frontiers in Applied Mathematics and Statistics

ORIGINAL RESEARCH
published: 25 July 2018
doi: 10.3389/fams.2018.00033

Dual Free Adaptive Minibatch SDCA for Empirical Risk Minimization

Xi He[1], *Rachael Tappenden*[2] *and Martin Takáč*[1]*

[1] Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, United States, [2] School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand

In this paper we develop an adaptive dual free Stochastic Dual Coordinate Ascent (adfSDCA) algorithm for regularized empirical risk minimization problems. This is motivated by the recent work on dual free SDCA of Shalev-Shwartz [1]. The novelty of our approach is that the coordinates to update at each iteration are selected non-uniformly from an adaptive probability distribution, and this extends the previously mentioned work which only allowed for a uniform selection of "dual" coordinates from a fixed probability distribution. We describe an efficient iterative procedure for generating the non-uniform samples, where the scheme selects the coordinate with the greatest potential to decrease the sub-optimality of the current iterate. We also propose a heuristic variant of adfSDCA that is more aggressive than the standard approach. Furthermore, in order to utilize multi-core machines we consider a mini-batch adfSDCA algorithm and develop complexity results that guarantee the algorithm's convergence. The work is concluded with several numerical experiments to demonstrate the practical benefits of the proposed approach.

Keywords: SDCA, importance sampling, non-uniform sampling, mini-batch, adaptive

OPEN ACCESS

Edited by:
Darinka Dentcheva,
Stevens Institute of Technology,
United States

Reviewed by:
Ya-Feng Liu,
Chinese Academy of Sciences, China
Wim Van Ackooij,
EDF R&D, France

***Correspondence:**
Martin Takáč
Takac.MT@gmail.com

Specialty section:
This article was submitted to Optimization, a section of the journal Frontiers in Applied Mathematics and Statistics

Received: *05 January 2018*
Accepted: *29 June 2018*
Published: *25 July 2018*

Citation:
He X, Tappenden R and Takáč M (2018) Dual Free Adaptive Minibatch SDCA for Empirical Risk Minimization. Front. Appl. Math. Stat. 4:33. doi: 10.3389/fams.2018.00033

1. INTRODUCTION

In this work we study the ℓ_2-regularized Empirical Risk Minimization (ERM) problem, which is widely used in the field of machine learning. The problem can be stated as follows. Given training examples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$, loss functions $\phi_1, \ldots, \phi_n : \mathbb{R} \to \mathbb{R}$ and a regularization parameter $\lambda > 0$, ℓ_2-regularized ERM is an optimization problem of the form

$$\min_{w \in \mathbb{R}^d} P(w) := \frac{1}{n} \sum_{i=1}^{n} \phi_i(w^T x_i) + \frac{\lambda}{2} \|w\|^2, \tag{P}$$

where the first term in the objective function is a *data fitting term* and the second is a *regularization term* that prevents over-fitting.

Many algorithms have been proposed to solve problem (P) over the past few years, including SGD [2], SVRG and S2GD [3–5], and SAG/SAGA [6–8]. However, another very popular approach to solving ℓ_2-regularized ERM problems is to consider the following dual formulation

$$\max_{\alpha \in \mathbb{R}^n} D(\alpha) := -\frac{1}{n} \sum_{i=1}^{n} \phi_i^*(-\alpha_i) - \frac{\lambda}{2} \| \frac{1}{\lambda n} X^T \alpha \|^2, \tag{D}$$

where $X^T = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$ is the data matrix and ϕ_i^* denotes the Fenchel conjugate of ϕ_i, namely, $\phi_i^*(u) = \max_z(zu - \phi_i(z))$. It is also known that $P(w^*) = D(\alpha^*)$, which implies that for all w and α, we have $P(w) \geq D(\alpha)$, and hence the duality gap, defined to be $P(w(\alpha)) - D(\alpha)$, can be regarded as an upper bound on the primal sub-optimality $P(w(\alpha)) - P(w^*)$. The structure of the dual formulation (D) makes it well suited to a multicore or distributed computational setting, and several algorithms have been developed to take advantage of this including [9–16].

A popular method for solving (D) is Stochastic Dual Coordinate Ascent (SDCA). The algorithm proceeds as follows. At iteration t of SDCA a coordinate $i \in \{1, \ldots, n\}$ is chosen uniformly at random and the current iterate $\alpha^{(t)}$ is updated to $\alpha^{(t+1)} := \alpha^{(t)} + \delta^* e_i$, where $\delta^* = \arg\max_{\delta\in\mathbb{R}} D(\alpha^{(t)} + \delta e_i)$. Much research has focused on analyzing the theoretical complexity of SDCA under various assumptions imposed on the functions ϕ_i^*, including the pioneering work of Nesterov in Nesterov [17] and others including [10, 13, 18–22].

A modification that has led to improvements in the practical performance of SDCA is the use of *importance sampling* when selecting the coordinate to update. That is, rather than using uniform probabilities, instead coordinate i is sampled with an arbitrary probability p_i.

In many cases algorithms that employ non-uniform coordinate sampling outperform naïve uniform selection, and in some cases help to decrease the number of iterations needed to achieve a desired accuracy by several orders of magnitude, see for example [15, 23].

Notation and Assumptions

In this work we use the notation $[n] \overset{\text{def}}{=} \{1, \ldots, n\}$, as well as the following assumption. For all $i \in [n]$, the loss function ϕ_i is $\tilde{L}_i$-smooth with $\tilde{L}_i > 0$, i.e., for any given $\beta, \delta \in \mathbb{R}$, we have

$$|\phi_i'(\beta) - \phi_i'(\beta + \delta)| \leq \tilde{L}_i|\delta|. \tag{1}$$

In addition, it is simple to observe that the function $\phi_i(x_i^T \cdot) : \mathbb{R}^d \to \mathbb{R}$ is L_i smooth, i.e., $\forall w, \bar{w} \in \mathbb{R}^d$, and for all $i \in [n]$ there exists a constant $L_i \leq \|x_i\|^2\tilde{L}_i$ such that

$$\|\nabla\phi_i(x_i^T w) - \nabla\phi_i(x_i^T \bar{w})\| \leq L_i\|w - \bar{w}\|. \tag{2}$$

We will use the notation

$$L = \max_{1\leq i\leq n} L_i, \qquad \text{and} \qquad \tilde{L} = \max_{1\leq i\leq n} \tilde{L}_i. \tag{3}$$

Throughout this work we let $\mathbb{R}_+$ denote the set of nonnegative real numbers and we let $\mathbb{R}_+^n$ denote the set of n-dimensional vectors with all components being real and nonnegative.

1.1. Contributions

In this section the main contributions of this paper are summarized (not in order of significance).

Adaptive SDCA We modify the dual free SDCA algorithm proposed in Shalev-Shwartz [24] to allow for the adaptive adjustment of probabilities and a non-uniform selection of coordinates. Note that the method is dual free, and hence in contrast to classical SDCA, where the update is defined by maximizing the dual objective (D), here we define the update slightly differently (see section 2 for details).

Allowing non-uniform selection of coordinates from an adaptive probability distribution leads to improvements in practical performance and the algorithm achieves a better convergence rate than in Shalev-Shwartz [24]. In short, we show that the error after T iterations is decreased by a factor of $\prod_{t=1}^T(1 - \theta^{(t)}) \geq (1 - \theta^*)^T$ on average, where θ^* is an uniformly lower bound for all $\theta^{(t)}$. Here $1 - \theta^{(t)} \in (0, 1)$ is a parameter that depends on the current iterate $\alpha^{(t)}$ and the nonuniform probability distribution. By changing the coordinate selection strategy from uniform selection to adaptive, each $1-\theta^{(t)}$ becomes smaller, which leads to an improvement in the convergence rate.

Non-uniform Sampling Procedure Rather than using a uniform sampling of coordinates, which is the commonly used approach, here we propose the use of non-uniform sampling from an adaptive probability distribution. With this novel sampling strategy, we are able to generate non-uniform non-overlapping and proper (see section 5) samplings for arbitrary marginal distributions under only one mild assumptions. Indeed, we show that without the assumption, there is no such non-uniform sampling strategy. We also extend our sampling strategy to allow the selection of mini-batches.

Better Convergence and Complexity Results By utilizing an adaptive probabilities strategy, we can derive complexity results for our new algorithm that, for the case when every loss function is convex, depend only on the *average* of the Lipschitz constants L_i. This improves upon the complexity theory developed in Shalev-Shwartz [24] (which uses a uniform sampling) and Csiba and Richtárik [25] (which uses an arbitrary but fixed probability distribution), because the results in those works depend on the *maximum* Lipschitz constant. Furthermore, even though adaptive probabilities are used here, we are still able to retain the very nice feature of the work in Shalev-Shwartz [24], and show that the variance of the update naturally goes to zero as the iterates converge to the optimum without any additional computational effort or storage costs. Our adaptive probabilities SDCA method also comes with an improved bound on the variance of the update in terms of the sub-optimality of the current iterate.

Practical Aggressive Variant Following from the work of Csiba et al. [15], we propose an efficient heuristic variant of adfSDCA. For adfSDCA the adaptive probabilities must be computed at every iteration (i.e., once a single coordinate has been selected), which can be computationally expensive. However, for our heuristic adfSDCA variant the (exact/true) adaptive probabilities are only computed once at the beginning of each epoch (where an epoch is one pass over the data/n coordinate updates), and during that epoch, once a coordinate has been selected we simply reduce the probability associated with that coordinate so it is not selected again during that epoch. Intuitively this is reasonable because, after a coordinate has been updated the dual residue associated with that coordinate decreases and thus the probability of choosing this coordinate should also reduce. We show that in practice this heuristic adfSDCA variant converges and the computational effort required by this algorithm is lower than adfSDCA (see sections 4 and 6).

Mini-Batch Variant We extend the (serial) adfSDCA algorithm to incorporate a mini-batch scheme. The motivation for this approach is that there is a computational cost associated with generating the adaptive probabilities, so it is important to utilize them effectively. We develop a non-uniform mini-batch strategy that allows us to update multiple coordinates in one iteration, and the coordinates that are selected have high potential to decrease the sub-optimality of the current iterate. Further, we make use of ESO framework (Expected Separable Overapproximation) [see for example [14, 26]] and present theoretical complexity results for mini-batch adfSDCA. In particular, for mini-batch adfSDCA used with batchsize b, we derive the optimal probabilities to use at each iteration, as well as the best step-size to use to guarantee speedup.

1.2. Outline

This paper is organized as follows. In section 2 we introduce our new Adaptive Dual Free SDCA algorithm (adfSDCA), and highlight its connection with a reduced variance SGD method. In section 3 we provide theoretical convergence guarantees for adfSDCA in the case when all loss functions $\phi_i(\cdot)$ are convex, and also in the case when individual loss functions are allowed to be nonconvex but the average loss functions $\sum_{i=1}^n \phi_i(\cdot)$ is convex. Section 4 introduces a practical heuristic version of adfSDCA, and in section 5 we present a mini-batch adfSDCA algorithm and provide convergence guarantees for that method. Finally, we present the results of our numerical experiments in section 6. Note that the proofs for all the theoretical results developed in this work are left to the Appendix.

2. THE ADAPTIVE DUAL FREE SDCA ALGORITHM

In this section we describe the Adaptive Dual Free SDCA (adfSDCA) algorithm, which is motivated by the dual free SDCA algorithm proposed by Shalev-Shwartz [24]. Note that in dual free SDCA two sequences of primal and dual iterates, $\{w^{(t)}\}_{t=0}^{\infty}$ and $\{\alpha^{(t)}\}_{t=0}^{\infty}$, respectively, are maintained. At every iteration of that algorithm, the variable updates are computed in such a way that the well-known primal-dual relational mapping holds; for every iteration t:

$$w^{(t)} = \frac{1}{\lambda n}\sum_{i=1}^{n} \alpha_i^{(t)} x_i. \tag{4}$$

The dual residue is defined as follows.

Definition 1 (Dual residue, [15]). The dual residue $\kappa^{(t)} = (\kappa_1^{(t)}, \ldots, \kappa_n^{(t)})^T \in \mathbb{R}^n$ associated with $(w^{(t)}, \alpha^{(t)})$ is given by:

$$\kappa_i^{(t)} \overset{\text{def}}{=} \alpha_i^{(t)} + \phi_i'(x_i^T w^{(t)}). \tag{5}$$

The Adaptive Dual Free SDCA algorithm is outlined in Algorithm 1 and is described briefly now; a more detailed description (including a discussion of coordinate selection and how to generate appropriate selection rules) will follow. An initial solution $\alpha^{(0)}$ is chosen, and then $w^{(0)}$ is defined via (4). In each iteration of Algorithm 1 the dual residue $\kappa^{(t)}$ is computed via (5), and this is used to generate a probability distribution $p^{(t)}$. Next, a coordinate $i \in [n]$ is selected (sampled) according to the generated probability distribution and a step of size $\theta^{(t)} \in (0, 1)$ is taken by updating the ith coordinate of α via

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} - \theta^{(t)}(p_i^{(t)})^{-1}\kappa_i^{(t)}. \tag{6}$$

Finally, the vector w is also updated

$$w^{(t+1)} = w^{(t)} - \theta^{(t)}(n\lambda p_i^{(t)})^{-1}\kappa_i^{(t)} x_i, \tag{7}$$

and the process is repeated. Note that the updates to α and w using the formulas (6) and (7) ensure that the equality (4) is preserved.

Also note that the updates in (6) and (7) involve a step size parameter $\theta^{(t)}$, which will play an important role in our complexity results. The step size $\theta^{(t)}$ should be large so that good progress can be made, but it must also be small enough to ensure that the algorithm is guaranteed to converge. Indeed, in section 3.1 we will see that the choice of $\theta^{(t)}$ depends on the choice of probabilities used at iteration t, which in turn depend upon a particular function that is related to the suboptimality at iteration t.

Algorithm 1 Adaptive Dual Free SDCA (adfSDCA)

Input: Data: $\{x_i, \phi_i\}_{i=1}^n$
Initialization: Choose $\alpha^{(0)} \in \mathbb{R}^n$
Set $w^{(0)} = \frac{1}{\lambda n}\sum_{i=1}^n \alpha_i^{(0)} x_i$
for $t = 0, 1, 2, \ldots$ **do**
  Calculate dual residual $\kappa_i^{(t)} = \phi_i'(x_i^T w^{(t)}) + \alpha_i^{(t)}$, for all $i \in [n]$
  Generate adaptive probability distribution $p^{(t)} \sim \kappa^{(t)}$
  Sample coordinate i according to $p^{(t)}$
  Set step-size $\theta^{(t)} \in (0, 1)$ as in (18)
  Update: $\alpha_i^{(t+1)} = \alpha_i^{(t)} - \theta^{(t)}(p_i^{(t)})^{-1}\kappa_i^{(t)}$
  Update: $w^{(t+1)} = w^{(t)} - \theta^{(t)}(n\lambda p_i^{(t)})^{-1}\kappa_i^{(t)} x_i$
end for

The dual residue $\kappa^{(t)}$ is informative and provides a useful way of monitoring suboptimality of the current solution $(w^{(t)}, \alpha^{(t)})$. In particular, note that if $\kappa_i = 0$ for some coordinate i, then by (5) $\alpha_i = -\phi_i'(w^T x_i)$, and substituting κ_i into (6) and (7) shows that $\alpha_i^{(t+1)} \leftarrow \alpha_i^{(t)}$ and $w_i^{(t+1)} \leftarrow w^{(t)}$, i.e., α and w remain unchanged in that iteration. On the other hand, a large value of $|\kappa_i|$ (at some iteration t) indicates that a large step will be taken, which is anticipated to lead to good progress in terms of improvement in sub-optimality of current solution.

The probability distributions used in Algorithm 1 adhere to the following definition.

Definition 2 (Coherence, [15]). Probability vector $p \in \mathbb{R}^n$ is coherent with dual residue $\kappa \in \mathbb{R}^n$ if for any index i in the support set of κ, denoted by $I_\kappa := \{i \in [n] : \kappa_i \neq 0\}$, we have $p_i > 0$. When $i \notin I_\kappa$ then $p_i = 0$. We use $p \sim \kappa$ to represent this coherent relation.

2.1. Adaptive Dual Free SDCA as a Reduced Variance SGD Method

Reduced variance SGD methods have became very popular in the past few years, see for example [3, 7, 8, 27]. It is show in Shalev-Shwartz [24] that uniform dual free SDCA is an instance of a reduced variance SGD algorithm (the variance of the stochastic gradient can be bounded by some measure of sub-optimality of the current iterate) and a similar result applies to adfSDCA in Algorithm 1. In particular, note that conditioned on $\alpha^{(t-1)}$, we have

$$\begin{aligned}\mathbf{E}[w^{(t)}|\alpha^{(t-1)}] &\overset{(7)}{=} w^{(t-1)} - \frac{\theta^{(t-1)}}{\lambda}\sum_{i=1}^{n}\frac{p_i}{np_i}\left(\left(\nabla\phi_i(x_i^T w^{(t-1)}) + \alpha_i^{(t-1)}\right)x_i\right)\\ &\overset{(4)}{=} w^{(t-1)} - \frac{\theta^{(t-1)}}{\lambda}\left(\nabla\left(\frac{1}{n}\sum_{i=1}^{n}\phi_i(x_i^T w^{(t-1)})\right) + \lambda w^{(t-1)}\right)\\ &\overset{(P)}{=} w^{(t-1)} - \frac{\theta^{(t-1)}}{\lambda}\nabla P(w^{(t-1)}). \end{aligned} \tag{8}$$

Combining (7) and (8) and replace $t-1$ by t gives

$$\mathbf{E}\left[\frac{1}{np_i}\kappa_i^{(t)}x_i|\alpha^{(t)}\right] = \nabla P(w^{(t)}), \tag{9}$$

which implies that $\frac{1}{np_i}\kappa_i^{(t)}x_i$ is an unbiased estimator of $\nabla P(w^{(t)})$. Therefore, Algorithm 1 is eventually a variant of the Stochastic Gradient Descent method. However, we can prove (see Corollary 1 and Corollary 2) that the variance of the update goes to zero as the iterates converge to an optimum, which is not true for vanilla Stochastic Gradient Descent.

3. CONVERGENCE ANALYSIS

In this section we state the main convergence results for adfSDCA (Algorithm 1). The analysis is broken into two cases. In the first case it is assumed that each of the loss functions ϕ_i is convex. In the second case this assumption is relaxed slightly and it is only assumed that the average of the ϕ_i's is convex, i.e., individual functions $\phi_i(\cdot)$ for some (several) $i \in [n]$ are allowed to be nonconvex, as long as $\frac{1}{n}\sum_{j=1}^{n}\phi_j(\cdot)$ is convex. The proofs for all the results in this section can be found in the Appendix.

3.1. Case I: All Loss Functions Are Convex

Here we assume that ϕ_i is convex for all $i \in [n]$. Define the following parameter

$$\gamma \overset{\text{def}}{=} \lambda\tilde{L}, \tag{10}$$

where $\tilde{L}$ is given in (3). It will also be convenient to define the following potential function. For all iterations $t \geq 0$,

$$D^{(t)} \overset{\text{def}}{=} \frac{1}{n}\|\alpha^{(t)} - \alpha^*\|^2 + \gamma\|w^{(t)} - w^*\|^2. \tag{11}$$

The potential function (11) plays a central role in the convergence theory presented in this work. It measures the distance from the optimum in both the primal and (pseudo) dual variables. Thus, our algorithm will generate iterates that reduce this suboptimality and therefore push the potential function toward zero.

Also define

$$v_i \overset{\text{def}}{=} \|x_i\|^2 \text{ for all } i \in [n]. \tag{12}$$

We have the following result.

LEMMA 1. *Let $\tilde{L}$, $\kappa_i^{(t)}$, γ, $D^{(t)}$, and v_i be as defined in (3), (5), (10), (11), and (12), respectively. Suppose that ϕ_i is $\tilde{L}$-smooth and convex for all $i \in [n]$ and let $\theta \in (0,1)$. Then at every iteration $t \geq 0$ of Algorithm 1, a probability distribution $p^{(t)}$ that satisfies Definition 2 is generated and*

$$\mathbf{E}\left[D^{(t+1)}|\alpha^{(t)}\right] - (1-\theta)D^{(t)} \leq \sum_{i=1}^{n}\left(-\frac{\theta}{n}\left(1 - \frac{\theta}{p_i^{(t)}}\right) + \frac{\theta^2 v_i\gamma}{n^2\lambda^2 p_i^{(t)}}\right)(\kappa_i^{(t)})^2. \tag{13}$$

Note that if the right hand side of (13) is negative, then the potential function decreases (in expectation) in iteration t:

$$\mathbf{E}\left[D^{(t+1)}|\alpha^{(t)}\right] \leq (1-\theta)D^{(t)}. \tag{14}$$

The purpose of Algorithm 1 is to generate iterates $(w^{(t)}, \alpha^{(t)})$ such that the above holds. To guarantee negativity of the right hand term in (13), or equivalently, to ensure that (14) holds, consider the parameter θ. Specifically, any θ that is less than the function $\Theta(\cdot,\cdot) : \mathbb{R}_+^n \times \mathbb{R}_+^n \to \mathbb{R}$ defined as

$$\Theta(\kappa, p) \overset{\text{def}}{=} \frac{n\lambda^2\sum_{i\in I_\kappa}\kappa_i^2}{\sum_{i\in I_\kappa}(n\lambda^2 + v_i\gamma)p_i^{-1}\kappa_i^2}, \tag{15}$$

will ensure negativity of the right hand term in (13). Moreover, the larger the value of θ, the better progress Algorithm 1 will make in terms of the reduction in $D^{(t)}$. The function Θ depends on the dual residue κ and the probability distribution p. Maximizing this function w.r.t. p will ensure that the largest possible value of θ can be used in Algorithm 1. Thus, we consider the following optimization problem:

$$\max_{p\in\mathbb{R}_+^n, \sum_{i\in I_\kappa} p_i = 1} \Theta(\kappa, p). \tag{16}$$

One may naturally be wary of the additional computational cost incurred by solving the optimization problem in (16) at every iteration. Fortunately, it turns out that there is an (inexpensive) closed form solution, as shown by the following Lemma.

LEMMA 2. *Let $\Theta(\kappa, p)$ be defined in (15). The optimal solution $p^*(\kappa)$ of (16) is*

$$p_i^*(\kappa) = \frac{\sqrt{v_i\gamma + n\lambda^2}|\kappa_i|}{\sum_{j\in I_\kappa}\sqrt{v_j\gamma + n\lambda^2}|\kappa_j|}, \qquad \text{for all } i = 1,\ldots,n. \tag{17}$$

The corresponding θ by using the optimal solution p^* is

$$\theta = \Theta(\kappa, p^*) = \frac{n\lambda^2\sum_{i\in I_\kappa}\kappa_i^2}{(\sum_{i\in I_\kappa}\sqrt{v_i\gamma + n\lambda^2}|\kappa_i|)^2}. \tag{18}$$

PROOF: This can be verified by deriving the KKT conditions of the optimization problem in (16). The details are moved to Appendix for brevity. □

The results in Csiba and Richtárik [25] are weaker because they require a fixed sampling distribution p throughout all iterations. Here we allow adaptive sampling probabilities as in (17), which enables the algorithm to utilize the data information more effectively, and hence we have a better convergence rate. Furthermore, the optimal probabilities found in Csiba et al. ([15] can be only applied to a quadratic loss function, whereas our results are more general because the optimal probabilities in (17) can used whenever the loss functions are convex, or when individual loss functions are non-convex but the average of the loss functions is convex (see section 3.2).

Before proceeding with the convergence theory we define several constants. Let

$$C_0 \overset{\text{def}}{=} \tfrac{1}{n}\|\alpha^{(0)} - \alpha^*\|^2 + \gamma\|w^{(0)} - w^*\|^2, \tag{19}$$

where γ is defined in (10). Note that C_0 in (19) is equivalent to the value of the potential function (11) at iteration $t = 0$, i.e., $C_0 \equiv D^{(0)}$. Moreover, let

$$M \overset{\text{def}}{=} Q\Big(1 + \frac{\gamma Q}{\lambda^2 n}\Big) \quad \text{where} \quad Q \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n \|x_i\|^2 \overset{(12)}{=} \frac{1}{n}\sum_{i=1}^n v_i. \tag{20}$$

Now we have the following theorem.

THEOREM **1.** *Let $\tilde{L}$, $\kappa_i^{(t)}$, γ, $D^{(t)}$, v_i, C_0 and Q be as defined in (3), (5), (10), (11), (12), (19), and (20), respectively. Suppose that ϕ_i is $\tilde{L}$-smooth and convex for all $i \in [n]$, let $\theta^{(t)} \in (0,1)$ be decided by (18) for all $t \geq 0$ and let p^* be defined via (17). Then, setting $p^{(t)} = p^*$ at every iteration $t \geq 0$ of Algorithm 1, gives*

$$\mathbf{E}[D^{(t+1)}|\alpha^{(t)}] \leq (1-\theta^*)D^{(t)}, \tag{21}$$

where

$$\theta^* \overset{def}{=} \frac{n\lambda^2}{\sum_{i=1}^n (v_i\gamma + n\lambda^2)} \leq \theta^{(t)}. \tag{22}$$

Moreover, for $\epsilon > 0$, if

$$T \geq \left(n + \frac{\tilde{L}Q}{\lambda}\right)\log\left(\frac{(\lambda + L)C_0}{2\lambda\tilde{L}\epsilon}\right), \tag{23}$$

then $\mathbf{E}[P(w^{(T)}) - P(w^*)] \leq \epsilon$.

Similar to Shalev-Swartz [24], we have the following corollary which bounds the quantity $\mathbf{E}[\|\frac{1}{np_i}\kappa_i^{(t)}x_i\|^2]$ in terms of the sub-optimality of the points $\alpha^{(t)}$ and $w^{(t)}$ by using optimal probabilities.

COROLLARY 1. *Let the conditions of Theorem 1 hold. Then at every iteration $t \geq 0$ of Algorithm 1,*

$$\mathbf{E}\left[\left\|\frac{\kappa_i^{(t)}x_i}{np_i}\right\|^2 |\alpha^{(t-1)}\right] \leq 2M(\mathbf{E}[\|\alpha^{(t)} - \alpha^*\|^2|\alpha^{(t-1)}] + L\mathbf{E}[\|w^{(t)} - w^*\|^2|\alpha^{(t-1)}]).$$

Note that Theorem 1 can be used to show that both $\mathbf{E}[\|\alpha^{(t)} - \alpha^*\|^2]$ and $\mathbf{E}[\|w^{(t)} - w^*\|^2]$ go to zero as $e^{-\theta^* t}$. We can then show that $\mathbf{E}[\|\frac{1}{np_i}\kappa_i^{(t)}x_i\|^2] \leq \epsilon$ as long as $t \geq \tilde{O}(\frac{1}{\theta^*}\log(\frac{1}{\epsilon}))$. Furthermore, we achieve the same variance reduction rate as shown in Shalev-Swartz [24], i.e., $\mathbf{E}[\|\frac{1}{np_i}\kappa_i^{(t)}x_i\|^2] \sim \tilde{O}(\|\kappa^{(t)}\|^2)$.

For the dual free SDCA algorithm in Shalev-Swartz [24] where uniform sampling is adopted, the parameter θ should be set to at most $\min \frac{\lambda}{\lambda n + \tilde{L}}$, where $\tilde{L} \geq \max_i v_i \cdot L$. However, from Corollary 1, we know that this θ is smaller than θ^*, so dual free SDCA will have a slower convergence rate than our algorithm. In Csiba and Richtárik [25], where they use a fixed probability distribution p_i for sampling of coordinates, they must choose θ less than or equal to $\min_i \frac{p_i n\lambda}{L_i v_i + n\lambda}$. This is consistent with Shalev-Swartz [24] where $p_i = 1/n$ for all $i \in [n]$. With respect to our adfSDCA Algorithm 1, at any iteration t, we have that $\theta^{(t)}$ is greater than or equal to θ^*, which again implies that our convergence results are better.

3.2. Case II: The Average of the Loss Functions is Convex

Here we follow the analysis in Shalev-Swartz [24] and consider the case where individual loss functions $\phi_i(\cdot)$ for $i \in [n]$ are allowed to be nonconvex as long as the average $\frac{1}{n}\sum_{j=1}^n \phi_j(\cdot)$ is convex. First we define several parameters that are analogous to the ones used in section 3.1. Let

$$\bar{\gamma} \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n L_i^2, \tag{24}$$

where L_i is given in (2), and define the following potential function. For all iterations $t \geq 0$, let

$$\bar{D}^{(t)} \overset{\text{def}}{=} \frac{1}{n}\|\alpha^{(t)} - \alpha^*\|^2 + \bar{\gamma}\|w^{(t)} - w^*\|^2. \tag{25}$$

We also define the following constants

$$\bar{C}_0 \overset{\text{def}}{=} \frac{1}{n}\|\alpha^{(0)} - \alpha^*\|^2 + \bar{\gamma}\|w^{(0)} - w^*\|^2, \tag{26}$$

and

$$\bar{M} \overset{\text{def}}{=} Q\left(1 + \frac{\bar{\gamma}Q}{\lambda^2 n}\right). \tag{27}$$

Then we have the following theoretical results.

LEMMA 3. *Let L_i, $\kappa_i^{(t)}$, $\bar{\gamma}$, $\bar{D}^{(t)}$, and v_i be as defined in (2), (5), (24), (25), and (12), respectively. Suppose that every $\phi_i, i \in [n]$ is L_i-smooth and that the average of the n loss functions $\frac{1}{n}\sum_{i=1}^n \phi_i(w^T x_i)$ is convex. Let $\theta \in (0,1)$. Then at every iteration $t \geq 0$ of Algorithm 1, a probability distribution $p^{(t)}$ that satisfies Definition 2 is generated and*

$$\mathbf{E}[\bar{D}^{(t+1)}|\alpha^{(t)}] - (1-\theta)\bar{D}^{(t)} \leq \sum_{i=1}^n \left(-\frac{\theta}{n}\left(1 - \frac{\theta}{p_i^{(t)}}\right) + \frac{\theta^2 v_i\bar{\gamma}}{n^2\lambda^2 p_i^{(t)}}\right)(\kappa_i^{(t)})^2. \tag{28}$$

THEOREM **2.** *Let L, $\kappa_i^{(t)}$, $\bar{\gamma}$ $\bar{D}^{(t)}$, v_i, and $\bar{C}_0$ be as defined in (3), (5), (24), (25), (12), and (26), respectively. Suppose that every $\phi_i, i \in [n]$ is L_i-smooth and that the average of the n loss functions $\frac{1}{n}\sum_{i=1}^n \phi_i(w^T x_i)$ is convex. Let $\theta^{(t)} \in (0,1)$ using (18) for all $t \geq 0$ and let p^* be defined via (17). Then, setting $p^{(t)} = p^*$ at every iteration $t \geq 0$ of Algorithm 1, gives*

$$\mathbf{E}\left[\bar{D}^{(t+1)}|\alpha^{(t)}\right] \leq (1-\theta^*)\bar{D}^{(t)}, \tag{29}$$

where

$$\theta^* = \frac{n\lambda^2}{\sum_{i=1}^n (v_i\bar{\gamma} + n\lambda^2)} \leq \theta^{(t)}.$$

Furthermore, for $\epsilon > 0$, if

$$T \geq \left(n + \frac{\bar{\gamma}Q}{\lambda^2}\right)\log\left(\frac{(\lambda+L)\bar{C}_0}{2\bar{\gamma}\epsilon}\right), \tag{30}$$

then $\mathbf{E}[P(w^{(T)}) - P(w^*)] \leq \epsilon$.

We remark that, $L_i \leq L$ for all $i \in [n]$, so $\bar{\gamma} \leq L^2$, which means that a conservative complexity bound is

$$T \geq \left(n + \frac{L^2 Q}{\lambda^2}\right)\log\left(\frac{(\lambda+L)\bar{C}_0}{2\bar{\gamma}\epsilon}\right).$$

We conclude this section with the following corollary.

COROLLARY 2. *Let the conditions of Theorem 2 hold and let $\bar{M}$ be defined in (27). Then at every iteration $t \geq 0$ of Algorithm 1,*

$$\mathbf{E}\left[\left\|\frac{\kappa_i^{(t)} x_i}{np_i}\right\|^2 |\alpha^{(t-1)}\right] \leq 2\bar{M}(\mathbf{E}[\|\alpha^{(t)} - \alpha^*\|^2|\alpha^{(t-1)}] + L\mathbf{E}[\|w^{(t)} - w^*\|^2|\alpha^{(t-1)}]).$$

4. HEURISTIC ADFSDCA

One of the disadvantages of Algorithm 1 is that it is necessary to update the entire probability distribution $p \sim \kappa$ at each iteration, i.e., every time a single coordinate is updated the probability distribution is also updated. Note that if the data are sparse and coordinate i is sampled during iteration t, then, one need only update probabilities p_j for which $x_j^T x_i \neq 0$; unfortunately for some datasets this can still be expensive. In order to overcome this shortfall we follow the recent work in Csiba et al. [15] and present a heuristic algorithm that allows the probabilities to be updated less frequently and in a computationally inexpensive way. The process works as follows. At the beginning of each epoch the (full/exact) nonuniform probability distribution is computed, and this remains fixed for the next n coordinate updates, i.e., it is fixed for the rest of that epoch. During that same epoch, if coordinate i is sampled (and thus updated) the probability p_i associated with that coordinate is reduced (it is shrunk by $p_i \leftarrow p_i/s$) , where s is the shrinkage parameter. The intuition behind this procedure is that, if coordinate i is updated then the dual residue $|\kappa_i|$ associated with that coordinate will decrease. Thus, there will be little benefit (in terms of reducing the sub-optimality of the current iterate) in sampling and updating that same coordinate i again. To avoid choosing coordinate i in the next iteration, we shrink the probability p_i associated with it, i.e., we reduce the probability by a factor of $1/s$. Moreover, shrinking the coordinate is less computationally expensive than recomputing the full adaptive probability distribution from scratch, and so we anticipate a decrease in the overall running time if we use this heuristic strategy, compared with the standard adfSDCA algorithm. This procedure is stated formally in Algorithm 2. Note that Algorithm 2 does not fit the theory established in section 3. Nonetheless, we have observed convergence in practice and a good numerical performance when using this strategy (see the numerical experiments in section 6).

Algorithm 2 Heuristic Adaptive Dual Free SDCA (adfSDCA+)

Input: Data: $\{x_i, \phi_i\}_{i=1}^n$, probability shrink parameter s
Initialization: Choose $\alpha^{(0)} \in \mathbb{R}^n$
Set $w^{(0)} = \frac{1}{\lambda n}\sum_{i=1}^n \alpha_i^{(0)} x_i$
for $t = 0, 1, 2, \ldots$ **do**
  if $\mod(t, n) == 0$ **then**
    Calculate dual residue $\kappa_i^{(t)} = \phi_i'(x_i^T w^{(t)}) + \alpha_i^{(t)}$, for all $i \in [n]$
    Generating adapted probabilities distribution $p^{(t)} \sim \kappa^{(t)}$
  end if
  Select coordinate i from $[n]$ according to $p^{(t)}$
  Set step-size $\theta^{(t)} \in (0,1)$ as in (18)
  Update: $\alpha_i^{(t+1)} = \alpha_i^{(t)} - \theta^{(t)}(p_i^{(t)})^{-1}\kappa_i^{(t)}$
  Update: $w^{(t+1)} = w^{(t)} - \theta^{(t)}(n\lambda p_i^{(t)})^{-1}\kappa_i^{(t)} x_i$
  Update: $p_i^{(t+1)} = p_i^{(t)}/s$
end for

5. MINI-BATCH ADFSDCA

In this section we propose a mini-batch variant of Algorithm 1. Before doing so, we stress that sampling a mini-batch non-uniformly is not easy. We first focus on the task of generating non-uniform random samples and then we will present our minibatch algorithm.

5.1. Efficient Single Coordinate Sampling

Before considering mini-batch sampling, we first show how to sample a single coordinate from a non-uniform distribution. Note that only discrete distributions are considered here.

There are multiple approaches that can be taken in this case. One naïve approach is to consider the Cumulative Distribution Function (CDF) of p, because a CDF can be computing in $O(n)$ time complexity and it also takes $O(n)$ time complexity to make a decision. One can also use a better data structure (e.g., a binary search tree) to reduce the decision cost to $O(\log n)$ time complexity, although the cost to set up the tree is $O(n \log n)$.

Some more advanced approaches like the so-called alias method of Kronmal and Peterson [28] can be used to sample a single coordinate in only $O(1)$, i.e., sampling a single coordinate can be done in constant time but with a cost of $O(n)$ setup time. The alias method works based on the fact that any n-valued distribution can be written as a mixture of n Bernoulli distributions.

In this paper we choose two sampling update strategies, one each for Algorithms 1 and 2. For adfSDCA in Algorithm 1 the probability distribution must be recalculated at every iteration, so we use the alias method, which is highly efficient. The heuristic approach in Algorithm 2 is a strategy that only alters the probability of a single coordinate (e.g., $p_i = p_i/s$) in each iteration. In this second case it is relatively expensive to use the alias method due to the linear time cost to update the alias structure, so instead we build a binary tree when the algorithm is initialized so that the update complexity reduces to $O(\log(n))$.

5.2. Non-uniform Mini-Batch Sampling

Many randomized coordinate descent type algorithms utilize a sampling scheme that assigns every subset of $[n]$ a probability p_S, where $S \in 2^{[n]}$. In this section, we consider a particular type of sampling called a *mini-batch* sampling that is defined as follows.

***Definition** 3.* A sampling $\hat{S}$ is called a mini-batch sampling, with batchsize b, consistent with the given marginal distribution $q := (q_1, \ldots, q_n)^T$, if the following conditions hold:

1. $|S| = b$;
2. $q_i \overset{\text{def}}{=} \sum_{S \in \hat{S}} P(\{S : i \in S\}) = bp_i$,

where $P(\{S : i \in S\})$ represents the probability of mini-batch sampling S containing the coordinate i.

Here we are going to derive a proper sampling strategy over coordinate i such that $i \in S \in \hat{S}$ and Definition 3 is satisfied. Note that we study samplings $\hat{S}$ that are *non-uniform* since we allow q_i to vary with i. The motivation to design such samplings arises from the fact that we wish to make use of the optimal probabilities that were studied in section 3.

We make several remarks about non-uniform mini-batch samplings below.

1. For a given probability distribution p, one can derive a corresponding mini-batch sampling only if we have $p_i \leq \frac{1}{b}$ for all $i \in [n]$. This is obvious in the sense that $q_i = bp_i = \sum_{S \in \hat{S}} P(\{S : i \in S\}) \leq \sum_{S \in \hat{S}} P(S) = 1$.
2. For a given probability distribution p and a batch size b, the mini-batch sampling may not be unique and it may not be proper, see for example Richtárik and Takáč [26]. (A proper sampling is a sampling for which any subset of size b must have a *positive* probability of being sampled).

In Algorithm 3 we describe an approach that we used to generate a non-uniform mini-batch sampling of batchsize b from a given marginal distribution q. Without loss of generality, we assume that the $q_i \in (0, 1)$ for $i \in [n]$ are sorted from largest to smallest.

We now state several facts about Algorithm 3.

1. Algorithm 3 will terminate in at most n iterations. This is because the update rules for q_i (which depend on r_k at each iteration), ensure that at least one q_i will reduce to become equal to some $q_j < q_i$ (i.e., either $q_{i^{k+1}-1} = q_b$ or $q_{j^{k+1}+1} = q_b$) and since there are n coordinates in total, after at most n iteration it must hold that $q_i = q_j$ for all $i, j \in [n]$. Note that if the algorithm begins with $q_i = q_j$ for all $i, j \in [n]$, which implies a uniform marginal distribution, the algorithm will terminated in a single step.
2. For Algorithm 3 we must have $\sum_{i=1}^{m} r_i = 1$, where we assume that the algorithm terminates at iteration $m \in [1, n]$, since overall we have $\sum_{i=1}^{m} br_i = \sum_{i=1}^{n} q_i = b$.
3. Algorithm 3 will always generate a proper sampling because when it terminates, the situation $p_i = p_j > 0$, for all $i \neq j$, will always hold. Thus, any subset of size b has a positive probability of being sampled.
4. It can be shown that this algorithm works on an arbitrary given marginal probabilities as long as $q_i \in (0, 1)$, for all $i \in [n]$.

Algorithm 3 Non-uniform mini-batch sampling

Input: Marginal distribution $q \in \mathbb{R}^n$ with $q_i \in (0, 1)\ \forall i \in [n]$ and batchsize b such that $\sum_{i=1}^{n} q_i = b$. Define $q_{n+1} = 0$
Output: A mini-batch sampling S (Definition 3)
Initialization: Index set $i, j \in \mathbb{N}^n$, and set $k = 1$.
for $k = 1, \ldots, n$ **do**
  $i^k = \min_i\{i : p_i = q_b\}, j^k = \max_i\{i : p_i = q_b\}$
  Obtain r_k:

$$r_k = \begin{cases} \min\left\{\frac{j^k - i^k + 1}{j^k - b}(q_{i^k - 1} - q_b), \frac{j^k - i^k + 1}{b - i^k + 1}(q_b - q_{j^k + 1})\right\}, & i^k > 1 \\ \frac{j}{b}(q_b - q_{j^k + 1}), & i^k = 1 \end{cases} \tag{31}$$

  Update q_i:

$$q_i = \begin{cases} q_i - r_k, & i \in [0, i^k - 1], \\ q_i - \frac{b - i^k + 1}{j^k - i^k + 1} r_k, & i \in [i^k, j^k] \end{cases} \tag{32}$$

  Terminate if $q = 0$, and set $m = k$
end for
Select $K \in [m]$ randomly with discrete distribution $(r_1, \ldots, r_m)$
Choose $b - i^K + 1$ coordinates uniformly at random from i^K to j^K, denote it by W
$S = \{1, \ldots, i^K - 1\} \cup W$

Figure 1 is a sample illustration of Algorithm 3, where we have a marginal distribution for 4 coordinates given by $(0.8, 0.6, 0.4, 0.2)^T$ and we set the batchsize to be $b = 2$. Then, the algorithm is run and finds r to be $(0.2, 0.4, 0.4)^T$. Afterwards, with probability $r_1 = 0.2$, we will sample 2-coordinates from $(1, 2)$. With probability $r_2 = 0.4$, we will sample 2-coordinates which has (1) for sure and the other coordinate is chosen from $(2, 3)$ uniformly at random and with probability $r_3 = 0.4$, we will sample 2-coordinates from $(1, 2, 3, 4)$ uniformly at random.

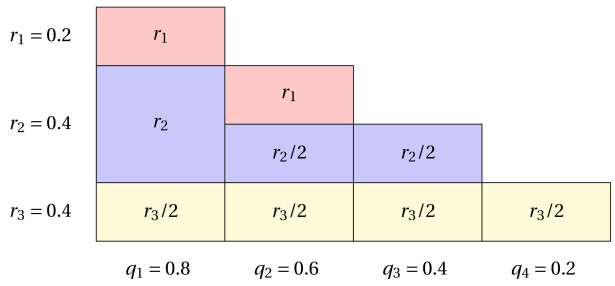

FIGURE 1 | Toy demo illustrating how to obtain a non-uniform mini-batch sampling with batch size $b = 2$ from $n = 4$ coordinates.

Note that, here we only need to perform two kinds of operations. The first one is to sample a single coordinate from distribution d (see section 5.1), and the second is to sample batches from a uniform distribution [see for example [26]].

5.3. Mini-Batch adfSDCA Algorithm

Here we describe a new adfSDCA algorithm that uses a mini-batch scheme. The algorithm is called mini-batch adfSDCA and is presented below as Algorithm 4.

Algorithm 4 Mini-Batch adfSDCA

Input: Data: $\{x_i, \phi_i\}_{i=1}^n$
Initialization: Choose $\alpha^{(0)} \in \mathbb{R}^n$ and set batchsize b
for $t = 0, 1, 2, \ldots$ **do**
  Calculate dual residue $\kappa_i^{(t)} = \phi_i'(x_i^T w^{(t)}) + \alpha_i^{(t)}$, for all $i \in [n]$
  Generate the adaptive probability distribution $p^{(t)} \sim \kappa^{(t)}$
  Choose mini-batch $S \subset [n]$ of size b according to probabilities distribution $p^{(t)}$
  Set step-size $\theta^{(t)} \in (0, 1)$ as in (76)
  for $i \in S$ **do**
    Update: $\alpha_i^{(t+1)} = \alpha_i^{(t)} - \theta^{(t)}(bp_i^{(t)})^{-1}\kappa_i^{(t)}$
  end for
  Update: $w^{(t+1)} = w^{(t)} - \sum_{i\in S}\theta^{(t)}(n\lambda bp_i^{(t)})^{-1}\kappa_i^{(t)}x_i$
end for

Briefly, Algorithm 4 works as follows. At iteration t, adaptive probabilities are generated in the same way as for Algorithm 1. Then, instead of updating only one coordinate, a mini-batch S of size $b \geq 1$ is chosen that is consistent with the adaptive probabilities. Next, the dual variables $\alpha_i^{(t)}, i \in S$ are updated, and finally the primal variable w is updated according to the primal-dual relation (4).

In the next section we will provide a convergence guarantee for Algorithm 4. As was discussed in section 3, theoretical results are detailed under two different assumptions on the type of loss function: (i) all loss function are convex; and (ii) individual loss functions may be non-convex but the average over all loss functions is convex.

5.4. Expected Separable Overapproximation

Here we make use of the Expected Separable Overapproximation (ESO) theory introduced in [26] and further extended, for example, in Richtárik and Takáč [29]. The ESO definition is stated below.

Definition 4 (Expected Separable Overapproximation, [29]). Let $\hat{S}$ be a sampling with marginal distribution $q = (q_1, \cdots, q_n)^T$. Then we say that the function f admits a v-ESO with respect to the sampling $\hat{S}$ if $\forall x, h \in \mathbb{R}^n$, we have $v_1, \ldots, v_n > 0$, such that the following inequality holds $\mathbf{E}[f(x + h_{[\hat{S}]})] \leq f(x) + \sum_{i=1}^n q_i(\nabla_i f(x)h_i + \frac{1}{2}v_i h_i^2)$.

REMARK 1. *Note that, here we do not assume that $\hat{S}$ is a uniform sampling, i.e., we do not assume that $q_i = q_j$ for all $i, j \in [n]$.*

The ESO inequality is useful in this work because the parameter v plays an important role when setting a suitable stepsize θ in our algorithm. Consequently, this also influences our complexity result, which depends on the sampling $\hat{S}$. For the proof of Theorem 4 (which will be stated in next subsection), the following is useful. Let $f(x) = \frac{1}{2}\|Ax\|^2$, where $A = (x_1, \ldots, x_n)$. We say that $f(x)$ admits a v-ESO if the following inequality holds

$$\mathbf{E}[\|Ah_{\hat{S}}\|^2] \leq \sum_{i=1}^n v_i q_i h_i^2. \tag{33}$$

To derive the parameter v we will make use of the following theorem.

THEOREM 3 ([29]). *Let f satisfy the following assumption $f(x + h) \leq f(x) + \langle \nabla f(x), h\rangle + \frac{1}{2}h^T A^T A h^T$, where A is some matrix. Then, for a given sampling $\hat{S}$, f admits a v-ESO, where v is defined by $v_i = \min\{\lambda'(\mathbf{P}(\hat{S})), \lambda'(A^T A)\}\sum_{j=1}^m A_{ji}^2, i \in [n]$.*

Here $\mathbf{P}(\hat{S})$ is called a sampling matrix [see [26]] where element p_{ij} is defined to be $p_{ij} = \sum_{\{i,j\}\in S, S\in\hat{S}}P(S)$. For any matrix M, $\lambda'(M)$ denotes the maximal regularized eigenvalue of M, i.e., $\lambda'(M) = \max_{\|h\|=1}\{h^T Mh : \sum_{i=1}^n M_{ii}h_i^2 \leq 1\}$. We may now apply Theorem 3 because $f(x) = \frac{1}{2}\|Ax\|^2$ satisfies its assumption. Note that in our mini-batch setting, we have $P_{S\in\hat{S}}(|S| = b) = 1$, so we obtain $\lambda'(\mathbf{P}(\hat{S})) \leq b$ [Theorem 4.1 in [29]]. In terms of $\lambda'(A^T A)$, note that $\lambda'(A^T A) = \lambda'(\sum_{j=1}^m x_j x_j^T) \leq \max_j \lambda'(x_j x_j^T) = \max_j |J_j|$, where $|J_j|$ is number of non-zero elements of x_j for each j. Then, a conservative choice from Theorem 3 that satisfies (33) is

$$v_i' = \min\{b, \max_j |J_j|\}\|x_i\|^2, \qquad i \in [n]. \tag{34}$$

Now we are ready to give our complexity result for mini-batch adfSDCA (Algorithm 4). Note that we use the same notation as that established in section 3 and we also define

$$Q' \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n v_i'. \tag{35}$$

THEOREM **4.** *Let $\tilde{L}$, $\kappa_i^{(t)}$, γ $D^{(t)}$, v_i', C_0, and Q' be as defined in (3), (5), (10), (11), (34), (19), and (35), respectively. Suppose that ϕ_i is L-smooth and convex for all $i \in [n]$. Then, at every iteration $t \geq 0$ of Algorithm 4, run with batchsize b we have*

$$\mathbf{E}[D^{(t+1)}|\alpha^{(t)}] \leq (1-\theta^*)D^{(t)}, \tag{36}$$

where $\theta^* = \frac{n\lambda^2 b}{\sum_{i=1}^n (v_i'\gamma + n\lambda^2)}$. *Moreover, it follows that whenever*

$$T \geq \left(\frac{n}{b} + \frac{\tilde{L}Q'}{b\lambda}\right)\log\left(\frac{(\lambda+\tilde{L})C_0}{\lambda\tilde{L}\epsilon}\right), \tag{37}$$

we have that $\mathbf{E}[P(w^{(T)} - P(w^*))] \leq \epsilon$.

It is also possible to derive a complexity result in the case when the *average* of the *n* loss functions is convex. The theorem is stated now.

THEOREM **5.** *Let L, $\kappa_i^{(t)}$, $\bar{\gamma}$ $\bar{D}^{(t)}$, v_i', $\bar{C}_0$, and Q' be as defined in (3), (5), (24), (25), (34), (26), and (35), respectively. Suppose that every $\phi_i, i \in [n]$ is L_i-smooth and that the average of the n loss functions $\frac{1}{n}\sum_{i=1}^n \phi_i(w^T x_i)$ is convex. Then, at every iteration $t \geq 0$ of Algorithm 4, run with batchsize b, we have*

$$\mathbf{E}[\bar{D}^{(t+1)}|\alpha^{(t)}] \leq (1-\theta^*)\bar{D}^{(t)}, \tag{38}$$

where $\theta^* = \frac{n\lambda^2 b}{\sum_{i=1}^n (v_i'\bar{\gamma} + n\lambda^2)}$. *Moreover, it follows that whenever*

$$T \geq \left(\frac{n}{b} + \frac{Q'\frac{1}{n}\sum_{i=1}^n L_i^2}{b\lambda}\right)\log\left(\frac{(\lambda+\tilde{L})\bar{C}_0}{\bar{\gamma}\epsilon}\right), \tag{39}$$

we have that $\mathbf{E}[P(w^{(T)}) - P(w^*)] \leq \epsilon$.

These theorems show that in worst case (by setting $b = 1$), this mini-batch scheme shares the same complexity performance as the serial adfSDCA approach (recall section 2). However, when the batch-size b is larger, Algorithm 4 converges in fewer iterations. This behavior will be confirmed computationally in the numerical results given in section 6.

6. NUMERICAL EXPERIMENTS

Here we present numerical experiments to demonstrate the practical performance of the adfSDCA algorithm. Throughout these experiments we used two loss functions, quadratic loss $\phi_i(w^T x_i) = \frac{1}{2}(w^T x_i - y_i)^2$ and logistic loss $\phi_i(w^T x_i) = \log(1 + \exp(-y_i w^T x_i))$. Note that these two losses have Lipschitz gradient. The regularization parameter λ in (P) is set to be $1/\sqrt{n}$, where n is the number of samples of the dataset. The experiments were run using datasets from the standard library of test problems [see [30] and http://www.csie.ntu.edu.tw/~cjlin/libsvm], as summarized in **Table 1**.

6.1. Comparison for a Variety of adfSDCA Approaches

In this section we compare the adfSDCA algorithm (Algorithm 1) with both dfSCDA, which is a uniform variant of adfSDCA described in Shalev-Shwartz [24], and also with Prox-SDCA from Shalev-Shwartz and Zhang [31]. We also report results using Algorithm 2, which is a heuristic version of adfSDCA, used with several different shrinkage parameters.

Figure 2 compares the evolution of the duality gap for the standard and heuristic variant of our adfSDCA algorithm with the two state-of-the-art algorithms dfSDCA and Prox-SDCA. For these problems both our algorithm variants out-perform the dfSDCA and Prox-SDCA algorithms. Note that this is consistent with our convergence analysis (recall section 3). Now consider the adfSDCA+ algorithm, which was tested using the parameter values $s = 1, 10, 20$. It is clear that adfSDCA+ with $s = 1$ shows the worst performance, which is reasonable because in this case the algorithm only updates the sampling probabilities after each epoch; it is still better than dfSDCA since it utilizes the sub-optimality at the beginning of each epoch. On the other hand, there does not appear to be an obvious difference between adfSDCA+ used with $s = 10$ or $s = 20$ with both variants performing similarly. We see that adfSDCA performs the best overall in terms of the number of passes through the data. However, in practice, even though adfSDCA+ may need more passes through the data to obtain the same sub-optimality as adfSDCA, it requires less computational effort than adfSDCA.

In **Figure 3**, we compare SGD, SVRG, dfSDCA, and our proposed adfSDCA(+) algorithm in terms of the number of passes through the data and total running time. For the SGD and SVRG algorithms, the duality gap is not directly computable. Hence, in this numerical experiment, the relative primal objective value $P(w) - P(\hat{w})$ is used as the stopping condition, where $\hat{w}$ is the optimal weight given by the best run among all algorithms. The SGD algorithm is implemented using the same set-up as in Shalev-Shwartz and Zhang [32], where a diminishing step-size is used, and SVRG is implemented following Johnson and Zhang [3].

We remark that for the SVRG algorithm, the user must tune its two hyper-parameters, namely, the number of iterations in the inner loop, and the step-size. Proper tuning of these hyper-parameters is essential to get the best performance from the SVRG algorithm. In this experiment, we tuned the hyper-parameters for SVRG, and we used SVRG+ to denote the best performing SVRG variant, and we use m to denote the

TABLE 1 | The datasets used in the numerical experiments, see Chang and Lin [30].

Dataset	#Samples	#Features	#Classes	Sparsity (%)
mushrooms	8,124	112	2	18.8
ijcnn1	49,990	22	2	59.1
rcv1	20,242	47,237	2	0.16
news20	19,996	1,355,191	2	0.034

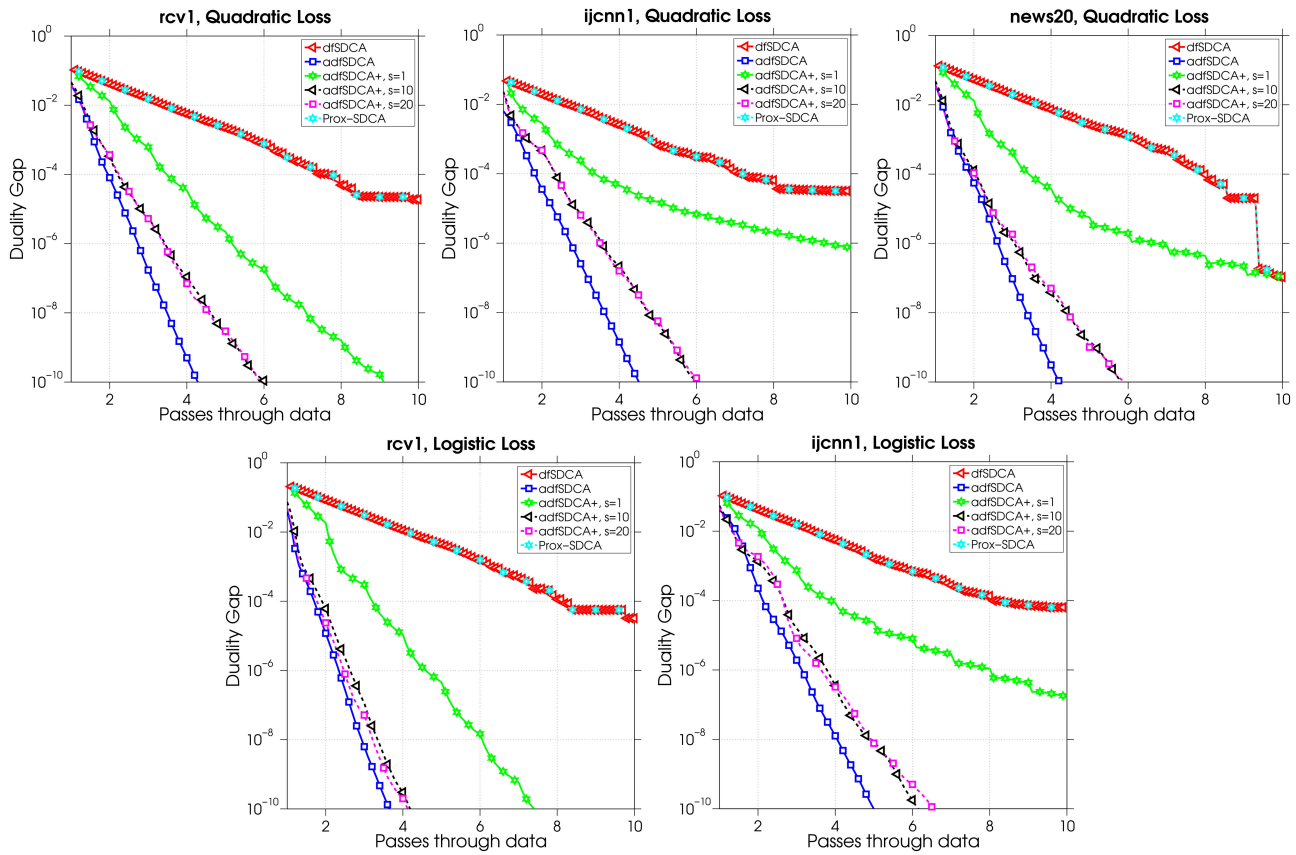

FIGURE 2 | A comparison of the number of epochs vs. the duality gap for the various algorithms.

corresponding "best" number of inner loop iterations. As a means of comparison, we also plot the performance of the SVRG algorithm using $m/2$ and 2 m inner loop iterations (i.e., SVRG without optimal tuning).

Figure 3 shows that, for the `rcv1` dataset with a quadratic loss, adfSDCA is the best performing algorithm in terms of the number of passes through the data; it is even better than the "best" tuned SVRG algorithm. For the `ijcnn1` dataset with a quadratic loss, SVRG+, the optimally tuned SVRG algorithm, performs better than the adfSDCA algorithm. However, tuning the hyper-parameters for SVRG is not free, and this is a computational cost that is not required for adfSDCA. This highlights one of the benefits of adfSDCA, which does not require parameter tuning, and the specific step-size needed is given explicitly in Theorem 1.

We also present plots showing the total running time for these algorithms. We follow the set up in Csiba et al. [15], and present the running time results using the heuristic algorithm adfSDCA+ with the shrinkage parameter set to $s = 5$ (see section 4). Recall that the `rcv1` dataset has $n = 20,242$ and $d = 47,237$, so the number of samples is comparable to the number of features. For this experiment, **Figure 3** shows that the total running time needed for adfSDCA+ is much less than SVRG. However, for the `ijcnn1` dataset, SVRG outperforms adfSDCA+ in terms of running time. To gain some insight into why this is happening, recall that the `ijcnn1` dataset has $n = 49,990$ and $d = 22$, so the number of samples is *much more* than the number of features. Note that adfSDCA+ must compute the residuals for each coordinate at every iteration, and because the number of samples is far greater than the number of feature, there is a high running time overhead for this non-uniform sampling of coordinates for adfSDCA+. This suggests that it is beneficial to use adfSDCA when the number of features is comparable with the number of samples.

Figure 4 shows the estimated density function of the dual residue $|\kappa^{(t)}|$ after $1, 2, 3, 4$, and 5 epochs for both uniform dfSDCA and our adaptive adfSDCA. One observes that the adaptive scheme is pushing the large residuals toward zero much faster than uniform dfSDCA. For example, notice that after 2 epochs, almost all residuals are below 0.03 for adfSDCA, whereas for uniform dfSDCA there are still many residuals larger than 0.06. This is evidence that, by using adaptive probabilities we are able to update the coordinate with a high dual residue more often and therefore reduce the sub-optimality much more efficiently.

6.2. Mini-Batch adfSDCA

Here we investigate the behavior of the mini-batch adfSDCA algorithm (Algorithm 4). In particular, we compare the practical

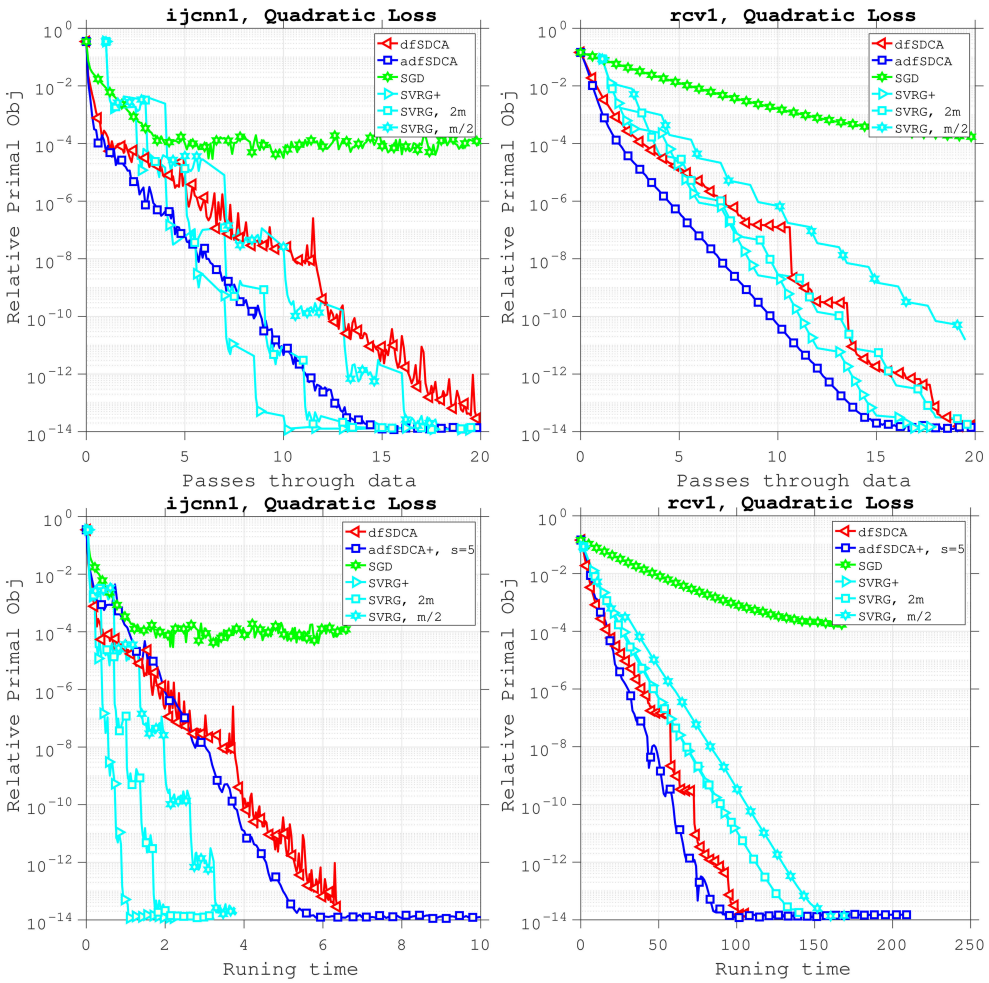

FIGURE 3 | A comparison of the number of epochs versus the relative primal object value for SGD, dfSDCA, adfSDCA(+) and SVRG. SVRG+ denotes the parameter-tuned, best performing SVRG algorithm, where m denotes the corresponding number of inner loop iterations. We also show results for the SVRG agorithm using both $m/2$ and $2m$ inner loop iterations, to demonstrate the performance of SVRG without optimal tuning.

performance of mini-batch adfSDCA using different mini-batch sizes b varying from 1 to 32. Note that if $b = 1$, then Algorithm 4 is equivalent to the adfSDCA algorithm (Algorithm 1). **Figure 5** shows that, with respect to the different batch sizes, the mini-batch algorithm with each batch size needs roughly the same number of passes through the data to achieve the same sub-optimality. However, when considering the computational time, the larger the batch size is, the faster the convergence will be. Recall that the results in section 5 show that the number of iterations needed by Algorithm 4 used with a batch size of b is roughly $1/b$ times the number of iterations needed by adfSDCA. Here we compute the adaptive probabilities every b samples, which leads to roughly the same number of passes through the data to achieve the same sub-optimality.

6.3. adfSDCA for Non-convex Losses

Here we investigate the behavior of adfSDCA when applied to problems that involve some nonconvex loss functions. We describe the experimental set-up now. Suppose that we have convex loss functions $\phi_i(x_i^T w)$, where $i \in [n]$. Then, it is possible to construct nonconvex loss functions by subtracting a quadratic from each of the convex losses as follows:

$$\bar{\phi}_i(x_i^T w) = \phi_i(x_i^T w) - C_i\|w\|^2. \tag{40}$$

Note that if $C_i > 0$ is large enough (up to the Lipschitz gradient constant of $\phi_i(x_i^T w)$), the new loss $\bar{\phi}_i(x_i^T w)$ derived by (40) will be nonconvex. On the other hand, if $C_i < 0$, we will have the new loss being strongly convex.

Now, functions of the form (40) will satisfy the requirements of Case II in section 3.2 (i.e., that the individual loss functions can be nonconvex, but that the average over all the losses is convex) as long as some of the hyperparameters C_i are large enough to make (40) nonconvex and $\sum_{i=1}^n C_i = 0$. Using this set-up, we present a numerical experiment to show the practical performance of adfSDCA. The quadratic loss is applied in this experiment due to

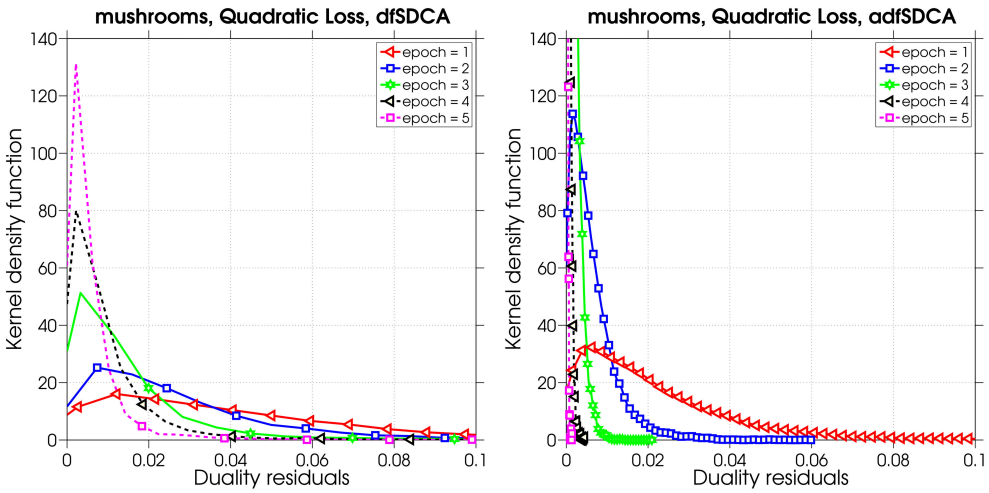

FIGURE 4 | Comparing the absolute value of the dual residuals for dfSDCA and adfSDCA over five epochs.

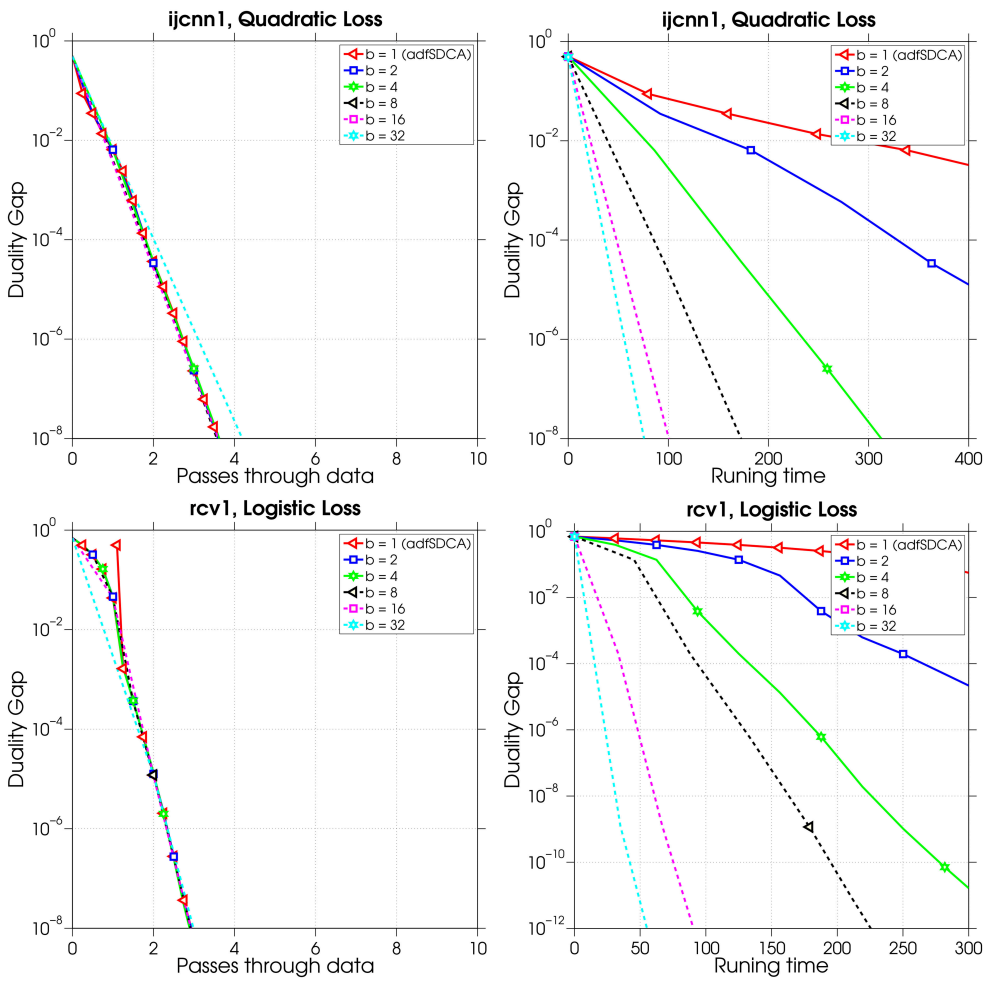

FIGURE 5 | Plots showing the performance of adfSDCA in terms of the number of passes through the data and running time, on different loss functions, as the batch size varies.

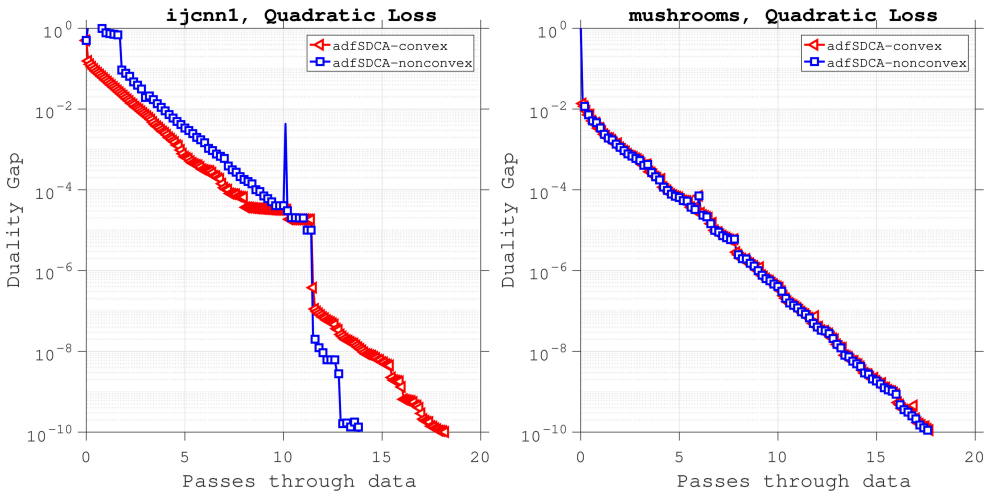

FIGURE 6 | Investigating the performance of adfSDCA when every loss function is convex, and when only the average of the loss functions is convex.

that the new loss (40) would be nonconvex when $C_i > 0$, since the Hessian of each quadratic loss of x_i has the smallest eigenvalue 0. In particular, we let $C_i = 0.01 \times (-1)^i$, where $i \in [n]$. We use the *mushrooms* and *ijcnn1* datasets for this experiment, and because these datasets both have an even number of samples, the property that $\sum_{i=1}^{n} C_i = 0$ will hold. The results of this experiment are shown in **Figure 6**, where we compare the performance of adfSDCA with respect to the running time and number of passes over the data. **Figure 6** shows that adfSDCA performs well on such problems and is able to find an accurate solution (where the duality gap is less than 10^{-10}) in less than 20 passes over the data.

7. CONCLUSION

In this work, we present dual free SDCA variants with adaptive probabilities for Empirical Risk Minimization problems. The theoretical complexity of the proposed methods is analyzed in two cases: when the individual loss functions are all convex and when the average over the losses is convex but individual loss functions may be nonconvex. A heuristic variant of adfSDCA is proposed to reduce the computational effort required and its practical convergence performance is demonstrated via a numerical experiment. We also extend our convergence theory to cover a mini-batch adfSDCA variant and a novel nonuniform sampling strategy for mini-batches is developed. Our experimental results show speedups in terms of the number of passes through the data and/or running time of the proposed methods, when compared with the original dual free SDCA, as well as other state-of-art primal methods. The numerical experiments related to the use of mini-batches match our theoretical analysis and suggest that using mini-batches is beneficial in practice.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

We would like to thank Professor Alexander L. Stolyar for his insightful help with Algorithm 3. The material is based upon work supported by the U.S. National Science Foundation, under award number NSF:CCF:1618717, NSF:CMMI:1663256 and NSF:CCF:1 740796.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fams.2018.00033/full#supplementary-material

REFERENCES

1. Shalev-Shwartz, S (2016). "SDCA without duality, regularization, and individual convexity," in *Proceedings of The 33rd International Conference on Machine Learning, Proceedings of Machine Learning Research*, Vol. 48, eds M. F. Balcan and K. Q. Weinberger (New York, NY: PMLR), 747–54.
2. Shalev-Shwartz S, Singer Y, Srebro N, Cotter A. Pegasos: Primal estimated sub-gradient solver for SVM. *Math Programm.* (2011) **127**:3–30. doi: 10.1007/s10107-010-0420-4
3. Johnson R, Zhang T. Accelerating stochastic gradient descent using predictive variance reduction. In: *Advances in Neural Information Processing Systems*. Lake Tahoe, NV (2013). p. 315–23.

4. Nitanda A. Stochastic proximal gradient descent with acceleration techniques. In: *Advances in Neural Information Processing Systems*. Montréal, QC (2014). p. 1574–82.
5. Konečný J, Liu J, Richtárik P, Takáč M. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE J Select Top Signal Process*. (2016) **10**:242–55. doi: 10.1109/JSTSP.2015.2505682
6. Schmidt M, Roux NL, Bach F. Minimizing finite sums with the stochastic average gradient. *Math Programm*. (2017) **162**:83–112. doi: 10.1007/s10107-016-1030-6
7. Defazio A, Bach F, Lacoste-Julien S. SAGA: a fast incremental gradient method with support for non-strongly convex composite objectives. In: *Advances in Neural Information Processing Systems*. Montréal, QC (2014). p. 1646–54.
8. Roux NL, Schmidt M, Bach FR. A stochastic gradient method with an exponential convergence rate for finite training sets. In: *Advances in Neural Information Processing Systems*. Lake Tahoe, NV (2012). p. 2663–71.
9. Hsieh CJ, Chang KW, Lin CJ, Keerthi SS, Sundararajan S. A dual coordinate descent method for large-scale linear SVM. In: *Proceedings of the 25th International Conference on Machine Learning*. Helsinki: ACM (2008). p. 408–15.
10. Takáč M, Bijral A, Richtárik P, Srebro N. Mini-batch primal and dual methods for SVMs. In: *Proceedings of the 30th International Conference on Machine Learning*. Atlanta, GA (2013). p. 1022–30. Available online at: http://proceedings.mlr.press/v28/
11. Jaggi M, Smith V, Takáč M, Terhorst J, Krishnan S, Hofmann T, et al. Communication-efficient distributed dual coordinate ascent. In: *Advances in Neural Information Processing Systems*. Montréal, QC (2014). p. 3068–76.
12. Ma C, Smith V, Jaggi M, Jordan MI, Richtárik P, Takáč M. Adding vs. averaging in distributed primal-dual optimization. In: *32th International Conference on Machine Learning, ICML 2015*. Lille (2015). **37:**1973–82 Available online at: http://proceedings.mlr.press/v37/
13. Takáč M, Richtárik P, Srebro N. Distributed Mini-Batch SDCA. *arXiv:150708322* [preprint]. (2015).
14. Qu Z, Richtárik P, Zhang T. Quartz: randomized dual coordinate ascent with arbitrary sampling. In: *Advances in Neural Information Processing Systems*. Montréal, QC (2015). p. 865–73.
15. Csiba D, Qu Z, Richtárik P. Stochastic dual coordinate ascent with adaptive probabilities. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. Lille (2015). p. 674–83. Available online at: http://proceedings.mlr.press/v37/
16. Zhang Y, Xiao L, JMLR. org. DiSCO: distributed optimization for self-concordant empirical loss. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML-15)*. Lille (2015). p. 362–70 Available online at: http://proceedings.mlr.press/v37/
17. Nesterov Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J Optim*. (2012) **22**:341–62. doi: 10.1137/100802001
18. Richtárik P, Takáč M. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math Programm*. (2014) **144**:1–38. doi: 10.1007/s10107-012-0614-z
19. Tappenden R, Takáč M, Richtárik P. On the complexity of parallel coordinate descent. *Optim Methods Softw*. (2017) **33**:372–95. doi: 10.1080/10556788.2017.1392517
20. Necoara I, Clipici D. Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: application to distributed MPC. *J Process Control* (2013) **23**:243–53. doi: 10.1016/j.jprocont.2012.12.012
21. Necoara I, Clipici D. Parallel random coordinate descent method for composite minimization. *SIAM J Optim*. (2016) **26**:197–226. doi: 10.1137/130950288
22. Liu J, Wright SJ. Asynchronous stochastic coordinate descent: parallelism and convergence properties. *SIAM J Optim*. (2015) **25**:351–76. doi: 10.1137/140961134
23. Zhao P, Zhang T. Stochastic optimization with importance sampling for regularized loss minimization. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* . Lille (2015). p. 1–9. Available online at: http://proceedings.mlr.press/v37/
24. Shalev-Shwartz S. *SDCA Without Duality. arXiv:150206177* [preprint]. (2015).
25. Csiba D, Richtárik P. Primal method for ERM with flexible mini-batching schemes and non-convex losses. *arXiv:150602227* [preprint]. (2015).
26. Richtárik P, Takáč M. Parallel coordinate descent methods for big data optimization. *Math Programm*. (2016) **156:**443–84. doi: 10.1007/s10107-015-0901-6
27. Konečný J, Richtárik P. Semi-stochastic gradient descent methods. *Front Appl Math Stat*. (2017) **3**:9. doi: 10.3389/fams.2017.00009
28. Kronmal RA, Peterson AV Jr. On the alias method for generating random variables from a discrete distribution. *Am Stat*. (1979) **33**:214–8.
29. Qu Z, Richtárik P. Coordinate descent with arbitrary sampling II: expected separable overapproximation. *Optim Methods Softw*. (2016) **31**:858–84. doi: 10.1080/10556788.2016.1190361
30. Chang CC, Lin CJ. LIBSVM: a library for support vector machines. *ACM Trans Intell Syst Technol*. (2011) **2**:27. doi: 10.1145/1961189.1961199
31. Shalev-Shwartz S, Zhang T. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Math Programm*. (2016) **155**:105–45. doi: 10.1007/s10107-014-0839-0
32. Shalev-Shwartz S, Zhang T. Stochastic dual coordinate ascent methods for regularized loss. *J Mach Learn Res*. (2013) **14**:567–99. Available online at: http://www.jmlr.org/papers/v14/shalev-shwartz13a.html
33. Shalev-Shwartz S, Ben-David S. *Understanding Machine Learning: From Theory to Algorithms*. New York, NY: Cambridge University Press (2014).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.