Doubly Stochastic Algorithms for Large-Scale Optimization

Alec Koppel, Aryan Mokhtari, and Alejandro Ribeiro

***Abstract*—We consider learning problems over training sets in which both, the number of training examples and the dimension of the feature vectors, are large. To solve these problems we propose the random parallel stochastic algorithm (RAPSA). We call the algorithm random parallel because it utilizes multiple processors to operate in a randomly chosen subset of blocks of the feature vector. We call the algorithm stochastic because processors choose elements of the training set randomly and independently. Algorithms that are parallel in either of these dimensions exist, but RAPSA is the first attempt at a methodology that is parallel in both, the selection of blocks and the selection of elements of the training set. In RAPSA, processors utilize the randomly chosen functions to compute the stochastic gradient component associated with a randomly chosen block. We show that this type of doubly stochastic approximation method, when executed on an asynchronous parallel computing architecture, exhibits comparable convergence behavior to that of classical stochastic gradient descent on strongly convex functions – for diminishing step-sizes, asynchronous RAPSA converges to the minimizer of the expected risk. We illustrate empirical algorithm performance on a linear estimation problem, as well as a binary image classification using the MNIST handwritten digit dataset.**

I. Introduction

Learning is often formulated as an optimization problem that finds a classifier $\mathbf{x}^* \in \mathbb{R}^p$ that minimizes the average of a loss function across the elements of a training set. Specifically, consider a training set with N elements and let $f_n : \mathbb{R}^p \to \mathbb{R}$ be a convex loss function associated with the nth sample. The optimal classifier $\mathbf{x}^* \in \mathbb{R}^p$ is the minimizer of the expected risk $F(\mathbf{x}) := (1/N)\sum_{n=1}^N f_n(\mathbf{x})$,

$$\mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{x}} \frac{1}{N}\sum_{n=1}^{N} f_n(\mathbf{x}). \tag{1}$$

Problems such as support vector machines, logistic regression, and matrix completion can be put in the form of problem (1). In this paper we are interested in large scale problems where both, the number of features p and the number of elements N in the training set are very large ($p = \mathcal{O}(N)$) – which arise, e.g., in text [2], image [3], and genomic [4] processing.

Problems that operate on blocks of the parameter vectors *or* subsets of the training set, but not on both, blocks *and* subsets, exist. Block coordinate descent (BCD) is the generic name for methods in which the variable space is divided in blocks that are processed separately. Early versions operate by cyclically updating all coordinates at each step [5], [6], while more recent parallelized versions of coordinate descent have been developed to accelerate convergence of BCD [7]–[10].

Methods that utilize a subset of sample points, called stochastic approximation, rely on the use of stochastic gradients. Classically, the gradient of the aggregate function is estimated by the gradient of a randomly chosen function f_n [11] called the stochastic gradient. Various recent developments have been aimed at accelerating the convergence of this method such as Hessian approximation schemes which incorporate higher-order information of the objective than just the gradient [12]–[15].

When N and p are large, fusing the complexity properties of parallel BCD and SGD becomes a necessity [16]–[19]. Motivated by this need, we propose the random adaptive parallel stochastic algorithm (RAPSA), which is the first effort to randomize over both parameters and sample functions. To do so, we consider the case in which the parameter vector $\mathbf{x}$ is divided into B blocks each of which contains $p_b \ll p$ features and a set of $I \ll B$ processors work in parallel on randomly chosen feature blocks while using a stochastic subset of elements of the training set. The blocks chosen for update and the functions fetched for determination of block updates are selected independently at random in subsequent slots (Section II-A). Moreover, we consider the case where this collection of I processors *does not* need to operate on a universal time index, which means that variability in the sparsity of data instances at distinct nodes will not cause a processing bottleneck (Section II-B). We establish that the convergence properties of asynchronous RAPSA are comparable to best-known guarantees for stochastic gradient method in the diminishing step-size regime (Section III). We then numerically evaluate the proposed method on a simple linear estimation problem as well as the MNIST digit recognition problem (Section IV).

II. Algorithm Development

We consider a generalization of (1) where the number N of functions f_n is not necessarily finite by introducing a random variable $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^q$ that determines the choice of the random smooth convex function $f(\cdot, \boldsymbol{\theta}) : \mathbb{R}^p \to \mathbb{R}$. We focus on the problem of minimizing the expected risk $F(\mathbf{x}) := \mathbb{E}_{\boldsymbol{\theta}}[f(\mathbf{x}, \boldsymbol{\theta})]$,

$$\mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} F(\mathbf{x}) := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} \mathbb{E}_{\boldsymbol{\theta}}\left[f(\mathbf{x}, \boldsymbol{\theta})\right]. \tag{2}$$

Problem (1) is a particular case of (2) in which each of the functions f_n is drawn with probability $1/N$. We refer to $f(\cdot, \boldsymbol{\theta})$ as instantaneous functions and to $F(\mathbf{x})$ as the average function.

Work in this paper is supported by NSF CCF-1017454, NSF CCF-0952867, ONR N00014-12-1-0997, ARL MAST CTA, and ASEE SMART. All proofs are given in [1].

The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104. Email: {aryanm, akoppel, aribeiro}@seas.upenn.edu.

A. Parallel Doubly Stochastic Approximation

RAPSA utilizes I processors to update a random subset of blocks of the variable $\mathbf{x}$, with each of the blocks relying on a subset of randomly and independently chosen elements of the training set. Formally, decompose the variable $\mathbf{x}$ into B blocks to write $\mathbf{x} = [\mathbf{x}_1; \ldots; \mathbf{x}_B]$, where block b has length p_b so that we have $\mathbf{x}_b \in \mathbb{R}^{p_b}$. At iteration t, processor i selects a random index b_i^t for updating and a random subset $\boldsymbol{\Theta}_i^t$ of L instantaneous functions. It then uses these instantaneous functions to determine stochastic gradient components for the subset of variables $\mathbf{x}_b = \mathbf{x}_{b_i^t}$ as an average of the components of the gradients of the functions $f(\mathbf{x}^t, \boldsymbol{\theta})$ for $\boldsymbol{\theta} \in \boldsymbol{\Theta}_i^t$,

$$\nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \boldsymbol{\Theta}_i^t) = \frac{1}{L} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}_i^t} \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \boldsymbol{\theta}), \qquad b = b_i^t. \tag{3}$$

Note that L can be interpreted as the mini-batch size for gradient approximation. The stochastic gradient block in (3) is then modulated by a possibly time varying stepsize γ^t and used by processor i to update the block $\mathbf{x}_b = \mathbf{x}_{b_i^t}$

$$\mathbf{x}_b^{t+1} = \mathbf{x}_b^t - \gamma^t \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \boldsymbol{\Theta}_i^t) \qquad b = b_i^t. \tag{4}$$

RAPSA is defined by the joint implementation of (3) and (4) in parallel among a collection of I processors. We would like to emphasize that the number of updated blocks which is equivalent to the number of processors I is not necessary equal to the total number of blocks B. In other words, we may update only a subset of coordinates $I/B < 1$ at each iteration. We define $r := I/B$ as the ratio of the updated blocks to the total number of blocks which is smaller than 1.

The selection of blocks is coordinated so that no processors operate in the same block. The selection of elements of $\mathbf{x}$ is uncoordinated across processors. The fact that at any point in time a random subset of blocks is being updated utilizing a random subset of elements of the training set means that RAPSA requires almost no coordination between processors. Moreover, these processors are not even required to operate on a global time index, as we discuss next.

B. Asynchronous Computing Architectures

Up to this point, the RAPSA method dictates that distinct parallel processors select blocks $b_i^t \in \{1, \ldots, B\}$ uniformly at random at each time step t. However, the requirement that each processor operates on a common time index is burdensome for parallel operations on large computing clusters, as it means that nodes must wait for the processor which has the longest computation time at each step before proceeding. Thus, we extend the method developed in Sections II-A such the parallel processors need not to operate on a globally coordinated clock, and establish its convergence, so long as the degree of their asynchronicity is bounded in a certain sense. In doing so, we alleviate the computational bottleneck in the parallel architecture, allowing processors to continue processing data as soon as their local task is complete.

Consider the case where each processor operates asynchronously. In this case, at an instantaneous time index t, only one processor executes an update, as all others are assumed to be busy. If two processors complete their prior task concurrently, then they draw the same time index at the next available slot, in which case the tie is broken at random. Suppose processor i selects block $b_i^t \in \{1, \ldots, B\}$ at time t. Then it grabs the associated component of the decision variable $\mathbf{x}_b^t$ and computes the stochastic gradient $\nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \boldsymbol{\Theta}_i^t)$ associated with the samples $\boldsymbol{\Theta}_i^t$. This process may take time and during this process other processors may overwrite the variable $\mathbf{x}_b$. Consider the case that the process time of computing stochastic gradient or equivalently the descent direction is τ. Thus, when processor i updates the block b using the evaluated stochastic gradient $\nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \boldsymbol{\Theta}_i^t)$, it performs the update

$$\mathbf{x}_b^{t+\tau+1} = \mathbf{x}_b^{t+\tau} - \gamma^{t+\tau} \nabla_{\mathbf{x}_b} f(\mathbf{x}^t, \boldsymbol{\Theta}_i^t) \qquad b = b_i^t. \tag{5}$$

Thus, the descent direction evaluated based on the available information at step t is used to update the variable at time $t+\tau$. Note that the delay comes from asynchronous implementation of the algorithm and the fact that other processors are able to modify the variable $\mathbf{x}_b$ during the time that processor i computes its descent direction. We assume the random time τ that each processor requires to compute its descent direction is bounded above by a constant Δ, i.e., $\tau \leq \Delta$ – see Assumption 4. Despite the minimal coordination of the asynchronous random parallel stochastic algorithm in (5), we establish the comparable performance guarantees to that of the classical SGD process for strongly-convex functions – see Section III.

Remark 1 One may raise the concern that there could be instances that two processors or more work on a same block. Although, this event is not very likely since $I << B$, there is a positive chance that it might happen. This is true since the available processor picks the block that it wants to operate on uniformly at random from the set $\{1, \ldots, B\}$. We show that this event does not cause any issues and the algorithm can eventually converge to the optimal argument even if more than one processor work on a specific block at the same time.

III. Convergence Analysis

In this section, we study the convergence of Asynchronous RAPSA and we characterize the effect of delay in the asynchronous implementation. All proofs are given in [1]. To do so, define the set $\mathcal{S}^t$ containing the blocks that are updated at step t with associated indices $\mathcal{I}^t \subset \{1, \ldots, B\}$. Then we may rewrite the asynchronous RAPSA update [cf. (5)] as

$$\mathbf{x}_i^{t+1} \;=\; \mathbf{x}_i^t - \gamma^t \; \nabla_{\mathbf{x}_i} f(\mathbf{x}^{t-\tau}, \boldsymbol{\Theta}_i^{t-\tau}) \qquad \forall \; \mathbf{x}_i \in \mathcal{S}^t, \tag{6}$$

while blocks $i \notin \mathcal{I}^t$ remain unchanged: $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t$, $\mathbf{x}_i \notin \mathcal{S}^t$.

Note that the random set $\mathcal{I}^t$ and the associated block set $\mathcal{S}^t$ are chosen at time $t - \tau$ in practice; however, for the sake of analysis we can assume that these sets are chosen at time t. In other words, we can assume that at step $t - \tau$ processor i computes the full (for all blocks) stochastic gradient $\nabla f(\mathbf{x}^{t-\tau}, \boldsymbol{\Theta}_i^{t-\tau})$ and after finishing this task at time t, it chooses uniformly at random the block that it wants to update. Thus, the block $\mathbf{x}_i$ in (6) is chosen at step t. This new interpretation of the update of asynchronous RAPSA is only important for convergence analysis.

Next we introduce technical conditions which are necessary to prove convergence.

(A1) The instantaneous objective functions $f(\mathbf{x}, \boldsymbol{\theta})$ are differentiable and the average function $F(\mathbf{x})$ is strongly convex with parameter $m > 0$.

(A2) The average objective function gradients $\nabla F(\mathbf{x})$ are Lipschitz continuous with respect to the Euclidian norm with parameter M. I.e., for all $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^p$, it holds

$$\|\nabla F(\mathbf{x}) - \nabla F(\hat{\mathbf{x}})\| \ \leq \ M \ \|\mathbf{x} - \hat{\mathbf{x}}\|. \tag{7}$$

(A3) The conditional second moment of the stochastic gradient is bounded for all $\mathbf{x}$, i.e., there exists a constant K such that for all variables $\mathbf{x}$, it holds

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\|\nabla f(\mathbf{x}^t, \boldsymbol{\theta}^t)\|^2 \,\middle|\, \mathbf{x}^t\right] \leq K. \tag{8}$$

(A4) The random variable τ which is the delay between reading and writing for processors does not exceed the constant Δ, i.e., $\tau \leq \Delta$.

Notice that Assumption (A1) only enforces strong convexity of the average function F, while the instantaneous functions f_i need not be convex. Further, notice that since the instantaneous functions f_i are differentiable the average function F is also differentiable. The Lipschitz continuity of the average function gradients ∇F is customary in proving objective function convergence for descent algorithms. The restriction imposed by Assumption (A3) is a standard condition in stochastic approximation literature [11], its intent being to limit the variance of the stochastic gradients [20]. The condition in Assumption (A4) implies that processors can finish their tasks in a time that is bounded by the constant Δ, which is typical in the analysis of asynchronous algorithms.

Our first result comes in the form of a expected descent lemma that relates the expected difference of subsequent iterates to the gradient of the instantanteous function, where the expectation is taken with respect to the block selection.

Lemma 1 *Consider the asynchronous random parallel stochastic algorithm defined in* (5)*. Recall the definitions of the set of updated blocks $\mathcal{I}^t$ which are randomly chosen from the total B blocks. Define $\mathcal{F}^t$ as a sigma algebra that measures the history of the system up until time t. Then, under Assumptions (A1) - (A4), the expected value of the difference $\mathbf{x}^{t+1} - \mathbf{x}^t$ with respect to the random set $\mathcal{I}^t$ given $\mathcal{F}^t$ is*

$$\mathbb{E}_{\mathcal{I}^t}\left[\mathbf{x}^{t+1} - \mathbf{x}^t \mid \mathcal{F}^t\right] = -\frac{\gamma^t}{B} \ \nabla f(\mathbf{x}^{t-\tau}, \boldsymbol{\Theta}^{t-\tau}). \tag{9}$$

Moreover, the expected value of the squared norm $\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2$ with respect to the random set $\mathcal{S}^t$ given $\mathcal{F}^t$ simplifies to

$$\mathbb{E}_{\mathcal{I}^t}\left[\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \mid \mathcal{F}^t\right] = \frac{(\gamma^t)^2}{B} \ \left\|\nabla f(\mathbf{x}^{t-\tau}, \boldsymbol{\Theta}^{t-\tau})\right\|^2. \tag{10}$$

In the asynchronous scheme only one of the blocks is updated at each iteration, so the ratio r can be simplified as $1/B$, which results in the coefficient $1/B$ on the right-hand side of the expressions in Lemma 1. We use these results to characterize the decrement in the expected sub-optimality in the following proposition.

Proposition 1 *Consider the asynchronous random parallel stochastic algorithm defined in* (5)*. If Assumptions (A1) - (A4) hold, then the objective error sequence $F(\mathbf{x}^t) - F(\mathbf{x}^*)$ satisfies*

$$\begin{aligned}\mathbb{E}\left[F(\mathbf{x}^{t+1}) - F(\mathbf{x}^*) \mid \mathcal{F}^{t-\tau}\right] \\ \leq \left(1 - \frac{2m\gamma^t}{B}\left[1 - \frac{\rho M}{2}\right]\right) \mathbb{E}\left[F(\mathbf{x}^t) - F(\mathbf{x}^*) \mid \mathcal{F}^{t-\tau}\right] \\ + \frac{MK(\gamma^t)^2}{2B} + \frac{\tau^2 MK\gamma^t(\gamma^{t-\tau})^2}{2\rho B^2}.\end{aligned} \tag{11}$$

The scalar parameter ρ comes from an application of a generalized triangle inequality expression of the form $ab \leq (\rho/2)a^2 + (1/2\rho)b^2$ with $a = \|\nabla F(\mathbf{x}^t)\|$ and $b = \sum_{s=t-\tau}^{t-1} \|\mathbf{x}^{s+1} - \mathbf{x}^s\|$ that appears in the proof. We proceed to use the result in Proposition 1 to prove that the sequence of iterates generated by asynchronous RAPSA converges to the optimal argument $\mathbf{x}^*$ defined by (2).

Theorem 1 *If Assumptions (A1) - (A4) hold true and the sequence of stepsizes are non-summable $\sum_{t=0}^{\infty} \gamma^t = \infty$ and square summable $\sum_{t=0}^{\infty} (\gamma^t)^2 < \infty$, then sequence of the variables $\{\mathbf{x}^t\}$ generated by RAPSA converges almost surely to the optimal argument $\mathbf{x}^*$,*

$$\liminf_{t \to \infty} \|\mathbf{x}^t - \mathbf{x}^*\|^2 \ = \ 0 \qquad a.s. \tag{12}$$

If step-size is defined as $\gamma^t := \gamma^0 T^0/(t + T^0)$ and the step-size parameters are chosen such that $2mr\gamma^0 T^0 > 1$, then the expected average function error $\mathbb{E}\left[F(\mathbf{x}^t) - F(\mathbf{x}^*)\right]$ converges to null at least with a sublinear convergence rate $O(1/t)$,

$$\mathbb{E}\left[F(\mathbf{x}^t) - F(\mathbf{x}^*)\right] \ \leq \ \frac{C}{t + T^0}, \tag{13}$$

where the constant C is defined as

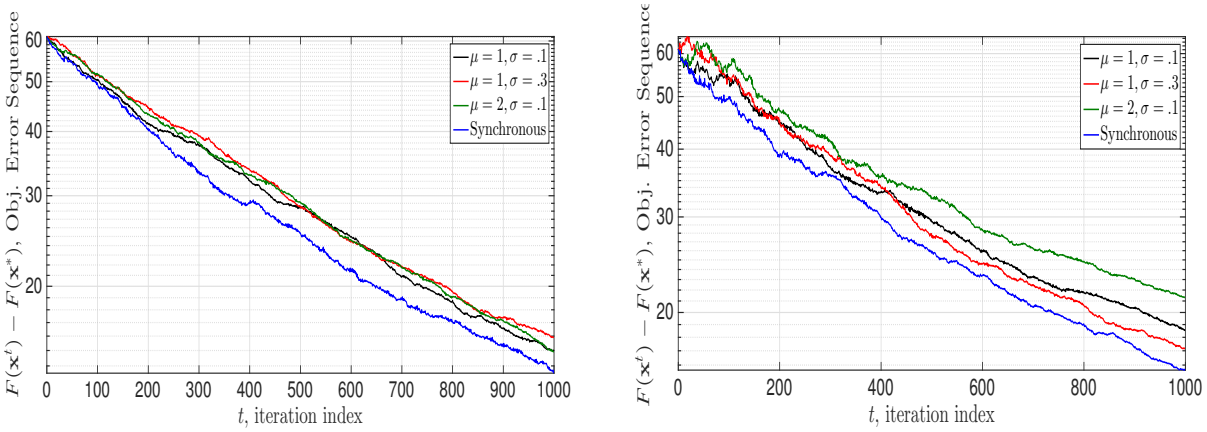
$$\begin{aligned}C = \max\Big\{\frac{MK(\gamma^0 T^0)^2/2B + (\tau^2 MK(\gamma^0 T^0)^3)(2\rho B^2)}{(2m\gamma^0 T^0/B)(1 - \rho M/2) - 1}, \\ T^0(F(\mathbf{x}^0) - F(\mathbf{x}^*))\Big\}\end{aligned} \tag{14}$$

The result in Theorem 1 shows that when the sequence of stepsize is diminishing as $\gamma^t = \gamma^0 T^0/(t + T^0)$, the average objective function value $F(\mathbf{x}^t)$ sequence converges to the optimal objective value $F(\mathbf{x}^*)$ with probability 1.[1] Further, the rate of convergence in expectation is at least in the order of $O(1/t)$. Moreover, the sequence $\mathbf{x}^t$ converges exactly to the optimal $\mathbf{x}^*$. These results for the attenuating step-size regime are comparable to those which are attainable by stochastic gradient method in the strongly convex case.

IV. Numerical analysis

In this section we study the numerical performance of the doubly stochastic approximation algorithm developed in Section II by first considering a linear regression problem. We then use RAPSA to develop an automated decision system to distinguish between distinct hand-written digits.

[1]The expectation on the left hand side of (13), and throughout the subsequent convergence rate analysis, is taken with respect to the algorithm history $\mathcal{F}_0$, which contains all randomness in both $\boldsymbol{\Theta}_t$ and $\mathcal{I}_t$ for all $t \geq 0$.



(a) Sub-optimality $F(\mathbf{x}^t) - F(\mathbf{x}^*)$ vs. iteration t

(b) Sub-optimality $F(\mathbf{x}^t) - F(\mathbf{x}^*)$ vs. iteration t

Fig. 1: Synchronous and Asynchronous RAPSA on the linear estimation problem in the constant ($\gamma = 10^4$, left) and diminishing ($\gamma_t = 10^6/(t+250)$, right) step-size schemes with no mini-batching $L = 1$ for a binary training subset of size $N = 10^3$ with no regularization $\lambda = 0$ when the algorithm is initialized as $\mathbf{x}_0 = 10^3 \times \mathbf{1}$. Varying the asynchronicity distribution has little effect, but we find that convergence behavior is slower than its synchronized counterpart, as expected.

A. Linear Regression

We consider a setting in which observations $\mathbf{z}_n \in \mathbb{R}^q$ are collected which noisy linear transformations $\mathbf{z}_n = \mathbf{H}_n\mathbf{x} + \mathbf{w}_n$ of a signal $\mathbf{x} \in \mathbb{R}^p$ which we would like to estimate, and $\mathbf{w} \sim \mathcal{N}(0, \sigma^2 I_q)$ is a Gaussian random variable. For a finite set of samples N, the optimal $\mathbf{x}^*$ is computed as the least squares estimate $\mathbf{x}^* := \operatorname{argmin}_{\mathbf{x}\in\mathbb{R}^p}(1/N)\sum_{n=1}^N \|\mathbf{H}_n\mathbf{x} - \mathbf{z}_n\|^2$. We run RAPSA on LMMSE estimation problem instances where observations are of dimension $q = 1$. The observation matrices $\mathbf{H}_n \in \mathbb{R}^{q\times p}$, when stacked over all n (an $N \times p$ matrix), are according to a matrix normal distribution whose mean is a tri-diagonal matrix. The main diagonal is 2, while the super and sub-diagonals are all set to $-1/2$. Moreover, the true signal has entries chosen uniformly at random from the fractions $\mathbf{x} \in \{1, \dots, p\}/p$. Additionally, the noise variance perturbing the observations is set to $\sigma^2 = 10^{-2}$. We assume that the number of processors $I = 16$ is fixed and each processor is in charge of 1 block. We consider different number of blocks $B = \{16, 32, 64, 128\}$. Note that when the number of blocks is B, there are $p/B = 1024/B$ coordinates in each block.

The model we use for asynchronicity is modeled after a random delay phenomenon in physical communication systems that in which each local server has a distinct local clock which is not required coincide with others. Each processor's clock begins at time $t_0^i = t_0$ for all processors $i = 1, \dots, I$ and selects subsequent times as $t_k = t_{k-1} + w_k^i$, where $w_k^i \sim \mathcal{N}(\mu, \sigma^2)$ is a normal random variable with mean μ and variance σ^2. The variance in this model effectively controls the amount of variability between the clocks of distinct processors.

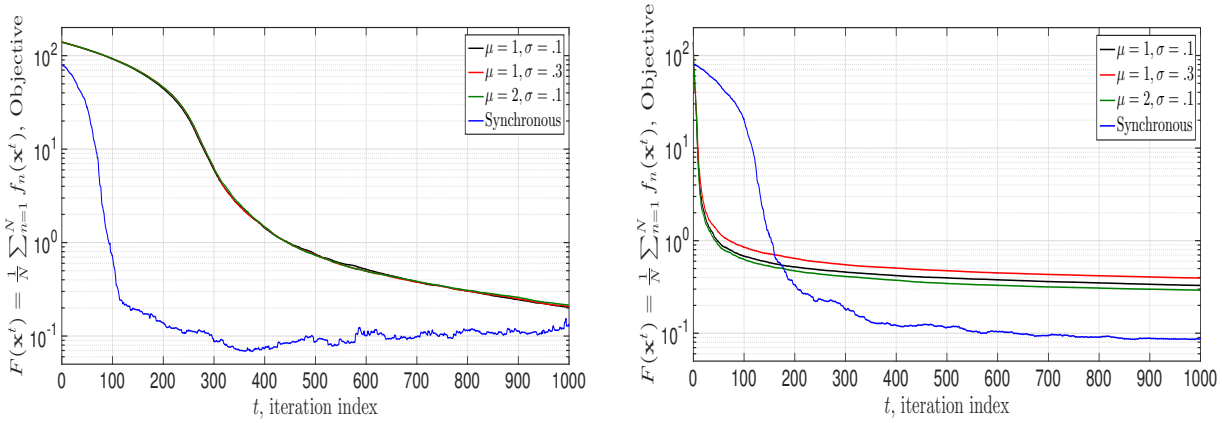
We run Asynchronous RAPSA for the linear estimation problem when the parameter vector $\mathbf{x}$ is $p = 500$ dimensional for $N = 10^3$ iterations with no mini-batching $L = 1$ for both the case that the algorithm step-size is diminishing and constant step-size regimes. The algorithm is initialized as $\mathbf{x}_0 = 10^3 \times \mathbf{1}$. We run the algorithm for a few different instantiations of asynchronicity, that is, $w_k^i \sim \mathcal{N}(\mu, \sigma^2)$ with $\mu = 1$ or $\mu = 2$, and $\sigma = .1$ or $\sigma = .3$.

The results of this numerical experiment are given in Figure 1 for both the constant and diminishing step-size schemes. We see that the performance of the asynchronous parallel scheme is comparable across different levels of variability among the local clocks of each processor. In particular, in Figure 1a which corresponds to the case where the algorithm is run with constant step-size $\gamma = 10^{-2}$, we observe comparable performance in terms of the objective function error sequence $F(\mathbf{x}^t) - F(\mathbf{x}^*)$ with iteration t – across the varying levels of asynchrony we have $F(\mathbf{x}^t) - F(\mathbf{x}^*) \leq 10$ by $t = 10^3$. This trend may also be observed in the diminishing step-size scheme $\gamma^t = 1/t$ which is given in Figure 1b. That is, the distance to the optimal objective is nearly identical across differing levels of asynchronicity. In both cases, the synchronized algorithm performs better than its asynchronous counterpart.

B. Hand-Written Digit Recognition

We now make use of RAPSA for digit classification. To do so, let $\mathbf{z} \in \mathbb{R}^p$ be a feature vector encoding pixel intensities of an image and let $y \in \{-1, 1\}$ be an indicator variable of whether the image contains the digit 0 or 8, in which case the binary indicator is respectively $y = -1$ or $y = 1$. We model the task of learning a hand-written digit detector as a logistic regression problem, where one aims to train a classifier $\mathbf{x} \in \mathbb{R}^p$ to determine the relationship between feature vectors $\mathbf{z}_n \in \mathbb{R}^p$ and their associated labels $y_n \in \{-1, 1\}$ for $n = 1, \dots, N$. The instantaneous function f_n in (1) for this setting is the regularized negative log-likelihood of a generalized linear model of the odds ratio of whether the label is $y_n = 1$ or $y_n = -1$. The empirical risk minimization associated with training set $\mathcal{T} = \{(\mathbf{z}_n, y_n)\}_{n=1}^N$ is to find $\mathbf{x}^*$ as the maximum a posteriori estimate

$$\mathbf{x}^* := \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^p} \frac{\lambda}{2}\|\mathbf{x}\|^2 + \frac{1}{N}\sum_{n=1}^N \log(1 + \exp(-y_n\mathbf{x}^T\mathbf{z}_n)) \,, \tag{15}$$



(a) Objective $F(\mathbf{x}^t)$ vs. iteration t

(b) Objective $F(\mathbf{x}^t)$ vs. iteration t

Fig. 2: Asynchronous RAPSA on MNIST data in the constant ($\gamma = 10^{-2}$, left) and diminishing ($\gamma^t = 1/t$, right) step-size schemes with no mini-batching $L = 1$ for a binary training subset of size $N = 10^3$ with no regularization $\lambda = 0$ when the algorithm is initialized as $\mathbf{x}_0 = \mathbf{1}$. The variability in local processor clocks does not significantly impact performance in both the diminishing and constant step-size settings; however, the synchronous algorithm converges at a faster rate.

where the regularization term $(\lambda/2)\|\mathbf{x}\|^2$ is added to avoid overfitting. We use the MNIST dataset [21], in which feature vectors $\mathbf{z}_n \in \mathbb{R}^p$ are $p = 28^2 = 784$ pixel images whose values are recorded as intensities, or elements of the unit interval $[0, 1]$. We consider a subset associated with digits 0 and 8.

The model we use for asynchronicity is the one outlined in Section IV-A, that is, each local processor has a distinct local clock which is not required coincide with others, begins at time $t_0^i = t_0$ for all processors $i = 1, \ldots, I$, and then selects subsequent times as $t_k = t_{k-1} + w_k^i$. Here $w_k^i \sim \mathcal{N}(\mu, \sigma^2)$ is a normal random variable with mean μ and variance σ^2 which controls the amount of variability between the clocks of distinct processors. We run the algorithm with no regularization $\lambda = 0$ or mini-batching $L = 1$ and initialization $\mathbf{x}_0 = \mathbf{1}$.

The results of this numerical setup are given in Figure 2. We consider the expected risk $F(\mathbf{x}^t)$ in both both the constant ($\gamma = 10^{-2}$, Figure 2a) and diminishing ($\gamma^t = 1/t$, Figure 2b) algorithm step-size schemes. We see that the level of asynchronicity does not significantly impact the performance in either scheme, and that the convergence guarantees established in Theorem 1 hold true in practice.

REFERENCES

[1] A. Mokhtari, A. Koppel, and A. Ribeiro, "A class of parallel doubly stochastic algorithms for large-scale learning," *arXiv preprint arXiv*, 1606.04991, 2016.

[2] G. Sampson, R. Haigh, and E. Atwell, "Natural language analysis by stochastic optimization: A progress report on project april," *J. Exp. Theor. Artif. Intell.*, vol. 1, no. 4, pp. 271–287, Oct. 1990. [Online]. Available: http://dx.doi.org/10.1080/09528138908953710

[3] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[4] M. Taşan, G. Musso, T. Hao, M. Vidal, C. A. MacRae, and F. P. Roth, "selecting causal genes from genome-wide association studies via functionally coherent subnetworks," *Nature methods*, 2014.

[5] P. Tseng and C. O. L. Mangasarian, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim Theory Appl*, pp. 475–494, 2001.

[6] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *arXiv preprint arXiv:1410.1386*, 2014.

[7] Z. Lu and L. Xiao, "On the complexity analysis of randomized block-coordinate descent methods," *Mathematical Programming*, pp. 1–28, 2013.

[8] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[9] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, "An asynchronous parallel stochastic coordinate descent algorithm," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 285–322, 2015.

[10] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel selective algorithms for nonconvex big data optimization," *Signal Processing, IEEE Transactions on*, vol. 63, no. 7, pp. 1874–1889, 2015.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400–407, 09 1951. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177729586

[12] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-newton method for online convex optimization," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 436–443.

[13] A. Bordes, L. Bottou, and P. Gallinari, "Sgd-qn: Careful quasi-newton stochastic gradient descent," *The Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, 2009.

[14] A. Mokhtari and A. Ribeiro, "Res: Regularized stochastic bfgs algorithm," *Signal Processing, IEEE Transactions on*, vol. 62, no. 23, pp. 6089–6104, 2014.

[15] ——, "Global convergence of online limited memory bfgs," *Journal of Machine Learning Research*, vol. 16, pp. 3151–3181, 2015. [Online]. Available: http://jmlr.org/papers/v16/mokhtari15a.html

[16] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.

[17] Y. Yang, G. Scutari, and D. P. Palomar, "Parallel stochastic decomposition algorithms for multi-agent systems," in *Signal Processing Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on*. IEEE, 2013, pp. 180–184.

[18] A. Koppel, B. M. Sadler, and A. Ribeiro, "Proximity without consensus in online multi-agent optimization." in *Proc. Int. Conf. Acoust. Speech Signal Process.*, March 20-25 2016.

[19] A. Koppel, F. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Trans. Signal Process.*, p. 15, Oct 2015.

[20] A. Nemirovski, A. Juditsky, and A. Shapiro, "Robust stochastic approximation approach to stochastic programming," *SIAM Journal on optimization*, vol. 19, no. 4, pp. 1574–1609, 2009.

[21] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/