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 Sidor, 2013; Shalev-Shwartz and Zhang, 2013; Richtárik and Takáč, 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
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:

\[
\min_{x \in \mathbb{R}^d} f(x),
\]

(3.1)

where \(d\) is the number of variables, \(f : \mathbb{R}^d \rightarrow \mathbb{R}\) is a convex and smooth function. Throughout the paper, we use \(x^*\) 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. \(f(x + \alpha e_i)\) is \(L_i\)-smoothness in terms of \(\alpha \forall i \in [d] : \)

\[
|\nabla_i f(x + \alpha e_i) - \nabla_i f(x)| \leq L_i |\alpha|, \ \forall x \in \mathbb{R}^d, \alpha \in \mathbb{R},
\]

where \(e_i\) is the \(i\)-th unit vector, \(\nabla_i f(x)\) denotes the \(i\)-th entry of \(\nabla f(x)\).

Assumption 3.2. \(f(x)\) is \(L\)-smoothness:

\[
f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2} \|x - y\|_2^2, \ \forall x, y \in \mathbb{R}^d.
\]
When we use random selection rule, the algorithm is RCD and we have $f$ constant with respect to $\mu$. Recall from previous section, $\mu$ refined the analysis of GCD, showing that with the GS rule, the convergence rate of RCD and GCD:

$$4 \text{ MOTIVATION}$$

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

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

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

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

Recall from previous section, $\mu_1$ 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 $\mu_2/d \leq \mu_1$, GCD tends to have similar convergence rate as RCD, when $\mu_1 \leq \mu_2$, 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.

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?

$$5 \text{ PROPOSED METHOD}$$

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 $B = \{B_i\}_{i=1}^k$. 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 showing that hybridCD could outperform RCD and GCD, all missing proofs are placed in §9 from supplementary materials.

6.1 CONVERGENCE

First, we introduce a new norm induced by the partition $B$.

**Definition 6.1.** Given a partition $B := \{ B_i \}_{i=1}^k$ 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 x \in \mathbb{R}^d$, we define the $l_{\infty}$ norm of $x$ induced by the partition $B$ as $\| x \|_{B, \infty} := \max_{i \in [k]} \frac{1}{\sqrt{|B_i|}} \| x \|_2$, where $x_B$ denotes a sub-vector with coordinates in $B$.

The dual norm of $\| \cdot \|_{B, \infty}$ 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 $\| \cdot \|_{B, \infty}$ as $\| \cdot \|_{B, 1}$, and it can be expressed as

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

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 $B$.

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

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

$$\frac{\| x \|_2^2}{d} \leq \| x \|_{B, \infty}^2 \leq \frac{\| x \|_2^2}{\min_i |B_i|} \forall x \in \mathbb{R}^d,$$

$$\frac{\| x \|_{\infty}^2}{\max_i |B_i|} \leq \| x \|_{B, \infty}^2 \leq \| x \|_\infty^2 \forall x \in \mathbb{R}^d.$$

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

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

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

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

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 $x^{(t)}$ as the iterate generated from Algorithm 2, assume $f$ is $\mu_1$ and $\mu_2$ strongly convex for 1 and 2-norm respectively, then we have the following convergence result:

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

where

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

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 $\| \nabla f(x^{(t)}) \|_{B, \infty}/\| \nabla f(x^{(t)}) \|_{\infty}$ 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 $x^{(t)}$ as the iterate generated from Algorithm 2, then we have the following convergence result:

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

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

The term $\rho := \inf_{x \in \mathbb{R}^d} \{ \| \nabla f(x) \|_{B, \infty} / \| \nabla f(x) \|_\infty \}$, 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 Takáč, 2014). We place the high probability error bounds in supplementary materials §9.

Our convergence rate analysis indicated that the quantity $\| \nabla f(x^{(t)}) \|_{B, \infty} / \| \nabla f(x^{(t)}) \|_\infty$ stays in the center of our analysis. A natural question arise at this point: how does the partition $B$ affect the quantity $\| \nabla f(x^{(t)}) \|_{B, \infty} / \| \nabla f(x^{(t)}) \|_\infty$? Intuitively, if there is a clustering pattern in $\nabla f(x^{(t)})$, $\| \nabla f(x^{(t)}) \|_{B, \infty} / \| \nabla f(x^{(t)}) \|_\infty$ will be close to 1. In this case, $B$ tends to partition similar values in $\nabla f(x^{(t)})$ 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

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

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

$$
\min_{x \in \mathbb{R}^d} f(Ax),
$$

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

**Theorem 6.7.** Assume that $j \in B_i$, if the $j$-th column of $A$ is generated from the $i$-th cluster $N(\mu_i, I_n \sigma^2)$, then

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

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 $\epsilon$-error as follows. When $f$ is $f = \mu_1$ and $\mu_2$ strongly convex for 1 and 2-norm respectively,

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

For not necessarily strongly convex objective $f$,

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

where $D_i := \sup_{x \in \mathbb{R}^d} \{ \| x - x^* \|_1 \mid f(x) \leq f(x^{(0)}) \}, i \in \{1, 2\}$.

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

- $\text{RCD}: O \left( \frac{nd \| A \|_{\infty} D_2^2}{\epsilon} \right)$;
- $\text{GCD}: O \left( \frac{nd \| A \|_{\infty} D_2^2}{\epsilon} \right)$;
- $\text{hybridCD}: O \left( \frac{n^2 k \max_i \log^2 |B_i| \| A \|_{\infty} D_2^2}{\epsilon} \right)$. 
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., $B \equiv B^*$. (We use $\ast$ 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 $A$ is generated from the $i$-th cluster $\mathcal{N}(\mu_i, I_n \sigma^2)$ and the clustering algorithm returns a partition $B = \{B_i\}_{i=1}^k$. Let $c_i$ be the center of $B_i$ i.e., $c_i := \frac{1}{|B_i|} \sum_{j \in B_i} a_{ij}$ and $\mu_{gap} := \max_{i,j \in [k]} \| \mu_i - \mu_j \|_{\infty}$. Assume that

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

\[ A2 \max_{j \in B_i} \| a_{ij} - c_i \|_2 \leq \min_{j \in B_i} C \| a_{ij} - c_i \|_2 \forall i \in [k] \text{ for some constant } C > 0. \]

Then we have

\[ \inf_{x \in \mathbb{R}^d} \frac{\| \nabla f(x) \|_{L_{\infty}}^2}{\| \nabla f(x) \|_{\infty}^2} = \Omega \left( \frac{1}{n C^2 \max_{i} \log^2(|B_i|)} \right) \]

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 $\sigma$ 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 $B$ approximates $B^*$.

### 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 $x^{(t)}$ and $z^{(t)}$, where $x^{(t)}$ is updated in a greedy manner and $z^{(t)}$ 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 $x^{(t)}$ as the iterate generated from Algorithm 3, then

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

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 $O(1/t^2)$ 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.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>synthetic</th>
<th>leukemia</th>
<th>rcv1</th>
<th>ijcnn1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>50</td>
<td>72</td>
<td>697,641</td>
<td>49,990</td>
</tr>
<tr>
<td>$d$</td>
<td>5,000</td>
<td>7,129</td>
<td>47,236</td>
<td>22</td>
</tr>
</tbody>
</table>

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:** $x_0, B = \{B_i\}_{i=1}^k$. Define the sequence $\{\theta_t\}$ as follows: $\theta_0 = 1$ and construct $\theta_t$ recursively by $\frac{1 - \theta_t}{\theta_{t-1}^2} = \frac{1}{\theta_{t-1}^2}$ for $t = 1, \ldots$.

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

$y^{(t)} = (1 - \theta_t)x^{(t)} + \theta_t z^{(t)}, \quad I = \emptyset$

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

[Random rule 1] uniformly random choose a $i_j \in B_j$ and let $I = I \cup \{i_j\}$

**end for**

[Greedy rule] $j_1 \in \arg \max_{j \in I} \| \nabla f(y^{(t)}) \|_{\infty}$

$x^{(t+1)} = y^{(t)} - \frac{1}{L_{\max}} \nabla f_{j_1}(y^{(t)}) e_{j_1}$

[Random rule 2] randomly choose a $j_2 \in I$ with probability $\{|B_1|/d, |B_2|/d, \ldots, |B_k|/d\}$

$z^{(t+1)} = z^{(t)} - \frac{1}{L_{\max} \sigma_{j_2}} \nabla f_{j_2}(y^{(t)}) e_{j_2}$

**end for**

^1https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
and leukemia for ridge regression, logistic regression and $\ell_1$ regularized logistic regression problems. Note that $\ell_1$ regularized logistic regression problem involves a $\ell_1$ regularization and we solve it with a proximal-gradient variant of hybridCD (algorithm described in §10).

- 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.
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-*k*-means 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 similar 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


9 PROOFS

Proof of Lemma 6.2

Proof. Given $x \in \mathbb{R}^d$, 

$$\|x\|_{B^1} := \sup_{\|z\|_{B^\infty} \leq 1} \langle z, x \rangle$$

$$= \sup_{z \in \mathbb{R}^d} \left\{ \langle z, x \rangle \mid \max_{i \in [k]} \frac{1}{\sqrt{|B_i|}} \|z_{B_i}\|_2 \leq 1 \right\}$$

$$= \sup_{z \in \mathbb{R}^d} \left\{ \sum_{i=1}^k \langle z_{B_i}, x_{B_i} \rangle \mid \frac{1}{\sqrt{|B_i|}} \|z_{B_i}\|_2 \leq 1 \forall i \in [k] \right\}$$

$$= \sum_{i=1}^k \sup_{z \in \mathbb{R}^d} \left\{ \langle z_{B_i}, x_{B_i} \rangle \mid \frac{1}{\sqrt{|B_i|}} \|z_{B_i}\|_2 \leq 1 \right\}$$

$$(i) = \sum_{i=1}^k \sqrt{|B_i|} \|x_{B_i}\|_2.$$

For (i), the maximum is attained when $z_{B_i} = \sqrt{|B_i|} x_{B_i} / \|x_{B_i}\|_2$.

Proof of Theorem 6.5

Proof. We begin with Equation (6.1),

$$\text{Equation (6.1)} \leq f(x(t)) - \frac{1}{2L_{\max}} \left( \frac{\|\nabla f(x(t))\|_{B^\infty}}{\|\nabla f(x(t))\|_{\infty}} \right)^2 \|\nabla f(x(t))\|_{\infty}^2$$

The next step follows from the refined analysis of GCD from Nutini et al. (2015), we present it here for completeness. Since $\mu_1$ is strongly convex, we have

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

By minimizing left-hand and right-hand sides over $x$, we get

$$f^* \geq f(y) - \sup_{x \in \mathbb{R}^d} \left( \langle \nabla f(y), y - x \rangle - \frac{\mu_1}{2} \|y - x\|_1^2 \right)$$

$$= f(y) - \left( \frac{\mu_1}{2} \|\cdot\|_1^2 \right)^* (\nabla f(y))$$

$$(i) = f(y) - \frac{1}{2\mu_1} \|\nabla f(y)\|_\infty^2,$$ (9.1)

where (i) uses the fact that the convex conjugate of $\frac{1}{2} \|\cdot\|_1^2$ is $\frac{1}{2} \|\cdot\|_\infty^2$. By subtracting $f^*$ from left-hand and right-hand sides of Eq. (6.1) and combining with Eq. (9.1), we get

$$\mathbb{E} \left[ f(x(t)) - f^* \right] \leq \left( 1 - \frac{\mu_1}{L_{\max}} \frac{\|\nabla f(x(t))\|_{B^\infty}}{\|\nabla f(x(t))\|_{\infty}} \right) (f(x(t)) - f^*).$$ (9.2)

Furthermore, with $\frac{\|x\|^2}{d} \leq \|x\|_{B^\infty}^2$ and Eq. (6.1), we get
\[ \mathbb{E}\left[f(x^{(t)})\right] \leq f(x^{(t)}) - \frac{1}{2dL_{\text{max}}} \|\nabla f(x^{(t)})\|_2^2, \]

Using the same argument to derive Eq. (9.2) or following the standard analysis for randomized coordinate descent, we get

\[ \mathbb{E}\left[f(x^{(t)}) - f^*\right] \leq \left(1 - \frac{\mu_2}{L_{\text{max}}} \frac{\|\nabla f(x^{(t)})\|_\infty^2}{\|\nabla f(x^{(t)})\|_2^2}\right) \left(f(x^{(t)}) - f^*\right). \tag{9.3} \]

We complete the proof by combining Eq. (9.2) and Eq. (9.3).

**Proof of Theorem 6.6**

Proof. We begin with Equation (6.1) and follow the standard proof template (Karimireddy et al., 2019; Dhillon et al., 2011).

\[ \mathbb{E}[f(x^{(t+1)}) | x^{(t)}] \leq f(x^{(t)}) - \frac{\eta^2}{2L_{\text{max}}} \|\nabla f(x^{(t)})\|_\infty^2 \]

\[ \leq (i) f(x^{(t)}) - \frac{\eta^2}{2L_{\text{max}}} \|x^{(t)} - x^*\|_1^2 (f(x^{(t)}) - f^*)^2. \]

\[ \leq f(x^{(t)}) - \frac{\eta^2}{2L_{\text{max}}D^2} (f(x^{(t)}) - f^*)^2. \]

where (i) is from the following inequality

\[ f(x^{(t)}) - f^* \leq \langle x^* - x^{(t)}, -\nabla f(x^{(t)}) \rangle \leq \|x^* - x^{(t)}\|_1 \|\nabla f(x^{(t)})\|_\infty. \]

Taking expectation on both sides,

\[ \mathbb{E}[f(x^{(t+1)})] \leq \mathbb{E}[f(x^{(t)})] - \frac{\eta^2}{2L_{\text{max}}D^2} (\mathbb{E}[f(x^{(t)})] - f^*)^2, \]

Note that we use the fact that \( \mathbb{E}[X^2] \geq \mathbb{E}[X]^2 \) to derive the above property. Denote \( \mathbb{E}[f(x^{(t)})] - f^* \) as \( h_t \), then we can get

\[ h_{t+1} \leq h_t - \frac{\eta^2}{2L_{\text{max}}D^2} h_t^2. \tag{9.4} \]

Dividing both side by \( h_{t+1}h_t \), we get

\[ \frac{1}{h_t} \leq \frac{1}{h_{t+1}} - \frac{\eta^2}{2L_{\text{max}}D^2} \frac{h_t}{h_{t+1}} \leq \frac{1}{h_{t+1}} - \frac{\eta^2}{2L_{\text{max}}D^2}, \tag{9.5} \]

where (i) is from the fact that \( \{h_t\}_{t=1}^\infty \) is a decreasing sequence and \( h_t/h_{t+1} \geq 1 \). Summing Equation (9.5) over \( t \in \{0, 1, \ldots, T\} \), we get

\[ \frac{1}{h_0} - \frac{1}{h_T} \leq -\frac{T\eta^2}{2L_{\text{max}}D^2} \]

\[ \implies h_T \leq \frac{2L_{\text{max}}D^2}{\eta^2 T}, \]

which completes the proof.
Proof of Theorem 6.7

Proof. Given any vector \( r \), we let \( z_i = a_i^T r \) and define \( m_j := \mu_j^T r \). Therefore,

\[
\sum_{i \in B_j} z_i^2 = (m_j + z_i - m_j)^2 = |B_j|m_j^2 + 2m_j \sum_{i \in B_j} (z_i - m_j) + \sum_{i \in B_j} (z_i - m_j)^2. \quad (9.6)
\]

According to Rudelson and Vershynin (2010), with probability at least \( 1 - 2 \exp\{-n/2\} \), we have that

\[
\sum_{i \in B_j} (z_i - m_j)^2 = \|A_{B_j} r\|^2 \geq (\sqrt{|B_j|} - 2 \sqrt{n}^2 \|r\|^2 \sigma^2
\]

and

\[
| \sum_{i \in B_j} (z_i - m_j)| \leq \sigma \|r\|_2 \sqrt{|B_j| \log n}
\]

hold for all \( r \). Here \( A_{B_j} \) is the \( j \)th-block submatrix of \( A \) by shifting mean to zero. Therefore, we have

\[
\sum_{i \in B_j} z_i^2 \geq |B_j|m_j^2 - 2m_j\sigma\|r\|_2 \sqrt{|B_j| \log n} + (\sqrt{|B_j|} - 2 \sqrt{n}^2 \|r\|^2 \sigma^2 \quad (9.7)
\]

by simplifying (9.6). On the other hand,

\[
\max_{i \in B_j} |z_i| = \max_{i \in B_j} |\mu_j r + (a_i - \mu_j)^T r| \leq m_j + \max_{i \in B_j} |(a_i - \mu_j)^T r| \leq m_j + 2 \log |B_j| \sqrt{n} \|r\|_2 \sigma
\]

holds with probability at least \( 1 - 2 |B_j| \exp\{-n/4\} \). Thus, when \( |B_j| \gg n \), we get

\[
\frac{\sqrt{\sum_{i \in B_j} z_i^2}}{\max_{i \in B_j} |z_i|} \geq \frac{\sqrt{|B_j|m_j^2 - 2m_j\sigma\|r\|_2 \sqrt{|B_j| \log n} + (\sqrt{|B_j|} - 2 \sqrt{n}^2 \|r\|^2 \sigma^2}}}{m_j + 2 \log |B_j| \sqrt{n} \|r\|_2 \sigma} \geq \frac{c \sqrt{|B_j|m_j^2 + \|r\|^2 \sigma^2}}{\log |B_j| \sqrt{n} \sqrt{m_j^2 + \|r\|^2 \sigma^2}} \quad (9.8)
\]

\[
= c \sqrt{|B_j|/\log |B_j| \sqrt{n}} \quad (9.9)
\]

for some universal constant \( c \). Notice that the above results hold for all \( r \). This implies that

\[
\frac{\|\nabla B_j f\|_2 / \sqrt{|B_j|}}{\|\nabla B_j f\|_\infty} \geq \frac{c}{\log |B_j| \sqrt{n}}, \quad (9.10)
\]

if we specifically take \( r = f'(Ax^t) \). This further gives

\[
\|\nabla f(x')\|_{\infty} \geq \frac{c}{\max_j \log |B_j| \sqrt{n}} \|\nabla f(x')\|_{\infty},
\]

When \( \max_j |B_j| \geq d/k \gg n \), there is huge improvement in lower bound, from \( 1/(\max_j \sqrt{|B_j|}) \) to \( 1/(\max_j \log |B_j| \sqrt{n}) \). This concludes the proof. \( \Box \)
Proof of Theorem 6.10

Proof. We first show that \( \| c_j - \mu_j \| \leq \delta \sqrt{n} \). We compare the difference between \( k \)th coordinates of \( c_j \) and \( \mu_j \). Then

\[
|c_j(k) - \mu_j(k)| = \frac{1}{|B_j|} \left| \sum_{i \in B_j} A_{ik} - \mu_j(k) \right|
\]

\[
\leq \frac{1}{|B_j|} \left( | \sum_{i \in B_j \cap B_j^c} (A_{ik} - \mu_j(k)) | + | \sum_{i \in B_j \cap B_j^c} (A_{ik} - \mu_j(k)) | \right)
\]

\[
\leq C \log |B_j| \sqrt{|B_j \cap B_j^c|} \sqrt{|B_j|} + \frac{|B_j \cap B_j^c|}{|B_j|} (\mu_{gap} + \sigma \log |B_j|), \tag{9.11}
\]

holds with probability at least \( 1 - \frac{1}{|B_j|} \), where \( \mu_{gap} := \max_{k \in [n]} \max_{j \neq j_2} | \mu_j(k) - \mu_{j_2}(k) | \) and \( B_j^c \) is the complement of \( B_j^* \). By assumption A1, it can be checked that \( \delta_j \leq 2\sigma \) when \( |B_j| \gg n \). Here, (9.11) holds since that \( \sum_{i \in B_j \cap B_j^c} (A_{ik} - \mu_j(k)) \) is a Gaussian random variable which is \( \mathcal{O}_p(\sqrt{|B_j \cap B_j^c|}) \). For each \( i \in B_j \cap B_j^c \), the difference between \( A_{ik} \) and \( \mu_j(k) \) is at most \( | \mu_j(k) - \mu_j(k) | \) plus noise term, which is further bounded by \( \mu_{gap} + \sigma \log |B_j| \).

We next compute the lower bound of \( \sum_{i \in B_j} z_i^2 \). By use of (9.7), we get

\[
\sum_{i \in B_j} z_i^2 \geq \sum_{i \in B_j \cap B_j^c} z_i^2 = |B_j \cap B_j^*| m_j^2 - 2m_j \sigma \| r \|_2 \sqrt{|B_j \cap B_j^*| n \log n} + (\sqrt{|B_j \cap B_j^*| - 2\sqrt{n}})^2 \| r \|_2^2 \sigma^2. \tag{9.12}
\]

We further calculate the upper bound of \( \max_{i \in B_j} | z_i | \).

\[
\max_{i \in B_j} | z_i | = \max \left\{ \max_{i \in B_j \cap B_j^c} | z_i |, \max_{i \in B_j \cap B_j^c} | z_i | \right\}
\]

\[
\leq \max \left\{ m_j + 2 \log |B_j| \sqrt{n} \| r \|_2 \sigma, \max_{i \in B_j \cap B_j^c} | z_i | \right\}
\]

\[
= \max \left\{ m_j + 2 \log |B_j| \sqrt{n} \| r \|_2 \sigma, \max_{i \in B_j \cap B_j^c} | c_j^T r - \mu_j^T r + (a_i - c_j)^T r | \right\}
\]

\[
\leq \max \left\{ m_j + 2 \log |B_j| \sqrt{n} \| r \|_2 \sigma, \sqrt{n} \delta_j \| r \| + m_j + \max_{i \in B_j \cap B_j^c} | (a_i - c_j)^T r | \right\}
\]

\[
\leq \max \left\{ m_j + 2 \log |B_j| \sqrt{n} \| r \|_2 \sigma, \sqrt{n} \delta_j \| r \| + m_j + \max C | (a_i - c_j)^T r | \right\}
\]

\[
\leq \max \left\{ m_j + 2 \log |B_j| \sqrt{n} \| r \|_2 \sigma, \sqrt{n} (C + 1) \delta_j \| r \| + m_j + 2C \log |B_j^*| \sqrt{n} \| r \|_2 \sigma \right\}
\]

\[
\leq m_j + \sqrt{n} (C + 1) \delta_j \| r \| + 2C \log |B_j^*| \sqrt{n} \| r \|_2 \sigma. \tag{9.13}
\]

By (9.12) and (9.13), when \( |B_j| \gg n \), we get

\[
\frac{\sqrt{\sum_{i \in B_j} z_i^2}}{\max_{i \in B_j} | z_i |} \geq \frac{|B_j \cap B_j^*| m_j^2 - 2m_j \sigma \| r \|_2 \sqrt{|B_j \cap B_j^*| n \log n} + (\sqrt{|B_j \cap B_j^*| - 2\sqrt{n}})^2 \| r \|_2^2 \sigma^2}{m_j + \sqrt{n} (C + 1) \delta_j \| r \| + 2C \log |B_j^*| \sqrt{n} \| r \|_2 \sigma}
\]

\[
\geq \frac{c \sqrt{|B_j \cap B_j^*|} \sqrt{m_j^2 + \| r \|_2^2 \sigma^2}}{\max \{| C \log |B_j^*|, C + 1 \} \sqrt{n} \sqrt{m_j^2 + \| r \|_2^2 \sigma^2 + \| r \|_2^2 \delta_j^2}}
\]

\[
\geq c \sqrt{|B_j|} / (C (\log |B_j^*| + 1) \sqrt{n}) \tag{9.14}
\]
by adjusting some universal constant $c$. Notice that the above results hold for all $r$. This implies that

$$\frac{\|\nabla f\|_2}{\|\nabla f\|_\infty} \geq \frac{c}{C(\log |B_j| + 1)\sqrt{n}}. \tag{9.15}$$

This further gives

$$\|\nabla f(x')\|_{B, \infty} \geq \frac{c}{C\max_j(\log |B_j^*| + 1)\sqrt{n}}\|\nabla f(x')\|_{\infty},$$

When $\max_j |B_j| \geq d/k \gg n$, there is huge improvement in lower bound, from $1/(\max_j \sqrt{|B_j|})$ to $1/(C(\log |B_j^*| + 1)\sqrt{n})$.

**Proof of Theorem 6.11**

Our proof follows the same pattern as the proof of ASCD (Lu et al., 2018).

**Lemma 9.1.** Define $s^{(t+1)} := y^{(t)} - \frac{1}{dL_{\max}} \nabla f(y^{(t)})$, then

$$\mathbb{E}_t f(x^{(t+1)}) \leq f(y^{(t)}) + \langle \nabla f(y^{(t)}), s^{(t+1)} - y^{(t)} \rangle + \frac{dL_{\max}}{2} \|s^{(t+1)} - y^{(t)}\|^2.$$  

**Proof.**

$$\begin{align*}
\mathbb{E}_t f(x^{(t+1)}) &\leq f(y^{(t)}) - \frac{1}{2L_{\max}} \mathbb{E}_t \langle \nabla_j f(y^{(t)}), \nabla_j f(y^{(t)}) \rangle^2, \\
&\leq f(y^{(t)}) - \frac{1}{2dL_{\max}} \|\nabla f(y^{(t)})\|^2, \\
&= f(y^{(t)}) + \langle \nabla f(y^{(t)}), s^{(t+1)} - y^{(t)} \rangle + \frac{dL_{\max}}{2} \|s^{(t+1)} - y^{(t)}\|^2.
\end{align*} \tag{9.16}$$

Here, (9.17) holds due to the following fact,

$$\begin{align*}
\mathbb{E}_t |\nabla_j f(y^{(t)})|^2 &= \mathbb{E}_t \max_j |\nabla_j f(y^{(t)})|^2 \\
&\geq \sum_j \frac{|B_j|}{d} \mathbb{E}_t |\nabla_j f(y^{(t)})|^2 \\
&= \sum_j \frac{|B_j|}{d} \mathbb{E}_t |\nabla f(y^{(t)})|^2 \\
&= \frac{1}{d} \|\nabla f(y^{(t)})\|^2.
\end{align*}$$

**Lemma 9.2.** Define $t^{(t+1)} := z^{(t)} - \frac{1}{n} L^{-1}_{\max} \nabla f(y^{(t)})$. Or equivalently,

$$t^{(t+1)} := \arg \min_z \langle \nabla f(y^{(t)}), z - z^{(t)} \rangle + \frac{d\theta_t L_{\max}}{2} \|z - z^{(t)}\|^2.$$

Then

$$\begin{align*}
\mathbb{E}_t f(x^{(t+1)}) &\leq (1 - \theta_t) f(x^{(t)}) + \theta_t f(x^*) + \frac{nL_{\max}\theta_t^2}{2} \|x^* - z^{(t)}\|^2 + \frac{nL_{\max}\theta_t^2}{2} \|x^* - t^{(t+1)}\|^2.
\end{align*}$$
Proof. By Lemma 9.1, we have
\[
\mathbb{E}_t f(x^{(t)}) \leq f(y^{(t)}) + \langle \nabla f(y^{(t)}), s^{(t+1)} - y^{(t)} \rangle + \frac{dL_{\max}}{2} \| s^{(t+1)} - y^{(t)} \|^2 \\
= f(y^{(t)}) + \theta_t \langle \nabla f(y^{(t)}), t^{(t+1)} - z^{(t)} \rangle + \frac{dL_{\max} \theta_t}{2} \| t^{(t+1)} - z^{(t)} \|^2 \\
= f(y^{(t)}) + \theta_t \langle \nabla f(y^{(t)}), x^* - z^{(t)} \rangle + \frac{dL_{\max} \theta_t}{2} \| x^* - z^{(t)} \|^2 - \frac{dL_{\max} L \theta_t}{2} \| x^* - (t^{(t+1)}) \| \\
= (1 - \theta_t) f(y^{(t)}) + \theta_t f(x^*) + \frac{nL_{\max} \theta_t^2}{2} \| x^* - z^{(t)} \|^2 - \frac{nL_{\max} \theta_t^2}{2} \| x^* - t^{(t+1)} \|^2 \\
\leq (1 - \theta_t) f(x^{(t)}) + \theta_t f(x^*) + \frac{nL_{\max} \theta_t^2}{2} \| x^* - z^{(t)} \|^2 - \frac{nL_{\max} \theta_t^2}{2} \| x^* - t^{(t+1)} \|^2.
\]
\(\square\)

Lemma 9.3. \(\frac{dL_{\max}}{2} \| x^* - z^{(t)} \|^2 - \frac{dL_{\max}}{2} \| x^* - t^{(t+1)} \|^2 = \frac{d^2L_{\max}}{2} \| x^* - z^{(t)} \|^2 - \frac{d^2L_{\max}}{2} \| x^* - z^{(t+1)} \|^2.\)

Proof.
\[
\frac{dL_{\max}}{2} \| x^* - z^{(t)} \|^2 - \frac{dL_{\max}}{2} \| x^* - t^{(t+1)} \|^2 = \frac{dL_{\max}}{2} \| t^{(t+1)} - z^{(t)}, 2x^* - 2z^{(t)} \| - \frac{dL_{\max}}{2} \| t^{(t+1)} - z^{(t)} \|^2 \\
= \frac{d^2L_{\max}}{2} \| z^{(t+1)} - z^{(t)} \| - \frac{d^2L_{\max}}{2} \| x^* - z^{(t+1)} \|^2. \tag{9.19}
\]
\[
= \frac{d^2L_{\max}}{2} \| x^* - z^{(t)} \|^2 - \frac{d^2L_{\max}}{2} \| x^* - z^{(t+1)} \|^2. \tag{9.20}
\]
Here, we use the fact that \(t^{(t+1)} - z^{(t)} = dE_{j_2}[z^{(t+1)} - z^{(t)}]\) and \(\| t^{(t+1)} - z^{(t)} \|^2 = dE_{j_2}[z^{(t+1)} - z^{(t)}].\) \(\square\)

Proof of Theorem 6.11. By Lemma 9.2 and Lemma 9.3, we obtain that
\[
\mathbb{E}_t f(x^{(t+1)}) \leq (1 - \theta_t) f(x^{(t)}) + \frac{d^2L_{\max}}{2} \| x^* - z^{(t)} \|^2 - \frac{d^2L_{\max}}{2} \| x^* - z^{(t+1)} \|^2.
\]
By using \(\frac{1 - \theta_{t+1}}{\theta_{t+1}^2} = \frac{1}{\theta_t^2},\) we arrive at:
\[
\frac{1 - \theta_{t+1}}{\theta_{t+1}^2} (\mathbb{E}_t f(x^{(t+1)}) - f(x^*)) + \frac{d^2L_{\max}}{2} \| x^* - z^{(t+1)} \|^2 \leq \frac{1 - \theta_t}{\theta_t^2} (f(x^t) - f(x^*)) + \frac{d^2L_{\max}}{2} \| x^* - z^t \|^2.
\]
We use \(\mathbb{E}^t\) to denote taking expectation over everything up to \(t,\) it follows that
\[
\mathbb{E}^{t+1} \left[ \frac{1 - \theta_{t+1}}{\theta_{t+1}^2} (f(x^{(t+1)}) - f(x^*)) + \frac{d^2L_{\max}}{2} \| x^* - z^{(t+1)} \|^2 \right] \leq \mathbb{E}^t \left[ \frac{1 - \theta_t}{\theta_t^2} (f(x^t) - f(x^*)) + \frac{d^2L_{\max}}{2} \| x^* - z^t \|^2 \right].
\]
By above recursive formula, we get
\[
\mathbb{E}^{t+1} \left[ \frac{1 - \theta_{t+1}}{\theta_{t+1}^2} (f(x^{(t+1)}) - f(x^*)) \right] \leq \mathbb{E}^0 \left[ \frac{1 - \theta_0}{\theta_0^2} (f(x^0) - f(x^*)) + \frac{d^2L_{\max}}{2} \| x^* - z^0 \|^2 \right] \tag{9.21}
\]
\[
= \frac{d^2L_{\max}}{2} \| x^* - z^0 \|^2. \tag{9.22}
\]
It is easy to check that \(\theta_t \leq \frac{\theta_t^2}{\theta_t^2},\) then it gives
\[
\mathbb{E}^t \left[ f(x^{(t)}) - f(x^*) \right] \leq \frac{\theta_t^2}{1 - \theta_t} \frac{d^2L_{\max}}{2} \| x^* - z^0 \|^2 = \frac{d^2L_{\max}}{2} \| x^* - z^0 \|^2 \leq \frac{2d^2L_{\max}}{(t+1)^2} \| x^* - x^0 \|^2. \tag{9.23}
\]
\(\square\)
High probability error bounds

The following high probability error bounds can be obtained by using (Richtárik and Takác, 2014, Theorem 1)

**Corollary 9.4.** Denote \(x^{(t)}\) as the iterate generated from Algorithm 2. For \(f\) that is \(\mu_1\) and \(\mu_2\) strongly convex with respect for 1 and 2-norm. Let

\[
\eta := \inf_{x \in \mathbb{R}^d} \max \left\{ \frac{\mu_2}{L_{\max}} \|\nabla f(x)\|_{B,\infty}, \frac{\mu_1}{L_{\max}} \|\nabla f(x)\|_{B,\infty}^2 \right\},
\]

then with probability at least \(1 - \beta\), we have

\[
\mathbb{E}[f(x^{(t)})] - f^* \leq \exp\left(\frac{-t\eta}{\beta}\right) \left( f(x^0) - f^* \right).
\]

Using Equation (9.4) and (Richtárik and Takác, 2014, Theorem 1), we can immediately get the following.

**Corollary 9.5.** Denote \(x^{(t)}\) as the iterate generated from Algorithm 2. For convex objective \(f\), with probability at least \(1 - \beta\),

\[
\mathbb{E}[f(x^{(t)})] - f^* = \mathcal{O} \left( \frac{L_{\max} D^2}{\eta^2 t} \left( 1 + \log \left( \frac{1}{\beta} \right) \right) \right),
\]

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

10 PROXIMAL HYBRIDCD

Proximal hybridCD is a proximal-gradient variant of hybridCD. It aims to solve the composite problem

\[
\min_{x \in \mathbb{R}^d} f(x) + \sum_{i=1}^d g_i(x_i).
\]

The detailed algorithm is shown in Algorithm 4, where

\[
\text{prox}_g(y) := \arg\min_{x \in \mathbb{R}^d} \frac{1}{2} \|y - x\|^2 + g(y)
\]

is the standard definition of proximal operator and the GS-s rule is the greedy selection rule extended to composite problem, see Nutini et al. (2015) for more details.

---

**Algorithm 4 Proximal hybrid coordinate descent**

| Input: \(x^{(0)}\), \(B = \{B_j\}_{i=1}^k\). |
| for \(t = 0, 1, 2, \ldots\) do |
| \(I = \emptyset\) |
| for \(j = 1, 2, \ldots, k\) do |
| [Random rule] uniformly randomly choose \(i_j \in B_j\) and let \(I = I \cup \{i_j\}\) |
| end for |
| [GS-s rule] \(i \in \arg\max_{j \in I} \{\min_{x \in g_i} \|\nabla_j f(x^{(t)}) + s\}\} |
| \(x^{(t+1)} = \text{prox}_{\frac{1}{L_i g_i}} \left( x^{(t)} - \frac{1}{L_i} \nabla f_i(x^{(t)}) e_i \right) \) |
| end for |