ONLINE DICTIONARY LEARNING FROM BIG DATA USING ACCELERATED STOCHASTIC APPROXIMATION ALGORITHMS

Konstantinos Slavakis and Georgios B. Giannakis

Dept. of ECE and Digital Technology Center, University of Minnesota, USA
Emails: {kslavaki,georgios}@umn.edu

ABSTRACT

Applications involving large-scale dictionary learning tasks motivate well online optimization algorithms for generally non-convex and non-smooth problems. In this big data context, the present paper develops an online learning framework by jointly leveraging the stochastic approximation paradigm with first-order acceleration schemes. The generally non-convex objective evaluated online at the resultant iterates enjoys quadratic rate of convergence. The generality of the novel approach is demonstrated in two online learning applications: (i) Online linear regression using the total least-squares approach; and, (ii) a semi-supervised dictionary learning approach to network-wide link load tracking and imputation of real data with missing entries. In both cases, numerical tests highlight the potential of the proposed online framework for big data network analytics.

1. INTRODUCTION

As pervasive sensors collect and record massive amounts of high-dimensional data from communication and social networks, and storage along with processing capacities of computers grow, new analytical tools are necessary to comb through these "big data" sets to distill out subsets of interest. Further, as many data sources continuously generate data in real time, analytics must often be performed in real time, without a chance to revisit past entries.

Given an $M \times 1$ real data vector $\boldsymbol{y}_k \in \mathbb{R}^M$, indexed by $k \in \mathbb{N}$ (the set of non-negative integers), dictionary learning (DL) has emerged as a prominent tool for modeling $\boldsymbol{y}_k$ as a product of an unknown over-complete dictionary $\boldsymbol{D} := [\boldsymbol{d}_1, \ldots, \boldsymbol{d}_Q] \in \mathbb{R}^{M\times Q}$, $Q \geq M$, times an unknown sparse coefficient vector $\boldsymbol{s}_k \in \mathbb{R}^Q$ [1–3]. If $\boldsymbol{D}$ were known, basis pursuit would yield $\boldsymbol{s}_k$ [4]; but with $\boldsymbol{D}$ unknown, one can use multiple vectors in $\boldsymbol{Y}_K := [\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{K-1}]$ to solve

$$\min_{(\boldsymbol{S}_K, \boldsymbol{D}) \in \mathbb{R}^{Q\times K}\times \mathscr{D}} \frac{1}{2}\|\boldsymbol{Y}_K - \boldsymbol{D}\boldsymbol{S}_K\|_{\mathrm{F}}^2 + \lambda_s \|\boldsymbol{S}_K\|_1 \qquad (1)$$

where $\boldsymbol{S}_K := [\boldsymbol{s}_0, \ldots, \boldsymbol{s}_{K-1}]$, $\mathscr{D} := \{\boldsymbol{D} \in \mathbb{R}^{M\times Q} \mid \|\boldsymbol{d}_q\| \leq 1, q \in \{1, \ldots, Q\}\}$, $\lambda_s \in \mathbb{R}_{>0}$, and $\|\cdot\|_{\mathrm{F}}$, $\|\cdot\|_1$ denote the Frobenius and ℓ_1-norms, respectively. The unit-norm constraint on the columns of $\boldsymbol{D}$ is incorporated to cope with the inherent scale ambiguity of the bilinear fit, and also ensure that the solution of (1) remains bounded.

If M is excessively large, solvers of (1) remain tractable either after splitting $\boldsymbol{Y}_K$ in multiple row sub-blocks, and running DL per sub-block on parallel processors; or, by simply down-sampling (a.k.a. sketching) the rows of $\boldsymbol{Y}_K$. Thanks to the sparsity of $\boldsymbol{S}_K$, DL remains tractable also when the collected data vectors have missing entries for reasons such as privacy, storage limitations or due to the high cost of data gathering [5]. But even when M is of manageable size, the streaming nature of data presents major challenges to solving (1) as K grows.

Although the cost in (1) is non-convex, owing to the bilinear form $\boldsymbol{D}\boldsymbol{S}_K$, it is convex in each of its arguments, $\boldsymbol{S}_K$ or $\boldsymbol{D}$, if the other one is held fixed. Block coordinate descent methods (BCDMs) have recently gained popularity in the big data optimization area [2, 3, 6–17], largely because they exploit structure of the objective functions, they have low memory requirements, and also incur low cost per iteration. BCDMs optimize (exactly or inexactly) the objective function over one (block) variable at a time, while holding all other fixed. However, the computationally expensive and time-consuming quest for *(almost) exact* minimizers per BCDM iteration can be prohibitive in the big data context, where the streaming nature and sheer dimensionality of data dictate stringent policies on computational power and CPU time. Notwithstanding the importance of accuracy in estimation, the sequential nature and non-stationarity of data places *time-adaptivity* attributes of algorithms at a central position; *online* solutions should "monitor" their batch counterparts, without "over-fitting" them, since that would degrade their agility to track time-varying and non-stationary big-data processes.

Projected proximal stochastic (sub)gradient methods are attractive low-complexity online alternatives to BCDMs mainly for optimizing convex objective functions [18–22]. Unfortunately, such first-order solutions tend to exhibit slow convergence even for convex problems, due to their perplexed means of choosing step-size parameters, which become increasingly complicated as the objective functions become more complex. On the other hand, accelerated variations for convex problems enjoy provable quadratic convergence rate of the objective function values, meaning they are optimal among first-order methods [23–28]. However, convergence claims for non-convex objective functions are largely uncharted territories.

The present work introduces a scalable, online, and low-complexity iterative learning approach for a class of non-convex optimization tasks, going beyond and subsuming (1). Using only first-order information of the underlying objective function, the convergence rate is proved to be quadratic. The novel algorithm employs the method of [26] as a starting point, developed originally for convex loss functions and equipped with flexibility in parameter selection. Subsequently, it adapts stochastic approximation (SA) [29] tools to ensure generalization to out-of-sample data. The analytical results are tested on two instances of broad practical interest: Online, robust, linear regression based on the total least-squares (TLS) criterion; and the online semi-supervised DL approach put forth in [5] for network-wide link load tracking and imputation.

Work in this paper was supported by the NSF Eager Grant 1343860. Moreover, it has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thalis - UoA - Secure Wireless Nonlinear Communications at the Physical Layer.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

2. PROBLEM STATEMENT

Although the proposed approach and theoretical claims apply to any finite number of block variables, exposition here will focus for brevity and simplicity on only two blocks, namely $(x^{(1)}, x^{(2)}) \in \mathscr{H}_1 \times \mathscr{H}_2$, where $\mathscr{H}_1, \mathscr{H}_2$ are any finite-dimensional linear vector spaces, with inner products $\langle \cdot, \cdot \rangle_{\mathscr{H}_1}$ and $\langle \cdot, \cdot \rangle_{\mathscr{H}_2}$, respectively.

With reference to (1), consider the following sequence of loss functions $(F_k : \mathscr{H}_1 \times \mathscr{H}_2 \to \mathbb{R} \cup \{+\infty\})_{k \in \mathbb{N}}$, defined as $F_k(x^{(1)}, x^{(2)}) := f_k(x^{(1)}, x^{(2)}) + g_1(x^{(1)}) + g_2(x^{(2)})$, where $f_k : \mathscr{H}_1 \times \mathscr{H}_2 \to \mathbb{R}$ is non-convex, while $g_i \in \Gamma_0(\mathscr{H}_i)$, where $\Gamma_0(\mathscr{H}_i)$ stands for all (proper lower semicontinuous) convex functions [30] defined on $\mathscr{H}_i$ with values in $\mathbb{R} \cup \{+\infty\}$, $i \in \{1, 2\}$. Since $\{g_1, g_2\}$ can be non-differentiable, F_k is generally non-smooth. The features of the DL problem in (1) appear in f_k. Indeed, $\forall k$ function f_k is assumed convex with respect to (w.r.t.) each one of its arguments, whenever the other one is fixed. Finally, f_k is assumed Lipschitz continuously differentiable in $x^{(1)}$ and $x^{(2)}$, with (local) Lipschitz constants $L_{f_k|x^{(1)}}(x^{(2)})$ and $L_{f_k|x^{(2)}}(x^{(1)})$, respectively.

To facilitate convergence analysis on the premises of stochastic approximation (SA) [29], the following stationarity assumption on $(F_k)_{k \in \mathbb{N}}$ is also adopted. (Generalizations are possible but go beyond the scope of this paper.)

Assumption 1. There exists a function $F : \mathscr{H}_1 \times \mathscr{H}_2 \to \mathbb{R} \cup \{+\infty\}$ such that (s.t.) the following stationarity holds true:

$$\mathbb{E}_{\xi|(x^{(1)}, x^{(2)})}\{F_k(x^{(1)}, x^{(2)}, \xi)\} = F(x^{(1)}, x^{(2)}), \quad \forall k \in \mathbb{N}$$

where ξ denotes a trial within the ensemble of trials Ξ of a probability space [31] that characterizes randomness in F_k originating from the observed data. Moreover, $\mathbb{E}_{\xi|(x^{(1)}, x^{(2)})}$ denotes expectation w.r.t. ξ, conditioned on $(x^{(1)}, x^{(2)})$ held fixed, since the arguments of F_k are in general vector- or matrix-valued random variables (r.vs.) $x^{(i)} : \Xi \to \mathscr{H}_i$. Indeed, this is true on the premises of Alg. 1, where $(x_k^{(1)}, x_k^{(2)})$ at time k generally depend on the observed data for time instants $\{0, 1, \ldots, k-1\}$.

For concreteness, examples of principal practical interest follow.

Example 1 (Total least-squares)**.** A special case of (1) with $M = 1$ is the linear regression model $y_k = \boldsymbol{u}_{*k}^\top \boldsymbol{s}_* + v_k$, $k \in \mathbb{N}$, where $(\boldsymbol{u}_{*k}, \boldsymbol{s}_*) \in \mathbb{R}^Q \times \mathbb{R}^Q$, with the regressor vector $\boldsymbol{u}_{*k}$ assumed random; $\cdot^\top$ denotes transposition; $\boldsymbol{s}_*$ is assumed sparse, and v_k stands for noise. Observed data are $(y_k, \boldsymbol{u}_k)_{k \in \mathbb{N}} \subset \mathbb{R} \times \mathbb{R}^Q$, where $\boldsymbol{u}_k$ is a noisy version of the true $\boldsymbol{u}_{*k}$. Here, $\mathscr{H}_1 = \mathscr{H}_2 = \mathbb{R}^Q$, with $\langle \cdot, \cdot \rangle_{\mathbb{R}^Q}$ being the dot-vector product. Following the *total least-squares (TLS)* criterion and the resultant *errors-in-variables (EIV)* modeling approach [32], the following sequence of functions is considered; $\forall k \in \mathbb{N}$:

$$F_k(\boldsymbol{s}, \boldsymbol{e}) := \underbrace{\frac{1}{2}\left[y_k - (\boldsymbol{u}_k + \boldsymbol{e})^\top \boldsymbol{s}\right]^2}_{f_k(\boldsymbol{s}, \boldsymbol{e})} + \underbrace{\lambda_s \|\boldsymbol{s}\|_1}_{g_1(\boldsymbol{s})} + \underbrace{\frac{\lambda_e}{2}\|\boldsymbol{e}\|^2}_{g_2(\boldsymbol{e})}, \quad (2)$$

where $\boldsymbol{e} \in \mathbb{R}^Q$ models EIV, and $\lambda_s, \lambda_e \in \mathbb{R}_{>0}$. Notice that a Lipschitz constant of $\nabla_{\boldsymbol{s}} f_k$ is $L_{f_k|\boldsymbol{s}}(\boldsymbol{e}) = \|(\boldsymbol{u}_k + \boldsymbol{e})(\boldsymbol{u}_k + \boldsymbol{e})^\top\| \leq \|(\boldsymbol{u}_k + \boldsymbol{e})(\boldsymbol{u}_k + \boldsymbol{e})^\top\|_{\mathrm{F}} = \|\boldsymbol{u}_k + \boldsymbol{e}\|^2$, where $\|\boldsymbol{A}\|$ denotes the spectral norm of a matrix $\boldsymbol{A}$, and $f_k|_{\boldsymbol{s}}$ the restriction of f_k on the $\boldsymbol{s}$-domain. On the other hand, a Lipschitz constant of $\nabla_{\boldsymbol{e}} f_k$ is $L_{f_k|\boldsymbol{e}}(\boldsymbol{s}) = \|\boldsymbol{s}\boldsymbol{s}^\top\| \leq \|\boldsymbol{s}\boldsymbol{s}^\top\|_{\mathrm{F}} = \|\boldsymbol{s}\|^2$.

It is not hard to verify that

$$\begin{aligned} &\mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{F_k(\boldsymbol{s}, \boldsymbol{e}, \xi)\} \\ &= \frac{1}{2}\boldsymbol{s}^\top \boldsymbol{e}\boldsymbol{e}^\top \boldsymbol{s} + \boldsymbol{s}^\top \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{\boldsymbol{u}_k\}\boldsymbol{e}^\top \boldsymbol{s} \\ &\quad + \frac{1}{2}\boldsymbol{s}^\top \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{\boldsymbol{u}_k \boldsymbol{u}_k^\top\}\boldsymbol{s} + \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{y_k\}\boldsymbol{e}^\top \boldsymbol{s} \\ &\quad + \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{y_k \boldsymbol{u}_k^\top\}\boldsymbol{s} + \frac{1}{2}\mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{y_k^2\}. \end{aligned}$$

Assuming that all expected values remain invariant w.r.t. k, then Assumption 1 holds if $F(\boldsymbol{s}, \boldsymbol{e}) := \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{e})}\{F_k(\boldsymbol{s}, \boldsymbol{e}, \xi)\}$.

Example 2 (Semi-supervised dictionary learning [5])**.** Consider an undirected weighted graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ denotes the vertex set, with cardinality $P \in \mathbb{N}_*$, and $\mathscr{E}$ is the edge set. Connectivity and edge strengths of $\mathscr{G}$ are described by the adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{P \times P}$, where $[\boldsymbol{W}]_{ij} > 0$ if nodes n_i and n_j are connected, while $[\boldsymbol{W}]_{ij} = 0$ otherwise. At every $k \in \mathbb{N}$, a variable $\chi_{kp} \in \mathbb{R}$, which describes a network-wide dynamical process of interest, corresponds to a node n_p. All node variables are collected in a single vector $\boldsymbol{\chi}_k := [\chi_{k1}, \ldots, \chi_{kP}]^\top \in \mathbb{R}^P$. A sparse representation of the process over $\mathscr{G}$ models $\boldsymbol{\chi}_k$ as a linear combination of "few" atoms in a dictionary $\boldsymbol{D} \in \mathbb{R}^{P \times Q}$, $Q \geq P$: $\boldsymbol{\chi}_k = \boldsymbol{D}\boldsymbol{s}_k$, where $\boldsymbol{s}_k \in \mathbb{R}^Q$ is sparse. Further, only a portion of $\boldsymbol{\chi}_k$ is observed per time slot k. Let now $\boldsymbol{M}_k \in \mathbb{R}^{M \times P}$, $M < P$, denote a binary measurement matrix, with each row of $\boldsymbol{M}_k$ corresponding to the canonical basis vector for $\mathbb{R}^P$, selecting the measured components of $\boldsymbol{y}_k \in \mathbb{R}^M$. In other words, the observed data per slot k are $\boldsymbol{y}_k = \boldsymbol{M}_k \boldsymbol{\chi}_k + \boldsymbol{v}_k$, where $\boldsymbol{v}_k$ denotes noise. To enable imputation of missing entries of $\boldsymbol{\chi}_k$ in $\boldsymbol{y}_k$, the topology of $\mathscr{G}$ will be utilized. The spatial correlation of the process is captured by the Laplacian matrix $\boldsymbol{L} := \mathrm{diag}(\boldsymbol{W}\boldsymbol{1}_P) - \boldsymbol{W}$, where $\boldsymbol{1}_P \in \mathbb{R}^P$ is the all-ones vector. In this setting, $\mathscr{H}_1 = \mathbb{R}^Q$, with $\langle \cdot, \cdot \rangle_{\mathscr{H}_1}$ denoting the dot-vector product, and $\mathscr{H}_2 = \mathbb{R}^{P \times Q}$, with $\langle \boldsymbol{D}_1, \boldsymbol{D}_2 \rangle_{\mathscr{H}_2} = \mathrm{trace}(\boldsymbol{D}_1^\top \boldsymbol{D}_2)$, $\forall (\boldsymbol{D}_1, \boldsymbol{D}_{\boldsymbol{2}}) \in \mathscr{H}_2^2$.

Given a "forgetting factor" $\delta \in (0, 1]$ to gradually diminish the effect of past data (and thus account for non-stationarity), define

$$\begin{aligned} F_k(\boldsymbol{s}, \boldsymbol{D}) := {}& \overbrace{\frac{1}{2\Delta_k}\sum_{\kappa=0}^{k}\delta^{k-\kappa}\|\boldsymbol{y}_\kappa - \boldsymbol{M}_\kappa \boldsymbol{D}\boldsymbol{s}\|^2 + \frac{\lambda_L}{2}\boldsymbol{s}^\top \boldsymbol{D}^\top \boldsymbol{L}\boldsymbol{D}\boldsymbol{s}}^{f_k(\boldsymbol{s}, \boldsymbol{D})} \\ & + \underbrace{\lambda_s \|\boldsymbol{s}\|_1}_{g_1(\boldsymbol{s})} + \underbrace{\iota_{\mathscr{D}}(\boldsymbol{D})}_{g_2(\boldsymbol{D})} \end{aligned} \quad (3)$$

where $\Delta_k := \sum_{\kappa=0}^{k}\delta^{k-\kappa}$, and $\iota_{\mathscr{D}}$ stands for the indicator function of $\mathscr{D}$, i.e., $\iota_{\mathscr{D}}(\boldsymbol{D}) = 0$ if $\boldsymbol{D} \in \mathscr{D}$, and $\iota_{\mathscr{D}}(\boldsymbol{D}) = +\infty$ if $\boldsymbol{D} \notin \mathscr{D}$. The term including the known $\boldsymbol{L}$ quantifies a priori information on the topology of $\mathscr{G}$, and promotes "smooth" solutions over strongly connected nodes of $\mathscr{G}$ [5]. This term is also instrumental in accommodating missing entries in $(\boldsymbol{\chi}_k)_{k \in \mathbb{N}}$.

It can be easily verified that

$$\begin{aligned} &\mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{D})}\{F_k(\boldsymbol{s}, \boldsymbol{D}, \xi)\} \\ &\quad = \frac{1}{2\Delta_k}\sum_{\kappa=0}^{k}\delta^{k-\kappa}\Big[\boldsymbol{s}^\top \boldsymbol{D}^\top \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{D})}\{\boldsymbol{M}_\kappa^\top \boldsymbol{M}_\kappa\}\boldsymbol{D}\boldsymbol{s} \\ &\qquad -2\,\mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{D})}\{\boldsymbol{y}_\kappa^\top \boldsymbol{M}_\kappa\}\boldsymbol{D}\boldsymbol{s} + \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{D})}\{\|\boldsymbol{y}_\kappa\|^2\}\Big] \\ &\qquad + \frac{\lambda_L}{2}\boldsymbol{s}^\top \boldsymbol{D}^\top \boldsymbol{L}\boldsymbol{D}\boldsymbol{s} + \lambda_s \|\boldsymbol{s}\|_1 + \iota_{\mathscr{D}}(\boldsymbol{D}). \end{aligned} \quad (4)$$

As before, if all expected values are invariant w.r.t. k, then Assumption 1 holds true with $F(\boldsymbol{s}, \boldsymbol{D}) := \mathbb{E}_{\xi|(\boldsymbol{s}, \boldsymbol{D})}\{F_k(\boldsymbol{s}, \boldsymbol{D}, \xi)\}$.

Notice that a Lipschitz constant of $\nabla_{\boldsymbol{s}} f_k$ is $L_{f_k|\boldsymbol{s}}(\boldsymbol{D}) = \|\boldsymbol{D}^\top \boldsymbol{A}_k \boldsymbol{D}\| \leq \|\boldsymbol{D}^\top \boldsymbol{A}_k \boldsymbol{D}\|_{\mathrm{F}} \leq \|\boldsymbol{D}\|_{\mathrm{F}}^2 \|\boldsymbol{A}_k\|_{\mathrm{F}}$, where $\boldsymbol{A}_k := \Delta_k^{-1}\sum_{\kappa=0}^{k}\delta^{k-\kappa}\boldsymbol{M}_\kappa^\top \boldsymbol{M}_\kappa + \lambda_L \boldsymbol{L}$; whereas a Lipschitz constant of $\nabla_{\boldsymbol{D}} f_k$ is $L_{f_k|\boldsymbol{D}}(\boldsymbol{s}) = \|\boldsymbol{s}\|^2 \|\boldsymbol{A}_k\|_{\mathrm{F}}$.

3. ALGORITHM

This section introduces Alg. 1, which is built on a few basic notions outlined next.

```
1  Choose any initial points $(x_0^{(i)}, y_1^{(i)}, \eta_1^{(i)})$, $i \in \{1,2\}$;
2  for $k = 1$ to $+\infty$ do
3    for $i = 1$ to $2$ do
4      if $\min_{x^{(i)}} f_k(x^{(i)} \mid x_{k+i-2}^{(-i)}) + g_i(x^{(i)})$ is feasible then
5          $x_k^{(i)} \in \arg\min_{x^{(i)}} f_k(x^{(i)} \mid x_{k+i-2}^{(-i)}) + g_i(x^{(i)})$;
6      else
         /* Acceleration for $i$-th block of coordinates:        */
         /* Initialization:                                      */
7        $(x_0, y_1, \eta_0) := (x_{k-1}^{(i)}, y_k^{(i)}, \eta_k^{(i)})$;
         /* Perform $R_i$ cycles of (5):                          */
8        for $r = 1$ to $R_i$ do
9          $\Pi_r := (x_{r-1}, y_r, \eta_r)$;
10         $(x_r, y_{r+1}, \eta_{r+1}) \leftarrow \mathrm{Accel}\big(f_k(\cdot \mid x_{k+i-2}^{(-i)}), g_i, \Pi_r\big)$;
11       end
         /* Update $i$-th block of coordinates:                  */
12       $(x_k^{(i)}, y_{k+1}^{(i)}, \eta_{k+1}^{(i)}) := (x_{R_i}, y_{R_i+1}, \eta_{R_i})$;
13     end
14   end
15 end
```

Algorithm 1: A Gauss-Seidel-inspired acceleration scheme.

Given $(\psi, \beta) \in \Gamma_0(\mathscr{H}) \times \mathbb{R}_{>0}$, consider the proximal mapping

$$\mathrm{Prox}_{\beta\psi}(x) := \arg\min_{\omega \in \mathscr{H}} \psi(\omega) + \frac{1}{2\beta}\|x - \omega\|^2, \quad \forall x \in \mathscr{H}.$$

Clearly, when ψ equals the indicator function $\iota_{\mathscr{D}}$ of a closed convex set $\mathscr{D}$, then $\forall\beta \in \mathbb{R}_{>0}$, $\mathrm{Prox}_{\beta\psi}$ boils down to the (metric) projection $P_{\mathscr{D}}$ onto $\mathscr{D}$ [cf. (1)].

Given functions $(\varphi, \psi) \in \Gamma_0(\mathscr{H})^2$, where $\nabla\varphi$ is Lipschitz continuous with constant $L_\varphi \in \mathbb{R}_{>0}$, $\Phi := \varphi + \psi$, and the parameters $\Pi_r := (x_{r-1}, y_r, \eta_r) \in \mathscr{H} \times \mathscr{H} \times \mathbb{R}_{>0}$, define the core block of acceleration in Alg. 1 (cf. [26]):

$$\mathrm{Accel}(\varphi, \psi, \Pi_r)$$
$$= \begin{cases} \lambda_r & \in [\check{\epsilon}_\lambda, 1] \\ \eta_r & \in \Big[\check{\epsilon}_\eta, \min\big\{\hat{\epsilon}_\eta, \dfrac{1}{\hat{L}}\big\}\Big], \quad \eta_{r+1} \leq \eta_r \\ L_r & \in [L_\varphi, \hat{L}] \\ \beta_r & \in \left[\dfrac{1 - \sqrt{1 - \eta_r L_r \lambda_r}}{L_r}, \dfrac{1 + \sqrt{1 - \eta_r L_r \lambda_r}}{L_r}\right] \\ z_r & := \mathrm{Prox}_{\beta_r\psi}\big(y_r - \beta_r \nabla\varphi(y_r)\big) \\ x_r & \in \arg\min\limits_x \big\{\Phi(x) \mid x \in \{x_{r-1} + \lambda_r(z_r - x_{r-1}), x_{r-1}\}\big\} \\ t_{r+1} & := \dfrac{\sqrt{4t_r^2 + \lambda_1^2\lambda_{r+1}^2} + \lambda_1\lambda_{r+1}}{2}, \quad t_1 := \lambda_1 \\ y_{r+1} & := \dfrac{t_r}{t_{r+1}} y_r + \Big(1 - \dfrac{\lambda_1}{t_{r+1}}\Big) x_r - \dfrac{t_r - \lambda_1}{t_{r+1}} x_{r-1} \\ & \quad - \dfrac{t_r}{t_{r+1}} \dfrac{\eta_r \lambda_r}{\beta_r}(y_r - z_r). \end{cases} \tag{5}$$

Parameters $(\check{\epsilon}_\lambda, \check{\epsilon}_\eta, \hat{\epsilon}_\eta) \in \mathbb{R}_{>0}^3$ are user-defined, while $\hat{L} \in \mathbb{R}_{>0}$ stands for an upper bound on all Lipschitz constants used in this paper. The acceleration operator (5) is applied to Alg. 1 in a successive or Gauss-Seidel fashion. First, it is applied to $x^{(1)}$ for $R_1 \in \mathbb{N}_*$ cycles, and then to $x^{(2)}$ for $R_2 \in \mathbb{N}_*$ cycles.

A few comments regarding Alg. 1 are now in order. Vector $x^{(-i)}$ denotes all variables in $(x^{(1)}, x^{(2)})$ other than $x^{(i)}$, $i \in \{1,2\}$. Line 4 in Alg. 1 allows for exact computations of the minimizer, whenever closed-form solutions are available (cf. Sec. 4.1); or, whenever time and CPU resources are available for finding a highly accurate estimate of the minimizer. The computational complexity of Alg. 1 on Examples 1 and 2, including computations of Lipschitz constants and function evaluations, are in the order of $\mathscr{O}[(R_1 + R_2)Q]$ and $\mathscr{O}[(R_1 + R_2)(Q + P)P]$ per k, respectively.

Theorem 1. Assuming that all the employed r.vs. have finite first- and second-order moments, Alg. 1 enjoys the following properties.
1) $(\mathbb{E}\{F_k(x_k^{(1)}, x_k^{(2)})\})_{k\in\mathbb{N}}$ is non-increasing; $\forall k \in \mathbb{N}$,

$$\mathbb{E}\{F_k(x_{k+1}^{(1)}, x_{k+1}^{(2)}, \xi)\} \leq \mathbb{E}\{F_k(x_k^{(1)}, x_k^{(2)}, \xi)\}$$

where expectation is taken both w.r.t. ξ and $(x_\kappa^{(1)}, x_\kappa^{(2)})_{\kappa=0}^{k+1}$.
2) If F is bounded from below, and $(x_k^{(1)}, x_k^{(2)})_{k\in\mathbb{N}}$ is uniformly bounded over Ξ (the ensemble of trials), then the convergence rate of the iterates is quadratic, i.e., $\exists(k_0, F_*, C) \in \mathbb{N} \times \mathbb{R} \times \mathbb{R}_{>0}$ s.t.

$$\big|\mathbb{E}\{F_k(x_k^{(1)}, x_k^{(2)}, \xi)\} - F_*\big| \leq \frac{C}{(1+k)^2}, \quad \forall k \geq k_0.$$

3) If g_1, g_2 are coercive, meaning $\lim_{k\to\infty} |g_i(\omega_k)| = +\infty$ whenever $\lim_{k\to\infty} \|\omega_k\|_{\mathscr{H}_i} = +\infty$, $i \in \{1,2\}$, if F is bounded from below, and if $(x_k^{(1)}, x_k^{(2)})_{k\in\mathbb{N}}$ is uniformly bounded over Ξ, then $\forall k$ there exist subgradients [30] $(h_k^{(1)}, h_k^{(2)}) \in \partial_{x^{(1)}} F_k(z_k^{(1)}, x_{k-1}^{(2)}, \xi) \times \partial_{x^{(2)}} F_k(x_k^{(1)}, z_k^{(2)}, \xi)$ s.t.

$$\lim_{k\to\infty} \mathbb{E}\big\{\|h_k^{(1)}\|^2 + \|h_k^{(2)}\|^2\big\} = 0.$$

Coercivity is a quite general property; any ℓ_ν-norm, with $\nu \in [1, +\infty)$, as well as the indicator function $\iota_{\mathscr{D}}$ associated with any set $\mathscr{D}$ are coercive.
4) If g_1, g_2 are coercive, if F is bounded from below, and if $(x_k^{(1)}, x_k^{(2)})_{k\in\mathbb{N}}$ is uniformly bounded over Ξ, then the previous F_* satisfies

$$F_* \leq \limsup_{k\to\infty} \min_{x^{(i)}} \mathbb{E}\big\{F_k(x^{(i)}, \xi \mid x_k^{(-i)})\big\}, \quad i \in \{1,2\},$$

provided also that the set of all minimizers of $\mathbb{E}\big\{F_k(\cdot, \xi \mid x_k^{(-i)})\big\}$ is uniformly bounded over $k \in \mathbb{N}$.

Due to lack of space, proofs of the previous properties will be presented elsewhere. It is worth stressing however that convergence analysis in [24–26] pertains only to the sequence of objective function iterates, and not to the sequence of primal variables.

4. NUMERICAL TESTS

Although on smaller dimensions than those involved with big data applications, preliminary tests of the novel approach were performed using both synthetic and real data.

4.1. Synthetic data

To test Alg. 1 on Example 1, both regressors $(\boldsymbol{u}_{*k})_{k\in\mathbb{N}}$ and noise $(\boldsymbol{v}_k)_{k\in\mathbb{N}}$ were generated as zero-mean, i.i.d. mutually independent Gaussian processes with variances 1 and 10^{-2}, respectively. The nonzero entries were placed randomly (following a uniform distribution) in the sparse $\boldsymbol{s}_*$, with values generated independently also by a zero-mean, unit-variance Gaussian process. Alg. 1 was tested in two scenarios, namely a low-dimensional one corresponding to $(Q, \|\boldsymbol{s}_*\|_0) = (100, 10)$, and a relatively high-dimensional one with $(Q, \|\boldsymbol{s}_*\|_0) = (750, 75)$, tagged "low-dim" and "high-dim" in Figs. 1a and 1b, respectively. The regressors $(\boldsymbol{u}_k)_{k\in\mathbb{N}}$ were formed by adding to $(\boldsymbol{u}_{*k})_{k\in\mathbb{N}}$ i.i.d., zero-mean Gaussian noise with variance 10^{-2}. The following additional parameters were used in Alg. 1: $\lambda_0 := 10^{-6}$, $\lambda_k = 1, \forall k \neq 0$, $\eta_0 := 10^{-6}$, $\lambda_s := 10^{-6}$, and $\lambda_e := 10^2$ was selected to prevent large values of $\|\boldsymbol{e}\|$.

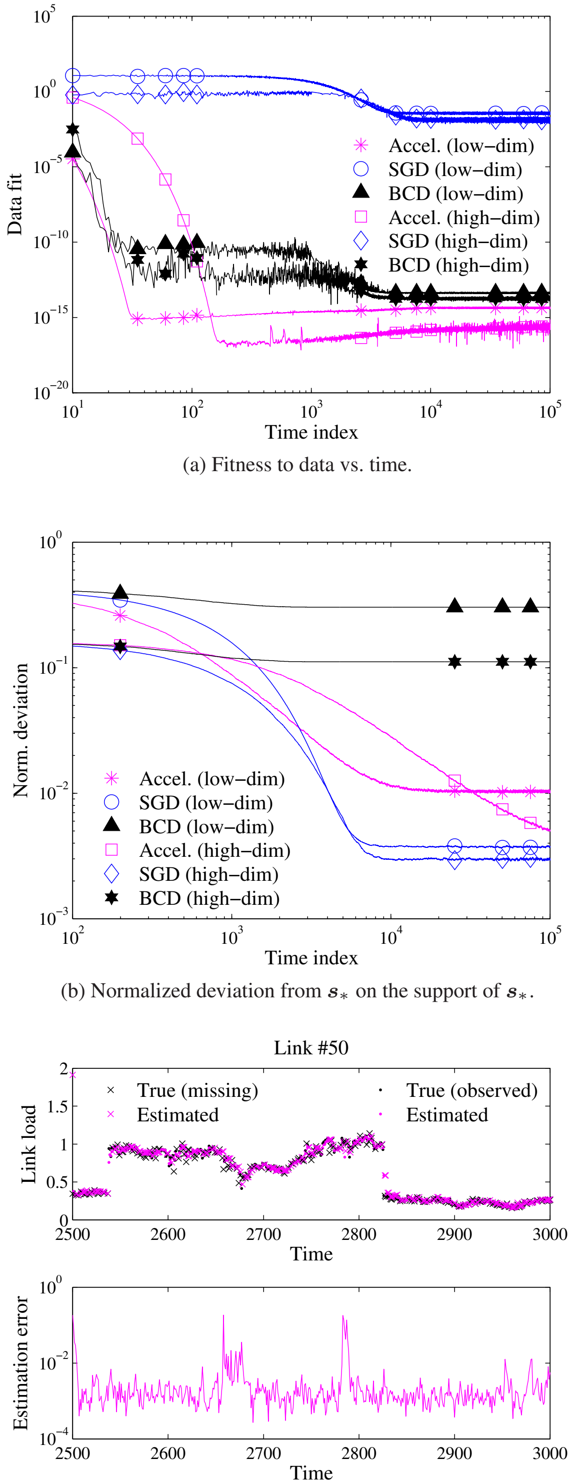
Alg. 1 was tested against the standard *subgradient descent (SGD)* method with constant step size 10^{-3}, and the *block coordinate descent (BCD)* scheme having the cost in (2) per BCD iteration of $\boldsymbol{s}$ maximally separated in scalar-valued blocks. Given the coordinates $\{s_j\}_{j\neq i}$ of $\boldsymbol{s}$, minimization of (2) w.r.t. s_i amounts to the following scalar-valued optimization task: $\min_{s_i\in\mathbb{R}} 0.5[y_k - \sum_{j\neq i}(u_{kj} + e_j)s_j - (u_{ki} + e_i)s_i]^2 + \lambda_s|s_i|$, which can be solved in closed form using the scalar-valued soft-thresholding operator. In all methods, the exact minimization step over $\boldsymbol{e}$ is straightforward: Given $\boldsymbol{s}$, the minimizer of (2) w.r.t. $\boldsymbol{e}$ is $\hat{\boldsymbol{e}} = (y_k - \boldsymbol{u}_k^\top \boldsymbol{s})(\boldsymbol{s}\boldsymbol{s}^\top + \lambda_e \boldsymbol{I}_Q)^{-1}\boldsymbol{s}$, where the required inverse is performed using the matrix inversion lemma: $(\boldsymbol{s}\boldsymbol{s}^\top + \lambda_e \boldsymbol{I}_Q)^{-1} = \lambda_e^{-1}[\boldsymbol{I}_Q - \boldsymbol{s}\boldsymbol{s}^\top(\lambda_e + \|\boldsymbol{s}\|^2)^{-1}]$. It is worth noticing here that $R_1 = 1$ for the inner loop of Alg. 1 in lines 8–11.

Figs. 1a and 1b illustrate the performance of all methods tested. Fig. 1a depicts the error fit $0.5[y_k - (\boldsymbol{u}_k + \hat{\boldsymbol{e}}_k)^\top \hat{\boldsymbol{s}}_k]^2$ across time, where $(\hat{\boldsymbol{s}}_k, \hat{\boldsymbol{e}}_k)$ denote estimates per slot k. Fig. 1b shows the normalized deviation $Q^{-1}\|\hat{\boldsymbol{s}}_k - \boldsymbol{s}_*\|$ versus k on the support of $\boldsymbol{s}_*$. The smooth curves of Figs. 1a and 1b were obtained after averaging 100 realizations. Although BCD exhibits fast convergence of the error function iterates, it does not identify correctly $\boldsymbol{s}_*$. The behavior of SGD confirms the known fact that (sub)gradient techniques are generally slow convergent. However, SGD shows the best performance as a system identification module. The proposed accelerated method outperforms both SGD and BCD in fitting the data accurately, and is only inferior to SGD in identifying $\boldsymbol{s}_*$.

4.2. Real data

Along the lines of Example 2, Alg. 1 was validated also on estimating and tracking network-wide link loads taken from the Internet2 measurement archive [33]. The Internet2 network consists of $P = 54$ links and 9 nodes. Using the network topology and routing information, network-wide link loads $(\boldsymbol{\chi}_k)_{k\in\mathbb{N}} \subset \mathbb{R}^P$ become available (in Gbps). Per time slot k, only $M = 30$ of the $\boldsymbol{\chi}_k$ components, chosen randomly via $\boldsymbol{M}_k \in \mathbb{R}^{M\times P}$, are observed in $\boldsymbol{y}_k \in \mathbb{R}^M$. The cardinality of the time-varying dictionaries is set constant to $Q = 80$, $\forall k$. To cope with pronounced temporal variations of the Internet2 link loads, the forgetting factor δ in Example 2 was set equal to 0.5. Initial values for both $(\boldsymbol{s}, \boldsymbol{D})$ were randomly drawn from the feasibility regions seen in Example 2. The parameters used in this realization of Alg. 1 were selected as follows: $\lambda_1 = 10^{-3}$, $\eta_0 = 10^{-6}$, $\lambda_s = 10^{-3}$, and $\lambda_L = 10^{-1}$.

Fig. 1c depicts estimated values of both observed (dots) and missing (crosses) link loads, for a randomly chosen link of the network. The normalized squared estimation error between the true $\boldsymbol{\chi}_k$ and the inferred $\hat{\boldsymbol{\chi}}_k$, namely $\|\boldsymbol{\chi}_k - \hat{\boldsymbol{\chi}}_k\|^2\|\boldsymbol{\chi}_k\|^{-2}$, is also plotted in Fig. 1c versus time k. Alg. 1 was compared with the state-of-the-art scheme in [5] that relies on the alternating direction method of multipliers (ADMM), see e.g., [34], to minimize a cost closely related to (3) w.r.t. $\boldsymbol{s}$, and uses BCD iterations requiring matrix inversions to optimize (3) w.r.t. $\boldsymbol{D}$. On the other hand, the number of inner loops in Alg. 1 w.r.t. $\boldsymbol{s}$ were set to $R_1 = 1$, while in order to retain the same overall estimation accuracy as [5], $R_2 = 10$ was used for the inner loops w.r.t. $\boldsymbol{D}$. It is worth noticing here that ADMM in [5] requires multiple iterations to achieve a prescribed estimation accuracy, and that no matrix inversion was incorporated in the realization of Alg. 1. Both Alg. 1 and [5] perform comparably in the simulated tests, but only those of Alg. 1 are shown here for clarity.



(a) Fitness to data vs. time.

(b) Normalized deviation from $\boldsymbol{s}_*$ on the support of $\boldsymbol{s}_*$.

(c) Link load tracking and imputation, as well as normalized squared estimation error.

Fig. 1. Numerical results for the synthetic [(a), (b)] and real data [(c)] of Secs. 4.1 and 4.2, respectively.

5. REFERENCES

[1] I. Tošić and P. Frossard, "Dictionary learning," *IEEE Signal Process. Magaz.*, vol. 28, no. 2, pp. 27–38, Mar. 2011.

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

[3] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. Pattern Analysis Machine Intelligence*, vol. 34, no. 4, pp. 791–804, Apr. 2012.

[4] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, pp. 1289–1306, 2006.

[5] P. A. Forero, K. Rajawat, and G. B. Giannakis, "Semi-supervised dictionary learning for network-wide link load prediction," in *Proc. CIP*, Baiona: Spain, May 2012.

[6] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss-Seidel method under convex constraints," *Operations Reseach Letters*, vol. 26, no. 3, pp. 127–136, 2000.

[7] P. Tseng, "Convergence of block coordinate decent method for nondifferentiable minimization," *J. of Optimization Theory and Applications*, vol. 109, pp. 475–494, June 2001.

[8] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Math. Program., Ser. B*, vol. 117, pp. 387–423, 2009.

[9] S. J. Wright, "Accelerated block-coordinate relaxation for regularized optimization," *SIAM J. Optim.*, vol. 22, no. 1, pp. 159–186, 2012.

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

[11] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imaging*, vol. 6, no. 3, pp. 1758–1789, 2013.

[12] P. Richtárik and M. Takáč, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Math. Program., Ser. A*, pp. 1–38, Dec. 2012.

[13] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Trans. Signal Process.*, vol. 57, no. 7, pp. 2479–2493, July 2009.

[14] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM J. Optimization*, vol. 23, no. 2, pp. 1126–1153, 2013.

[15] S. Shalev-Schwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *J. Machine Learning Research*, vol. 14, pp. 567–599, 2013.

[16] Q. Tao, K. Kong, D. Chu, and G. Wu, "Stochastic coordinate descent methods for regularized smooth and nonsmooth losses," in *Lecture Notes in Computer Science*, ser. Machine Learning and Knowledge Discovery in Databases, P. A. Flach, T. de Bie, and N. Cristianini, Eds., vol. 7523. Springer, 2012, pp. 537–552.

[17] M. Mardani, G. Mateos, and G. B. Giannakis, "Decentralized sparsity-regularized rank minimization: Algorithms and applications," *IEEE Trans. Signal Process.*, vol. 61, no. 11, pp. 5374–5388, Nov. 2013.

[18] B. Recht and C. Re, "Parallel stochastic gradient algorithms for large-scale matrix completion," 2011, submitted for publication. [Online]. Available: http://pages.cs.wisc.edu/ brecht/papers/11.Rec.Re.IPGM.pdf

[19] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[20] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *J. Machine Learning Research*, vol. 10, pp. 2899–2934, Dec. 2009.

[21] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[22] A. Agarwal, S. Negahban, and M. J. Wainwright, "Stochastic optimization and sparse statistical recovery: An optimal algorithm for high dimensions," 2012, submitted for publication. [Online]. Available: arXiv:1207.4421v1

[23] Y. Nesterov, "A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$," *Dokl. Akad. Nauk SSSR*, vol. 269, pp. 543–547, 1983, in Russian.

[24] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[25] ——, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Processing*, vol. 18, pp. 2419–2439, 2009.

[26] M. Yamagishi and I. Yamada, "Over-relaxation of the fast iterative shrinkage-thresholding algorithm with variable stepsize," *Inverse Problems*, vol. 27, no. 10, 2011.

[27] M. Yamagishi, M. Yukawa, and I. Yamada, "Acceleration of adaptive proximal forward-backward splitting method and its application to sparse system identification," in *Proc. IEEE ICASSP*, Prague: Czech Republic, May 22–27 2011, pp. 4296–4299.

[28] C. Hu, J. T. Kwok, and W. Pan, "Accelerated gradient methods for stochastic optimization and online learning," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, 2009.

[29] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, 2003.

[30] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York: Springer, 2011.

[31] J. Neveu, *Mathematical Foundations of the Calculus of Probability*. San Francisco: Holden-Day, 1965.

[32] S. Van Huffel and P. Lemmerling, *Total Least Squares and Errors-in-Variables Modeling*. Springer, 2002.

[33] [Online]. Available: http://www.internet2.edu/observatory/

[34] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.