Accelerated Gradient Methods for Stochastic Optimization and Online Learning

Authored by:

James T. Kwok
Chonghai Hu
Weike Pan

Abstract

Regularized risk minimization often involves non-smooth optimization, either because of the loss function (e.g., hinge loss) or the regularizer (e.g., ell_1-regularizer). Gradient descent methods, though highly scalable and easy to implement, are known to converge slowly on these problems. In this paper, we develop novel accelerated gradient methods for stochastic optimization while still preserving their computational simplicity and scalability. The proposed algorithm, called SAGE (Stochastic Accelerated GradiEnt), exhibits fast convergence rates on stochastic optimization with both convex and strongly convex objectives. Experimental results show that SAGE is faster than recent (sub)gradient methods including FOLOS, SMIDAS and SCD. Moreover, SAGE can also be extended for online learning, resulting in a simple but powerful algorithm.

1 Paper Body

Risk minimization is at the heart of many machine learning algorithms. Given a class of models parameterized by w and a loss function ?(?, ?), the goal is to minimize EXY [?(w; X, Y)] w.r.t. w, where the expectation is over the joint distribution of input X and output Y . However, since the joint distribution is typically unknown in practice, a surrogate problem is to replace the expectation by its empirical average on a training sample {(x1 , y1), . . . , (xm , ym)}. Moreover, a regularizer ?(?) is often added for well-posedness. This leads to the minimization of the regularized risk m

min w

1 X ?(w; xi , yi) + ??(w), m i=1

(1)

where ? is a regularization parameter. In optimization terminology, the deterministic optimization problem in (1) can be considered as a sample average approximation (SAA) of the corresponding stochastic optimization problem: min EXY [?(w; X, Y)] + ??(w). w

(2)

Since both ?(?, ?) and ?(?) are typically convex, (1) is a convex optimization problem which can be conveniently solved even with standard off-the-shelf optimization packages. However, with the proliferation of data-intensive applications in the text and web domains, data sets with millions or trillions of samples are nowadays not uncommon. Hence, off-the-shelf optimization solvers are too slow to be used. Indeed, even tailor-made softwares for specific models, such as the sequential minimization optimization (SMO) method for the SVM, have superlinear computational 1

complexities and thus are not feasible for large data sets. In light of this, the use of stochastic methods have recently drawn a lot of interest and many of these are highly successful. Most are based on (variants of) the stochastic gradient descent (SGD). Examples include Pegasos [1], SGD-QN [2], FOLOS [3], and stochastic coordinate descent (SCD) [4]. The main advantages of these methods are that they are simple to implement, have low per-iteration complexity, and can scale up to large data sets. Their runtime is independent of, or even decrease with, the number of training samples [5, 6]. On the other hand, because of their simplicity, these methods have a slow convergence rate, and thus may require a large number of iterations. While standard gradient schemes have a slow convergence rate, they can often be ?accelerated?. This stems from the pioneering work of Nesterov in 1983 [7], which is a deterministic algorithm for smooth optimization. Recently, it is also extended for composite optimization, where the objective has a smooth component and a non-smooth component [8, 9]. This is particularly relevant to machine learning since the loss ? and regularizer ? in (2) may be non-smooth. Examples include loss functions such as the commonly-used hinge loss used in the SVM, and regularizers such as the popular ?1 penalty in Lasso [10], and basis pursuit. These accelerated gradient methods have also been successfully applied in the optimization problems of multiple kernel learning [11] and trace norm minimization [12]. Very recently, Lan [13] made an initial attempt to further extend this for stochastic composite optimization, and obtained the convergence rate of

? O L/N 2 + (M + ?)/ N . (3)

Here, N is the number of iterations performed by the algorithm, L is the Lipschitz parameter of the gradient of the smooth term in the objective, M is the Lipschitz parameter of the nonsmooth term, and ? is the variance of the stochastic subgradient. Moreover, note that the first term of (3) is related to the smooth component in the objective while the second term is related to the nonsmooth component. Complexity results [14, 13] show that (3) is the optimal convergence rate for any iterative algorithm solving stochastic (general) convex composite optimization.

However, as pointed out in [15], a very useful property that can improve the convergence rates in machine learning optimization problems is strong convexity. For example, (2) can be strongly convex either because of the strong convexity of ? (e.g., log loss, square loss) or ? (e.g., ?2 regularization). On the other hand, [13] is more interested in general convex optimization problems and so strong convexity is not utilized. Moreover, though theoretically interesting, [13] may

be of limited practical use as (1) the stepsize in its update rule depends on the often unknown ?; and (2) the number of iterations performed by the algorithm has to be fixed in advance. Inspired by the successes of Nesterov?s method, we develop in this paper a novel accelerated subgradient stochastic composite optimization. It achieves the optimal convergence rate scheme for ?

2 of O L/N + ?/ N for general convex objectives, and O (L + ?)/N 2 + ???1 /N for ?strongly convex objectives. Moreover, its per-iteration complexity is almost as low as that for standard (sub)gradient? methods. Finally, we also extend the accelerated gradient scheme to online learning. We obtain O(N) regret for general convex problems and O(log N) regret for strongly convex problems, which are the best regret bounds currently known for these problems.

2

Setting and Mathematical Background

First, we recapitulate a few notions in convex analysis. (Lipschitz continuity) A function f (x) is L-Lipschitz if kf (x) ? f (y)k ? Lkx ? yk. Lemma 1. [14] The gradient of a differentiable function f (x) is Lipschitz continuous with Lipschitz parameter L if, for any x and y, L (4) f (y) ? f (x) + h?f (x), y ? xi + kx ? yk2 . 2 (Strong convexity) A function ?(x) is ?-strongly convex if ?(y) ? ?(x)+hg(x), y?xi+ ?2 ky?xk2 for any x, y and subgradient g(x) ? ??(x). Lemma 2. [14] Let ?(x) be ?-strongly convex and x? = arg minx ?(x). Then, for any x, ? ?(x) ? ?(x?) + kx ? x? k2 . (5) 2 2

We consider the following stochastic convex stochastic optimization problem, with a composite objective function min{?(x) ? E[F (x, ?)] + ?(x)}, (6) x

where ? is a random vector, f (x) ? E[F (x, ?)] is convex and differentiable, and ?(x) is convex but non-smooth. Clearly, this includes the optimization problem (2). Moreover, we assume that the gradient of f (x) is L-Lipschitz and ?(x) is ?-strongly convex (with ? ? 0). Note that when ?(x) is smooth (?(x) = 0), ? lower bounds the smallest eigenvalue of its Hessian. Recall that in smooth optimization, the gradient update xt+1 = xt ? ??f (xt) on a function f (x) can be seen as proximal regularization of the linearized f at the current iterate xt [16]. In other 1 words, xt+1 = arg minx (h?f (xt), x ? xt i + 2? kx ? xt k2). With the presence of a non-smooth component, we have the following more general notion. (Gradient mapping) [8] In minimizing f (x) + ?(x), where f is convex and differentiable and ? is convex and non-smooth,

1 kx ? xt k2 + ?(x) (7) xt+1 = arg min h?f (x), x ? xt i + x 2?

is called the generalized gradient update, and ? = ?1 (xt ? xt+1) is the (generalized) gradient mapping. Note that the quadratic approximation is made to the smooth component only. It can be shown that the gradient mapping is analogous to the gradient in smooth convex optimization [14, 8]. This is also a common construct used in recent stochastic subgradient methods [3, 17].

3

Accelerated Gradient Method for Stochastic Learning

Let G(xt , ?t) ? ?x F (x, ?t)—x=xt be the stochastic gradient of F (x, ?t). We assume that it is an unbiased estimator of the gradient ?f (x), i.e., E? [G(x, ?)] = ?f (x). Algorithm 1 shows the proposed algorithm, which will be called SAGE (Stochastic Accelerated GradiEnt). It involves the updating of three

sequences {xt }, {yt } and {zt }. Note that yt is the generalized gradient update, and xt+1 is a convex combination of yt and zt . The algorithm also maintains two parameter sequences {?t } and {Lt }. We will see in Section 3.1 that different settings of these parameters lead to different convergence rates. Note that the only expensive step of Algorithm 1 is the computation of the generalized gradient update yt , which is analogous to the subgradient computation in other subgradient-based methods. In general, its computational complexity depends on the structure of ?(x). As will be seen in Section 3.3, this can often be efficiently obtained in many regularized risk minimization problems. Algorithm 1 SAGE (Stochastic Accelerated GradiEnt). Input: Sequences {Lt } and {?t }. Initialize: y?1 = z?1 = 0, ?0 = ?0 = 1. L0 = L + ?. for t = 0 to N do xt = (1 ? ?t)yt?1 + ?t zt?1 .

yt = arg minx hG(xt , ?t), x ? xt i + L2t kx ? xt k2 + ?(x) . zt = zt?1 ? (Lt ?t + ?)?1 [Lt (xt ? yt) + ?(zt?1 ? xt)]. end for Output yN . 3.1

Convergence Analysis

Define ?t ? G(xt , ?t) ? ?f (xt). Because of the unbiasedness of G(xt , ?t), E?t [?t] = 0. In the following, we will show that the value of ?(yt) ? ?(x) can be related to that of ?(yt?1) ? ?(x) for any x. Let ?t ? Lt (xt ? yt) be the gradient mapping involved in updating yt . First, we introduce the following lemma. Lemma 3. For t ? 0, ?(x) is quadratically bounded from below as 2Lt ? L ? k?t k2 . ?(x) ? ?(yt) + h?t , x ? xt i + kx ? xt k2 + h?t , yt ? xi + 2 2L2t 3

Proposition 1. Assume that for each t ? 0, k?t k? ? ? and Lt ¿ L, then

Lt ?t2 + ??t kx ? zt k2 2 ?2 Lt ?t2 kx ? zt?1 k2 + + ?t h?t , x ? zt?1 i. ? (1 ? ?t)[?(yt?1) ? ?(x)] + 2 2(Lt ? L)

?(yt) ? ?(x) +

(8)

Proof. Define Vt (x) = h?t , x ? xt i + ?2 kx ? xt k2 + Lt2?t kx ? zt?1 k2 . It is easy to see that zt = arg minx?Rd Vt (x). Moreover, notice that Vt (x) is (Lt ?t + ?)-strongly convex. Hence on applying Lemmas 2 and 3, we obtain that for any x, Lt ?t + ? kx ? zt k2 Vt (zt) ? Vt (x) ? 2 ? Lt ?t Lt ?t + ? = h?t , x ? xt i + kx ? xt k2 + kx ? zt?1 k2 ? kx ? zt k2 2 2 2 2Lt ?L Lt ?t Lt ?t +? ? ?(x)??(yt)? k?t k2 + kx?zt?1 k2 ? kx?zt k2 +h?t , x?yt i. 2L2t 2 2 Then, ?(yt) can be bounded from above, as: Lt ?t 2Lt ? L k?t k2 ? kzt ? zt?1 k2 ?(yt) ??(x) + h?t , xt ? zt i ? 2 2Lt 2 (9) Lt ?t Lt ?t + ? 2 2 + kx ? zt?1 k ? kx ? zt k + h?t , x ? yt i, 2 2 where the non-positive term ? ?2 kzt ? xt k2 has been dropped from its right-hand-side. On the other hand, by applying Lemma 3 with x = yt?1 , we get 2Lt ? L k?t k2 , (10) ?(yt) ? ?(yt?1) ? h?t , xt ? yt?1 i + h?t , yt?1 ? yt i ? 2L2t where the non-positive term ? ?2 kyt?1 ? xt k2 has also been dropped from the right-hand-side. On multiplying (9) by ?t and (10) by 1 ? ?t , and then adding them together, we obtain

2Lt ? L Lt ?t2 k?t k2 + A + B + C ? kzt ? zt?1 k2 , (11) 2 2Lt 2 where A = h?t , ?t (xt ? zt) + (1 ? ?t)(xt ? yt?1)i, B = ?t h?t , x ? yt i + (1 ? ?t)h?t , yt?1 ? yt i, L ?2 +?? L ?2 and C = t2 t kx ? zt?1 k2 ? t t2 t kx ? zt k2 . In the following, we consider to upper bound A and B. First, by using the update rule of xt in Algorithm 1 and the Young?s inequality1 , we have ?(yt)

? ?(x) ? (1 ? ?t)[?(yt?1) ? ?(x)] ?

A = h?t , ?t (xt ? zt?1) + (1 ? ?t)(xt ? yt?1)i + ?t h?t , zt?1 ? zt i = ?t h?t , zt?1 ? zt i ?

Lt ?t2 k?t k2 . kzt ? zt?1 k2 + 2 2Lt

On the other hand, B can be bounded as B = h?t , ?t x + (1 ? ?t)yt?1 ? xt i + h?t , xt ? yt i = ?t h?t , x ? zt?1 i +

(12) h?t , ?t i Lt

?k?t k , (13) Lt where the second equality is due to the update rule of xt , and the last step is from the CauchySchwartz inequality and the boundedness of ?t . Hence, plugging (12) and (13) into (11), ? ?t h?t , x ? zt?1 i +

(Lt ?L)k?t k2 ?k?t k + + ?t h?t , x?zt?1 i + C 2L2t Lt ?2 ? (1 ? ?t)[?(yt?1) ? ?(x)] + + ?t h?t , x ? zt?1 i + C, 2(Lt ? L)

?(yt) ? ?(x) ? (1??t)[?(yt?1)??(x)]?

where the last step is due to the fact that ?ax2 + bx ? obtain (8). 1

The Young?s inequality states that hx, yi ?

kxk2 2a

+

4

b2 4a

akyk2 2

with a, b ¿ 0. On re-arranging terms, we

for any a ¿ 0.

Let the optimal solution in problem (6) be x? . From the update rules in Algorithm 1, we observe that the triplet (xt , yt?1 , zt?1) depends on the random process ?[t?1] ? {?0 , . . . , ?t?1 } and hence is also random. Clearly, zt?1 and x? are independent of ?t . Thus, E?[t] h?t , x? ? zt?1 i

= E?[t?1] E?[t] [h?t , x? ? zt?1 i—?[t?1]] = E?[t?1] E?t [h?t , x? ? zt?1 i]

= E?[t?1] hx? ? zt?1 , E?t [?t]i = 0,

where the first equality uses Ex [h(x)] = Ey Ex [h(x)—y], and the last equality is from our assumption that the stochastic gradient G(x, ?) is unbiased. Taking expectations on both sides of (8) with x = x? , we obtain the following corollary, which will be useful in proving the subsequent theorems. Corollary 1. Lt ?t2 + ??t E[kx? ? zt k2] 2 ?2 Lt ?t2 E[kx? ? zt?1 k2] + . ? (1 ? ?t)(E[?(yt?1)] ? ?(x?)) + 2 2(Lt ? L)

E[?(yt)] ? ?(x?) +

So far, the choice of Lt and ?t in Algorithm 1 has been left unspecified. In the following, we will show that with a good choice of Lt and ?t , (the expectation of) ?(yt) converges rapidly to ?(x?). Theorem 1. Assume that E[kx? ? zt k2] ? D2 for some D. Set 3

Lt = b(t + 1) 2 + L,

?t =

2 , t+2

where b ¿ 0 is a constant. Then the expected error of Algorithm 1 can be bounded as

3D2 L 1 5? 2 ? 2 ? . E[?(yN)] ? ?(x) ? + 3D b + N2 3b N If ? were known, we can set b to the optimal choice of ? 2 ?5?D . N

?
5? 3D ,
and the bound in (15) becomes
(14)
(15) 3D 2 L N2
+

Note that so far ?(x) is only assumed to be convex. As is shown in the following theorem, the convergence rate can be further improved by assuming strong convexity. This also requires another setting of ?t and Lt which is different from that in (14). Theorem 2. Assume the same conditions as in Theorem 1, except that ?(x) is ?-strongly convex. Set s ?2 ?t?1 , for t ? 1, (16) Lt = L + ???1 ?t = ?t?1 + t?1 ? t?1 , for t ? 1; 4 2 where ?t ? ?tk=1 (1 ? ?t) for t ? 1 and ?0 = 1. Then, the expected error of Algorithm 1 can be bounded as 2(L + ?)D2 6? 2 E[?(yN)] ? ?(x?) ? + . (17) 2 N N? In comparison, FOLOS only converges as O(log(N)/N) for strongly convex objectives. 3.2

Remarks

As in recent studies on stochastic composite optimization [13], the error bounds in (15) and (17) consist of two terms: a faster term which is related to the smooth component and a slower term related to the non-smooth component. SAGE benefits from using the structure of the problem and accelerates the convergence of the smooth component. On the other hand, many stochastic (sub)gradient-based algorithms like FOLOS do not separate the smooth from the non-smooth part, but simply treat the whole objective ?as non-smooth. Consequently, convergence of the smooth component is also slowed down to O(1/ N). As can be seen from (15) and (17), the convergence of SAGE is essentially encumbered by the variance of the stochastic subgradient. Recall that the variance of the average of p i.i.d. random 5

variables is equal to 1/p of the original variance. Hence, as in Pegasos [1], ? can be reduced by estimating the subgradient from a data subset. Unlike the AC-SA algorithm in [13], the settings of Lt and ?t in (14) do not require knowledge of ? and the number of iterations, both of which can be difficult to estimate in practice. Moreover, with the use of a sparsity-promoting ?(x), SAGE can produce a sparse solution (as will be experimentally demonstrated in Section 5) while AC-SA cannot. This is because in SAGE, the output yt is obtained from a generalized gradient update. With a sparsity-promoting ?(x), this reduces to a (soft) thresholding step, and thus ensures a sparse solution. On the other hand, in each iteration of AC-SA, its output is a convex combination of two other variables. Unfortunately, adding two vectors is unlikely to produce a sparse vector. 3.3

Efficient Computation of yt

The computational efficiency of Algorithm 1 hinges on the efficient computation of yt . Recall that yt is just the generalized gradient update, and so is not significantly more expensive than the gradient update in traditional algorithms. Indeed, the generalized gradient update is often a central component in various optimization and machine learning algorithms. In particular, Duchi and Singer [3] showed how this can be efficiently computed with the various smooth and

non-smooth regularizers, including the ?1 , ?2 , ?22 , ?? , Berhu and matrix norms. Interested readers are referred to [3] for details.

4

Accelerated Gradient Method for Online Learning

In this section, we extend the proposed accelerated gradient scheme for online learning of (2). The algorithm, shown in Algorithm 2, is similar to the stochastic version in Algorithm 1. Algorithm 2 SAGE-based Online Learning Algorithm. Inputs: Sequences {Lt } and {?t }, where Lt ¿ L and 0 ¡ ?t ¡ 1. Initialize: z1 = y1 . loop xt = (1 ? ?t)yt?1 + ?t zt?1 .

Output yt = arg minx h?ft?1 (xt), x ? xt i + L2t kx ? xt k2 + ?(x) . zt = zt?1 ? ?t (Lt + ??t)?1 [Lt (xt ? yt) + ?(zt?1 ? xt)]. end loop First, we introduce the following lemma, which plays a similar role as its stochastic counterpart of Lemma 3. Moreover, let ?t ? Lt (xt ? yt) be the gradient mapping related to the updating of yt . Lemma 4. For t ¿ 1, ?t (x) can be quadratically bounded from below as ?t?1 (x) ? ?t?1 (yt) + h?t , x ? xt i +

? 2Lt ? L kx ? xt k2 + k?t k2 . 2 2L2t

Proposition 2. For any x and t ? 1, assume that there exists a subgradient g?(x) ? ??(x) such that k?ft (x) + g?(x)k? ? Q. Then for Algorithm 2, ?t?1 (yt?1) ? ?t?1 (x) ?

Q2 Lt Lt + ??t + kx ? zt?1 k2 ? kx ? zt k2 2(1 ? ?t)(Lt ? L) 2?t 2?t (18) Lt (1 ? ?t2)Lt ? ?t (1 ? ?t)L 2 2 kyt?1 ? zt?1 k ? kzt ? yt k . + 2 2

Proof Sketch. Define ?t = Lt ?t?1 . From the update rule of zt , one can check that ? ?t zt = arg min Vt (x) ? h?t , x ? xt i + kx ? xt k2 + kx ? zt?1 k2 . x 2 2 Similar to the analysis in obtaining (9), we can obtain 2Lt ?L ?t ?t ?t +? ?t?1 (yt)??t?1 (x) ? h?t , xt?zt i? k?t k2? kzt?zt?1 k2+ kx?zt?1 k2? kx?zt k2 . (19) 2 2Lt 2 2 2 6

On the other hand, h?t , xt ? zt i ?

Lt k?t k2 = (kzt ? xt k2 ? kzt ? yt k2) 2Lt 2 Lt (1 ? ?t) Lt Lt kzt ? zt?1 k2 + kzt?1 ? yt?1 k2 ? kzt ? yt k2 , (20) ? 2?t 2 2

on using the convexity of k ? k2 . Using (20), the inequality (19) becomes ?t?1 (yt) ? ?t?1 (x) ?

Lt (1 ? ?t) Lt kzt?1 ? yt?1 k2 ? kzt ? yt k2 2 2 ?t ?t + ? Lt ? L 2 k?t k + kx ? zt?1 k2 ? kx ? zt k2 . ? 2 2Lt 2 2

(21)

On the other hand, by the convexity of ?t?1 (x) and the Young?s inequality, we have ?t?1 (yt?1) ? ?t?1 (yt) ? h?ft?1 (yt?1) + g?t?1 (yt?1), yt?1 ? yt i ?

Q2 (1 ? ?t)(Lt ? L) + kyt?1 ? yt k2 . 2(1 ? ?t)(Lt ? L) 2

(22)

Moreover, by using the update rule of xt and the convexity of k ? k2 , we have

kyt?1 ? yt k2 = k(yt?1 ? xt) + (xt ? yt)k2 = k?t (yt?1 ? zt?1) + (xt ? yt)k2 ? ?t kyt?1 ? zt?1 k2 + (1 ? ?t)?1 kxt ? yt k2 = ?t kyt?1 ? zt?1 k2 +

k?t k2 . (1 ? ?t)L2t

(23)

On using (23), it follows from (22) that ?t?1 (yt?1) ? ?t?1 (yt) ?

Q2 ?t (1??t)(Lt ?L) Lt ?L k?t k2 . + kyt?1 ?zt?1 k2 + 2(1??t)(Lt ?L) 2 2L2t

Inequality (18) then follows immediately by adding this to (21). ? Theorem 3. Assume that ? = 0, and kx? ?zt k ? D for t ? 1. Set ?t = a and Lt = aL t ? 1+L, where a ? (0, 1) is a constant. Then the regret of Algorithm 2 can be bounded as

N X ? Q2 LD2 LD2 ? + + [?t (yt) ? ?t (x)] ? N. 2a 2 a(1 ? a)L t=1

Theorem 4. Assume that ? ¿ 0, and kx? ? zt k ? D for t ? 1. Set ?t = a, and Lt = a?t + L + a?1 (? ? L)+ , where a ? (0, 1) is a constant. Then the regret of Algorithm 2 can be bounded as

N X Q2 (2a + a?1)? + L D2 + log(N + 1). [?t (yt) ? ?t (x?)] ? 2a 2a(1 ? a)? t=1 In particular, with a = 12 , the regret bound reduces to

5

Experiments

3? 2

+ L D2 +

2Q2 ?

log(N + 1).

In this section, we perform experiments on the stochastic optimization of (2). Two data sets are used2 (Table 1). The first one is the pcmac data set, which is a subset of the 20-newsgroup data set from [18], while the second one is the RCV1 data set, which is a filtered collection of the Reuters RCV1 from [19]. We choose the square loss for ?(?, ?) and the ?1 regularizer for ?(?) in (2). As discussed in Section 3.3 and [3], the generalized gradient update can be efficiently computed by soft thresholding in this case. Moreover, we do not use strong convexity and so ? = 0. We compare the proposed SAGE algorithm (with Lt and ?t in (14)) with three recent algorithms: (1) FOLOS [3]; (2) SMIDAS [4]; and (3) SCD [4]. For fair comparison, we compare their convergence 2

Downloaded from http://people.cs.uchicago.edu/?vikass/svmlin.html and http://www.cs.ucsb.edu/?wy

7

behavior w.r.t. both the number of iterations and the number of data access operations, the latter of which has been advocated in [4] as an implementation-independent measure of time. Moreover, the efficiency tricks for sparse data described in [4] are also implemented. Following [4], we set the regularization parameter ? in (2) to 10?6 . The ? parameter in SMIDAS is searched over the range of {10?6 , 10?5 , 10?4 , 10?3 , 10?2 , 10?1 }, and the one with the lowest ?1 -regularized loss is used. As in Pegasos [1], the (sub)gradient is computed from small sample subsets. The subset size p is set to min(0.01m, 500), where m is the data set size. This is used on all the algorithms except SCD, since SCD is based on coordinate descent and is quite different from the other stochastic subgradient algorithms.3 All the algorithms are trained with the same maximum amount of ?time? (i.e., number of data access operations). Table 1: Summary of the data sets. data set pcmac RCV1

#features 7,511 47,236

#instances 1,946 193,844

sparsity 0.73% 0.12%

Results are shown in Figure 1. As can be seen, SAGE requires much fewer iterations for convergence than the others (Figures 1(a) and 1(e)). Moreover, the additional costs on maintaining xt and zt are small, and the most expensive step in each SAGE iteration is in computing the generalized gradient update. Hence, its per-iteration complexity is comparable with the other (sub)gradient schemes, and its convergence in terms of the number of data access operations is still the fastest (Figures 1(b), 1(c), 1(f) and 1(g)). Moreover, the sparsity of the SAGE solution is comparable with those of the other algorithms (Figures 1(d) and 1(h)). 100

0.8 0.6 0.4 0.2 0 0

1000

2000

3000

Number of Iterations

80 SAGE FOLOS SMIDAS SCD

0.6 0.4

40

2

4

6

8

0 0

10

Number of Data Accessesx 106

(a)

SAGE FOLOS SMIDAS SCD

60

20

0.2 0 0

4000

8000

1 0.8

Density of w

SAGE FOLOS SMIDAS

Error (%)

L1 regularized loss

L1 regularized loss

1

2

4

6

8

4000 2000 0 0

10

Number of Data Accessesx 106

(b)

SAGE FOLOS SMIDAS SCD
6000
(c)
2
4
6
8
10
Number of Data Accesses x 106
(d)
4
0.6 0.4 0.2 0 0
1000
2000
3000
4000
Number of Iterations
(e)
SAGE FOLOS SMIDAS SCD
0.8 0.6
80
0.4
60 40 20
0.2 0 0
SAGE FOLOS SMIDAS SCD
Density of w
SAGE FOLOS SMIDAS
0.8
4
1
Error (%)
L1 regularized loss
L1 regularized loss
100 1
0.5
1
1.5
2
0 0
2.5
Number of Data Accessesx 108
0.5
1
1.5
2
Number of Data Accesses

(f)
(g)
2.5 8 x 10
x 10
3 2
SAGE FOLOS SMIDAS SCD
1 0 0
0.5
1
1.5
2
2.5
Number of Data Accessesx 108
(h)

Figure 1: Performance of the various algorithms on the pcmac (upper) and RCV1 (below) data sets.

6

Conclusion

In this paper, we developed a novel accelerated gradient method (SAGE) for stochastic convex composite optimization. It enjoys the computational simplicity and scalability of traditional (sub)gradient methods but are much faster, both theoretically and empirically. Experimental results show that SAGE outperforms recent (sub)gradient descent methods. Moreover, SAGE can also be extended to online learning, obtaining the best regret bounds currently known.

Acknowledgment This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant 615209. 3 For the same reason, an SCD iteration is also very different from an iteration in the other algorithms. Hence, SCD is not shown in the plots on the regularized loss versus number of iterations.

8

2 References

[1] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In Proceedings of the 24th International Conference on Machine Learning, pages 807?814, Corvalis, Oregon, USA, 2007. [2] A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. Journal of Machine Learning Research, 10:1737?1754, 2009. [3] J. Duchi and Y. Singer. Online and batch learning using forward looking subgradients. Technical report, 2009. [4] S. Shalev-Shwartz and A. Tewari. Stochastic methods for ?1 regularized loss minimization. In Proceedings of the 26th International Conference on Machine Learning, pages 929?936, Montreal, Quebec, Canada, 2009. [5] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In Advances in Neural Information Processing Systems 20. 2008. [6] S. Shalev-Shwartz and N. Srebro. SVM optimization: Inverse

dependence on training set size. In Proceedings of the 25th International Conference on Machine Learning, pages 928?935, Helsinki, Finland, 2008. [7] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(k12). Doklady AN SSSR (translated as Soviet. Math. Docl.), 269:543?547, 1983. [8] Y. Nesterov. Gradient methods for minimizing composite objective function. CORE Discussion Paper 2007/76, Catholic University of Louvain, September 2007. [9] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2:183?202, 2009. [10] R. Tibshirani. Regression shrinkage and selection via the Lasso. Journal of the Royal Statistical Society: Series B, 58:267?288, 1996. [11] S. Ji, L. Sun, R. Jin, and J. Ye. Multi-label multiple kernel learning. In Advances in Neural Information Processing Systems 21. 2009. [12] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In Proceedings of the International Conference on Machine Learning. Montreal, Canada, 2009. [13] G. Lan. An optimal method for stochastic composite optimization. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2009. [14] Y. Nesterov and I.U.E. Nesterov. Introductory Lectures on Convex Optimization: A Basic Course. Kluwer, 2003. [15] S.M. Kakade and S. Shalev-Shwartz. Mind the duality gap: Logarithmic regret algorithms for online optimization. In Advances in Neural Information Processing Systems 21. 2009. [16] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. Operations Research Letters, 31(3):167?175, 2003. [17] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by separable approximation. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Las Vegas, Nevada, USA, March 2008. [18] V. Sindhwani and S.S. Keerthi. Large scale semi-supervised linear SVMs. In Proceedings of the SIGIR Conference on Research and Development in Information Retrieval, pages 477?484, Seattle, WA, USA, 2006. [19] Y. Song, W.Y. Chen, H. Bai, C.J. Lin, and E.Y. Chang. Parallel spectral clustering. In Proceedings of the European Conference on Machine Learning, pages 374?389, Antwerp, Belgium, 2008.

9