Without-Replacement Sampling for Stochastic Gradient Methods

Authored by:

Ohad Shamir

Abstract

Stochastic gradient methods for machine learning and optimization problems are usually analyzed assuming data points are sampled *with* replacement. In contrast, sampling *without* replacement is far less understood, yet in practice it is very common, often easier to implement, and usually performs better. In this paper, we provide competitive convergence guarantees for without-replacement sampling under several scenarios, focusing on the natural regime of few passes over the data. Moreover, we describe a useful application of these results in the context of distributed optimization with randomly-partitioned data, yielding a nearly-optimal algorithm for regularized least squares (in terms of both communication complexity and runtime complexity) under broad parameter regimes. Our proof techniques combine ideas from stochastic optimization, adversarial online learning and transductive learning theory, and can potentially be applied to other stochastic optimization and learning problems.

1 Paper Body

Many canonical machine learning problems boil down to solving a convex empirical risk minimization problem of the form m

min F (w) =

w?W

1 X fi (w), m i=1

(1)

where each individual function fi (?) is convex (e.g. the loss on a given example in the training data), and the set W ? Rd is convex. In large-scale applications, where both m, d can be huge, a very popular approach is to employ stochastic gradient methods. Generally speaking, these methods maintain some iterate wt ? W, and at each iteration, sample an individual function fi (?), and perform some update to wt based on ?fi (wt). Since the update is with respect to a single function, this update is usually computationally cheap. Moreover, when the sampling is done independently and uniformly at random,

?fi (wt) is an unbiased estimator of the true gradient ?F (wt), which allows for good convergence guarantees after a reasonable number of iterations (see for instance [18, 15]). However, in practical implementations of such algorithms, it is actually quite common to use withoutreplacement sampling, or equivalently, pass sequentially over a random shuffling of the functions fi . Intuitively, this forces the algorithm to process more equally all data points, and often leads to better empirical performance. Moreover, without-replacement sampling is often easier and faster to implement, as it requires sequential data access, as opposed to the random data access required by with-replacement sampling (see for instance [3, 16, 8]). 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

1.1

What is Known so Far?

Unfortunately, without-replacement sampling is not covered well by current theory. The challenge is that unlike with-replacement sampling, the functions processed at every iteration are not statistically independent, and their correlations are difficult to analyze. Since this lack of theory is the main motivation for our paper, we describe the existing known results in some detail, before moving to our contributions. To begin with, there exist classic convergence results which hold deterministically for every order in which the individual functions are processed, and in particular when we process them by sampling without replacement (e.g. [14]). However, these can be exponentially worse than those obtained using random without-replacement sampling, and this gap is inevitable (see for instance [16]). More recently, Recht and R? [16] studied this problem, attempting to show that at least for least squares optimization, without-replacement sampling is always better (or at least not substantially worse) than with-replacement sampling on a given dataset. They showed this reduces to a fundamental conjecture about arithmetic-mean inequalities for matrices, and provided partial results in that direction, such as when the individual functions themselves are assumed to be generated i.i.d. from some distribution. However, the general question remains open. In a recent breakthrough, G?rb?zbalaban et al. [8] provided a new analysis of gradient descent algorithms for solving Eq. (1) based on random reshuffling: Each epoch, the algorithm draws a new permutation on {1, . . . , m} uniformly at random, and processes the individual functions in that order. Under smoothness and strong convexity assumptions, the authors obtain convergence guarantees of essentially O(1/k 2) after k epochs, vs. O(1/k) using with-replacement sampling (with the O(?) notation including certain dependencies on the problem parameters and data size). Thus, withoutreplacement sampling is shown to be strictly better than with-replacement sampling, after sufficiently many passes over the data. However, this leaves open the question of why without-replacement sampling works well after a few ? or even just one ? passes over the data. Indeed, this is often the regime at which stochastic gradient methods are most useful, do not require repeated data reshuffling, and their good convergence properties are well-understood in the with-replacement case. 1.2

Our Results

In this paper, we provide convergence guarantees for stochastic gradient methods, under several scenarios, in the natural regime where the number of passes over the data is small, and in particular that no data reshuffling is necessary. We emphasize that our goal here will be more modest than those of [16, 8]: Rather than show superiority of without-replacement sampling, we only show that it will not be significantly worse (in a worst-case sense) than with-replacement sampling. Nevertheless, such guarantees are novel, and still justify the use of with-replacement sampling, especially in situations where it is advantageous due to data access constraints or other reasons. Moreover, these results have a useful application in the context of distributed learning and optimization, as we will shortly describe. Our main contributions can be summarized as follows: ? For convex functions on some convex domain W, we consider algorithms which perform a single pass over a random permutation of m individual functions, and show that their suboptimality can be characterized by a combination of two quantities, each from a different field: First, the regret which the algorithm can attain in the setting of adversarial online convex optimization [17, 9], and second, the transductive Rademacher complexity of W with respect to the individual functions, a notion stemming from transductive learning theory [22, 6]. ? As a concrete application of the above, we show that if each function fi (?) corresponds to a convex Lipschitz loss of a linear? predictor, and the algorithm belongs to the class of algorithms which in the online setting attain O(T) regret on T such functions (which includes, for example, stochastic gradient descent), then ? the suboptimality using without-replacement sampling, after processing T functions, is O(1/ T). Up to numerical constants, the guarantee is the same as that obtained using with-replacement sampling. ? We turn to consider more specifically the stochastic gradient descent algorithm, and show that if the objective function F (?) is ?-strongly convex, and the functions fi (?) are also smooth, then the suboptimality bound becomes O(1/?T), which matches the with-replacement guarantees 2

(although with replacement, smoothness is not needed, and the dependence on some parameters hidden in the O(?) is somewhat better). ? In recent years, a new set of fast stochastic algorithms to solve Eq. (1) has emerged, such as SAG, SDCA, SVRG, and quite a few other variants. These algorithms are characterized by cheap stochastic iterations, involving computations of individual function gradients, yet unlike traditional stochastic algorithms, enjoy a linear convergence rate (runtime scaling logarithmically with the required accuracy). To the best of our knowledge, all existing analyses require sampling with replacement. We consider a representative algorithm from this set, namely SVRG, and the problem of regularized least squares, and show that similar guarantees can be obtained using without-replacement sampling. This result has a potentially interesting implication: Under the mild assumption that the problem?s condition number is smaller than the data size, we get that SVRG can converge to an arbitrarily accurate solution (even up to machine precision), without the need to reshuffle the data ? only a single shuffle at the beginning suffices. Thus, at least for this problem, we can obatin fast and high-quality solutions even if random data access is expensive. ? A further application of the SVRG result is in the

context of distributed learning: By simulating without-replacement SVRG on data randomly partitioned between several machines, we get a nearly-optimal algorithm for regularized least squares, in terms of communication and computational complexity, as long as the condition number is smaller than the data size per machine (up to logarithmic factors). This builds on the work of Lee et al. [13], who were the first to recognize the applicability of SVRG to distributed optimization. However, their results relied on with-replacement sampling, and are applicable only for much smaller condition numbers. ? We note that our focus is on scenarios where no reshufflings are necessary. In particular, the O(1/ T) and O(1/?T) bounds apply for all T ? {1, 2, . . . , m}, namely up to one full pass over a random permutation of the entire data. However, our techniques are also applicable to a constant (¿ 1) number of passes, by randomly reshuffling the data after every pass. In a similar vein, our SVRG result can be readily extended to a situation where each epoch of the algorithm is done on an independent permutation of the data. We leave a full treatment of this to future work.

2

Preliminaries and Notation

We use bold-face symbols to denote vectors. Given a vector w, wi denotes it?s i-th coordinate. We ? ?(?), ? ? to hide constants utilize the standard O(?), ?(?), ?(?) notation to hide constants, and O, ?(?) and logarithmic factors. Given convex functions f1 (?), f2 (?), . . . , fm (?) from Rd to R, we define our objective function F : Rd ? R as m 1 X F (w) = fi (w), m i=1 with some fixed optimum w? ? arg minw?W F (w). In machine learning applications, each individual fi (?) usually corresponds to a loss with respect to a data point, hence will use the terms ?individual function?, ?loss function? and ?data point? interchangeably throughout the paper. We let ? be a permutation over {1, . . . , m} chosen uniformly at random. In much of the paper, we consider methods which draw loss functions without replacement according to that permutation (that is, f?(1) (?), f?(2) (?), f?(3) (?), . . .). We will use the shorthand notation t?1

F1:t?1 (?) =

m

X 1 X 1 f?(i) (?) , Ft:m (?) = f?(i) (?) t ? 1 i=1 m ? t + 1 i=t

to denote the average loss with respect to the first t ? 1 and last m ? t + 1 loss functions respectively, as ordered by the permutation (intuitively, the losses in F1:t?1 (?) are those already observed by the algorithm at the beginning of iteration t, whereas the losses in Ft:m (?) are those not yet observed). We use the convention that F1:1 (?) ? 0, and the same goes for other expressions of the form Pt?1 1 i=1 ? ? ? throughout the paper, when t = 1. We also define quantities such as ?F1:t?1 (?) and t?1 ?Ft:m (?) similarly. 3

A function f : Rd ? R is ?-strongly convex, if for any w, w0 , f (w0) ? f (w) + 2 hg, w0 ? wi + ?2 kw0 ? wk , where g is any (sub)-gradient of f at w, and is ?-smooth if for 2 any w, w0 , f (w0) ? f (w) + hg, w0 ? wi + ?2 kw0 ? wk . ?-smoothness also implies that the function f is differentiable, and its gradient is ?-Lipschitz. Based on these properties, it is easy to verify that if w? ? arg

min f (w), and f is ?-strongly convex and ?-smooth, then 2 ? ? 2 ? f (w) ? f (w?) ? ?2 kw ? w? k . 2 kw ? w k We will also require the notion of trandsuctive Rademacher complexity, as developed by El-Yaniv and Pechyony [6, Definition 1], with a slightly different notation adapted to our setting: Definition 1. Let V be a set of vectors v = (v1 , . . . , vm) in Rm . Let s, u be positive integers such su that s + u = m, and denote p := (s+u) 2 ? (0, 1/2). We define the transductive Rademacher

Pm 1 Complexity Rs,u (V) as Rs,u (V) = s + u1 ? Er1 ,...,rm [supv?V i=1 ri vi] , where r1 , . . . , rm are i.i.d. random variables such that ri = 1 or ?1 with probability p, and ri = 0 with probability 1 ? 2p. This quantity is an important parameter is studying the richness of the set V, and will prove crucial in providing some of the convergence results presented later on. Note that it differs slightly from standard Rademacher complexity, which is used in the theory of learning from i.i.d. data, where the Rademacher variables ri only take ?1, +1 values, and (1/s + 1/u) is replaced by 1/m).

3

Convex Lipschitz Functions

We begin by considering loss functions f1 (?), f2 (?), . . . , fm (?) which are convex and L-Lipschitz over some convex domain W. We assume the algorithm sequentially goes over some permuted ordering of the losses, and before processing the t-th loss function, produces an iterate wt ? W. Moreover, we assume that the algorithm has a regret bound in the adversarial online setting, namely that for any sequence of T convex Lipschitz losses f1 (?), . . . , fT (?), and any w ? W, T X t=1

ft (wt) ?

T X

ft (w) ? RT

t=1

for some RT scaling sub-linearly in T 1 . For example, online gradient descent (which is equivalent to stochastic gradient descent when the losses are i.i.d.), with a suitable step size, satisfies RT = ? O(BL T), where L is the Lipschitz parameter and B upper bounds the norm of any vector in W. A similar regret bound can also be shown for other online algorithms (see [9, 17, 23]). Since the ideas used for analyzing this setting will also be used in the more complicated results which follow, we provide the analysis in some detail. We first have the following straightforward theorem, which upper bounds the expected suboptimality in terms of regret and the expected difference between the average loss on prefixes and suffixes of the data. Theorem 1. Suppose the algorithm has a regret bound RT , and sequentially processes f?(1) (?), . . . , f?(T) (?) where ? is a random permutation on {1, . . . , m}. Then in expectation over ?, " # T T 1X RT 1 X ? E F (wt) ? F (w) ? + (t ? 1) ? E[F1:t?1 (wt) ? Ft:m (wt)]. T t=1 T mT t=2 The left hand side in the inequality above can be interpreted as an expected bound on F (wt)?F (w?), where t is drawn uniformly at random from 1, 2, . . . , T . Alternatively, by Jensen?s inequality ? T) ? F (w?)], where and the fact that F (?) is convex, the same bound also applies on E[F (w P T ? T = T1 t=1 wt . w The proof of the theorem relies on the following simple but

key lemma, which expresses the expected difference between with-replacement and without-replacement sampling in an alternative form, similar to Thm. 1 and one which lends itself to tools and ideas from transductive learning theory. This lemma will be used in proving all our main results, and its proof appears in Subsection A.2 1 For simplicity, we assume the algorithm is deterministic given f1 , . . . , fm , but all results in this section also hold for randomized algorithms (in expectation over the algorithm?s randomness), assuming the expected regret of the algorithm w.r.t. any w ? W is at most RT .

4

Lemma 1. Let ? be a permutation over {1, . . . , m} chosen uniformly at random. Let s1 , . . . , sm ? R be random variables which conditioned on ?(1), . . . , ?(t ? 1), are independent of ?(t), . . . , ?(m). 1 P m t?1 Then E m i=1 si ? s?(t) equals m ? E [s1:t?1 ? st:m] for t ¿ 1, and 0 for t = 1. Proof of Thm. 1. Adding and subtracting terms, and using the facts that ? is a permutation chosen uniformly at random, and w? is fixed, ”

” # ” # T T T

1 X 1 X 1 X ? ? E F (wt) ? F (w) = E f?(t) (wt) ? F (w) + E F (wt) ? f?(t) (wt) T t=1 T t=1 T t=1 ” # ” # T T

1 X 1 X ? f?(t) (wt) ? f?(t) (w) + E F (wt) ? f?(t) (wt) = E T t=1 T t=1

Applying the regret bound assumption on the sequence of losses f?(1) (?), . . . , f?(T) (?), the above is

PT at most RTT + T1 t=1 E F (wt) ? f?(t) (wt) . Since wt (as a random variable over the permutation ? of the data) depends only on ?(1), . . . , ?(t ? 1), we can use Lemma 1 (where si = fi (wt), and PT 1 noting that the expectation above is 0 when t = 1), and get that the above equals RTT + mT t=2 (t ? 1) ? E[F1:t?1 (wt) ? Ft:m (wt)]. Having reduced the expected suboptimality to the expected difference E[F1:t?1 (wt) ? Ft:m (wt)], the next step is to upper bound it with E[supw?W (F1:t?1 (w) ? Ft:m (w))]: Namely, having split our loss functions at random to two groups of size t ? 1 and m ? t + 1, how large can be the difference between the average loss of any w on the two groups? Such uniform convergence bounds are exactly the type studied in transductive learning theory, where a fixed dataset is randomly split to a training set and a test set, and we consider the generalization performance of learning algorithms ran on the training set. Such results can be provided in terms of the transductive Rademacher complexity of W, and combined with Thm. 1, lead to the following bound in our setting: Theorem 2. Suppose that each wt is chosen from a fixed domain W, that the algorithm enjoys a regret bound RT , and that supi,w?W —fi (w)— ? B. Then in expectation over the random permutation ?, # ” T T RT 1 X 24B 1X ? F (wt) ? F (w) ? + (t ? 1)Rt?1:m?t+1 (V) + ? , E T t=1 T mT t=2 m where V = {(f1 (w), . . . , fm (w) — w ? W}. Thus, we obtained a generic bound which depends on the online learning characteristics of the algorithm, as well as the statistical learning properties of the class W on the loss functions. The proof (as the proofs of all our results from now on) appears in Section A. We now instantiate the theorem to the prototypical case of bounded-norm linear predictors, where the loss is

some convex and Lipschitz function of the prediction hw, xi of a predictor w on an instance x, possibly with some regularization: ? and each loss Corollary 1. Under the conditions of Thm. 2, suppose that W ? {w : kwk ? B}, function fi has the form ‘i (hw, xi i) + r(w) for some L-Lipschitz ‘ , kx k ? 1, and a fixed function i i ? h P i ? 2(12+ 2)BL T RT 1 ? ? r. Then E T t=1 F (wt) ? F (w) ? T + . m ? ? As discussed earlier, in the setting of Corollary?1, typical regret bounds are on the order of O(BL T). ? Thus, the expected suboptimality is O(BL/ T), all the way up to T = m (i.e. a full pass over a random permutation of the data). Up to constants, this is the same as the suboptimality attained by T iterations of with-replacement sampling, using stochastic gradient descent or similar algorithms.

4

Faster Convergence for Strongly Convex Functions

We now consider more specifically the stochastic gradient descent algorithm on a convex domain W, which can be described as follows: We initialize at some w1 ? W, and perform the update steps wt+1 = ?W (wt ? ?t gt), 5

where ?t ¿ 0 are fixed step sizes, ?W is projection on W, and gt is a subgradient of f?(t) (?) at wt . Moreover, we assume the objective function F (?) is ?-strongly convex for some ? ¿ 0. In this scenario, using with-replacement sampling (i.e. gt is a subgradient of an independently drawn fi (?)), performing T iterations as above and returning a randomly sampled iterate wt or their average results ? 2 /?T), where G2 is an upper bound on the expected squared norm of in expected suboptimality O(G gt [15, 18]. Here, we study a similar situation in the without-replacement case. In the result below, we will consider specifically the case where each fi (w) is a Lipschitz and smooth loss of a linear predictor w, possibly with some regularization. The smoothness assumption is needed to get a good bound on the transductive Rademacher complexity of quantities such as h?fi (w), wi. However, the technique can be potentially applied to more general cases. Theorem 3. Suppose W has diameter B, and that F (?) is ?-strongly convex on W. Assume that fi (w) = ‘i (hw, xi i) + r(w) where kxi k ? 1, r(?) is possibly some regularization term, and each ‘i is L-Lipschitz and ?-smooth on {z : z = hw, xi , w ? W, kxk ? 1}. Furthermore, suppose supw?W k?fi (w)k ? G. Then for any 1 ¡ T ? m, if we run SGD for T iterations with step size ?t = 2/?t, we have (for a universal positive constant c) # ” T ((L + ?B)2 + G2) log(T) 1X ? F (wt) ? F (w) ? c ? . E T t=1 ?T As in the results of the previous section, the left hand side is the expected optimality of a single wt where t is chosen uniformly at random, or an upper bound on the expected suboptimality of the P ? T = T1 Tt=1 wt . This result is similar to the with-replacement case, up to numerical average w constants and the additional (L + ?B 2) term in the numerator. We note that in some cases, G2 is the dominant term anyway2 . However, it is not clear that the (L + ?B 2) term is necessary, and removing it is left to future work. We also note that the log(T) factor in the theorem can be removed PT by considering not T1 t=1 F (wt), but rather only an average over some suffix of the iterates, or weighted averaging (see for instance [15, 12, 21], where the same techniques can be applied here). The proof of Thm. 3 is somewhat more challenging than the results ? of the previous section, since we

are attempting to get a faster rate of O(1/T) rather than O(1/ T), all the way up to T = m. A significant technical obstacle is that our proof technique relies on concentration of averages around ? expectations, which on T samples does not go down faster than O(1/ T). To overcome this, we apply concentration results not on the function values (i.e. F1:t?1 (wt) ? Ft:m (wt) as in the previous section), but rather on gradient-iterate inner products, i.e. h?F1:t?1 (wt) ? ?Ft:m (wt), wt ? w? i, where w? is the optimal solution. To get good bounds, we need to assume these gradients have a certain structure, which is why we need to make the assumption in the theorem about the form of each fi (?). Using transductive Rademacher complexity tools, we manage to upper bound the expectation q 2 ? of these inner products by quantities roughly of the form E[kwt ? w? k]/ t (assuming here t ¡ m/2 for simplicity). We now utilize the fact that in the strongly convex case, kwt ? w? k itself decreases to zero with t at a certain rate, to get fast rates decreasing as 1/t.

5

Without-Replacement SVRG for Least Squares

In this section, we will consider a more sophisticated stochastic gradient approach, namely the SVRG algorithm of [11], designed to solve optimization problems with a finite sum structure as in Eq. (1). Unlike purely stochastic gradient procedures, this algorithm does require several passes over the data. However, assuming the condition number 1/? is smaller than the data size (assuming each fi (?) is O(1) smooth, and ? is the strong convexity parameter of F (?)), only O(m log(1/)) gradient evaluations are required to get an -accurate solution, for any . Thus, we can get a high-accuracy solution after the equivalent of a small number of data passes. As discussed in the introduction, over the past few years several other algorithms have been introduced and shown to have such a behavior. We will focus on the algorithm in its basic form, where the domain W equals Rd . The existing analysis of SVRG ([11]) assumes stochastic iterations, which involves sampling the data with replacement. Thus, it is natural to consider whether a similar convergence behavior occurs using 2

G is generally on the order of L + ?B, which is the same as L + ?B if L is the dominant term. This happens for instance with the squared loss, whose Lipschitz parameter is on the order of ?B.

6

without-replacement sampling. As we shall see, a positive reply has at least two implications: The first is that as long as the condition number is smaller than the data size, SVRG can be used to obtain a high-accuracy solution, without the need to reshuffle the data: Only a single shuffle at the beginning suffices, and the algorithm terminates after a small number of sequential passes (logarithmic in the required accuracy). The second implication is that such without-replacement SVRG can be used to get a nearly-optimal algorithm for convex distributed learning and optimization on randomly partitioned data, as long as the condition number is smaller than the data size at each machine. The SVRG algorithm using without-replacement sampling on a dataset of size m is described as Algorithm 1. It is composed of multiple epochs (indexed by s), each involving one gradient computation on the entire dataset, and T stochas-

tic iterations, involving gradient computations with respect to individual data points. Although the gradient computation on the entire dataset is expensive, it is only needed to be done once per epoch. Since the algorithm will be shown to have linear convergence as a function of the number of epochs, this requires only a small (logarithmic) number of passes over the data. Algorithm 1 SVRG using Without-Replacement Sampling Parameters: ?, T , permutation ? on {1, . . . , m} ? 1 at 0 Initialize w for s = 1, 2, . . . do ?s w(s?1)T +1 := w Pm 1 ? := ?F (w ? s) = m ? s) n i=1 ?fi (w for t = (s ? 1)T + 1, . . . , sT do

? s) + n ? wt+1 := wt ? ? ?f?(t) (wt) ? ?f?(t) (w end for ? s+1 be the average of w(s?1)T +1 , . . . , wsT , or one of them drawn uniformly at random. Let w end for We will consider specifically the regularized least mean squares problem, where ? 1 ? 2 2 fi (w) = (hw, xi i ? yi) + kwk . (2) 2 2 P ? ¿ 0. Moreover, we assume that F (w) = 1 m fi (w) is ?-strongly convex for some xi , yi and ? i=1 m ? For convenience, we will assume that kxi k , —yi —, ? are all at most 1 (this (note that necessarily ? ? ?). is without much loss of generality, since we can always re-scale the loss functions by an appropriate ? ? 2-smooth. factor). Note that under this assumption, each fi (?) as well as F (?) are also 1 + ? Theorem 4. Suppose each loss function fi (?) corresponds to Eq. (2), where xi ? Rd , maxi kxi k ? 1, ? ¿ 0, and that F (?) is ?-strongly convex, where ? ? (0, 1). Moreover, let B ? 1 maxi —yi — ? 1, ? 2 be such that kw? k ? B and maxt F (wt) ? F (w?) ? B with probability 1 over the random permutation. There is some universal constant c0 ? 1, such that for any c ? c0 and any ? (0, 1), if we run algorithm 1, using parameters ?, T satisfying

1 9 64dmB 2 ?= , T ? , m ? c log2 T, c ?? ? ? S+1 satisfies E[F (w ? S+1)?minw F (w)] ? . then after S = dlog4 (9/)e epochs of the algorithm, w In particular, by taking ? = ?(1) and T = ?(1/?), the algorithm attains an -accurate solution after O(log(1/)/?) stochastic iterations of without-replacement sampling, and O(log(1/)) sequential passes over the data to compute gradients of F (?). This implies that as long as 1/? (which stands for the condition number of the problem) is smaller than O(m/ log(1/)), the number of withoutreplacement stochastic iterations is smaller than the data size m. Thus, assuming the data is randomly shuffled, we can get a solution using only sequential data passes, without the need to reshuffle. In terms of the log factors, we note that the condition maxt F (wt) ? F (w?) ? B with probability 1 is needed for technical reasons in our analysis, and we conjecture that it can be improved. However, since B appears only inside log factors, even a crude bound would suffice. In appendix C, we indeed show that under there is always a valid B satisfying log(B) = O (log(1/) log(T) + log(1/?)). Regarding the logarithmic dependence on the dimension d, it is due to an application of a matrix Bernstein inequality for d ? d matrices, and can possibly be improved. 7

5.1 Application of Without-Replacement SVRG to distributed learning An important variant of the problems we discussed so far is when training data (or equivalently, the individual loss functions f1 (?), . . . , fm (?)) are split between different machines, who need to communicate in order to reach a good solution. This naturally models situations where data is too large to fit at a single machine, or where we wish to speed up the optimization process by par-

allelizing it on several computers. Over the past few years, there has been much research on this question in the machine learning community, with just a few examples including [24, 2, 1, 5, 4, 10, 20, 19, 25, 13]. A substantial number of these works focus on the setting where the data is split equally at random between k machines (or equivalently, that data is assigned to each machine by sampling without replacement from {f1 (?), . . . , fm (?)})). Intuitively, this creates statistical similarities between the data at different machines, which can be leveraged to aid the optimization process. Recently, Lee et al. [13] proposed a simple and elegant approach, which applies at least in certain parameter regimes. This is based on the observation that SVRG, according to its with-replacement analysis, requires O(log(1/)) epochs, P where in each epoch one needs to compute an exact gradient of the m 1 objective function F (?) = m i=1 fi (?), and O(1/?) gradients of individual losses fi (?) chosen uniformly at random (where is the required suboptimality, and ? is the strong convexity parameter of F (?)). Therefore, if each machine had i.i.d. samples from {f1 (?), . . . , fm (?)}, whose union cover {f1 (?), . . . , fm (?)}, the machines could just simulate SVRG: First, each machine splits its data to batches of sizeP O(1/?). Then, each SVRG epoch is simulated by the machines computing a m 1 gradient of F (?) = m i=1 fi (?) ? which can be fully parallelized and involves one communication round (assuming a broadcast communication model) ? and one machine computing gradients with respect to one of its batches. Overall, this would require O(log(1/)) communication rounds, and O(m/k + 1/?) log(1/) runtime, where m/k is the number of data points per machine (ignoring communication time, and assuming constant time to compute a gradient of fi (?)). Under the reasonable assumption that the strong convexity parameter ? is at least k/m, this requires runtime linear in the data size per machine, and logarithmic in the target accuracy , with only a logarithmic number of communication rounds. Up to log factors, this is essentially the best one can hope for with a distributed algorithm. Moreover, a lower bound in [13] indicates that at least in the worst case, O(log(1/)) communication rounds is impossible to obtain if ? is significantly smaller than k/m. Unfortunately, the reasoning above crucially relies on each machine having access to i.i.d. samples, which can be reasonable in some cases, but is different than the more common assumption that the data is randomly split among the machines. To circumvent this issue, [13] propose to communicate individual data points / losses fi (?) between machines, so as to simulate i.i.d. sampling. However, by the birthday paradox, this only works well in the?regime where the overall number of samples required (O((1/?) log(1/)) is not much larger than m. Otherwise, with-replacement and withoutreplacement sampling becomes statistically very different, and a large number of data points would need to be communicated. In other words, if communication is an expensive resource, then the ? solution proposed in [13] only works well when ? is at least order of 1/ m. In machine learning applications, the strong convexity parameter ? often comes from explicit regularization designed to ? prevent over-fitting, and needs to scale with the data size, usually between 1/ m and 1/m. Thus, the solution above is communication-efficient only when ? is relatively large. However, the situation immediately improves if we can use

a without-replacement version of SVRG, which can easily be simulated with randomly partitioned data: The stochastic batches can now be simply subsets of each machine?s data, which are statistically identical to sampling {f1 (?), . . . , fm (?)} without replacement. Thus, no data points need to be sent across machines, even if ? is small. For clarity, we present an explicit pseudocode as Algorithm 2 in Appendix D. Let us consider the analysis of no-replacement SVRG provided in Thm. 4. According to this analysis, by setting T = ?(1/?), then as long as the total number of batches is at least ?(log(1/)), and ? ? = ?(1/m), then the algorithm will attain an -suboptimal solution in expectation. In other words, without any additional communication, we extend the applicability of distributed SVRG (at least for ? ?m) regime to 1/? = ?(1/m). ? regularized least squares) from the ? = ?(1/ We emphasize that this formal analysis only applies to regularized squared loss, which is the scope of Thm. 4. However, this algorithmic approach can be applied to any loss function, and we conjecture that it will have similar performance for any smooth losses and strongly-convex objectives. Acknowledgments: This research is supported in part by an FP7 Marie Curie CIG grant, an ISF grant 425/13, and by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI). 8

2 References

[1] A. Agarwal, O. Chapelle, M. Dud?k, and J. Langford. A reliable effective terascale linear learning system. CoRR, abs/1110.4198, 2011. [2] M.-F. Balcan, A. Blum, S. Fine, and Y. Mansour. Distributed learning, communication complexity and privacy. In COLT, 2012. [3] L. Bottou. Stochastic gradient descent tricks. In Neural Networks: Tricks of the Trade. Springer, 2012. [4] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In NIPS, 2011. [5] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. Journal of Machine Learning Research, 13:165?202, 2012. [6] R. El-Yaniv and D. Pechyony. Transductive rademacher complexity and its applications. Journal of Artificial Intelligence Research, 35:193?234, 2009. [7] D. Gross and V. Nesme. Note on sampling without replacing from a finite collection of matrices. arXiv preprint arXiv:1001.2738, 2010. [8] M. G?rb?zbalaban, A. Ozdaglar, and P. Parrilo. Why random reshuffling beats stochastic gradient descent. arXiv preprint arXiv:1510.08560, 2015. [9] E. Hazan. Introduction to online convex optimization. Book draft, 2015. [10] M. Jaggi, V. Smith, M. Tak?c, J. Terhorst, S. Krishnan, T. Hofmann, and M. Jordan. Communicationefficient distributed dual coordinate ascent. In NIPS, 2014. [11] R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In NIPS, 2013. [12] S. Lacoste-Julien, M. Schmidt, and F. Bach. A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. arXiv preprint arXiv:1212.2002, 2012. [13] J. Lee, T. Ma, and Q. Lin. Distributed stochastic variance reduced gradient methods. arXiv preprint arXiv:1507.07595, 2015. [14] A. Nedi?c and D. Bertsekas. Convergence rate

of incremental subgradient algorithms. In Stochastic optimization: algorithms and applications, pages 223?264. Springer, 2001. [15] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. arXiv preprint arXiv:1109.5647, 2011. [16] B. Recht and C. R?. Beneath the valley of the noncommutative arithmetic-geometric mean inequality: conjectures, case-studies, and consequences. In COLT, 2012. [17] S. Shalev-Shwartz. Online learning and online convex optimization. Foundations and Trends in Machine Learning, 4(2):107?194, 2011. [18] S. Shalev-Shwartz and S. Ben-David. Understanding machine learning: From Theory to Algorithms. Cambridge University Press, 2014. [19] O. Shamir and N. Srebro. On distributed stochastic optimization and learning. In Allerton, 2014. [20] O. Shamir, N. Srebro, and T. Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In ICML, 2014. [21] O. Shamir and T. Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In ICML, 2013. [22] V. Vapnik. Statistical learning theory. Wiley New York, 1998. [23] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. Journal of Machine Learning Research, 11:2543?2596, 2010. [24] Y. Zhang, J. Duchi, and M. Wainwright. Communication-efficient algorithms for statistical optimization. Journal of Machine Learning Research, 14:3321?3363, 2013. [25] Y. Zhang and L. Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. In ICML, 2015.

9