Bregman Alternating Direction Method of Multipliers

Authored by:

Huahua Wang
Arindam Banerjee

Abstract

The mirror descent algorithm (MDA) generalizes gradient descent by using a Bregman divergence to replace squared Euclidean distance. In this paper, we similarly generalize the alternating direction method of multipliers (ADMM) to Bregman ADMM (BADMM), which allows the choice of different Bregman divergences to exploit the structure of problems. BADMM provides a unified framework for ADMM and its variants, including generalized ADMM, inexact ADMM and Bethe ADMM. We establish the global convergence and the $O(1/T)$ iteration complexity for BADMM. In some cases, BADMM can be faster than ADMM by a factor of $O(n/ln n)$ where n is the dimensionality. In solving the linear program of mass transportation problem, BADMM leads to massive parallelism and can easily run on GPU. BADMM is several times faster than highly optimized commercial software Gurobi.

1 Paper Body

In recent years, the alternating direction method of multipliers (ADMM) [4] has been successfully used in a broad spectrum of applications, ranging from image processing [11, 14] to applied statistics and machine learning [26, 25, 12]. ADMM considers the problem of minimizing composite objective functions subject to an equality constraint: min

x?X ,z?Z

f (x) + g(z) s.t. Ax + Bz = c ,

(1)

where f and g are convex functions, A ? Rm?n1 , B ? Rm?n2 , c ? Rm?1 , x ? X ? Rn1 ?1 , z ? Z ? Rn2 ?1 , and X ? Rn1 and Z ? Rn2 are nonempty closed convex sets. f and g can be non-smooth functions, including indicator functions of convex sets. For further understanding of ADMM, we refer the readers to the comprehensive review by [4] and references therein. Many machine learning problems can be cast into the framework of minimizing a composite objective [22, 10], where f is a loss function such as hinge or logistic loss, and g is a regularizer, e.g., ‘1 norm, ‘2 norm, nuclear norm or total variation. The functions and constraints usually have different structures. Therefore, it is useful and sometimes necessary to split and solve them separately, which is exactly the forte of ADMM. In each iteration, ADMM updates splitting variables separately and alternatively by solving the partial augmented Lagrangian of (1), where only the equality constraint is considered: ? (2) L? (x, z, y) = f (x) + g(z) + hy, Ax + Bz ? ci + kAx + Bz ? ck22 , 2 where y ? Rm is dual variable, ? ¿ 0 is penalty parameter, and the quadratic penalty term is to penalize the violation of the equality constraint. ADMM consists of the following three updates: ? xt+1 = argminx?X f (x) + hyt , Ax + Bzt ? ci + kAx + Bzt ? ck22 , (3) 2 ? zt+1 = argminz?Z g(z) + hyt , Axt+1 + Bz ? ci + kAxt+1 + Bz ? ck22 , (4) 2 yt+1 = yt + ?(Axt+1 + Bzt+1 ? c) . (5) 1

Since the computational complexity of the y update (5) is trivial, the computational complexity of ADMM is determined by the x and z updates (3)-(4) which amount to solving proximal minimization problems using the quadratic penalty term. Inexact ADMM [26, 4] and generalized ADMM [8] have been proposed to solve the updates inexactly by linearizing the functions and adding additional quadratic terms. Recently, online ADMM [25] and Bethe-ADMM [12] add an additional Bregman divergence on the x update by keeping or linearizing the quadratic penalty term kAx + Bz ? ck22 . As far as we know, all existing ADMMs use quadratic penalty terms. A large amount of literature shows that replacing the quadratic term by Bregman divergence in gradient-type methods can greatly boost their performance in solving constrained optimization problem. First, the use of Bregman divergence could effectively exploit the structure of problems [6, 2, 10] , e.g., in computerized tomography [3], clustering problems and exponential family distributions [1]. Second, in some cases, the gradient descent method with Kullback-Leibler (KL) ? divergence can outperform the method with the quadratic term by a factor of O(n ln n) where n is the dimensionality of the problem [2, 3]. Mirror descent algorithm (MDA) and composite objective mirror descent (COMID) [10] use Bregman divergence to replace the quadratic term in gradient descent or proximal gradient [7]. Proximal point method with D-functions (PMD) [6, 5] and Bregman proximal minimization (BPM) [20] generalize proximal point method by using generalized Bregman divegence to replace the quadratic term. For ADMM, although the convergence of ADMM is well understood, it is still unknown whether the quadratic penalty term in ADMM can be replaced by Bregman divergence. The proof of global convergence of ADMM can be found in [13, 4]. Recently, it has been shown that ADMM converges at a rate of O(1/T) [25, 17], where T is the number of iterations. For strongly convex functions, the dual objective of an accelerated version of ADMM can converge at a rate of O(1/T 2) [15]. Under suitable assumptions like strongly convex functions or a sufficiently small step size for the dual variable update, ADMM can achieve a linear convergence rate [8, 19]. However, as pointed out by [4], ?There is currently no proof of convergence known for ADMM with nonquadratic penalty terms.? In this paper, we propose Bregman ADMM (BADMM) which uses Bregman divergences to replace the quadratic penalty term in ADMM, answering the question raised in [4]. More specifically, the quadratic penalty term in the x and z updates (3)-(4) will be replaced by a Bregman divergence in BADMM. We also introduce a generalized version of BADMM where two additional Bregman divergences are added to the x and z updates. The generalized BADMM (BADMM for short) provides a unified framework for solving (1), which allows one to choose suitable Bregman divergence so that the x and z updates can be solved efficiently. BADMM includes ADMM and its variants as special cases. In particular, BADMM replaces all quadratic terms in generalized ADMM [8] with Bregman divergences. By choosing a proper Bregman divergence, we also show that inexact ADMM [26] and Bethe ADMM [12] can be considered as special cases of BADMM. BADMM generalizes ADMM similar to how MDA generalizes gradient descent and how PMD generalizes proximal methods. In BADMM, the x and z updates can take the form of MDA or PMD. We establish the global convergence and the O(1/T) iteration complexity for BADMM. In some cases, we show that BADMM can outperform ADMM by a factor O(n/ ln n). We evaluate the performance of BADMM in solving the linear program problem of mass transportation [18]. Since BADMM takes use of the structure of the problem, it leads to closed-form solutions which amounts to elementwise operations and can be done in parallel. BADMM is faster than ADMM and can even be orders of magnitude faster than highly optimized commercial software Gurobi when implemented on GPU. The rest of the paper is organized as follows. In Section 2, we propose Bregman ADMM and discuss several special cases of BADMM. In Section 3, we establish the convergence of BADMM. In Section 4, we consider illustrative applications of BADMM, and conclude in Section 5.

2

Bregman Alternating Direction Method of Multipliers

Let ? : ? ? R be a continuously differentiable and strictly convex function on the relative interior of a convex set ?. Denote ??(y) as the gradient of ? at y. We define Bregman divergence B? : ? ? ri(?) ? R+ induced by ? as B? (x, y) = ?(x) ? ?(y) ? h??(y), x ? yi . 2

Since ? is strictly convex, B? (x, y) ? 0 where the equality holds if and only if x = y. More details about Bregman divergence can be found in [6, 1]. Note the definition of Bregman divergence has been generalized for the nondifferentiable functions [20, 23]. In this paper, our discussion uses the definition of classical Bregman divergence. Two of the most commonly usedPexamples are squared n Euclidean distance B? (x, y) = 12 kx ? yk22 and KL divergence B? (x, y) = i=1 xi log xyii . Assuming B? (c ? Ax, Bz) is well defined, we replace the quadratic penalty term in the partial augmented Lagrangian (2) by a Bregman divergence as follows: L?? (x, z, y) = f (x) + g(z) + hy, Ax + Bz ? ci + ?B? (c ? Ax, Bz).

(6)

Unfortunately, we can not derive Bregman ADMM (BADMM) updates by simply solving L?? (x, z, y) alternatingly as ADMM does because Bregman divergences are not necessarily convex in the second argument. More specifically, given (zt , yt), xt+1 can be obtained by solving minx?X L?? (x, zt , yt), where the quadratic penalty term 21 kAx + Bzt ? ck22 for ADMM in (3) is replaced with B? (c ? Ax, Bzt) in the x update of BADMM. However, given (xt+1 , yt), we cannot obtain zt+1 by solving minz?Z L?? (xt+1 , z, yt), since the term B? (c ? Axt+1 , Bz) need not be convex in z. The observation motivates a closer look at the role of the quadratic term in ADMM. In standard ADMM, the quadratic augmentation term added to the Lagrangian is just a penalty term to ensure the new updates do not violate the equality constraint significantly. Staying with these goals, we propose the z update augmentation term of BADMM to be: B? (Bz, c ? Axt+1), instead of the quadratic penalty term 12 kAxt+1 + Bz ? ck22 in (3). Then, we get the following updates for BADMM: xt+1 =argminx?X f (x) + hyt , Ax + Bzt ? ci + ?B? (c ? Ax, Bzt) , zt+1 =argminz?Z g(z) + hyt , Axt+1 + Bz ? ci + ?B? (Bz, c ? Axt+1) , yt+1 =yt + ?(Axt+1 + Bzt+1 ? c) .

(7) (8) (9)

Compared to ADMM (3)-(5), BADMM simply uses a Bregman divergence to replace the quadratic penalty term in the x and z updates. It is worth noting that the same Bregman divergence B? is used in the x and z updates. We consider a special case when A = ?I, B = I, c = 0. (7) is reduced to xt+1 = argminx?X f (x) + hyt , ?x + zt i + ?B? (x, zt) .

(10)

If ? is a quadratic function, the constrained problem (10) requires the projection onto the constraint set X . However, in some cases, by choosing a proper Bregman divergence, (10) can be solved efficiently or has a closed-form solution. For example, assuming f is a linear function and X is the unit simplex, choosing B? to be KL divergence leads to the exponentiated gradient [2, 3, 21]. Interestingly, if the z update is also the exponentiated gradient, we have alternating exponentiated gradients. In Section 4, we will show the mass transportation problem can be cast into this scenario. While the updates (7)-(8) use the same Bregman divergences, efficiently solving the x and z updates may not be feasible, especially when the structure of the original functions f, g, the function ? used for augmentation, and the constraint sets X , Z are rather different. For example, if f (x) is a logistic function in (10), it will not have a closed-form solution even B? is the KL divergence and X is the unit simplex. To address such concerns, we propose a generalized version of BADMM. 2.1

Generalized BADMM

To allow the use of different Bregman divergences in the x and z updates (7)-(8) of BADMM, the generalized BADMM simply introduces an additional Bregman divergence for each update. The generalized BADMM has the following updates: xt+1 =argminx?X f (x) + hyt , Ax + Bzt ? ci + ?B? (c ? Ax, Bzt) + ?x B?x (x, xt) , (11) zt+1 =argminz?Z g(z) + hyt , Axt+1 + Bz ? ci + ?B? (Bz, c ? Axt+1) + ?z B?z (z, zt) , (12) yt+1 = yt + ? (Axt+1 + Bzt+1 ? c) . (13) where ? ¿ 0, ? ¿ 0, ?x ? 0, ?z ? 0. Note that we allow the use of a different step size ? in the dual variable update [8, 19]. There are three Bregman divergences in the generalized BADMM. While 3

the Bregman divergence B? is shared by the x and z updates, the x update has its own Bregman divergence B?x and the z update has its own Bregman divergence B?z . The two additional Bregman divergences in generalized

BADMM are variable specific, and can be chosen to make sure that the xt+1 , zt+1 updates are efficient. If all three Bregman divergences are quadratic functions, the generalized BADMM reduces to the generalized ADMM [8]. We prove convergence of generalized BADMM in Section 3, which yields the convergence of BADMM with ?x = ?z = 0. In the following, we illustrate how to choose a proper Bregman divergence B?x so that the x update can be solved efficiently, e.g., a closed-form solution, noting that the same arguments apply to the z-updates. Consider the first three terms in (11) as s(x) + h(x), where s(x) denotes a simple term and h(x) is the problematic term which needs to be linearized for an efficient x-update. We illustrate the idea with several examples later in the section. Now, we have xt+1 = minx?X s(x) + h(x) + ?x B?x (x, xt) .

(14)

where efficient updates are difficult due to the mismatch in structure between h and X . The goal is to ?linearize? the function h by using the fact that the Bregman divergence Bh (x, xt) captures all the higher-order (beyond linear) terms in h(x) so that: h(x) ? Bh (x, xt) = h(xt) + hx ? xt , ?h(xt)i

(15)

is a linear function of x. Let ? be another convex function such that one can efficiently solve minx?X s(x) + ?(x) + hx, bi for any constant b. Assuming ?x (x) = ?(x) ? ?1x h(x) is continuously differentiable and strictly convex, we construct a Bregman divergence based proximal term to the original problem so that: argminx?X s(x)+h(x)+?x B?x (x,xt) = argminx?X s(x)+h?h(xt), x?xt i+?x B?x (x,xt),(16) where the latter problem can be solved efficiently, by our assumption. To ensure ?x is continuously differentiable and strictly convex, we need the following condition: Proposition 1 If h is smooth and has Lipschitz continuous gradients with constant ? under a pnorm, then ?x is ?/?x -strongly convex w.r.t. the p-norm. This condition has been widely used in gradient-type methods, including MDA and COMID. Note that the convergence analysis of generalized ADMM in Section 4 holds for any additional Bregman divergence based proximal terms, and does not rely on such specific choices. Using the above idea, one can ?linearize? different parts of the x update to yield an efficient update. We consider three special cases, respectively focusing on linearizing the function f (x), linearizing the Bregman divergence based augmentation term B? (c ? Ax, Bzt), and linearizing both terms, along with examples for each case. Case 1: Linearization of smooth function f : Let h(x) = f (x) in (16), we have xt+1 = argminx?X h?f (xt), x ? xt i + hyt , Axi + ?B? (c ? Ax, Bzt) + ?x B?x (x, xt) . (17) where ?f (xt) is the gradient of f (x) at xt . Example 1 Consider the following ADMM form for sparse logistic regression problem [16, 4]: minx h(x) + ?kzk1 , s.t. x = z ,

(18)

where h(x) is the logistic function. If we use ADMM to solve (18), the x update is as follows [4]: ? xt+1 = argminx h(x) + hyt , x ? zt i + kx ? zt k22 , 2

(19)

which is a ridge-regularized logistic regression problem and one needs an iterative algorithm like L-BFGS to solve it. Instead, if we linearize h(x) at xt and set B? to be a quadratic function, then ? ?x kx ? xt k22 , xt+1 = argminx h? h(xt), x ? xt i + hyt , x ? zt i + kx ? zt k22 + 2 2 the x update has a simple closed-form solution. 4

(20)

Case 2: Linearization of the quadratic penalty term: In ADMM, B? (c ? Ax, Bzt) = 12 kAx + Bzt ? ck22 . Let h(x) = ?2 kAx + Bzt ? ck22 . Then ?h(xt) = ?AT (Axt + Bzt ? c), we have xt+1 = argminx?X f (x) + hyt + ?(Axt + Bzt ? c), Axi + ?x B? (x, xt) .

(21)

kAxk22

The case mainly solves the problem due to the term which makes x updates nonseparable, whereas the linearized version can be solved with separable (parallel) updates. Several problems have been benefited from the linearization of quadratic term [8], e.g., when f is '1 loss function [16], and projection onto the unit simplex or '1 ball [9]. Case 3: Mirror Descent: In some settings, we want to linearize both the function f and the quadratic augmentation term B? (c ? Ax, Bzt) = 21 kAx + Bzt ? ck22 . Let h(x) = f (x) + hyt , Axi + ?2 kAx + Bzt ? ck22 , we have xt+1 = argminx?X h?h(xt), xi + ?x B? (x, xt) .

(22)

Note that (22) is a MDA-type update. Further, one can do a similar exercise with a general Bregman divergence based augmentation term B? (c ? Ax, Bzt), although there has to be a good motivation for going to this route. Example 2 [Bethe-ADMM [12]] Given an undirected graph G = (V, E), where V is the vertex set and E is the edge set. Assume a random discrete variable Xi associated with node i ? V can take K values. In a pairwise MRF, the joint distribution of a set of discrete random variables X = {X1 , ? ? ? , Xn } (n is the number of nodes in the graph) is defined in terms of nodes and cliques [24]. Consider solving the following graph-structured linear program (LP) : min l(?) s.t. ? ? L(G) , ?

(23)

where l(?) is a linear function of ? and L(G) is the so-called local polytope [24] determined by the marginalization and normalization (MN) constraints for each node and edge in the graph G: X X L(G) = {? ? 0 , ?i (xi) = 1 , ?ij (xi , xj) = ?i (xi)} , (24) xi

xj

where ?i , ?ij are pseudo-marginal distributions of node i and edge ij respectively. The LP in (23) contains O(nK + —E—K 2) variables and that order of constraints. In particular, (23) serves as a LP relaxation of MAP inference probem in a pairwise MRF if l(?) is defined as follows: XX XX l(?) = ?i (xi)?i (xi) + ?ij (xi , xj)?ij (xi , xj), (25) i

xi

ij?E xij

where ?i , ?ij are the potential functions of node i and edge ij respectively. For a grid graph (e.g., image) of size 1000?1000, (23) contains millions of variables and constraints, posing a challenge to LP solvers. An efficient way is to decompose the graph into trees such that X min c? l? (??) s.t. ?? ? T? , ?? = m? , (26) ??

?

where T? denotes the MN constraints (24) in the tree ? . ?? is a vector of pseudo-marginals of nodes and edges in the tree ? . m is a global variable which contains all trees and m? corresponds to the tree ? in the global variable. c? is the weight for sharing variables. The augmented Lagrangian is X ? (27) L? (?? , m, ??) = c? l? (??) + h?? , ?? ? m? i + k?? ? m? k22 . ? 2 which leads to the following update for ?t+1 in ADMM: ? ? ?t+1 = argmin?? ?T? c? l? (??) + h?t? , ?? i + k?? ? mt? k22 (28) ? 2 (28) is difficult to solve due to the MN constraints in the tree. Let h(??) be the objective of (28). Linearizing h(??) and adding a Bregman divergence in (28), we have: ?t+1 = argmin?? ?T? h?h(?t?), ?? i + ?x B? (?? , ?t?) ? = argmin?? ?T? h?h(?t?) ? ?x ??(?t?), ?? i + ?x ?(??) , If ?(??) is the negative Bethe entropy of ?? , the update of ?t+1 becomes the Bethe entropy prob? lem [24] and can be solved exactly using the sum-product algorithm in linear time for any tree. 5

3

Convergence Analysis of BADMM

We need the following assumption in establishing the convergence of BADMM: Assumption 1 (a) f : Rn1 7?R?{+?} and g : Rn2 7?R?{+?} are closed, proper and convex. (b) An optimal solution exists. (c) The Bregman divergence B? is defined on an ?-strongly convex function ? with respect to a p-norm k ? k2p , i.e., B? (u, v) ? ?2 ku ? vk2p , where ? ¿ 0. Assume that {x? , z? , y? } satisfies the KKT conditions of the Lagrangian of (1) (? = 0 in (2)), i.e., ?AT y? ? ?f (x?) , ?BT y? ? ?g(z?) , Ax? + Bz? ? c = 0 , (29) and x ? X , z ? Z. Note X and Z are always satisfied in (11) and (12). Let f 0 (xt+1) ? ?f (xt+1) and g 0 (zt+1) ? ?g(zt+1). For x? ? X , z? ? Z, the optimality conditions of (11) and (12) are ?

?

hf 0 (xt+1)+AT {yt +?(???(c?Axt+1)+??(Bzt)}+?x (??x (xt+1)???x (xt)), xt+1 ?x? i ? 0 , hg 0 (zt+1)+BT {yt +?(??(Bzt+1)???(c?Axt+1)}+?z (??z (zt+1)???z (zt)), zt+1 ? z? i ? 0 . If Axt+1 + Bzt+1 = c, then yt+1 = yt . Further, if B?x (xt+1 , xt) = 0, B?z (zt+1 , zt) = 0, then the KKT conditions in (29) will be satisfied. Therefore, we have the following sufficient conditions for the KKT conditions: B?x (xt+1 , xt) = 0 , B?z (zt+1 , zt) = 0 , (30a) Axt+1 + Bzt ? c = 0 , Axt+1 + Bzt+1 ? c = 0 . (30b) For the exact BADMM, ?x = ?z = 0 in (11) and (12), the optimality conditions are (30b), which is equivalent to the optimality conditions of ADMM [4], i.e., Bzt+1 ?Bzt = 0 , Axt+1 +Bzt+1 ?c = 0. Define the residuals of optimality conditions (30) at (t + 1) as: ?x ?z R(t+1) = B?x(xt+1 ,xt)+ B?z(zt+1 ,zt)+B? (c?Axt+1 ,Bzt)+?kAxt+1+Bzt+1?ck22 , (31) ? ? where ? ¿ 0. If R(t + 1) = 0, the optimality conditions (30a) and (30b) are satisfied. It is sufficient to show the convergence of BADMM by showing R(t+1) converges to zero. The following theorem establishes the global convergence for BADMM. Theorem 1 Let the sequence {xt , zt , yt } be generated by BADMM (11)-(13), {x? , z? , y? } satisfy (29) and x? ? X , z? ? Z. Let the Assumption 1 hold and ? ? (?? ?

2?)?, where 2 ? = min{1, m p ?1 } and 0 ¡ ? ¡ ?? 2 . Then R(t + 1) converges to zero and {xt , zt , yt } converges to a KKT point {x? , z? , y? }. Remark 1 (a) If 0 ¡ p ? 2, then ? = 1 and ? ? (? ? 2?)?. The case that 0 ¡ p ? 2 includes two widely used Bregman divergences, i.e., Euclidean distance and KL divergence. For KL divergence in the unit simplex, we have ? = 1, p = 1 in the Assumption 1 (c), i.e., KL(u, v) ? 21 ku ? vk21 [2]. (b) Since we often set B? to be a quadratic function (p = 2), the three special cases in Section 2.1 could choose step size ? = (? ? 2?)?. (c) If p ¿ 2, ? will be small, leading to a small step size ? which may be not be necessary in practice. It would be interesting to see whether a large step size can be used for any p ¿ 0. The following theorem establishes a O(1/T) iteration complexity for the objective and residual of constraints in an ergodic sense. Theorem 2 Let the sequences {xt , zt , yt } be generated by BADMM (11)-(13). Set ? ? (???2?)?, PT P 2 1 ? ?T = T1 Tt=1 zt and y0 = 0. where ? = min{1, m p ?1 } and 0 ¡ ? ¡ ?? t=1 xt , z 2 . Let xT = T For any x? ? X , z? ? Z and (x? , z? , y?) satisfying KKT conditions (29), we have D1 f (? xT) + g(? zT) ? (f (x?) + g(z?)) ? , (32) T D(w? , w0) kA? xT + B? zT ? ck22 ? , (33) ?T where D1 = ?B? (Bz? , Bz0) + ?x B?x (x? , x0) + ?z B?z (z? , z0) and D(w? , w0) = 2?1? ky? ? y0 k22 + B? (Bz? , Bz0) + ??x B?x (x? , x0)+ ??z B?z (z? , z0). 6

We consider one special case of BADMM where B = I and X , Z are the unit simplex. Let B? be the KL divergence. For z? ? Z ? Rn2 ?1 , choosing z0 = e/n2 , we have B? (z? , z0) = Pn2 ? Pn2 ? zi? ? i=1 zi ln zi,0 = i=1 zi ln zi + ln n2 ? ln n2 . Similarly, if ?x ¿ 0, by choosing x0 = e/n1 , B?x (x? , x0) ? ln n1 . Setting ? = 1, ? = 1 and ? = 14 in Theorem 2 yields the following result: Corollary 1 Let the sequences {xt , zt , yt } be generated by Bregman ADMM (11),(12),(13) and y0 = 0. Assume B = I, and X and Z is the unit simplex. Let B? , B?x , B?z be KL divergence. P P ? ? ? ? ? ? T = T1 Tt=1 xt , z ?T = T1 Tt=1 zt . Set ? = 3? Let x 4 . For any x ? X , z ? Z and (x , z , y) satisfying KKT conditions (29), we have ? ln n2 + ?x ln n1 + ?z ln n2 , (34) f (? xT) + g(? zT) ? (f (x?) + g(z?)) ? T 4?z 4?x 2 ? 2 ? ? ky ?y0 k2 + 4 ln n2 + ? ln n1 + ? ln n2 kA? xT + B? zT ? ck22 ? , (35) T Remark 2 (a) [2] shows that MDA yields a smilar O(ln n) bound ?where n is dimensionality ? of the problem. If the diminishing step size of MDA is propotional to ln n, the bound is O(ln n). Therefore, MDA is faster than the gradient descent method by a factor O((n/ ln n)1/2). Pn Pn (b) In ADMM, B? (z? , z0) = 12 kz? ? z0 k22 = 21 k i=1 z?i ? zi,0 k22 ? n2 i=1 kz?i ? zi,0 k22 ? n. Therefore, BADMM is faster than ADMM by a factor O(n/ ln n) in an ergodic sense.

4

Experimental Results

In this section, we use BADMM to solve the mass transportation problem [18]: min hC, Xi s.t.

Xe = a, XT e = b, X ? 0 .

(36)

where hC, Xi denotes Tr(CT X), C ? Rm?n is a cost matrix, X ? Rm?n , a ? Rm?1 , b ? Rm?1 , e is a column vector of ones. The mass transportation problem (36) is a linear program and thus can be solved by the simplex method. We now show that (36) can be solved by ADMM and BADMM. We first introduce a variable Z to split the constraints into two simplex such that ?x = {X—X ? 0, Xe = a} and ?z = {Z—Z ? 0, ZT e = b}. (36) can be rewritten in the following ADMM form: min hC, Xi s.t.

X ? ?x , Z ? ?z , X = Z .

(37)

(37) can be solved by ADMM which requires the Euclidean projection onto the simplex ?x and ?z , although the projection can be done efficiently [9]. We use BADMM to solve (37): Xt+1 = argminX??x hC, Xi + hYt , Xi + ?KL(X, Zt) , Z

t+1

Y

t+1

t

= argminZ??z hY , ?Zi + ?KL(Z, X t

= Y + ?(X

t+1

?Z

t+1

t+1

),

).

(38) (39) (40)

Both (38) and (39) have closed-form solutions, i.e., t+1 Xij =

t Cij +Yij) ? ai t Pn C +Y ij ij t) j=1 Zij exp(? ?

t Zij exp(?

,

t+1 Zij =

t Yij ?) bj t Pm Yij t+1 i=1 Xij exp(?)

t+1 Xij exp(

(41)

which are exponentiated graident updates and can be done in O(mn). Besides the sum operation (O(ln n) or O(ln m)), (41) amounts to elementwise operation and thus can be done in parallel. According to Corollary 1, BADMM can be faster than ADMM by a factor of O(n/ ln n). We compare BADMM with ADMM and a commercial LP solver Gurobi on the mass transportation problem (36) with m = n and a = b = e. C is randomly generated from the uniform distribution. We run the experiments 5 times and the average is reported. We choose the ?best?parameter for BADMM (? = 0.001) and ADMM (? = 0.001). The stopping condition is either when the number of iterations exceeds 2000 or when the primal-dual residual is less than 10?4 . BADMM vs ADMM: Figure 1 compares BADMM and ADMM with different dimensions n = {1000, 2000, 4000} running on a single CPU. Figure 1(a) plots the primal and dual residual against 7

?3

?3

x 10
1
0.8
0.6
0.4
0.2
0 0
100
200
300
400
runtime (s)
(a) m = n = 1000
500
600
x 10
20 BADMM ADMM
BADMM ADMM 0.8
Objective value
BADMM ADMM
Primal and dual residual
Primal and dual residual
1
0.6
0.4
15
10
5 0.2
0 0
500
1000
1500
Iteration
(b) m = n = 2000
2000
0 0
2000
4000
6000
8000
10000
runtime (s)
(c) m = n = 4000

Figure 1: Comparison BADMM and ADMM. BADMM converges faster than ADMM. (a): the primal and dual residual agaist the runtime. (b): the primal and dual residual over iterations. (c): The convergence of objective value

against the runtime. Table 1: Comparison of BADMM (GPU) with Gurobi in solving mass transportation problem number of variables Gurobi (Laptop) Gurobi (Server) BADMM (GPU) m?n time (s) objective time (s) objective time (s) objective (210)2 ¿ 1 million 4.22 1.69 2.66 1.69 0.54 1.69 (5 ? 210)2 ¿ 25 million 377.14 1.61 92.89 1.61 22.15 1.61 (10 ? 210)2 ¿ 0.1 billion 1235.34 1.65 117.75 1.65 (15 ? 210)2 ¿ 0.2 billion 303.54 1.63 the runtime when n = 1000, and Figure 1(b) plots the convergence of primal and dual residual over iteration when n = 2000. BADMM converges faster than ADMM. Figure 1(c) plots the convergence of objective value against the runtime when n = 4000. BADMM converges faster than ADMM even when the initial point is further from the optimum. BADMM vs Gurobi: Gurobi (http://www.gurobi.com/) is a highly optimized commercial software where linear programming solvers have been efficiently implemented. We run Gurobi on two settings: a Mac laptop with 8G memory and a server with 86G memory, respectively. For comparison, BADMM is run in parallel on a Tesla M2070 GPU with 5G memory and 448 cores1 . We experiment with large scale problems and use m = n = {1, 5, 10, 15} ? 210 . Table 1 shows the runtime and the objective values of BADMM and Gurobi, where a ?-? indicates the algorithm did not terminate. In spite of Gurobi being one of the most optimized LP solvers, BADMM running in parallel is several times faster than Gurobi. In fact, for larger values of n, Gurobi did not terminate even on the 86G server, whereas BADMM was efficient even with just 5G memory! The memory consumption of Gurobi increases rapidly with the increase of n, especially at the scales we consider. When n = 5 ? 210 , the memory required by Gurobi surpassed the memory in the laptop, leading to the rapid increase of time. A similar situation was also observed in the server with 86G when n = 10 ? 210 . In contrast, the memory required by BADMM is O(n2)?even when n = 15 ? 210 (more than 0.2 billion parameters), BADMM can still run on a single GPU with only 5G memory. The results clearly illustrate the promise of BADMM. With more careful implementation and code optimization, BADMM has the potential to solve large scale problems efficiently in parallel with small memory foot-print.

5 Conclusions

In this paper, we generalized the alternating direction method of multipliers (ADMM) to Bregman ADMM, similar to how mirror descent generalizes gradient descent. BADMM defines a unified framework for ADMM, generalized ADMM, inexact ADMM and Bethe ADMM. The global convergence and the O(1/T) iteration complexity of BADMM are also established. In some cases, BADMM is faster than ADMM by a factor of O(n/ ln n). BADMM is also faster than highly optimized commercial software in solving the linear program of mass transportation problem.

Acknowledgment The research was supported by NSF grants IIS-1447566, IIS-1422557, CCF-1451986, CNS-1314560, IIS0953274, IIS-1029711, IIS-0916750, and by NASA grant NNX12AQ39A. H.W. and A.B. acknowledge the technical support from the University of Minnesota Supercomputing Institute. H.W. acknowledges the support of DDF (2013-2014) from the University of Minnesota.

A.B. acknowledges support from IBM and Yahoo. 1
GPU code is available on https://github.com/anteagle/GPU_BADMM_MT
8

2 References

[1] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. JMLR, 6:1705? 1749, 2005. [2] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. Operations Research Letters, 31:167?175, 2003. [3] A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. SIAM Journal on Optimization, 12:79?108, 2001. [4] 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. [5] Y. Censor and S. Zenios. Parallel Optimization: Theory, Algorithms, and Applications. Oxford University Press, 1998. [6] G. Chen and M. Teboulle. Convergence analysis of a proximal-like minimization algorithm using bremgan functions. SIAM Journal on Optimization, 3:538?543, 1993. [7] P. Combettes and J. Pesquet. Proximal splitting methods in signal processsing. Fixed-Point Algorithms for Inverse Problems in Science and Engineering Springer (Ed.), pages 185?212, 2011. [8] W. Deng and W. Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. ArXiv, 2012. [9] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1 -ball for learning in high dimensions. In ICML, pages 272?279, 2008. [10] J. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. Composite objective mirror descent. In COLT, 2010. [11] M. A. T. Figueiredo and J. M. Bioucas-Dias. Restoration of Poissonian images using alternating direction optimization. IEEE Transactions on Image Processing, 19:3133?3145, 2010. [12] Q. Fu, H. Wang, and A. Banerjee. Bethe-ADMM for tree decomposition based parallel MAP inference. In UAI, 2013. [13] D. Gabay. Applications of the method of multipliers to variational inequalities. In Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems. M. Fortin and R. Glowinski, eds., North-Holland: Amsterdam, 1983. [14] T. Goldstein, X. Bresson, and S. Osher. Geometric applications of the split Bregman method: segmentation and surface reconstruction. Journal of Scientific Computing, 45(1):272?293, 2010. [15] T. Goldstein, B. Donoghue, and S. Setzer. Fast alternating direction optimization methods. CAM report 12-35, UCLA, 2012. [16] T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2009. [17] B. He and X. Yuan. On the O(1/n) convergence rate of the Douglas-Rachford alternating direction method. SIAM Journal on Numerical Analysis, 50:700?709, 2012. [18] F. L. Hitchcock. The distribution of a product from several sources to numerous localities. Journal of Mathematical Physics, 20:224?230, 1941. [19] M. Hong and Z. Luo. On the linear convergence of the alternating direction method of multipliers. ArXiv, 2012.

[20] K. C. Kiwiel. Proximal minimization methods with generalized Bregman functions. SIAM Journal on Control and Optimization, 35:1142?1168, 1995. [21] A. Nemirovski and D. Yudin. Problem Complexity and Method Efficiency in Optimization. Wiley, 1983. [22] Y. Nesterov. Gradient methods for minimizing composite objective function. Technical Report 76, Center for Operation Research and Economics (CORE), Catholic University of Louvain (UCL), 2007. [23] M. Telgarsky and S. Dasgupta. Agglomerative Bregman clustering. In ICML, 2012. [24] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Foundations and Trends in Machine Learning, 1:1?305, 2008. [25] H. Wang and A. Banerjee. Online alternating direction method. In ICML, 2012. [26] J. Yang and Y. Zhang. Alternating direction algorithms for L1-problems in compressive sensing. ArXiv, 2009.
9