Efficient Greedy Coordinate Descent via Variable Partitioning

Huang Fang, Guanhua Fang, Tan Yu, Ping Li

Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{fangazq877, fanggh2018, Tanyuuynat, pingli98}@gmail.com

Abstract

Greedy coordinate descent (GCD) is an efficient optimization algorithm for a wide range of machine learning and data mining applications. GCD could be significantly faster than randomized coordinate descent (RCD) if they have similar per iteration cost. Nevertheless, in some cases, the greedy rule used in GCD cannot be efficiently implemented, leading to huge per iteration cost and making GCD slower than RCD. To alleviate the cost per iteration, the existing solutions rely on maximum inner product search (MIPS) as an approximate greedy rule. But it has been empirically shown that GCD with approximate greedy rule could suffer from slow convergence even with the state-of-the-art MIPS algorithms. We propose a hybrid coordinate descent algorithm with a simple variable partition strategy to tackle the cases when greedy rule cannot be implemented efficiently. The convergence rate and theoretical properties of the new algorithm are presented. The proposed method is shown to be especially useful when the data matrix has a group structure. Numerical experiments with both synthetic and real-world data demonstrate that our new algorithm is competitive against RCD, GCD, approximate GCD with MIPS and their accelerated variants.

1 INTRODUCTION

With an immense growth of data in recent years, classical optimization algorithms tend to struggle with the current huge amount of data. For example, some of today's social networks could have hundreds of millions of users and even solving a linear system using the classical approach in this scale is challenging. Many efficient algorithms are proposed or re-discovered in the last two decades to solve today's large-scale optimization problems. Due to the low numerical accuracy required by most machine learning and data mining tasks, first-order methods with cheap cost per iteration dominate certain fields recently. In particular stochastic gradient descent (SGD) and coordinate descent (CD) are two most important representatives. SGD and its variants are dominant for the big-N problems i.e., problems with a large number of samples (Robbins and Monro, 1951; Ghadimi and Lan, 2013; Johnson and Zhang, 2013; Defazio et al., 2014; Nguyen et al., 2017; Schmidt et al., 2017) while CD and its variants are highly effective in handling the structured big-p problems i.e., problems with a large number of features (Bertsekas and Tsitsiklis, 1989; Luo and Tseng, 1992, 1993; Tseng, 2001; Nesterov, 2012; Lee and Sidford, 2013; Shalev-Shwartz and Zhang, 2013; Richtárik and Takác, 2014; Wright, 2015; Lu and Xiao, 2015; Allen-Zhu et al., 2016; Zhang and Xiao, 2017). Starting with Nesterov's seminal work (Nesterov, 2012), coordinate descent and its variants has been an active research topic in both academia community and industrial practice.

Greedy coordinate descent (GCD) belongs to the family of CD. Differentiating from randomized coordinate descent (RCD), which randomly selects a coordinate to update in each iteration, GCD selects the coordinate that makes the most progress. This selection rule is also known as the Gauss-Southwell (GS) rule. It has been theoretically verified that GCD could converge faster than RCD if the GS rule can be implemented efficiently (Nutini et al., 2015). However, different from RCD which can be implemented efficiently for a wide range of problems, only a small class of problems are suitable for the GS rule (Nutini et al., 2015). For the cases when the GS rule cannot be implemented efficiently, one could only resort to the approximate GS rule based on maximum inner product search (MIPS) algorithms (Dhillon et al., 2011; Shrivastava and Li, 2014; Karimireddy et al., 2019). However, most existing algorithms for MIPS do not have strong theoretical guarantee and extensive numerical experiments in the literature (Karimireddy et al., 2019) have empirically shown that the approximate GS rule with the

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

state-of-the-art MIPS algorithm usually performs worse than RCD on real-world datasets.

In this work, we propose a hybrid coordinate descent (hybridCD) algorithm with a simple variable partitioning strategy as an alternative of the approximate GS rule when GCD cannot be implemented efficiently. The proposed hybridCD algorithm aims to achieve the fast convergence rate as GCD while preserving the low iteration cost as randomized CD via devised variable partitioning. On the theory side, we show that the convergence rate of our hybridCD algorithm could match GCD and outperform RCD when the data matrix has an underlying group structure. We also provide an accelerated hybridCD scheme which further improves the convergence rate. On the empirical side, we evaluate the new algorithm with ridge regression, logistic regression, linear support vector machine (SVM) and graph-based label propagation models. The results show that the proposed method empirically outperforms RCD, GCD, approximate GCD with MIPS and their accelerated variants on synthetic datasets when the data group structure exists and also achieves the better performance on real-world datasets.

2 RELATED WORK

2.1 COORDINATE DESCENT

Coordinate descent is an old optimization method that can be traced back to 1940s (Southwell, 1940). It is gaining increasing interest in the past decade due to its superior empirical performance on the structured big-p (high dimensional) machine learning and data mining applications including LASSO (Friedman et al., 2007; Hastie et al., 2008; Friedman et al., 2010), support vector machine (SVM) (Platt, 1999; Tseng and Yun, 2010), non-negative matrix factorization (Cichocki and Phan, 2009), graph-based label propagation (Bengio et al., 2006), Mazumder et al. (2011) etc. Despite its popularity in practice, the convergence rate of CD is not clear until Nesterov (2012)'s work in 2012, in which Nesterov presented the first non-asymptotic convergence rate of RCD. Subsequent works include block CD (Beck and Tetruashvili, 2013) proximal RCD (Richtárik and Takác, 2014), accelerated RCD (Lee and Sidford, 2013; Lin et al., 2015; Allen-Zhu et al., 2016; Nesterov and Stich, 2017).

2.2 GREEDY COORDINATE DESCENT

Nesterov (2012) demonstrated that GCD has the same convergence rate as RCD. However, GCD is observed to be much faster than RCD in terms of convergence rate empirically, and some clever implementations of GCD do constitute the state-of-the-art solvers for machine learning problems like kernel SVM (Joachims, 1999; Chang and Lin, 2011) and non-negative matrix factorization (Cichocki and Phan, 2009). This gap between practice and theory was filled with refined analysis (Nutini et al., 2015) of GCD (see Tseng and Yun (2009); Nutini et al. (2017) for an extension to blocked GCD), in which Nutini et al. (2015) theoretically explained why GCD could be faster than RCD for certain class of problems. Subsequent works (Karimireddy et al., 2019; Fang et al., 2020) further extended the refined analysis to composite optimization problems.

When the GS rule cannot be implemented efficiently, Dhillon et al. (2011) proposed to use MIPS algorithm as an approximate GS rule for least square problems, Karimireddy et al. (2019) further discussed how to map the approximate GS rule to MIPS for a richer family of problems beside least square problems. These two papers are closely related to this work as all of them are considering how to improve GCD when the GS rule cannot be implemented efficiently.

2.3 PARALLEL COORDINATE DESCENT

Another line of work in recent years is to accelerate CD under multi-core or multi-machine environments (Liu et al., 2015; Richtárik and Takác, 2016). This line of work is orthogonal to the purpose of this work, but it is worth to note that the variable partitioning strategy used in this work has been mentioned as heuristics for parallel greedy coordinate algorithms (Scherrer et al., 2012b,a; You et al., 2016; Moreau et al., 2018). It is also worth to note that these works apply greedy rule within each partition in a parallel fashion and is different from the algorithm proposed in this paper which advocates greedy rule among all partitions.

3 PRELIMINARIES

We consider the following optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}), \tag{3.1}$$

where d is the number of variables, $f : \mathbb{R}^d \to \mathbb{R}$ is a convex and smooth function. Throughout the paper, we use $\mathbf{x}^*$ to denote a solution of problem 3.1 and f^* as the optimal objective value. In addition, we make the following assumptions on f.

Assumption 3.1. *$f(\mathbf{x}+\alpha\mathbf{e}_i)$ is L_i-smoothness in terms of α $\forall i \in [d]$:*

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

where $\mathbf{e}_i$ is the i-th unit vector, $\nabla_i f(\mathbf{x})$ denotes the i-th entry of $\nabla f(\mathbf{x})$.

Assumption 3.2. *$f(\mathbf{x})$ is L-smoothness:*

$$f(\mathbf{x}) \le f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle + \frac{L}{2}\|\mathbf{x}-\mathbf{y}\|_2^2,\ \forall \mathbf{x},\mathbf{y}\in\mathbb{R}^d.$$

Algorithm 1 Pseudocode for coordinate descent

Input: $\mathbf{x}_0$.
for $t = 0, 1, 2, \ldots$ **do**
  [**RCD rule**] uniform randomly choose a i from $\{1, 2, \ldots, d\}$
  [**GS rule**] $i \in \arg\max_{j \in [d]} |\nabla_i f(\mathbf{x}^{(t)})|$
  [**Approx-GS rule**] choose i from $\{1, 2, \ldots, d\}$ by MIPS algorithm.
  $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{L_i} \nabla f_i(\mathbf{x}^{(t)}) \mathbf{e}_i$
end for

In the following content of this paper, we denote $L_{\max} := \max_{i \in [d]} L_i$. It is worth mentioning here that $L_{\max}$ is usually much smaller than the global smoothness constant L. Then we introduce the definition of strong convexity, which is the standard assumption used to obtain linear convergence rate in the literature.

Definition 3.3. $f(\mathbf{x})$ is μ_2 and μ_1 strongly convex with respect to $\|\cdot\|_2$ and $\|\cdot\|_1$ respectively:

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{\mu_2}{2} \|\mathbf{x} - \mathbf{y}\|_2^2, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

and

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle + \frac{\mu_1}{2} \|\mathbf{x} - \mathbf{y}\|_1^2, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d.$$

It is easy to verify that $\mu_2/d \leq \mu_1 \leq \mu_2$, we refer readers to Nutini et al. (2015) for more discussions on the relationship between μ_2 and μ_1. Note that in order to prove linear convergence rate, recent works show that the strongly convex condition can be relaxed to some weaker conditions (Karimi et al., 2016). For simplicity, we use the strongly convex condition for part of our analysis. A general template for coordinate descent is shown in Algorithm 1. When we use random selection rule, the algorithm is RCD and when we apply the GS rule, it reduces to GCD.

4 MOTIVATION

Nesterov (2012)'s seminal work established the first global convergence rate of RCD and GCD:

$$\mathbb{E}\left[f(\mathbf{x}^{(t+1)}) - f^*\right] \leq \left(1 - \frac{\mu_2}{dL_{\max}}\right)\left(f(\mathbf{x}^{(t)}) - f^*\right),$$

where f^* is the optimum value. Later on, Nutini et al. (2015) refined the analysis of GCD, showing that with the GS rule, we have

$$f(\mathbf{x}^{(t+1)}) - f^* \leq \left(1 - \frac{\mu_1}{L_{\max}}\right)\left(f(\mathbf{x}^{(t)}) - f^*\right).$$

Recall from previous section, μ_1 is the strongly convexity constant with respect to $\|\cdot\|_1$, it satisfy the condition $\mu_2/d \leq \mu_1 \leq \mu_2$. It is straightforward to see that when $\mu_2/d \lessapprox \mu_1$, GCD tends to have similar convergence rate as RCD, when $\mu_1 \lessapprox \mu_2$, GCD would have d times convergence rate as RCD. In this case, GCD will be much faster than RCD if they have similar computation cost per iteration.

Algorithm 2 Hybrid coordinate descent

Input: $\mathbf{x}^{(0)}, \mathcal{B} = \{B_i\}_{i=1}^k$.
for $t = 0, 1, 2, \ldots$ **do**
  $I = \emptyset$
  for $j = 1, 2, \ldots, k$ **do**
    [**Random rule**] uniform randomly choose a $i_j \in B_j$ and let $I = I \cup \{i_j\}$
  end for
  [**Greedy rule**] $i \in \arg\max_{j \in I} |\nabla_j f(\mathbf{x}^{(t)})|$
  $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} - \frac{1}{L_i} \nabla f_i(\mathbf{x}^{(t)}) \mathbf{e}_i$
end for

Unfortunately, for a wide range of applications including linear regression, logistic regression, LASSO and dual SVM, the GS rule cannot be implemented efficiently as random selection rule unless the data in both row and column sparse (Nutini et al., 2015, 2017). These observations naturally raise a question: can we design an algorithm that preserve the advantage of both GCD and CD – converge as fast as GCD with cost per iteration as cheap as RCD?

5 PROPOSED METHOD

We propose a hybrid coordinate descent algorithm incorporating a variable partitioning strategy, the detailed algorithm is shown in Algorithm 2.

The algorithm is straightforward to understand:

- First, we partition all variables into k blocks.
- In each iteration, we perform random selection rule within each blocks and get k candidate coordinates.
- Then we apply the GS rule over the k candidate coordinates to get the best coordinate among them.
- Apply standard coordinate gradient step on the chosen coordinate.

We denote k as the number of blocks and the partitions as $\mathcal{B} = \{B_i\}_{i=1}^k$. In fact, we can change the number of blocks and the partition during the optimization process, we will discuss the influence of the choice of partition on the algorithm convergence in following section.

One can immediately observe from Algorithm 2 that when $k = 1$, the greedy selection disappears and the algorithm reduces to RCD. Meanwhile, when $k = d$, each block represents one coordinate and the algorithm is essentially GCD.

6 ANALYSIS

We analyze the convergence of hybridCD and present a concrete example show that hybridCD could outperform RCD and GCD, all missing proofs are placed in §1 from supplementary materials.

6.1 CONVERGENCE

First, we introduce a new norm induced by the partition $\mathcal{B}$.

Definition 6.1. Given a partition $\mathcal{B} \coloneqq \{B_i\}_{i=1}^k$ such that $B_i \cap B_j = \emptyset \ \forall i \neq j \in [d], \bigcup_{i=1}^k B_i = \{1, 2, \ldots, d\}$. $\forall \mathbf{x} \in \mathbb{R}^d$, we define the l_∞ norm of $\mathbf{x}$ induced by the partition $\mathcal{B}$ as $\|\mathbf{x}\|_{\mathcal{B},\infty} := \max_{i\in[k]} \frac{1}{\sqrt{|B_i|}} \|\mathbf{x}_{B_i}\|_2$, where x_B denotes a sub-vector with coordinates in B.

The dual norm of $\|\cdot\|_{\mathcal{B},\infty}$ is given as follows. Interestingly, its dual norm coincides with the group-norm (Yuan and Lin, 2006).

Lemma 6.2. *We denote the dual norm of $\|\cdot\|_{\mathcal{B},\infty}$ as $\|\cdot\|_{\mathcal{B},1}$, and it can be expressed as*

$$\|\mathbf{x}\|_{\mathcal{B},1} := \sup_{\|\mathbf{z}\|_{\mathcal{B},\infty}\leq 1} \langle \mathbf{z}, \mathbf{x}\rangle = \sum_{i=1}^k \sqrt{|B_i|}\|\mathbf{x}_{B_i}\|_2.$$

We know that gradient descent is the steepest descent method in 2-norm and GCD is performing steepest descent in 1-norm (Boyd and Vandenberghe, 2004, § 9.4.3). Our hybridCD can be viewed as the steepest descent method in the group-norm induced by the partition $\mathcal{B}$.

It is easy to check that the new norm satisfies the following conditions:

Lemma 6.3. *The new norm satisfy the following condition:*

$$\begin{aligned} \frac{\|x\|_2^2}{d} \leq \quad \|x\|_{\mathcal{B},\infty}^2 \leq \quad & \frac{\|x\|_2^2}{\min_i |B_i|} \quad \forall x \in \mathbb{R}^d, \\ \frac{\|x\|_\infty^2}{\max_i |B_i|} \leq \quad \|x\|_{\mathcal{B},\infty}^2 \leq \quad & \|x\|_\infty^2 \qquad \forall x \in \mathbb{R}^d. \end{aligned}$$

Lemma 6.4 (Descent lemma)**.** *Denote $\mathbf{x}^{(t)}$ as the iterate generated from Algorithm 2, then we have the following result:*

$$\mathbb{E}\left[f(\mathbf{x}^{(t+1)}) \,\middle|\, \mathbf{x}^{(t)}\right] \leq f(\mathbf{x}^{(t)}) - \frac{1}{2L_{\max}}\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}^2.$$

Proof. Given $\mathbf{x}^{(t)}$, following Algorithm 2,

$$\begin{aligned} & \mathbb{E}\left[f(\mathbf{x}^{(t+1)}) \,\middle|\, \mathbf{x}^{(t)}\right] \\ \overset{\text{(i)}}{\leq} & \mathbb{E}\Big[f(\mathbf{x}^{(t)}) + \left\langle \nabla f(\mathbf{x}^{(t)}), -\frac{1}{L_i}\nabla_i f(\mathbf{x}^{(t)})\mathbf{e}_i \right\rangle \\ & \qquad + \frac{L_i}{2L_i^2}\|\nabla_i f(\mathbf{x}^{(t)})\|_2^2\Big] \\ \leq & f(\mathbf{x}^{(t)}) - \frac{1}{2L_{\max}}\mathbb{E}\left[\left(\nabla_i f(\mathbf{x}^{(t)})\right)^2\right] \\ = & f(\mathbf{x}^{(t)}) - \frac{1}{2L_{\max}}\mathbb{E}\left[\max_{j\in[k]}\left\{\left(\nabla_{i_j} f(\mathbf{x}^{(t)})\right)^2\right\}\right] \\ \overset{\text{(ii)}}{\leq} & f(\mathbf{x}^{(t)}) - \frac{1}{2L_{\max}}\max_{j\in[k]}\left\{\mathbb{E}\left[\left(\nabla_{i_j} f(\mathbf{x}^{(t)})\right)^2\right]\right\} \\ = & f(\mathbf{x}^{(t)}) - \frac{1}{2L_{\max}}\max_{j\in[k]}\left\{\frac{1}{|B_j|}\|\nabla_{B_j} f(\mathbf{x}^{(t)})\|_2^2\right\} \\ = & f(\mathbf{x}^{(t)}) - \frac{1}{2L_{\max}}\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}^2, \end{aligned} \tag{6.1}$$

where (i) is true by plugging $\alpha = -\nabla_i f(\mathbf{x}^{(t)})/L_i$ to the Assumption 3.1 and (ii) is from Jensen's inequality. □

Based on Lemma 6.4, we can immediately obtain the following convergence rate for strongly convex objective.

Theorem 6.5 (Convergence for strongly convex objective)**.** *Denote $\mathbf{x}^{(t)}$ as the iterate generated from Algorithm 2, assume f is μ_1 and μ_2 strongly convex for 1 and 2-norm respectively, then we have the following convergence result:*

$$\mathbb{E}\left[f(\mathbf{x}^{(t+1)}) - f^* \,\middle|\, \mathbf{x}^{(t)}\right] \leq (1-\eta_t)\left(f(\mathbf{x}^{(t)}) - f^*\right),$$

where

$$\eta_t := \max\left\{\frac{\mu_2}{L_{\max}}\frac{\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}^2}{\|\nabla f(\mathbf{x}^{(t)})\|_2^2}, \frac{\mu_1}{L_{\max}}\frac{\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}^2}{\|\nabla f(\mathbf{x}^{(t)})\|_\infty^2}\right\}.$$

By using Lemma 6.3, it is easy to verify that $\eta_t \in [\mu_2/(dL_{\max}), \mu_1/L_{\max}]$. Different from GCD with approximate GS rule (with additive error) that might suffer from non-convergent, the convergence rate of hybridCD is at least as fast as RCD in the worst case scenario. On the other side. It is clear that the quantity $\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x}^{(t)})\|_\infty$ plays a crucial role in the convergence rate at the t-th iteration. If this quantity is close to 1, the proposed algorithm will converge as fast as GCD.

Following the same argument as Dhillon et al. (2011); Karimireddy et al. (2019), we can obtain the convergence of hybridCD for convex but not necessarily strongly convex objectives.

Theorem 6.6 (Convergence for convex objective)**.** *Denote* $\mathbf{x}^{(t)}$ *as the iterate generated from Algorithm 2, then we have the following convergence result:*

$$\mathbb{E}\left[f(\mathbf{x}^{(t)}) - f^*\right] \leq \frac{2L_{\max}D^2}{\rho t},$$

where $\rho := \inf_{\mathbf{x}\in\mathbb{R}^d}\{\|\nabla f(\mathbf{x})\|^2_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x})\|^2_\infty\}$ *and* $D = \sup_{\mathbf{x}\in\mathbb{R}^d}\{\|\mathbf{x}-\mathbf{x}^*\|_1 \mid f(\mathbf{x}) \leq f(\mathbf{x}^{(0)})\}$.

The term $\rho := \inf_{\mathbf{x}\in\mathbb{R}^d}\{\|\nabla f(\mathbf{x})\|^2_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x})\|^2_\infty\}$, similar to Theorem 6.5, controls the convergence rate, hybridCD could converge as fast as GCD when it is close to 1.

Theorem 6.5 and Theorem 6.6 described the convergence rate on expectation. Based on these results, we can obtain convergence rates with high probability using standard techniques (Richtárik and Takác, 2014). We place the high probability error bounds in supplementary materials §1.

Our convergence rate analysis indicated that the quantity $\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x}^{(t)})\|_\infty$ stays in the center of our analysis. A natural question arise at this point: how does the partition $\mathcal{B}$ affect the quantity $\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x}^{(t)})\|_\infty$? Intuitively, if there is a clustering pattern in $\nabla f(\mathbf{x}^{(t)})$, $\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x}^{(t)})\|_\infty$ will be close to 1. In this case, $\mathcal{B}$ tends to partition similar values in $\nabla f(\mathbf{x}^{(t)})$ into the same block. To formally choose a good partition strategy, one option is to maximize the worst-case convergence rate, which leads to a combinatorial optimization problem, namely

$$\max_{\mathcal{B}} \min_{\mathbf{x}\in\mathbb{R}^d} \frac{\|\nabla f(\mathbf{x}^{(t)})\|_{\mathcal{B},\infty}}{\|\nabla f(\mathbf{x}^{(t)})\|_\infty}.$$

This is, however, a hard combinatorial optimization problem even with simple objective functions like least square. In fact, solving this problem could be even harder than the original optimization problem.

Here we would like to have a simple and effective partitioning strategy and clustering is an intuitive choice. We present a more formal discussion on how clustering helps hybridCD when our data has an underlying group structure in the following subsection.

6.2 PARTITIONING STRATEGY

We consider the optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{A}\mathbf{x}), \tag{6.2}$$

where f is a convex and smooth function, $\mathbf{A} \in \mathbb{R}^{n\times d}$, where n is the number of data points and d is the number of features. Assume that each column of $\mathbf{A}$ is generated from a mixture Gaussian distribution i.e., $\sum_{i=1}^k \pi_i \mathcal{N}(\boldsymbol{\mu}_j, \mathbf{I}_n\sigma^2)$. Then we have the following improved bound on $\inf_{\mathbf{x}\in\mathbb{R}^d}\{\|\nabla f(\mathbf{x})\|^2_{\mathcal{B},\infty}/\|\nabla f(\mathbf{x})\|^2_\infty\}$.

Theorem 6.7. *Assume that* $j \in B_i$ *if the* j*-th column of* $\mathbf{A}$ *is generated from the* i*-th cluster* $\mathcal{N}(\boldsymbol{\mu}_i, \mathbf{I}_n\sigma^2)$*, then*

$$\inf_{\mathbf{x}\in\mathbb{R}^d} \frac{\|\nabla f(\mathbf{x})\|^2_{\mathcal{B},\infty}}{\|\nabla f(\mathbf{x})\|^2_\infty} = \Omega\left(\frac{1}{n\max_i \log^2|B_i|}\right) \tag{6.3}$$

holds with probability at least $1 - 4d\exp\{-n/4\}$.

Theorem 6.7 improves the bound from $1/\max_i|B_i|$ to $1/(n\max_i\log^2|B_i|)$, the improvement is significant when $\max_{i\in[k]}|B_i| \gg n$.

Remark 6.8. *Under the condition stated in Equation (6.3), we summarize the iteration complexities of RCD, GCD and hybridCD to obtain an* ϵ*-error as follows. When* f *is* f *is* μ_1 *and* μ_2 *strongly convex for 1 and 2-norm respectively,*

- *RCD:* $\mathcal{O}\left(\frac{dL_{\max}}{\mu_2}\log\left(\frac{1}{\epsilon}\right)\right)$;
- *GCD:* $\mathcal{O}\left(\frac{L_{\max}}{\mu_1}\log\left(\frac{1}{\epsilon}\right)\right)$;
- *hybridCD:* $\mathcal{O}\left(\frac{n\max_i\log^2|B_i|L_{\max}}{\mu_1}\log\left(\frac{1}{\epsilon}\right)\right)$.

For not necessarily strongly convex objective f*,*

- *RCD:* $\mathcal{O}\left(\frac{dL_{\max}D_2^2}{\epsilon}\right)$;
- *GCD:* $\mathcal{O}\left(\frac{L_{\max}D_1^2}{\epsilon}\right)$;
- *hybridCD:* $\mathcal{O}\left(\frac{n\max_i\log^2|B_i|L_{\max}D_1^2}{\epsilon}\right)$,

where $D_i := \sup_{\mathbf{x}\in\mathbb{R}^d}\{\|\mathbf{x}-\mathbf{x}^*\|_i \mid f(\mathbf{x}) \leq f(\mathbf{x}^{(0)})\}, i \in \{1,2\}$.

Remark 6.9. *Consider the logistic regression problem, where* f *in Equation (6.2) is set to* $f(\mathbf{y}) = \sum_{i=1}^n \log(1+\exp(-\mathbf{y}_i))$*. Assume that the condition stated in Equation (6.3) holds. Then the number of flops required to achieve* ϵ*-error is:*

- *RCD:* $\mathcal{O}\left(\frac{nd\|A\|_\infty D_2^2}{\epsilon}\right)$;
- *GCD:* $\mathcal{O}\left(\frac{nd\|A\|_\infty D_1^2}{\epsilon}\right)$;
- *hybridCD:* $\mathcal{O}\left(\frac{n^2k\max_i\log^2|B_i|\|A\|_\infty D_1^2}{\epsilon}\right)$.

When $d \gg n$ and $d \gg k$, hybridCD requires significantly less flops than RCD and GCD. This again shows the effectiveness of our proposed algorithm.

In Theorem 6.7, we do the partition according to the true underlying groups, i.e., $\mathcal{B} \equiv \mathcal{B}^*$. (We use $*$ to denote the true group.) However, in practice, we usually do not have a prior knowledge on the group structure and the partition is obtained by some clustering algorithms. Therefore, we develop the following theorem under inexact partitioning.

Theorem 6.10. *Assume $j \in B_i^*$ if the j-th column of $\mathbf{A}$ is generated from the i-th cluster $\mathcal{N}(\boldsymbol{\mu}_i, \mathbf{I}_n\sigma^2)$ and the clustering algorithm returns a partition $\mathcal{B} = \{B_i\}_{i=1}^k$. Let $\mathbf{c}_i$ be the center of B_i i.e., $\mathbf{c}_i := \frac{1}{|B_i|}\sum_{j\in B_i}\mathbf{a}_j$ and $\mu_{gap} := \max_{i,j\in[k]} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\|_\infty$. Assume that*

A1 $\forall i \in [k]$, $\frac{|B_i \cap B_i^*|}{|B_i|} \geq 1 - \frac{\sigma}{\mu_{gap}+\sigma\log|B_i|}$.

A2 $\max_{j\in B_i} \|\mathbf{a}_j - \mathbf{c}_i\|_2 \leq \min_{j'\in B_i^*} C\|\mathbf{a}_{j'} - \mathbf{c}_i\|_2$ $\forall i \in [k]$ *for some constant* $C > 0$.

Then we have

$$\inf_{\mathbf{x}\in\mathbb{R}^d} \frac{\|\nabla f(\mathbf{x})\|_{\mathcal{B},\infty}^2}{\|\nabla f(\mathbf{x})\|_\infty^2} = \Omega\left(\frac{1}{nC^2\max_i \log^2(|B_i|)}\right)$$

with probability at least $1 - 4d\exp\{-n/4\} - 1/n$.

Conditions *A1* and *A2* require the clustering algorithm to output a reasonable partition. When the noise parameter σ gets smaller, the clustering algorithm is required to be more accurate. The bound in Theorem 6.10 is within a multiplicative factor C^2 of the bound in Theorem 6.7, where C approximately measures how close our partition $\mathcal{B}$ approximates $\mathcal{B}^*$.

6.3 ACCELERATION

Next, we apply Nesterov's acceleration technique to hybridCD. Our algorithm is inspired by the accelerated semi-greedy CD (ASCD) algorithm developed by Lu et al. (2018). ASCD maintains two variables $\mathbf{x}^{(t)}$ and $\mathbf{z}^{(t)}$, where $\mathbf{x}^{(t)}$ is updated in a greedy manner and $\mathbf{z}^{(t)}$ is updated in the same way as randomized CD. We naturally extend their acceleration technique to hybridCD and the detailed algorithm described in Algorithm 3. By modifying the original convergence analysis of ASCD, we obtain the following convergence result.

Theorem 6.11. *Denote $\mathbf{x}^{(t)}$ as the iterate generated from Algorithm 3, then*

$$\mathbb{E}[f(\mathbf{x}^{(t)}) - f^*] \leq \frac{2d^2L_{\max}\|\mathbf{x}^{(0)} - \mathbf{x}^*\|_2^2}{(t+1)^2}. \tag{6.4}$$

The convergence rate in Equation (6.4) matches exactly the same as the convergence rate of ASCD, but ASCD requires a greedy search over all d coordinates and can not resolve the issue of high per iteration cost. Note that this rate is also the same as the rate of accelerated RCD (Lee and Sidford, 2013; Allen-Zhu et al., 2016) (ARCD). Therefore the rate in Equation (6.4) cannot explain why greedy selection rule works better than its randomized counterpart even though we obtained a $\mathcal{O}(1/t^2)$ rate. We empirically show the effectiveness of accelerated hybridCD in numerical experiments.

7 EXPERIMENTS

Table 1: Data statistics, n is the number of data samples, d is the number of features.

Datasets	synthetic	leukemia	rcv1	ijcnn1
n	50	72	697,641	49,990
d	5,000	7,129	47,236	22

In this section, we evaluate our algorithm on ridge regression, logistic regression, linear SVM and label propagation models on both synthetic and real-world datasets, the statistics of our experimental data are shown in Table 1, where leukemia, rcv1, ijcnn1 are downloadable from LIBSVM's (Chang and Lin, 2011) webpage[1]. More details on the use of each dataset is presented as follows,

- synthetic is constructed by the make_blobs function from the sklearn package (Pedregosa et al., 2011) with number of clusters set to 8, this data has an inherent clustering pattern by its construction and it is expected to work well with hybridCD. We use both synthetic

Algorithm 3 Accelerated hybrid coordinate descent

Input: $\mathbf{x}_0, \mathcal{B} = \{B_i\}_{i=1}^k$. Define the sequence $\{\theta_t\}$ as follows: $\theta_0 = 1$ and construct θ_t recursively by $\frac{1-\theta_t}{\theta_t^2} = \frac{1}{\theta_{t-1}^2}$ for $t = 1, \ldots$.
for $t = 0, 1, 2, \ldots$ **do**
  $\mathbf{y}^{(t)} = (1-\theta_t)\mathbf{x}^{(t)} + \theta_t\mathbf{z}^{(t)}$, $\quad I = \emptyset$
  for $j = 1, 2, \ldots, k$ **do**
    [Random rule 1] uniform randomly choose a $i_j \in B_j$ and let $I = I \cup \{i_j\}$
  end for
  [Greedy rule] $j_1 \in \arg\max_{j\in I} |\nabla_j f(\mathbf{y}^{(t)})|$
  $\mathbf{x}^{(t+1)} = \mathbf{y}^{(t)} - \frac{1}{L_{\max}}\nabla f_{j_1}(\mathbf{y}^{(t)})\mathbf{e}_{j_1}$
  [Random rule 2] randomly choose a $j_2 \in I$ with probability $\{|B_1|/d, |B_2|/d, \ldots, |B_k|/d\}$
  $\mathbf{z}^{(t+1)} = \mathbf{z}^{(t)} - \frac{1}{dL_{\max}\theta_t}\nabla f_{j_2}(\mathbf{y}^{(t)})\mathbf{e}_{j_2}$
end for

[1] https: /www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/

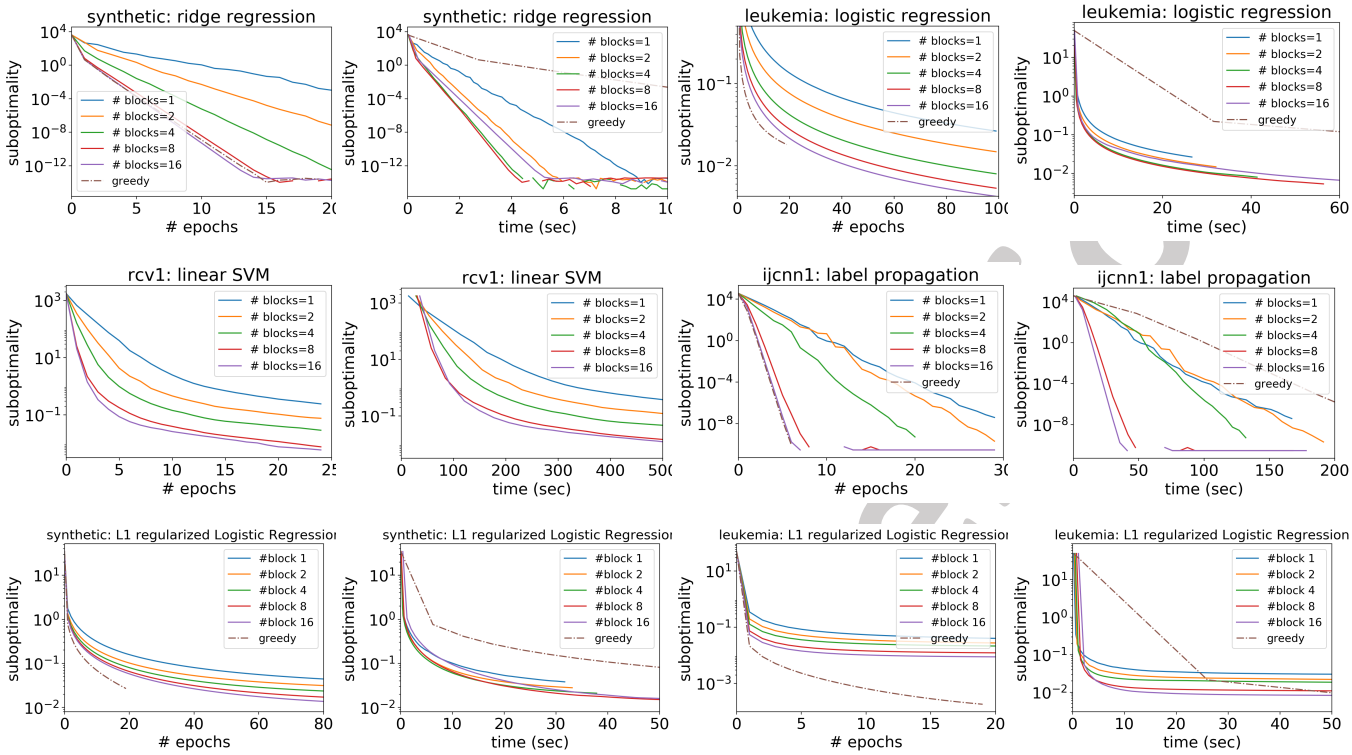

Figure 6.1: Convergence of hybridCD, RCD and GCD. We compare their convergence in terms of the number of iteration and runtime. For a fair comparison in runtime, the overhead of clustering used in hybridCD are included in the total runtime. Note that the curve of GCD is not included in the linear SVM task since the cost per iteration is too expensive with the naive implementation.

and leukemia for ridge regression, logistic regression and ℓ_1 regularized logistic regression problems. Note that ℓ_1 regularized logistic regression problem involves a ℓ_1 regularization and we solve it with a proximal-gradient variant of hybridCD (algorithm described in §2).

- rcv1 is used for linear SVM tasks, we use CD to solve the dual problem and compare different algorithms on objective value and test accuracy, we adopt the default training/test splitting of rcv1 where $20,242$ data points are used for training and $677,399$ are used for testing.
- We construct a 5 nearest neighbour graph with ijcnn1 under Euclidean distance and use the resulting graph for label propagation task. We randomly set $39,992$ samples as labeled data and the remaining $9,998$ samples as unlabeled data. It is worth to note that the GS-rule can be implemented efficiently using maxheap, but the implementation is sophisticated (Meshi et al., 2012). Here we use a naive implementation of GCD for simplicity, this will increase the total runtime of GCD but will not affect the objective gap v.s. iteration curve and we still have a fair comparison in terms of convergence rate.

We use the *k-means* algorithm as our variable partition strategy for experiments if not otherwise specified. We try the number of blocks k in $\{1, 2, 4, 8, 16\}$ in our numerical experiments, it is worth to note that when $k = 1$, hybridCD reduces RCD. To obtain a fair comparison on runtime, we include the overhead induced by the clustering algorithm into the total runtime in our experiments though the time spent on *k-means* is usually far less than the total runtime.

We conduct all experiments on a Linux server with 11 Intel(R) Xeon(R) CPUs E5-2620 v2 (2.10GHz) and 128GB memory. All algorithms are implemented in Python.

7.1 COMPARISON: HYBRIDCD VS. RCD & GCD

In Figure 6.1, we can clearly see that hybridCD with 8 or 16 blocks achieves similar convergence rate as GCD on synthetic. Since that synthetic has an inherent clustering pattern with 8 clusters, the convergence result on synthetic implies that hybridCD with simple clustering partition strategy and appropriate number of blocks indeed can be as fast as GCD and we could get diminishing returns if we set the number of blocks more than the real number of underlying clusters of the data. These observations are consistent with our analysis in previous sections.

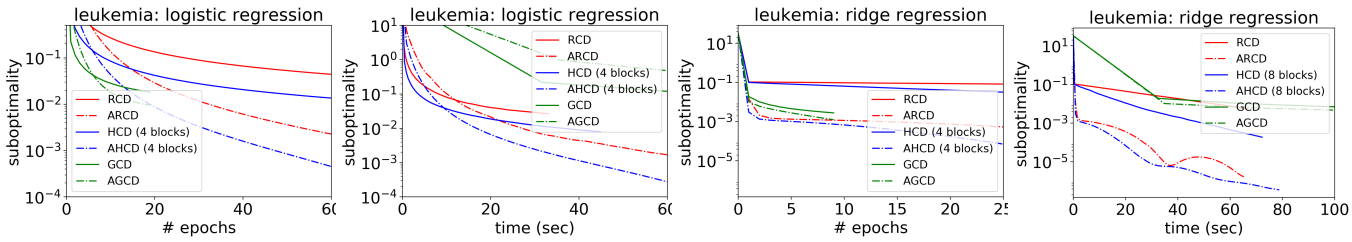

Figure 7.1: Convergence comparison among RCD, ARCD, HCD, AHCD, GCD and AGCD.

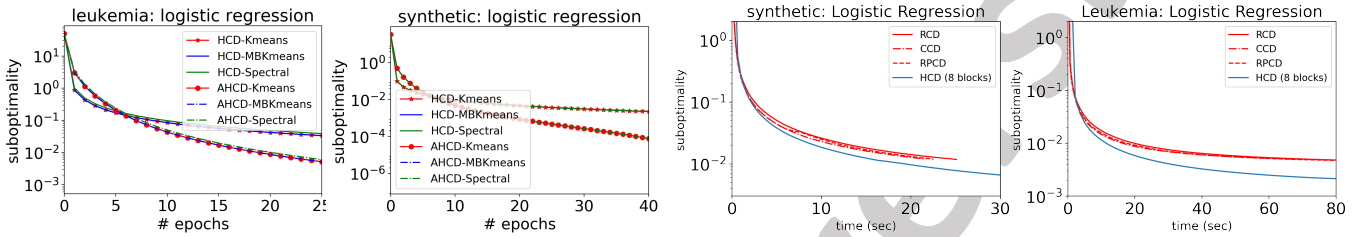

Figure 7.2: Left two figures: the convergence of HCD and AHCD with different clustering algorithms. Right two figures: the convergence of RCD, CCD, RPCD and HCD.

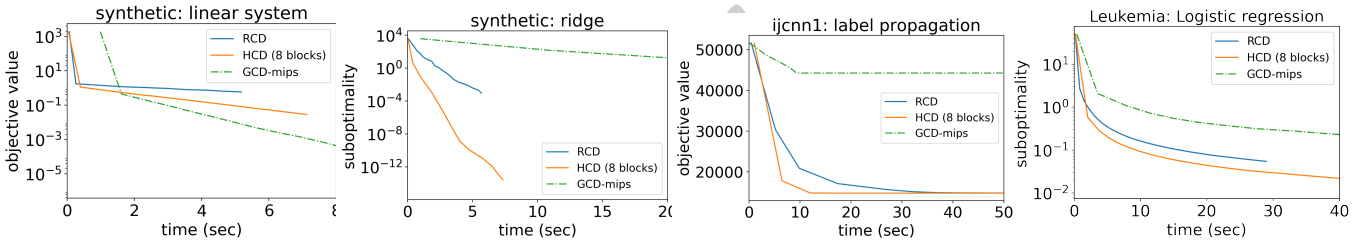

Figure 7.3: The convergence of objective value (training error) of hybridCD (with 8 blocks), RCD and approximate GCD with the state-of-the-art MIPS algorithm.

All experiments on real world datasets share the same pattern, hybridCD with more blocks tends to have a better convergence rate and too many blocks could increase the time per iteration and increase the total runtime. In terms of convergence rate, hybridCD with more blocks behaves similar as GCD and much better than RCD, while the cost per iteration of hybridCD with few blocks is significantly cheaper than GCD and comparable to RCD. As a result, the proposed hybridCD with appropriate number of blocks outperforms both RCD and GCD.

7.2 COMPARISON: ACCELERATED VARIANTS

Figure 7.1 presents the convergence comparison among RCD, accelerated RCD (ARCD), hybridCD (HCD), accelerated hybridCD (AHCD), GCD and accelerated GCD (AGCD). For the implementation of ARCD and AHCD, we adopt the change of variable techniques developed by Lee and Sidford (2013) and therefore the per iteration cost of ARCD and AHCD are the same as RCD and HCD respectively. We can observe that RCD, HCD, GCD with Nesterov acceleration indeed exhibit faster convergence than their non-accelerated counterparts. AHCD with appropriate number of blocks is the fastest among all competitors.

7.3 COMPARISON: CLUSTERING STRATEGIES AND OTHER SELECTION RULES

We experiment hybridCD with different clustering algorithms including *k-means*, *minibatch-kmeans* and *spectral clustering*. The convergence curves of HCD and AHCD with different clustering algorithms are given in the left two figures from Figure 7.2. The convergence curves under different clustering algorithms almost coincide with each other and therefore hybridCD is empirically robust to the change of partitioning strategy.

The right two figures from 7.2 present the convergence of CD with some other commonly used selection rules including cyclic (CCD) and randomly permuted selection rule (RPCD). We can see that RCD, CCD and RPCD has similiar convergence behaviour and hybridCD could outperform them with appropriate number of blocks.

7.4 COMPARISON: HYBRIDCD VS. GCD-MIPS

Similar to Karimireddy et al. (2019), we use the widely used package nmslib[2] for the MIPS algorithm.

As shown in Figure 7.3, GCD with MIPS (GCD-MIPS) is competitive against RCD and hybridCD in solving over-determined linear system but a much inferior than RCD and hybridCD in ridge regression and label propagation tasks. Different from RCD and hybridCD, GCD with MIPS could suffer from non-convergence. The reason is that ridge regression and label propagation require the MIPS algorithm used in GCD to search in high dimensional space, see Karimireddy et al. (2019) for more details and explanations. In this scenario, the cost per iteration of GCD-MIPS tends to be high and its searching quality tends to be low, and therefore GCD-MIPS could suffer from slower or even non-convergence in these tasks.

8 CONCLUSION

In this work, we propose a hybrid coordinate descent (hybridCD) algorithm to improve greedy coordinate descent (GCD) for optimization problems when the exact GS-rule cannot be implemented efficiently. We provide a relatively complete convergence analyses for the proposed hybridCD and its accelerated version. Noticeably, our theory demonstrates that the proposed method can converge as fast as GCD if the data matrix has an underlying group structure and the algorithm adopts the reasonable partitioning strategy. Multiple numerical experiments illustrate that our hybridCD with k-means clustering partitioning strategy is highly effective, hybridCD and its accelerated variant outperform other existing baselines, such as RCD, ARCD, GCD, AGCD and GCD with MIPS, on both synthetic and real-world datasets in the tasks of ridge regression, logistic regression and label propagation. Although we empirically show that the simple clustering is surprisingly effective, we believe there is still space for further improvement for the partitioning strategy, for example, one may use the partitioning strategy based on different metrics; one may also design a dynamic partitioning strategy that adaptively change the partition during the optimization process. Furthermore, the current theoretical analysis only considers the smooth objective functions. It will be interesting and challenging to extend the theory of hybridCD algorithm to general non-smooth situations. We left these as the directions of the future work.

ACKNOWLEDGEMENT

We sincerely thank the anonymous reviewers and area chair from the UAI program committee for their constructive suggestions, which helped improve the quality of this paper.

[2]https://github.com/nmslib/nmslib

Bibliography

Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even faster accelerated coordinate descent using non-uniform sampling. In *Proceedings of the 33nd International Conference on Machine Learning (ICML)*, pages 1110–1119, New York City, NY, 2016.

Amir Beck and Luba Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM J. Optim.*, 23(4):2037–2060, 2013.

Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pages 193–216. MIT Press, 2006.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., USA, 1989. ISBN 0136487009.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

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

Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1646–1654, Montreal, Canada, 2014.

Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2160–2168, Granada, Spain, 2011.

Huang Fang, Zhenan Fan, Yifan Sun, and Michael P. Friedlander. Greed meets sparsity: Understanding and improving greedy coordinate descent for sparse optimization. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 434–444, Online [Palermo, Sicily, Italy], 2020.

Jerome Friedman, Trevor Hastie, Holger Hoefling, and Robert Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 2(1):302–332, 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1): 1–22, 2010.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.*, 23(4):2341–2368, 2013.

Trevor Hastie, Jerome H. Friedman, and Robert Tibshirani. Regularization paths and coordinate descent. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, page 3, Las Vegas, NV, 2008.

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

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, Lake Tahoe, NV, 2013.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD), Part I*, pages 795–811, Riva del Garda, Italy, 2016.

Sai Praneeth Karimireddy, Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Efficient greedy coordinate descent for composite problems. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2887–2896, Naha, Okinawa, Japan, 2019.

Yin Tat Lee and Aaron Sidford. Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 147–156, Berkeley, CA, 2013.

Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM J. Optim.*, 25(4):2244–2273, 2015.

Ji Liu, Stephen J. Wright, Christopher Ré, Victor Bittorf, and Srikrishna Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. *J. Mach. Learn. Res.*, 16: 285–322, 2015.

Haihao Lu, Robert M. Freund, and Vahab S. Mirrokni. Accelerating greedy coordinate descent methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3263–3272, Stockholmsmässan, Stockholm, Sweden, 2018.

Zhaosong Lu and Lin Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Math. Program.*, 152(1-2):615–642, 2015.

Zhiquan Luo and Paul Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):361–379, 1992.

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

Rahul Mazumder, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparsenet : Coordinate descent with non-convex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011.

Ofer Meshi, Tommi S. Jaakkola, and Amir Globerson. Convergence rate analysis of MAP coordinate minimization algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3023–3031, Lake Tahoe, NV, 2012.

Thomas Moreau, Laurent Oudre, and Nicolas Vayatis. DICOD: distributed convolutional coordinate descent for convolutional sparse coding. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3623–3631, Stockholmsmässan, Stockholm, Sweden, 2018.

Yurii E. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22(2):341–362, 2012.

Yurii E. Nesterov and Sebastian U. Stich. Efficiency of the accelerated coordinate descent method on structured optimization problems. *SIAM J. Optim.*, 27(1):110–123, 2017.

Lam M. Nguyen, Jie Liu, Katya Scheinberg, and Martin Takác. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2613–2621, Sydney, Australia, 2017.

Julie Nutini, Mark Schmidt, Issam H. Laradji, Michael P. Friedlander, and Hoyt A. Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1632–1641, Lille, France, 2015.

Julie Nutini, Issam Laradji, and Mark Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*, 2017.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

John C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*, pages 185–208, 1999.

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

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

Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3): 400–407, 1951.

Chad Scherrer, Mahantesh Halappanavar, Ambuj Tewari, and David Haglin. Scaling up coordinate descent algorithms for large l_1 regularization problems. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, UK, 2012a.

Chad Scherrer, Ambuj Tewari, Mahantesh Halappanavar, and David Haglin. Feature clustering for accelerating parallel coordinate descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 28–36, Lake Tahoe, NV, 2012b.

Mark Schmidt, Nicolas Le Roux, and Francis R. Bach. Minimizing finite sums with the stochastic average gradient. *Math. Program.*, 162(1-2):83–112, 2017.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *J. Mach. Learn. Res.*, 14(1):567–599, 2013.

Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems (NIPS)*, pages 2321–2329, Montreal, Canada, 2014.

R. V. Southwell. Relaxation methods in engineering science : a treatise on approximate computation. 1940.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001.

Paul Tseng and Sangwoon Yun. Block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *Journal of optimization theory and applications*, 140(3):513, 2009.

Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training. *Computational Optimization and Applications*, 47(2):179–206, 2010.

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

Yang You, Xiangru Lian, Ji Liu, Hsiang-Fu Yu, Inderjit S. Dhillon, James Demmel, and Cho-Jui Hsieh. Asynchronous parallel greedy coordinate descent. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4682–4690, Barcelona, Spain, 2016.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 68: 49–67, 2006.

Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *J. Mach. Learn. Res.*, 18:84:1–84:42, 2017.