StingyCD: Safely Avoiding Wasteful Updates in Coordinate Descent

Tyler B. Johnson [1] Carlos Guestrin [1]

Abstract

Coordinate descent (CD) is a scalable and simple algorithm for solving many optimization problems in machine learning. Despite this fact, CD can also be very computationally wasteful. Due to sparsity in sparse regression problems, for example, often the majority of CD updates result in no progress toward the solution. To address this inefficiency, we propose a modified CD algorithm named "StingyCD." By skipping over many updates that are guaranteed to not decrease the objective value, StingyCD significantly reduces convergence times. Since StingyCD only skips updates with this guarantee, however, StingyCD does not fully exploit the problem's sparsity. For this reason, we also propose StingyCD+, an algorithm that achieves further speed-ups by skipping updates more aggressively. Since StingyCD and StingyCD+ rely on simple modifications to CD, it is also straightforward to use these algorithms with other approaches to scaling optimization. In empirical comparisons, StingyCD and StingyCD+ improve convergence times considerably for ℓ_1-regularized optimization problems.

1. Introduction

Known to be simple and fast, coordinate descent is a highly popular algorithm for training machine learning models. For ℓ_1-regularized loss minimization problems, such as the Lasso (Tibshirani, 1996), CD iteratively updates just one weight variable at a time. As it turns out, these small yet inexpensive updates efficiently lead to the desired solution. Another attractive property of CD is its lack of parameters that require tuning, such as a learning rate.

Due to its appeal, CD has been researched extensively in recent years. This includes theoretical (Nesterov, 2012; Shalev-Shwartz & Tewari, 2011) and more applied (Fan et al., 2008; Friedman et al., 2010) contributions. Many works also consider scaling CD using parallelism (Bradley et al., 2011; Richtárik & Takáč, 2016). For surveys of research on CD, see (Wright, 2015) or (Shi et al., 2016).

Despite its popularity, CD has a significant drawback: in many applications, the majority of coordinate updates yield no progress toward convergence. In sparse regression, most entries of the optimal weight vector equal zero. When CD updates these weights during optimization, the weights often equal zero both before and after they are updated. This is immensely wasteful! Computing these "zero updates" requires time yet leaves the iterate unchanged.

In this work, we propose StingyCD, an improved CD algorithm for sparse optimization and linear SVM problems. With minimal added overhead, StingyCD identifies many coordinate updates that are guaranteed to result in no change to the current iterate. By skipping over these zero updates, StingyCD obtains much faster convergence times.

StingyCD is related to safe screening tests (El Ghaoui et al., 2012), which for Lasso problems, guarantee some weights equal zero *at the solution*. The algorithm can subsequently ignore screened weights for the remainder of optimization. Unfortunately, for screening to be effective, a good approximate solution must already be available. For this reason, screening often has little impact until convergence is near (Johnson & Guestrin, 2016).

By identifying zero *updates* rather than zero-valued weights at the solution, StingyCD drastically improves convergence times compared to safe screening. At the same time, we find that skipping only updates that are guaranteed to be zero is limiting. For this reason, we also propose StingyCD+, an algorithm that estimates a probability that each update is zero. By also skipping updates that are likely zero, StingyCD+ achieves even greater speed-ups.

StingyCD and StingyCD+ require only simple changes to CD. Thus, we can combine these algorithms with other improvements to CD. In this work, we apply StingyCD+ to proximal Newton and working set algorithms. In both cases, incorporating StingyCD+ leads to efficiency improvements, demonstrating that "stingy updates" are a ver-

[1]University of Washington, Seattle, WA. Correspondence to: Tyler Johnson <tbjohns@washington.edu>, Carlos Guestrin <guestrin@cs.washington.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

Algorithm 1 Coordinate descent for solving (P)

```
initialize $\mathbf{x}^{(0)} \leftarrow \mathbf{0}^m$; $\mathbf{r}^{(0)} \leftarrow \mathbf{b}$
for $t = 1, 2, \ldots T$ do
    $i \leftarrow$ get_next_coordinate()
    $\delta \leftarrow \max\left\{-x_i^{(t-1)}, \frac{\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} \rangle - \lambda}{\|\mathbf{A}_i\|^2}\right\}$
    $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} + \delta \mathbf{e}_i$
    $\mathbf{r}^{(t)} \leftarrow \mathbf{r}^{(t-1)} - \delta \mathbf{A}_i$
return $\mathbf{x}^{(T)}$
```

satile and effective tool for scaling CD algorithms.

2. StingyCD for nonnegative Lasso

We introduce StingyCD for solving the problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^m}{\text{minimize}} & f(\mathbf{x}) := \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda \langle \mathbf{1}, \mathbf{x} \rangle \\ \text{s.t.} & \mathbf{x} \geq 0 \end{array} . \tag{P}$$

(P) is known as the "nonnegative Lasso." Importantly, applications of StingyCD are not limited to (P). In §4, we explain how to apply StingyCD to general Lasso and sparse logistic regression objectives as well as SVM problems.

In (P), $\mathbf{A}$ is an $n \times m$ design matrix, while $\mathbf{b} \in \mathbb{R}^n$ is a labels vector. Solving (P) results in a set of learned weights, which define a linear predictive model. The right term in the objective—commonly written as $\lambda \|\mathbf{x}\|_1$ for Lasso problems without the nonnegativity constraint—is a regularization term that encourages the weights to have small value. The parameter $\lambda > 0$ controls the impact of the regularization term. Due to the nonnegativity constraint, a solution to (P) is *sparse* for sufficiently large λ. That is, the majority of entries in a solution to (P) have value zero.

Advantages of sparsity include reduced resources needed at test time, more interpretable models, and statistical efficiency (Wainwright, 2009). In this paper, we propose an algorithm that exploits sparsity for efficient optimization.

2.1. Coordinate descent

Coordinate descent (CD) is a popular algorithm for solving (P). Algorithm 1 defines a CD algorithm for this problem. During iteration t, a coordinate $i \in [m]$ is selected, usually at random or in round-robin fashion. The ith entry in $\mathbf{x}^{(t)}$ is updated via $x_i^{(t)} \leftarrow x_i^{(t-1)} + \delta$, while remaining weights do not change. The value of δ is chosen to maximally decrease the objective subject to the nonnegativity constraint. Defining the residuals vector $\mathbf{r}^{(t-1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(t-1)}$, we can write δ as

$$\delta = \max\left\{-x_i^{(t-1)}, \frac{1}{\|\mathbf{A}_i\|^2}\left(\left\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} \right\rangle - \lambda\right)\right\} . \tag{1}$$

Iteration t requires $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ time, where $\mathrm{NNZ}(\mathbf{A}_i)$ is the number of nonzero entries in column $\mathbf{A}_i$. Bottleneck operations are computing the dot product $\left\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} \right\rangle$ and updating $\mathbf{r}^{(t)}$. We note implementations typically compute $\|\mathbf{A}_i\|^2$ once and then cache this value for later updates.

2.2. Wasteful updates in coordinate descent

Because of the nonnegativity constraint and regularization penalty in (P), often $x_i^{(t-1)} = 0$ in Algorithm 1. In this case, if $\mathbf{r}^{(t-1)}$ lies outside of the "active update" region

$$\mathcal{A}_i = \{\mathbf{r} : \langle \mathbf{A}_i, \mathbf{r} \rangle - \lambda > 0\} ,$$

meaning $\left\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} \right\rangle - \lambda \leq 0$, then (1) implies that $\delta = 0$. In this scenario, weight i equals zero at the beginning and end of iteration t. When solutions to (P) are sufficiently sparse, these "zero updates" account for the majority of iterations in naive CD algorithms. Computing these updates is very wasteful! Each zero update requires $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ time yet results in no progress toward convergence.

2.3. Stingy updates

Our proposed algorithm, StingyCD, improves convergence times for CD by "skipping over" many zero updates. Put differently, StingyCD computes some zero updates in $\mathcal{O}(1)$ time rather than $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ time by guaranteeing $\delta = 0$ without computing this quantity via (1).

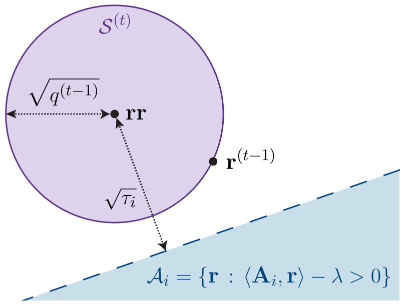
We saw in §2.2 that sufficient conditions for $\delta = 0$ are (i) $x_i^{(t-1)} = 0$ and (ii) $\mathbf{r}^{(t-1)} \notin \mathcal{A}_i$. Since directly testing the second condition requires $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ time, simply checking these conditions does not lead to a useful method for quickly guaranteeing $\delta = 0$.

The insight that enables StingyCD is that we can relax the condition $\mathbf{r}^{(t-1)} \notin \mathcal{A}_i$ to form a condition that is testable in constant time. This relaxation depends on a region $\mathcal{S}^{(t)}$ for which $\mathbf{r}^{(t-1)} \in \mathcal{S}^{(t)}$. In particular, $\mathcal{S}^{(t)}$ is a ball:

$$\mathcal{S}^{(t)} = \left\{\mathbf{r} : \|\mathbf{r} - \mathbf{rr}\|^2 < q^{(t-1)}\right\} ,$$
$$\text{where } q^{(t-1)} = \left\|\mathbf{r}^{(t-1)} - \mathbf{rr}\right\|^2 .$$

Above, $\mathbf{rr}$ is a "reference residuals" vector—a copy of the residuals from a previous iteration. Formally, $\mathbf{rr} = \mathbf{r}^{(t-k)}$ for some $k \geq 1$ (to be defined more precisely later). Note that $\mathbf{r}^{(t-1)}$ lies on the boundary of $\mathcal{S}^{(t)}$.

At any iteration t such that $x_i^{(t-1)} = 0$, StingyCD considers whether $\mathcal{S}^{(t)} \cap \mathcal{A}_i = \emptyset$ before computing δ. If this condition is true, it is guaranteed that $\delta = 0$, and StingyCD continues to iteration $t+1$ without computing δ directly. We illustrate this concept in Figure 1. Defining $g_i = -\langle \mathbf{A}_i, \mathbf{rr} \rangle + \lambda$, we



$\mathcal{S}^{(t)}$
$\sqrt{q^{(t-1)}}$
$\mathbf{rr}$
$\mathbf{r}^{(t-1)}$
$\sqrt{\tau_i}$
$\mathcal{A}_i = \{\mathbf{r} : \langle \mathbf{A}_i, \mathbf{r} \rangle - \lambda > 0\}$

Figure 1. **Geometry of StingyCD.** At iteration t of CD, if $x_i^{(t-1)} = 0$ and $\mathbf{r}^{(t-1)} \notin \mathcal{A}_i$, then $\delta = 0$. In this case, computing δ is wasteful because the "update" makes no change to $\mathbf{x}^{(t-1)}$. StingyCD skips over many zero updates by establishing a region $\mathcal{S}^{(t)}$ for which $\mathbf{r}^{(t-1)} \in \mathcal{S}^{(t)}$. If $\mathcal{S}^{(t)} \cap \mathcal{A}_i = \emptyset$, it is guaranteed that $\delta = 0$, and StingyCD continues to iteration $t+1$ without computing δ directly. In the illustration, StingyCD successfully guarantees $\delta = 0$, since $q^{(t-1)} \leq \tau_i$. In contrast, StingyCD would compute δ directly if $q^{(t-1)} > \tau_i$. We note $\sqrt{\tau_i}$ is well-defined in the illustration; since $\mathbf{rr} \notin \mathcal{A}_i$, we have $\tau_i \geq 0$.

can simplify the condition $\mathcal{S}^{(t)} \cap \mathcal{A}_i = \emptyset$ as follows:

$$\begin{aligned} \mathcal{S}^{(t)} \cap \mathcal{A}_i = \emptyset \quad &\Leftrightarrow \quad \max_{\mathbf{r} \in \mathcal{S}^{(t)}} \langle \mathbf{A}_i, \mathbf{r} \rangle - \lambda \leq 0 \\ &\Leftrightarrow \quad -g_i + \|\mathbf{A}_i\| \sqrt{q^{(t-1)}} \leq 0 \\ &\Leftrightarrow \quad q^{(t-1)} \leq \operatorname{sign}(g_i) \tfrac{g_i^2}{\|\mathbf{A}_i\|^2} := \tau_i \,. \end{aligned}$$

Thus, if $q^{(t-1)} \leq \tau_i$, then $\mathbf{r}^{(t-1)} \notin \mathcal{A}_i$ (implying $\delta = 0$ if also $x_i^{(t-1)} = 0$). We note that τ_i can take positive or negative value, depending on if $\mathbf{rr} \in \mathcal{A}_i$. If $\mathbf{rr} \notin \mathcal{A}_i$, then $g_i \geq 0$, which implies $\tau_i \geq 0$. However, if $\mathbf{rr} \in \mathcal{A}_i$, then $\tau_i < 0$, and since $q^{(t-1)}$ is nonnegative, it cannot be true that $q^{(t-1)} \leq \tau_i$—StingyCD does not skip over coordinate i in this case. Thus, the magnitude of τ_i is not significant to StingyCD when $\tau_i < 0$, though this magnitude has greater importance for StingyCD+ in §3.

Importantly, the condition $q^{(t-1)} \leq \tau_i$ can be tested with minimal overhead by (i) updating $\mathbf{rr}$ only occasionally, (ii) precomputing $\langle \mathbf{A}_i, \mathbf{rr} \rangle$ and τ_i for all i whenever $\mathbf{rr}$ is updated, and (iii) caching the value of $q^{(t-1)}$, which is updated appropriately after each nonzero coordinate update.

2.4. StingyCD definition and guarantees

StingyCD is defined in Algorithm 2. StingyCD builds upon Algorithm 1 with three simple changes. First, during some iterations, StingyCD updates a reference residuals vector, $\mathbf{rr} \leftarrow \mathbf{r}^{(t-1)}$. When $\mathbf{rr}$ is updated, StingyCD also computes a thresholds vector, $\boldsymbol{\tau}$. This requires evaluating $\langle \mathbf{A}_i, \mathbf{rr} \rangle$ for all columns in $\mathbf{A}$. While updating $\mathbf{rr}$ is relatively costly, more frequent updates to $\mathbf{rr}$ result in greater computational savings due to skipped updates.

Algorithm 2 StingyCD for solving (P)

```
initialize $\mathbf{x}^{(0)} \leftarrow \mathbf{0}^m$; $\mathbf{r}^{(0)} \leftarrow \mathbf{b}$; $\mathbf{rr} \leftarrow \mathbf{r}^{(0)}$;
        $q^{(0)} \leftarrow 0$; $\boldsymbol{\tau} \leftarrow$ compute_thresholds($\mathbf{x}^{(0)}$)
for $t = 1, 2, \ldots T$ do
    # Update reference residuals on occasion:
    if should_update_reference() then
        $\mathbf{rr} \leftarrow \mathbf{r}^{(t-1)}$
        $\boldsymbol{\tau} \leftarrow$ compute_thresholds($\mathbf{x}^{(t-1)}$)
        $q^{(t-1)} \leftarrow 0$

    $i \leftarrow$ get_next_coordinate()

    if $q^{(t-1)} \leq \tau_i$ and $x_i^{(t-1)} = 0$ then
        # Skip update:
        $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)}$; $\mathbf{r}^{(t)} \leftarrow \mathbf{r}^{(t-1)}$; $q^{(t)} \leftarrow q^{(t-1)}$
        continue

    # Perform coordinate update:
    $\delta \leftarrow \max\left\{-x_i^{(t-1)}, \frac{\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} \rangle - \lambda}{\|\mathbf{A}_i\|^2}\right\}$
    $\mathbf{x}^{(t)} \leftarrow \mathbf{x}^{(t-1)} + \delta \mathbf{e}_i$
    $\mathbf{r}^{(t)} \leftarrow \mathbf{r}^{(t-1)} - \delta \mathbf{A}_i$
    $q^{(t)} \leftarrow q^{(t-1)} - 2\delta \left\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} - \mathbf{rr} \right\rangle + \delta^2 \|\mathbf{A}_i\|^2$
return $\mathbf{x}^{(T)}$

function compute_thresholds($\mathbf{x}$)
    initialize $\boldsymbol{\tau} \leftarrow \mathbf{0}^m$
    for $i \in [m]$ do
        $g_i \leftarrow \langle \mathbf{A}_i, \mathbf{Ax} - \mathbf{b} \rangle + \lambda$
        $\tau_i \leftarrow \operatorname{sign}(g_i) \frac{g_i^2}{\|\mathbf{A}_i\|^2}$
    return $\boldsymbol{\tau}$
```

The second change to CD is that StingyCD tracks the quantity $q^{(t)} = \left\|\mathbf{r}^{(t)} - \mathbf{rr}\right\|^2$. After each update to $\mathbf{rr}$, StingyCD sets $q^{(t)}$ to 0. After each nonzero residuals update, $\mathbf{r}^{(t)} \leftarrow \mathbf{r}^{(t-1)} - \delta \mathbf{A}_i$, StingyCD makes a corresponding update to $q^{(t)}$. Importantly, the quantities required for this update to $q^{(t)}$—$\|\mathbf{A}_i\|^2$, $\langle \mathbf{A}_i, \mathbf{r}^{(t-1)} \rangle$, $\langle \mathbf{A}_i, \mathbf{rr} \rangle$, and δ—have all been computed earlier by the algorithm. Thus, by caching these values, updating $q^{(t)}$ requires negligible time.

The final modification to CD is StingyCD's use of stingy updates. Before computing δ during each iteration t, StingyCD checks whether $q^{(t-1)} \leq \tau_i$ and $x_i^{(t-1)} = 0$. If both are true, StingyCD continues to the next iteration without computing δ. The threshold τ_i is computed after each update to $\mathbf{rr}$. If $\mathbf{rr} \notin \mathcal{A}_i$, the value of τ_i equals the squared distance between $\mathbf{rr}$ and $\mathcal{A}_i$. If $\mathbf{rr} \in \mathcal{A}_i$, this quantity is the negative squared distance between $\mathbf{rr}$ and $\mathcal{A}_i^C$.

StingyCD's choice of $\boldsymbol{\tau}$ ensures that each skipped update is "safe." We formalize this concept with our first theorem:

Theorem 2.1 (Safeness of StingyCD)**.** *In Algorithm 2, every skipped update would, if computed, result in $\delta = 0$.*

That is, if $q^{(t-1)} \leq \tau_i$ and $x_i^{(t-1)} = 0$, then

$$\max\left\{-x_i^{(t-1)}, \frac{\langle \mathbf{A}_i, \mathbf{b} - \mathbf{A}\mathbf{x}^{(t-1)}\rangle - \lambda}{\|\mathbf{A}_i\|^2}\right\} = 0\,.$$

We prove Theorem 2.1 in Appendix A.

Theorem 2.1 is useful because it guarantees that although StingyCD skips many updates, CD and StingyCD have identical weight vectors for all iterations (assuming each algorithm updates coordinates in the same order). Our next theorem formalizes the notion that these skipped updates come nearly "for free." We prove this result in Appendix B.

Theorem 2.2 (Per iteration time complexity of StingyCD)**.** *Algorithm 2 can be implemented so that iteration t requires*

- *Less time than an identical iteration of Algorithm 1 if $q^{(t-1)} \leq \tau_i$ and $x_i^{(t-1)} = 0$ (the update is skipped) and $\mathbf{rr}$ is not updated. Specifically, StingyCD requires $\mathcal{O}(1)$ time, while CD requires $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ time.*
- *The same amount of time (up to an $\mathcal{O}(1)$ term) as a CD iteration if the update is not skipped and $\mathbf{rr}$ is not updated. In particular, both algorithms require the same number of $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ operations.*
- *More time than a CD iteration if $\mathbf{rr}$ is updated. In this case, StingyCD requires $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}))$ time.*

Note StingyCD requires no more computation than CD for nearly all iterations (and often much less). However, the cost of updating $\mathbf{rr}$ is not negligible. To ensure updates to $\mathbf{rr}$ do not overly impact convergence times, we schedule reference updates so that StingyCD invests less than 20% of its time in updating $\mathbf{rr}$. Specifically, StingyCD first updates $\mathbf{rr}$ after the second epoch and records the time that this update requires. Later on, $\mathbf{rr}$ is updated each time an additional 5x of this amount of time has passed.

3. Skipping extra updates with StingyCD+

As we will see in §6, StingyCD can significantly reduce convergence times. However, StingyCD is also limited by the requirement that only updates *guaranteed* to be zero are skipped. In cases where $q^{(t-1)}$ is only slightly greater than τ_i, intuition suggests that these updates will likely be zero too. Perhaps StingyCD should skip these updates as well.

In this section, we propose StingyCD+, an algorithm that also skips many updates that are *not* guaranteed to be zero. To do so, StingyCD+ adds two components to StingyCD: (i) a computationally inexpensive model of the probability that each update is zero, and (ii) a decision rule that applies this model to determine whether or not to skip each update.

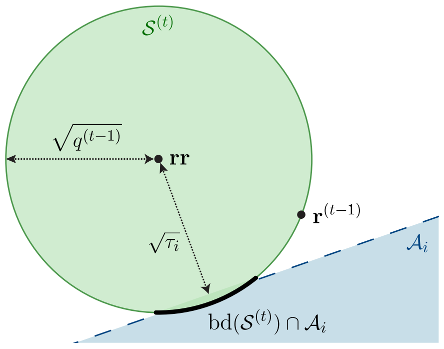


Figure 2. **Probability of a useful update.** StingyCD skips update t iff $q^{(t-1)} \leq \tau_i$ and $x_i^{(t-1)} = 0$, which guarantee $\delta = 0$. To skip more updates, StingyCD+ applies the intuition that if $q^{(t-1)}$ is only slightly larger than τ_i, it is unlikely that $\mathbf{r}^{(t-1)} \in \mathcal{A}_i$, making it unlikely that $\delta \neq 0$. To do so, StingyCD+ models the probability that $\delta \neq 0$ during iteration t, denoted $P(U^{(t)})$. Assuming $x_i^{(t-1)} = 0$ in the illustrated scenario, StingyCD+ computes $P(U^{(t)})$ by dividing the length of the black arc $(\mathrm{bd}(\mathcal{S}^{(t)}) \cap \mathcal{A}_i)$ by the circumference of $\mathcal{S}^{(t)}$.

3.1. Modeling the probability of nonzero updates

During iteration t of StingyCD, suppose $x_i^{(t-1)} = 0$ but $\tau_i < q^{(t-1)}$. StingyCD does not skip update t. For now, we also assume $\tau_i > -q^{(t-1)}$ (otherwise we can guarantee $\mathbf{r}^{(t-1)} \in \mathcal{A}_i$, which implies $\delta \neq 0$). Let $U^{(t)}$ be a variable that is true if $\delta \neq 0$—update t is *useful*—and false otherwise. From the algorithm's perspective, it is uncertain whether $U^{(t)}$ is true or false when iteration t begins. Whether or not $U^{(t)}$ is true depends on whether $\mathbf{r}^{(t-1)} \in \mathcal{A}_i$, which requires $\mathcal{O}(\mathrm{NNZ}(\mathbf{A}_i))$ time to test. This computation will be wasteful if $U^{(t)}$ is false.

StingyCD+ models the probability that $U^{(t)}$ is true using information available to the algorithm. Specifically, $\mathbf{r}^{(t-1)}$ lies on the boundary of $\mathcal{S}^{(t)}$, which is a ball with center $\mathbf{rr}$ and radius $\sqrt{q^{(t-1)}}$. This leads to a simple assumption:

Assumption 3.1 (Distribution of $\mathbf{r}^{(t-1)}$)**.** *To model the probability $P(U^{(t)})$, StingyCD+ assumes $\mathbf{r}^{(t-1)}$ is uniformly distributed on the boundary of $\mathcal{S}^{(t)}$.*

By making this assumption, $P(U^{(t)})$ is tractable. In particular, we have the following equation for $P(U^{(t)})$:

Theorem 3.2 (Equation for $P(U^{(t)})$)**.** *Assume $x_i^{(t-1)} = 0$ and $\tau_i \in (-q^{(t-1)}, q^{(t-1)})$. Then Assumption 3.1 implies*

$$P(U^{(t)}) = \begin{cases} \frac{1}{2} I_{(1-\tau_i/q^{(t-1)})}\left(\frac{n-1}{2}, \frac{1}{2}\right) & \text{if } \tau_i \geq 0, \\ 1 - \frac{1}{2} I_{(1+\tau_i/q^{(t-1)})}\left(\frac{n-1}{2}, \frac{1}{2}\right) & \text{otherwise,} \end{cases}$$

where $I_x(a,b)$ is the regularized incomplete beta function.

Included in Appendix C, Theorem 3.2's proof calculates the probability that $\mathbf{r}^{(t-1)} \in \mathcal{A}_i$ by dividing the area of

$\mathcal{A}_i \cap \mathrm{bd}(\mathcal{S}^{(t)})$ by that of $\mathrm{bd}(\mathcal{S}^{(t)})$ (illustrated in Figure 2). This fraction is a function of the incomplete beta function since $\mathcal{A}_i \cap \mathrm{bd}(\mathcal{S}^{(t)})$ is a hyperspherical cap (Li, 2011).

Using Theorem 3.2, StingyCD+ can approximately evaluate $P(U^{(t)})$ efficiently using a lookup table. Specifically, for 128 values of $x \in (0, 1)$, our implementation defines an approximate lookup table for $I_x(\frac{n-1}{2}, \frac{1}{2})$ prior to iteration 1. Before update t, StingyCD+ computes $\tau_i / q^{(t-1)}$ and then finds an appropriate estimate of $P(U^{(t)})$ using the table. We elaborate on this procedure in Appendix D.

So far, $P(U^{(t)})$ models the probability that $\delta \neq 0$ when $\tau_i \in (-q^{(t-1)}, q^{(t-1)})$ and $x_i^{(t-1)} = 0$. We can also define $P(U^{(t)})$ for other $x_i^{(t-1)}$ and τ_i. When $x_i^{(t-1)} \neq 0$, we define $P(U^{(t)}) = 1$. If $\tau_i \geq q^{(t-1)}$ and $x_i^{(t-1)} = 0$ (the scenario in which StingyCD skips update t), we let $P(U^{(t)}) = 0$. If $\tau_i \leq -q^{(t-1)}$ and $x_i^{(t-1)} = 0$, we define $P(U^{(t)}) = 1$ (in this final case, we can show that $\mathcal{S}^{(t)} \subseteq \mathcal{A}_i$, which guarantees $\mathbf{r}^{(t-1)} \in \mathcal{A}_i$ and $\delta \neq 0$).

3.2. Decision rule for skipping updates

Given $P(U^{(t)})$, consider the decision of whether to skip update t. Let t_i^{last} denote the most recent iteration during which StingyCD+ updated (did not skip) coordinate i. If this has not yet occurred, define $t_i^{\text{last}} = 0$. We define the "delay" $D_i^{(t)}$ as the number of updates that StingyCD+ did not skip between iterations t_i^{last} and $t - 1$ inclusive.

Our intuition for StingyCD+ is that during iteration t, if $D_i^{(t)}$ is large and $U^{(t)}$ is true, then StingyCD+ should not skip update t. However, if $D_i^{(t)}$ is small and $U^{(t)}$ is true, the algorithm may want to skip the update in favor of updating a coordinate with larger delay. Finally, if $U^{(t)}$ is false, StingyCD+ should skip the update, regardless of $D_i^{(t)}$.

Based on this intuition, StingyCD+ skips update t if the "expected relevant delay," defined as $\mathbb{E}[D_i^{(t)} U^{(t)}]$, is small. That is, given a threshold $\xi^{(t)}$, StingyCD+ skips update t if

$$P(U^{(t)}) D_i^{(t)} < \xi^{(t)} . \tag{2}$$

Inserting (2) in place of StingyCD's condition for skipping updates is the only change from StingyCD to StingyCD+. In practice, we define $\xi^{(t)} = \mathrm{NNZ}\left(\mathbf{x}^{(t-1)}\right)$. Defining $\xi^{(t)}$ in this way leads to the following convergence guarantee:

Theorem 3.3 (StingyCD+ converges to a solution of (P)). *In StingyCD+, assume $\xi^{(t)} \leq \mathrm{NNZ}\left(\mathbf{x}^{(t-1)}\right)$ for all $t > 0$. Also, for each $i \in [m]$, assume the largest number of consecutive iterations during which* `get_next_coordinate()` *does not return i is bounded as $t \to \infty$. Then*

$$\lim_{t \to \infty} f(\mathbf{x}^{(t)}) = f(\mathbf{x}^\star) .$$

Proven in Appendix E, Theorem 3.3 ensures StingyCD+ convergences to a solution when $\xi^{(t)} \leq \mathrm{NNZ}\left(\mathbf{x}^{(t-1)}\right)$ for all t. As long as $\xi^{(t)}$ is smaller than this limit, at least one coordinate—specifically a coordinate for which $x_i^{(t-1)} \neq 0$—will satisfy (2) during a future iteration. By defining $\xi^{(t)}$ as this limit in practice, StingyCD+ achieves fast convergence times by skipping many updates.

4. Extending StingyCD to other objectives

For simplicity, we introduced StingyCD for nonnegative Lasso problems. In this section, we briefly describe how to apply StingyCD to some other objectives.

4.1. General (not nonnegative) Lasso problems

It is simple to extend StingyCD to general Lasso problems:

$$\underset{\mathbf{x} \in \mathbb{R}^m}{\text{minimize}} \; f_{\mathrm{L}}(\mathbf{x}) := \tfrac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 + \lambda \|\mathbf{x}\|_1 \tag{PL}$$

(PL) can be transformed into an instance of (P) by introducing two features (a positive and negative copy) for each column of $\mathbf{A}$. That is, we can solve (PL) with design matrix $\mathbf{A}$ by solving (P) with design matrix $[\mathbf{A}, -\mathbf{A}]$. Importantly, we perform this feature duplication implicitly in practice.

Two modifications to Algorithm 2 are needed to solve (PL). First, we adapt each update δ to the new objective:

$$\delta \leftarrow \underset{\delta}{\operatorname{argmin}} \; f_{\mathrm{L}}(\mathbf{x}^{(t-1)} + \delta \mathbf{e}_i) .$$

Second, we consider a positive and negative copy of $\mathbf{A}_i$ in the condition for skipping update t. Specifically, we define

$$\tau_i^+ \leftarrow \operatorname{sign}\left(\lambda - \langle \mathbf{A}_i, \mathbf{rr} \rangle\right) \frac{(\lambda - \langle \mathbf{A}_i, \mathbf{rr} \rangle)^2}{\|\mathbf{A}_i\|^2} , \quad \text{and}$$
$$\tau_i^- \leftarrow \operatorname{sign}\left(\lambda + \langle \mathbf{A}_i, \mathbf{rr} \rangle\right) \frac{(\lambda + \langle \mathbf{A}_i, \mathbf{rr} \rangle)^2}{\|\mathbf{A}_i\|^2} .$$

StingyCD skips update t if and only if $x_i^{(t-1)} = 0$ and $q^{(t-1)} \leq \min\{\tau_i^+, \tau_i^-\}$. Modifying StingyCD+ to solve (PL) is similar. $P(U^{(t)})$ becomes the sum of two probabilities corresponding to features $+\mathbf{A}_i$ and $-\mathbf{A}_i$. Specifically, $P(U^{(t)}) = P(U_+^{(t)}) + P(U_-^{(t)})$. We define $P(U_+^{(t)})$ and $P(U_-^{(t)})$ the same way as we define $P(U^{(t)})$ in §3.1 except we use τ_i^+ and τ_i^- in place of τ_i.

4.2. General ℓ_1-regularized smooth loss minimization

We can also use StingyCD to solve problems of the form

$$\underset{\mathbf{x} \in \mathbb{R}^m}{\text{minimize}} \; \textstyle\sum_{i=1}^n \phi_i(\langle \mathbf{a}_i, \mathbf{x} \rangle) + \lambda \|\mathbf{x}\|_1 , \tag{PL1}$$

where each ϕ_i is smooth. To solve this problem, we redefine $\mathbf{r}^{(t-1)}$ as a vector of derivatives:

$$\mathbf{r}^{(t-1)} = [-\phi_1'(\langle \mathbf{a}_1, \mathbf{x}^{(t-1)} \rangle), \ldots, -\phi_n'(\langle \mathbf{a}_n, \mathbf{x}^{(t-1)} \rangle)]^T .$$

When updating coordinate i, it remains true that $\delta = 0$ if $x_i^{(t-1)} = 0$ and $\mathbf{r}^{(t-1)} \notin \mathcal{A}_i$—the same geometry from Figure 1 applies. Unfortunately, updating $q^{(t-1)}$ no longer requires negligible computation. This is because in general, $\mathbf{r}^{(t)} \neq \mathbf{r}^{(t-1)} - \delta \mathbf{A}_i$. Thus, the update to $q^{(t)}$ in Algorithm 2 no longer applies. In other words, $q^{(t)} = \left\|\mathbf{r}^{(t)} - \mathbf{rr}\right\|^2$ cannot be computed from $q^{(t-1)}$ using negligible time.

Nevertheless, we can use StingyCD to efficiently solve (PL1) by incorporating StingyCD into a proximal Newton algorithm. At each outer-iteration, the loss $\sum_i \phi_i(\langle \mathbf{a}_i, \mathbf{x}\rangle)$ is approximated by a second-order Taylor expansion. This results in a subproblem of the form (PL), which we solve using StingyCD. CD-based proximal Newton methods are known to be very fast for solving (PL1), especially in the case of sparse logistic regression (Yuan et al., 2012).

4.3. Linear support vector machines

We can also apply StingyCD to train SVMs:

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\left\|\mathbf{M}\mathbf{x}\right\|^2 - \langle \mathbf{1}, \mathbf{x}\rangle \\ \text{s.t.} & \mathbf{x} \in [0, C]^n \end{array} . \qquad \text{(PSVM)}$$

This is the dual problem for training linear support vector machines. For this problem, we can apply concepts from §2.3 to guarantee $\delta = 0$ for many updates when $x_i^{(t-1)} = 0$ or $x_i^{(t-1)} = C$. To do so, the changes to StingyCD are straightforward. Due to limited space, we provide details in Appendix F.

5. Related work

StingyCD is related to safe screening tests as well as alternative strategies for prioritizing coordinate updates in CD.

5.1. Safe screening

Introduced in (El Ghaoui et al., 2012) for ℓ_1-regularized objectives, safe screening tests use sufficient conditions for which entries of the solution equal zero. Coordinates satisfying these conditions are discarded to simplify the problem. Follow-up works considered other problems, including sparse group Lasso (Wang & Ye, 2014), SVM training (Wang et al., 2014), and low-rank problems (Zhou & Zhao, 2015). Recent works proposed more flexible tests that avoid major issues of prior tests (Bonnefoy et al., 2014; 2015; Fercoq et al., 2015; Ndiaye et al., 2015; 2016), such as the fact that initial tests apply only prior to optimization.

The current state-of-the-art screening test was proposed in (Johnson & Guestrin, 2016). For the problem (P), this test relies on geometry similar to Figure 1. Specifically, the test defines a ball, $\mathcal{S}^{\text{Screen}}$, that is proven to contain the residual vector *of a solution to* (P). If $\mathcal{S}^{\text{Screen}} \cap \text{cl}(\mathcal{A}_i) = \emptyset$, it is guaranteed that the ith weight in (P)'s solution has value 0.

The radius of $\mathcal{S}^{\text{Screen}}$ is typically *much* larger than that of $\mathcal{S}^{(t)}$ in StingyCD, however. Unlike $\mathcal{S}^{(t)}$, $\mathcal{S}^{\text{Screen}}$ must contain the optimal residual vector. Unless a good approximate solution is available already, $\mathcal{S}^{\text{Screen}}$ is overly large, often resulting in few screened features (Johnson & Guestrin, 2016). By ensuring only that $\mathcal{S}^{(t)}$ contains the *current* residual vector and identifying zero-valued updates rather than zero-valued entries in a solution, StingyCD improves convergence times drastically more compared to screening.

5.2. Other approaches to prioritizing CD updates

Similar to StingyCD, recent work by (Fujiwara et al., 2016) also uses a reference vector concept for prioritizing updates in CD. Unlike StingyCD, this work focuses on identifying nonzero-valued coordinates, resulting in an active set algorithm. The reference vector is also a primal weight vector as opposed to a residual vector.

Similarly, shrinking heuristics (Fan et al., 2008; Yuan et al., 2012) and working set algorithms (Johnson & Guestrin, 2015; 2016) have been shown to be effective for prioritizing computation in CD algorithms. These algorithms solve a sequence of smaller subproblems which consider only prioritized subsets of coordinates. In these algorithms, StingyCD could be used to solve each subproblem to further prioritize computation. In §6, we show that using StingyCD+ instead of CD for solving subproblems in the working set algorithm from (Johnson & Guestrin, 2015) can lead to further convergence time improvements.

Finally, recent work has also considered adaptive sampling approaches for CD (Csiba et al., 2015). While also an interesting direction, this work does not apply to (P) due to a strong convexity requirement. Currently this approach also requires an additional pass over the data before each epoch as well as additional overhead for non-uniform sampling.

6. Empirical comparisons

This section demonstrates the impact of StingyCD and StingyCD+ in practice. We first compare these algorithms to CD and safe screening for Lasso problems. Later, we show that StingyCD+ also leads to speed-ups when combined with working set and proximal Newton algorithms.

6.1. Lasso problem comparisons

We implemented CD, CD with safe screening, StingyCD, and StingyCD+ to solve (PL). Coordinates are updated in round-robin fashion. We normalize columns of $\mathbf{A}$ and include an unregularized intercept term. We also remove features that have nonzero values in fewer than ten examples. For CD with safe screening, we apply the test from (Johnson & Guestrin, 2016), which is state-of-the-art to our knowledge. Following (Fercoq et al., 2015), screening is

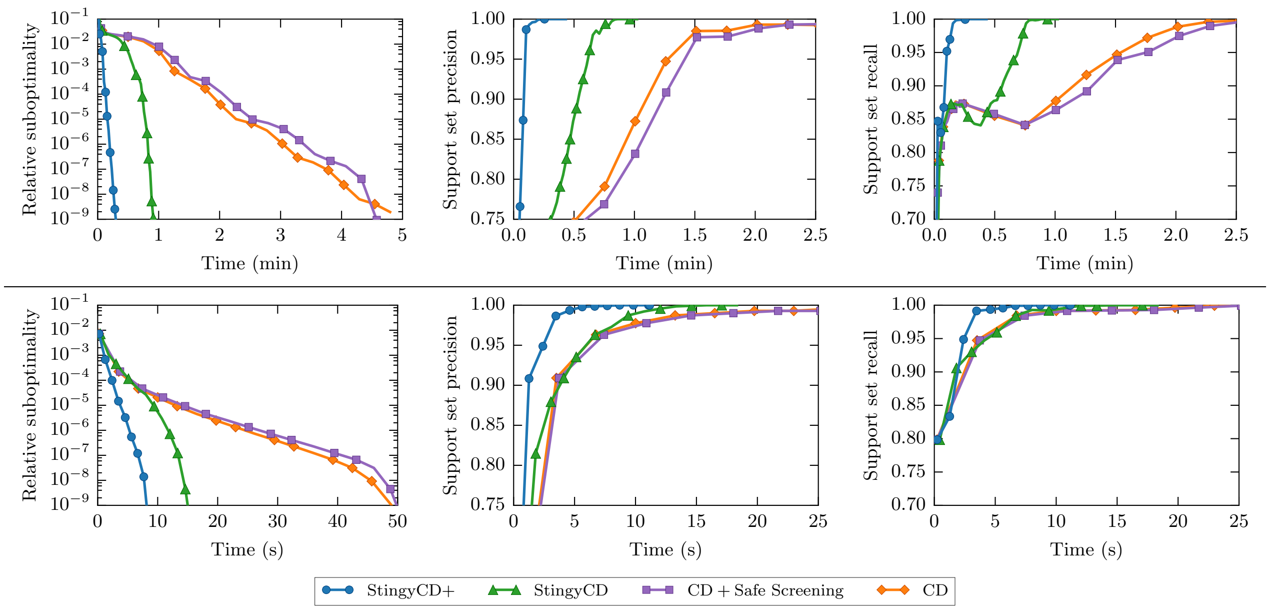


Figure 3. **Lasso results. (above)** finance. **(below)** allstate.

performed after every ten epochs. Performing screening requires a full pass over the data, which is non-negligible.

We compare the algorithms using a financial document dataset (finance)[1] and an insurance claim prediction task (allstate)[2]. finance contains 1.6×10^4 examples, 5.5×10^5 features, and 8.8×10^7 nonzero values. The result included in this section uses regularization $\lambda = 0.05\lambda_{\max}$, where $\lambda_{\max}$ is the smallest λ value that results in an all-zero solution. The solution contains 1746 nonzero entries.

The allstate data contains 2.5×10^5 examples, 1.5×10^4 features, and 1.2×10^8 nonzero values. For this problem, we set $\lambda = 0.05\lambda_{\max}$, resulting in 1404 selected features. We include results for additional λ values in Appendix G. StingyCD seems to have slightly greater impact when λ is larger, but the results generally do not change much with λ.

We evaluate the algorithms using three metrics. The first metric is relative suboptimality, defined as

$$\text{Relative suboptimality} = \frac{f(\mathbf{x}^{(t)}) - f(\mathbf{x}^\star)}{f(\mathbf{x}^\star)},$$

where $f(\mathbf{x}^{(t)})$ is the objective value at iteration t, and $\mathbf{x}^\star$ is the problem's solution. The other metrics are support set precision and recall. Let $\mathcal{F}^{(t)} = \{i : x_i^{(t)} \neq 0\}$, and let $\mathcal{F}^\star$ be the analogous set for $\mathbf{x}^\star$. We define

$$\text{Precision} = \frac{|\mathcal{F}^{(t)} \cap \mathcal{F}^\star|}{|\mathcal{F}^{(t)}|}, \quad \text{Recall} = \frac{|\mathcal{F}^{(t)} \cap \mathcal{F}^\star|}{|\mathcal{F}^\star|}.$$

[1]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#E2006-log1p

[2]https://www.kaggle.com/c/allstate-claims-severity

Precision and recall are arguably more important than suboptimality since (PL) is typically used for feature selection.

Results of these experiments are included in Figure 3. We see that StingyCD and StingyCD+ both greatly improve convergence times. For the reasons discussed in §5, safe screening provides little improvement compared to CD in these cases—even with the relative suboptimality is plotted until 10^{-9}. StingyCD provides a "safeness" similar to safe screening yet with drastically greater impact.

6.2. Combining StingyCD+ with working sets

This section demonstrates that StingyCD+ can be useful when combined with other algorithms. We consider the problem of sparse logistic regression, an instance of (PL1) in which each ϕ_i term is a logistic loss function. For each training example $(\mathbf{a}_i, b_i) \in \mathbb{R}^m \times [-1, 1]$, we have

$$\phi_i(\langle \mathbf{a}_i, \mathbf{x} \rangle) = \log(1 + \exp(-b_i \langle \mathbf{a}_i, \mathbf{x} \rangle)).$$

In this section, we use StingyCD+ as a subproblem solver for a proximal Newton algorithm and a working set algorithm. Specifically, we implement StingyCD+ within the "Blitz" working set algorithm proposed in (Johnson & Guestrin, 2015). At each iteration of Blitz, a subproblem is formed by selecting a set of priority features. The objective is then approximately minimized by updating weights only for features in this working set. Importantly, each subproblem in Blitz is solved approximately with a proximal Newton algorithm (overviewed in §4.2), and each proximal Newton subproblem is solved approximately with CD.

For these comparisons, we have replaced the aforemen-

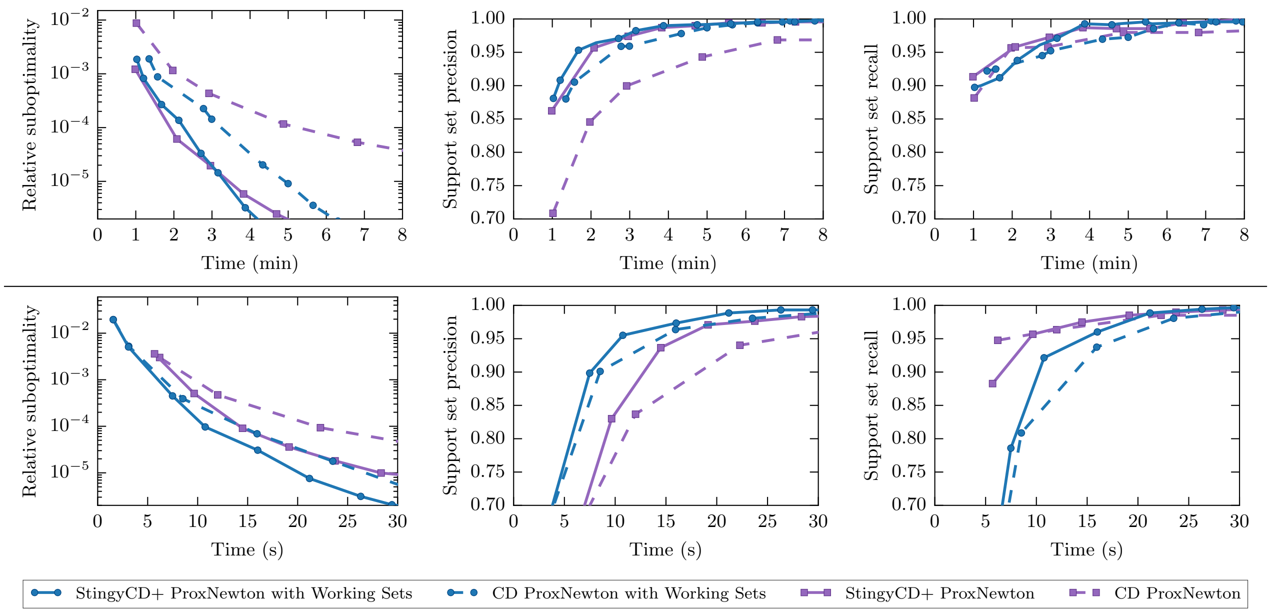


Figure 4. **Combining StingyCD+ with other algorithms for sparse logistic regression. (above)** kdda **(below)** lending_club

tioned CD implementation with a StingyCD+ implementation. We demonstrate the effects of this change when working sets are and are not used. In the case that working sets are omitted, we refer to the algorithm as "StingyCD+ ProxNewton" or "CD ProxNewton," depending on whether StingyCD+ is incorporated. We note that Blitz and the proximal Newton solver have not otherwise been modified, although it is likely possible to achieve improved convergence times by accounting for the use of StingyCD+. For example, Blitz could likely be improved by including more features in each working set, since StingyCD+ provides an additional layer of update prioritization.

The datasets used for this comparison are an educational performance dataset (kdda)[3] and a loan default prediction task (lending_club)[4]. After removing features with fewer than ten nonzeros, kdda's design matrix contains 8.4×10^6 examples, 2.2×10^6 features, and 2.8×10^8 nonzero values. We solve this problem with $\lambda = 0.005\lambda_{\max}$, which results in 692 nonzero weights at the problem's solution. The lending_club data contains 1.1×10^5 examples, 3.1×10^4 features, and 1.0×10^8 nonzero values. We solve this problem with $\lambda = 0.02\lambda_{\max}$, resulting in 878 selected features. We include plots for additional λ values in Appendix H.

Results of this experiment are shown in Figure 4. We see that replacing CD with StingyCD+ in both Blitz and ProxNewton can result in immediate efficiency improvements. We remark that the amount that StingyCD+ improved upon the working set approach depended significantly on λ, at least in the case of lending_club. For this dataset, when λ is relatively large (and thus the solution is very sparse), we observed little or no improvement due to StingyCD+. However, for smaller values of λ, StingyCD+ produced more significant gains. Moreover, StingyCD+ was the best performing algorithm in some cases (though in other cases, Blitz was much faster). This observation suggests that there likely exists a better approach to using working sets with StingyCD+—an ideal algorithm would obtain excellent performance across all relevant λ values.

7. Discussion

We proposed StingyCD, a coordinate descent algorithm that avoids large amounts of wasteful computation in applications such as sparse regression. StingyCD borrows geometric ideas from safe screening to guarantee many updates will result in no progress toward convergence. Compared to safe screening, StingyCD achieves considerably greater convergence time speed-ups. We also introduced StingyCD+, which applies a probabilistic assumption to StingyCD in order to further prioritize coordinate updates.

In general, we find the idea of "stingy updates" to be deserving of significantly more exploration. Currently this idea is limited to CD algorithms and, for the most part, objectives with quadratic losses. However, it seems likely that similar ideas would apply in many other contexts. For example, it could be useful to use stingy updates in distributed optimization algorithms in order to significantly reduce communication requirements.

[3]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#kdd2010(algebra)

[4]https://www.kaggle.com/wendykan/lending-club-loan-data

Acknowledgments

We thank our anonymous reviewers for their thoughtful feedback. This work is supported in part by PECASE N00014-13-1-0023, NSF IIS-1258741, and the TerraSwarm Research Center 00008169.

References

Bonnefoy, A., Emiya, V., Ralaivola, L., and Gribonval, R. A dynamic screening principle for the lasso. In *22nd European Signal Processing Conference*, 2014.

Bonnefoy, A., Emiya, V., Ralaivola, L., and Gribonval, R. Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso. *IEEE Transactions on Signal Processing*, 63(19):5121–5132, 2015.

Bradley, J. K., Kyrola, A., Bickson, D., and Guestrin, C. Parallel coordinate descent for L_1-regularized loss minimization. In *International Conference on Machine Learning*, 2011.

Csiba, D., Qu, Z., and Richtárik, P. Stochastic dual coordinate ascent with adaptive probabilities. In *International Conference on Machine Learning*, 2015.

El Ghaoui, L., Viallon, V., and Rabbani, T. Safe feature elimination for the Lasso and sparse supervised learning problems. *Pacific Journal of Optimization*, 8(4):667–698, 2012.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008.

Fercoq, O., Gramfort, A., and Salmon, J. Mind the duality gap: safer rules for the lasso. In *International Conference on Machine Learning*, 2015.

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

Fujiwara, Y., Ida, Y., Shiokawa, H., and Iwamura, S. Fast lasso algorithm via selective coordinate descent. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Johnson, T. B. and Guestrin, C. Blitz: a principled meta-algorithm for scaling sparse optimization. In *International Conference on Machine Learning*, 2015.

Johnson, T. B. and Guestrin, C. Unified methods for exploiting piecewise linear structure in convex optimization. In *Advances in Neural Information Processing Systems 29*, 2016.

Li, S. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistic*, 4(1):66–70, 2011.

Ndiaye, E., Fercoq, O., Gramfort, A., and Salmon, J. GAP safe screening rules for sparse multi-task and multi-class models. In *Advances in Neural Information Processing Systems 28*, 2015.

Ndiaye, E., Fercoq, O., Gramfort, A., and Salmon, J. Gap safe screening rules for sparse-group lasso. In *Advances in Neural Information Processing Systems 29*, 2016.

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

Richtárik, P. and Takáč, M. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1):433–484, 2016.

Shalev-Shwartz, S. and Tewari, A. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12(June):1865–1892, 2011.

Shi, H.-J. M., Tu, S., Xu, Y., and Yin, W. A primer on coordinate descent algorithms. Technical Report arXiv:1610.00040, 2016.

Tibshirani, R. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

Wainwright, M. J. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (Lasso). *IEEE Transactions on Information Theory*, 55(5):2183–2202, 2009.

Wang, J. and Ye, J. Two-layer feature reduction for sparse-group lasso via decomposition of convex sets. In *Advances in Neural Information Processing Systems 27*, 2014.

Wang, J., Wonka, P., and Ye, J. Scaling SVM and least absolute deviations via exact data reduction. In *International Conference on Machine Learning*, 2014.

Wright, S. J. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Yuan, G. X., Ho, C. H., and Lin, C. J. An improved GLMNET for L1-regularized logistic regression. *Journal of Machine Learning Research*, 13:1999–2030, 2012.

Zhou, Q. and Zhao, Q. Safe subspace screening for nuclear norm regularized least squares problems. In *International Conference on Machine Learning*, 2015.