A Dual Augmented Block Minimization Framework for Learning with Limited Memory

Authored by:

Ian En-Hsu Yen
Shou-De Lin
Shan-Wei Lin

Abstract

In past few years, several techniques have been proposed for training of linear Support Vector Machine (SVM) in limited-memory setting, where a dual block-coordinate descent (dual-BCD) method was used to balance cost spent on I/O and computation. In this paper, we consider the more general setting of regularized emph{Empirical Risk Minimization (ERM)} when data cannot fit into memory. In particular, we generalize the existing block minimization framework based on strong duality and emph{Augmented Lagrangian} technique to achieve global convergence for ERM with arbitrary convex loss function and regularizer. The block minimization framework is flexible in the sense that, given a solver working under sufficient memory, one can integrate it with the framework to obtain a solver globally convergent under limited-memory condition. We conduct experiments on L1-regularized classification and regression problems to corroborate our convergence theory and compare the proposed framework to algorithms adopted from online and distributed settings, which shows superiority of the proposed approach on data of size ten times larger than the memory capacity.

1 Paper Body

Nowadays data of huge scale are prevalent in many applications of statistical learning and data mining. It has been argued that model performance can be boosted by increasing both number of samples and features, and through crowdsourcing technology, annotated samples of terabytes storage size can be generated [3]. As a result, the performance of model is no longer limited by the sample size but the amount of available computational resources. In other words, the data size can easily go beyond the size of physical memory of available machines. Under this setting, most of learning algorithms become slow due to expensive I/O from secondary storage device [26]. When it comes to huge-scale data, two settings are often considered ? online and distributed learning. In the online setting, each sample is processed only once without storage, while in the distributed setting, one has several machines that can jointly fit the data into memory. However, the real cases are often not as extreme as these two ? there are usually machines that can fit part of the data, but not all of them. In this setting, an algorithm can only process a block of data at a time. Therefore, balancing the time spent on I/O and computation becomes the key issue [26]. Although one can employ an online-fashioned learning algorithm in this setting, it has been observed that online method requires large number of epoches to achieve comparable performance to batch method, and at each epoch it spends most of time on I/O instead of computation [2, 21, 26]. The situation for online method could become worse for problem of non-smooth, non-strongly convex objective function, where a qualitatively slower convergence of online method is exhibited [15, 16] than that proved for strongly-convex problem like SVM [14]. In the past few years, several algorithms have been proposed to solve large-scale linear Support Vector Machine (SVM) in the limited memory setting [2, 21, 26]. These approaches are based on a dual 1

Block Coordinate Descent (dual-BCD) algorithim, which decomposes the original problem into a series of block sub-problems, each of them requires only a block of data loaded into memory. The approach was proved linearly convergent to the global optimum, and demonstrated fast convergence empirically. However, the convergence of the algorithm relies on the assumption of a smooth dual problem, which, as we show, does not hold generally for other regularized Empirical Risk Minimizaton (ERM) problem. As a result, although the dual-BCD approach can be extended to the more general setting, it is not globally convergent except for a class of problems with L2-regularizer. In this paper, we first show how to adapt the dual block-coordinate descnet method of [2, 26] to the general setting of regularized Empirical Risk Mimization (ERM), which subsumes most of supervised learning problems ranging from classification, regression to ranking and recommendation. Then we discuss the convergence issue arises when the underlying ERM is not strongly-convex. A Primal Proximal Point (or Dual Augmented Lagrangian) method is then proposed to address this issue, which as we show, results in a block minimization algorithm with global convergence to optimum for convex regularized ERM problems. The framework is flexible in the sense that, given a solver working under sufficient-memory condition, it can be integrated into the block minimization framework to obtain a solver globally convergent under limited-memory condition. We conduct experiments on L1-regularized classification and regression problems to corroborate our convergence theory, which shows that the proposed simple dual-augmented technique changes the convergence behavior dramatically. We also compare the proposed framework to algorithms adopted from online and distributed settings. In particular, we describe how to adapt a distributed optimization framework ? Alternating Direction Method of Multiplier (ADMM) [1] ? to the limitedmemory setting, and show that, although the adapted algorithm is effective, it is not as efficient as the proposed framework specially designed for limited-memory setting. Note our experiment does not adapt into comparison some recently proposed distributed learning algorithms (CoCoA etc.) [7, 10]

that only apply to ERM with L2-regularizer or some other distributed method designed for some specific loss function [19].

2

Problem Setup

In this work, we consider the regularized Empirical Risk Minimization problem, which given a data set D = {(?n , y n)}N n=1 , estimates a model through

min

w?Rd ,?n ?Rp

s.t.

F (w, ?) = R(w) +

N X n=1

Ln (? n)

(1)

?n w = ? n , n ? [N]

where w ? Rd is the model parameter to be estimated, ?n is a p by d design matrix that encodes features of the n-th data sample, Ln (? n) is a convex loss function that penalizes the discrepancy between ground truth and prediction vector ? n ? Rp , and R(w) is a convex regularization term penalizing model complexity. The formulation (1) subsumes a large class of statistical learning problems ranging from classification [27], regression [17], ranking [8], and convex clustering [24]. For example, in classification problem, we have p = —Y— where YPconsists of the set of all possible labels and Ln (?) can be defined as the logistic loss Ln (?) = log(k?Y exp(?k)) ? ?yn as in logistic regression or the hinge loss Ln (?) = maxk?Y (1 ? ?k,yn + ?k ? ?yn) as used in support vector machine; in a (multi-task) regression problem, the target variable consists of K real values Y = RK , the prediction vector has p = K dimensions, and a square loss Ln (?) = 21 k??y n k22 is often used. There are also a variety of regularizers R(w) employed in different applications, which includes the L2-regularizer R(w) = ?2 kwk2 in ridge regression, L1-regularizer R(w) = ?kwk1 in Lasso, nuclear-norm R(w) = ?kwk? in matrix completion, and a family of structured group norms R(w) = ?kwkG [11]. Although the specific form of Ln (?), R(w) does not affect the implementation of the limited-memory training procedure, two properties of the functions ? strong convexity and smoothness ? have key effects on the behavior of the block minimization algorithm. 2

Definition 1 (Strong Convexity). A function f (x) is strongly convex iff it is lower bounded by a simple quadratic function m f (y) ? f (x) + ?f (x)T (y ? x) + kx ? yk2 (2) 2 for some constant m ¿ 0 and ?x, y ? dom(f). Definition 2 (Smoothness). A function f (x) is smooth iff it is upper bounded by a simple quadratic function M f (y) ? f (x) + ?f (x)T (y ? x) + kx ? yk2 (3) 2 for some constant 0 ? M ¡ ? and ?x, y ? dom(f). For instance, the square loss and logistic loss are both smooth and strongly convex 1 , while the hingeloss satisfies neither of them. On the other hand, most of regularizers such as L1-norm, structured group norm, and nuclear norm are neither smooth nor strongly convex, except for the L2-regularizer, which satifies both. In the following we will demonstrate the effects of these properties to Block Minimization algorithms. Throughout this paper, we will assume that a solver for (1) that works in sufficient-memory condition is given, and our task is to design an algorithmic framework that integrates with the solver to efficiently solve (1) when data cannot fit into memory. We will assume, however, that the d-dimensional parameter vector w can be fit into memory.

3

Dual Block Minimization

In this section, we extend the block minimization framework of [26] from linear SVM to the general setting of regularized ERM (1).The dual of (1) can be expressed as min

??Rd ,?n ?Rp

R? (??) +

L?n (?n)

n=1 N X

s.t.

N X

?Tn ?n

(4)

=?

n=1

where R? (??) is the convex conjugate of R(w) and L?n (?n) is the convex conjugate of Ln (? n). The block minimization algorithm of [26] basically performs a dual Block-Coordinate Descent (dual-BCD) over (4) by dividing the whole data set D into K blocks DB1 , ..., DBK , and optimizing a block of dual variables (?Bk , ?) at a time, where DBk = {(?n , y n)}n?Bk and ?Bk = {?n —n ? Bk }. In [26], the dual problem (4) is derived explicitly in order to perform the algorithm. However, for many sparsity-inducing regularizer such as L1-norm and nuclear norm, it is more efficient and convenient to solve (1) in the primal [6, 28]. Therefore, here instead of explicitly forming the dual problem, we express it implicitly as G(?) = min L(?, w, ?), (5) w,?

where L(?, w, ?) is the Lagrangian function of (1), and maximize (5) w.r.t. a block of variables ?Bk from the primal instead of dual by strong duality

max min L(?, w, ?) = min max L(?, w, ?) (6) ?Bk

w,?

w,?

?Bk

?tBj }j6=k

with other dual variables {?Bj = fixed. The maximization of dual variables ?Bk in (6) then enforces the primal equalities ?n w = ? n , n ? Bk , which results in the block minimization problem X min R(w) + Ln (? n) + ?tT Bk w w?Rd ,?n ?Rp (7) n?Bk s.t. ? n w = ? n , n ? Bk , 1 The logistic loss is strongly convex when its input ? are within a bounded range, which is true as long as we have a non-zero regularizer R(w).

3

P T t where ?tBk = n?B / k have been dropped since they / k ?n ?n . Note that, in (7), variables {? n }n?B are not relevant to the block of dual variables ?Bk , and thus given the d dimensional vector ?tBk , one can solve (7) without accessing data {(?n , y n)}n?B / k outside the block Bk . Throughout the PN dual-BCD algorithm, we maintain d-dimensional vector ?t = n=1 ?Tn ?tn and compute ?tB via X ?tB = ?t ? ?Tn ?tn (8) n?Bk

in the beginning of solving each block subproblem (7). Since subproblem (7) is of the same form to the original problem (1) except for one additional linear augmented term ?TBk w, one can adapt the solver of (1) to solve (7) easily by providing an augmented version of the gradient ?w F? (w, ?) = ?w F (w, ?) + ?tBk to the solver, where F? (.) denotes the function with augmented terms and F (.) denotes the function without augmented terms. Note the augmented term ?tBk is constant and separable w.r.t. coordinates, so it adds little overhead to the solver. After obtaining solution (w? , ? ?Bk) from (7), we can derive the corresponding optimal dual variables ?Bk for (6) according to the KKT condition and maintain ? subsequently by ?t+1 = ??n Ln (? ?n), n ? Bk n X ?t+1 = ?tBk + ?Tn ?t+1 n .

(9) (10)

n?Bk

The procedure is summarized in Algorithm 1, which requires a total memory capacity of O(d + —DBk — + p—Bk —). The factor d comes from the storage of ?t , wt , factor —DBk — comes from the storage of data block, and the factor p—Bk — comes from the storage of ?Bk . Note this requires the same space complexity as that required in the original algorithm proposed for linear SVM [26], where p = 1 for the binary classification setting.

4

Dual-Augmented Block Minimization

The Block Minimization Algorithm 1, though can be applied to the general regularized ERM problem (1), it is not guaranteed that the sequence {?t }? t=0 produced by Algorithm 1 converges to global optimum of (1). In fact, the global convergence of Algorithm 1 only happens for some special cases. One sufficient condition for the global convergence of a Block-Coordinate Descent algorithm is that the terms in objective function that are not separable w.r.t. blocks must be smooth (Definition 2). The dual objective function (4) (expressed using only ?) comprises two terms PN PN R? (? n=1 ?Tn ?n) + n=1 L?n (?n), where second term is separable w.r.t. to {?n }N n=1 , K and thus is also separable w.r.t. {?Bk }k=1 , while the first term couples variables ?B1 , ..., ?BK involving all the blocks. As a result, if R? (??) is a smooth function according to Definition 2, then Algorithm 1 has global convergence to the optimum. However, the following theorem states this is true only when R(w) is strongly convex. Theorem 1 (Strong/Smooth Duality). Assume f (.) is closed and convex. Then f (.) is smooth with 1 parameter M if and only if its convex conjugate f ? (.) is strongly convex with parameter m = M . A proof of above theorem can be found in [9]. According to Theorem 1, the Block Minimization Algorithm 1 is not globally convergent if R(w) is not strongly convex, which however, is the case for most of regularizers other than the L2-norm R(w) = 21 kwk2 , as discussed in Section 2. In this section, we propose a remedy to this problem, which by a Dual-Augmented Lagrangian method (or equivalently, Primal Proximal Point method), creates a dual objective function of desired property that iteratively approaches the original objective (1), and results in fast global convergence of the dual-BCD approach. 4

Algorithm 1 Dual Block Minimization 1. Split data D into blocks B1 , B2 , ..., BK . 2. Initialize ?0 = 0. for t = 0, 1, ... do 3.1. Draw k uniformly from [K]. 3.2. Load DBk and ?tBk into memory. 3.3. Compute ?tBk from (8). 3.4. Solve (7) to obtain (w? , ? ?Bk). 3.5. Compute ?t+1 Bk by (9). 3.6. Maintain ?t+1 through (10). 3.7. Save ?t+1 Bk out of memory. end for

4.1

Algorithm 2 Dual-Aug. Block Minimization 1. Split data D into blocks B1 , B2 , ..., BK . 2. Initialize w0 = 0, ?0 = 0. for t = 0, 1, ... (outer iteration) do for s = 0, 1, ..., S do 3.1.1. Draw k uniformly from [K]. 3.1.2. Load DBk , ?sBk into memory. 3.1.3. Compute ?sBk from (15). 3.1.4. Solve (14) to obtain (w? , ? ?Bk). 3.1.5. Compute ?s+1 Bk by (16). 3.1.6. Maintain ?s+1 through (17). 3.1.7. Save ?s+1 Bk out of memory. end for 3.2. wt+1 = w? (?S). end for

Algorithm

The Dual Augmented Lagrangian (DAL) method (or equivalently, Proximal Point Method) modifies the original problem by introducing a sequence of Proximal Maps 1 wt+1 = arg min F (w) + kw ? wt k2 , (11) 2? w t where F (w) denotes the ERM problem (1) Under this simple modification, instead of doing BlockCoordinate Descent in the dual of original problem (1), we perform Dual-BCD on the proximal subproblem (11). As we show in next section, the dual formulation of (11) has the required property for global convergence of the Dual BCD algorithm ? all terms involving more than one block of variables ?Bk are smooth. Given the current iterate wt , the Dual-Augmented Block Minimization algorithm optimizes the dual of proximal-point problem (11) w.r.t. one block of variables ?Bk at a (t,s) time, keeping others fixed {?Bj = ?Bj }j6=k : max min L(w, ?, ?) = min max L(w, ?, ?) ?Bk w,?

where L(.) is the Lagrangian of (11) L(w, ?, ?) = F (w, ?) +

w,? ?Bk

N X

?Tn (?n w ? ? n) +

n=1

1 kw ? wt k2 . 2?t

(12)

(13)

Once again, the maximization w.r.t. ?Bk in (12) enforces the equalities ?n w = ? n , n ? Bk and thus leads to a primal sub-problem involving only data in block Bk : X 1 (t,s)T min R(w) + Ln (? n) + ?Bk w + kw ? wt k2 2?t w?Rd ,?n ?Rp (14) n?Bk s.t. ?n w = ? n , n ? Bk , P T (t,s) where = . Note that (14) is almost the same as (7) except that it has a n?B / k ?n ? n proximal-point augmented term. Therefore, one can follow the same procedure as in Algorithm 1 to PN (t,s) maintain the vector ?(t,s) = n=1 ?Tn ?n and computes X (t,s) ?Bk = ?(t,s) ? ?Tn ?(t,s) (15) n (t,s) ?Bk

n?Bk

before solving each block subproblem (14). After obtaining solution (w? , ? ?Bk) from (14), we update dual variables ?Bk as ?(t,s+1) = ??n Ln (? ?n), n

? Bk . (16) n and maintain ? subsequently as X (t,s) ?(t,s+1) = ?Bk + ?Tn ?(t,s+1) . (17) n n?Bk

5

The sub-problem (14) is of similar form to the original ERM problem (1). Since the augmented term is a simple quadratic function separable w.r.t. each coordinate, given a solver for (1) working in sufficient-memory condition, one can easily adapt it by modifying ?w F? (w, ?) = ?w F (w, ?) + ?tBk + (w ? wt)/?t ?2w F? (w, ?) = ?2w F (w, ?) + I/?t , where F? (.) denotes the function with augmented terms and F (.) denotes the function without augmented terms. The Block Minimization procedure is repeated until every sub-problem (14) reaches a tolerance in . Then the proximal point method update wt+1 = w? (?(t,s)) is performed, where w? (?(t,s)) is the solution of (14) for the latest dual iterate ?(t,s) . The resulting algorithm is summarized in Algorithm 2. 4.2

Analysis

In this section, we analyze the convergence rate of Algorithm 2 to the optimum of (1). First, we show that the proximal-point formulation (11) has a dual problem with desired property for the global convergence of Block-Coordinate Descent. In particular, since the dual of (11) takes the form minp

?n ?R

? ? (? R

N X

?Tn ?n) +

n=1

N X

L?n (?n)

(18)

n=1

? ? (.) is the convex conjugate of R(w) ? ? where R = R(w)+ 2?1 t kw ?wt k2 , and since R(w) is strongly ? ? convex with parameter m = 1/?t , the convex conjugate R (.) is smooth with parameter M = ?t according to Theorem 1. Therefore, (18) is in the composite form of a convex, smooth function plus a convex, block-separable function. This type of function has been widely studied in the literature of Block-Coordinate Descent [13]. In particular, one can show that a Block-Coordinate Descent applied on (18) has global convergence to optimum with a fast rate by the following theorem. Theorem 2 (BCD Convergence). Let the sequence {?s }? s=1 be the iterates produced by Block Coordinate Descent in the inner loop of Algorithm 2, and K be the number of blocks. Denote F? ? (?) ? the optimal value of (18). Then with probability 1??, as the dual objective function of (18) and F?opt ? F? ? (?0) ? F?opt) (19) ? for some constant ? ¿ 0 if (i) Ln (.) is smooth, or (ii) Ln (.) is polyhedral function and R(.) is also polyhedral or smooth. Otherwise, for any convex Ln (.), R(.) we have ? F? ? (?s) ? F?opt ? , for s ? ?K log(

? F? ? (?0) ? F?opt cK ? F? ? (?s) ? F?opt ? , for s ? log( )

?

(20)

for some constant c ¿ 0. Note the above analysis (in appendix) does not assume exact solution of each block subproblem. Instead, it only assumes each block minimization step leads to a dual ascent amount proportional to that produced by a single (dual) proximal gradient ascent step on the block of dual variables. For the outer loop of Primal Proximal-Point (or Dual Augmented Lagrangian) iterates (11), we show the following convergence theorem. Theorem 3 (Proximal Point Convergence). Let F (w) be objective of the regularized ERM problem (1), and R = maxv maxw {kv ? wk : F (w) ? F (w0), F (v) ? F (w0)} be the radius of initial level set. The sequence {wt }? t=1 produced by the Proximal-Point update (11) with ?t = ? has ? F (wt+1) ? Fopt ? , for t ? ? log( ). (21)

for some constant ?, ? ¿ 0 if both Ln (.) and R(.) are (i) strictly convex and smooth or (ii) polyhedral. Otherwise, for any convex F (w) we have F (wt+1) ? Fopt ? R2 /(2?t). 6

The following theorem further shows that solving sub-problem (11) inexactly with tolerance /t suffices for convergence to overall precision, where t is the number of outer iterations required. Theorem 4 (Inexact Proximal Map). Suppose, for a given dual iterate wt , each sub-problem (11) ? t+1 has is solved inexactly s.t. the solution w ? t+1 ? prox?t F (wt)k ? 0 . kw ? t }? {w t=1

Then let be the sequence of iterates produced by inexact proximal updates and as that generated by exact updates. After t iterations, we have ? t ? wt k ? t0 . kw

(22) {wt }? t=1 (23)

Note for Ln (.), R(.) being strictly convex and smooth, or polyhedral, t is of order O(log(1/)), and thus it only requires O(K log(1/) log(t/)) = O(K log2 (1/)) overall number of block minimization steps to achieve suboptimality. Otherwise, as long as Ln (.) is smooth, for any convex regularizer R(.), t is of order O(1/), so it requires O(K(1/) log(t/)) = O(K log(1/)) total

number of block minimization steps. 4.3

Practical Issues

4.3.1 Solving Sub-Problem Inexactly While the analysis in Section 4.2 assumes exact solution of subproblems, in practice, the Block Minimization framework does not require solving subproblem (11), (14) exactly. In our experiments, it suffices for the fast convergence of proximal-point update (11) to solve subproblem (14) for only a single pass of all blocks of variables ?B1 ,..., ?BK , and limit the number of iterations the designated solver spends on each subproblem (7), (14) to be no more than some parameter Tmax . 4.3.2 Random Selection w/o Replacement In Algorithm 1 and 2, the block to be optimized is chosen uniformly at random from k ? {1, ..., K}, which eases the analysis for proving a better convergence rate [13]. However, in practice, to avoid unbalanced update frequency among blocks, we do random sampling without replacement for both Algorithm 1 and 2, that is, for every K iterations, we generate a random permutation ?1 , ..., ?K of block index 1, .., K and optimize block subproblems (7), (14) according to the order ?1 , .., ?K . This also eases the checking of inner-loop stopping condition. 4.3.3 Storage of Dual Variables Both the algorithms 1 and 2 need to store the dual variables ?Bk into memory and load/save them from/to some secondary storage units, which requires a time linear to p—Bk —. For some problems, such as multi-label classification with large number of labels or structured prediction with large number of factors, this can be very expensive. In this situation, one can instead maintain ?B?k = P T ? k has I/O and storage cost linear to d, which can be n?Bk ?n ?n = ? ? ?Bk directly. Note ?B much smaller than p—Bk — in a low-dimensional problem.

5

Experiment

In this section, we compare the proposed Dual Augmented Block Minimization framework (Algorithm 2) to the vanilla Dual Block Coordinate Descent algorithm [26] and methods adopted from Online and Distributed Learning. The experiments are conducted on the problem of L1-regularized L2-loss SVM [27] and the (Lasso) (L1-regularized Regression) problem [17] in the limited-memory setting with data size 10 times larger than the available memory. For both problems, we use stateof-the-art randomized coordinate descent method [13, 27] as the solver for solving sub-problems (7), (14), (59), (63), and we set parameter ?t = 1, ? = 1 (of L1-regularizer) for all experiments. Four public benchmark data sets are used? webspam, rcv1-binary for classification and year-pred, E2006 for regression, which can be obtained from the LIBSVM data set collections. For year-pred and E2006, the features are generated from Random Fourier Features [12, 23] that approximate the effect of Gaussian RBF kernel. Table 1 summarizes the data statistics. The algorithms in comparison and their shorthands are listed below, where all solvers are implemented in C/C++ and run on 64-bit machine with 2.83GHz Intel(R) Xeon(R) CPU. We constrained the process to use no more than 1/10 of memory required to store the whole data. ? OnlineMD: Stochastic Mirror Descent method specially designed for L1-regularized problem proposed in [15] with step size chosen from 10?2 -102 for best performance. 7

Table 1: Data Statistics: Summary of data statistics when stored using sparse format. The last two columns specify memory consumption in (MB) of the whole data and that of a block when data is split into K = 10 partitions. Data #train #test dimension #non-zeros Memory Block webspam 315,000 31,500 680,714 1,174,704,031 20,679 2,068 rcv1 202,420 20,242 7,951,176 656,977,694 12,009 1,201 year-pred 463,715 51,630 2,000 927,893,715 13,702 1,370 E2006 16,087 3,308 30,000 8,088,636 8,088 809 Figure 1: Relative function value difference to the optimum and Testing RMSE (Accuracy) on LASSO (top) and L1-regularized L2-SVM (bottom). (RMSE best for year-pred: 9.1320; for E2006: 0.4430), (Accuracy best for for webspam: 0.4761%; best for rcv1: 2.213%). year?pred?obj

year?rmse ADMM BC?ADMM DA?BCD D?BCD onlineMD

e2006?obj

ADMM BC?ADMM DA?BCD D?BCD onlineMD

ADMM BC?ADMM DA?BCD D?BCD onlineMD

?1

10

?2

ADMM BC?ADMM DA?BCD D?BCD onlineMD
obj
RMSE
10
rmse
objective
e2006?rmse 0
10
0
10
?2
10
?3
10
?3
10 ?2
1000
2000
3000 4000 time
5000
10
6000
?1
1000
webspam?obj 1
10
3000 4000 time
5000
6000
1000
3000 time
4000
1
10
ADMM BC?ADMM DA?BCD D?BCD onlineMD
1
10
2000
5000
10
6000
1000
2000
rcv1?obj
webspam?error ADMM BC?ADMM DA?BCD D?BCD onlineMD
0

2000
3000 time
4000
ADMM BC?ADMM DA?BCD D?BCD onlineMD
0
10
5000
6000
rcv1?error ADMM BC?ADMM DA?BCD D?BCD onlineMD
10
0
10
error
obj
obj
error
10 0
?1
10
?1
10
?2
10
?1
10 ?2
10
2000
4000
6000 time
8000
10000
12000
2000
4000
6000 time
8000
10000
2000
12000
4000
6000 8000 time
10000 12000
2000
4000
6000 8000 time
10000 12000

? D-BCD2 : Dual Block-Coordinate Descent method (Algorithm 1). ? DA-BCD: Dual-Augmented Block Minimization (Algorithm 2). ? ADMM: ADMM for limited-memory learning (Algorithm 3 in appendix-B). ? BC-ADMM: Block-Coordinate ADMM that updates a randomly chosen block of dual variables at a time for limited-memory learning (Algorithm 4 in appendix-B) . We use wall clock time that includes both I/O and computation as measure for training time in all experiments. In Figure 5, three measures are plotted versus the training time: Relative objective function difference to the optimum, Testing RMSE and Accuracy. Figure 5 shows the results, where as expected, the dual Block Coordinate Descent (D-BCD) method without augmentation cannot improve the objective after certain number of iterations. However, with extremely simple modification, the Dual-Augmented Block Minimization (DA-BCD) algorithm becomes not only globally convergent but with a rate several times faster than other approaches. Among all methods, the convergence of Online Mirror Descent (SMIDAS) is significantly slower, which is expected since (i) the online Mirror Descent on a non-smooth, non-strongly convex function converges at a rate qualitatively slower than the linear convergence rate of DA-BCD and ADMM [15, 16], and (ii) Online method does not utilize the available memory capacity and thus spends unbalanced time on I/O and computation. For methods adopted from distributed optimization, the experiment shows BC-ADMM consistently, but only slightly, improves ADMM, and both of them converge much slower than the DA-BCD approach, presumably due to the conservative updates on the dual variables. Acknowledgement We thank to the support of Telecommunication Lab., Chunghwa Telecom Co., Ltd via TL-103-8201, AOARD via No. FA2386-13-1-4045, Ministry of Science and Technology, National Taiwan University and Intel Co. via MOST102-2911-I-002-001, NTU103R7501, 1022923-E-002-007-MY2, 102-2221-E-002-170, 103-2221-E-002-104-MY2. 2 The objective value obtained from D-BCD fluctuates a lot, in figures we plot the lowest values achieved by D-BCD from the beginning to time t.

8

2 References

[1] 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, 2011. [2] K. Chang and D. Roth. Selective block minimization for faster convergence of limited memory large-scale linear models. In SIGKDD. ACM, 2011. [3] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. F. Fei. Imagenet: A large-scale hierarchical image database. In CVPR, 2009. [4] A. Hoffman. On approximate solutions of systems of linear inequalities. Journal of Research of the National Bureau of Standards, 1952. [5] M. Hong and Z. Luo. On the linear convergence of the alternating direction method of multipliers, 2012. [6] C. Hsieh, I. Dhillon, P. Ravikumar, S. Becker, and P. Olsen. Quic & dirty: A quadratic approximation approach

for dirty statistical models. In NIPS, 2014. [7] M. Jaggi, V. Smith, M. Tak?ac, J. Terhorst, S. Krishnan, T. Hofmann, and M. Jordan. Communicationefficient distributed dual coordinate ascent. In NIPS, 2014. [8] T. Joachims. A support vector method for multivariate performance measures. In ICML, 2005. [9] S. Kakade, S. Shalev-Shwartz, and A. Tewari. Applications of strong convexity?strong smoothness duality to learning with matrices. CoRR, 2009. [10] C. Ma, V. Smith, M. Jaggi, M. Jordan, P. Richt?arik, and M. Tak?ac? . Adding vs. averaging in distributed primal-dual optimization. ICML, 2015. [11] G. Obozinski, L. Jacob, and J. Vert. Group lasso with overlaps: the latent group lasso approach. arXiv preprint, 2011. [12] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In NIPS, 2007. [13] P. Richt?arik and M. Tak?ac? . Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Mathematical Programming, 2014. [14] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal estimated sub-gradient solver for svm. Mathematical programming, 2011. [15] S. Shalev-Shwartz and A. Tewari. Stochastic methods for l1-regularized loss minimization. JMLR, 2011. [16] N. Srebro, K. Sridharan, and A. Tewari. On the universality of online mirror descent. In NIPS, 2011. [17] R. Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, 1996. [18] R. Tomioka, T. Suzuki, and M. Sugiyama. Super-linear convergence of dual augmented lagrangian algorithm for sparsity regularized estimation. JMLR, 2011. [19] I. Trofimov and A. Genkin. Distributed coordinate descent for l1-regularized logistic regression. arXiv preprint, 2014. [20] P. Wang and C. Lin. Iteration complexity of feasible descent methods for convex optimization. JMLR, 2014. [21] I. Yen, C. Chang, T. Lin, S., and S. Lin. Indexed block coordinate descent for large-scale linear classification with limited memory. In SIGKDD. ACM, 2013. [22] I. Yen, C. Hsieh, P. Ravikumar, and I. Dhillon. Constant nullspace strong convexity and fast convergence of proximal methods under high-dimensional settings. In NIPS, 2014. [23] I. Yen, T. Lin, S. Lin, P. Ravikumar, and I. Dhillon. Sparse random feature algorithm as coordinate descent in hilbert space. In NIPS, 2014. [24] I. Yen, X. Lin, K. Zhong, P. Ravikumar, and I. Dhillon. A convex exemplar-based approach to MADBayes dirichlet process mixture models. In ICML, 2015. [25] I. Yen, K. Zhong, C. Hsieh, P. Ravikumar, and I. Dhillon. Sparse linear programming via primal and dual augmented coordinate descent. In NIPS, 2015. [26] H. Yu, C. Hsieh, . Chang, and C. Lin. Large linear classification when data cannot fit in memory. SIGKDD, 2010. [27] G. Yuan, K. Chang, C. Hsieh, and C. Lin. A comparison of optimization methods and software for large-scale L1-regularized linear classification. JMLR, 2010. [28] K. Zhong, I. Yen, I. Dhillon, and P. Ravikumar. Proximal quasi-Newton for computationally intensive l1-regularized m-estimators. In NIPS, 2014.

9