A Second-Order Method for Convex ℓ_1-Regularized Optimization with Active Set Prediction

Nitish Shirish Keskar[a], Jorge Nocedal[a], Figen Öztoprak[b] and Andreas Wächter[a]

[a]*Northwestern University, USA;* [b] *Istabul Bilgi University*

(*In Review*)

We describe an active-set method for the minimization of an objective function ϕ that is the sum of a smooth convex function and an ℓ_1-regularization term. A distinctive feature of the method is the way in which active-set identification and second-order subspace minimization steps are integrated to combine the predictive power of the two approaches. At every iteration, the algorithm selects a candidate set of free and fixed variables, performs an (inexact) subspace phase, and then assesses the quality of the new active set. If it is not judged to be acceptable, then the set of free variables is restricted and a new active-set prediction is made. We establish global convergence for our approach, and compare the new method against the state-of-the-art code LIBLINEAR.

Keywords: ℓ_1-minimization; second-order; active-set prediction; active-set correction; subspace-optimization

AMS Subject Classification: 49M; 65K; 65H; 90C

1. Introduction

The problem of minimizing a composite objective that is the sum of a smooth convex function and a regularization term has received much attention; see e.g. [2, 25] and the references therein. This problem arises in statistics, signal processing, machine learning and in many other areas of applications. In this paper we focus on the case when the regularizer is defined in terms of an ℓ_1-norm, and propose an algorithm that employs a recursive active-set selection mechanism designed to make a good prediction of the active subspace at each iteration. This mechanism combines first- and second-order information, and is designed with the large-scale setting in mind.

The problem under consideration is given by

$$\min_{x\in\mathbb{R}^n} \ \phi(x) = f(x) + \mu\|x\|_1. \tag{1}$$

We assume that f is a smooth convex function and $\mu > 0$ is a fixed penalty parameter.

The algorithm proposed in this paper is different in nature from the most popular methods proposed for solving problem (1). These include first-order methods, such as ISTA, SpaRSA and FISTA [3, 9, 30], and proximal Newton methods that compute a step by minimizing a piecewise quadratic model of (1) using (for example) a coordinate descent iteration [4, 12, 14, 18, 21, 24, 26, 31]. The proposed algorithm also differs from methods that solve (1) by reformulating it as a bound constrained problem; for e.g.

[13, 22, 23, 27, 28].

Our algorithm belongs, instead, to the class of *orthant-based methods* [1] that minimize a smooth quadratic model of ϕ on a sequence of orthant faces of $\mathbb{R}^n$ until the optimal solution is found. But unlike the orthant-based methods described in [1, 7] and the bound-constrained approaches in [22], every iteration of our algorithm consists of a *corrective cycle* of orthant-face predictions and subspace minimization steps. This cycle is terminated when the orthant-face prediction is deemed to be reliable. After a trial iterate has been computed, a globalization mechanism accepts or modifies it (if necessary) to ensure overall convergence of the iteration.

The idea of employing a correction mechanism for refining the selection of the orthant face was introduced in [6] for the case when f is a convex quadratic function. That algorithm is, however, not competitive with state-of-the-art methods in terms of CPU time because each iteration requires the exact solution of a subspace problem, which is expensive, and because the orthant-face prediction mechanism is too liberal and can lead to long corrective cycles. These deficiencies are overcome in our algorithm, which introduces two key components. We employ an adaptive filtering mechanism that in conjunction with the corrective cycle yields an efficient prediction of zero variables at each iteration. We also design a strategy for solving, inexactly, the subproblems arising during each corrective step in a way that does not degrade the accuracy of the orthant-face prediction and yields important savings in computation. We show that the algorithm is globally convergent for strongly convex problems. Numerical tests on a variety of machine learning data sets suggest that our algorithm is competitive with a leading state-of-the-art code.

The main features of our algorithm can also be highlighted by contrasting them with recently proposed proximal Newton methods for solving problem (1). The algorithms proposed by [12, 21, 31] and others first chose an active set of variables using first-order sensitivity information. The active variables are set to zero, and the rest of the variables are updated by minimizing a *piecewise quadratic* approximation to (1) given by

$$q^k(x) = f(x^k) + (x - x^k)^T \nabla f(x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k) + \mu\|x\|_1. \qquad (2)$$

This minimization is performed inexactly using a randomized coordinate descent method. After a trial iterate is computed in this manner, a backtracking line search is performed to ensure decrease in $\phi(x)$.

The proximal Newton methods just outlined employ a very simple mechanism (the minimum norm subgradient) to determine the set of active variables at each iteration. On the other hand, they solve the sophisticated lasso subproblem (2) that inherits the non-smooth structure of the original problem and permits iterates to cross points of non-differentiability of $\phi(x)$. The latter property allows proximal Newton methods to refine the active set with respect to its initial choice. In contrast, our method invests a significant amount of computation in the identification of a working orthant face in $\mathbb{R}^n$, and then minimizes a simple smooth quadratic approximation of the problem on that orthant face,

$$\bar{q}^k(x) = f(x^k) + (x - x^k)^T \nabla f(x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k) + \mu\zeta^T x, \qquad (3)$$

where ζ is an indicator with values 0, 1 or -1, that identifies the orthant face. The working orthant is selected carefully, by verifying that the predictions made at each corrective step are realized. We do so because a simpler selection of the orthant face, such as that

performed in the OWL method [1], or the method described in [7] can generate poor steps in some circumstances.

Given that the two approaches (proximal Newton with coordinate descent solver and our proposed method) are different in nature, it is natural to ask if one of them will emerge as the preferred second-order technique for the solution of problem (1). To answer this question we compared a MATLAB implementation of our approach on binary classification problems with the well-known solver LIBLINEAR (written in C), based on CPU time. One of the main conclusions of this paper is that *both approaches* have their strengths. Orthant-based methods have the attractive property that the subspace minimization can be performed by a direct linear solver or by an iterative method such as the conjugate gradient method, which is efficient on a wide range of applications. On the other hand, the proximal Newton approach method is very effective on applications where the Hessian matrix is diagonally dominant (or nearly so). In this case, the coordinate descent iteration is particularly efficient in computing an approximate solution of problem (2). Both approaches share the need for effective criteria for deciding when an approximate solution of the subproblem is acceptable. Most implementations of the proximal Newton method employ adaptive techniques (heuristics or rules based in randomized analysis), while our implementation employs the classic termination criteria based on the relative error in the residue of the linear system [17].

This paper is organized in 5 sections. In Section 2 we outline the algorithm, paying particular attention to the orthant-face identification mechanism. Section 3 discusses the procedure by which we safeguard against poor steps and ensure global convergence of the algorithm. In Section 4, we present a comparison of our algorithm against the state-of-the-art code LIBLINEAR for the solution of binary classification problems; some final remarks are made in Section 5.

2. The Proposed Algorithm

The algorithm exploits the fact that the objective function ϕ is smooth in any *orthant face* of $\mathbb{R}^n$, which is defined as the intersection of an orthant in $\mathbb{R}^n$ and a subspace $\{x : x_i = 0,\, i \in I \subset \{1, \dots n\}\}$.

At every iteration, the algorithm identifies an orthant face in $\mathbb{R}^n$ using sensitivity information, performs a minimization on that orthant face to produce a trial point, refines the orthant-face selection (if necessary), and repeats the process until the choice of the orthant face is judged to be acceptable. Upon termination of this cycle, a backtracking line search is performed where the trial points are projected onto the active orthant.

To describe the algorithm in detail, we introduce some notation. Let $g(x)$ denote the minimum norm subgradient of the objective function (1) at a point x. Thus, we have

$$g_i(x) = \begin{cases} \nabla_i f(x) + \mu \text{ if } x_i > 0 \text{ or } \; (x_i = 0 \text{ and } \nabla_i f(x) + \mu < 0) \\ \nabla_i f(x) - \mu \text{ if } x_i < 0 \text{ or } \; (x_i = 0 \text{ and } \nabla_i f(x) - \mu > 0) \\ 0 \qquad\qquad \text{otherwise,} \end{cases} \tag{4}$$

for $i = 1, \dots n$, where

$$\nabla_i f(x) \stackrel{\text{def}}{=} \frac{\partial f(x)}{\partial x_i}.$$

At an iterate x^k, we define three sets:

$$\mathcal{A}^k = \{i |\, x_i^k = 0 \text{ and } |\nabla_i f(x^k)| \leq \mu\} \tag{5}$$

$$\mathcal{F}^k = \{i |\, x_i^k \neq 0\} \tag{6}$$

$$\mathcal{U}^k = \{i |\, x_i^k = 0 \text{ and } |\nabla_i f(x^k)| > \mu\}. \tag{7}$$

The variables in $\mathcal{A}^k$ are kept at zero (since the corresponding components of $g_i(x^k)$ are zero), while those in $\mathcal{F}^k$ are free to move. The remaining variables are in the set $\mathcal{U}^k$. The decision of which of these are allowed to move significantly impacts the efficiency of the algorithm. Using the *selection mechanism* described below, we first create a partition of $\mathcal{U}^k$,

$$\mathcal{U}^k = \mathcal{U}_A \cup \mathcal{U}_F, \tag{8}$$

where the variables in $\mathcal{U}_A$ are fixed at zero and the variables in $\mathcal{U}_F$ are allowed to move. We then update the active set as

$$\mathcal{A}^k \leftarrow \mathcal{A}^k \cup \mathcal{U}_A, \tag{9}$$

and compute a trial step d^k as the (approximate) solution of the smooth quadratic problem

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & \psi(d) = d^T g(x^k) + \frac{1}{2} d^T H^k d \\ \text{s.t.} \quad & d_i = 0, \ i \in \mathcal{A}^k, \end{aligned} \tag{10}$$

where $H^k = \nabla^2 f(x^k)$. The trial iterate is defined as

$$\hat{x}^k = x^k + d^k.$$

We then start the *corrective cycle* and check whether all variables in the set $\mathcal{U}_F$ moved as predicted; i.e., whether

$$\operatorname{sgn}([\hat{x}^k]_i) = \operatorname{sgn}(-[g(x^k)]_i) \quad \text{for all } i \in \mathcal{U}_F. \tag{11}$$

Any variable $j \in \mathcal{U}_F$ for which this equality does not hold, is removed from the set $\mathcal{U}_F$ and added to $\mathcal{U}_A$. The set $\mathcal{A}^k$ is then updated according to (9) and a new trial step is recomputed by solving (10). We repeat this corrective cycle until all predictions are correct and the trial point $\hat{x}^k$ satisfies (11).

The algorithm then performs a projected backtracking line search along d^k to ensure that the resulting point yields a decrease in the piecewise quadratic model $q^k(x)$ defined in (2). (We do not perform the line search on the objective function (1) as that is more expensive, and the globalization mechanism described in Section 3 only requires a decrease in $q^k(x)$.)

At iteration k, we identify the current orthant face based on sensitivity information (4) and define the vector ζ^k by

$$\zeta_i^k = \begin{cases} \operatorname{sgn}([x^k]_i) & \text{if } [x^k]_i \neq 0 \\ \operatorname{sgn}(-[g(x^k)]_i) & \text{if } [x^k]_i = 0. \end{cases} \tag{12}$$

Let $\mathcal{P}^k(x)$ be the projection operator that projects $x \in \mathbb{R}^n$ onto the orthant defined by ζ^k; i.e.,

$$\mathcal{P}_i^k(x) = \begin{cases} x_i & \text{if } \operatorname{sgn}(x_i) = \operatorname{sgn}(\zeta_i^k) \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

We then search for the largest step size $\alpha \in \{2^0, 2^{-1}, 2^{-2}, \cdots\}$ such that

$$q(x^k) \geq q(\mathcal{P}^k(x^k + \alpha \cdot d^k)),$$

where q is the non-smooth quadratic approximation given by (2). Such a step size exists because d^k is a descent direction for the smooth quadratic function $\bar{q}^k$ and because the trial point lies within the orthant defined by ζ^k for sufficiently small steps (see Theorem A.4 in the appendix).

Before giving a detailed description of the algorithm, we describe the *selection mechanism* that, at the beginning of each corrective cycle, defines the splitting (8) of the set $\mathcal{U}^k$ into variables $\mathcal{U}_A$, that are kept at zero, and variables $\mathcal{U}_F$, that are allowed to move.

At the start of the algorithm, we select a scalar $\eta \in (0, 1)$ and set $|\mathcal{U}_F| = \tau^0 \stackrel{\text{def}}{=} \lfloor \eta \times n \rfloor$; i.e., the cardinality of the set $\mathcal{U}_F$ is a fraction of the dimension of the problem. On subsequent iterations, we update the parameter τ^k based on its previous value τ^{k-1} and the number of iterations in the previous corrective cycle. If there were no corrections in the previous corrective cycle, we set $\tau^{k+1} = 2\tau^k$ to allow more variables to change at the next outer iteration; otherwise we keep the value of τ^k unchanged. Since the number of variables in $\mathcal{U}_F$ cannot be larger than $|\mathcal{U}^k|$, the actual size of $\mathcal{U}_F$ is given by

$$|\mathcal{U}_F| = \hat{\tau}^k \stackrel{\text{def}}{=} \min(|\mathcal{U}^k|, \tau^k).$$

We use a greedy strategy to populate the sets $\mathcal{U}_A$ and $\mathcal{U}_F$: we collect in $\mathcal{U}_F$ the $\hat{\tau}^k$ variables in $\mathcal{U}^k$ with the largest components of the subgradient $|g(x^k)|$. Thus, for any $i \in \mathcal{U}_F$ and $j \in \mathcal{U}_A$, we have $|g_i(x^k)| \geq |g_j(x^k)|$.

A formal description of the overall method is given in Algorithm 1.

Algorithm 1 Preliminary Adaptive Orthant-Based Method

Given $x^0 \in \mathbb{R}^n$, $L > 0$, $\mu > 0$, $\eta \in (0, 1)$.
Let $\tau^0 = \lfloor \eta \times n \rfloor$.
while $k = 0, 1, 2, \cdots$ and stopping criterion not met **do**
Active-Set Identification:

$$\mathcal{A}^k = \{i | (x_i)^k = 0 \text{ and } |\nabla_i f(x^k)| \leq \mu\}$$
$$\mathcal{F}^k = \{i | (x_i)^k \neq 0\}$$
$$\mathcal{U}^k = \{i | (x_i)^k = 0 \text{ and } |\nabla_i f(x^k)| > \mu\}$$

Selection Mechanism:
Compute $g(x^k)$ by (4) and ζ^k by (12).
Set $\hat{\tau}^k \leftarrow \min(|\mathcal{U}^k|, \tau^k)$.
Choose $\mathcal{U}_F, \mathcal{U}_A \subseteq \mathcal{U}^k$ such that $\mathcal{U}_F \cap \mathcal{U}_A = \emptyset$, $|\mathcal{U}_F| = \hat{\tau}^k$ and for any $i \in \mathcal{U}_F$ and $j \in \mathcal{U}_A$, $|g_i(x^k)| \geq |g_j(x^k)|$.
Set $\mathcal{A}^k \leftarrow \mathcal{A}^k \cup \mathcal{U}_A$.
Compute or update second-order approximation H^k.
Corrective Cycle:
Set $V^k \leftarrow \mathcal{U}_F$ and $j \leftarrow 0$.
while $V^k \neq \emptyset$ **do**

$$d^k = \underset{d_i = 0, i \in \mathcal{A}^k}{\operatorname{argmin}} \; d^T g(x^k) + \tfrac{1}{2} d^T H^k d$$

Set $\hat{x}^k \leftarrow x^k + d^k$.
Set $V^k = \{i \in \mathcal{U}_F \setminus \mathcal{A}^k | \operatorname{sgn}(\zeta_i^k) \neq \operatorname{sgn}(\hat{x}_i^k)\}$.
Set $\mathcal{A}^k \leftarrow \mathcal{A}^k \cup V^k$ and $j \leftarrow j + 1$.
end while
if $j = 1$ **then**
Set $\tau^{k+1} = 2 \cdot \tau^k$.
end if
Projected Line Search:
Set $\alpha \leftarrow 1$.
while $q(x^k) > q(\mathcal{P}^k(x^k + \alpha \cdot d^k))$ **do**
Set $\alpha \leftarrow \alpha/2$.
end while
Set $x^{k+1} = \mathcal{P}^k(x^k + \alpha \cdot d^k)$.
end while

In this paper we assume that the quadratic model (10) employs exact Hessian information, i.e. $H^k = \nabla^2 f(x^k)$, and that we perform an approximate minimization of this problem using the conjugate gradient method in the appropriate subspace of dimension $(n - |\mathcal{A}^k|)$; see e.g., [17]. The matrix H^k can also be defined by quasi-Newton updates, specifically using the compact representations of limited-memory BFGS matrices [5]. Although we do not explore a quasi-Newton variant in this paper, we expect it to be effective in many applications.

Our *selection mechanism* for defining the splitting (8) is motivated by the following

considerations. If all variables in $\mathcal{U}^k$ were allowed to move, the algorithm would have similar properties to the OWL method [1], whose performance is not uniformly successful (see Section 4). Indeed, we observed a more reliable performance when the size of $\mathcal{U}_F$ is limited. This also has computational benefits because the subproblem (10) is less expensive to solve when the number of free variables is smaller (i.e., when the set $\mathcal{A}^k$ is larger). On the other hand, in the extreme case $|\mathcal{U}_F| = 1$, the algorithm resembles a classical active-set method, which is not well-suited for large-scale problems.

These trade-offs are addressed by the dynamic strategy employed in steps 5 and 17 of Algorithm 1. Initially, we choose $\mathcal{U}_F$ to be a small subset of $\mathcal{U}^k$ (by selecting η to be small). The algorithm increases the size of $\mathcal{U}_F$ in subsequent iterations if there is evidence that the current choice is too restrictive. As as indicator we observe the number of iterations in the previous corrective cycle. A small number of corrections (in our implementation this number is 1) suggests that the choice of $\mathcal{U}_F$ may be too conservative and the size of $\mathcal{U}_F$ is doubled at the next outer iteration. We have found that this selection mechanism leads to more gradual and controlled changes in the active set compared to other orthant-based methods like OWL and the method proposed in Section 5 of [7].

The projected backtracking line search in Algorithm 1 differs from that used by other orthant-based methods in that it is based on the quadratic model and not the objective function. As in other orthant-based methods, the projection promotes sparsity in the iterates and provides some control for steps that leave the current orthant, outside of which the smooth approximation in (10) is not valid. But in contrast to other orthant-based methods, such as OWL, the line search is not the main globalization mechanism in our algorithm, as described next.

3. Globalization Strategy

While Algorithm 1 generally works well in practice, it may fail (cycle) when the changes in the active set are not sufficiently controlled. By adding a globalization mechanism we ensure that all iterates generated by the algorithm provide sufficient reduction in the objective function and converge to the solution. Our mechanism employs the iterative soft-thresholding algorithm (ISTA) [8, 9] to generate a reference point. Because the ISTA method enjoys a global linear rate of convergence on strongly convex problems, it provides a benchmark for the progress of our algorithm.

We modify Algorithm 1 as follows. The iterate computed in line 23 is now regarded as a trial iterate and denoted by $\hat{x}^k$. To decide if this point is acceptable we check whether it produces a lower function value than the ISTA step computed from the starting point of the iteration, x^k. If so, we accept the trial point; otherwise, we search along the segment joining $\hat{x}^k$ and the ISTA point x^k_{ISTA} to find an acceptable point. Given a Lipschitz constant L for the gradient of f, the cost of computing the ISTA step is negligible since gradient information is already available at x^k. However, the evaluation of $\phi(x^k_{\text{ISTA}})$ incurs an additional cost. To get around this expense, we use the value of an upper quadratic approximation of ϕ at x^k_{ISTA} as a surrogate to $\phi(x^k_{\text{ISTA}})$. More specifically, assuming that L is a Lipschitz constant of ∇f, we define the value of the surrogate function as

$$\Gamma^k = f(x^k) + \nabla f(x^k)^T(x^k_{\text{ISTA}} - x^k) + \tfrac{L}{2}\|x^k_{\text{ISTA}} - x^k\|_2^2 + \mu\|x^k_{\text{ISTA}}\|_1. \tag{14}$$

The computation of Γ^k requires only one inner product. The complete version of the algorithm, including the globalization mechanism, is given in Algorithm 2.

Algorithm 2 Orthant-Based Adaptive Method (OBA)

Given $x^0 \in \Re^n$, $L > 0$, $\mu > 0$, $\eta \in (0,1)$ and $\epsilon > 0$.
Let $\tau^0 = \lfloor \eta \times n \rfloor$
while $k = 0, 1, 2, \cdots$ and stopping criterion not met **do**
  Carry out steps 1 – 22 of Algorithm 1.
  Set $\hat{x}^k = \mathcal{P}^k(x^k + \alpha \cdot d^k)$.
  Globalization:
  Compute ISTA step at x^k as

$$x^k_{\text{ISTA}} = \mathcal{S}_{\mu/L}(x^k - \tfrac{1}{L}\nabla f(x^k))$$

  where $\mathcal{S}_\alpha(x)$ is a component-wise operator defined as $\mathcal{S}_\alpha(x)_i = \max\{|x_i| - \alpha, 0\} \cdot \text{sgn}(x_i)$.
  Set $\bar{d}^k \leftarrow \hat{x}^k - x^k_{\text{ISTA}}$ and $\bar{\alpha} \leftarrow 1$.
  Calculate Γ^k using (14).
  while $\phi(x^k_{\text{ISTA}} + \bar{\alpha} \cdot \bar{d}^k) > \Gamma^k$ **do**
    Set $\bar{\alpha} \leftarrow \bar{\alpha}/2$.
    if $\bar{\alpha} < \epsilon$ **then**
      Set $\bar{\alpha} \leftarrow 0$.
    end if
  end while
  Set $x^{k+1} = x^k_{\text{ISTA}} + \bar{\alpha} \cdot \bar{d}^k$.
end while

The following convergence result is proven in the appendix.

THEOREM *Assume that f is continuously differentiable and strongly convex and that ∇f is Lipschitz continuous. Then, the iterates $\{x^k\}$ generated by Algorithm 2 converge to the optimal solution $x^\star$ of problem* (1) *at a linear rate.*

4. Numerical Experiments

In this section, we demonstrate the viability of our approach. While our method applies to any convex function with an additive ℓ_1-regularizer, we focus on the specific problem of binary classification using logistic regression. This problem is well studied with theoretical guarantees and many data sets available of varying sizes, structures and fields of study. Further, the results reported on this problem are representative of the performance of OBA on other functions (including multi-class logistic regression, probit regression and LASSO) where similar trends are observed. We direct the reader to [10] and the references therein for details regarding the function f and the statistical justifications of this choice.

The data sets chosen for comparison are listed in Table 1. Synthetic is a randomly generated, balanced, non-diagonally dominant problem; the process for generating this problem is described in the appendix. Alpha is a data set from the Pascal Large Scale Learning Challenge [19]. Both these data sets have been feature-wise normalized to $[-1, 1]$. Details for the other data sets along with their preprocessing steps can be found in `http://www.csie.ntu.edu.tw/~cjlin/liblinear` and the references therein.

A variety of methods has been proposed for solving problem (1), and high-performance implementations of some of these methods are available. One of the most popular codes is newGLMNET [31], which is a C-implementation of a proximal Newton method and is

Table 1. Data sets

Data set	number of data points	number of features
Gisette	6000	5000
RCV1	20242	47236
Alpha	500000	500
KDDA	8407752	20216830
KDDB	19264097	29890095
Epsilon	400000	2000
News20	19996	1355191
Synthetic	5000	5000

a part of the LIBLINEAR package. Every iteration of this method identifies the active set as a subset of $\mathcal{A}^k$ as defined in (5), and then solves problem (2) inexactly using a randomized coordinate descent algorithm. The termination criterion for this inner loop is based on the ℓ_1-norm of the minimum norm subgradient and adjusted by a heuristic as the iteration progresses.

We implemented Algorithm 2 in MATLAB, where we chose $\eta = 0.01$ and $\epsilon = 10^{-4}$. Subproblem (10) is solved inexactly via the conjugate gradient algorithm. The termination criterion is based on the relative tolerance of the linear system: The inner loop stops as soon as the conjugate gradient iterate p satisfies

$$\frac{\|H^k p + g^k\|_\infty}{\|g^k\|_\infty} \leq 0.1.$$

We also compare with the OWL method [1], as implemented by Schmidt [22]. The OBA algorithm with the selective-corrective mechanism removed is somewhat related to OWL. The primary differences between the two include the procedure of handling the active set constraints in the subproblem and the alignment step included in OWL. Specifically, OWL minimizes the quadratic model in (10) over $\mathbb{R}^n$, aligns the search direction and then carries out a projected line search onto the active set.

For all test problems, the regularization parameter μ was chosen through a 5–fold cross validation. LIBLINEAR and OBA use the exact Hessian in defining the quadratic model (10) and (2) while OWL uses a limited-memory BFGS approximation. Further, in order to solve singular problems, LIBLINEAR adds a small multiple (specifically, 10^{-12}) of the identity to the Hessian and our algorithm uses the value of 10^{-8}. LIBLINEAR employs a secondary mechanism to guard against singularity: it projects the result of the one dimensional optimization in the coordinate descent step onto the set $[-10, 10]$.

4.1 *Test Results*

The comparison of the method proposed in this paper, Algorithm 2 (OBA), against LIBLINEAR and OWL is presented in Figures 1 and 2. We plot the relative function error defined as

$$\frac{\phi(x^k) - \phi(x^\star)}{1 + \phi(x^\star)} \tag{15}$$

against CPU time. The value of $\phi(x^\star)$ was obtained by running our algorithm to a tight tolerance of 10^{-10} or until a time limit of 5000 CPU seconds was exceeded. The tolerance used corresponds to the one defined in [4]. The initial iterate for all methods was the zero vector.

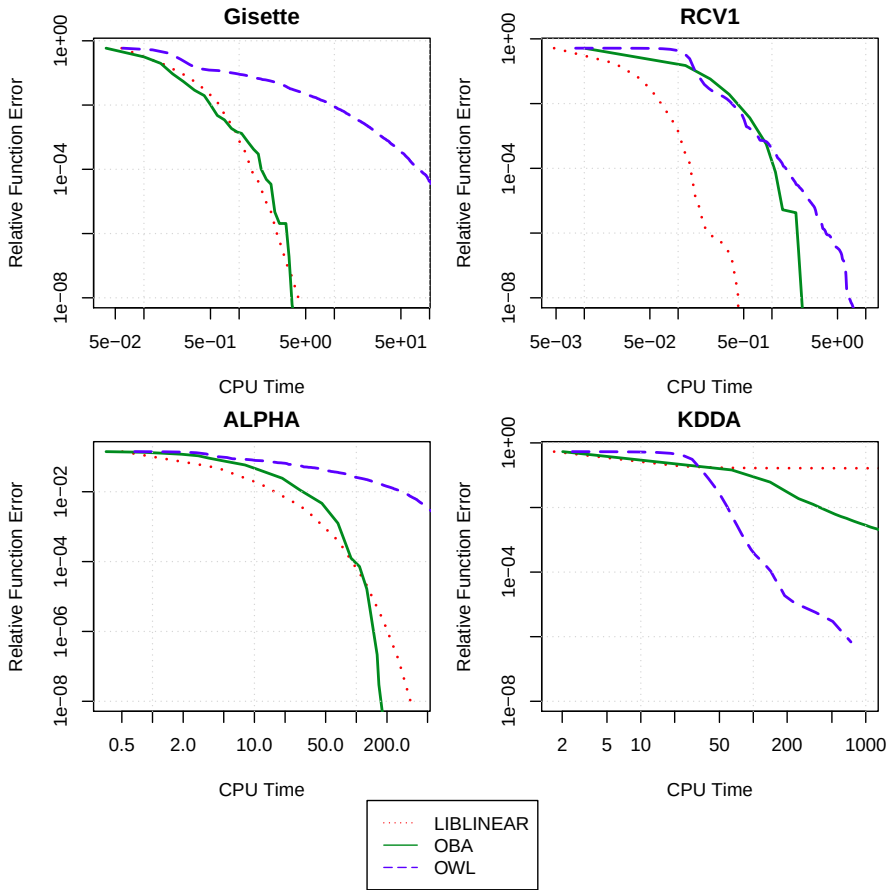

Figure 1. Relative error (15) in the objective function (vertical axis) versus CPU time – Part 1

We can see that for RCV1 and News20, the performance of OBA is inferior to LIBLINEAR; however, for KDDA, KDDB, Epsilon and Synthetic, the performance is superior. For Gisette and Alpha, the performance is roughly comparable irrespective of the value of the relative function error. OWL has gained a reputation as an algorithm which performs well but unreliably so. The experiments support this opinion. For problems like KDDA or KDDB, the performance of OWL is superior to both LIBLINEAR and OBA; however, for other problems like Synthetic, Alpha, Epsilon and Gisette, OWL fails to be competitive due to poor steps and rapid changes in working orthant faces.

We emphasize that the improved performance of the proposed method is driven by the selective-corrective mechanism as opposed to the ISTA backup. The backup was *never* required in the reported experiments for our method which, in contrast to other orthant-based methods, enjoys global convergence properties.

4.2 *Sparsity*

It is natural to ask whether an orthant-based method such as OBA is as effective at generating sparsity in the solution as a proximal Newton method, such as LIBLINEAR. In proximal Newton methods the non-smoothness of the original problem is retained in the subproblem (2), and sparsity arises because the solution of the subproblem typ-

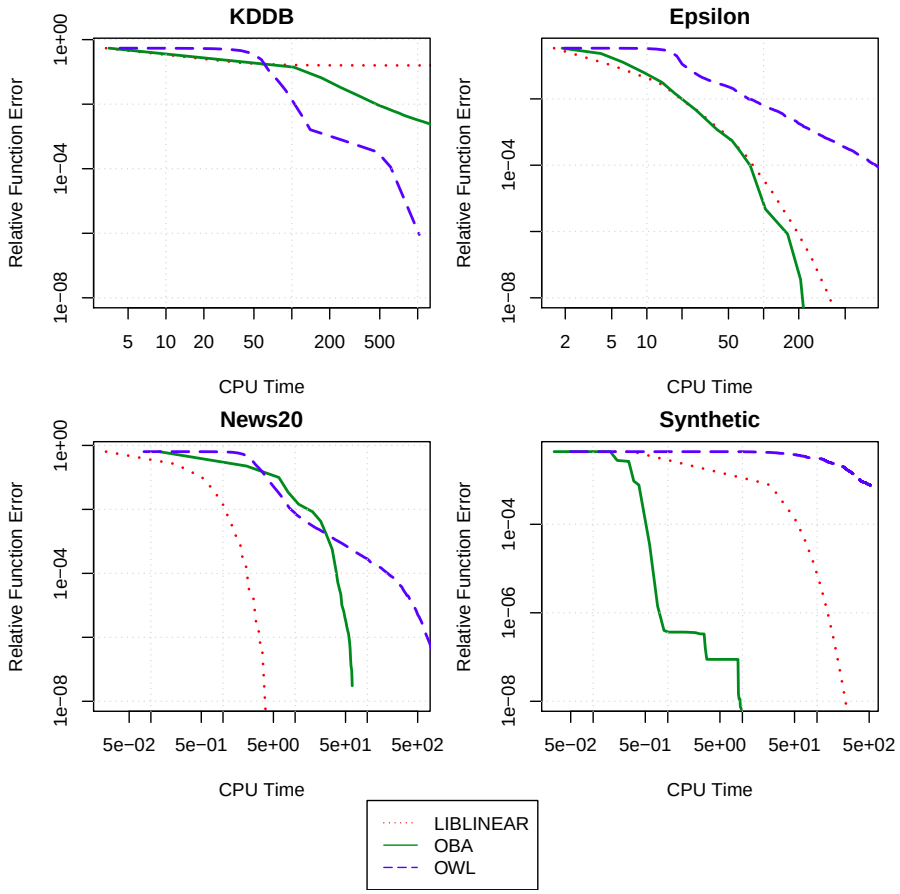

Figure 2. Relative error (15) in the objective function (vertical axis) versus CPU time – Part 2

ically lies at points of non-differentiability. In contrast, orthant-based methods solve a series of smooth problems that have no tendency of inducing sparsity in the solution by themselves, but achieve it through the projection of the trial point onto the working orthant.

Both methods, LIBLINEAR and OBA, also promote sparsity through the definition of the active set at the beginning of each (outer) iteration, but the construction of the active set differs in the two methods. LIBLINEAR fixes only a subset of the variables in the set $\mathcal{A}^k$ to zero; thus allowing some variables in $\mathcal{A}^k$ and all variables in the set $\mathcal{U}^k$ to move. On the other hand, OBA fixes all of the variables in $\mathcal{A}^k$ to zero and additionally fixes more variables in $\mathcal{U}^k$ through the selection mechanism and the corrective cycle. Therefore, the approach in LIBLINEAR can be considered more liberal in that it releases more zero variables, while the approach in OBA can be regarded as more restrictive. Nevertheless, OBA becomes increasingly more liberal as the iteration progresses because the selection mechanism allows the size of the set $\mathcal{U}_F$ to double under certain circumstances (see step 17 of Algorithm 1).

In the light of these algorithmic differences, it is difficult to predict the relative ability of the two methods at generating sparse solutions. To explore this, we performed the following experiments using our data sets and recorded the sparsity in the solution. LIBLINEAR was used to solve the problems with the tolerance of their stopping criterion

Table 2. Percentage of Zeros in Solution

Data set	LIBLINEAR	OBA
Gisette	88.92	**90.52**
RCV1	97.62	**97.65**
Alpha	**5.60**	5.20
KDDA	98.43	**98.71**
KDDB	97.11	**97.87**
Epsilon	44.60	**69.20**
News20	**99.60**	99.37
Synthetic	56.86	**58.82**

set to 10^{-3} (which yielded better misclassification rates than the default value of 10^{-2}), and OBA was then used to solve the problems to a similar accuracy in the objective function. The results are presented in Table 2 and show that the two methods achieve similar values of sparsity, with OBA being somewhat more effective.

4.3 *Conjugate Gradients versus Coordinate Descent*

Let us now focus on the methods used for solving the subproblems that incorporate second-order information about the objective function. It is natural to employ the conjugate gradient (CG) method in OBA, given that the subproblem (3) is smooth and that the CG method is an optimal Krylov process that can exploit problem structure effectively. An alternative to the CG method is the randomized coordinate descent algorithm, which has gained much popularity in recent years [10, 16, 20]

A drawback of coordinate descent for smooth unconstrained optimization is that it can be slow when the Hessian is not diagonally dominant. We experimented with a coordinate descent solver for the subproblem in OBA and found that its overall performance is inferior to that of the CG method.

The situation is quite different in a proximal Newton method where the subproblem is non-smooth. In that case, it is easy to compute the exact minimizer of (2) along each coordinate direction, thereby dealing explicitly with the non-differentiability of the original problem. Since this one dimensional minimization may return zero as the exact solution, the proximal coordinate descent method provides an active-set identification mechanism for the overall algorithm. Thus, although sensitivity to the lack of diagonal dominance may still be present, it is of a lesser concern due the benefits of its active set identification properties. Furthermore, applications in text classification and other areas often lead to problems with Hessians that are diagonally dominant [11].

This discussion motivates us to look more closely at the issue of diagonal dominance and its effect on the two methods. In order to quantify the level of diagonal dominance, we use the metric employed, for example, in [29]. Given any symmetric matrix A, we define the level of diagonal dominance of A as

$$\mathcal{D}(A) = \frac{\max_i \|A_i\|_2}{\max_i |A_{ii}|}, \tag{16}$$

where A_i denotes the i^{th} column of A and A_{ii} denotes the i^{th} diagonal element of A. The smaller the value of $\mathcal{D}$, the closer is A to being diagonally dominant. In Table 3 we report the values of $\mathcal{D}\left(\nabla^2 f(x^0)\right)$ for all data sets in Table 1.

Let us begin by considering problem Synthetic, which was specifically constructed to have a high value of $\mathcal{D}$ (see Appendix B for details). We observe from Figure 2

Table 3. The value of $\mathcal{D}(\nabla^2 f(x^0))$ as defined in (16)

Data set	$\mathcal{D}(\nabla^2 f(x^0))$
Gisette	57.99
RCV1	1.88
Alpha	9.93
KDDA	1.80
KDDB	1.50
Epsilon	5.55
News20	3.29
Synthetic	69.42

that LIBLINEAR performs poorly compared to OBA. This may be an indication that proximal Newton methods are sensitive to a lack of diagonal dominance. In fact, by altering this problem so that $\mathcal{D}$ increases, the performance of LIBLINEAR deteriorates. The text classification tasks (RCV1 and News20), which are empirically observed to be diagonally dominant [11], have low values of $\mathcal{D}$ and indeed, LIBLINEAR converges quickly[1]. However, LIBLINEAR also performs well on problem Gisette for which $\mathcal{D}$ is large and poorly on KDDB for which $\mathcal{D}$ is low. An examination of the rest of the results prevents us from establishing a clear correlation between the value of $\mathcal{D}$ and the relative performance of the two methods. We conclude that in ℓ_1-regularized problems the adverse effects of diagonal dominance appear to be less pronounced than for smooth optimization. Other factors such as the frequency of orthant changes and the inexactness in the subproblem solution may also play a crucial role in explaining the performance differences. The identification of problem characteristics that determine which method performs better for a given instances requires further investigation.

5. Final Remarks

In this paper, we presented a second-order algorithm for solving convex ℓ_1-regularized problems. At each iteration, the algorithm tries to predict the orthant face containing the solution, solves a smooth quadratic subproblem on this orthant face, and then invokes a corrective cycle that greatly improves the efficiency and robustness of the algorithm. We globalized the method by using the ISTA step as a reference for the desired progress. This enabled us to prove a linear convergence rate of the iterates for strongly convex problems. The ISTA backup is rarely used in practice (and never in the reported experiments) and thus, our theoretical result applies to a very robust method that invokes the safeguarding very rarely. This globalization procedure is analogous to a Newton trust-region method where the underlying method is known to be very effective but convergence can only be proved by overcoming pathological situations with a first-order Cauchy step. Numerical experiments for logistic regression data sets show that our algorithm is competitive in terms of CPU time with the LIBLINEAR C-code, even though our implementation is in MATLAB. The algorithm is also effective in generating sparse solutions quickly. Overall, our experiments indicate that orthant-based methods are a viable alternative to proximal

[1]Interestingly, the LIBLINEAR website and manual convey that LIBLINEAR is known to perform well on document classification tasks but not necessarily on others.

Newton methods.

References

[1] G. Andrew and J. Gao, *Scalable training of L_1-regularized log-linear models*, in *Proceedings of the 24th international conference on Machine Learning*, 2007, pp. 33–40.

[2] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, *Optimization with sparsity-inducing penalties*, Foundations and Trends in Machine Learning 4 (2012), pp. 1–106.

[3] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences 2 (2009), pp. 183–202.

[4] R. Byrd, J. Nocedal, and F. Oztoprak, *An inexact successive quadratic approximation method for convex l1 regularized optimization*, arXiv preprint arXiv:1309.3529 (2013).

[5] R.H. Byrd, J. Nocedal, and R. Schnabel, *Representations of quasi-Newton matrices and their use in limited memory methods*, Mathematical Programming 63 (1994), pp. 129–156.

[6] R.H. Byrd, G.M. Chin, J. Nocedal, and F. Oztoprak, *A family of second-order methods for convex L1 regularized optimization*, Tech. Rep., Optimization Center Report 2012/2, Northwestern University, 2012.

[7] R.H. Byrd, G.M. Chin, J. Nocedal, and Y. Wu, *Sample size selection in optimization methods for machine learning*, Mathematical Programming 134 (2012), pp. 127–155.

[8] I. Daubechies, M. Defrise, and C. De Mol, *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint*, Communications on Pure and Applied Mathematics 57 (2004), pp. 1413–1457.

[9] D. Donoho, *De-noising by soft-thresholding*, Information Theory, IEEE Transactions on 41 (1995), pp. 613–627.

[10] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, Journal of Statistical Software 33 (2010), p. 1, Available at `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2929880/`.

[11] D. Greene and P. Cunningham, *Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering*, in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, Pittsburgh, Pennsylvania, Available at `http://doi.acm.org/10.1145/1143844.1143892`, ACM, New York, NY, USA, 2006, pp. 377–384.

[12] C.J. Hsieh, M.A. Sustik, P. Ravikumar, and I.S. Dhillon, *Sparse inverse covariance matrix estimation using quadratic approximation*, Advances in Neural Information Processing Systems (NIPS) 24 (2011), pp. 2330–2338.

[13] K. Koh, S.J. Kim, and S.P. Boyd, *An interior-point method for large-scale l1-regularized logistic regression.*, Journal of Machine learning research 8 (2007), pp. 1519–1555.

[14] J. Lee, Y. Sun, and M. Saunders, *Proximal Newton-type methods for convex optimization*, in *Advances in Neural Information Processing Systems*, 2012, pp. 836–844.

[15] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Boston: Kluwer Academic Publishers, 2004.

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

[17] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed., Springer New York, 1999.

[18] P. Olsen, F. Oztoprak, J. Nocedal, and S. Rennie, *Newton-like methods for sparse inverse covariance estimation*, in *Advances in Neural Information Processing Systems 25*, P. Bartlett, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds., 2012, pp. 764–772, Available at `http://books.nips.cc/papers/files/nips25/NIPS2012_0344.pdf`.

[19] *Pascal large scale learning challenge*, `http://largescale.ml.tu-berlin.de/` (2008), accessed: 2015-01-01.

[20] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Mathematical Programming 144 (2014), pp. 1–38.

[21] K. Scheinberg and X. Tang, *Practical inexact proximal quasi-Newton method with global complexity analysis.*, arXiv preprint arXiv:1311.6547 (2014).

[22] M. Schmidt, *Graphical model structure learning with l1-regularization*, Ph.D. thesis, University of British Columbia, 2010.

[23] M. Schmidt, G. Fung, and R. Rosales, *Fast optimization methods for l1 regularization: A comparative study and two new approaches*, in *Machine Learning: ECML 2007*, Springer, 2007, pp. 286–297.

[24] M. Schmidt, D. Kim, and S. Suvrit, *Projected Newton-type methods in machine learning*, Optimization for Machine Learning (2011).
[25] S. Sra, S. Nowozin, and S. Wright, *Optimization for Machine Learning*, Mit Press, 2011.
[26] P. Tseng and S. Yun, *A coordinate gradient descent method for nonsmooth separable minimization*, Mathematical Programming 117 (2009), pp. 387–423.
[27] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang, *A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation*, SIAM J. Sci. Comput. 32 (2010), pp. 1832–1857, Available at `http://dx.doi.org/10.1137/090747695`.
[28] Z. Wen, W. Yin, H. Zhang, and D. Goldfarb, *On the convergence of an active-set method for 1 minimization*, Optimization Methods and Software 27 (2012), pp. 1127–1146, Available at `http://dx.doi.org/10.1080/10556788.2011.591398`.
[29] S. Wright, *Coordinate descent algorithms*, Technical report, University of Wisconsin, Madison, Wisconsin, U.S.A., 2014.
[30] S. Wright, R. Nowak, and M. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Transactions on Signal Processing 57 (2009), pp. 2479–2493.
[31] G.X. Yuan, C.H. Ho, and C.J. Lin, *An improved glmnet for l1-regularized logistic regression*, The Journal of Machine Learning Research 13 (2012), pp. 1999–2030.

Appendix A. Convergence Analysis

Recall that we wish to solve the problem

$$\min_{x\in\mathbb{R}^n} \ \phi(x) = f(x) + \mu\|x\|_1.$$

For the purpose of our analysis, we make two assumptions:

ASSUMPTION A.1 *The function f is in C^1 and strongly convex with parameter $\lambda > 0$; i.e., for any $x, y \in \mathbb{R}^n$ and $t \in [0, 1]$:*

$$f(tx + (1-t)y) \le tf(x) + (1-t)f(y) - \frac{1}{2}\lambda t(1-t)\|x-y\|_2^2. \tag{A1}$$

As shown in Nesterov (2004), for continuously differentiable functions, this assumption is equivalent to

$$f(y) \ge f(x) + \nabla f(x)^T(y-x) + \frac{\lambda}{2}\|y-x\|_2^2 \qquad \text{for all } x, y \in \mathbb{R}^n. \tag{A2}$$

ASSUMPTION A.2 *The gradient of f is Lipschitz continuous with constant $L > 0$; i.e., for any $x, y \in \mathbb{R}^n$,*

$$\|\nabla f(x) - \nabla f(y)\|_2 \le L\|x-y\|_2$$

The first theorem shows that the algorithm is well-defined.

THEOREM A.3 *The backtracking projected line search (steps 20–22 of Algorithm 1) terminates in a finite number of iterations.*

Proof. Consider the k^{th} iteration of Algorithm 1. For notational simplicity, we drop the iteration index and denote the iterate as x, the direction obtained after the corrective loop (steps 10–15) as d, and the smooth and non-smooth quadratic approximations as

$\bar{q}$ and q, respectively. Along the same lines, let $\mathcal{A}$ and $\mathcal{U}$ be the active and unsure sets during this iteration.

We first show that there exists an $\bar{\alpha} > 0$ such that for any $\alpha > 0$ with $\alpha \leq \bar{\alpha}$, we have $\mathcal{P}(x + \alpha d) = x + \alpha d$. Let $\mathcal{I}_1 = \{i \in \{1, 2, 3, \cdots, n\} : x_i \neq 0\}\}$ and $\mathcal{I}_2 = \{i \in \{1, 2, 3, \cdots, n\} : x_i = 0\}\}$, and let $\bar{\alpha} > 0$ such that $\bar{\alpha} < \left|\frac{x_i}{d_i}\right|$ for all $i \in \mathcal{I}_1$ with $d_i \neq 0$.

Let $\alpha > 0$ be such that $\alpha \leq \bar{\alpha}$ and ζ be defined in (12). We consider two cases:

- **Case 1:** $i \in \mathcal{I}_1$
  Because $\alpha \leq \bar{\alpha} < \left|\frac{x_i}{d_i}\right|$, it is clear that $\text{sgn}(x_i + \alpha d_i) = \text{sgn}(x_i) = \zeta_i$, and therefore $\mathcal{P}(x + \alpha d) = x + \alpha d$.
- **Case 2:** $i \in \mathcal{I}_2$
  By definition, $x_i = 0$. If $i \in \mathcal{A}$ in step 19 of Algorithm 1, $d_i = 0$ and $\text{sgn}(x_i + \alpha d_i) = \text{sgn}(x_i) = \zeta_i$. Otherwise, $i \in \mathcal{U}_F \subseteq \mathcal{U}$, and therefore $\text{sgn}(-g_i) = \zeta_i \in \{-1, 1\}$. Assume that $\zeta_i = 1$. Thus, $\text{sgn}(-g_i) = 1$, and since $V^k = \emptyset$ and $i \in \mathcal{U}_F$, $\text{sgn}(x^k + d^k) = \text{sgn}(d^k) = 1$, so $d_i > 0$ which in turn implies $\alpha d_i > 0$. The same conclusion can be made if $\zeta_i = -1$. Thus, $\mathcal{P}(x_i + \alpha d_i) = \mathcal{P}(\alpha d_i) = \alpha d_i = x_i + \alpha d_i$.

Because d is a minimizer of $\bar{q}(x + d)$ in some subspace, we have $\bar{q}(x + \alpha d) \leq \bar{q}(x)$ for sufficiently small $\alpha \leq \bar{\alpha}$. Then, $\mathcal{P}(x + \alpha d) = x + \alpha d$, i.e., $x + \alpha d$ is in the same orthant as x, and therefore $q(x + \alpha d) = \bar{q}(x + \alpha d) \leq \bar{q}(x) = q(x)$. As a consequence, the termination condition in the while-loop is satisfied after a finite number of iterations.

■

We now show that by ensuring that ϕ at the new iterate is no larger than the majorizing function Γ^k, we can establish linear convergence.

THEOREM A.4 *Suppose that Assumptions A.1 and A.2 hold. Then, the iterates $\{x^k\}$ generated by Algorithm 2 converge to the optimal solution $x^\star$ of problem* (1) *at a linear rate.*

Proof. Consider the k^{th} iteration of Algorithm 2. For notational simplicity, let us drop the iteration index and denote the minimum norm subgradient as g, the Hessian approximation as H, and the iterate as x. Further, as is well known, the ISTA point x^k_{ISTA}, computed in step 5 of Algorithm 2, is the minimizer of a proximal approximation of $\phi(x)$,

$$x^k_{\text{ISTA}} = \arg\min_y f(x) + (y - x)^T \nabla f(x) + \frac{L}{2}\|y - x\|_2^2 + \mu\|y\|_1. \tag{A3}$$

Because of Assumption A.2, for any $z_1, z_2 \in \mathbb{R}^n$,

$$f(z_2) \leq f(z_1) + \nabla f(z_1)^T (z_2 - z_1) + \frac{L}{2}\|z_2 - z_1\|_2^2. \tag{A4}$$

In particular, by setting $z_1 = x$, $z_2 = x^k_{\text{ISTA}}$, we get

$$\begin{aligned} \phi(x^k_{\text{ISTA}}) &= f(x^k_{\text{ISTA}}) + \mu\|x^k_{\text{ISTA}}\|_1 \\ &\leq f(x) + \nabla f(x)^T (x^k_{\text{ISTA}} - x) + \frac{L}{2}\|x^k_{\text{ISTA}} - x\|_2^2 + \mu\|x^k_{\text{ISTA}}\|_1 \equiv \Gamma^k. \end{aligned} \tag{A5}$$

Let us denote the point obtained as a consequence of the globalization mechanism, which will be the new iterate, as x^+. This corresponds to x^{k+1} in step 14 of Algorithm 2. Realize that the loop in steps 8–13 of Algorithm 2 terminates finitely because once

$\bar{\alpha}$ drops to a value below ϵ, it is set to 0 and then $\phi(x^k_{\text{ISTA}} + \bar{\alpha}\bar{d}) = \phi(x^k_{\text{ISTA}})$ and the sufficiency condition (step 8 of Algorithm 2) is trivially satisfied by (A5).

By design, our algorithm generates the point x^+ such that

$$\phi(x^+) \leq f(x) + \nabla f(x)^T(x^k_{\text{ISTA}} - x) + \frac{L}{2}\|x^k_{\text{ISTA}} - x\|^2 + \mu\|x^k_{\text{ISTA}}\|_1. \tag{A6}$$

Combining this equation with the fact that x^k_{ISTA} is the minimizer in objective (A3), we have for any $d \in \mathbb{R}^n$ and $y = x + \frac{\lambda}{L}d$ that

$$\begin{aligned}
\phi(x^+) &\leq f(x) + \nabla f(x)^T\left(\frac{\lambda}{L}d\right) + \frac{L}{2}\left\|\frac{\lambda}{L}d\right\|_2^2 + \mu\left\|x + \frac{\lambda}{L}d\right\|_1 \\
&\leq \phi\left(x + \frac{\lambda}{L}d\right) - \frac{\lambda}{2}\left\|\frac{\lambda}{L}d\right\|_2^2 + \frac{L}{2}\left\|\frac{\lambda}{L}d\right\|_2^2 \\
&= \phi\left(x + \frac{\lambda}{L}d\right) + \frac{\lambda^2}{2L}\left(1 - \frac{\lambda}{L}\right)\|d\|_2^2,
\end{aligned}$$

where the second inequality follows from (A2) with $y = x + \frac{\lambda}{L}d$. In particular, we can set d to be $x^\star - x$ and obtain

$$\phi(x^+) \leq \phi\left(x + \frac{\lambda}{L}(x^\star - x)\right) + \frac{\lambda^2}{2L}\left(1 - \frac{\lambda}{L}\right)\|x^\star - x\|_2^2. \tag{A7}$$

Using Assumption A.1 and the convexity of the ℓ_1-norm, we have, for any $z_1, z_2 \in \mathbb{R}^n$ and $t \in [0, 1]$,

$$\phi(tz_1 + (1-t)z_2) \leq t\phi(z_1) + (1-t)\phi(z_2) - \frac{1}{2}\lambda t(1-t)\|z_1 - z_2\|_2^2.$$

Setting $z_1 = x$, $z_2 = x^\star$, and $t = \left(1 - \frac{\lambda}{L}\right)$, we get

$$\phi\left(x + \frac{\lambda}{L}(x^\star - x)\right) \leq \frac{\lambda}{L}\phi(x^\star) + \left(1 - \frac{\lambda}{L}\right)\phi(x) - \frac{\lambda^2}{2L}\left(1 - \frac{\lambda}{L}\right)\|x^\star - x\|_2^2.$$

Combining this result with (A7), we get

$$\begin{aligned}
\phi(x^+) &\leq \frac{\lambda}{L}\phi(x^\star) + \left(1 - \frac{\lambda}{L}\right)\phi(x) - \frac{\lambda^2}{2L}\left(1 - \frac{\lambda}{L}\right)\|x^\star - x\|_2^2 + \frac{\lambda^2}{2L}\left(1 - \frac{\lambda}{L}\right)\|x^\star - x\|_2^2 \\
&= \phi(x^\star) + \left(1 - \frac{\lambda}{L}\right)(\phi(x) - \phi(x^\star)),
\end{aligned}$$

and therefore

$$\phi(x^+) - \phi(x^\star) \leq \left(1 - \frac{\lambda}{L}\right)(\phi(x) - \phi(x^\star)).$$

By reintroducing the iteration index and using recursion, we have that

$$\phi(x^k) - \phi(x^\star) \leq \left(1 - \frac{\lambda}{L}\right)^k (\phi(x^0) - \phi(x^\star))$$

as required.

■

Appendix B. Reproducible Research

The MATLAB code used to generate the "Synthetic" problem is presented below. Given a dimension `n`, we use the following code snippet to generate the vector of labels (denoted by `y`) and the data matrix (denoted by `X`).

```
y=-1+(rand(n,1)>0.5)*2;
X = rand(n,n);
X = X + X';
mineig = min(eig(X));
if(mineig<0)
 X = eye(size(X))*mineig*-2+X;
end
X = chol(X);
```