Greed Meets Sparsity: Understanding and Improving Greedy Coordinate Descent for Sparse Optimization

Huang Fang
UBC

Zhenan Fan
UBC

Yifan Sun
INRIA-Paris

Michael P. Friedlander
UBC

Abstract

We consider greedy coordinate descent (GCD) for composite problems with sparsity inducing regularizers, including ℓ_1 regularization and non-negative constraints. Empirical evidence strongly suggests that GCD, when started at an all-zero initialization, has an implicit screening ability; i.e., the selected coordinate at each iteration is often nonzero at the solution. Thus, GCD for problems with sparse solutions can converge significantly faster than randomized coordinate descent. We present an improved convergence analysis of GCD for sparse optimization and a formal analysis of its screening properties. We also propose and analyze an improved selection rule that has a stronger ability to produce sparse iterates. Numerical experiments on both synthetic and real-world data support our analysis and show improved performance with the new selection rule.

1 Introduction

Recent works on large-scale optimization [Nesterov, 2012, Shalev-Shwartz and Zhang, 2013, Zhang and Lin, 2015, Wright, 2015, Allen-Zhu et al., 2016] has brought about renewed interest in coordinate descent (CD) methods, due to their efficiency in handling optimization problems with a large number of variables. Greedy coordinate descent (GCD) methods choose a single coordinate at each iteration to maximize progress at that iteration. It has been observed in practice that GCD using the Gauss-Southwell rule (GS rule) with an all-zero initialization can converge to a sparse solution significantly faster than randomized coordinate descent (RCD), especially for high dimensional problems (e.g., LASSO [Hastie et al., 2008, Li and Osher, 2009, Nutini et al., 2015] and kernel SVM [Platt, 1999, Joachims, 1999, Chang and Lin, 2011]) and can sometimes get close to the optimum even before executing one full pass of all coordinates. This suggests that GCD has an inherent screening ability for sparse optimization, preferring to update coordinates at which the final solution is non-zero.

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

Our aim is twofold. First, we present a formal analysis to understand why GCD works well for problems with sparse solutions and an all-zero initialization. Secondly, we propose and analyze an improved selection rule for GCD, and show its connection with existing algorithms. Experiments on both real world and synthetic data illustrate our analysis and the effectiveness of the approach.

2 Related Work

2.1 Coordinate descent (CD)

Coordinate descent methods for optimization have a long history ([Southwell, 1940, Powell, 1973, Luo and Tseng, 1993, Bertsekas, 1999]; see [Wright, 2015] for a much more comprehensive survey). The idea is to reduce computational complexity by updating only one variable component at a time, either through complete minimization or a single gradient step on a single coordinate. These methods have been recognized for their superior empirical performance on many machine learning problems, including LASSO [Hastie et al., 2008], support vector machines (SVM) [Platt, 1999], non-negative matrix factorization [Cichocki and Phan, 2009], graph-based label propagation [Bengio et al., 2006]. Despite its wide use in practice, the convergence rate of CD was not clear until Nesterov's seminal work [Nesterov, 2012], which presented the first non-asymptotic convergence rate for RCD. Subsequent studies [Richtárik and Takác, 2014, Allen-Zhu et al., 2016, Nesterov and Stich, 2017] offered refined analyses on CD and its variants and explained why these methods are effective for many machine learning problems.

2.2 Greedy coordinate descent (GCD)

In practice, a greedy coordinate selection rule can greatly accelerate a CD method. There are many such update rules, including the maximum improvement rule (which maximizes descent in each step), the GS rule (picking the coordinate with the largest gradient magnitude), and more [Nutini et al., 2015]. However, previous theoretical works struggled to describe improved rates for greedy over randomized methods. More recently, Nutini *et al.* [Nutini et al., 2015] and Karimireddy *et al.* [Karimireddy et al., 2019] offered more refined analysis based on strong convexity with respect to the ℓ_1 norm that eliminates the rate dependency on problem dimension, which may be very large in practice. Our work primarily builds upon these works.

Implementation One practical challenge in using the GS rule is that it requires computing the maximum element of the full gradient, which in general requires a full gradient computation. Therefore the GS rule is most interesting in cases where the maximum gradient coordinate can be identified efficiently, this is possible by exploiting the special problem structure [Nutini et al., 2015] or using a maximum inner-product search (MIPS) algorithm to approximate the GS rule [Dhillon et al., 2011, Karimireddy et al., 2019]. Other notable improvements on CD implementation includes parallel or distributed variants [Liu et al., 2015, Richtárik and Takác, 2016] in multi-core or multi-machine environments. This line of works is orthogonal to the purpose of this paper and we do not include them into our discussion.

2.3 Sparsity pattern identification

A related line of work seeks to analyze the iteration complexity of identifying the final sparsity pattern [Dunn, 1987, Burke and Moré, 1988, Wright, 1993, Nutini et al., 2017b, Nutini et al., 2017a, Liang et al., 2017, Sun et al., 2019], etc. This problem is related to the screening ability studied in this work, and part our analysis is actually built on their techniques.

3 Problem statement

We consider the optimization problem

$$\min_{x\in\mathbb{R}^d} F(x) := f(x) + g(x)$$

where d is the number of variables, $f(x)$ is a smooth and convex function and $g(x) := \sum_{i=1}^d g_i(x_i)$ is a separable, convex, but not necessarily smooth function. More explicitly, we have the following assumptions.

Algorithm 1 A general template for GCD

```
Input: $f(\cdot), g_i(\cdot)\ \forall i \in [d]$.
$W_0 = \emptyset$
$x^0 = 0$
for $t = 0, 1, 2, ...$ do
    Coordinate selection: select $i$ according to rule
    Gradient step: $x^{t+\frac{1}{2}} = x^t - \frac{1}{L_i}\nabla f_i(x^t)e_i$
    Prox. step: $x^{t+1} = \mathrm{prox}_{\frac{1}{L_i}g_i}\left[x^{t+\frac{1}{2}}\right]$
    Optional post-processing (see e.g. (2))
    Update working set: $W_{t+1} = W_t \cup \{i\}$
end for
```

- $f(x + \alpha e_i)$ is L_i-smooth in terms of $\alpha\ \forall i \in [d]$:

  $$|\nabla_i f(x + de_i) - \nabla_i f(x)| \le L_i|d|\ \forall x \in \mathbb{R}^d,$$

  where e_i is the ith unit vector. Let $L := \max_{i\in[d]} L_i$. Note that L can be much smaller than the gradient Lipschitz constant in $\mathbb{R}^d$.

- $f(x)$ is L_∞-smooth with respect to $\|\cdot\|_\infty$:

  $$\|\nabla f(x) - \nabla f(y)\|_\infty \le L_\infty\|x - y\|_1\ \forall x, y \in \mathbb{R}^d.$$

- $f(x)$ is μ_p-strongly convex with respect to $\|\cdot\|_p$:

  $$f(x) \ge f(y) + \langle\nabla f(y), x-y\rangle + \frac{\mu_p}{2}\|x-y\|_p^2\ \forall x, y \in \mathbb{R}^d,$$

  where $p \in \{1, 2\}$.

- $g(x) = \lambda\|x\|_1$ the ℓ_1 norm or $g(x) = \delta_{\ge 0}(x)$ the indicator function for $x \ge 0$. However, intuitively we believe the results can be extended to more general cases, e.g. where $g_i(\cdot)$ are non-smooth at 0 for all $i \in [d]$.

A general template for GCD is presented in Algorithm 1. We consider GCD under the following rule:
Selection rule 1 (GS-s rule)**.** *Select* $i \in \arg\max_j Q_j(x^t)$ *where*

$$Q_i(x) = \min_{s\in\partial g_i} |\nabla_i f(x) + s|. \tag{1}$$

Here we present results with GS-s rule, but our analysis can easily be extended to simple coordinate rule generalizations, as in [Nutini et al., 2015, Bertsekas, 1999, Tseng and Yun, 2009, Dhillon et al., 2011].

An optional 'post-processing' step is proposed in [Karimireddy et al., 2019] and is useful to derive a convergence rate that depends on μ_1 instead of μ_2, which is essential in removing the dependency on the dimension d. Specifically, after each prox-gradient step, we set

$$x_i^{t+1} := 0 \text{ if } x_i^{t+1}x_i^t < 0. \tag{2}$$

We adopt this post-processing technique in the following sections. For simplicity, we assume that there is only one element in the set $\arg\max_j Q_j(x_t)$, which makes Algorithm 1 deterministic. Note that this assumption is easy to satisfy, for example by choosing lexicographically.

Definition 1. At iteration t, we define the *working set* W_t as the set of indices accrued thus far. Define also $W := \bigcup_{t=0}^{\infty} W_t$ as the working set overall.

Definition 2. We define the *support* of a vector x as $\text{supp}(x) = \{i \mid x_i \neq 0\}$.

Definition 3. [Rockafellar, 1970, §23] For a closed, convex function $g : \mathbb{R}^d \to \mathbb{R}$, the subdifferential of g at x is defined as

$$\partial g(x) := \{v \in \mathbb{R}^d \mid g(y) \geq g(x) + v^T(y - x),\ \forall y\}$$

The set ∂g is always closed and convex. In particular, for $g_i : \mathbb{R} \to \mathbb{R}$ convex, the set ∂g_i is a closed interval in $\mathbb{R}$; e.g.

- for $g(x) = \lambda\|x\|_1$, $\partial g_i(0) = [-\lambda, \lambda]$
- for $g(x) = \delta_{\geq 0}(x)$, $\partial g_i(0) = (-\infty, 0]$.

3.1 The merits of keeping the iterates sparse

Karimireddy *et al.* [Karimireddy et al., 2019] derived a linear convergence rate for strongly convex objective with l_1 regularization or non-negative constraint:

$$F(x^t) - F^* \leq \left(1 - \frac{\mu_1}{L}\right)^{\lceil \frac{t}{2} \rceil} \left(F(x^0) - F^*\right), \qquad (3)$$

where μ_1 is the strongly convex constant with respect to the l_1-norm and satisfies $\mu_1 \in [\mu_2/d, \mu_2]$. Such rates theoretically illustrate the advantage of GCD over randomized CD (see also [Nutini et al., 2015]), where in (3) the term μ_1/L is replaced with $\mu_2/(dL)$, and in particular the variable dimension d may be very large. A weakness of this rate is that it suggests that GCD applied to strongly convex composite problems with l_1 regularization or non-negative constraints has the same rate as it does for problems without regularizers i.e. solving only $\min f(x)$ instead of $\min f(x) + g(x)$. However, GCD solving the composite problem with sparsity inducing regularization is usually significantly faster than its non-sparse counterpart, and there is a need for an improved convergence analysis to explain this. One benefit from producing sparse iterates and keep the working set small is an improved convergence rate for GCD:

Theorem Sketch 2 (Improved convergence rate). *Denote x^t as the iterate generated from Algorithm 1 with selection rule 1. Then,*

$$F(x^t) - F^* \leq \left(1 - \frac{\tilde{\mu}}{L}\right)^{\lceil \frac{t}{2} \rceil} \left(F(x^0) - F^*\right).$$

where $\tilde{\mu}$ is a constant satisfies $\max\{\mu_2/|W|, \mu_1\} \leq \tilde{\mu} \leq \mu_2$, and in particular can be large when the working set is small.

In particular, if $|W| \ll d$, then this rate is much better than the rate in (3), especially when the constant $\mu_1 \approx \mu_2/d$.

In practice, there is the added benefit of less memory usage and faster matrix-vector multiplication by using a sparse data structure to store the iterates.

4 Analysis

In this section, we study the screening ability of GCD and present a bound on $|W|$. First we introduce an important quantity used in sparsity pattern identification [Hare and Lewis, 2007, Lewis and Wright, 2011, Nutini et al., 2017b, Sun et al., 2019]:

$$\delta_i := \min\left\{-\nabla_i f(x^*) - l_i, u_i + \nabla_i f(x^*)\right\}, \qquad (4)$$

where $\partial g_i(x_i^*) = [l_i, u_i]$ and $x^* = \arg\min F(x)$. The constant δ_i is closely related to the distance to the relative interior of the sparse manifold, and an important quantity in sparse manifold identification [Lewis and Wright, 2011]. Optimality conditions indictate that $\delta_i = 0$ if $x_i^* \neq 0$ and $\delta_i \geq 0$ if $x_i^* = 0$. Because x^* is unique, these quantities are problem-specific and algorithmically invariant. The definition in (4) leads to the following identification result, extracted from [Nutini et al., 2017b]:

Lemma 3. *If for some $t > 0$,*

$$|\nabla_i f(x^t) - \nabla_i f(x^*)| \leq \delta_i,$$

then after one coordinate proximal gradient step,

$$x_i^t = 0 \;\Rightarrow\; x_i^{t+1} = 0.$$

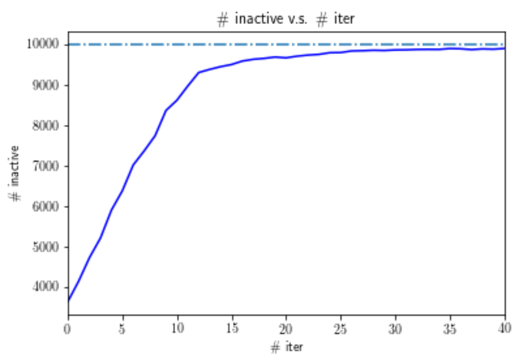
This Lemma suggests that if $\nabla_i f(x^t)$ is close to $\nabla_i f(x^*)$ and $x_i^* = 0$, then the i-th entry of x^t will be correctly identified as 0.

4.1 Numerical motivation

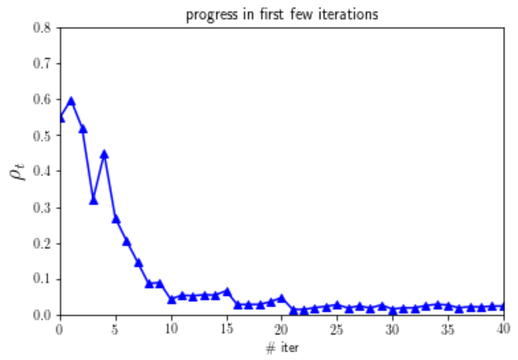
Our analysis approach is based on the following numerical observations, which we illustrate with a LASSO problem on random synthetic data. In particular,

$$F(x) = \|Ax - b\|_2^2/2 + \lambda\|x\|_1,$$

where $A \in \mathbb{R}^{50\times 10^4}$ and $b := Ax^\sharp + \epsilon$. The elements A_{ij}, ϵ_i, and nonzeros $x_i^\sharp$ are distributed as standard Gaussian and we randomly select 10 elements of $x^\sharp$ to be nonzero. Here, $\lambda = 2$.



(a) Number of inactive variables plot.



(b) Objective progress, where ρ_t is defined in Eq. (6).

Figure 1: Exploratory investigations.

Figures 1a and 1b show the evolution of the number of “inactive” variables and objective progress for Algorithm 1.

In Figure 1a, we define

$$\#\ \text{inactive} := \sum_{i=1}^{d} \mathbf{1}\left\{|\nabla_i f(x^t) - \nabla_i f(x^*)| \le \delta_i\right\}. \quad (5)$$

According to Lemma 3, this quantity measures how many variables are staying “inactive” i.e., not going to move from 0 in next iteration. From Figure 1a, we find that most variables are initially incorrectly labeled as “active” (e.g. $|\nabla_i f(x^t) - \nabla_i f(x^*)| > \delta_i$), but a large number of them quickly switch to “inactive”.

In Figure 1b, we illustrate the objective progress at each step by plotting ρ_t, defined to satisfy

$$F(x^{t+1}) - F^* = (1-\rho_t)\left(F(x^t) - F^*\right), \quad (6)$$

These experiments illustrate the fact that initial convergence of GCD, which, for sparse solutions, may be sufficient to quickly identify the few nonzeros. In particular, from this experiment we observe that

- GCD converges fast initially and $\nabla f(x^t)$ quickly approaches $\nabla f(x^*)$ when x^t is still sparse; and
- before $|\text{supp}(x^t)|$ has grown significantly, the coordinates i where $x^*_i = 0$ have mostly become inactive, and thus future coordinates for W is constrained to $\text{supp}(x^t)$.

We will now mathematically characterize and prove these observations. Before proceeding to our results, we introduce some useful concepts.

Definition 4. $f(x)$ is $\mu_p^{(\tau)}$ strongly convex with respect to $\|\cdot\|_p$ and sparse vectors if $\forall x, y \in \mathbb{R}^d$ such that whenever $|\text{supp}(x) \cup \text{supp}(y)| \le \tau$,

$$f(x) \ge f(y) + \langle \nabla f(y), x-y\rangle + \frac{\mu_p^{(\tau)}}{2}\|x-y\|_p^2,$$

where $p \in \{1,2\}$.

It is easy to check that $\mu_1^{(\tau)}$ and $\mu_2^{(\tau)}$ satisfy the following conditions

$$\begin{aligned}
&\mu_1 = \mu_1^{(d)} \le \mu_1^{(d-1)} \le \ldots \le \mu_1^{(1)},\\
&\mu_2 = \mu_2^{(d)} \le \mu_2^{(d-1)} \le \ldots \le \mu_2^{(1)},\\
&\mu_2^{(\tau)}/\tau \le \mu_1^{(\tau)} \le \mu_2^{(\tau)}\ \forall \tau \in \{1,\ldots,d\}. \quad (7)
\end{aligned}$$

Next, we present a formal analysis to answer why GCD may converge fast initially, and give a bound on the size of the working set W.

4.2 Fast initial convergence

Theorem 4. *Let $\tau = |\text{supp}(x^*)|$ and denote $\{x^i\}_{i=1}^{\infty}$ as the iterates generated from Algorithm 1 with the GS-s rule (selection rule 1). Then for $t < d - \tau$,*

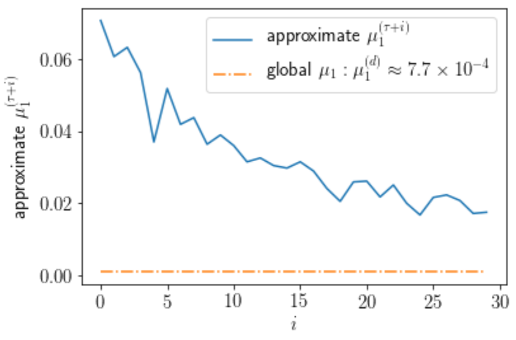
$$\begin{aligned}
F(x^t) - F^* &\le \prod_{i=1}^{\lceil \frac{t}{2}\rceil}\left(1 - \frac{\mu_1^{(\tau+i-1)}}{L}\right)(F(0) - F^*) \quad (8)\\
&\le \prod_{i=1}^{\lceil \frac{t}{2}\rceil}\left(1 - \frac{\mu_2}{(\tau+i-1)L}\right)(F(0) - F^*). \quad (9)
\end{aligned}$$

The bound in Eq. (9) results from using the inequality from Eq. (7).

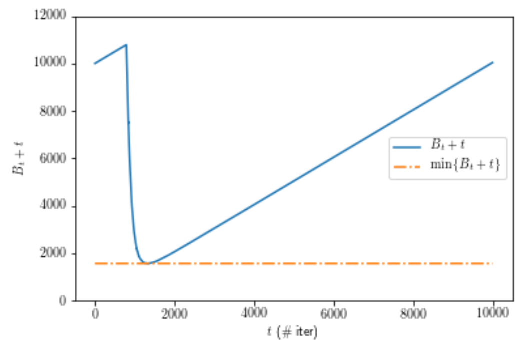
Remark 5. *We give two bounds in Theorem (4) for interpretability.*

- *Bound (9) should be compared with the previous standard analysis of RCD [Nesterov, 2012, Richtárik and Takác, 2014], where*

$$\mathbb{E}\left[F(x^t) - F(x^*)\right] \le \left(1 - \frac{\mu_2}{dL}\right)\left(F(x^0) - F^*\right). \quad (10)$$



(a) The trend of approximate $\mu_1^{(\tau+i)}$, where $\tau = 10$ in this example.



(b) The curve illustrates the upper bound of $|W|$ in Theorem 6.

Figure 2: Illustrations for Theorem 4 and 6

In particular, the variable dimension d is effectively replaced by $\tau + i - 1$, which, if $\tau = \text{supp}(x^*)$ is small and we are in the first few iterations, may be much smaller than d–essentially, this illustrates fast initial convergence.

- *Bound (8) can be compared against the previous GCD analysis of [Nutini et al., 2015, Karimireddy et al., 2019], in particular as written in (3). Here the main imporvement is replacing μ_1 with $\mu_1^{(\tau+i-1)}$, which, as shown in Figure 2a, can be much larger in practice. The bound in Eq. (8) is a refinement over the bound in [Nutini et al., 2015, Karimireddy et al., 2019] because it replaces μ_1 with $\mu_1^{(\tau+i-1)}$, which may be much larger, especially if τ is small (x^* is sparse) and i is small (initial convergence stage). In particular, when τ and i are small, we are guaranteed to be in the case $\mu_2/d \ll \mu_1^{(\tau+i-1)}$ and thus GCD is significantly faster than RCD initially even in worse case scenario.*

The rate we derived here is based on two important ingredients — zero initialization and sparse solution. The screening ability of GCD does not hold without either of these.

To better understand the effect of $\mu_1^{(\tau+i-1)}$, we conduct a simulation to illustrate its trend. Noting that the term $\mu_1^{(\tau+i-1)}$ is hard to compute in general, here we set $\tau = 10$ and for each $i \in \{1, 2, \dots, 30\}$ we generate 10^3 random $(\tau + i - 1)$-sparse vectors and approximate $\mu_1^{(\tau+i-1)}$ as the minimum of $\|Ax\|_2^2/\|x\|_1^2$ over the sample vectors. The plot of approximate $\mu_1^{(\tau+i-1)}$ against i is shown in Figure 2a, and its pattern clearly supports our previous argument.

4.3 Fast sparsity pattern identification

First we define a new error measure $p_\delta(\cdot) : \mathbb{R}_+ \to [d]$:

$$p_\delta(\alpha) = \sum_{i=1}^{d} \mathbf{1}\{\alpha \le \delta_i\},$$

which is used to help quantify the number of inactive elements in the iterates, as in (5). Based on this function and the fast convergence result stated in Theorem 4, we are ready to bound the size of the working set W.

Theorem 6. *Denote $\{x^i\}_{i=1}^{\infty}$ as the iterates generated from Algorithm 1 with the GS-s rule (selection rule 1). Then*

$$|W| \le \min_{t\in[d]} \{B_t + t\} \tag{11}$$

where

$$B_t := d - p_\delta\left(L_\infty \sup_{i\ge t}\left\{\|x^i - x^*\|_1\right\}\right).$$

Here we present an intuitive explanation to this abstract bound. It is easy to verify that B_t here is a decreasing function with t; thus $|W|$ is bounded by the infimum of the sum of a deceasing and increasing function. (See Fig. 2b.) Theorem 6 implies that if $\|x^t - x^*\|_1 \overset{t\to\infty}{\to} 0$ quickly (with $t \ll d$), then this bound will be far less than d.

Again, consider the synthetic LASSO problem ($A \in \mathbb{R}^{50\times 10^4}, \lambda = 2$) as a concrete example to illustrate this bound. In this example, the curve of $B_t + t$ is shown in Figure 2b and the infimum of $B_t + t$ is about 1500 in this case. This experiment demonstrates that the bound we derived in Theorem 6 is non-trivial, especially for problems where d is large.

Based on Theorem 4 and 6 we can derive another bound that depends only on constant $\mu_1^{(\tau+i)}, i \in [d-\tau]$ instead of the iterates x^i.

Corollary 7. *Let $\tau = |\mathrm{supp}(x^*)|$ and denote $\{x^i\}_{i=1}^{\infty}$ as the iterates generated from Algorithm 1 with the GS-s rule (selection rule 1). Then B_t in bound Eq. 11 can be replaced by*

$$B_t := d - p_\delta \left(\sqrt{\frac{2L_\infty^2}{\mu_1} \prod_{i=0}^{t-1} \left(1 - \frac{\mu_1^{(\tau+i)}}{L}\right) R} \right),$$

where $R = F(0) - F^*$ is the initial objective gap.

5 Improved selection rule

Our analysis in section 4 provides a bound on the size of the working set W, and thus provides an explanation for why GCD is fast for sparse optimization. But GCD could be potentially slow in some situations, for example when $|W|$ is large. It is natural to ask if we can improve GCD by trying to keep $|W|$ small? Thus, we propose the GCD variant Δ-GCD which features a selection rule that favours a small final working set. The Δ-GCD algorithm is essentially Algorithm 1 with the following improved selection rule:

Selection rule 8 (Δ-GS-s rule)**.** *Let Given $\Delta \in (0, 1]$, then at the t-th iteration, we select i according to the rule:*

$$i \in \begin{cases} \arg\max\limits_{i\in[d]} Q_i(x^t), & \Delta \max\limits_{i\in[d]} Q_i(x)^2 > \max\limits_{i\in W_t} Q_i(x)^2 \\ \arg\max\limits_{i\in W_t} Q_i(x^t), & \Delta \max\limits_{i\in[d]} Q_i(x)^2 \le \max\limits_{i\in W_t} Q_i(x)^2 \end{cases}$$

where W_t denotes the set of indices accrued thus far and Q_i is defined by Eq. (1)*.*

Note that when $\Delta = 1$, Δ-GCD is equivalent to GCD with the original GS-s rule.

Intuitively, Δ-GCD with small Δ is more likely to focus on current working set, and large Δ encourages the algorithm to explore new coordinates and expand the current working set. Thus Δ controls the trade-off between the size of working set and the progress we made when staying in the current working set. This is similar to the exploration/exploitation trade-off in the context of online learning [Auer et al., 1995].

Theorem 9. *Denote $\{x^i\}_{i=1}^{\infty}$ as the iterates generated from Algorithm 1 with the Δ-GS-s rule (selection rule 8) and W_Δ as its final working set. Then $\forall t > 0$*

$$F(x^t) - F^* \le \left(1 - \frac{\Delta\mu_1^{(|W_\Delta|)}}{L}\right)^{\lceil \frac{t}{2} \rceil} (F(0) - F^*) \quad (12)$$

$$\le \left(1 - \frac{\Delta\mu_2}{|W_\Delta| L}\right)^{\lceil \frac{t}{2} \rceil} (F(0) - F^*). \quad (13)$$

Theorem 9 makes explicit the trade-off between the convergence rate and the size of working set. Again, we provide two bounds for easier interpretation: (12) as a refinement of the strong convexity parameter in [Karimireddy et al., 2019, Nutini et al., 2015]; and (13) where the variable dimension dependency in [Nesterov and Stich, 2017, Richtárik and Takác, 2014] is replaced by the size of the final working set. The Δ-GCD is expected to outperform naive GCD when naive GCD has a comparatively large working set and Δ-GCD can reduce the size of working set with an appropriate value of Δ.

To better understand the relationship between W_Δ and Δ, we present an description of W_Δ as Δ goes to 0. Consider Algorithm 1 with selection rule 1, where additionally we use as a post-processing step

$$x^{t+1} := \arg\min_{\mathrm{supp}(x) \subseteq W_{t+1}} f(x) + g(x)$$

e.g., after each step, we re-solve the problem completely over the coordinates specified in the working set. This is also known as the totally corrective greedy algorithm, and is closely related to orthogonal matching pursuit [Pati et al., 1993, Davis et al., 1997], [Foucart and Rauhut, 2013, §3.2]. We denote the final working set from this scheme as $W^\sharp$. Intuitively, as $\Delta \to 0$, Δ-GCD tends to focus on the current working set until it is close the exact minimizer among the current working set, and approaches $W^\sharp$.

Theorem 10. *Let $k = |W^\sharp|$, denote the iterates from the totally corrective greedy algorithm as $\{x^t\}_{i=0}^{k}$. Assume that $\arg\max\limits_{i\in[d]} \{Q_i(x^t)\}$ are singletons for $t = 0, 1, ..., k-1$ and $\delta_i > 0\ \forall x_i^* = 0$. Then*

$$\lim_{\Delta\to 0} W_\Delta = W^\sharp.$$

If the totally corrective greedy algorithm can yield a small working set, then we expect that a small enough Δ would also yield a small working set (but could probably slow down the convergence according to Theorem 9). Hence, our new algorithm Δ-GCD can be viewed as a flexible greedy algorithm between the two extreme cases – vanila GS-GCD and the totally corrective greedy algorithm.

6 Experiments

In this section, we conduct experiments on both real world data and synthetic data to illustrate the impor-

Table 1: Data description. d denotes number of features and n denotes number of samples.

Datasets	colon	leukemia	make_circle	ijcnn1
d	2,000	7,129	2	22
n	62	72	1,000	35,000

tance of zero initialization and evaluate the effectiveness of Δ-GCD.

The statistics of our experimental data is shown in Table 1, where colon, leukemia and ijcnn1 can be downloaded from LIBSVM's [Chang and Lin, 2011] dataset webpage, and make_circle is generated from the scikit-learn package [Pedregosa et al., 2011]. We solve the LASSO problem over colon and leukemia and the dual RBF kernel SVM for make_circle and ijcnn1 For ijcnn1, we follow the parameter settings in [Hsieh et al., 2014]($\gamma = 2$ and $C = 32$) where γ is the free parameter in the RBF kernel and $1/C$ is the hinge loss weight parameter. All experiments are conducted on a machine with 4 CPUs and 16GB memory.

The code to reproduce our experimental results is public available at `https://github.com/fanghgit/Greed_Meets_Sparsity`.

6.1 Zero v.s. other initializations

In this part, we compare the convergence of GCD for solving LASSO problems over different initializations.

- Zero initialization: $x^0 = 0$.
- Random initialization: x^0 is generated from Gaussian distributions $\mathcal{N}(0, \sigma I_d)$, for $\sigma \in \{1, 0.1, 0.01\}$.
- Least-squares initialization: $x^0 = (A^T A + \lambda I_d)^{-1} A^T b$. This initialization starts the method at a low objective value, but is not sparse.

In Figure 4, we can see that zero initialization clearly outperforms other initialization strategies and random initialization tends to be the worst. In particular, GCD with zero initialization can get close to optimum even before one pass of all coordinates. On the other hand, although GCD with least-squares initialization has a smaller initial objective value than zero initialization, it suffers from slow convergence and requires a full pass of all coordinates before reaching the same low error, which is consistent with our intuition. Random initialization with different standard deviation also varies in their performance and random initialization, with smaller variance tends to converge faster; however they are still outperformed by the zero initialization for the same reasons as the Least-squares initialization.

6.2 Evaluation of Δ-GCD

In this part we evaluate the proposed Δ-GCD algorithm on LASSO, l_1 regularized logistic regression, and kernel SVM problems. As shown in Figure 5, the value of Δ has a clear impact on the size of working set, where smaller Δ tends to promote sparser iterates for all the test problems. This trend is more obvious when the underlying solution is less sparse as shown in Figures 5b and 5f since here vanilla GCD produces a working set that is much larger than needed. Sometimes this stronger screening ability of a smaller Δ can lead to slightly faster convergence compared to vanilla GCD (i.e. $\Delta = 1$) as shown in Figure 5a and 5b. Another by product of Δ-GCD is early identification of the final sparsity pattern, which can be leveraged in two-stage methods [Bertsekas, 1976, Ko et al., 1994, Daniilidis et al., 2009, Wright, 2012]. However, the acceleration functionality of Δ-GCD is not present for all test problems, since vanilla GCD already has a very strong screening ability for constraining the size of working set.

In Figure 3, we present the size of working set after 10^5 iterations (as an approximation for the true $|W_\Delta|$) with different choices of $\Delta \in \{2^{-k} \mid k = 0, 1, \ldots, 6\}$ on a LASSO problem with dataset colon and $\lambda = 0.1$. As shown in the figure, the size of $|W_\Delta|$ is monotony decreasing with the decrease of Δ.

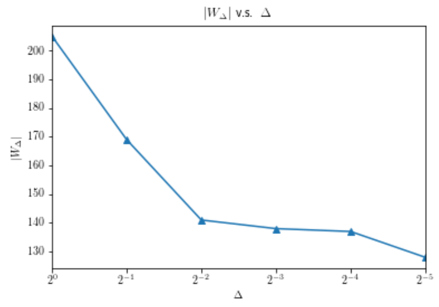
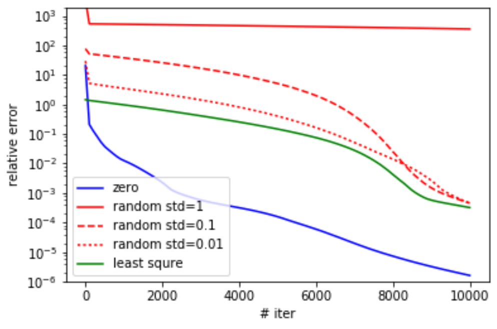


Figure 3: The effect of Δ on W_Δ

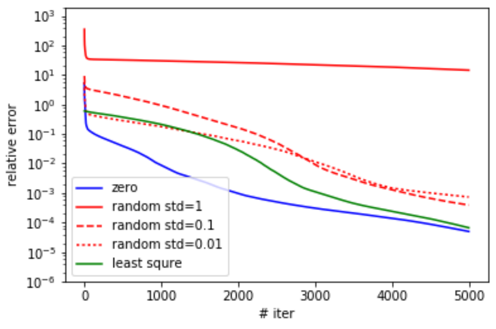
7 Conclusions

By bringing techniques from sparsity pattern identification and convergence analysis of GCD, we formally analyze the screening ability of GCD and explicitly answered why GCD is usually fast for sparse optimization. We also propose an improved selection rule with a stronger ability to encourage sparse iterates and connect it existing algorithms.

For future work, we would like to generalize our analysis to convex smooth functions f that are not necessarily strongly convex, and in particular where x^* may not be unique (but $\text{supp}(x^*)$ may be). The core of our analysis is the convergence of iterates and the challenge here is that general smooth objective does not guarantee convergence in iterates. Another direction is how to further tighten the bound of the working set, the bound we illustrated in Figure 2b is still about 10 times worse than the actual size of the working set, and for small value of λ, the actual working set become larger and our bound can be more than d and become trivial.

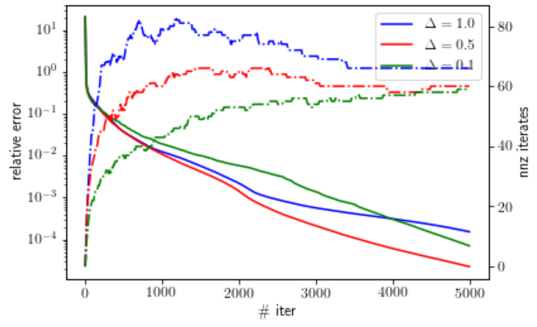


(a) LASSO, data: leukemia, $\lambda = 0.1$.

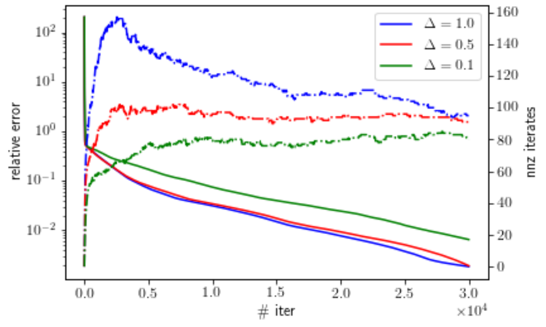


(b) LASSO, data: colon, $\lambda = 0.1$.

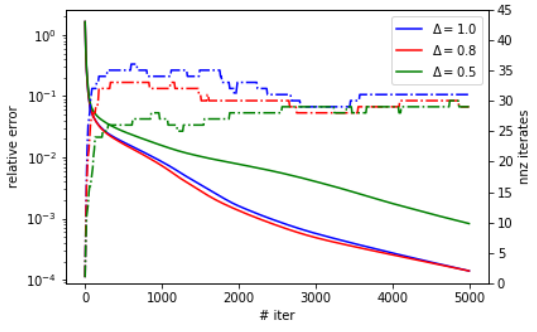
Figure 4: Comparison between different kinds of initialization



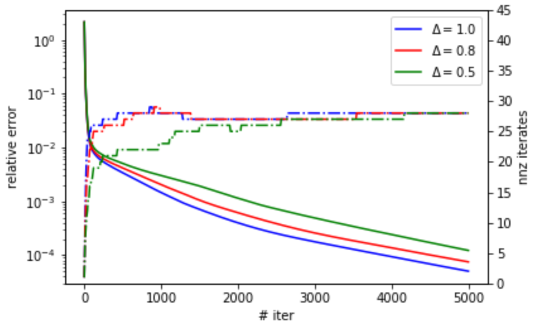
(a) LASSO, data: leukemia, $\lambda = 0.1$. Solid = obj. value, dashed = # nonzeros in x^t.



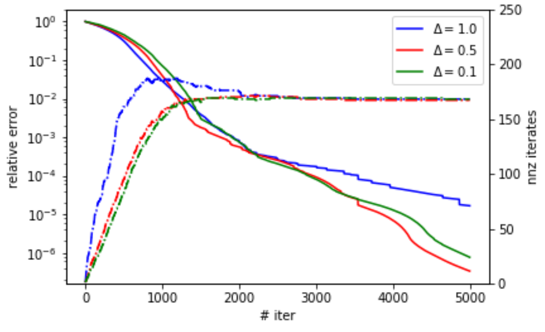
(b) LASSO, data: leukemia, $\lambda = 0.01$. Solid = obj. value, dashed = # nonzeros in x^t.



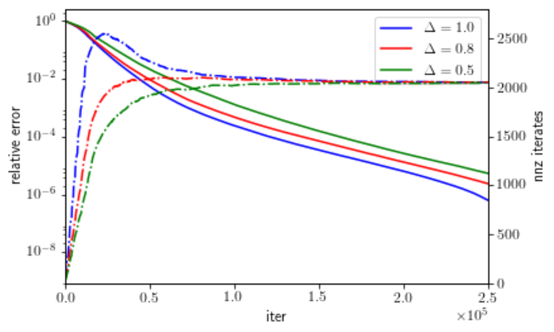
(c) L_1-regularized logistic regression, data: colon, $\lambda = 0.1$.



(d) L_1-regularized logistic regression, data: leukemia, $\lambda = 0.1$.



(e) kernel SVM, data: make_circle, $C = 10, \gamma = 0.5$.



(f) kernel SVM, data: ijcnn1, $C = 32, \gamma = 2$

Figure 5: Compare Δ-GCD with different choices of Δ.

References

[Allen-Zhu et al., 2016] Allen-Zhu, Z., Qu, Z., Richtarik, P., and Yuan, Y. (2016). Even faster accelerated coordinate descent using non-uniform sampling. In *Proceedings of ICML*, volume 48, pages 1110–1119.

[Auer et al., 1995] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-arm bandit problem. In *Proceedings of FOCS*, pages 322–331.

[Bengio et al., 2006] Bengio, Y., Delalleau, O., and Le Roux, N. (2006). Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press.

[Bertsekas, 1976] Bertsekas, D. P. (1976). On the goldstein-levitin-polyak gradient projection method. *IEEE Transactions on automatic control*, 21(2):174–184.

[Bertsekas, 1999] Bertsekas, D. P. (1999). *Nonlinear Programming*, volume second edition. Athena Scientific.

[Burke and Moré, 1988] Burke, J. V. and Moré, J. J. (1988). On the identification of active constraints. *SIAM Journal on Numerical Analysis*, 25(5):1197–1211.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Cichocki and Phan, 2009] Cichocki, A. and Phan, A. H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions*, 92-A(3):708–721.

[Daniilidis et al., 2009] Daniilidis, A., Sagastizábal, C., and Solodov, M. (2009). Identifying structure of nonsmooth convex functions by the bundle technique. *SIAM Journal on Optimization*, 20(2):820–840.

[Davis et al., 1997] Davis, G., Mallat, S., and Avellaneda, M. (1997). Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98.

[Dhillon et al., 2011] Dhillon, I. S., Ravikumar, P., and Tewari, A. (2011). Nearest neighbor based greedy coordinate descent. In *Proceedings of NIPS*, pages 2160–2168.

[Dunn, 1987] Dunn, J. C. (1987). On the convergence of projected gradient processes to singular critical points. *Journal of Optimization Theory and Applications*, 55(2):203–216.

[Foucart and Rauhut, 2013] Foucart, S. and Rauhut, H. (2013). *A Mathematical Introduction to Compressive Sensing*. Birkh&#228;user Basel.

[Hare and Lewis, 2007] Hare, W. L. and Lewis, A. S. (2007). Identifying active manifolds. *Algorithmic Operations Research*, 2(2):75.

[Hastie et al., 2008] Hastie, T., Friedman, J. H., and Tibshirani, R. (2008). Regularization paths and coordinate descent. In *Proceedings of ACM-SIGKDD*, page 3.

[Hsieh et al., 2014] Hsieh, C.-J., Si, S., and Dhillon, I. (2014). A divide-and-conquer solver for kernel support vector machines. In *Proceedings of ICML*, volume 32 of *Proceedings of Machine Learning Research*, pages 566–574. PMLR.

[Joachims, 1999] Joachims, T. (1999). Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

[Karimireddy et al., 2019] Karimireddy, S. P., Koloskova, A., Stich, S. U., and Jaggi, M. (2019). Efficient greedy coordinate descent for composite problems. In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2887–2896. PMLR.

[Ko et al., 1994] Ko, M., Zowe, J., et al. (1994). An iterative two-step algorithm for linear complementarity problems. *Numerische Mathematik*, 68(1):95–106.

[Lewis and Wright, 2011] Lewis, A. S. and Wright, S. J. (2011). Identifying activity. *SIAM Journal on Optimization*, 21(2):597–614.

[Li and Osher, 2009] Li, Y. and Osher, S. (2009). Coordinate descent optimization for l_1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Probl. Imaging*, 3:487–503.

[Liang et al., 2017] Liang, J., Fadili, J., and Peyré, G. (2017). Activity identification and local linear convergence of forward-backward-type methods. *SIAM Journal on Optimization*, 27(1):408–437.

[Liu et al., 2015] Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. (2015). An asynchronous parallel stochastic coordinate descent algorithm. *J. Mach. Learn. Res.*, 16:285–322.

[Luo and Tseng, 1993] Luo, Z. and Tseng, P. (1993). Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46-47(1):157–178.

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

[Nesterov and Stich, 2017] Nesterov, Y. and Stich, S. U. (2017). Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM Journal on Optimization*, 27(1):110–123.

[Nutini et al., 2017a] Nutini, J., Laradji, I., and Schmidt, M. (2017a). Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence.

[Nutini et al., 2017b] Nutini, J., Schmidt, M., and Hare, W. (2017b). "active-set complexity" of proximal gradient: How long does it take to find the sparsity pattern? *Optimization Letter*.

[Nutini et al., 2015] Nutini, J., Schmidt, M. W., Laradji, I. H., Friedlander, M. P., and Koepke, H. A. (2015). Coordinate descent converges faster with the gauss-southwell rule than random selection. In *Proceedings of ICML*, pages 1632–1641.

[Pati et al., 1993] Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar conference on signals, systems and computers*, pages 40–44. IEEE.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830.

[Platt, 1999] Platt, J. C. (1999). Advances in kernel methods. chapter Fast Training of Support Vector Machines Using Sequential Minimal Optimization, pages 185–208. MIT Press.

[Powell, 1973] Powell, M. J. D. (1973). On search directions for minimization algorithms. *Math. Program.*, 4(1):193–201.

[Richtárik and Takác, 2014] Richtárik, P. and Takác, M. (2014). Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program.*, 144(1-2):1–38.

[Richtárik and Takác, 2016] Richtárik, P. and Takác, M. (2016). Parallel coordinate descent methods for big data optimization. *Math. Program.*, 156(1-2):433–484.

[Rockafellar, 1970] Rockafellar, R. T. (1970). *Convex Analysis.* Princeton University Press, Princeton.

[Shalev-Shwartz and Zhang, 2013] Shalev-Shwartz, S. and Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss minimization. *J. Mach. Learn. Res.*, 14(1):2013.

[Southwell, 1940] Southwell, R. V. (1940). Relaxation methods in engineering science : a treatise on approximate computation.

[Sun et al., 2019] Sun, Y., Jeong, H., Nutini, J., and Schmidt, M. W. (2019). Are we there yet? manifold identification of gradient-related proximal methods. In *Proceedings of AISTATS*, pages 1110–1119.

[Tseng and Yun, 2009] Tseng, P. and Yun, S. (2009). A coordinate gradient descent method for nonsmooth separable minimization. *Math. Program.*, 117(1-2):387–423.

[Wright, 1993] Wright, S. J. (1993). Identifiable surfaces in constrained optimization. *SIAM Journal on Control and Optimization*, 31(4):1063–1079.

[Wright, 2012] Wright, S. J. (2012). Accelerated block-coordinate relaxation for regularized optimization. *SIAM Journal on Optimization*, 22(1):159–186.

[Wright, 2015] Wright, S. J. (2015). Coordinate descent algorithms. *Math. Program.*, 151(1):3–34.

[Zhang and Lin, 2015] Zhang, Y. and Lin, X. (2015). Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of ICML*.