiments for the L1 regularized logistic regression

In this section, we test the proposed proximal ADMM by solving a popular sparse learning problem, L1 regularized logistic regression model. All the experiments are implemented by Matlab R2018b on Windows 10 pro with a 2.10 GHz Intel Xeon E5-2620 v4 processor and 128 GB of RAM. The L1 regularized logistic regression model is given as:

$$\min \left\{ \frac{1}{m} \sum_{i=1}^m \log \left(1 + \exp(-r_i(A_i x + \sigma)) \right) + \rho \|x\|_1 \mid x \in \mathbb{R}^n \right\}, \quad (4.1)$$

where $A \in \mathbb{R}^{m \times n}$ is a feature matrix, $A_i \in \mathbb{R}^{1 \times n}$ is the row vector of matrix A , and $r \in \mathbb{R}^m$ is a response vector. The scalar m is the number of data points, and n is the dimension of data. Moreover, $\sigma \in \mathbb{R}$ is a decided intercept scalar, and $\rho > 0$ is a regularization parameter. The decision variable of (4.1) is $x \in \mathbb{R}^n$.

4.1 Reformulation and algorithms

Our purpose is to justify the advantages of the proximal ADMM with the Broyden family update (Algorithm 1). We choose some classical ADMMs as the benchmark for numerical comparison in this subsection.

By introducing some auxiliary variables $y_i \in \mathbb{R}$ ($i = 1, 2, \dots, m$) and $z \in \mathbb{R}^n$, the L1 regularized logistic regression model (4.1) can be reformulated as

$$\begin{aligned} & \text{minimize} && \frac{1}{m} \sum_{i=1}^m \log\left(1 + \exp(-r_i(y_i + \sigma))\right) + \rho \|z\|_1 \\ & \text{subject to} && y_i = A_i x, \quad i = 1, 2, \dots, m \\ & && z = x, \\ & && x, z \in \mathbb{R}^n, y_i \in \mathbb{R}. \end{aligned} \quad (4.2)$$

Note that, letting $\tilde{y} = \begin{pmatrix} y \\ z \end{pmatrix}$, $g(\tilde{y}) = \frac{1}{m} \sum_{i=1}^m \log\left(1 + \exp(-r_i(y_i + \sigma))\right) + \rho \|z\|_1$, $\tilde{A} = \begin{bmatrix} A \\ I_n \end{bmatrix}$, $B = -I_{m+n}$, and $b = 0$, problem (4.2) is reduced to (1.1).

The augmented Lagrangian function of (4.2) can be written as:

$$\mathcal{L}_\beta(x, y, z, \lambda, \mu) = g(\tilde{y}) - \langle \lambda, Ax - y \rangle - \langle \mu, x - z \rangle + \frac{\beta}{2} \|Ax - y\|^2 + \frac{\beta}{2} \|x - z\|^2, \quad (4.3)$$

where $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^n$ are multipliers associated to the linear constraints and $\beta > 0$ is the penalty parameter.

We further define $\tilde{\lambda} = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$. Let M be the Hessian matrix of the augmented Lagrangian function (4.3), that is,

$M := \nabla_{xx}^2 \mathcal{L}_\beta(x, y, z, \lambda, \mu) = \beta \tilde{A}^\top \tilde{A} = \beta A^\top A + \beta I$. Note that $M \succ 0$ whenever $\beta > 0$. Therefore, we set $\delta_1 = 0$ in Algorithm 1.

The maximum eigenvalue $\lambda_{\max}(\beta I + \beta A^\top A)$ is needed in some algorithms. We adopt the the following codes in Matlab compute the maximum eigenvalue $\lambda_{\max}(A^\top A)$:

$$\text{AAfun} = @(x)A'* (A*x); \quad \text{eig_max} = \text{eigs}(\text{AAfun}, n, 1);$$

The inverse of matrix M will also be used later for some x -subproblems. We use a Cholesky factorization to solve it. When $m \geq n$, we use “chol” in Matlab for $(A^\top A + \beta I)$. When $m < n$, we apply Sherman-Morrison formula to $(\beta I + A^\top A)^{-1}$:

$$(\beta I + A^\top A)^{-1} = \frac{1}{\beta} I - \frac{1}{\beta^2} \cdot A^\top \cdot \left(I + \frac{1}{\beta} A A^\top \right)^{-1} \cdot A,$$

and compute the factorization LL^\top of a smaller matrix $(I + (1/\beta)AA^\top)$ by the “chol” function. Note that the maximum eigenvalue and the Cholesky factorization are calculated only once for each test problem.

Besides, a soft-thresholding operator $S_\kappa: \mathbb{R}^n \rightarrow \mathbb{R}^n$ will be used in the y -subproblems, which is defined as $(S_\kappa(a))_i = (1 - \kappa/|a_i|)_+ \cdot a_i$, for all $i = 1, \dots, m$, $\kappa > 0$ and $a \in \mathbb{R}^m$;

We test the following methods:

ADMM-1: the classical ADMM [13, 14] which is applied for the original problem (4.1):

$$\begin{cases} x^{k+1} = \arg \min_x \frac{1}{m} \sum_{i=1}^m \log\left(1 + \exp(-r_i(A_i x + \sigma))\right) + \frac{\beta}{2} \|x - y^k - \lambda^k/\beta\|_2^2, \\ y^{k+1} = S_{\rho/\beta}(x^{k+1} - \lambda^k/\beta), \\ \lambda^{k+1} = \lambda^k - \beta(x^{k+1} - y^{k+1}); \end{cases} \quad (4.4)$$

ADMM-2: the classical ADMM [13, 14] applied for problem (4.2);

ADM-PRO: the proximal ADMM for problem (4.2) with an indefinite proximal matrix T as [18, 21]:

$$T = \xi I - \beta I - \beta A^\top A, \quad \text{with } \xi = 0.8 * \lambda_{\max}(\beta I + \beta A^\top A);$$

ADM-Broyden: the proximal ADMM with the Broyden family update for problem (4.2) with a semidefinite proximal matrix sequence $\{T_k\}$:

$$\begin{cases} x^{k+1} = x^k + (B_k)^{-1} (A^\top \lambda^k + \mu^k + \beta A^\top y^k + \beta z^k - Mx^k), \\ y^{k+1} = \arg \min_y \frac{1}{m} \sum_{i=1}^m \log(1 + \exp(-r_i(y_i + \sigma))) + \frac{\beta}{2} \|Ax^{k+1} - y - \lambda^k / \beta\|_2^2, \\ z^{k+1} = S_{\rho/\beta}(x^{k+1} - \mu^k / \beta), \\ \lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} - y^{k+1}), \\ \mu^{k+1} = \mu^k - \beta(x^{k+1} - z^{k+1}), \end{cases} \quad (4.5)$$

where B_k is generated by Broyden family (2.4), and the initial matrix B_0 is chosen as:

$$B_0 = \xi I, \text{ with } \xi = 1.01 * \lambda_{\max}(\beta I + \beta A^\top A). \quad (4.6)$$

It becomes the pure BFGS update when $t = 0$ (ADM-BFGS) and becomes the pure DFP update when $t = 1$;

ADM-BFGS (or ADM-LBFGS): the proximal ADMM with the BFGS (or L-BFGS) update for problem (4.2) with a semidefinite proximal matrix sequence $\{T_k\}$. Matrix B_k is generated by BFGS (2.8), and the initial matrix B_0 is chosen as (4.6);

ADM-ILBFGS: the proximal ADMM with the L-BFGS update for problem (4.2) with an indefinite proximal matrix sequence $\{T_k\}$ where the initial matrix B_0 is chosen as:

$$B_0 = \xi I, \xi = 0.8 * \lambda_{\max}(\beta I + \beta A^\top A).$$

Since the x -subproblems in (4.4) and y -subproblems in (4.5) have no closed-form solution, we adopt a custom Newton solver for these subproblems with the tolerance of 10^{-6} . We set a maximum iteration number of the Newton solver to 50. The codes for the original ADMM (4.4) are referred to the paper [3] and can be found at [1].

4.2 Problem data and algorithm settings

Now we specify the data for the L1 regularized logistic regression model (4.1) to be tested. We generate data by the codes of [1] as follows. We first generate $D \in \mathbb{R}^{m \times n}$ as a sparse matrix normally distributed with p sparsity nonzero entries, where $p \in (0, 1]$. Then we generate σ , r , A and ρ as follows:

- the intercept σ is chosen from $\mathcal{N}(0, 1)$;
- the vector r is generated by $r = \text{sign}(Dw + \sigma + \epsilon)$, where ϵ is the noise drawn from $\mathcal{N}(0, 0.1)$, and $w \in \mathbb{R}^n$ is a random and sparse vector with approximately 10% normally distributed nonzero entries;
- matrix A can be written as $A = \text{spdiags}(r, 0, m, m) * D$ by MATLAB, which is the product of a banded sparse matrix with D ;
- $\rho = 0.1\rho_{\max}$, where ρ_{\max} is the maximum regularization parameter when the solution is $x^* = 0$. The concrete definition of ρ_{\max} can be found in [20, Subsection 2.1].

Then we give the settings of initial points, maximum iterations and stopping criterions for the above algorithms. We set the initial points are $x^0 = y^0 = z^0 = 0$ and $\lambda^0 = \mu^0 = 0$. The maximum outer iteration step is 5000.

We adopt the stopping criterion as in [3] that the primal residual and dual residual are small at the iteration k :

$$\|\tilde{A}x^k + B\tilde{y}^k\|_2 \leq \epsilon_k^{\text{pri}} \quad \text{and} \quad \|\beta\tilde{A}^\top B(\tilde{y}^k - \tilde{y}^{k-1})\|_2 \leq \epsilon_k^{\text{dual}}, \quad (4.7)$$

where $\epsilon_k^{\text{pri}} > 0$ and $\epsilon_k^{\text{dual}} > 0$ are feasibility tolerances for the primal and dual feasibility conditions, respectively. These tolerances can be chosen using absolute and relative criteria from the suggestion in [3], such as

$$\begin{aligned} \epsilon_k^{\text{pri}} &= \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \max\{\|\tilde{A}x^k\|_2, \|B\tilde{y}^k\|_2\}, \\ \epsilon_k^{\text{dual}} &= \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}} \|\tilde{A}^\top \tilde{\lambda}^k\|_2, \end{aligned}$$

where $\epsilon^{\text{abs}} > 0$ is an absolute tolerance and $\epsilon^{\text{rel}} > 0$ is a relative tolerance. We set $\epsilon^{\text{abs}} = 10^{-4}$, $\epsilon^{\text{rel}} = 10^{-3}$.

We give the notations that will be used in the following tables for numerical results.

- Iter.: the outer iteration steps for each algorithm;
- Int.Iter.: the total internal iterations of the Newton method for each algorithm;
- Time: the total CPU time for each algorithm;
- T-M: the CPU time of computing for the Cholesky factorization of ADMM-2; or the CPU time of computing for the maximum eigenvalue of ADM-PRO, ADM-LBFGS, and ADM-ILBFGS;
- T-A: the CPU time for the algorithm proceed without T-M.

4.3 Numerical results

It is well known that the BFGS update is more effective than the DFP update from computational experience. We first choose $t \in \{-0.1, 0, 0.1\}$ in (2.4). Note that $t = -0.1$ can not guarantee the positive definiteness in theory but work in practice. At first, we set the sizes and sparsity of the matrix D be $m \in \{500, 1000\}$, $n \in \{200, 500\}$, and $p = 0.1$. We test different $\beta \in (0, 500]$ and chose the best one. The results on iterations and CPU time (in seconds) for ADM-Broyden with different t are provided in Table 1. All the results are averaged over 10 trials.

Table 1: Comparison on iteration steps and CPU time (seconds) for different t of ADM-Broyden

Setting			ADM-Broyden $t = -0.1$				ADM-Broyden $t = 0$				ADM-Broyden $t = 0.1$			
m	n	p	β	Iter.	Int.Iter.	Time	β	Iter.	Int.Iter.	Time	β	Iter.	Int.Iter.	Time
500	200	0.1	2.4	72.0	216.0	0.16	2.3	74.0	222.0	0.16	2.4	72.0	216.0	0.16
500	500	0.1	3.3	81.0	243.0	0.46	3.5	83.0	249.0	0.49	2.2	82.0	246.0	0.44
1000	500	0.1	2.3	158.0	474.0	1.14	2.2	155.0	465.0	1.11	2.2	157.0	471.0	1.12

Table 1 shows that ADM-Broyden with $t = -0.1$ and 0.1 some times work well from the viewpoint of the number of iteration.

Next we test the same problems as those in Table 1 among the different classical ADMMs, proximal ADMM and ADMM with the BFGS update. Table 2 shows the iterative steps and the CPU time (in seconds) results.

Table 2: Comparison on iteration steps and CPU time (seconds) among the four methods

Setting			ADMM-1				ADMM-2				ADM-PRO				ADM-BFGS			
m	n	p	β	Iter.	Int.Iter.	Time	β	Iter.	Int.Iter.	Time	β	Iter.	Int.Iter.	Time	β	Iter.	Int.Iter.	Time
500	200	0.1	4.0	15.0	60.0	0.35	0.6	77.0	306.0	0.14	0.3	131.0	524.0	0.23	2.3	74.0	222.0	0.16
500	500	0.1	4.0	20.0	100.0	2.50	0.6	105.0	418.0	0.26	0.3	390.0	1559.0	0.75	3.5	83.0	249.0	0.49
1000	500	0.1	9.0	19.0	85.0	4.59	0.8	128.0	510.0	0.51	0.3	233.0	931.0	0.88	2.2	155.0	465.0	1.11

It is obvious to see from Table 2 that the classical ADMM-1 always gets a solution within least iterative steps but takes a lot of time computing the x -subproblems by the Newton method. We also know that classical ADMM-2 spends less CPU time to get the solution. The ADMM with the BFGS update (ADM-BFGS) is better than the proximal ADMM (ADM-PRO) on the number of iterations. When the data matrix A is ill-conditioned, or the inverse of the Hessian matrix of augmented Lagrangian function is impossible to be computed, it is meaningful to use the proximal matrix H_k . ADM-BFGS needs more memories to save the H_k , and thus we consider to use the L-BFGS update to construct the H_k for some larger cases.

We take the memory as 40 for the ADMM with the L-BFGS update. The upper results in Table 3 show the iteration steps and CPU time (in seconds) for a smaller and sparse matrix D when $m = 1000$, $n \in \{500, 1000, 2000\}$ and $p = 0.1$. The middle part in Table 3 shows the iteration steps and CPU time results when $m = 5000$, $n = 1000$ and $p \in \{0.1, 0.5, 1\}$. The results are compared among ADMM-1, ADMM-2, ADM-PRO, ADM-LBFGS, and

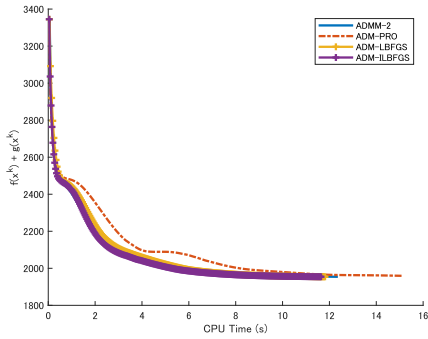
ADM-ILBFGS. We also plot the objective function values with respect to the CPU time of matrix D (same as matrix A) with different sparsities among the methods except for ADMM-1 in Figure 1. In the lower part of Table 3, we test the behaviours of ADMMs for a larger matrix D when $m = 10000$, $n = 5000$ and $p \in \{0.1, 0.5, 1\}$.

Table 3: Comparison on iteration steps and CPU time (seconds) among the five methods

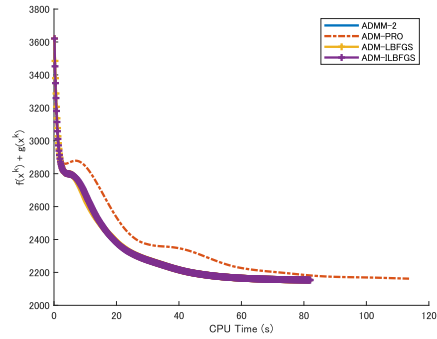
Algorithm	$m = 1000, n = 500, p = 0.1$						$m = 1000, n = 1000, p = 0.1$						$m = 1000, n = 2000, p = 0.1$					
	β	Iter.	Int.Iter.	Time	T-A	T-M	β	Iter.	Int.Iter.	Time	T-A	T-M	β	Iter.	Int.Iter.	Time	T-A	T-M
ADMM-1	9.0	17.0	85.0	4.59	—	—	10.0	19.0	95.0	13.46	—	—	9.0	23.0	115.0	53.47	—	—
ADMM-2	0.8	128.0	510.0	0.51	0.46	0.05	0.8	159.0	633.0	1.18	0.99	0.19	0.6	223.0	891.0	3.13	2.93	0.20
ADM-PRO	0.3	233.0	931.0	0.86	0.83	0.03	0.4	550.0	2199.0	2.64	2.59	0.05	0.2	941.0	3764.0	10.68	10.61	0.07
ADM-LBFGS	0.7	143.0	570.0	0.55	0.52	0.03	0.8	184.0	733.0	0.94	0.89	0.05	0.8	305.0	1217.0	3.75	3.68	0.07
ADM-ILBFGS	0.7	139.0	554.0	0.55	0.52	0.03	1.0	185.0	736.0	0.95	0.90	0.05	0.9	294.0	1173.0	3.58	3.51	0.07

	$m = 5000, n = 1000, p = 0.1$						$m = 5000, n = 1000, p = 0.5$						$m = 5000, n = 1000, p = 1.0$					
	β	Iter.	Int.Iter.	Time	T-A	T-M	β	Iter.	Int.Iter.	Time	T-A	T-M	β	Iter.	Int.Iter.	Time	T-A	T-M
ADMM-1	40	16.0	80.0	153.79	—	—	161	15.0	74.0	612.32	—	—	250	15.0	74.0	984.23	—	—
ADMM-2	1.7	266.0	1057.0	11.67	11.41	0.26	3.9	526.0	1578.0	81.18	79.56	1.62	4.6	665.0	1995.0	142.19	138.47	3.72
ADM-PRO	1.1	344.0	1373.0	14.40	14.19	0.12	2.1	719.0	2859.0	111.96	111.41	0.55	2.6	920.0	3625.0	196.81	195.91	0.90
ADM-LBFGS	1.7	268.0	1064.0	11.22	11.10	0.12	3.7	531.0	1593.0	80.30	79.75	0.55	4.6	669.0	2007.0	142.71	141.81	0.90
ADM-ILBFGS	1.7	267.0	1061.0	11.02	10.90	0.12	3.8	528.0	1584.0	79.80	79.25	0.55	4.6	667.0	2001.0	141.63	140.73	0.90

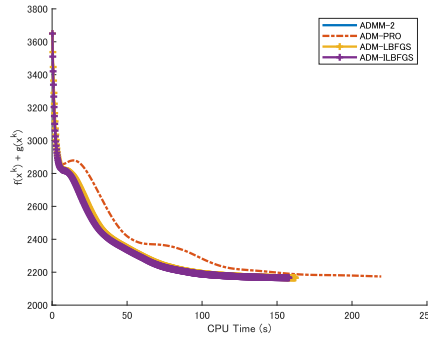
	$m = 10000, n = 5000, p = 0.1$						$m = 10000, n = 5000, p = 0.5$						$m = 10000, n = 5000, p = 1.0$					
	β	Iter.	Int.Iter.	Time	T-A	T-M	β	Iter.	Int.Iter.	Time	T-A	T-M	β	Iter.	Int.Iter.	Time	T-A	T-M
ADMM-2	2.2	475.0	1892.0	193.10	181.67	11.43	4.3	1005.0	3015.0	1407.36	1316.69	90.67	5.3	1258.0	3774.0	2769.45	2554.13	215.32
ADM-PRO	0.6	736.0	2944.0	262.77	259.87	2.90	0.9	1405.0	5618.0	1843.21	1834.80	8.41	1.0	1780.0	7118.0	3664.39	3653.84	10.55
ADM-LBFGS	2.2	486.0	1935.0	178.95	176.05	2.90	4.0	1038.0	3114.0	1370.20	1361.79	8.41	4.5	1319.0	3957.0	2712.25	2701.70	10.55
ADM-ILBFGS	2.3	477.0	1898.0	175.98	173.08	2.90	4.2	1024.0	3072.0	1359.35	1350.94	8.41	5.0	1278.0	3834.0	2629.89	2619.34	10.55



(a) $m = 5000, n = 1000, p = 0.1$



(b) $m = 5000, n = 1000, p = 0.5$



(c) $m = 5000, n = 1000, p = 1$

Figure 1: Evolution of the objective function values with respect to CPU time for small problems

From Table 3 and Figure 1, we conclude that ADM-LBFGS performs well. When matrix A is larger and has more non-zero elements, ADMM-1 spends too much time. The usual proximal ADMM (ADM-PRO) takes much more iteration steps. ADM-LBFGS and ADM-ILBFGS algorithms can reach the solutions within the same levels both at iteration and CPU time for the algorithm (T-A) as the ADMM-2 while the indefinite ADM-ILBFGS is a slight better than the semidefinite ADM-LBFGS. If matrix A is large enough, non-sparse or ill-conditioned (it is difficult or impossible to compute the inverse of the matrix $(\beta I + \beta A^\top A)$), the CPU time of computing for the Cholesky factorization of ADMM-2 (T-M) is longer than the CPU time of computing for the maximum eigenvalue of ADM-LBFGS, and thus it is useful to choose ADMM with the L-BFGS update methods.

5 Conclusions

In this paper, we proposed a proximal ADMM where the proximal matrix derived from the Broyden family update for the general convex optimization problems (1.7) and (1.8). The x -subproblems of these convex problems can be rewritten as unconstrained quadratic programming problems in Subsection 2.2 and 2.3 as that in the paper [16], and hence the Hessian matrix of the augmented Lagrangian function is a constant matrix. The global convergences of such methods have also been established under some standard conditions. The numerical results for the L1 regularized logistic regression problem are given to show the feasibility and effectiveness of the proposed algorithms.

References

- [1] https://web.stanford.edu/~boyd/papers/admm_distr_stats.html.
- [2] S. BANERT, R. I. BOT, AND E. R. CSETNEK, *Fixing and extending some recent results on the ADMM algorithm*, arXiv preprint arXiv:1612.05057, (2016).
- [3] S. BOYD, N. PARIKH, E. CHU, B. PELEATO, AND J. ECKSTEIN, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends® in Machine Learning, 3 (2011), pp. 1–122.
- [4] G. CHEN AND M. TEBoulLE, *A proximal-based decomposition method for convex minimization problems*, Mathematical Programming, 64 (1994), pp. 81–101.
- [5] W. DENG AND W. YIN, *On the global and linear convergence of the generalized alternating direction method of multipliers*, tech. rep., DTIC Document, 2012.
- [6] J. ECKSTEIN AND D. P. BERTSEKAS, *On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators*, Mathematical Programming, 55 (1992), pp. 293–318.
- [7] J. ECKSTEIN AND M. FUKUSHIMA, *Some reformulations and applications of the alternating direction method of multipliers*, in Large scale optimization, Springer, 1994, pp. 115–134.
- [8] J. ECKSTEIN AND W. YAO, *Approximate ADMM algorithms derived from Lagrangian splitting*, Computational Optimization and Applications, 68 (2017), pp. 363–405.
- [9] ———, *Relative-error approximate versions of Douglas–Rachford splitting and special cases of the ADMM*, Mathematical Programming, 170 (2018), pp. 417–444.
- [10] M. FAZEL, T. K. PONG, D. SUN, AND P. TSENG, *Hankel matrix rank minimization with applications to system identification and realization*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 946–977.
- [11] R. FLETCHER, *Practical methods of optimization*, John Wiley & Sons, 2013.
- [12] C. A. FLOUDAS AND P. M. PARDALOS, *Encyclopedia of optimization*, vol. 1, Springer Science & Business Media, 2001.
- [13] D. GABAY AND B. MERCIER, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Computers & Mathematics with Applications, 2 (1976), pp. 17–40.
- [14] R. GLOWINSKI AND A. MARROCO, *Sur l’approximation, par éléments finis d’ordre un, et la résolution, par pénalisation-dualité d’une classe de problèmes de dirichlet non linéaires*, ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 9 (1975), pp. 41–76.

- [15] M. L. N. GONÇALVES, M. M. ALVES, AND J. G. MELO, *Pointwise and ergodic convergence rates of a variable metric proximal alternating direction method of multipliers*, Journal of Optimization Theory and Applications, 177 (2018), pp. 448–478.
- [16] Y. GU AND N. YAMASHITA, *An alternating direction method of multipliers with the BFGS update for structured convex quadratic optimization*, arXiv e-prints arXiv:1903.02270, (2019).
- [17] B. HE, L.-Z. LIAO, D. HAN, AND H. YANG, *A new inexact alternating directions method for monotone variational inequalities*, Mathematical Programming, 92 (2002), pp. 103–118.
- [18] B. HE, F. MA, AND X. YUAN, *Optimal linearized alternating direction method of multipliers for convex programming*, Available on <http://www.optimization-online.org>, (2017).
- [19] B. HE AND X. YUAN, *On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 700–709.
- [20] K. KOH, S.-J. KIM, AND S. BOYD, *An interior-point method for large-scale l_1 -regularized logistic regression*, Journal of Machine learning research, 8 (2007), pp. 1519–1555.
- [21] M. LI, D. SUN, AND K.-C. TOH, *A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization*, SIAM Journal on Optimization, 26 (2016), pp. 922–950.
- [22] P. A. LOTITO, L. A. PARENTE, AND M. SOLODOV, *A class of variable metric decomposition methods for monotone variational inclusions*, J. Convex Anal, 16 (2009), pp. 857–880.
- [23] D. G. LUENBERGER, Y. YE, ET AL., *Linear and nonlinear programming*, vol. 2, Springer, 1984.
- [24] J. NOCEDAL AND S. WRIGHT, *Numerical optimization*, Springer Science & Business Media, 2006.
- [25] R. T. ROCKAFELLAR AND R. J.-B. WETS, *Variational analysis*, vol. 317, Springer Science & Business Media, 2009.
- [26] M. XU AND T. WU, *A class of linearized proximal alternating direction methods*, Journal of Optimization Theory and Applications, 151 (2011), pp. 321–337.
- [27] X. YUAN, *The improvement with relative errors of he et al.'s inexact alternating direction method for monotone variational inequalities*, Mathematical and computer modelling, 42 (2005), pp. 1225–1236.