Parallel Direction Method of Multipliers

Authored by:

Huahua Wang
Arindam Banerjee
Zhi-Quan Luo

Abstract

We consider the problem of minimizing block-separable convex functions subject to linear constraints. While the Alternating Direction Method of Multipliers (ADMM) for two-block linear constraints has been intensively studied both theoretically and empirically, in spite of some preliminary work, effective generalizations of ADMM to multiple blocks is still unclear. In this paper, we propose a parallel randomized block coordinate method named Parallel Direction Method of Multipliers (PDMM) to solve the optimization problems with multi-block linear constraints. PDMM randomly updates some blocks in parallel, behaving like parallel randomized block coordinate descent. We establish the global convergence and the iteration complexity for PDMM with constant step size. We also show that PDMM can do randomized block coordinate descent on overlapping blocks. Experimental results show that PDMM performs better than state-of-the-arts methods in two applications, robust principal component analysis and overlapping group lasso.

1 Paper Body

In this paper, we consider the minimization of block-seperable convex functions subject to linear constraints, with a canonical form: J J X X min f (x) = fj (xj) , s.t. Ax = Acj xj = a , (1) {xj ?Xj }
j=1
j=1

where the objective function f (x) is a sum of J block separable P (nonsmooth) convex functions, Acj ? Rm?nj is the j-th column block of A ? Rm?n where n = j nj , xj ? Rnj ?1 is the j-th block coordinate of x, Xj is a local convex constraint of xj and a ? Rm?1 . The canonical form can be extended to handle linear inequalities by introducing slack variables, i.e., writing Ax ? a as Ax + z = a, z ? 0. A variety of machine learning problems can be cast into the linearly-constrained optimization problem (1) [8, 4, 24, 5, 6, 21, 11]. For example, in robust Principal Component Analysis (RPCA) [5], one attempts to recover a

low rank matrix L and a sparse matrix S from an observation matrix M, i.e., the linear constraint is M = L + S. Further, in the stable version of RPCA [29], an noisy matrix Z is taken into consideration, and the linear constraint has three blocks, i.e., M = L + S + Z. Problem (1) can also include composite minimization problems which solve a sum of a loss function and a set of non-smooth regularization functions. Due to the increasing interest in structural sparsity [1], composite regularizers have become widely used, e.g., overlapping group lasso [28]. As the blocks are overlapping in this class of problems, it is difficult to apply block coordinate descent methods for large scale problems [16, 18] which assume block-separable. By simply splitting blocks and introducing equality constraints, the composite minimization problem can also formulated as (1) [2]. A classical approach to solving (1) is to relax the linear constraints using the (augmented) Lagrangian, i.e., ? (2) L? (x, y) = f (x) + hy, Ax ? ai + kAx ? ak22 , 2 1

where ? ? 0 is called the penalty parameter. We call x the primal variable and y the dual variable. (2) usually leads to primal-dual algorithms which update the primal and dual variables alternatively. While the dual update is simply dual gradient descent, the primal update is to solve a minimization problem of (2) given y. If ? = 0, the primal update can be solved in a parallel block coordinate fashion [3, 19], leading to the dual ascent method. While the dual ascent method can achieve massive parallelism, a careful choice of stepsize and some strict conditions are required for convergence, particularly when f is nonsmooth. To achieve better numerical efficiency and convergence behavior compared to the dual ascent method, it is favorable to set ? ¿ 0 in the augmented Lagrangian (2) which we call the method of multipliers. However, (2) is no longer separable and solving entire augmented Lagrangian (2) exactly is computationally expensive. In [20], randomized block coordinate descent (RBCD) [16, 18] is used to solve (2) exactly, but leading to a double-loop algorithm along with the dual step. More recent results show (2) can be solved inexactly by just sweeping the coordinates once using the alternating direction method of multipliers (ADMM) [12, 2]. This paper attempts to develop a parallel randomized block coordinate variant of ADMM. When J = 2, ADMM has been widely used to solve the augmented Lagragian (2) in many applications [2]. Encouraged by the success of ADMM with two blocks, ADMM has also been extended to solve the problem with multiple blocks [15, 14, 10, 17, 13, 7]. The variants of ADMM can be mainly divided into two categories. The first category considers Gauss-Seidel ADMM (GSADMM) [15, 14], which solves (2) in a cyclic block coordinate manner. In [13], a back substitution step was added so that the convergence of ADMM for multiple blocks can be proved. In some cases, it has been shown that ADMM might not converge for multiple blocks [7]. In [14], a block successive upper bound minimization method of multipliers (BSUMM) is proposed to solve the problem (1). The convergence of BSUMM is established under some fairly strict conditions: (i) certain local error bounds hold; (ii) the step size is either sufficiently small or decreasing. However, in general, Gauss-Seidel ADMM with multiple blocks is not well understood and its iteration complexity is largely open. The second category considers Jacobian variants of ADMM [26, 10, 17],

which solves (2) in a parallel block coordinate fashion. In [26, 17], (1) is solved by using two-block ADMM with splitting variables (sADMM). [10] considers a proximal Jacobian ADMM (PJADMM) by adding proximal terms. A randomized block coordinate variant of ADMM named RBSUMM was proposed in [14]. However, RBSUMM can only randomly update one block. Moreover, the convergence of RBSUMM is established under the same conditions as BSUMM and its iteration complexity is unknown. In this paper, we propose a parallel randomized block coordinate method named parallel direction method of multipliers (PDMM) which randomly picks up any number of blocks to update in parallel, behaving like randomized block coordinate descent [16, 18]. Like the dual ascent method, PDMM solves the primal update in a parallel block coordinate fashion even with the augmentation term. Moreover, PDMM inherits the merits of the method of multipliers and can solve a fairly large class of problems, including nonsmooth functions. Technically, PDMM has three aspects which make it distinct from such state-of-the-art methods. First, if block coordinates of the primal x is solved exactly, PDMM uses a backward step on the dual update so that the dual variable makes conservative progress. Second, the sparsity of A and the number of randomized blocks are taken into consideration to determine the step size of the dual update. Third, PDMM can randomly update arbitrary number of primal blocks in parallel. Moreover, we show that sADMM and PJADMM are the two extreme cases of PDMM. The connection between sADMM and PJADMM through PDMM provides better understanding of dual backward step. PDMM can also be used to solve overlapping groups in a randomized block coordinate fashion. Interestingly, the corresponding problem for RBCD [16, 18] with overlapping blocks is still an open problem. We establish the global convergence and O(1/T) iteration complexity of PDMM with constant step size. We evaluate the performance of PDMM in two applications: robust principal component analysis and overlapping group lasso. The rest of the paper is organized as follows: We introduce PDMM in Section 2, and establish convergence results in Section 3. We evaluate the performance of PDMM in Section 4 and conclude in Section 5. The technical analysis and detailed proofs are provided in the supplement. Notations: Assume that A ? Rm?n is divided into I ? J blocks. Let Ari ? Rmi ?n be the i-th row block of A, Acj ? Rm?nj be the j-th column block of A, and Aij ? Rmi ?nj be the ij-th block of A. Let yi ? Rmi ?1 be the i-th block of y ? Rm?1 . Let N (i) be a set of nonzero blocks Aij in the

2

? i = min{di , K} where i-th row block Ari and di = —N (i)— be the number of nonzero blocks. Let K K is the number of blocks randomly chosen by PDMM and T be the number of iterations.

2

Parallel Direction Method of Multipliers

Consider a direct Jacobi version of ADMM which updates all blocks in parallel: xt+1 = argminxj ?Xj L? (xj , xtk6=j , yt) , j y

t+1

t

= y + ? ?(Ax
t+1
? a) .
(3) (4)

where ? is a shrinkage factor for the step size of the dual gradient ascent update. However, empirical results show that it is almost impossible to make the direct Jacobi updates (3)-(4) to converge even when ? is extremely small. [15, 10] also noticed that the direct Jacobi updates may not converge. To address the problem in (3) and (4), we propose a backward step on the dual update. Moreover, instead of updating all blocks, the blocks xj will be updated in a parallel randomized block coordinate fashion. We call the algorithm Parallel Direction Method of Multipliers (PDMM). PDMM first randomly select K blocks denoted by set Jt at time t, then executes the following iterates: t ? t) + ?jt B?jt (xjt , xtjt) , jt ? Jt , xt+1 jt = argmin L? (xjt , xk6=jt , y
(5)
yit+1 = yit + ?i ?(Ai xt+1 ? ai) ,
(6)
? it+1 y
(7)
xjt ?Xjt
=
yit+1
? ?i ?(Ai x
t+1
? ai) ,

where ?i ¿ 0, 0 ? ?i ¡ 1, ?jt ? 0, and B?jt (xjt , xtjt) is a Bregman divergence. Note xt+1 = t (xt+1 Jt , xk?J / t) in (6) and (7). (6) and (7) update all dual blocks. We show that PDMM can also do ? i = min{di , K}. ?i and randomized dual block coordinate ascent in an extended work [25]. Let K ?i can take the following values: K 1 , ?i = 1 ? ?i = . (8) ? ? Ki (2J ? K) Ki In the xjt -update (5), a Bregman divergence is addded so that exact PDMM and its inexact variants can be analyzed in an unified framework [23, 11]. In particular, if ?jt = 0, (5) is an exact update. If ?jt ¿ 0, by choosing a suitable Bregman divergence, (5) can be solved by various inexact updates, often yielding a closed-form for the xjt update (see Section 2.1). To better understand PDMM, we discuss the following three aspects which play roles in choosing ?i and ?i : the dual backward step (7), the sparsity of A, and the choice of randomized blocks. Dual Backward Step: We attribute the failure of the Jacobi updates (3)-(4) to the following observation in (3), which can be rewritten as: ? xt+1 = argminxj ?Xj fj (xj) + hyt + ?(Axt ? a), Acj xj i + kAcj (xj ? xtj)k22 . (9) j 2 In the primal xj update, the quadratic penalty term implicitly adds full gradient ascent step to the dual variable, i.e., yt +?(Axt ?a), which we call implicit dual ascent. The implicit dual ascent along with the explicit dual ascent (4) may lead to too aggressive progress on the dual variable, particularly when the number of blocks is large. Based on this observation, we introduce an intermediate variable ? t to replace yt in (9) so that the implicit

dual ascent in (9) makes conservative progress, e.g., y ? t + ?(Axt ? a) = yt + (1 ? ?)?(Axt ? a) , where 0 ¡ ? ¡ 1. y ? t is the result of a ?backward y ? t = yt ? ??(Axt ? a). step? on the dual variable, i.e., y Moreover, one can show that ? and ? have also been implicitly used when using two-block ADMM with splitting variables (sADMM) to solve (1) [17, 26]. Section 2.2 shows sADMM is a special case of PDMM. The connection helps in understanding the role of the two parameters ?i , ?i in PDMM. Interestingly, the step sizes ?i and ?i can be improved by considering the block sparsity of A and the number of random blocks K to be updated. Sparsity of A: Assume A is divided into I ? J blocks. While xj can be updated in parallel, the matrix multiplication Ax in the dual update (4) requires synchronization to gather messages from coordinates jt ? Jt . For updating the i-th block of the dual yi , we need Ai xt+1 = P all blockt+1 P t jt ?Jt Aijt xjt + k?J / t Aik xk which aggregates ?messages? from all xjt . If Aijt is a block of 3

P zeros, there is no ?message? from xjt to yi . More precisely, Ai xt+1 = jt ?Jt ?N (i) Aijt xt+1 jt + P t k?J / t Aik xk where N (i) denotes a set of nonzero blocks in the i-th row block Ai . N (i) can be considered as the set of neighbors of the i-th dual block yi and di = —N (i)— is the degree of the i-th dual block yi . If A is sparse, di could be far smaller than J. According to (8), a low di will lead to bigger step sizes ?i for the dual update and smaller step sizes for the dual backward step (7). Further, as shown in Section 2.3, when using PDMM with all blocks to solve composite minimization with overlapping blocks, PDMM can use ?i = 0.5 which is much larger than 1/J in sADMM. Randomized Blocks: The number of blocks to be randomly chosen also has the effect on ?i , ?i . 1 If randomly choosing one block (K = 1), then ?i = 0, ?i = 2J?1 . The dual backward step (7) 1 1 vanishes. As K increases, ?i increases from 0 to 1 ? di and ?i increases from 2J?1 to d1i . If updating all blocks (K = J), ?i = d1i , ?i = 1 ? d1i . PDMM does not necessarily choose any K combination of J blocks. The J blocks can be randomly partitioned into J/K groups where each group has K blocks. Then PDMM randomly picks some groups. A simple way is to permutate the J blocks and choose K blocks cyclically. 2.1

Inexact PDMM

If ?jt ¿ 0, there is an extra Bregman divergence term in (5), which can serve two purposes. First, choosing a suitable Bregman divergence can lead to an efficient solution for (5). Second, if ?jt is sufficiently large, the dual update can use a large step size (?i = 1) and the backward step (7) can be removed (?i = 0), leading to the same updates as PJADMM [10] (see Section 2.2). Given a continuously differentiable and strictly convex function ?jt , its Bregman divergence is defiend as B?jt (xjt , xtjt) = ?jt (xjt) ? ?jt (xtjt) ? h??jt (xtjt), xjt ? xtjt i,

(10)

where ??jt denotes the gradient of ?jt . Rearranging the terms yields ?jt (xjt) ? B?jt (xjt , xtjt) = ?jt (xtjt) + h??jt (xtjt), xjt ? xtjt i,

(11)

which is exactly the linearization of ?jt (xjt) at xtjt . Therefore, if solving (5) exactly becomes difficult due to some problematic terms, we can use the

Bregman divergence to linearize these problematic terms so that (5) can be solved efficiently. More specifically, in (5), we can choose ?jt = ?jt ? ?1j ?jt assuming ?jt is the problematic term. Using the linearity of Bregman divert gence, 1 B?jt (xjt , xtjt) = B?jt (xjt , xtjt) ? B?jt (xjt , xtjt) . (12) ?jt For instance, if fjt is a logistic function, solving (5) exactly requires an iterative algorithm. Setting ?jt = fjt , ?jt = 21 k? k22 in (12) and plugging into (5) yield X ? t xt+1 yt , Ajt xjt i+ kAjt xjt + Ak xtk ?ak22 +?jt kxjt ?xtjt k22 , jt = argmin h?fjt (xjt), xjt i+h? 2 xjt ?Xjt k6=jt

which has a closed-form solution. Similarly, if the quadratic penalty term ?2 kAcjt xjt + P ? c 2 c t 2 k6=jt Ak xk ? ak2 is a problematic term, we can set ?jt (xjt) = 2 kAjt xjt k2 , then ? c t 2 t B?jt (xjt , xjt) = 2 kAjt (xjt ? xjt)k2 can be used to linearize the quadratic penalty term. In (12), the nonnegativeness of B?jt implies that B?jt ? ?1j B?jt . This condition can be satisfied t as long as ?jt is more convex than ?jt . Technically, we assume that ?jt is ?/?jt -strongly convex and ?jt has Lipschitz continuous gradient with constant ?, which has been shown in [23]. 2.2

Connections to Related Work

Consider the case when all blocks are used in PDMM. There are also two other methods which update all blocks in parallel. If solving the primal updates exactly, two-block ADMM with splitting variables (sADMM) is considered in [17, 26]. We show that sADMM is a special case of PDMM when setting ?i = J1 and ?i = 1 ? J1 (Appendix B in [25]). If the primal updates are solved inexactly, [10] considers a proximal Jacobian ADMM (PJADMM) by adding proximal terms where 4

the converge rate is improved to o(1/T) given the sufficiently large proximal terms. We show that PJADMM [10] is also a special case of PDMM (Appendix C in [25]). sADMM and PJADMM are two extreme cases of PDMM. The connection between sADMM and PJADMM through PDMM can provide better understanding of the three methods and the role of dual backward step. If the primal update is solved exactly which makes sufficient progress, the dual update should take small step, e.g., sADMM. On the other hand, if the primal update takes small progress by adding proximal terms, the dual update can take full gradient step, e.g. PJADMM. While sADMM is a direct derivation of ADMM, PJADMM introduces more terms and parameters. In addition to PDMM, RBUSMM [14] can also randomly update one block. The convergence of RBSUMM requires certain local error bounds to be hold and decreasing step size. Moreover, the iteration complexity of RBSUMM is still unknown. In contast, PDMM converges at a rate of O(1/T) with the constant step size. 2.3

Randomized Overlapping Block Coordinate Descent

Consider the composite minimization problem of a sum of a loss function ‘(w) and composite regularizers gj (wj): L X min ‘(w) + gj (wj) , (13) w

j=1

which considers L overlapping groups wj ? Rb?1 . Let J = L + 1, xJ = w. For 1 ? j ? L, denote xj = wj , then xj = UTj xJ , where Uj ? Rb?L is the columns of an identity matrix and extracts the coordinates of xJ . Denote U = [U1 , ? ? ? , UL] ? Rn?(bL) and A = [IbL , ?UT] where bL denotes b ? L.

By letting fj (xj) = gj (wj) and fJ (xJ) = ‘(w), (13) can be written as: J X fj (xj) s.t. Ax = 0. (14) min x

j=1 b?J

where x = [x1 ; ? ? ? ; xL ; xL+1] ? R . (14) can be solved by PDMM in a randomized block coordinate fashion. In A, for b rows block, there are only two nonzero blocks, i.e., di = 2. ThereK fore, ?i = 2(2J?K) , ?i = 0.5. In particular, if K = J, ?i = ?i = 0.5. In contrast, sADMM uses ?i = 1/J 0.5, ?i = 1 ? 1/J ¿ 0.5 if J is larger. Remark 1 (a) ADMM [2] can solve (14) where the equality constraint is xj = UTj xJ . (b) In this setting, Gauss-Seidel ADMM (GSADMM) and BSUMM [14] are the same as ADMM. BSUMM should converge with constant stepsize ? (not necessarily sufficiently small), although the theory of BSUMM does not include this special case.

3

Theoretical Results

We establish the convergence results for PDMM under fairly simple assumptions: Assumption 1 (1) fj : Rnj 7? R ? {+?} are closed, proper, and convex. (2) A KKT point of the Lagrangian (? = 0 in (2)) of Problem (1) exists. Assumption 1 is the same as that required by ADMM [2, 22]. Assume that {x?j ? Xj , yi? } satisfies the KKT conditions of the Lagrangian (? = 0 in (2)), i.e., ? ATj y? ? ?fj (x?j) ,

(15)

?

Ax ? a = 0.

(16) fj0 (xt+1 j)

t+1

?fj (xt+1 j)

= a. Let During iterations, (16) is satisfied if Ax ? where ?fj be the subdifferential of fj . For x?j ? Xj , the optimality conditions for the xj update (5) is t+1 t+1 t t ? hfj0 (xjt+1)+Acj [yt +(1??)?(Axt?a)+Acj (xt+1 j ?xj)]+?j (??j (xj)???j (xj)), xj ?xj i ? 0 .

When Axt+1 = a, yt+1 = yt . If Acj (xt+1 ? xtj) = 0, then Axt ? a = 0. When ?j ? 0, further j t+1 t assuming B?j (xj , xj) = 0, (15) will be satisfied. Note x?j ? Xj is always satisfied in (5) in 5

PDMM. Overall, the KKT conditions (15)-(16) are satisfied if the following optimality conditions are satisfied by the iterates: Axt+1 = a , Acj (xt+1 ? xtj) = 0 , (17) j t B?j (xt+1 (18) j , xj) = 0 . The above optimality conditions are sufficient for the KKT conditions. (17) are the optimality conditions for the exact PDMM. (18) is needed only when ?j ¿ 0.

Let zij = Aij xj ? Rmi ?1 , zri = [zTi1 , ? ? ? , zTiJ]T ? Rmi J?1 and z = [(zr1)T , ? ? ? , (zrI)T]T ? RJm?1 . Define the residual of optimality conditions (17)-(18) as I J X ?X ? t+1 t 2 r t+1 2 t+1 t ? z kPt + ?i kAi x ? ai k2 + R(x) = kz ?j B?j (xt+1 (19) j , xj) . 2 2 i=1 j=1 t+1) ? 0, (17)-(18) will be where Pt is some positive semi-definite matrix and ?i = JK ? i . If R(x K ? ? satisfied and thus PDMM converges to the KKT point {x , y }. Define the current iterate vt = (xtj , yit) and h(v? , vt) as a distance from vt to a KKT point v? = (x?j ? Xj , yi?):

h(v? , vt) =

I J X KX 1 ? kyi? ? yit?1 k22 + L?? (xt , yt) + kz? ? zt k2Q + ?j B?j (x?j , xtj) , (20) J i=1 2?i ? 2 j=1

where Q is a positive semi-definite matrix and L?? (xt , yt) with ?i = K?2(J?K) + d1i ? JK ? i is K i (2J?K)

I X (?i ? ?i)? r t L?? (xt , yt) = f (xt) ? f (x?) + hyit , Ari xt ? ai i + kAi x ? ai k22 . (21) 2 i=1 The following Lemma shows that h(v? , vt) ? 0. Lemma 1 Let vt = (xtj , yit) be generated by PDMM (5)-(7) and h(v? , vt) be defined in (20). K Setting ?i = 1 ? K?1 and ?i = K? (2J?K) , we have i

h(v? , vt) ? where ?i =

J?K ? i (2J?K) K

i

I ?X

J X ? ?i kAri xt ? ai k22 + kz? ? zt k2Q + ?j B?j (x?j , xtj) ? 0 . 2 i=1 2 j=1

+

1 di

?

K ?i JK

(22)

? 0. Moreover, if h(v? , vt) = 0, then Ari xt = ai , zt = z? and

B?j (x?j , xtj) = 0. Thus, (15)-(16) are satisfied. In PDMM, yt+1 depends on xt+1 , which in turn depends on Jt . xt and yt are independent of Jt . xt depends on the observed realizations of the random variable ?t?1 = {J1 , ? ? ? , Jt?1 } .The following theorem shows that h(v? , vt) decreases monotonically and thus establishes the global convergence of PDMM. Theorem 1 (Global Convergence) Let vt = (xtj , yit) be generated by PDMM (5)-(7) and v? = K (x?j ? Xj , yi?) be a KKT point satisfying (15)-(16). Setting ?i = 1 ? K?1 and ?i = K? (2J?K) , we i i have 0 ? E?t h(v? , vt+1) ? E?t?1 h(v? , vt) , E?t R(xt+1) ? 0 . (23) The following theorem establishes the iteration complexity of PDMM in an ergodic sense. ?T = Theorem 2 (Iteration Complexity) Let (xtj , yit) be generated by PDMM (5)-(7). Let x PT 1 K t ? and ?i = K ? i (2J?K) , we have t=1 x . Setting ?i = 1 ? K nP i o I 1 J ? 2 ?? (x1 , y1) + ? kz? ? z1 k2 + PJ ?j B? (x? , x1) ky k + L 2 j i j j Q i=1 j=1 K 2?i ? 2 Ef (? xT) ? f (x?) ? , T 2 I ? 0 X ? h(v , v) r T 2 ? ? ai k2 ? ?i kAi x E . T i=1 where ?i =

K ?i , JK

Q is a positive semi-definite matrix, and the expectation is over Jt . 6

4

residual (log)

2 1 0 ?1 ?2 ?3

2 1 0 ?1 ?2

200

300

400

500

600
700
800
time (s)
?5 0
8.1 8.05 8 7.95
PDMM1 PDMM2 PDMM3 GSADMM RBSUMM sADMM
7.85
?4 100
8.15
7.9
?3
?4 ?5 0
PDMM1 PDMM2 PDMM3 GSADMM RBSUMM sADMM
3
residual (log)
PDMM1 PDMM2 PDMM3 GSADMM RBSUMM sADMM
3
objective (log)
4
50
100
150
200
250
iterations
7.8
50
100
150
200
250
300
time (s)

Figure 1: Comparison of the convergence of PDMM with ADMM methods in RPCA. Table 1: The best results of PDMM with tuning parameters ?i , ?i in RPCA. time (s) iteration residual(?10?5) objective (log) PDMM1 118.83 40 3.60 8.07 PDMM2 137.46 34 5.51 8.07 PDMM3 147.82 31 6.54 8.07 GSADMM 163.09 28 6.84 8.07 RBSUMM 206.96 141 8.55 8.07 sADMM1 731.51 139 9.73 8.07 Remark 2 PDMM converges at the same rate as ADMM and its variants. In Theorem 2, PDMM can achieve the fastest convergence by setting J = K = 1, ?i = 1, ?i = 0, i.e., the entire matrix A is considered as a single block, indicating PDMM reduces to the method of multipliers. In this case, however, the resulting subproblem may be difficult to solve, as discussed in Section 1. Therefore, the number of blocks in PDMM depends on the trade-off between the number of subproblems and how efficiently each subproblem can be solved.

4

Experimental Results

In this section, we evaluate the performance of PDMM in solving robust principal component analysis (RPCA) and overlapping group lasso [28]. We compared PDMM with ADMM [2] or GSADMM (no theory guarantee), sADMM [17, 26], and RBSUMM [14]. Note GSADMM includes BSUMM [14]. All experiments are implemented in Matlab and run sequentially. We run the experiments 10 times and report the average results. The stopping criterion is either when the residual is smaller than 10?4 or when the number of iterations exceeds 2000. RPCA: RPCA is used to obtain a low rank and sparse decomposition of a given matrix A corrupted by noise [5, 17]: 1 (24) min kX1 k2F + ?2 kX2 k1 + ?3 kX3 k? s.t. A = X1 + X2 + X3 . 2 where A ? Rm?n , X1 is a noise matrix, X2 is a sparse matrix and X3 is a low rank matrix. A = L + S + V is generated in the same way as [17]1 . In this experiment, m = 1000, n = 5000 and the rank is 100. The number appended to PDMM denotes the number of blocks (K) to be chosen in PDMM, e.g., PDMM1 randomly updates one block. Figure 1 compares the convegence results of PDMM with ADMM methods. In PDMM, ? = 1 and ?i , ?i are chosen according to (8), i.e., (?i , ?i) = {(51 , 0), (14 , 12), (13 , 13)} for PDMM1, PDMM2 and PDMM3 respectively. We choose the ?best?results for GSADMM (? = 1) and RBSUMM 11 (? = 1, ? = ? ?t+10) and sADMM (? = 1). PDMMs perform better than RBSUMM and sADMM. Note the public available code of sADMM1 does not have dual update, i.e., ?i = 0. sADMM should be the same as PDMM3 if ?i = 13 . Since ?i = 0, sADMM is the slowest algorithm. Without tuning the parameters of PDMM, GSADMM converges faster than PDMM. Note PDMM can run in parallel but GSADMM only runs sequentially. PDMM3 is faster than two randomized version of PDMM since the costs of extra iterations in PDMM1 and PDMM2 have surpassed the savings at each iteration. For the two randomized one block coordinate methods, PDMM1 converges faster than RBSUMM in terms of both the number of iterations and runtime. The effect of ?i , ?i : We tuned the parameter ?i , ?i in PDMMs. Three randomized methods (RBSUMM, PDMM1 and PDMM2) choose the blocks cyclically instead of randomly. Table 1 compares the ?best?results of PDMM with other ADMM methods. In PDMM, (?i , ?i) = 1

http://www.stanford.edu/ boyd/papers/prox algs/matrix decomp.html

7

0.5 PA?APG S?APG PDMM ADMM sADMM

0.3

0.2

0.1

0 0

0 PA?APG S?APG PDMM ADMM sADMM

0.4

objective

objective

0.4

0.3
0.2
0.1
50
100
time (s)
150
200
0 0
1 21 41 61 81 101
?1
residual (log)
0.5
?2
?3
?4
200
400
600
iteration
800
1000
?5
20
30
40
50
60
70
time (s)

Figure 2: Comparison of convergence of PDMM and other methods in overlapping group Lasso. {(12 , 0), (13 , 12), (12 , 21)}. GSADMM converges with the smallest number of iterations, but PDMMs can converge faster than GSADMM in terms of runtime. The computation per iteration in GSADMM is slightly higher than PDMM3 because GSADMM updates the sum X1 + X2 + X3 but PDMM3 can reuse the sum. Therefore, if the numbers of iterations of the two methods are close, PDMM3 can be faster than GSADMM. PDMM1 and PDMM2 can be faster than PDMM3. By simply updating one block, PDMM1 is the fastest algorithm and achieves the lowest residual. Overlapping Group Lasso: We consider solving the overlapping group lasso problem [28]: X 1 min kAw ? bk22 + dg kwg k2 . w g?G 2L?

(25)

where A ? Rm?n , w ? Rn?1 and wg ? Rb?1 is the vector of overlapping group indexed by g. dg is some positive weight of group g ? G. As shown in Section 2.3, (25) can be rewritten as the form (14). The data is generated in a same way as [27, 9]: the elements of A are sampled from normal distribution,

b = Ax + with noise sampled from normal distribution, and xj = (?1)j exp(?(j ? 1)/100). In this experiment, m = 5000, the number of groups is L = 100, and dg = L1 , ? = L5 in (25). The size of each group is 100 and the overlap is 10. The total number of blocks in PDMM and sADMM is J = 101. ?i , ?i in PDMM are computed according to (8). In Figure 2, the first two figures plot the convergence of objective in terms of the number of iterations and time. PDMM uses all 101 blocks and is the fastest algorithm. ADMM is the same as GSADMM in this problem, but is slower than PDMM. Since sADMM does not consider the sparsity, it uses 1 1 ?i = J+1 , ?i = 1 ? J+1 , leading to slow convergence. The two accelerated methods, PA-APG [27] and S-APG [9], are slower than PDMM and ADMM. The effect of K: The third figure shows PDMM with different number of blocks K. Although the complexity of each iteration is the lowest when K = 1, PDMM takes much more iterations than other cases and thus takes the longest time. As K increases, PDMM converges faster and faster. When K = 20, the runtime is already same as using all blocks. When K ¿ 21, PDMM takes less time to converge than using all blocks. The runtime of PDMM decreases as K increases from 21 to 61. However, the speedup from 61 to 81 is negligable. We tried different set of parameters for 2 +1 (0 ? i ? 5, ? = 0.01, 0.1, 1) or sufficiently small step size, but could not see the RBSUMM ? ii+t convergence of the objective within 5000 iterations. Therefore, the results are not included here.

5

Conclusions

We proposed a randomized block coordinate variant of ADMM named Parallel Direction Method of Multipliers (PDMM) to solve the class of problem of minimizing block-separable convex functions subject to linear constraints. PDMM considers the sparsity and the number of blocks to be updated when setting the step size. We show two existing Jacobian ADMM methods are special cases of PDMM. We also use PDMM to solve overlapping block problems. The global convergence and the iteration complexity are established with constant step size. Experiments on robust PCA and overlapping group lasso show that PDMM is faster than existing methods.

Acknowledgment H. W. and A. B. acknowledge the support of NSF via IIS-1447566, IIS-1422557, CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711, IIS-0916750, and NASA grant NNX12AQ39A. H. W. acknowledges the support of DDF (2013-2014) from the University of Minnesota. A.B. acknowledges support from IBM and Yahoo. Z.Q. Luo is supported in part by the US AFOSR via grant number FA9550-12-1-0340 and the National Science Foundation via grant number DMS-1015346.

8

2 References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex Optimization with Sparsity-Inducing Norms. S. Sra, S. Nowozin, S. J. Wright., editors, Op-

timization for Machine Learning, MIT Press, 2011. [2] S. Boyd, E. Chu N. Parikh, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundation and Trends Machine Learning, 3(1):1?122, 2011. [3] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004. [4] T. Cai, W. Liu, and X. Luo. A constrained ‘1 minimization approach to sparse precision matrix estimation. Journal of American Statistical Association, 106:594?607, 2011. [5] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis ?. Journal of the ACM, 58:1?37, 2011. [6] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Latent variable graphical model selection via convex optimization. Annals of Statistics, 40:1935?1967, 2012. [7] C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. Preprint, 2013. [8] S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. SIAM review, 43:129?159, 2001. [9] X. Chen, Q. Lin, S. Kim, J. G. Carbonell, and E. P. Xing. Smoothing proximal gradient method for general structured sparse regression. The Annals of Applied Statistics, 6:719752, 2012. [10] W. Deng, M. Lai, Z. Peng, and W. Yin. Parallel multi-block admm with o(1/k) convergence. ArXiv, 2014. [11] Q. Fu, H. Wang, and A. Banerjee. Bethe-ADMM for tree decomposition based parallel MAP inference. In UAI, 2013. [12] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finiteelement approximations. Computers and Mathematics with Applications, 2:17?40, 1976. [13] B. He, M. Tao, and X. Yuan. Alternating direction method with Gaussian back substitution for separable convex programming. SIAM Journal of Optimization, pages 313?340, 2012. [14] M. Hong, T. Chang, X. Wang, M. Razaviyayn, S. Ma, and Z. Luo. A block successive upper bound minimization method of multipliers for linearly constrained convex optimization. Preprint, 2013. [15] M. Hong and Z. Luo. On the linear convergence of the alternating direction method of multipliers. ArXiv, 2012. [16] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization methods. SIAM Journal on Optimization, 22(2):341362, 2012. [17] N. Parikh and S. Boyd. Proximal algorithms. Foundations and Trends in Optimization, 1:123?231, 2014. [18] P. Richtarik and M. Takac. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Mathematical Programming, 2012. [19] N. Z. Shor. Minimization Methods for Non-Differentiable Functions. Springer-Verlag, 1985. [20] R. Tappenden, P. Richtarik, and B. Buke. Separable approximations and decomposition methods for the augmented lagrangian. Preprint, 2013. [21] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning, 1:1?305, 2008. [22] H. Wang and A. Banerjee. Online alternating direction method. In ICML, 2012. [23] H. Wang and A. Banerjee. Bregman alternating direction method of multipliers. In NIPS, 2014. [24] H. Wang, A. Banerjee, C. Hsieh, P. Ravikumar, and I. Dhillon. Large scale distributed sparse precesion estimation. In NIPS, 2013. [25] H. Wang, A. Banerjee, and Z. Luo. Parallel direction method of multipliers. ArXiv, 2014. [26] X. Wang, M. Hong, S. Ma, and Z. Luo. Solving multiple-block

separable convex minimization problems using two-block alternating direction method of multipliers. Preprint, 2013. [27] Y. Yu. Better approximation and faster algorithm using the proximal average. In NIPS, 2012. [28] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. Annals of Statistics, 37:34683497, 2009. [29] Z. Zhou, X. Li, J. Wright, E. Candes, and Y. Ma. Stable principal component pursuit. In IEEE International Symposium on Information Theory, 2010.

9