Accelerating Greedy Coordinate Descent Methods

Haihao Lu [1] **Robert M. Freund** [2] **Vahab Mirrokni** [3]

Abstract

We introduce and study two algorithms to accelerate greedy coordinate descent in theory and in practice: Accelerated Semi-Greedy Coordinate Descent (ASCD) and Accelerated Greedy Coordinate Descent (AGCD). On the theory side, our main results are for ASCD: we show that ASCD achieves $O(1/k^2)$ convergence, and it also achieves accelerated linear convergence for strongly convex functions. On the empirical side, while both AGCD and ASCD outperform Accelerated Randomized Coordinate Descent on most instances in our numerical experiments, we note that AGCD significantly outperforms the other two methods in our experiments, in spite of a lack of theoretical guarantees for this method. To complement this empirical finding for AGCD, we present an explanation why standard proof techniques for acceleration cannot work for AGCD, and we introduce a technical condition under which AGCD is guaranteed to have accelerated convergence. Finally, we confirm that this technical condition holds in our numerical experiments.

1. Introduction

Coordinate descent methods have received much-deserved attention recently due to their capability for solving large-scale optimization problems (with sparsity) that arise in machine learning applications and elsewhere. With inexpensive updates at each iteration, coordinate descent algorithms obtain faster running times than similar full gradient descent algorithms in order to reach the same near-optimality tolerance; indeed some of these algorithms now comprise the state-of-the-art in machine learning algorithms for loss minimization.

Most recent research on coordinate descent has focused on versions of randomized coordinate descent, which can essentially recover the same results (in expectation) as full gradient descent, including obtaining "accelerated" (i.e., $O(1/k^2)$) convergence guarantees. On the other hand, in some important machine learning applications, greedy coordinate methods demonstrate superior numerical performance while also delivering much sparser solutions. For example, greedy coordinate descent is one of the fastest algorithms for the graphical LASSO implemented in DP-GLASSO (Mazumder & Hastie, 2012). And sequence minimization optimization (SMO) (a variant of greedy coordinate descent) is widely known as the best solver for kernel SVM (Joachims, 1999)(Platt, 1999) and is implemented in LIBSVM and SVMLight.

In general, for smooth convex optimization the standard first-order methods converge at a rate of $O(1/k)$ (including greedy coordinate descent). In 1983, Nesterov (Nesterov, 1983) proposed an algorithm that achieved a rate of $O(1/k^2)$ – which can be shown to be the optimal rate achievable by any first-order method (Nemirovsky & Yudin, 1983). This method (and other similar methods) is now referred to as Accelerated Gradient Descent (AGD).

However, there has not been much work on accelerating the standard Greedy Coordinate Descent (GCD) due to the inherent difficulty in demonstrating $O(1/k^2)$ computational guarantees (we discuss this difficulty further in Section 4.1). The only work that might be close as far as the authors are aware is (Song et al., 2017), which updates the z-sequence using the full gradient and thus should not be considered as a coordinate descent method in the standard sense.

In this paper, we study ways to accelerate greedy coordinate descent in theory and in practice. We introduce and study two algorithms: Accelerated Semi-Greedy Coordinate Descent (ASCD) and Accelerated Greedy Coordinate Descent (AGCD). While ASCD takes greedy steps in the x-updates and randomized steps in the z-updates, AGCD is a straightforward extension of GCD that only takes greedy steps. On the theory side, our main results are for ASCD: we show that ASCD achieves $O(1/k^2)$ convergence, and it also achieves accelerated linear convergence when the objective function is furthermore strongly convex. However, a direct extension of convergence proofs for ARCD does not work for ASCD due to the different coordinates we use to update x-sequence and z-sequence. Thus, we present a new

[1]Department of Mathematics and Operations Research Center, MIT [2]Sloan School of Management, MIT [3]Google Research. Correspondence to: Haihao Lu <haihao@mit.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

proof technique – which shows that a greedy coordinate step yields a better objective function value than a full gradient step with a modified smoothness condition.

On the empirical side, we first note that in most of our experiments ASCD outperforms Accelerated Randomized Coordinate Descent (ARCD) in terms of running time. On the other hand, we note that AGCD significantly outperforms the other accelerated coordinate descent methods in all instances, in spite of a lack of theoretical guarantees for this method. To complement the empirical study of AGCD, we present a Lyapunov energy function argument that points to an explanation for why a direct extension of the proof for AGCD does not work. This argument inspires us to introduce a technical condition under which AGCD is guaranteed to converge at an accelerated rate. Interestingly, we confirm that the technical condition holds in a variety of instances in our empirical study, which in turn justifies our empirical observation that AGCD works so well in practice.

1.1. Related Work

Coordinate Descent. Coordinate descent methods have a long history in optimization, and convergence of these methods has been extensively studied in the optimization community in the 1980s-90s, see (Bertsekas & Tsitsiklis, 1989), (Luo & Tseng, 1992), and (Luo & Tseng, 1993). There are roughly three types of coordinate descent methods depending on how the coordinate is chosen: randomized coordinate descent (RCD), cyclic coordinate descent (CCD), and greedy coordinate descent (GCD). RCD has received much attention since the seminal paper of Nesterov (Nesterov, 2012). In RCD, the coordinate is chosen randomly from a certain fixed distribution. (Richtarik & Takac, 2014) provides an excellent review of theoretical results for RCD. CCD chooses the coordinate in a cyclic order, see (Beck & Tetruashvili, 2013) for basic convergence results. More recent results show that CCD is inferior to RCD in the worst case (Sun & Ye, 2016), while it is better than RCD in certain situations (Gurbuzbalaban et al., 2017). In GCD, we select the coordinate yielding the largest reduction in the objective function value. GCD usually delivers better function values at each iteration in practice, though this comes at the expense of having to compute the full gradient in order to select the gradient coordinate with largest magnitude. The recent work (Nutini et al., 2015) shows that GCD has faster convergence than RCD in theory, and also provides several applications in machine learning where the full gradient can be computed cheaply. A parallel GCD method is proposed in (You et al., 2016) and numerical results show its advantage in practice.

Accelerated Randomized Coordinate Descent. Since Nesterov's paper on RCD (Nesterov, 2012) there has been significant focus on accelerated versions of RCD. (Nesterov, 2012) developed the first accelerated randomized coordinate gradient method. (Lu & Xiao, 2015) present a sharper convergence analysis of Nesterov's method using a randomized estimate sequence framework. (Fercoq & Richtarik, 2015) proposed the APPROX (Accelerated, Parallel and PROXimal) coordinate descent method and obtained an accelerated sublinear convergence rate, and (Lee & Sidford, 2013) developed an efficient implementation of ARCD.

1.2. Accelerated Coordinate Descent Framework

Our optimization problem of interest is:

$$P : f^* := \text{ minimum}_x \ f(x) \,, \tag{1}$$

where $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function.

Definition 1.1. *$f(\cdot)$ is coordinate-wise L-smooth for the vector of parameters $L := (L_1, L_2, \ldots, L_n)$ if $\nabla f(\cdot)$ is coordinate-wise Lipschitz continuous for the corresponding coefficients of L, i.e., for all $x \in \mathbb{R}^n$ and $h \in \mathbb{R}$ it holds that:*

$$|\nabla_i f(x + he_i) - \nabla_i f(x)| \leq L_i |h| \,, \ i = 1, \ldots, n \,, \tag{2}$$

where $\nabla_i f(\cdot)$ denotes the i^{th} coordinate of $\nabla f(\cdot)$ and e_i is i^{th} unit coordinate vector, for $i = 1, \ldots, n$.

We presume throughout that $L_i > 0$ for $i = 1, \ldots, n$. Let $\mathbf{L}$ denote the diagonal matrix whose diagonal coefficients correspond to the respective coefficients of L. Let $\langle \cdot, \cdot \rangle$ denote the standard coordinate inner product in $\mathbb{R}^n$, namely $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$, and let $\| \cdot \|_p$ denote the ℓ_p norm for $1 \leq p \leq \infty$. Let $\langle x, y \rangle_L := \sum_{i=1}^n L_i x_i y_i = \langle x, \mathbf{L}y \rangle = \langle \mathbf{L}x, y \rangle$ denote the L-inner product. Define the norm $\|x\|_L := \sqrt{\langle x, \mathbf{L}x \rangle}$. Letting $\mathbf{L}^{-1}$ denote the inverse of $\mathbf{L}$, we will also use the norm $\| \cdot \|_{L^{-1}}$ defined by $\|v\|_{L^{-1}} := \sqrt{\langle v, \mathbf{L}^{-1} v \rangle} = \sqrt{\sum_{i=1}^n \mathbf{L}_i^{-1} v_i^2}$.

Algorithm 1 presents a generic framework for accelerated coordinate descent methods that is flexible enough to encompass deterministic as well as randomized methods. One specific case is the standard Accelerated Randomized Coordinate Descent (ARCD). In this paper we propose and study two other cases. The first is Accelerated Greedy Coordinate Descent (AGCD), which is a straightforward extension of greedy coordinate descent to the acceleration framework and which, surprisingly, has not been previously studied (that we are aware of). The second is a new algorithm which we call Accelerated Semi-Greedy Coordinate Descent (ASCD) that takes greedy steps in the x-updates and randomized steps in the z-update.

Algorithm 1 Accelerated Coordinate Descent Framework without Strong Convexity

Input: Objective function f with coordinate-wise smoothness parameter L, initial point $z^0 = x^0$, parameter sequence $\{\theta_k\}$ defined as follow: $\theta_0 = 1$, and define θ_k recursively by the relationship $\frac{1-\theta_k}{\theta_k^2} = \frac{1}{\theta_{k-1}^2}$ for $k = 1, 2, \ldots$.

for $k = 0, 1, 2, \ldots$ **do**

Define $y^k := (1-\theta_k)x^k + \theta_k z^k$

Choose coordinate j_k^1 (by some rule)

Compute $x^{k+1} := y^k - \frac{1}{L_{j_k^1}} \nabla_{j_k^1} f(y^k) e_{j_k^1}$

Choose coordinate j_k^2 (by some rule)

Compute $z^{k+1} := z^k - \frac{1}{nL_{j_k^2}\theta_k} \nabla_{j_k^2} f(y^k) e_{j_k^2}$

end for

In the framework of Algorithm 1 we choose a coordinate j_k^1 of the gradient $\nabla f(y^k)$ to perform the update of the x-sequence, and we choose (a possibly different) coordinate j_k^2 of the gradient $\nabla f(y^k)$ to perform the update of the z-sequence. Herein we will study three different rules for choosing the coordinates j_k^1 and j_k^2 which then define three different specific algorithms:

- ARCD (Accelerated Randomized Coordinate Descent): use the rule

$$j_k^2 = j_k^1 :\sim \mathcal{U}[1, \cdots, n] \tag{3}$$

- AGCD (Accelerated Greedy Coordinate Descent): use the rule

$$j_k^2 = j_k^1 := \arg\max_i \tfrac{1}{\sqrt{L_i}} |\nabla_i f(y^k)| \tag{4}$$

- ASCD (Accelerated Semi-Greedy Coordinate Descent): use the rule

$$\begin{array}{l} j_k^1 := \arg\max_i \tfrac{1}{\sqrt{L_i}} |\nabla_i f(y^k)| \\ j_k^2 :\sim \mathcal{U}[1, \cdots, n] \, . \end{array} \tag{5}$$

In ARCD a random coordinate j_k^1 is chosen at each iteration k, and this coordinate is used to update both the x-sequence and the z-sequence. ARCD is well studied, and is known to have the following convergence guarantee in expectation (see (Fercoq & Richtarik, 2015) for details):

$$E\left[f(x^k) - f(x^*)\right] \leq \tfrac{2n^2}{(k+1)^2} \|x^* - x^0\|_L^2 \, , \tag{6}$$

where the expectation is on the random variables used to define the first k iterations.

In AGCD we choose the coordinate j_k^1 in a "greedy" fashion, i.e., corresponding to the maximal (weighted) absolute value coordinate of the the gradient $\nabla f(y^k)$. This greedy coordinate is used to update both the x-sequence and the z-sequence. As far as we know AGCD has not appeared in the first-order method literature. One reason for this is that while AGCD is the natural accelerated version of greedy coordinate descent, the standard proof methodologies for establishing acceleration guarantees (i.e., $O(1/k^2)$ convergence) fail for AGCD. Nevertheless, we show in Section 5 that AGCD is extremely effective in numerical experiments on synthetic linear regression problems as well as on practical logistic regression problems, and dominates other coordinate descent methods in terms of numerical performance. Furthermore, we observe that AGCD attains $O(1/k^2)$ convergence (or better) on these problems in practice. Thus AGCD is worthy of significant further study, both computationally as well as theoretically. Indeed, in Section 4 we will present a technical condition that implies $O(1/k^2)$ convergence when satisfied, and we will argue that this condition ought to be satisfied in many settings.

ASCD, which we consider to be the new theoretical contribution of this paper, combines the salient features of AGCD and ARCD. In ASCD we choose the greedy coordinate of the gradient to perform the x-update, while we choose a random coordinate to perform the z-update. In this way we achieve the practical advantage of greedy x-updates, while still guaranteeing $O(1/k^2)$ convergence in expectation by virtue of choosing the random coordinate used in the z-update, see Theorem 2.1. And under strong convexity, ASCD achieves accelerated linear convergence as will be shown in Section 3.

The paper is organized as follows. In Section 2 we present the $O(1/k^2)$ convergence guarantee for ASCD. In Section 3 we present an extension of the accelerated coordinate descent framework to the case of strongly convex functions, and we present the associated linear convergence guarantee for ASCD under strong convexity. In Section 4 we study AGCD; we present a Lyapunov energy function argument that points to why standard analysis of accelerated gradient descent methods fails in the analysis of AGCD. In Section 4.2 we present a technical condition under which AGCD achieves $O(1/k^2)$ convergence. In Section 5, we present results of our numerical experiments using AGCD and ASCD on synthetic linear regression problems as well as practical logistic regression problems.

2. Accelerated Semi-Greedy Coordinate Descent (ASCD)

In this section we present our computational guarantee for the Accelerated Semi-Greedy Coordinate Descent (ASCD) method in the non-strongly convex case, which is Algorithm 1 with rule (5). At each iteration k the ASCD method chooses the greedy coordinate j_k^1 to do the x-update, and chooses a randomized coordinate $j_k^2 \sim \mathcal{U}[1, \cdots, n]$ to do

the z-update. Unlike ARCD where the same randomized coordinate is used in both the x-update and the z-update – in ASCD j_k^1 is chosen in a deterministic greedy way, j_k^1 and j_k^2 are likely to be different.

At each iteration k of ASCD the random variable j_k^2 is introduced, and therefore x^k depends on the realization of the random variable

$$\xi_k := \{j_0^2, \ldots, j_{k-1}^2\} .$$

For convenience we also define $\xi_0 := \emptyset$.

The following theorem presents our computational guarantee for ASCD for the non-strongly convex case:

Theorem 2.1. *Consider the Accelerated Semi-Greedy Coordinate Descent method (Algorithm 1 with rule (5)). If $f(\cdot)$ is coordinate-wise L-smooth, it holds for all $k \geq 1$ that:*

$$E_{\xi_k}\left[f(x^k) - f(x^*)\right] \leq \frac{2n^2}{(k+1)^2}\|x^* - x^0\|_L^2 . \quad (7)$$

In the interest conveying some intuition on proofs of accelerated methods in general, we will present the proof of Theorem 2.1 after first stating some intermediary results along with some explanatory comments. The proofs for these results can be found in supplementary materials.

We start with the Three-Point Property (Tseng, 2008). Given a differentiable convex function $h(\cdot)$, the Bregman distance for $h(\cdot)$ is $D_h(y,x) := h(y) - h(x) - \langle \nabla h(x), y - x\rangle$. The Three-Point property can be stated as follows:

Lemma 2.1. (Three-Point Property (Tseng, 2008)) *Let $\phi(\cdot)$ be a convex function, and let $D_h(\cdot,\cdot)$ be the Bregman distance for $h(\cdot)$. For a given vector z, let*

$$z^+ := \arg\min_{x \in \mathbb{R}^n} \{\phi(x) + D_h(x,z)\} .$$

Then for all $x \in \mathbb{R}^n$,

$$\phi(x) + D_h(x,z) \geq \phi(z^+) + D_h(z^+,z) + D_h(x,z^+)$$

with equality holding in the case when $\phi(\cdot)$ is a linear function and $h(\cdot)$ is a quadratic function.

It follows from integration and the coordinate Lipschitz condition (2) that for all $x \in \mathbb{R}^n$ and $h \in \mathbb{R}$:

$$f(x + he_i) \leq f(x) + h \cdot \nabla_i f(x) + \frac{h^2 L_i}{2} . \quad (8)$$

At each iteration $k = 0, 1, \ldots$ of ASCD, notice that x^{k+1} is one step of greedy coordinate descent from y^k in the norm $\|\cdot\|_L$. Now define $s^{k+1} := y^k - \frac{1}{n}\mathbf{L}^{-1}\nabla f(y^k)$, which is a full steepest-descent step from y^k in the norm $\|\cdot\|_{nL}$. We first establish that the greedy coordinate descent step yields a good objective function value as compared to the quadratic model that yields s^{k+1}.

Lemma 2.2.

$$f(x^{k+1}) \leq f(y^k) + \langle \nabla f(y^k), s^{k+1} - y^k\rangle + \frac{n}{2}\|s^{k+1} - y^k\|_L^2 .$$

Utilizing the interpretation of s^{k+1} as a gradient descent step from y^k but with a larger smoothness descriptor (nL as opposed to L), we can invoke the standard proof for accelerated gradient descent derived in (Tseng, 2008) for example. We define $t^{k+1} := z^k - \frac{1}{n\theta_k}\mathbf{L}^{-1}\nabla f(y^k)$, or equivalently we can define t^{k+1} by:

$$t^{k+1} = \arg\min_z \langle \nabla f(y^k), z - z^k\rangle + \frac{n\theta_k}{2}\|z - z^k\|_L^2 \quad (9)$$

(which corresponds to z^{k+1} in (Tseng, 2008) for standard accelerated gradient descent). Then we have:

Lemma 2.3.

$$f(x^{k+1}) \leq (1-\theta_k)f(x^k) + \theta_k f(x^*) + \frac{n\theta_k^2}{2}\|x^* - z^k\|_L^2 - \frac{n\theta_k^2}{2}\|x^* - t^{k+1}\|_L^2 . \quad (10)$$

Notice that t^{k+1} is an all-coordinate update of z^k, and computing t^{k+1} can be very expensive. Instead we will use z^{k+1} to replace t^{k+1} in (10) by using the equality in the next lemma.

Lemma 2.4.

$$\begin{aligned}&\frac{n}{2}\|x^* - z^k\|_L^2 - \frac{n}{2}\|x^* - t^{k+1}\|_L^2 \\ =&\frac{n^2}{2}\|x^* - z^k\|_L^2 - \frac{n^2}{2}E_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right] .\end{aligned} \quad (11)$$

We now have all the ingredients needed to prove Theorem 2.1.

Proof of Theorem 2.1 Substituting (11) into (10), we obtain:

$$\begin{aligned}f(x^{k+1}) \leq&(1-\theta_k)f(x^k) + \theta_k f(x^*) + \frac{n^2\theta_k^2}{2}\|x^* - z^k\|_L^2 \\ &- \frac{n^2\theta_k^2}{2}E_{j_k^2}\left[\|x^* - z^{k+1}\|_L^2\right] .\end{aligned}$$

Rearranging and substituting $\frac{1-\theta_{k+1}}{\theta_{k+1}^2} = \frac{1}{\theta_k^2}$ yields:

$$\begin{aligned}&\frac{1-\theta_{k+1}}{\theta_{k+1}^2}\left(f(x^{k+1}) - f(x^*)\right) + \frac{n^2}{2}E_{j_k^2}\left\|x^* - z^{k+1}\right\|_L^2 \\ \leq&\left[\frac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right) + \frac{n^2}{2}\left\|x^* - z^k\right\|_L^2\right] .\end{aligned}$$

Taking the expectation over the random variables $j_1^2, j_2^2, \ldots, j_k^2$, it follows that:

$$\begin{aligned}&E_{\xi_{k+1}}\left[\frac{1-\theta_{k+1}}{\theta_{k+1}^2}\left(f(x^{k+1}) - f(x^*)\right) + \frac{n^2}{2}\left\|x^* - z^{k+1}\right\|_L^2\right] \\ \leq&E_{\xi_k}\left[\frac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right) + \frac{n^2}{2}\left\|x^* - z^k\right\|_L^2\right] .\end{aligned}$$

Algorithm 2 Accelerated Coordinate Descent Framework (μ-strongly convex case)

Input: Objective function $f(\cdot)$ with known smoothness parameter L and strongly convex parameter μ, initial point $z^0 = x^0$, parameters $a = \frac{\sqrt{\mu}}{n+\sqrt{\mu}}$ and $b = \frac{\mu a}{n^2}$.
for $k = 0, 1, 2, \ldots$ **do**
  Define $y^k = (1-a)x^k + az^k$
  Choose j_k^1 (by some rule)
  Compute $x^{k+1} = y^k - \frac{1}{L_{j_k^1}} \nabla_{j_k^1} f(y^k) e_{j_k^1}$
  Compute $u^k = \frac{a^2}{a^2+b} z^k + \frac{b}{a^2+b} y^k$
  Choose j_k^2 (by some rule)
  Compute $z^{k+1} = u^k - \frac{a}{a^2+b} \frac{1}{nL_{j_k^2}} \nabla f_{j_k^2}(y^k) e_{j_k^2}$
end for

Applying the above inequality in a telescoping manner for $k = 1, 2, \ldots$, yields:

$$\begin{aligned}
&E_{\xi_k}\left[\tfrac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right)\right] \\
\leq &E_{\xi_k}\left[\tfrac{1-\theta_k}{\theta_k^2}\left(f(x^k) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^k\right\|_L^2\right] \\
&\vdots \\
\leq &E_{\xi_0}\left[\tfrac{1-\theta_0}{\theta_0^2}\left(f(x^0) - f(x^*)\right) + \tfrac{n^2}{2}\left\|x^* - z^0\right\|_L^2\right] \\
= &\tfrac{n^2}{2}\left\|x^* - x^0\right\|_L^2 .
\end{aligned}$$

Note from an induction argument that $\theta_i \leq \frac{2}{i+2}$ for all $i = 0, 1, \ldots$, whereby the above inequality rearranges to:

$$\begin{aligned}
E_{\xi_k}\left[\left(f(x^k) - f(x^*)\right)\right] \leq &\tfrac{\theta_k^2}{1-\theta_k}\tfrac{n^2}{2}\left\|x^* - x^0\right\|_L^2 \\
\leq &\tfrac{2n^2}{(k+1)^2}\left\|x^* - x^0\right\|_L^2 . \quad \square
\end{aligned}$$

3. Accelerated Coordinate Descent Framework under Strong Convexity

We begin with the definition of strong convexity as developed in (Lu & Xiao, 2015):

Definition 3.1. *$f(\cdot)$ is μ-strongly convex with respect to $\|\cdot\|_L$ if for all $x, y \in \mathbb{R}^n$ it holds that:*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \tfrac{\mu}{2}\|y - x\|_L^2 .$$

Note that μ can be viewed as an extension of the condition number of $f(\cdot)$ in the traditional sense since μ is defined relative to the coordinate smoothness coefficients through $\|\cdot\|_L$, see (Lu & Xiao, 2015). Algorithm 2 presents the generic framework for accelerated coordinate descent methods when $f(\cdot)$ is μ-strongly convex for known μ.

Just as in the non-strongly convex case, we extend the three algorithms ARCD, AGCD, and ASCD to the strongly convex case by using the rules (3), (4), and (5) in Algorithm 2. The following theorem presents our computational guarantee for ASCD for the strongly convex case:

Theorem 3.1. *Consider the Accelerated Semi-Greedy Coordinate Descent method for the strongly convex case (Algorithm 2 with rule (5)). If $f(\cdot)$ is coordinate-wise L-smooth and μ-strongly convex with respect to $\|\cdot\|_L$, it holds for all $k \geq 1$ that:*

$$\begin{aligned}
&E_{\xi_k}\left[f(x^k) - f^* + \tfrac{n^2}{2}(a^2+b)\|z^k - x^*\|_L^2\right] \\
&\leq \left(1 - \tfrac{\sqrt{\mu}}{n+\sqrt{\mu}}\right)^k \left(f(x^0) - f^* + \tfrac{n^2}{2}(a^2+b)\|x^0 - x^*\|_L^2\right).
\end{aligned} \tag{12}$$

We provide a concise proof of Theorem 3.1 in the supplementary materials.

4. Accelerated Greedy Coordinate Descent

In this section we discuss accelerated greedy coordinate descent (AGCD), which is Algorithm 1 with rule (4). In the interest of clarity we limit our discussion to the non-strongly convex case. We present a Lyapunov function argument which shows why the standard type of proof of accelerated gradient methods fails for AGCD, and we propose a technical condition under which AGCD is guaranteed to have an $O(1/k^2)$ accelerated convergence rate. Although there are no guarantees that the technical condition will hold for a given function $f(\cdot)$, we provide intuition as to why the technical condition ought to hold in most cases.

4.1. Why AGCD fails (in theory)

The mainstream research community's interest in Nesterov's accelerated method (Nesterov, 1983) started around 15 years ago; and yet even today most researchers struggle to find basic intuition as to what is really going on in accelerated methods. Indeed, Nesterov's estimation sequence proof technique seems to work out arithmetically but with little fundamental intuition. There are many recent works trying to explain this acceleration phenomenon (Su et al., 2016)(Wilson et al., 2016)(Hu & Lessard, 2017)(Lin et al., 2015)(Frostig et al., 2015)(Allen-Zhu & Orecchia, 2014) (Bubeck et al., 2015). A line of recent work has attempted to give a physical explanation of acceleration techniques by studying the continuous-time interpretation of accelerated gradient descent via dynamical systems, see (Su et al., 2016), (Wilson et al., 2016), and (Hu & Lessard, 2017). In particular, (Su et al., 2016) introduced the continuous-time dynamical system model for accelerated gradient descent,

and presented a convergence analysis using a Lyapunov energy function in the continuous-time setting. (Wilson et al., 2016) studied discretizations of the continuous-time dynamical system, and also showed that Nesterov's estimation sequence analysis is equivalent to the Lyapunov energy function analysis in the dynamical system in the discrete-time setting. And (Hu & Lessard, 2017) presented an energy dissipation argument from control theory for understanding accelerated gradient descent.

In the discrete-time setting, one can construct a Lyapunov energy function of the form (Wilson et al., 2016):

$$E_k = A_k(f(x^k) - f^*) + \tfrac{1}{2}\|x^* - z^k\|_L^2 \ , \qquad (13)$$

where A_k is a parameter sequence with $A_k \sim O(k^2)$, and one shows that E_k is nonincreasing in k, yielding:

$$f(x^k) - f^* \leq \frac{E_k}{A_k} \leq \frac{E_0}{A_k} \sim O\left(\frac{1}{k^2}\right) \ .$$

The earlier proof techniques of acceleration methods such as (Nesterov, 1983) and (Tseng, 2008), as well as the recent proof techniques for accelerated randomized coordinate descent (such as (Nesterov, 2012), (Lu & Xiao, 2015), and (Fercoq & Richtarik, 2015)) can all be rewritten in the above form (up to expectation) each with slightly different parameter sequences $\{A_k\}$.

Now let us return to accelerated greedy coordinate descent. Let us assume for simplicity that $L_1 = \cdots = L_n$ (as we can always do rescaling to achieve this condition). Then the greedy coordinate j_k^1 is chosen as the coordinate of the gradient with the largest magnitude, which corresponds to the coordinate yielding the greatest guaranteed decrease in the objective function value. However, in the proof of acceleration using the Lyapunov energy function, one needs to prove a decrease in E_k (13) instead of a decrease in the objective function value $f(x^k)$. The coordinate j_k^1 is not necessarily the greedy coordinate for decreasing the energy function E_k due to the presence of the second term $\|x^* - z^k\|_L^2$ in (13). This explains why the greedy coordinate can fail to decrease E_k, at least in theory. And because x^* is not known when running AGCD, there does not seem to be any way to find the greedy descent coordinate for the energy function E_k.

That is why in ASCD we use the greedy coordinate to perform the x-update (which corresponds to the fastest coordinate-wise decrease for the first term in energy function), while we choose a random coordinate to perform the z-update (which corresponds to the second term in the energy function); thereby mitigating the above problem in the case of ASCD.

4.2. How to make AGCD work (in theory)

Here we propose the following technical condition under which the proof of acceleration of AGCD can be made to work.

Technical Condition 4.1. *There exists a positive constant γ and an iteration number K such that for all $k \geq K$ it holds that:*

$$\sum_{i=0}^{k} \frac{1}{\theta_i}\langle \nabla f(y^i), z^i - x^* \rangle \leq \sum_{i=0}^{k} \frac{n\gamma}{\theta_i} \nabla_{j_i} f(y^i)(z_{j_i}^i - x_{j_i}^*) \ , \qquad (14)$$

where $j_i = \arg\max_i \frac{1}{\sqrt{L_i}}|\nabla_i f(y^k)|$ is the greedy coordinate at iteration i.

One can show that this condition is sufficient to prove an accelerated convergence rate $O(1/k^2)$ for AGCD. Therefore let us take a close look at Technical Condition 4.1. The condition considers the weighted sum (with weights $\frac{1}{\theta_i} \sim O(i^2)$) of the inner product of $\nabla f(y^k)$ and $z^k - x^*$, and the condition states that the inner product corresponding to the greedy coordinate (the right side above) is larger than the average of all coordinates in the inner product, by a factor of γ. In the case of ARCD and ASCD, it is easy to show that Technical Condition 4.1 holds automatically up to expectation, with $\gamma = 1$.

Here is an informal explanation of why Technical Condition 4.1 ought to hold for most convex functions and most iterations of AGCD. When k is sufficiently large, the three sequence $\{x^k\}$, $\{y^k\}$ and $\{z^k\}$ ought to all converge to x^* (which always is observed in practice though not justified by theory), whereby z^k is close to y^k. Thus we can instead consider the inner product $\langle \nabla f(y^k), y^k - x^* \rangle$ in (14). Notice that for any coordinate j it holds that $|y_j^k - x_j^*| \geq \frac{1}{L_j}|\nabla_j f(y^k)|$, and therefore $|\nabla_j f(y^k) \cdot (y_j^k - x_j^*)| \geq \frac{1}{L_j}|\nabla_j f(j^k)|^2$. Now the greedy coordinate is chosen by $j_i := \arg\max_j \frac{1}{L_j}|\nabla_j f(j^k)|^2$, and therefore it is reasonably likely that in most cases the greedy coordinate will yield a better product than the average of the components of the inner product.

The above is not a rigorous argument, and we can likely design some worst-case functions for which Technical Condition 4.1 fails. But the above argument provides some intuition as to why the condition ought to hold in most cases, thereby yielding the observed improvement of AGCD as compared with ARCD that we will shortly present in Section 5, where we also observe that Technical Condition 4.1 holds empirically on all of our problem instances.

With a slight change in the proof of Theorem 2.1, we can show the following result:

Theorem 4.1. *Consider the Accelerated Greedy Coordinate*

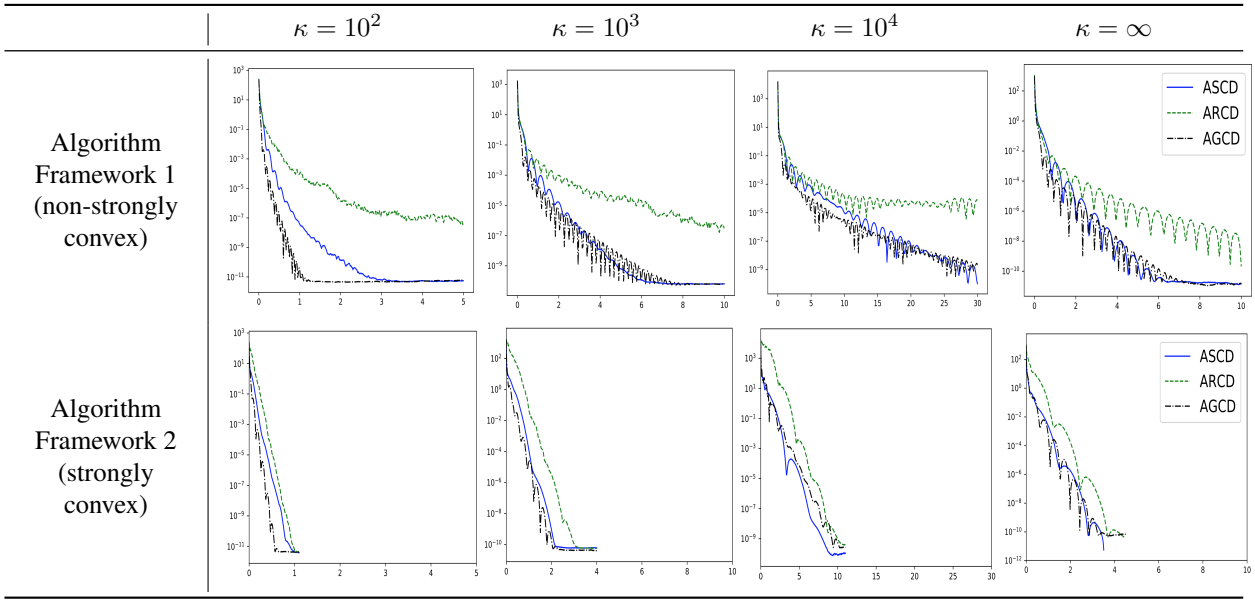

Figure 1. Plots showing the optimality gap versus run-time (in seconds) for synthetic linear regression problems solved by ASCD, ARCD and AGCD.

Descent (Algorithm 1 with rule (4)). If $f(\cdot)$ is coordinate-wise L-smooth and satisfies Technical Condition 4.1 with constant $\gamma \leq 1$ and iteration number K, then it holds for all $k \geq K$ that:

$$f(x^k) - f(x^*) \leq \frac{2n^2\gamma}{(k+1)^2}\|x^* - x^0\|_L^2 \ . \qquad (15)$$

We note that if $\gamma < 1$ (which we always observe in practice), then AGCD will have a better convergence guarantee than ARCD.

Remark 4.1. *The arguments in Section 4.1 and Section 4.2 also work for strongly convex case, albeit with suitable minor modifications.*

5. Numerical Experiments

5.1. Linear Regression

We consider solving synthetic instances of the linear regression model with least-squares objective function:

$$f^* := \min_{\beta \in \mathbb{R}^p} f(\beta) := \|y - X\beta\|_2^2$$

using ASCD, ARCD and AGCD, where the mechanism for generating the data (y, X) and the algorithm implementation details are described in the supplementary materials. Figure 1 shows the optimality gap versus time (in seconds) for solving different instances of linear regression with different condition numbers of the matrix X^TX using ASCD, ARCD and AGCD. In each plot, the vertical axis is the objective value optimality gap $f(\beta^k) - f^*$ in log scale, and the horizontal axis is the running time in seconds. Each column corresponds to an instance with the prescribed condition number κ of X^TX, where $\kappa = \infty$ means that the minimum eigenvalue of X^TX is 0. The first row of plots is for Algorithm Framework 1 which is ignorant of any strong convexity information. The second row of plots is for Algorithm Framework 2, which uses given strong convexity information. And because the linear regression optimization problem is quadratic, it is straightforward to compute κ as well as the true parameter μ for the instances where $\kappa > 0$. The last column of the figure corresponds to $\kappa = \infty$, wherein we set μ using the smallest positive eigenvalue of X^TX, which can be shown to work in theory since all relevant problem computations are invariant in the nullspace of X.

Here we see in Figure 1 that AGCD and ASCD consistently have superior performance over ARCD for both the non-strongly convex case and the strongly convex case, with ASCD performing almost as well as AGCD in most instances.

We remark that the behavior of any convex function near the optimal solution is similar to the quadratic function defined by the Hessian at the optimum, and therefore the above numerical experiments show promise that AGCD and ASCD are likely to outperform ARCD asymptotically for any twice-differentiable convex function.

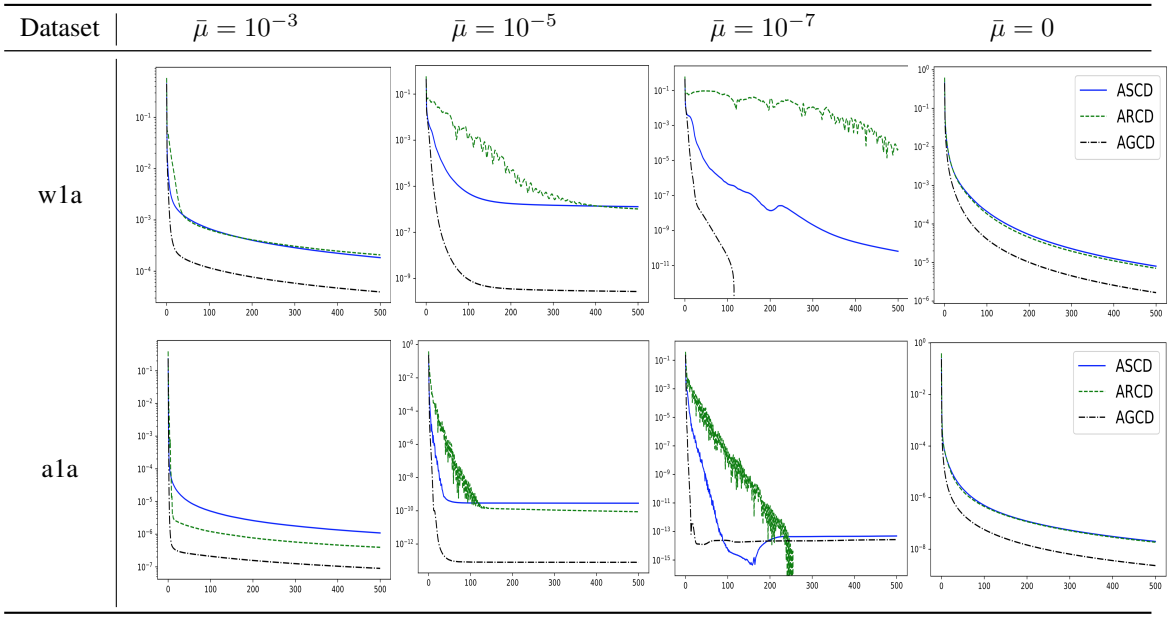

Figure 2. Plots showing the optimality gap versus run-time (in seconds) for the logistic regression instances w1a and a1a in LIBSVM, solved by ASCD, ARCD and AGCD.

5.2. Logistic Regression

Here we consider solving instances of the logistic regression loss minimization problem:

$$f^* := \min_{\beta \in \mathbb{R}^p} f(\beta) := \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \beta^T x_i)) ,$$

using ASCD, ARCD and AGCD, where $\{x_i, y_i\}$ is the feature-response pair for the i-th data point and $y_i \in \{-1, 1\}$. Although the loss function $f(\beta)$ is not in general strongly convex, it is essentially locally strongly convex around the optimum but with unknown local strong convexity parameter $\bar{\mu}$. And although we do not know the local strong convexity parameter $\bar{\mu}$, we can still run the strongly convex algorithm (Algorithm Framework 2) by assigning a value of $\bar{\mu}$ that is hopefully close to the actual value. Using this strategy, we solved a large number of logistic regression instances from LIBSVM (Chang & Lin, 2011). Figure 2 shows the optimality gap versus time (in seconds) for solving two of these instances, namely w1a and a1a; we present similar comparisons for more datasets in the supplementary materials. In each plot, the vertical axis is the objective value optimality gap $f(\beta^k) - f^*$ in log scale, and the horizontal axis is the running time in seconds. Each column corresponds to a different assigned value of the local strong convexity parameter $\bar{\mu}$. The right-most column in the figure uses the assignment $\bar{\mu} = 0$, in which case the algorithms are implemented as in the non-strongly convex case (Algorithm Framework 1). The implementation details are described in the supplementary materials, which also contains plots of the optimality gaps versus the number of iterations.

Table 1. Largest observed values of γ for five different datasets in LIBSVM for $k \geq \bar{K} := 5000$.

Dataset:	w1a	a1a	heart	madelon	rcv1
γ:	0.25	0.17	0.413	0.24	0.016

Here we see in Figure 2 that AGCD always has superior performance as compared to both ASCD and ARCD. In the relevant range of optimality gaps ($\geq 10^{-9}$), ASCD typically outperforms ARCD for smaller values of the assigned strong convexity parameter $\bar{\mu}$. However, the performance of ASCD and ARCD are essentially the same when no strong convexity is presumed.

Last of all, we attempt to estimate the parameter γ that arises in Technical Condition 4.1 for AGCD in several of the datasets in SVMLIB. Although for small k, the ratio between $\sum_{i=0}^{k} \frac{1}{\theta_i} \langle \nabla f(y^i), z^i - x^* \rangle$ and $\sum_{i=0}^{k} \frac{n}{\theta_i} \nabla_{j_i} f(y^i)(z^i_{j_i} - x^*_{j_i})$ can fluctuate widely, this ratio stabilizes after a number of iterations in all of our numerical tests. From Technical Condition 4.1, we know that γ is the upper bound of such ratio for all $k \geq K$ for some large enough value of K. Table 1 presents the observed values of γ for all $K \geq \bar{K} := 5,000$. Recalling from Theorem 4.1 that the γ value represents how much better AGCD can perform compared with ARCD in terms of computational guarantees, we see from Table 1 that AGCD should outperform ARCD for these representative instances – and indeed this is what we observe in practice in our tests.

Acknowledgment

The authors thank Martin Jaggi for pointing out the related work (Locatello et al., 2018), whose connection to this paper is discussed in the supplementary materials. The authors are also grateful to the referees for their comprehensive efforts and their suggestions to improve the readability of the paper.

References

Allen-Zhu, Z. and Orecchia, L. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014.

Beck, A. and Tetruashvili, L. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

Bertsekas, D. and Tsitsiklis, J. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

Bubeck, S., Lee, Y. T., and Singh, M. A geometric alternative to nesterov's accelerated gradient descent. *arXiv preprint arXiv:1506.08187*, 2015.

Chang, C.-C. and Lin, C.-J. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Fercoq, O. and Richtarik, P. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

Frostig, R., Ge, R., Kakade, S., and Sidford, A. Unregularizing: approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, 2015.

Gurbuzbalaban, M., Ozdaglar, A., Parrilo, P. A., and Vanli, N. When cyclic coordinate descent outperforms randomized coordinate descent. In *Advances in Neural Information Processing Systems*, pp. 7002–7010, 2017.

Hu, B. and Lessard, L. Dissipativity theory for Nesterov's accelerated method. *arXiv preprint arXiv:1706.04381*, 2017.

Joachims, T. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pp. 169–184. MIT Press, Cambridge, MA, USA, 1999.

Lee, Y. T. and Sidford, A. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, FOCS '13, pp. 147–156, Washington, DC, USA, 2013. IEEE Computer Society.

Lin, H., Mairal, J., and Harchaoui, Z. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, 2015.

Locatello, F., Raj, A., Reddy, S. P., Rätsch, G., Schölkopf, B., Stich, S. U., and Jaggi, M. On matching pursuit and coordinate descent. *ICML 2018 - Proceedings of the 35th International Conference on Machine Learning*, 2018.

Lu, Z. and Xiao, L. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152(1-2):615–642, 2015.

Luo, Z.-Q. and Tseng, P. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

Luo, Z.-Q. and Tseng, P. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

Mazumder, R. and Hastie, T. The graphical lasso: new insights and alternatives. *Electronic Journal of Statistics*, 6:2125, 2012.

Nemirovsky, A. S. and Yudin, D. B. *Problem Complexity and Method Efficiency in Optimization*. Wiley, New York, 1983.

Nesterov, Y. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pp. 372–376, 1983.

Nesterov, Y. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Nutini, J., Schmidt, M., Laradji, I., Friedlander, M., and Koepke, H. Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *International Conference on Machine Learning*, pp. 1632–1641, 2015.

Platt, J. C. Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pp. 185–208. MIT Press, Cambridge, MA, USA, 1999.

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

Song, C., Cui, S., Jiang, Y., and Xia, S.-T. Accelerated stochastic greedy coordinate descent by soft thresholding projection onto simplex. In *Advances in Neural Information Processing Systems*, pp. 4841–4850, 2017.

Su, W., Boyd, S., and Candes, E. J. A differential equation for modeling Nesterovs accelerated gradient method: theory and insights. *Journal of Machine Learning Research*, 17(153):1–43, 2016.

Sun, R. and Ye, Y. Worst-case complexity of cyclic coordinate descent: $O(n^2)$ gap with randomized version. *arXiv preprint arXiv:1604.07130*, 2016.

Tseng, P. On accelerated proximal gradient methods for convex-concave optimization. Technical report, May 21, 2008.

Wilson, A. C., Recht, B., and Jordan, M. I. A Lyapunov analysis of momentum methods in optimization. *arXiv preprint arXiv:1611.02635*, 2016.

You, Y., Lian, X., Liu, J., Yu, H.-F., Dhillon, I. S., Demmel, J., and Hsieh, C.-J. Asynchronous parallel greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pp. 4682–4690, 2016.