QUICKEST CONVERGENCE OF ONLINE ALGORITHMS VIA DATA SELECTION

Daniel Romero, Dimitris Berberidis, and Georgios B. Giannakis

Department of Electrical and Computer Engineering and Digital Technology Center
University of Minnesota, USA, {dromero,bermp001,georgios}@umn.edu

ABSTRACT

Big data applications demand efficient solvers capable of providing accurate solutions to large-scale problems at affordable computational costs. Processing data sequentially, online algorithms offer attractive means to deal with massive data sets. However, they may incur prohibitive complexity in high-dimensional scenarios if the entire data set is processed. It is therefore necessary to confine computations to an informative subset. While existing approaches have focused on selecting a prescribed fraction of the available data vectors, the present paper capitalizes on this degree of freedom to accelerate the convergence of a generic class of online algorithms in terms of processing time/computational resources by balancing the required burden with a metric of how informative each datum is. The proposed method is illustrated in a linear regression setting, and simulations corroborate the superior convergence rate of the recursive least-squares algorithm when the novel data selection is effected.

***Index Terms*—** Big data, online learning, adaptive data selection, stochastic approximation.

1. INTRODUCTION

Big data problems are characterized by huge volumes of data that exceed the available computational capabilities [1]. Time-sensitive and cost-constrained applications impose stringent requirements that cannot be met unless optimization tasks are confined to a judiciously selected subset of the data. Randomized linear algebra [2] and experiment design [3, 4] techniques were proposed to minimize the resulting increase in estimation error, but they presume availability of the entire data set before the selection can commence. This observation has favored the adoption of *online* approaches, where each datum is sequentially visited a single time [5], a feature that makes them particularly attractive for processing big data streams and even mining massive data sets.

Online selection rules decide on-the-fly whether each datum must be processed or ignored based on some prior estimate of how informative it is. For example, in [6, 7] observations with low innovations are discarded to reduce computational complexity of estimating linear regression coefficients. Similar principles are exploited in [6–10], where non-informative observations are censored to minimize the communication overhead in bandwidth-constrained sensor networks. The related randomized Kaczmarz algorithm [11–13] accomplishes data selection according to the norm of the prediction vectors, but cannot be implemented in a purely online fashion since the selection probabilities depend on the norm of all other vectors.

Many problems in big data applications deal with computing solutions that attain the highest accuracy possible given budgeted computational resources, typically materialized as upper bounds on the processing time or the number of arithmetic operations. Thus, we informally say in this context that a *fast yet approximate* solution is preferable over an *accurate yet slow* one. Unfortunately, existing online approaches do not obey this principle since they hinge on constraining the number of selected data vectors rather than saving the computational load. Moreover, those constraints, typically given in terms of the *selection* or *compression ratio*, must be adjusted by the user to meet application-specific restrictions.

The distinctive viewpoint in this paper is that the decision on selecting a data vector must be based on balancing how informative it is relative to the computational cost required to process it. An inherent trade-off underlies this decision: if an exceedingly large portion of the data is selected, non-informative data vectors will be processed, thus wasting computational resources. Conversely, if the selection is so strict that only a small fraction of the data survive, the overhead of the selection algorithm may not pay off for the savings due to data reduction.

Our main contribution here is an online data selection module intended to accelerate the convergence of a broad class of online learning/optimization algorithms. As opposed to complexity-agnostic methods, the selection/compression ratio is no longer set by the user — it is adjusted on-the-fly to speed up the convergence in terms of time or computational cost. The stochastic approximation perspective adopted allows us to devise a fully adaptive and data-driven method that needs no probabilistic knowledge about the data. We illustrate its operation using the *recursive least squares* (RLS) algorithm and demonstrate its effectiveness via simulations.

Notation: Throughout the paper, $\mathcal{I}[\cdot]$ denotes the indicator function, while $u(t) := \mathcal{I}[t \geq 0]$ represents the unit step function.

2. ONLINE DATA SELECTION ALGORITHM

This section introduces the notation and describes how a generic online algorithm can be modified to accommodate data selection.

2.1. Modeling the Online Algorithm

Suppose that a certain online algorithm is employed to find the vector $\boldsymbol{w} \in \mathbb{R}^p$ minimizing the objective $J(\boldsymbol{w})$. At the n-th time slot, the algorithm processes the n-th data vector $\boldsymbol{d}_n$ to refine its estimate. The form of $\boldsymbol{d}_n$ is determined by the specific setting where the optimization problem arises. For instance, in unsupervised learning, this datum is typically a vector in Euclidean space; in supervised learning, it may correspond to an input-output pair $\boldsymbol{d}_n = (\boldsymbol{x}_n, y_n)$ used to train a regression algorithm or a classifier. Motivated by big data and streaming applications, we will consider that the data set comprises an unlimited collection of such vectors.

This work has been supported in part by NSF grants 1202135 and 1500713.

Algorithm 1: Algorithm with Data Selection

Initialize $\hat{\boldsymbol{w}}_0$, $\mathcal{S}_0$ and set $k = 0$
for $n = 1, 2, \ldots$ **do**
  if $a_n = 1$ **then**
    $k \leftarrow k + 1$
    $\hat{\boldsymbol{w}}_k = \mathcal{F}_{\boldsymbol{w}}(\boldsymbol{d}_n, \hat{\boldsymbol{w}}_{k-1}, \mathcal{S}_{k-1})$.
    $\mathcal{S}_k = \mathcal{F}_{\mathcal{S}}(\boldsymbol{d}_n, \hat{\boldsymbol{w}}_{k-1}, \mathcal{S}_{k-1})$.
  end if
end for

The algorithm is generally represented by the following updates:

$$\begin{aligned} \hat{\boldsymbol{w}}_n &= \mathcal{F}_{\boldsymbol{w}}(\boldsymbol{d}_n, \hat{\boldsymbol{w}}_{n-1}, \mathcal{S}_{n-1}) \\ \mathcal{S}_n &= \mathcal{F}_{\mathcal{S}}(\boldsymbol{d}_n, \hat{\boldsymbol{w}}_{n-1}, \mathcal{S}_{n-1}) \end{aligned} \tag{1}$$

where $\mathcal{S}_n$ represents a collection of state variables where the algorithm stores information about previous updates. For example, the well-known RLS algorithm stores the inverse of the sample scatter matrix, that is, $\mathcal{S}_n = \{\boldsymbol{C}_n\}$, where $\boldsymbol{C}_n \approx (\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^T)^{-1}$. Algorithms without memory, such as the least mean-squares (LMS) algorithm or the perceptron, need no state information ($\mathcal{S}_n = \emptyset$).

We will be primarily concerned with the decreasing rate of the sequence of objective values $\{J(\hat{\boldsymbol{w}}_n)\}$ that results from iterated application of (1). Processing $\boldsymbol{d}_n$ entails an *improvement* $\Delta_{n|n-1} := J(\hat{\boldsymbol{w}}_{n-1}) - J(\hat{\boldsymbol{w}}_n)$, which can be regarded as a measure of how informative $\boldsymbol{d}_n$ is. Telescoping this improvement yields $J(\hat{\boldsymbol{w}}_n) = J(\hat{\boldsymbol{w}}_0) - \sum_{i=1}^n \Delta_{i|i-1}$. The time evolution of the objective value can be characterized by the function $J(t) = J(\hat{\boldsymbol{w}}_0) - \sum_{i=1}^\infty \Delta_{i|i-1} u(t - iT_u)$, where T_u is the time required to perform the update (1), and can be measured either in time units or in the number of arithmetic operations.

2.2. Algorithm with Data Selection

The generic algorithm described in the previous section can be modified to execute (1) only for a subset of the data vectors, while discarding the rest. In particular, suppose that a selection sequence $a_1, a_2, \ldots$ is given, where $a_n = 1$ if $\boldsymbol{d}_n$ is to be processed and $a_n = 0$ otherwise. In Sec. 3 we will address the design of such a sequence. The modified algorithm is listed as Algorithm 1.

Let $\bar{a}(n) := \sum_{i=1}^n a_i$ denote the number of updates performed up to (and including) the n-th time slot. If $\boldsymbol{d}_n$ is selected, the most recent estimate $\hat{\boldsymbol{w}}_{\bar{a}(n-1)}$ is updated to $\hat{\boldsymbol{w}}_{\bar{a}(n)} = \mathcal{F}_{\boldsymbol{w}}(\boldsymbol{d}_n, \hat{\boldsymbol{w}}_{\bar{a}(n-1)}, \mathcal{S}_{\bar{a}(n-1)})$, and the resulting improvement becomes $\Delta_{n|\bar{a}(n-1)}$, where $\Delta_{n|k} := J(\hat{\boldsymbol{w}}_k) - J(\mathcal{F}_{\boldsymbol{w}}(\boldsymbol{d}_n, \hat{\boldsymbol{w}}_k, \mathcal{S}_k))$ is the improvement resulting from processing $\boldsymbol{d}_n$ when the most recent estimate is $\hat{\boldsymbol{w}}_k$. Conversely, if $\boldsymbol{d}_n$ is not selected, the estimate is not updated and the improvement becomes zero. The time evolution of the objective is therefore given by

$$J(t) = J(\hat{\boldsymbol{w}}_0) - \sum_{n=1}^\infty a_n \Delta_{n|\bar{a}(n-1)} u(t - nT_c - \bar{a}(n)T_u) \tag{2}$$

where T_c is the time required per time slot to *check* $\boldsymbol{d}_n$ and decide whether to select it or not. In the scheme proposed in Sec. 3, T_c will be dominated by the time required to estimate $\Delta_{n|\bar{a}(n-1)}$.

3. DATA SELECTION VIA THRESHOLDING

In this section, we start by discussing different criteria for designing data selection rules. We next establish that thresholding selection rules are approximately optimal and lead to a novel online implementation via stochastic approximation.

3.1. Design Criteria

In many real-time and big data applications, a fast yet approximate estimate is preferable over a slow but accurate one. Our plan is to formalize this principle in the form of a criterion that enables us to compare different selection sequences. To do so, it is natural to look at the evolution of the objective represented by (2), which depends on the sequence $\{a_n\}$, and characterizes how fast it descends. Pertinent criteria include the following:

- **Accuracy criterion:** Given $\bar{J}$, we are interested in selection sequences with minimum
$$T_A(\{a_n\}) := \arg\min_t \{t : J(t, \{a_n\}) \leq \bar{J}\}.$$
- **Deadline criterion:** Given T_{DL}, we are interested in those sequences with minimum $J_{DL}(\{a_n\}) := J(T_{\text{DL}}, \{a_n\})$.
- **Quickest descent:** Given an offset t_0 and a window length $\bar{T}$, we seek sequences minimizing the *average* slope of $J(t)$:
$$s_{QD}(\{a_n\}) = \frac{J(t_0 + \bar{T}, \{a_n\}) - J(t_0, \{a_n\})}{\bar{T}}. \tag{3}$$

Our focus in the rest of the paper will be the last criterion. Informally, this criterion is especially well motivated when the optimal rule does not depend on the parameters t_0 and $\bar{T}$, since in those cases the average descent rate is maximized everywhere.

Let $t_0 = 0$ and select $\bar{T} = \bar{N}T_c + \bar{a}(\bar{N})T_u$, which is the time required to process $\bar{N}$ data vectors. Then, (3) becomes (c.f. (2))

$$s_{QD}(\{a_n\}) = -\frac{\sum_{n=1}^{\bar{N}} a_n \Delta_{n|\bar{a}(n-1)}}{\bar{N}T_c + \bar{a}(\bar{N})T_u}. \tag{4}$$

The rest of this section deals with finding $\{a_n\}$ to approximately minimize (4) given the data. We will start by establishing the form of an approximately optimal selection rule and then explain how to practically implement such a strategy. As we will see, the final algorithm will not be sensitive to the choice of t_0 and $\bar{T}$.

3.2. Thresholding is Approximately Optimal

When attempting to minimize (4), one finds that the $\{\Delta_{n|\bar{a}(n-1)}\}$ depend on the $\{a_n\}$, whereas the optimal $\{a_n\}$ depend on the $\{\Delta_{n|\bar{a}(n-1)}\}$. Thus, unless one knows the $\{\Delta_{n|\bar{a}(n-1)}\}$ for all possible $\{a_n\}$, finding optimal sequences is intractable. To circumvent this issue, we introduce an approximation as described next.

Assume that convergence of the estimates is slow, which implies that $\hat{\boldsymbol{w}}_k$ and $\mathcal{S}_k$ experience slow changes over k. Due to data selection, this effect is even more pronounced in $\hat{\boldsymbol{w}}_{\bar{a}(n-1)}$ and $\mathcal{S}_{\bar{a}(n-1)}$ when seen as sequences of n. One may therefore consider that the improvement $\Delta_{n|k}$ is approximately constant in k for a certain range of values of n, which means that there exists a function f such that $\Delta_n := f(\boldsymbol{d}_n) \approx \Delta_{n|\bar{a}(n-1)}$ for n in a certain window. As seen in Sec. 3.4, the effects of this approximation will be compensated to some extent when we replace $\Delta_{n|k}$ with their estimates.

Given the $\{\Delta_n\}$, an approximately optimal selection sequence may be found as the solution of

$$\max_{a_1, \ldots, a_{\bar{N}}} \frac{\sum_{n=1}^{\bar{N}} a_n \Delta_n}{T_c \bar{N} + T_u \bar{a}(n)}. \tag{5}$$

Simple inspection of (5) reveals that the optimal sequence satisfies $a_n = 0$ for all n satisfying $\Delta_n < 0$. For this reason, we will focus on the case where $\Delta_n \geq 0$ for all n (the extension to accommodate negative values is straightforward).

Proposition 1. *The solution of* (5) *is* $a_n = \mathcal{I}[\Delta_n \geq \gamma]$, *where*

$$\gamma := \arg\max_{\gamma} \frac{\sum_{n=1}^{\bar{N}} \mathcal{I}[\Delta_n \geq \gamma]\Delta_n}{T_c\bar{N} + T_u \sum_{n=1}^{\bar{N}} \mathcal{I}[\Delta_n \geq \gamma]}. \tag{6}$$

Proof. Let Π denote a permutation of $\{1, \ldots, \bar{N}\}$ such that $\Delta_{\Pi(1)} \geq \Delta_{\Pi(2)} \geq \ldots \geq \Delta_{\Pi(\bar{N})}$. Introducing the auxiliary variable $\bar{n}$ in (5) yields the equivalent program

$$\max_{\bar{n}, a_1, \ldots, a_{\bar{N}}} \quad \frac{\sum_{n=1}^{\bar{N}} a_{\Pi(n)}\Delta_{\Pi(n)}}{T_c\bar{N} + T_u\bar{n}} \quad \text{s.t.} \quad \sum_{n=1}^{\bar{N}} a_n = \bar{n}. \tag{7}$$

For fixed $\bar{n}$, the solution clearly satisfies $a_{\Pi(1)} = \ldots = a_{\Pi(\bar{n})} = 1$ and $a_{\Pi(\bar{n}+1)} = \ldots = a_{\Pi(\bar{N})} = 0$. Equivalently $a_n = 1$ if and only if $\Delta_n \geq \Delta_{\Pi(\bar{n})} := \gamma$. It remains only to maximize with respect to (w.r.t.) $\bar{n}$, but this is equivalent to maximizing w.r.t. γ. □

From Proposition 1, we deduce that an approximately optimal selection rule in the sense of (3) is that selecting $\boldsymbol{d}_n$ whenever $\Delta_n \geq \gamma$, for γ a suitably selected threshold. The problem of optimizing over the space of binary sequences therefore reduces to the problem of optimizing w.r.t. a scalar variable.

3.3. Threshold Selection

Finding γ via (6) presents two difficulties: first, this expression depends on Δ_n for all $n = 1, \ldots, \bar{N}$, which includes *future* values since the threshold is required for data selection from the beginning. Second, the $\{\Delta_n\}$ are not directly available in view of the data and have to be somehow estimated. In this section, we address the first concern by proposing an online rule for threshold computation, postponing estimation of $\{\Delta_n\}$ to Sec. 3.4 – here we assume that the Δ_n are known.

Dividing numerator and denominator by $\bar{N}$, (6) becomes

$$\gamma := \arg\max_{\gamma} \frac{\frac{1}{\bar{N}}\sum_{n=1}^{\bar{N}} \mathcal{I}[\Delta_n \geq \gamma]\Delta_n}{T_c + \frac{1}{\bar{N}}T_u \sum_{n=1}^{\bar{N}} \mathcal{I}[\Delta_n \geq \gamma]}. \tag{8}$$

If the $\{\boldsymbol{d}_n\}$ are i.i.d. random variables, the slow convergence assumption (c.f. Sec. 3.2) implies that the $\Delta_n = f(\boldsymbol{d}_n)$ are also approximately i.i.d. Invoking the law of large numbers, we can approximate (8) as

$$\gamma \approx \arg\max_{\gamma} \frac{\mathbb{E}\{\mathcal{I}[\Delta_n \geq \gamma]\Delta_n\}}{T_c + T_u\mathbb{P}\{\Delta_n \geq \gamma\}}. \tag{9}$$

Suppose that the $\{\Delta_n\}$ have a continuous distribution. Then, setting the derivative of the quotient in (9) w.r.t. γ equal to zero yields

$$\mathbb{E}\left\{\max(\Delta_n - \gamma, 0) - \gamma\frac{T_c}{T_u}\right\} = 0.$$

A threshold γ satisfying this necessary condition can be found using the Robbins-Monro iteration [14], which iteratively sets

$$\gamma_{n+1} = \gamma_n + \mu_n\left[\max(\Delta_n - \gamma_n, 0) - \gamma_n\frac{T_c}{T_u}\right] \tag{10}$$

where $\mu_n > 0$ represents the step size.

Algorithm 2: Data-selection-based Algorithm

```
Initialize ŵ_0, S_0, and γ_1.
Set k = 0
for n = 1, 2, ... do
    Compute Δ̂_{n|k}
    if Δ̂_{n|k} ≥ γ_n then
        k ← k + 1
        ŵ_k = F_w(d_n, ŵ_{k-1}, S_{k-1})
        S_k = F_S(d_n, ŵ_{k-1}, S_{k-1})
        γ_{n+1} = ((T_u − μ_n(T_c + T_u))/T_u) γ_n + μ_n Δ̂_{n|k}
    else
        γ_{n+1} = ((T_u − μ_n T_c)/T_u) γ_n
    end if
end for
```

3.4. Quickest Descent Algorithm

Evaluating the right-hand side of (10) requires knowledge of the $\{\Delta_n\}$, which is not directly available from the data. We presume that there is a means to somehow estimate $\Delta_n \approx \Delta_{n|\bar{a}(n-1)}$ from the data, even roughly, in a time $T_c < T_u$. For example, in least-squares (LS) regression, one may estimate the improvement using the squared residual, which is nothing but the squared difference between y_n and its predicted value using the most recent estimate of $\boldsymbol{w}$ (see Sec. 4).

With $\hat{\Delta}_{n|\bar{a}(n-1)}$ denoting the estimate of $\Delta_{n|\bar{a}(n-1)}$, we summarize the algorithm modified with data selection as Algorithm 2. Note that at no point does this algorithm depend on the window parameters $\bar{T}$ or $\bar{N}$, which suggests that the resulting selection sequence minimizes the slope of $J(t)$ at any averaging window whose length is sufficiently high so that the approximation in (9) holds.

Remark 1. It can be readily seen that if the $\{\hat{\Delta}_{n|\bar{a}(n-1)}\}$ are scaled by a positive constant $\alpha > 0$, the thresholds resulting from iterative application of (10) will be — neglecting transient effects due to the initialization — scaled similarly and, therefore, will generate the same selection sequence. This means that positive scaling factors need not be accounted for when designing the estimate $\hat{\Delta}_{n|\bar{a}(n-1)}$.

Remark 2. For simplicity, we assumed that $J(\hat{\boldsymbol{w}})$ is the objective function that the algorithm described by (1) aims to minimize. However, our derivation still applies if $J(\hat{\boldsymbol{w}})$ is any other function of the estimates whose decreasing rate is to be accelerated. For instance, in Sec. 4 we will set $J(\hat{\boldsymbol{w}}) := ||\boldsymbol{w}_o - \hat{\boldsymbol{w}}||^2$, where $\boldsymbol{w}_o$ denotes the ground truth, in place of the LS cost function.

Remark 3 (Fixed selection rate). For comparison purposes, we next provide an alternative threshold selection rule that, instead of minimizing the quickest descent criterion, it simply seeks to achieve a target selection probability π. The goal is therefore to find γ such that $\mathbb{P}\{\Delta_n \geq \gamma\} = \pi$ or, alternatively, $\mathbb{E}\{\mathcal{I}[\Delta_n \geq \gamma] - \pi\} = 0$. Again, this equation can be solved using a Robbins-Monro iteration, which in this case reads as

$$\gamma_{n+1} = \gamma_n + \mu_n(\mathcal{I}[\hat{\Delta}_n \geq \gamma_n] - \pi)$$

or, equivalently,

$$\gamma_{n+1} = \begin{cases} \gamma_n + \mu_n(1-\pi) & \text{if } \boldsymbol{d}_n \text{ is selected} \\ \gamma_n - \mu_n\pi & \text{otherwise.} \end{cases}$$

4. APPLICATION EXAMPLE: RLS

In this section, we illustrate how the RLS algorithm, which is a second-order online method to solve LS regression problems, can be modified to accommodate the proposed data selection rule. Specifically, if the data are generated according to $y_n = \boldsymbol{x}_n^T \boldsymbol{w}_o + v_n$, where v_n is i.i.d. zero-mean noise with variance σ^2, RLS iteratively computes the LS estimate $\hat{\boldsymbol{w}}_n = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2$.

When data selection is effected, the updating equations in Algorithm 2 can be written as (note that $\mathcal{S}_k = \{\boldsymbol{C}_k\}$ in RLS)

$$\begin{aligned} \boldsymbol{C}_{k+1} &= \boldsymbol{C}_k - \frac{\boldsymbol{C}_k \boldsymbol{x}_n \boldsymbol{x}_n^T \boldsymbol{C}_k}{1 + \boldsymbol{x}_n^T \boldsymbol{C}_k \boldsymbol{x}_n} \\ \hat{\boldsymbol{w}}_{k+1} &= \hat{\boldsymbol{w}}_k + e_{n|k} \boldsymbol{C}_n \boldsymbol{x}_n \end{aligned} \tag{11}$$

where $e_{n|k} := y_n - \boldsymbol{x}_n^T \hat{\boldsymbol{w}}_k$. Note that $\boldsymbol{C}_{\bar{a}(n)} = \left(\sum_{i=1}^{n} a_i \boldsymbol{x}_i \boldsymbol{x}_i^T\right)^{-1}$ after neglecting transient effects due to initialization.

Since we are primarily interested in accelerating the convergence of $\hat{\boldsymbol{w}}_n$ to the ground truth $\boldsymbol{w}_o$, we set $J(\hat{\boldsymbol{w}}) := ||\boldsymbol{w}_o - \hat{\boldsymbol{w}}||^2$ (see Remark 2), which converges to zero due to the consistency of the LS estimate [15]. Simple algebra reveals that

$$\begin{aligned} \Delta_{n|k} &= J(\hat{\boldsymbol{w}}_k) - J(\hat{\boldsymbol{w}}_k + e_{n|k} \boldsymbol{C}_{k+1} \boldsymbol{x}_n) \\ &= e_{n|k} \boldsymbol{x}_n^T \boldsymbol{C}_{k+1} \left[2(\boldsymbol{w}_o - \hat{\boldsymbol{w}}_k) - e_{n|k} \boldsymbol{C}_{k+1} \boldsymbol{x}_n\right]. \end{aligned}$$

Neglecting noise effects, that is $y_n \approx \boldsymbol{x}_n^T \boldsymbol{w}_o$, enables us to approximate $e_{n|k} \approx \boldsymbol{x}_n^T (\boldsymbol{w}_o - \hat{\boldsymbol{w}}_k)$, which in turn produces

$$\Delta_{n|k} \approx e_{n|k} \boldsymbol{x}_n^T \boldsymbol{C}_{k+1} (2\boldsymbol{I}_p - \boldsymbol{C}_{k+1} \boldsymbol{x}_n \boldsymbol{x}_n^T)(\boldsymbol{w}_o - \hat{\boldsymbol{w}}_k). \tag{12}$$

Defining $\boldsymbol{\Sigma} := \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^T$ and assuming that the series converges, allows us to write $\boldsymbol{C}_k \approx \left(\sum_{i=1}^{n} a_i \boldsymbol{x}_i \boldsymbol{x}_i^T\right)^{-1} \approx (k\boldsymbol{\Sigma})^{-1}$. Together with (12), this shows that

$$\Delta_{n|k} \approx \frac{2}{k} e_{n|k} \boldsymbol{x}_n^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{w}_o - \hat{\boldsymbol{w}}_k)$$

and in the extreme case of $\boldsymbol{\Sigma} \approx \boldsymbol{I}_p$, one finds $\Delta_{n|k} \approx (2/k) e_{n|k}^2$. Again, due to the slow variation assumption from Sec. 3.2, the scaling factor $2/k$ experiences slow changes and can be neglected (see Remark 1). Therefore, one may simply consider the estimator $\hat{\Delta}_{n|\bar{a}(n-1)} = e_{n|\bar{a}(n-1)}^2$. Evaluating $\hat{\Delta}_{n|\bar{a}(n-1)}$ incurs a time $T_c = 2p + 1$ (set one time unit equal to the time required to perform an addition/multiplication), which is smaller than the time $T_u = 5p^2 + 3p + 1$ required to compute (11).

In order to appreciate the acceleration effected by the proposed sample selection algorithm, we illustrate its operation via a simple numerical example. The ground truth vector $\boldsymbol{w}_o$ was drawn from a Gaussian distribution $\mathcal{N}(\boldsymbol{0}_p, \boldsymbol{I}_p)$, where $p = 200$. The data vectors were generated independently as $\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{0}_p, \boldsymbol{B}\boldsymbol{B}^T)$, where the entries of $\boldsymbol{B} \in \mathbb{R}^{p\times p}$ were independently generated using a standardized Gaussian distribution. The noise variance was set to $\sigma^2 = 25$ and the step size μ_n was set equal to 0.2 for the quickest descent algorithm (Algorithm 2) and 1 for the fixed selection-rate algorithm from Remark 3.

Fig. 1 compares the time evolution of $J(t)/J(0)$ for the proposed algorithm along with the implementation with no data selection and the fixed selection-rate algorithm with $\pi = 0.4$ and $\pi = 0.1$. It is observed that data selection accelerates convergence and that the quickest descent algorithm leads to an approximate estimate of $\boldsymbol{w}_o$ much faster than the algorithm with no data selection. The downside of the proposed algorithm is that the error in the steady state is increased due to the fact that noise effects are no longer negligible and the approximation in (12) becomes too coarse. Therefore, after convergence it is recommended to switch to a fixed selection-rate mode. It is also insightful to look at the evolution of the threshold used by the quickest descent algorithm. Fig. 2 represents γ_n versus the iteration index along with the values of $\{e_{n|\bar{a}(n-1)}^2\}$. It is observed that the algorithm is capable of effectively tracking the changes in the distribution of the squared residual.

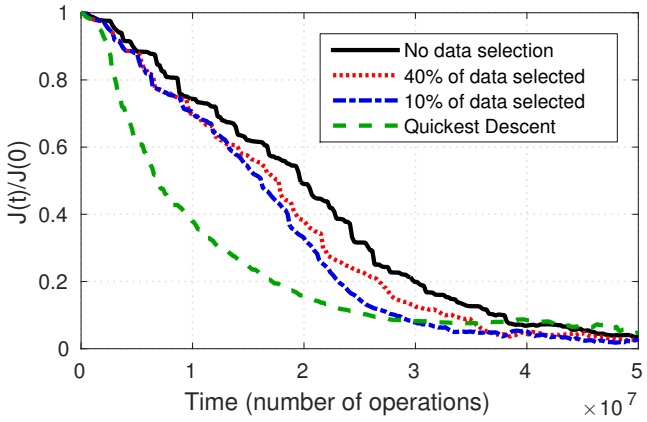

Fig. 1: The proposed algorithm (dashed line) is capable of providing an approximate estimate much earlier than its fixed-selection rate counterparts.

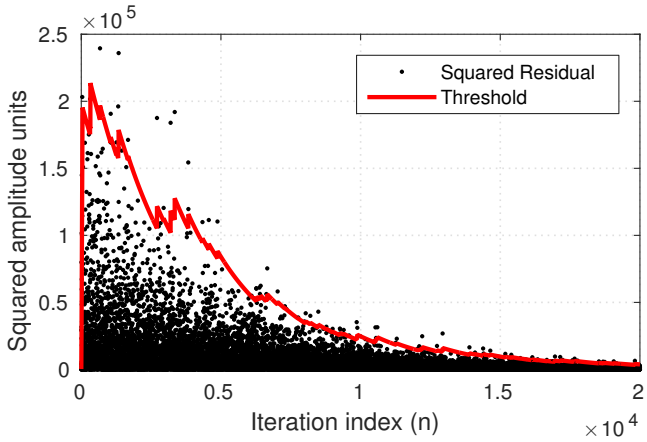

Fig. 2: The Quickest Descent algorithm adaptively adjusts the threshold, thus tracking changes in the distribution of $\hat{\Delta}_{n|\bar{a}(n-1)}$.

5. CONCLUSIONS

We have proposed a simple, time-adaptive, and data-driven online selection algorithm that adjusts the selection ratio on-the-fly to accelerate convergence of an online algorithm, as quantified by the quickest descent criterion. We applied this technique to RLS, but a number of other algorithms are yet to be considered in our future research, including LMS, logistic regression, and subspace tracking, to name a few.

6. REFERENCES

[1] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge," *IEEE Sig. Process. Mag.*, vol. 31, no. 5, pp. 18–31, Sept 2014.

[2] M. Mahoney, "Randomized algorithms for matrices and data," *Found. Trends. Mach. Learn.*, vol. 3, no. 2, pp. 123–224, 2011.

[3] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Trans. Sig. Process.*, vol. 57, no. 2, pp. 451–462, 2009.

[4] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in *Proc. 49th Conf. Decision Control*, Atlanta, Dec. 2010, pp. 2572–2577.

[5] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[6] D. Berberidis, V. Kekatos, and G. B. Giannakis, "Online censoring for large-scale regressions with application to streaming big data," Preprint at `http://arXiv.org/abs/1507.07536`, 2015.

[7] D. Berberidis, V. Kekatos, G. Wang, and G. B. Giannakis, "Adaptive censoring for large-scale regressions," in *Proc. IEEE Int. Conf. Acoust., Speech, Sig. Process.*, Brisbane, Australia, April 2015, pp. 5475–5479.

[8] Y. Zheng, R. Niu, and P. K. Varshney, "Sequential Bayesian estimation with censored data for multi-sensor systems," *IEEE Trans. Sig. Process.*, vol. 62, no. 10, pp. 2626–2641, Oct. 2014.

[9] G. Battistelli, A. Benavoli, and L. Chisci, "Data-driven strategies for selective data transmission in sensor networks," in *Proc. 51st Conf. Decision Control*, Grand Wailea, Maui, 2012, pp. 800–805.

[10] K. You, L. Xie, and S. Song, "Asymptotically optimal parameter estimation with scheduled measurements," *IEEE Trans. Sig. Process.*, vol. 61, no. 14, pp. 3521–3531, Jul. 2013.

[11] D. Needell, N. Srebro, and R. Ward, "Stochastic gradient descent and the randomized Kaczmarz algorithm," *ArXiv e-prints. [Online]. Available: arXiv:1310.5715v2.*, 2014.

[12] T. Strohmer and R. Vershynin, "A randomized Kaczmarz algorithm with exponential convergence," *J. Fourier Anal. Appl.*, vol. 15, no. 2, pp. 262–278, 2009.

[13] A. Agaskar, C. Wang, and Y. M. Lu, "Randomized Kaczmarz algorithms: Exact MSE analysis and optimal sampling probabilities," in *Proc. Global Conf. Sig. Info. Proc.*, Atlanta, Dec. 2014, pp. 389–393.

[14] T. Y Young and T. W. Calvert, *Classification, Estimation and Pattern Recognition*, North-Holland, 1974.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer, 2009.