Adding vs. Averaging in Distributed Primal-Dual Optimization

Chenxin Ma* CHM514@LEHIGH.EDU
Industrial and Systems Engineering, Lehigh University, USA

Virginia Smith* VSMITH@BERKELEY.EDU
University of California, Berkeley, USA

Martin Jaggi JAGGI@INF.ETHZ.CH
ETH Zürich, Switzerland

Michael I. Jordan JORDAN@CS.BERKELEY.EDU
University of California, Berkeley, USA

Peter Richtárik PETER.RICHTARIK@ED.AC.UK
School of Mathematics, University of Edinburgh, UK

Martin Takáč TAKAC.MT@GMAIL.COM
Industrial and Systems Engineering, Lehigh University, USA

*Authors contributed equally.

Abstract

Distributed optimization methods for large-scale machine learning suffer from a communication bottleneck. It is difficult to reduce this bottleneck while still efficiently and accurately aggregating partial work from different machines. In this paper, we present a novel generalization of the recent communication-efficient primal-dual framework (CoCoA) for distributed optimization. Our framework, CoCoA$^+$, allows for *additive* combination of local updates to the global parameters at each iteration, whereas previous schemes only allow conservative averaging. We give stronger (primal-dual) convergence rate guarantees for both CoCoA as well as our new variants, and generalize the theory for both methods to cover non-smooth convex loss functions. We provide an extensive experimental comparison that shows the markedly improved performance of CoCoA$^+$ on several real-world distributed datasets, especially when scaling up the number of machines.

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

1. Introduction

With the wide availability of large datasets that exceed the storage capacity of single machines, distributed optimization methods for machine learning have become increasingly important. Existing methods require significant communication between workers, frequently equaling the amount of local computation (or reading of local data). As a result, distributed machine learning suffers significantly from a communication bottleneck on real world systems, where communication is typically several orders of magnitudes slower than reading data from main memory.

In this work we focus on optimization problems with empirical loss minimization structure, i.e., objectives that are a sum of the loss functions of each datapoint. This includes the most commonly used regularized variants of linear regression and classification methods. For this class of problems, the recently proposed CoCoA approach (Yang, 2013; Jaggi et al., 2014) develops a communication-efficient primal-dual scheme that targets the communication bottleneck, allowing more computation on data-local subproblems native to each machine before communication. By appropriately choosing the amount of local computation per round, this framework allows one to control the trade-off between *communication* and *local computation* based on the systems hardware at hand.

However, the performance of CoCoA (as well as related primal SGD-based methods) is significantly reduced by the

need to average updates between all machines. As the number of machines K grows, the updates get diluted and slowed by $1/K$, e.g., in the case where all machines except one would have already reached the solutions of their respective partial optimization tasks. On the other hand, if the updates are instead added, the algorithms can diverge, as we will observe in the practical experiments below.

To address both described issues, in this paper we develop a novel generalization of the local CoCoA subproblems assigned to each worker, making the framework more powerful in the following sense: Without extra computational cost, the set of locally computed updates from the modified subproblems (one from each machine) can be combined more efficiently between machines. The proposed CoCoA$^+$ updates can be aggressively *added* (hence the '+'-suffix), which yields much faster convergence both in practice and in theory. This difference is particularly significant as the number of machines K becomes large.

1.1. Contributions

Strong Scaling. To our knowledge, our framework is the first to exhibit favorable *strong scaling* for the class of problems considered, as the number of machines K increases and the data size is kept fixed. More precisely, while the convergence rate of CoCoA degrades as K is increased, the stronger theoretical convergence rate here is – in the worst case – *independent* of K. Our experiments in Section 7 confirm the improved speed of convergence. Since the number of communicated vectors is only one per round and worker, this favorable scaling might be surprising. Indeed, for existing methods, splitting data among more machines generally increases communication requirements (Shamir & Srebro, 2014), which can severely affect overall runtime.

Theoretical Analysis of Non-Smooth Losses. While the existing analysis for CoCoA in (Jaggi et al., 2014) only covered smooth loss functions, here we extend the class of functions where the rates apply, additionally covering, e.g., Support Vector Machines and non-smooth regression variants. We provide a primal-dual convergence rate for both CoCoA as well as our new method CoCoA$^+$ in the case of general convex (L-Lipschitz) losses.

Primal-Dual Convergence Rate. Furthermore, we additionally strengthen the rates by showing stronger primal-dual convergence for both algorithmic frameworks, which are almost tight to their objective-only counterparts. Primal-dual rates for CoCoA had not previously been analyzed in the general convex case. Our primal-dual rates allow efficient and practical certificates for the optimization quality, e.g., for stopping criteria. The new rates apply to both smooth and non-smooth losses, and for both CoCoA as well as the extended CoCoA$^+$.

Experimental Results. We provide a thorough experimental comparison with competing algorithms using several real-world distributed datasets. Our practical results confirm the strong scaling of CoCoA$^+$ as the number of machines K grows, while competing methods, including the original CoCoA, slow down significantly with larger K. We implement all algorithms in Spark, and our code is publicly available at: `github.com/gingsmith/cocoa`.

1.2. History and Related Work

While optimal algorithms for the serial (single machine) case are already well researched and understood, the literature in the distributed setting is relatively sparse. In particular, details on optimal trade-offs between computation and communication, as well as optimization or statistical accuracy, are still widely unclear. For an overview over this currently active research field, we refer the reader to (Balcan et al., 2012; Richtárik & Takáč, 2013; Duchi et al., 2013; Yang, 2013; Liu & Wright, 2014; Fercoq et al., 2014; Jaggi et al., 2014; Shamir & Srebro, 2014; Shamir et al., 2014; Zhang & Xiao, 2015; Qu & Richtárik, 2014) and the references therein. We provide a detailed comparison of our proposed framework to the related work in Section 6.

2. Setup

We consider regularized empirical loss minimization problems of the following well-established form:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left\{ \mathcal{P}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} \ell_i(\mathbf{x}_i^T\mathbf{w}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \right\} \quad (1)$$

Here the vectors $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^d$ represent the training data examples, and the $\ell_i(.)$ are arbitrary convex real-valued loss functions (e.g., hinge loss), possibly depending on label information for the i-th datapoints. The constant $\lambda > 0$ is the regularization parameter.

The above class includes many standard problems of wide interest in machine learning, statistics, and signal processing, including support vector machines, regularized linear and logistic regression, ordinal regression, and others.

Dual Problem, and Primal-Dual Certificates. The conjugate dual of (1) takes following form:

$$\max_{\boldsymbol{\alpha}\in\mathbb{R}^n} \left\{ \mathcal{D}(\boldsymbol{\alpha}) := -\frac{1}{n}\sum_{j=1}^{n} \ell_j^*(-\alpha_j) - \frac{\lambda}{2}\left\| \frac{A\boldsymbol{\alpha}}{\lambda n} \right\|^2 \right\} \quad (2)$$

Here the data matrix $A = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ collects all data-points as its columns, and ℓ_j^* is the conjugate function to ℓ_j. See, e.g., (Shalev-Shwartz & Zhang, 2013c) for several concrete applications.

It is possible to assign for any dual vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ a corresponding primal feasible point

$$\mathbf{w}(\boldsymbol{\alpha}) = \tfrac{1}{\lambda n} A\boldsymbol{\alpha} \tag{3}$$

The duality gap function is then given by:

$$G(\boldsymbol{\alpha}) := \mathcal{P}(\mathbf{w}(\boldsymbol{\alpha})) - \mathcal{D}(\boldsymbol{\alpha}) \tag{4}$$

By weak duality, every value $\mathcal{D}(\boldsymbol{\alpha})$ at a dual candidate $\boldsymbol{\alpha}$ provides a lower bound on every primal value $\mathcal{P}(\mathbf{w})$. The duality gap is therefore a certificate on the approximation quality: The distance to the unknown true optimum $\mathcal{P}(\mathbf{w}^*)$ must always lie within the duality gap, i.e., $G(\boldsymbol{\alpha}) = \mathcal{P}(\mathbf{w}) - \mathcal{D}(\boldsymbol{\alpha}) \geq \mathcal{P}(\mathbf{w}) - \mathcal{P}(\mathbf{w}^*) \geq 0$.

In large-scale machine learning settings like those considered here, the availability of such a computable measure of approximation quality is a significant benefit during training time. Practitioners using classical primal-only methods such as SGD have no means by which to accurately detect if a model has been well trained, as $P(\mathbf{w}^*)$ is unknown.

Classes of Loss-Functions. To simplify presentation, we assume that all loss functions ℓ_i are non-negative, and

$$\ell_i(0) \leq 1 \qquad \forall i \tag{5}$$

Definition 1 (L-Lipschitz continuous loss)**.** A function $\ell_i : \mathbb{R} \to \mathbb{R}$ is L-Lipschitz continuous if $\forall a, b \in \mathbb{R}$, we have

$$|\ell_i(a) - \ell_i(b)| \leq L|a - b| \tag{6}$$

Definition 2 ($(1/\mu)$-smooth loss)**.** A function $\ell_i : \mathbb{R} \to \mathbb{R}$ is $(1/\mu)$-smooth if it is differentiable and its derivative is $(1/\mu)$-Lipschitz continuous, i.e., $\forall a, b \in \mathbb{R}$, we have

$$|\ell_i'(a) - \ell_i'(b)| \leq \frac{1}{\mu}|a - b| \tag{7}$$

3. The CoCoA$^+$ Algorithm Framework

In this section we present our novel CoCoA$^+$ framework. CoCoA$^+$ inherits the many benefits of CoCoA as it remains a highly flexible and scalable, communication-efficient framework for distributed optimization. CoCoA$^+$ differs algorithmically in that we modify the form of the local subproblems (9) to allow for more aggressive additive updates (as controlled by γ). We will see that these changes allow for stronger convergence guarantees as well as improved empirical performance. Proofs of all statements in this section are given in the supplementary material.

Data Partitioning. We write $\{\mathcal{P}_k\}_{k=1}^K$ for the given partition of the datapoints $[n] := \{1, 2, \ldots, n\}$ over the K worker machines. We denote the size of each part by $n_k = |\mathcal{P}_k|$. For any $k \in [K]$ and $\boldsymbol{\alpha} \in \mathbb{R}^n$ we use the notation $\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n$ for the vector

$$(\boldsymbol{\alpha}_{[k]})_i := \begin{cases} 0, & \text{if } i \notin \mathcal{P}_k, \\ \alpha_i, & \text{otherwise.} \end{cases}$$

Local Subproblems in CoCoA$^+$. We can define a data-local subproblem of the original dual optimization problem (2), which can be solved on machine k and only requires accessing data which is already available locally, i.e., datapoints with $i \in \mathcal{P}_k$. More formally, each machine k is assigned the following local subproblem, depending only on the previous shared primal vector $\mathbf{w} \in \mathbb{R}^d$, and the change in the local dual variables α_i with $i \in \mathcal{P}_k$:

$$\max_{\Delta\boldsymbol{\alpha}_{[k]} \in \mathbb{R}^n} \; \mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]}) \tag{8}$$

where

$$\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]}) := -\frac{1}{n}\sum_{i \in \mathcal{P}_k} \ell_i^*(-\alpha_i - (\Delta\boldsymbol{\alpha}_{[k]})_i) - \frac{1}{K}\frac{\lambda}{2}\|\mathbf{w}\|^2 - \frac{1}{n}\mathbf{w}^T A\Delta\boldsymbol{\alpha}_{[k]} - \frac{\lambda}{2}\sigma'\left\|\frac{1}{\lambda n}A\Delta\boldsymbol{\alpha}_{[k]}\right\|^2 \tag{9}$$

Interpretation. The above definition of the local objective functions $\mathcal{G}_k^{\sigma'}$ are such that they closely approximate the global dual objective $\mathcal{D}$, as we vary the 'local' variable $\Delta\boldsymbol{\alpha}_{[k]}$, in the following precise sense:

Lemma 3. *For any dual $\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha} \in \mathbb{R}^n$, primal $\mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$ and real values γ, σ' satisfying (11), it holds that*

$$\mathcal{D}\Big(\boldsymbol{\alpha} + \gamma\sum_{k=1}^K \Delta\boldsymbol{\alpha}_{[k]}\Big) \geq (1-\gamma)\mathcal{D}(\boldsymbol{\alpha}) + \gamma\sum_{k=1}^K \mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]}) \tag{10}$$

The role of the parameter σ' is to measure the difficulty of the given data partition. For our purposes, we will see that it must be chosen not smaller than

$$\sigma' \geq \sigma'_{min} := \gamma \max_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{\|A\boldsymbol{\alpha}\|^2}{\sum_{k=1}^K \|A\boldsymbol{\alpha}_{[k]}\|^2} \tag{11}$$

In the following lemma, we show that this parameter can be upper-bounded by γK, which is trivial to calculate for all values $\gamma \in \mathbb{R}$. We show experimentally (Section 7) that this safe upper bound for σ' has a minimal effect on the overall performance of the algorithm. Our main theorems below show convergence rates dependent on $\gamma \in [\frac{1}{K}, 1]$, which we refer to as the *aggregation parameter*.

Lemma 4. *The choice of $\sigma' := \gamma K$ is valid for (11), i.e.,*

$$\gamma K \geq \sigma'_{min}$$

Notion of Approximation Quality of the Local Solver.

Assumption 1 (Θ-approximate solution)**.** *We assume that there exists $\Theta \in [0, 1)$ such that $\forall k \in [K]$, the local solver at any outer iteration t produces a (possibly) randomized approximate solution $\Delta\boldsymbol{\alpha}_{[k]}$, which satisfies*

$$\mathbb{E}\big[\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}^*_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]}) - \mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]})\big] \leq \Theta\left(\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}^*_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]}) - \mathcal{G}_k^{\sigma'}(\mathbf{0}; \mathbf{w}, \boldsymbol{\alpha}_{[k]})\right), \tag{12}$$

where

$$\Delta\boldsymbol{\alpha}_{[k]}^{*} \in \arg\max_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]}) \quad \forall k \in [K] \tag{13}$$

We are now ready to describe the CoCoA$^+$ framework, shown in Algorithm 1. The crucial difference compared to the existing CoCoA algorithm (Jaggi et al., 2014) is the more general local subproblem, as defined in (9), as well as the aggregation parameter γ. These modifications allow the option of directly adding updates to the global vector $\mathbf{w}$.

Algorithm 1 CoCoA$^+$ Framework

1: **Input:** Datapoints A distributed according to partition $\{\mathcal{P}_k\}_{k=1}^K$. Aggregation parameter $\gamma \in (0, 1]$, subproblem parameter σ' for the local subproblems $\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}; \mathbf{w}, \boldsymbol{\alpha}_{[k]})$ for each $k \in [K]$.
Starting point $\boldsymbol{\alpha}^{(0)} := \mathbf{0} \in \mathbb{R}^n$, $\mathbf{w}^{(0)} := \mathbf{0} \in \mathbb{R}^d$.
2: **for** $t = 0, 1, 2, \dots$ **do**
3: **for** $k \in \{1, 2, \dots, K\}$ **in parallel over computers do**
4: call the local solver, computing a Θ-approximate solution $\Delta\boldsymbol{\alpha}_{[k]}$ of the local subproblem (9)
5: update $\boldsymbol{\alpha}_{[k]}^{(t+1)} := \boldsymbol{\alpha}_{[k]}^{(t)} + \gamma\, \Delta\boldsymbol{\alpha}_{[k]}$
6: return $\Delta\mathbf{w}_k := \frac{1}{\lambda n} A\Delta\boldsymbol{\alpha}_{[k]}$
7: **end for**
8: reduce $\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} + \gamma \sum_{k=1}^K \Delta\mathbf{w}_k$. (14)
9: **end for**

4. Convergence Guarantees

Before being able to state our main convergence results, we introduce some useful quantities and the following main lemma characterizing the effect of iterations of Algorithm 1, for any chosen internal local solver.

Lemma 5. *Let ℓ_i^* be strongly[1] convex with convexity parameter $\mu \geq 0$ with respect to the norm $\|\cdot\|$, $\forall i \in [n]$. Then for all iterations t of Algorithm 1 under Assumption 1, and any $s \in [0, 1]$, it holds that*

$$\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^{(t+1)}) - \mathcal{D}(\boldsymbol{\alpha}^{(t)})] \geq \gamma(1-\Theta)\Big(sG(\boldsymbol{\alpha}^{(t)}) - \frac{\sigma'}{2\lambda}\big(\frac{s}{n}\big)^2 R^{(t)}\Big), \tag{15}$$

where

$$R^{(t)} := -\tfrac{\lambda\mu n(1-s)}{\sigma' s}\|\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)}\|^2 + \textstyle\sum_{k=1}^K \|A(\mathbf{u}^{(t)} - \boldsymbol{\alpha}^{(t)})_{[k]}\|^2, \tag{16}$$

for $\mathbf{u}^{(t)} \in \mathbb{R}^n$ with

$$-u_i^{(t)} \in \partial\ell_i(\mathbf{w}(\boldsymbol{\alpha}^{(t)})^T\mathbf{x}_i). \tag{17}$$

[1]Note that the case of weakly convex $\ell_i^*(.)$ is explicitly allowed here as well, as the Lemma holds for the case $\mu = 0$.

The following Lemma provides a uniform bound on $R^{(t)}$:

Lemma 6. *If ℓ_i are L-Lipschitz continuous for all $i \in [n]$, then*

$$\forall t : R^{(t)} \leq 4L^2 \underbrace{\sum_{k=1}^K \sigma_k n_k}_{=:\sigma}, \tag{18}$$

where

$$\sigma_k := \max_{\boldsymbol{\alpha}_{[k]}\in\mathbb{R}^n} \frac{\|A\boldsymbol{\alpha}_{[k]}\|^2}{\|\boldsymbol{\alpha}_{[k]}\|^2}. \tag{19}$$

Remark 7. *If all data-points $\mathbf{x}_i$ are normalized such that $\|\mathbf{x}_i\| \leq 1\ \forall i \in [n]$, then $\sigma_k \leq |\mathcal{P}_k| = n_k$. Furthermore, if we assume that the data partition is balanced, i.e., that $n_k = n/K$ for all k, then $\sigma \leq n^2/K$. This can be used to bound the constants $R^{(t)}$, above, as $R^{(t)} \leq \frac{4L^2n^2}{K}$.*

4.1. Primal-Dual Convergence for General Convex Losses

The following theorem shows the convergence for non-smooth loss functions, in terms of objective values as well as primal-dual gap. The analysis in (Jaggi et al., 2014) only covered the case of smooth loss functions.

Theorem 8. *Consider Algorithm 1 with Assumption 1. Let $\ell_i(\cdot)$ be L-Lipschitz continuous, and $\epsilon_G > 0$ be the desired duality gap (and hence an upper-bound on primal sub-optimality). Then after T iterations, where*

$$T \geq T_0 + \max\{\Big\lceil \frac{1}{\gamma(1-\Theta)} \Big\rceil, \frac{4L^2\sigma\sigma'}{\lambda n^2 \epsilon_G \gamma(1-\Theta)}\}, \tag{20}$$

$$T_0 \geq t_0 + \left(\frac{2}{\gamma(1-\Theta)}\left(\frac{8L^2\sigma\sigma'}{\lambda n^2\epsilon_G} - 1\right)\right)_+,$$

$$t_0 \geq \max(0, \Big\lceil \tfrac{1}{\gamma(1-\Theta)} \log(\tfrac{2\lambda n^2(\mathcal{D}(\boldsymbol{\alpha}^*) - \mathcal{D}(\boldsymbol{\alpha}^{(0)}))}{4L^2\sigma\sigma'}) \Big\rceil),$$

we have that the expected duality gap satisfies

$$\mathbb{E}[\mathcal{P}(\mathbf{w}(\overline{\boldsymbol{\alpha}})) - \mathcal{D}(\overline{\boldsymbol{\alpha}})] \leq \epsilon_G,$$

at the averaged iterate

$$\overline{\boldsymbol{\alpha}} := \tfrac{1}{T-T_0}\textstyle\sum_{t=T_0+1}^{T-1} \boldsymbol{\alpha}^{(t)}. \tag{21}$$

The following corollary of the above theorem clarifies our main result: The more aggressive adding of the partial updates, as compared averaging, offers a very significant improvement in terms of total iterations needed. While the convergence in the 'adding' case becomes independent of the number of machines K, the 'averaging' regime shows the known degradation of the rate with growing K, which is a major drawback of the original CoCoA algorithm. This important difference in the convergence speed is not a theoretical artifact but also confirmed in our practical experiments below for different K, as shown e.g. in Figure 2.

We further demonstrate below that by choosing γ and σ' accordingly, we can still recover the original CoCoA algorithm and its rate.

Corollary 9. *Assume that all datapoints $\mathbf{x}_i$ are bounded as $\|\mathbf{x}_i\| \leq 1$ and that the data partition is balanced, i.e. that $n_k = n/K$ for all k. We consider two different possible choices of the aggregation parameter γ:*

- *(CoCoA Averaging, $\gamma := \frac{1}{K}$): In this case, $\sigma' := 1$ is a valid choice which satisfies (11). Then using $\sigma \leq n^2/K$ in light of Remark 7, we have that T iterations are sufficient for primal-dual accuracy ϵ_G, with*

$$T \geq T_0 + \max\{\left\lceil \frac{K}{(1-\Theta)} \right\rceil, \frac{4L^2}{\lambda \epsilon_G (1-\Theta)}\},$$
$$T_0 \geq t_0 + \left(\frac{2K}{(1-\Theta)} \left(\frac{8L^2}{\lambda K \epsilon_G} - 1 \right) \right)_+,$$
$$t_0 \geq \max(0, \left\lceil \tfrac{K}{(1-\Theta)} \log(\tfrac{2\lambda(\mathcal{D}(\boldsymbol{\alpha}^*) - \mathcal{D}(\boldsymbol{\alpha}^{(0)}))}{4KL^2}) \right\rceil)$$

  Hence the more machines K, the more iterations are needed (in the worst case).

- *(CoCoA$^+$ Adding, $\gamma := 1$): In this case, the choice of $\sigma' := K$ satisfies (11). Then using $\sigma \leq n^2/K$ in light of Remark 7, we have that T iterations are sufficient for primal-dual accuracy ϵ_G, with*

$$T \geq T_0 + \max\{\left\lceil \frac{1}{(1-\Theta)} \right\rceil, \frac{4L^2}{\lambda \epsilon_G (1-\Theta)}\},$$
$$T_0 \geq t_0 + \left(\frac{2}{(1-\Theta)} \left(\frac{8L^2}{\lambda \epsilon_G} - 1 \right) \right)_+,$$
$$t_0 \geq \max(0, \left\lceil \tfrac{1}{(1-\Theta)} \log(\tfrac{2\lambda n(\mathcal{D}(\boldsymbol{\alpha}^*) - \mathcal{D}(\boldsymbol{\alpha}^{(0)}))}{4L^2 K}) \right\rceil)$$

  This is significantly better than the averaging case.

In practice, we usually have $\sigma \ll n^2/K$, and hence the actual convergence rate can be much better than the proven worst-case bound. Table 1 shows that the actual value of σ is typically between one and two orders of magnitudes smaller compared to our used upper-bound n^2/K.

Table 1. The ratio of upper-bound $\frac{n^2}{K}$ divided by the true value of the parameter σ, for some real datasets.

K	16	32	64	128	256	512
news	15.483	14.933	14.278	13.390	12.074	10.252
real-sim	42.127	36.898	30.780	23.814	16.965	11.835
rcv1	40.138	23.827	28.204	21.792	16.339	11.099
K	256	512	1024	2048	4096	8192
covtype	17.277	17.260	17.239	16.948	17.238	12.729

4.2. Primal-Dual Convergence for Smooth Losses

The following theorem shows the convergence for smooth losses, in terms of the objective as well as primal-dual gap.

Theorem 10. *Assume the loss functions functions ℓ_i are $(1/\mu)$-smooth $\forall i \in [n]$. We define $\sigma_{\max} = \max_{k \in [K]} \sigma_k$. Then after T iterations of Algorithm 1, with*

$$T \geq \tfrac{1}{\gamma(1-\Theta)} \tfrac{\lambda\mu n + \sigma_{\max}\sigma'}{\lambda \mu n} \log \tfrac{1}{\epsilon_{\mathcal{D}}},$$

it holds that

$$\mathbb{E}[\mathcal{D}(\boldsymbol{\alpha}^*) - \mathcal{D}(\boldsymbol{\alpha}^{(T)})] \leq \epsilon_{\mathcal{D}}.$$

Furthermore, after T iterations with

$$T \geq \tfrac{1}{\gamma(1-\Theta)} \tfrac{\lambda\mu n + \sigma_{\max}\sigma'}{\lambda \mu n} \log \left(\tfrac{1}{\gamma(1-\Theta)} \tfrac{\lambda\mu n + \sigma_{\max}\sigma'}{\lambda \mu n} \tfrac{1}{\epsilon_G} \right),$$

we have the expected duality gap

$$\mathbb{E}[\mathcal{P}(\mathbf{w}(\boldsymbol{\alpha}^{(T)})) - \mathcal{D}(\boldsymbol{\alpha}^{(T)})] \leq \epsilon_G.$$

The following corollary is analogous to Corollary 9, but for the case of smooth loses. It again shows that while the CoCoA variant degrades with the increase of the number of machines K, the CoCoA$^+$ rate is independent of K.

Corollary 11. *Assume that all datapoints $\mathbf{x}_i$ are bounded as $\|\mathbf{x}_i\| \leq 1$ and that the data partition is balanced, i.e., that $n_k = n/K$ for all k. We again consider the same two different possible choices of the aggregation parameter γ:*

- *(CoCoA Averaging, $\gamma := \frac{1}{K}$): In this case, $\sigma' := 1$ is a valid choice which satisfies (11). Then using $\sigma_{\max} \leq n_k = n/K$ in light of Remark 7, we have that T iterations are sufficient for suboptimality $\epsilon_{\mathcal{D}}$, with*

$$T \geq \tfrac{1}{1-\Theta} \tfrac{\lambda\mu K + 1}{\lambda \mu} \log \tfrac{1}{\epsilon_{\mathcal{D}}}$$

  Hence the more machines K, the more iterations are needed (in the worst case).

- *(CoCoA$^+$ Adding, $\gamma := 1$): In this case, the choice of $\sigma' := K$ satisfies (11). Then using $\sigma_{\max} \leq n_k = n/K$ in light of Remark 7, we have that T iterations are sufficient for suboptimality ϵ_G, with*

$$T \geq \tfrac{1}{(1-\Theta)} \tfrac{\lambda\mu + 1}{\lambda \mu} \log \tfrac{1}{\epsilon_{\mathcal{D}}}$$

  This is significantly better than the averaging case. Both rates hold analogously for the duality gap.

4.3. Comparison with Original CoCoA

Remark 12. *If we choose averaging ($\gamma := \frac{1}{K}$) for aggregating the updates, together with $\sigma' := 1$, then the resulting Algorithm 1 is identical to CoCoA analyzed in (Jaggi et al., 2014). However, they only provide convergence for smooth loss functions ℓ_i and have guarantees for dual suboptimality and not the duality gap. Formally, when $\sigma' = 1$, the subproblems (9) will differ from the original dual $\mathcal{D}(.)$ only by an additive constant, which does not affect the local optimization algorithms used within CoCoA.*

5. SDCA as an Example Local Solver

We have shown convergence rates for Algorithm 1, depending solely on the approximation quality Θ of the used local solver (Assumption 1).

As an illustrative example for a local solver, in this section we will show that coordinate ascent (SDCA) applied on the local subproblem will indeed deliver such suitable solutions. The LOCALSDCA solver is summarized in Algorithm 2. As an input, the methods receives the local $\boldsymbol{\alpha}$ variables, as well as a shared vector $\mathbf{w} \overset{(3)}{=} \mathbf{w}(\boldsymbol{\alpha})$ being compatible with the last state of all local $\boldsymbol{\alpha} \in \mathbb{R}^n$ variables. The LOCALSDCA algorithm will then produce a random sequence of iterates $\left\{\Delta\boldsymbol{\alpha}_{[k]}^{(h)}\right\}_{h=1}^{H}$, on each part $k \in [K]$.

Algorithm 2 LOCALSDCA $(\mathbf{w}, \boldsymbol{\alpha}_{[k]}, k, H)$

Input: $\boldsymbol{\alpha}_{[k]}, \mathbf{w} = \mathbf{w}(\boldsymbol{\alpha})$
Data: Local $\{(\mathbf{x}_i, y_i)\}_{i \in \mathcal{P}_k}$
Initialize: $\Delta\boldsymbol{\alpha}_{[k]}^{(0)} := \mathbf{0} \in \mathbb{R}^n$
for $h = 0, 1, \ldots, H-1$ **do**
  choose $i \in \mathcal{P}_k$ uniformly at random
  $\delta_i^* := \arg\max_{\delta_i \in \mathbb{R}} \mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}^{(h)} + \delta_i \mathbf{e}_i; \mathbf{w}, \boldsymbol{\alpha}_{[k]})$
  $\Delta\boldsymbol{\alpha}_{[k]}^{(h+1)} := \Delta\boldsymbol{\alpha}_{[k]}^{(h)} + \delta_i^* \mathbf{e}_i$
end for
Output: $\Delta\boldsymbol{\alpha}_{[k]}^{(H)}$

The following two Theorems (13, 14) characterize the convergence of the LOCALSDCA method given in Algorithm 2, for both smooth and non-smooth functions. In all the results we will use $r_{\max} := \max_{i \in [n]} \|\mathbf{x}_i\|^2$.

Theorem 13. *Assume the functions ℓ_i are $(1/\mu)-$smooth for $i \in [n]$. Then Assumption 1 on the local approximation quality Θ is satisfied for* LOCALSDCA *as given in Algorithm 2, if we choose the number of inner iterations H as*

$$H \geq n_k \frac{\sigma' r_{\max} + \lambda n \mu}{\lambda n \mu} \log \frac{1}{\Theta}. \tag{22}$$

Theorem 14. *Assume the functions ℓ_i are L-Lipschitz for $i \in [n]$. Then Assumption 1 on the local approximation quality Θ is satisfied for* LOCALSDCA *as given in Algorithm 2, if we choose the number of inner iterations H as*

$$H \geq n_k \left(\frac{1-\Theta}{\Theta} + \frac{\sigma' r_{\max}}{2\Theta\lambda n^2} \frac{\|\Delta\boldsymbol{\alpha}_{[k]}^*\|^2}{\mathcal{G}_k^{\sigma'}(\Delta\boldsymbol{\alpha}_{[k]}^*; .) - \mathcal{G}_k^{\sigma'}(\mathbf{0}; .)} \right). \tag{23}$$

Remark 15. *Between the different regimes allowed in* COCOA$^+$ *(ranging between averaging and adding the updates) the computational cost for obtaining the required local approximation quality varies with the choice of σ'. From the above worst-case upper bound, we note that the cost can increase with σ', as aggregation becomes more aggressive. However, as we will see in the practical experiments in Section 7 below, the additional cost is negligible compared to the gain in speed from the different aggregation, when measured on real datasets.*

6. Discussion and Related Work

SGD-based Algorithms. For the empirical loss minimization problems of interest here, stochastic subgradient descent (SGD) based methods are well-established. Several distributed variants of SGD have been proposed (Niu et al., 2011; Liu et al., 2014; Duchi et al., 2013). Many build on the idea of a parameter-server, receiving all SGD updates performed by each local worker. The downside of this approach, even when carefully implemented, is that the amount of required communication is equal to the amount of data read locally (e.g., mini-batch SGD with a batch size of 1 per worker). These variants are in practice not competitive with the more communication-efficient methods considered here, which allow more local updates per round.

One-Shot Communication Schemes. At the other extreme, there are distributed methods using only a single round of communication, such as (Zhang et al., 2013; Zinkevich et al., 2010; Mann et al., 2009; McWilliams et al., 2014). These require additional assumptions on the partitioning of the data, and furthermore can not guarantee convergence to the optimum solution for all regularizers, as shown in, e.g., (Shamir et al., 2014). (Balcan et al., 2012) shows additional relevant lower bounds on the minimum number of communication rounds necessary for a given approximation quality for similar machine learning problems.

Methods Allowing Local Optimization. High-performance, flexible methods lie in the middle of the two extremes of the communication vs computation tradeoff. It is therefore important to design more meaningful data-local subproblems to be solved per round of communication. In this spirit, (Shamir et al., 2014; Zhang & Xiao, 2015) have proposed distributed Newton-type algorithms. In this approach, the subproblems have to be solved to very high accuracy for the convergence rate to hold, which is often prohibitive as the size of the data on one machine is still relative large.
The COCOA framework (Jaggi et al., 2014) allows using local solvers of weak local approximation quality in each round, while still giving a convergence rate for smooth losses. By making use of the primal-dual structure in the line of work of (Yu et al., 2012; Pechyony et al., 2011; Yang, 2013; Yang et al., 2013; Jaggi et al., 2014; Lee & Roth, 2015), the COCOA framework as well as the extension here allow more control over the meaningful aggregation of updates between different machines.

Mini-Batch Methods. Mini-batch versions of both SGD and coordinate descent (CD) (Richtárik & Takáč, 2013; Shalev-Shwartz & Zhang, 2013b; Yang, 2013; Qu & Richtárik, 2014; Qu et al., 2014) suffer from their convergence rate degrading towards the rate of batch gradient descent as the size of the mini-batch is increased. This fol-

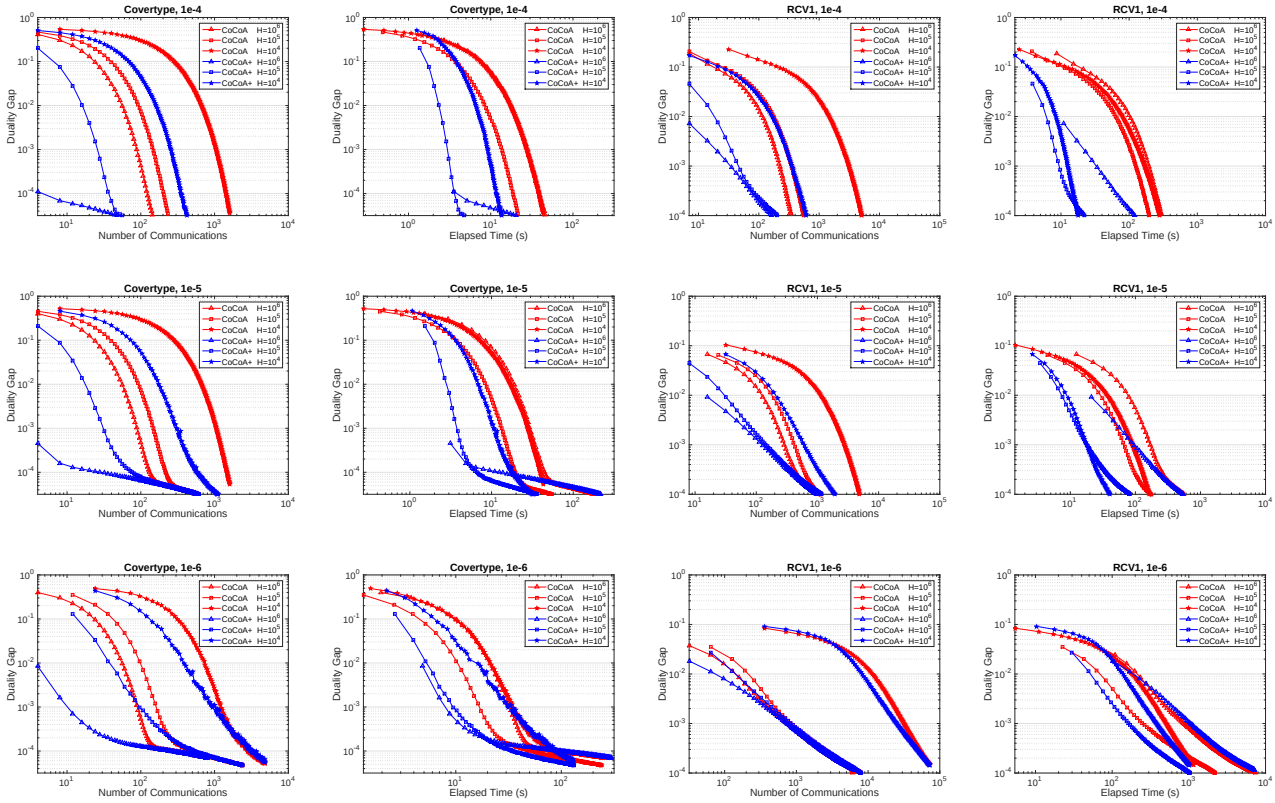

Figure 1. Duality gap vs. the number of communicated vectors, as well as duality gap vs. elapsed time in seconds for two datasets: Covertype (left, K=4) and RCV1 (right, K=8). Both are shown on a log-log scale, and for three different values of regularization (λ=1e-4; 1e-5; 1e-6). Each plot contains a comparison of CoCoA (red) and CoCoA$^+$ (blue), for three different values of H, the number of local iterations performed per round. For all plots, across all values of λ and H, we see that CoCoA$^+$ converges to the optimal solution faster than CoCoA, in terms of both the number of communications and the elapsed time.

lows because mini-batch updates are made based on the outdated previous parameter vector $\mathbf{w}$, in contrast to methods that allow immediate local updates such as CoCoA or local-SGD (Jaggi et al., 2014). Furthermore, the aggregation parameter for mini-batch methods is harder to tune, as it can lie anywhere in the order of mini-batch size. In order to balance computation and communication, the mini-batch size must typically be chosen at least two orders of magnitude larger than the number of machines K. In the CoCoA setting, the aggregation parameter is easier to tune, as it is in the smaller range given by K. Our CoCoA$^+$ extension avoids needing to tune this parameter entirely, by adding.

ADMM. An alternative approach to distributed optimization is to use the alternating direction method of multipliers (ADMM), as used for distributed SVM training in, e.g., (Forero et al., 2010). This uses a penalty parameter balancing between the equality constraint $\mathbf{w}$ and the optimization objective (Boyd et al., 2011). However, the known convergence rates for ADMM are weaker than the more problem-tailored methods mentioned previously, and the choice of the penalty parameter is often unclear.

Batch Proximal Methods. In spirit, for the special case of adding ($\gamma = 1$), our method resembles a batch proximal method, using the separable approximation (9) instead of the original dual (2). Known batch proximal methods require high accuracy subproblem solutions, and don't allow arbitrary solvers of weak accuracy Θ such as here.

7. Numerical Experiments

We present experiments on several large real-world datasets distributed across multiple machines, running on Apache Spark (Zaharia et al., 2012). We show that CoCoA$^+$ converges to the optimal solution faster in terms of total rounds as well as elapsed time as compared to CoCoA in all cases, despite varying: the dataset, values of regularization, batch size, and cluster size (Section 7.2). In Section 7.3 we demonstrate that this performance translates to orders of magnitude improvement in convergence when scaling up the number of machines K, as compared to CoCoA as well as to several other state-of-the-art methods. Finally, in Section 7.4 we investigate the impact of the local subproblem parameter σ' in the CoCoA$^+$ framework.

7.1. Implementation Details

We implement CoCoA$^+$ and all other algorithms for comparison in Apache Spark and run them on Amazon EC2, using m3.large instances. We apply all methods to the binary hinge-loss support vector machine and use SDCA as our local solver. The analysis for this non-smooth loss was not covered in (Jaggi et al., 2014) but has been captured here, and thus is both theoretically and practically justified. A summary of the datasets used is shown in Table 2.

Table 2. Datasets for Numerical Experiments.

Dataset	n	d	**Sparsity**
covertype	522,911	54	22.22%
epsilon	400,000	2,000	100%
RCV1	677,399	47,236	0.16%

7.2. Comparison of CoCoA$^+$ and CoCoA

We compare the CoCoA$^+$ and CoCoA frameworks directly using two datasets (Covertype and RCV1) across various values of λ, the regularizer, in Figure 1. For each value of λ we consider both methods with different values of H, the number of local iterations performed before communicating to the master. For all runs of CoCoA$^+$ we use the safe upper bound of γK for σ'. In terms of both the total number of communications made and the elapsed time, CoCoA$^+$ (shown in blue) converges to the optimal solution faster than CoCoA (red). The discrepancy is larger for greater values of λ, where the strongly convex regularizer has more of an impact and the problem difficulty is reduced. We also see a greater performance gap for smaller values of H, where there is frequent communication between the machines and the master, and changes between the algorithms therefore play a larger role.

7.3. Scaling the Number of Machines K

In Figure 2 we demonstrate the ability of CoCoA$^+$ to scale with an increasing number of machines K. The experiments confirm the ability of strong scaling of the new method, as predicted by our theory in Section 4, in contrast to the competing methods. Unlike CoCoA, which becomes linearly slower when increasing the number of machines, the performance of CoCoA$^+$ improves with additional machines, only starting to degrade slightly once K=16 for the RCV1 dataset.

7.4. Impact of the Subproblem Parameter σ'

Finally, in Figure 3, we consider the effect of the choice of the subproblem parameter σ' on convergence. We plot both the number of communications and clock time on a log-log scale for the RCV1 dataset with K=8 and H=$1e4$. For $\gamma = 1$ (the most aggressive variant of CoCoA$^+$ in which updates are added) we consider several different values of

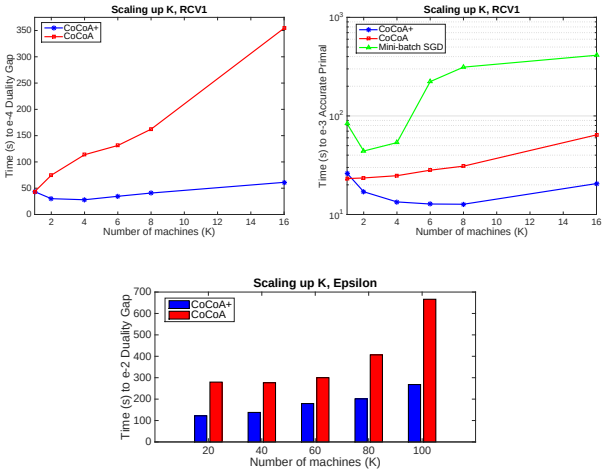

Figure 2. The effect of increasing K on the time (s) to reach an $\epsilon_{\mathcal{D}}$-accurate solution. We see that CoCoA$^+$ converges twice as fast as CoCoA on 100 machines for the Epsilon dataset, and nearly 7 times as quickly for the RCV1 dataset. Mini-batch SGD converges an order of magnitude more slowly than both methods.

σ', ranging from 1 to 8. The value σ'=8 represents the safe upper bound of γK. The optimal convergence occurs around σ'=4, and diverges for $\sigma' \leq 2$. Notably, we see that the easy to calculate upper bound of $\sigma' := \gamma K$ (as given by Lemma 4) has only slightly worse performance than best possible subproblem parameter in our setting.

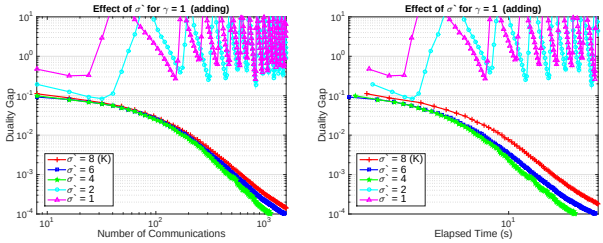

Figure 3. The effect of σ' on convergence of CoCoA$^+$ for the RCV1 dataset distributed across K=8 machines. Decreasing σ' improves performance in terms of communication and overall run time until a certain point, after which the algorithm diverges. The "safe" upper bound of σ':=K=8 has only slightly worse performance than the practically best "un-safe" value of σ'.

8. Conclusion

In conclusion, we present a novel framework CoCoA$^+$ that allows for fast and communication-efficient *additive aggregation* in distributed algorithms for primal-dual optimization. We analyze the theoretical performance of this method, giving strong primal-dual convergence rates with outer iterations scaling independently of the number of machines. We extended our theory to allow for non-smooth losses. Our experimental results show significant speedups over previous methods, including the original CoCoA framework as well as other state-of-the-art methods.

References

Balcan, M.-F., Blum, A., Fine, S., and Mansour, Y. Distributed Learning, Communication Complexity and Privacy. In *COLT*, pp. 26.1–26.22, April 2012.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

Duchi, J. C., Jordan, M. I., and McMahan, H. B. Estimation, Optimization, and Parallelism when Data is Sparse. In *NIPS*, 2013.

Fercoq, O. and Richtárik, P. Accelerated, parallel and proximal coordinate descent. *arXiv:1312.5799*, 2013.

Fercoq, O., Qu, Z., Richtárik, P., and Takáč, M. Fast distributed coordinate descent for non-strongly convex losses. *IEEE Workshop on Machine Learning for Signal Processing*, 2014.

Forero, P. A., Cano, A., and Giannakis, G. B. Consensus-Based Distributed Support Vector Machines. *JMLR*, 11: 1663–1707, 2010.

Jaggi, M., Smith, V., Takáč, M., Terhorst, J., Krishnan, S., Hofmann, T., and Jordan, M. I. Communication-efficient distributed dual coordinate ascent. In *NIPS*, 2014.

Lee, C.-P. and Roth, D. Distributed Box-Constrained Quadratic Optimization for Dual Linear SVM. In *ICML*, 2015.

Liu, J. and Wright, S. J. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *arXiv:1403.3862*, 2014.

Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. An Asynchronous Parallel Stochastic Coordinate Descent Algorithm. In *ICML*, 2014.

Lu, Z. and Xiao, L. On the complexity analysis of randomized block-coordinate descent methods. *arXiv preprint arXiv:1305.4723*, 2013.

Mann, G., McDonald, R., Mohri, M., Silberman, N., and Walker, D. D. Efficient Large-Scale Distributed Training of Conditional Maximum Entropy Models. *NIPS*, 2009.

Mareček, J., Richtárik, P., and Takáč, M. Distributed block coordinate descent for minimizing partially separable functions. *arXiv:1406.0238*, 2014.

McWilliams, B., Heinze, C., Meinshausen, N., Krummenacher, G., and Vanchinathan, H. P. LOCO: Distributing Ridge Regression with Random Projections. *arXiv stat.ML*, June 2014.

Niu, F., Recht, B., Ré, C., and Wright, S. J. Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. In *NIPS*, 2011.

Pechyony, D., Shen, L., and Jones, R. Solving Large Scale Linear SVM with Distributed Block Minimization. In *NIPS Workshop on Big Learning*, 2011.

Qu, Z. and Richtárik, P. Coordinate descent with arbitrary sampling I: Algorithms and complexity. *arXiv:1412.8060*, 2014.

Qu, Z., Richtárik, P., and Zhang, T. Randomized dual coordinate ascent with arbitrary sampling. *arXiv:1411.5873*, 2014.

Richtárik, P. and Takáč, M. Distributed coordinate descent method for learning with big data. *arXiv preprint arXiv:1310.2059*, 2013.

Richtárik, P. and Takáč, M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, April 2014.

Richtárik, P. and Takáč, M. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, pp. 1–52, 2015.

Shalev-Shwartz, S. and Zhang, T. Accelerated mini-batch stochastic dual coordinate ascent. In *NIPS*, 2013a.

Shalev-Shwartz, S. and Zhang, T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *arXiv:1309.2375*, 2013b.

Shalev-Shwartz, S. and Zhang, T. Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. *JMLR*, 14:567–599, February 2013c.

Shamir, O. and Srebro, N. Distributed Stochastic Optimization and Learning . In *Allerton*, 2014.

Shamir, O., Srebro, N., and Zhang, T. Communication efficient distributed optimization using an approximate newton-type method. In *ICML*, 2014.

Tappenden, R., Takáč, M., and Richtárik, P. On the complexity of parallel coordinate descent. Technical report, 2015. ERGO 15-001, University of Edinburgh.

Yang, T. Trading Computation for Communication: Distributed Stochastic Dual Coordinate Ascent. In *NIPS*, 2013.

Yang, T., Zhu, S., Jin, R., and Lin, Y. On Theoretical Analysis of Distributed Stochastic Dual Coordinate Ascent. *arXiv:1312.1031*, December 2013.

Yu, H.-F., Hsieh, C.-J., Chang, K.-W., and Lin, C.-J. Large Linear Classification When Data Cannot Fit in Memory. *TKDD*, 5(4):1–23, 2012.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *NSDI*, 2012.

Zhang, Y. and Xiao, L. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv:1501.00263*, 2015.

Zhang, Y., Duchi, J. C., and Wainwright, M. J. Communication-Efficient Algorithms for Statistical Optimization. *JMLR*, 14:3321–3363, November 2013.

Zinkevich, M. A., Weimer, M., Smola, A. J., and Li, L. Parallelized Stochastic Gradient Descent. *NIPS*, 2010.