A Family of Second-Order Methods for Convex ℓ_1-Regularized Optimization

Richard H. Byrd* Gillian M. Chin† Jorge Nocedal ‡ Figen Oztoprak§

June 25, 2012

Abstract

This paper is concerned with the minimization of an objective that is the sum of a convex function f and an ℓ_1 regularization term. Our interest is in methods that incorporate second-order information about the function f to accelerate convergence. We describe a semi-smooth Newton framework that can be used to generate a variety of second-order methods, including block active-set methods, orthant-based methods and a second-order iterative soft-thresholding method. We also propose a new active set method that performs multiple changes in the active manifold estimate, and incorporates a novel mechanism for correcting estimates, when needed. This corrective mechanism is also evaluated in an orthant-based method. Numerical tests comparing the performance of several second-order methods are presented.

1 Introduction

We consider convex optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \ \phi(x) \stackrel{\text{def}}{=} f(x) + \mu\|x\|_1, \tag{1.1}$$

where f is a smooth, convex function and $\mu > 0$ is a (fixed) regularization parameter. The motivation for this work stems from machine learning applications in which f is a stochastic loss function and the regularization term promotes both sparsity in the solution and a lower generalization error, but the methods discussed here are also useful in applications where f is a deterministic function.

*Department of Computer Science, University of Colorado, Boulder, CO, USA. This author was supported by National Science Foundation grant CMMI 0728190 and Department of Energy grant DE-SC0001774.

†Department of Industrial Engineering and Management Sciences, Northwestern University. This author was supported by an NSERC fellowship and a grant from Google Inc.

‡Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA. This author was supported by National Science Foundation grant DMS-0810213, and by Department of Energy grant DE-FG02-87ER25047.

§Department of Computer Science, University of Colorado, Boulder, CO, USA. This author was supported by Department of Energy grant DE-SC0001774.

Various first-order methods have been proposed for solving problem (1.1), and have proved effective in some important large scale applications (see, e.g. [23, 28] and the references therein). The focus of this paper is on methods that incorporate second-order information about the function f to accelerate convergence. These methods are effective in many large scale applications, particularly when parallel computing environments are available.

The contributions of this paper are two-fold. First, we propose a unified framework for generating a wide variety of second-order methods for the convex regularized problem (1.1). This framework is based on a semi-smooth Newton approach, and allows us to contrast various algorithms, both theoretically and numerically. The second contribution is to propose a new active-set method that performs multiple changes in the active manifold estimate at every iteration. It can be derived using the semi-smooth Newton framework, and includes a novel mechanism that dynamically corrects undesirable updates in the active manifold selection. The new algorithm is designed for the case when f is quadratic, but it can be used as a basis for a Newton-like method for solving the general convex problem (1.1).

Other methods that can be derived by appropriate choices of the free parameters in our semi-smooth Newton framework include orthant-based methods [2], and a second-order iterative thresholding method. All these methods can be viewed as two-phase methods, consisting of an active manifold identification phase and a second-order subspace phase.

Notation. Throughout the paper we define $\nabla_i f = \partial f / \partial x_i$. We use calligraphic $\mathcal{A}, \mathcal{P}, \mathcal{N}$ to denote index sets in the semi-smooth Newton method of section 2, and Roman A, P, N for the index sets in the block active-set method of section 3.

2 A Framework for Second-Order Methods

The convex problem (1.1) is non-smooth, but its subdifferential is easy to compute and is employed in various first-order methods; see, e.g. [3, 10, 11, 31, 32]. The goal of this section is to describe a framework for generating methods that incorporate second-order information about the function f.

This framework is based on the semi-smooth Newton methodology [12, 25, 26, 22, 29], which has been developed for solving non-differentiable systems of algebraic equations that can be classified as being semi-smooth. This methodology is relevant because the optimality conditions for problem (1.1) can be written as a semi-smooth system, as we now discuss.

Let x^* denote an optimal solution of (1.1), and let us define the sets

$$\mathcal{A}^* = \{i : x_i^* = 0\}, \quad \mathcal{P}^* = \{i : x_i^* > 0\}, \quad \mathcal{N}^* = \{i : x_i^* < 0\}.$$

Then, x^* satisfies the optimality conditions

$$0 \in \partial\phi(x^*) \quad \Longleftrightarrow \quad \begin{cases} 0 = \nabla_i f(x^*) + \mu & \text{for } i \in \mathcal{P}^* \\ 0 = \nabla_i f(x^*) - \mu & \text{for } i \in \mathcal{N}^* \\ 0 \in [\nabla_i f(x^*) - \mu, \nabla_i f(x^*) + \mu] & \text{for } i \in \mathcal{A}^*. \end{cases} \tag{2.1}$$

It is well-known and easy to verify that these conditions are equivalent to the nonlinear system $F(x^*) = 0$, where $F : \mathbb{R}^n \to \mathbb{R}^n$ is given by

$$F_i(x) = \max\left\{\min\{\tau(\nabla_i f(x) + \mu), x_i\}, \tau(\nabla_i f(x) - \mu)\right\}, \quad i = 1, ..., n, \tag{2.2}$$

and τ is any positive scalar. The function F can also be written as

$$F_i(x) = \ \text{proj}_{[(\tau(\nabla_i f(x)-\mu),\tau(\nabla_i f(x)+\mu)]}(x_i), \quad i = 1, ..., n, \tag{2.3}$$

where $\text{proj}_{[a,b]}(x)$ denotes the projection of x onto the interval $[a, b]$. In stating the optimality conditions $F(x^*) = 0$, the scalar $\tau > 0$ has no effect, but it plays a vital role in the algorithms we derive below.

The search direction in a semi-smooth Newton method for solving $F(x) = 0$ is given by

$$G(x)d = -F(x), \tag{2.4}$$

where $G(x)$ denotes the generalized Jacobian of F. In order to define G, we note that the function $F_i(x)$ is non-differentiable when

$$x_i = \tau(\nabla_i f(x) - \mu) \qquad \text{or} \qquad x_i = \tau(\nabla_i f(x) + \mu). \tag{2.5}$$

Based on this observation, we define the following index sets at an iterate x^k:

$$\mathcal{N}^k = \{i : x_i^k \le \tau(\nabla_i f(x^k) - \mu)\}, \tag{2.6}$$

$$\mathcal{A}^k = \{i : \tau(\nabla_i f(x^k) - \mu) \le x_i^k \le \tau(\nabla_i f(x^k) + \mu)\}, \tag{2.7}$$

$$\mathcal{P}^k = \{i : x_i^k \ge \tau(\nabla_i f(x^k) + \mu)\}. \tag{2.8}$$

The generalized Jacobian G is defined in terms of the convex hull of all directional derivatives of the component functions F_i at x^k [12]. As a result of the index sets defined in (2.6)-(2.8), the semi-smooth Newton iteration (2.4) for the function (2.2) can be written as:

$$e_i^T d^k = -x_i^k, \qquad i \in \mathcal{A}^k \setminus (\mathcal{N}^k \cup \mathcal{P}^k) \tag{2.9a}$$

$$\nabla^2_{i:} f(x^k) d^k = -(\nabla_i f(x^k) + \mu), \qquad i \in \mathcal{P}^k \setminus \mathcal{A}^k \tag{2.9b}$$

$$\nabla^2_{i:} f(x^k) d^k = -(\nabla_i f(x^k) - \mu), \qquad i \in \mathcal{N}^k \setminus \mathcal{A}^k \tag{2.9c}$$

$$\left(\delta_i \nabla^2_{i:} f(x^k) + (1 - \delta_i) e_i^T\right) d^k = -\hat{y}_i^k, \qquad i \in \mathcal{N}^k \cap \mathcal{A}^k \tag{2.9d}$$

$$\left(\delta_i \nabla^2_{i:} f(x^k) + (1 - \delta_i) e_i^T\right) d^k = -\bar{y}_i^k, \qquad i \in \mathcal{P}^k \cap \mathcal{A}^k. \tag{2.9e}$$

$$x^{k+1} = x^k + d^k, \tag{2.9f}$$

where $\nabla^2_{i:} f$ indicates the i^{th} row of the Hessian $\nabla^2 f$, the vector e_i is the unit vector in $\mathbb{R}^n$, and

$$\hat{y}_i^k \stackrel{\text{def}}{=} \tau(\nabla_i f(x^k) - \mu) = x_i^k, \ i \in \mathcal{N}^k \cap \mathcal{A}^k, \qquad \bar{y}_i^k \stackrel{\text{def}}{=} \tau(\nabla_i f(x^k) + \mu) = x_i^k, \quad i \in \mathcal{P}^k \cap \mathcal{A}^k.$$

Different choices of τ and δ_i define different semi-smooth Newton methods. The value of τ determines the points of non-differentiability (2.5), and thus the definitions of the sets $\mathcal{N}^k, \mathcal{P}^k$, and $\mathcal{A}^k$. Correspondingly, the choice of δ_i determines the choice of the generalized Jacobian at these points of non-differentiability. For the components in (2.9a), we have $x_i^{k+1} = x_i^k + d_i^k = 0$, meaning that these variables are set to zero. If we chose to set $\delta_i = 0$, equations (2.9d), (2.9e) coincide with (2.9a), and these variables are also set to zero. If we instead set $\delta_i = 1$, we allow the variables in (2.9d), (2.9e) to move; we refer to these indices as free variables, and update them with a Newton-like iteration.

The following algorithms are obtained by suitable choices of the parameters τ and δ in the semi-smooth Newton framework.

Iterative Shrinkage Plus Subspace Step. If we set $\delta_i = 0$, and choose τ as a step length parameter, the semi-smooth Newton iteration (2.9) yields a two-phase method in which the iterative shrinkage thresholding (ISTA) iteration [8, 10] is used to identify fixed variables, and free variables are updated using a second-order step.

To see this, note that by (2.7) variables x_i^k with

$$-\tau\mu \leq x_i^k - \tau\nabla_i f(x^k) \leq \tau\mu \tag{2.10}$$

are assigned to the set $\mathcal{A}^k$ in the semi-smooth Newton method, and if we set $\delta_i = 0$ in (2.9d)-(2.9e), all variables satisfying (2.10) are set to zero. This is precisely the effect of the ISTA iteration, which is given by

$$\bar{x}^k = \mathcal{T}_{\alpha^k\mu}(x^k - \alpha^k\nabla f(x^k)), \quad \text{where} \quad \mathcal{T}_\sigma(y)_i = (|y_i| - \sigma)^+\text{sgn}(y_i), \tag{2.11}$$

provided we choose $\tau = \alpha^k$. The rest of the variables in this second-order ISTA method, namely those in (2.9b)-(2.9c), are free and are updated by a Newton-like iteration. Several choices for the steplength α^k in (2.11) have been proposed in the literature [28], and any of these step lengths can be used as the value of τ in the semi-smooth Newton iteration. The methods described in [14, 30] can be characterized as second-order ISTA methods.

Orthant-Based Methods (OBM). These are two-phase methods in which an *active orthant face* is first identified, and a subspace minimization is then performed with respect to the variables that define the orthant face. A line search ensures that the new iterate belongs to the active orthant [2, 7, 18].

Given an iterate x^k, an orthant-based method defines the orthant face Ω_k by

$$\Omega_k = \mathbf{cl}(\{d \in \mathbb{R}^n : \text{sgn}(d_i) = \text{sgn}(\xi_i^k),\ i = 1, ..., n\}), \quad \text{with} \quad \xi_i^k = \begin{cases} \text{sgn}(x_i^k) & \text{if } x_i^k \neq 0 \\ \text{sgn}(-g_i^k) & \text{if } x_i^k = 0, \end{cases} \tag{2.12}$$

where g^k is the minimum norm subgradient of ϕ computed at x^k, [27]:

$$g_i^k = \begin{cases} \nabla_i f(x^k) + \mu & \text{if } x_i^k > 0 \quad \text{or} \quad (x_i^k = 0 \ \wedge \ \nabla_i f(x^k) + \mu < 0) \\ \nabla_i f(x^k) - \mu & \text{if } x_i^k < 0 \quad \text{or} \quad (x_i^k = 0 \ \wedge \ \nabla_i f(x^k) - \mu > 0) \\ 0 & \text{if } x_i^k = 0 \quad \text{and} \quad 0 \in [\nabla_i f(x^k) - \mu, \nabla_i f(x^k) + \mu]. \end{cases} \tag{2.13}$$

In the relative interior of Ω_k, the function ϕ is differentiable. The active set in an orthant-based method, defined as $A^k = \{i : \xi_i^k = 0\}$, determines the variables that are kept at zero, while the rest of the variables are chosen to minimize a quadratic model of ϕ at x^k. Specifically, the step d^k of the algorithm is computed as the solution of

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \Psi(d) = \phi(x^k) + d^T g^k + \tfrac{1}{2} d^T \nabla^2 f(x^k) d \\ \text{s.t.} \quad & d_i = 0,\ i \in A^k. \end{aligned} \tag{2.14}$$

The algorithm then performs a line search along the step d^k, projecting the iterate back onto the orthant face, Ω_k.

To generate orthant-based methods from the semi-smooth Newton iteration (2.9), we choose τ sufficiently small such that

$$\operatorname{sgn}\Big(x_i^k - \tau\big(\nabla_i f(x^k) + \operatorname{sgn}(x_i^k)\mu\big)\Big) = \operatorname{sgn}(x_i^k), \qquad \text{for all } i \text{ such that } x_i^k \neq 0, \tag{2.15}$$

and set

$$\delta_i = 0, \qquad \text{for all } i \in (\mathcal{N}^k \cap \mathcal{A}^k) \cup (\mathcal{P}^k \cap \mathcal{A}^k), \tag{2.16}$$

that is, when F is non-differentiable, which, by (2.15), can only occur when $x_i^k = 0$. To observe the effect of this choice of τ and δ, let us first consider those variables for which $x_i^k \neq 0$. The choice of τ given in (2.15) implies that

$$\begin{aligned} &\text{If } \operatorname{sgn}(x_i^k) = -1, \text{ then } \quad x_i^k - \tau(\nabla_i f(x^k) - \mu) < 0, \text{ which by (2.6) implies } i \in \mathcal{N}^k \setminus \mathcal{A}^k; \\ &\text{If } \operatorname{sgn}(x_i^k) = +1, \text{ then } \quad x_i^k - \tau(\nabla_i f(x^k) + \mu) > 0, \text{ which by (2.8) implies } i \in \mathcal{P}^k \setminus \mathcal{A}^k. \end{aligned}$$

By (2.9*b*)-(2.9*c*) these are free variables in the semi-smooth Newton iteration—as well as in the orthant-based method, by the first condition in (2.12). For variables such that $x_i^k = 0$ we have:

$$\begin{aligned} &\text{If } \quad \nabla_i f(x^k) - \mu > 0, \text{ then by (2.6), } i \in \mathcal{N}^k, \text{and by (2.12), (2.13), } \xi_i^k = -1; \\ &\text{If } \quad \nabla_i f(x^k) + \mu < 0, \text{ then by (2.8), } i \in \mathcal{P}^k, \text{and by (2.12), (2.13), } \xi_i^k = 1; \\ &\text{Else } \quad 0 \in [\nabla_i f(x^k) - \mu, \nabla_i f(x^k) + \mu]. \text{ By (2.12)-(2.13) we have } \xi_i^k = 0. \end{aligned}$$

For the semi smooth Newton method, by (2.7), there are two cases:

1. If (2.5) holds, then either $i \in \mathcal{A}^k \cap \mathcal{N}^k$ or $i \in \mathcal{A}^k \cap \mathcal{P}^k$. Given (2.16), $\delta_i = 0$ and these variables are held at zero.
2. If (2.5) does not hold, then $i \in \mathcal{A}^k \setminus (\mathcal{N}^k \cup \mathcal{P}^k)$ which are also kept at zero.

Therefore, for the choices of τ and δ mentioned above, the sets of free and fixed variables in the semi-smooth Newton method identify the same orthant face that is used in orthant-based methods. Furthermore, both methods compute the same subspace step, as evidenced by the comparison of (2.9a)-(2.9e) with (2.14). Thus, if we were to replace (2.9f) with an update resulting from a projected line search along the direction generated by (2.9a)-(2.9e), both methods would yield the same iterate.

Block Active-Set Method. If the function f in (1.1) is a convex quadratic, and we set τ such that

$$\tau > \frac{|x_i^k|}{2\mu}, \text{ for all } i, \quad \text{and} \quad \delta_i = 1, \text{ for } i \text{ such that (2.5) holds,} \tag{2.17}$$

then the semi-smooth Newton iteration (2.9) gives rise to the block active-set method (BAS), discussed in the next section. The proof of this claim is given in Section 3.1.

Other interesting methods, such as a two metric gradient projection method [6], can be generated by appropriate choices of τ and δ in the semi-smooth Newton framework. Furthermore, we can derive quasi-Newton variants for all methods described above by replacing $\nabla^2 f(x)$ by a BFGS approximation in (2.9*b*)-(2.9*e*). In fact, in Section 6 we report results for a block active-set method and an orthant-based method that both employ limited memory BFGS approximations.

3 The Block Active-Set Method (BAS)

This method was originally proposed for solving linear complementarity problems involving M-matrices [1] (see also [19]), and has been extended to regularized linear regression problems [20]. It has recently received much attention for the solution of bound constrained quadratic problems [5, 16], and we review the algorithm in this context.

Let us consider the problem

$$\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2} x^T Q x + q^T x, \quad \text{subject to} \quad x \geq 0, \tag{3.1}$$

where $Q \in \mathbb{R}^{n \times n}$ is symmetric positive definite. The KKT conditions of this problem are

$$Qx + q = w, \tag{3.2}$$

$$w_i = 0 \quad \text{for all } i \text{ such that } x_i > 0, \tag{3.3}$$

$$w \geq 0, \quad x \geq 0. \tag{3.4}$$

The block active-set (BAS) method promotes fast changes in the active set using a non-standard primal-dual mechanism. Every iteration begins with the specification of two index sets: A^k, which indicates the components of x that will be held at zero at the current iteration, and I^k, which represents the free components of x that are allowed to change. Given a primal-dual iterate (x^k, w^k), the set A^k is composed of variables that violate the constraints, in addition to variables at their bounds with positive ("correct") multipliers, i.e.,

$$A^k = \{i : x_i^k < 0\} \ \cup \ \{i : x_i^k = 0 \text{ and } w_i^k > 0\}. \tag{3.5}$$

The set I^k is composed of the rest of the variables.

The next primal-dual iterate (x^{k+1}, w^{k+1}) is obtained by first setting

$$x_i^{k+1} = 0, \text{ for } i \in A^k, \quad \text{and} \quad w_i^{k+1} = 0, \text{ for } i \in I^k, \tag{3.6}$$

and then computing the rest of the variables so as to satisfy (3.2) and (3.3) i.e.,

$$[Qx^{k+1} + q]_i = 0, \quad \text{for } i \in I^k, \qquad w_i^{k+1} = [Qx^{k+1} + q]_i, \quad \text{for } i \in A^k. \tag{3.7}$$

To contrast this algorithm with more traditional active-set methods [24, S16.5], a classical active-set algorithm drops at most one constraint (with a negative multiplier) at each iteration. In the BAS method, many variables are allowed to violate their bounds, and each iteration drops and adds many constraints to the active set. While this methodology may seem overly aggressive in comparison to classical methods, it has proved to be very efficient for problems of the form (3.1) where Q is an M-matrix, particularly on some problems arising from the discretization of partial differential equations [16].

The BAS method has been modified in [19] to solve problems associated with symmetric positive definite matrices, where the algorithm reverts to a traditional active set iteration when sufficient progress is not observed; Section 4.1 provides a general outline of this strategy.

To extend the BAS algorithm to the ℓ_1 regularized quadratic problem given by

$$\min_{x \in \mathbb{R}^n} \ \varphi(x) = \tfrac{1}{2} x^T Q x + q^T x + \mu \|x\|_1, \tag{3.8}$$

where Q is symmetric positive definite, we introduce auxiliary variables $u, v \geq 0$, write $x = u - v$, and reformulate (3.8) as a bound constrained problem of the form (3.1). Next, we apply the BAS method given by (3.5)-(3.7) to this bound constrained problem. While introducing the auxiliary variables seemingly doubles the dimension of the problem, it can be shown that the BAS algorithm, if properly initialized, maintains complementarity between the u and v variables. Therefore, it suffices to keep track of the positive and negative components and maintain a single vector x, together with index sets P^k and N^k. As shown in the Appendix, this strategy gives rise to Algorithm 3.1.

Algorithm 3.1: Block Active-Set Algorithm for the Quadratic L1 Problem (3.8)

Choose an initial iterate x^0 and compute $w^0 = [Qx^0 + q]$.
Initialize the sets: $P^{-1} = \{i : x_i^0 > 0\}$, $N^{-1} = \{i : x_i^0 < 0\}$ and $A^{-1} = \{i : x_i^0 = 0\}$.

for $k = 0, \cdots$, until the optimality conditions of (3.8) are satisfied:

1. Compute the index sets: P^k, N^k, A^k:

$$P^k = \{i \in P^{k-1} : x_i^k \geq 0\} \cup \{i \in A^{k-1} : w_i^k \leq -\mu\}, \tag{3.9a}$$
$$N^k = \{i \in N^{k-1} : x_i^k \leq 0\} \cup \{i \in A^{k-1} : w_i^k \geq \mu\}, \tag{3.9b}$$
$$A^k = \{i \in A^{k-1} : w_i^k \in (-\mu, \mu)\} \cup \{i \in P^{k-1} : x_i^k < 0\} \tag{3.9c}$$
$$\cup \{i \in N^{k-1} : x_i^k > 0\}.$$

2. Compute the iterate (x^{k+1}, w^{k+1}) satisfying

$$x_i^{k+1} = 0 \qquad \forall\ i \in A^k, \tag{3.10a}$$
$$[Qx^{k+1} + q + \mu e]_i = 0 \qquad \forall\ i \in P^k, \tag{3.10b}$$
$$[Qx^{k+1} + q - \mu e]_i = 0 \qquad \forall\ i \in N^k, \tag{3.10c}$$
$$w^{k+1} = Qx^{k+1} + q. \tag{3.10d}$$

end(for)

A point of note is that Algorithm 3.1 maintains pairwise disjoint sets for P^k, N^k and A^k at each iteration, k. Furthermore, we note that Algorithm 3.1 is very similar to the block active set strategy presented in [20] for ℓ_1 regularized linear regression problems. That algorithm operates on the dual formulation, for which the problem is a bound constrained quadratic problem.

To describe the stopping conditions of Algorithm 3.1, an iterate x^k satisfies the optimality conditions of problem (3.8) if

$$(Qx^k + q)_i \in [-\mu, \mu],\ i \in A^{k-1}, \qquad x_i^k \geq 0,\ \ i \in P^{k-1}, \qquad x_i^k \leq 0,\ i \in N^{k-1}. \tag{3.11}$$

To demonstrate the veracity of this statement, we establish that these conditions are equivalent to (2.1). For $i \in A^{k-1}$ we have $x_i^k = 0$ and the first condition in (3.11) matches the third condition in (2.1). For $i \in P^{k-1}$ we have from (3.10*b*) that $[Qx^k + q + \mu e]_i = 0$, or equivalently $\nabla_i f(x^k) + \mu = 0$. If $x_i^k > 0$, this is the first condition of (2.1); if $x_i^k = 0$, which satisfies the third condition of (2.1). Therefore, the second condition in (3.11) is consistent with (2.1), and a similar argument applies to indices satisfying the third condition in (3.11).

3.1 Derivation of the BAS method from the semi-smooth Newton framework

The BAS method has been derived from the semi-smooth Newton framework by [16]. Here we provide a derivation based on a different semi-smooth function (2.2), which is useful in the derivation of the algorithms presented in this paper. Since the problem under consideration is of the form (3.8), it follows from (1.1), that $f(x) = \frac{1}{2}x^T Qx + q^T x$. The variable w^k used in Algorithm 3.1 therefore satisfies $w^k = \nabla f(x^k)$. To establish the relationship between the two methods, we will use an induction-based argument. Let us assume that x^{k-1} is the same in both methods, and that the index sets $\mathcal{P}^{k-1}$, $\mathcal{N}^{k-1}$, $\mathcal{A}^{k-1}$ of the semi-smooth Newton method and the indices $P^{k-1}, N^{k-1}, A^{k-1}$ of Algorithm 3.1 satisfy

$$P^{k-1} = \mathcal{P}^{k-1}, \quad N^{k-1} = \mathcal{N}^{k-1}, \quad A^{k-1} = \mathcal{A}^{k-1} \setminus (\mathcal{P}^{k-1} \cup \mathcal{N}^{k-1}). \tag{3.12}$$

Given this assumption, and for the choices τ and δ in (2.17), we can show that at iteration $k-1$ the step computed by both methods result in the same iterate x^k, and that the relationship (3.12) continues to hold at iteration k. To do so, we consider the following cases:

1. Suppose that $i \in \mathcal{A}^{k-1} \setminus (\mathcal{P}^{k-1} \cup \mathcal{N}^{k-1})$, and hence by (3.12), $i \in A^{k-1}$. It follows from (2.9a), (2.9f) and (3.10a) that $x_i^k = 0$ for both methods, and thus, for such variables, the step computations yield the same iterate. Furthermore, we have from (3.9c) that if Algorithm 3.1 assigns $i \in A^k$, then by (2.7), the semi-smooth Newton method sets $i \in \mathcal{A}^k \setminus (\mathcal{P}^k \cup \mathcal{N}^k)$. If instead, Algorithm 3.1 sets $i \in P^k$, then by (3.9a), we must have $\nabla_i f^k \leq -\mu$, and by (2.8), this results in the semi-smooth Newton method assigning $i \in \mathcal{P}^k$. (A similar argument applies to the case where Algorithm 3.1 sets $i \in N^k$). Thus, for variables $i \in \mathcal{A}^{k-1} \setminus (\mathcal{P}^{k-1} \cup \mathcal{N}^{k-1})$, both methods yield the same iterate, and (3.12) continues to hold at iteration k.

2. Suppose $i \in P^{k-1} = \mathcal{P}^{k-1}$. Given the selection of $\delta_i = 1$ in (2.17), the semi-smooth Newton equation (2.9e) coincides with equation (2.9b), and since f is a quadratic function, these equations are as follows:

$$Q_{i:}d^{k-1} = -(Qx^{k-1} + q)_i - \mu = -Q_{i:}x^{k-1} - q_i - \mu, \quad \text{for } i \in \mathcal{P}^{k-1},$$

   where $Q_{i:}$ indicates the i^{th} row of matrix Q. Therefore,

$$[Qx^k + q + \mu e]_i = \nabla_i f(x^k) + \mu = 0, \quad i \in \mathcal{P}^{k-1}, \tag{3.13}$$

   which coincides with (3.10*b*), for $i \in P^{k-1}$, in Algorithm 3.1.

   Let us now show that the assignment of the indices $i \in P^{k-1} = \mathcal{P}^{k-1}$ at the next iteration k, satisfies (3.12).

   A. If $x_i^k > 0$ then by (3.13) and (2.8), $i \in \mathcal{P}^k$ in the semi-smooth Newton method, and by (3.9*a*), $i \in P^k$ in Algorithm 3.1.

   B. If $x_i^k = 0$, it follows from (3.13), (2.7) and (2.8) that $i \in \mathcal{P}^k \cap \mathcal{A}^k$. Similarly, by (3.9*a*), $i \in P^k$ for Algorithm 3.1.

C. If $x_i^k < 0$ then by (3.9c), $i \in A^k$ for the Algorithm 3.1. Since $x_i^k < 0$, and by (3.13), we have that the second inequality in (2.7) is satisfied strictly. On the other hand, by (3.13), (2.17) and the assumption $x_i^k < 0$, we obtain

$$\tau(\nabla_i f(x^k) - \mu) = -2\tau\mu < x_i^k,$$

showing that the first inequality in (2.7) is satisfied as a strict inequality. Thus, $i \in \mathcal{A}^k \setminus (\mathcal{P}^k \cup \mathcal{N}^k)$ for the semi-smooth method.

As the analysis for indices $i \in N^{k-1} = \mathcal{N}^{k-1}$ yields similar conclusions, we have proven for $i \in N^{k-1} \cup P^{k-1} = \mathcal{P}^{k-1} \cup \mathcal{N}^{k-1}$ that both methods yield the same iterate during iteration $k-1$, and that relations (3.12) are preserved at iteration k.

Therefore, we have shown by induction that, if initialized so that their starting point coincide and the index sets satisfy (3.12), the semi-smooth Newton method with parameters (2.17) and Algorithm 3.1 are equivalent, when applied to problem (3.8).

4 A New Block Active-Set Algorithm

Algorithm 3.1, like the original BAS method, can fail to converge when Q in (3.8) is not an M-matrix [4]. It is, in fact, common to observe both methods fail (cycle) in practice. To address these shortcomings, we modify the active set mechanism of the BAS method, based on the observation that the aggressive changes in index sets is the cause of cycling across recurring instances of the sets P, N and A. We identify which of these changes may not be productive, and correct them.

We note that P^k can be interpreted as the set of variables that are *predicted* to be non-negative. If $i \in A^{k-1} \cap P^k$, then $x_i^k = 0$, and it follows from (3.9) that $w_i^k \leq -\mu$, which indicates that this variable should be increased. Moreover, if $i \in P^{k-1} \cap P^k$, it follows that this variable was predicted to be non-negative at iteration $k-1$, and x_i^k was indeed non-negative; thus we predict this variable to also be non-negative at the next iteration. Similarly, we interpret N^k as the set of variables that are predicted to be non-positive. However, since the algorithm is an infeasible method, it is possible that x_i^{k+1} could be negative for $i \in P^k$, or could be positive for $i \in N^k$, at the end of iteration k, in which case our prediction was incorrect.

Our strategy is as follows. After computing a trial iterate (x^{k+1}, w^{k+1}), we observe whether variables *that are zero* at the beginning of the iteration were incorrectly predicted; that is, if zero-valued variables that were predicted to be non-negative, became negative; or if zero variables that were predicted non-positive, became positive. In other words, we define the sets

$$V_k^P = \{i \in P^k \,|\, x_i^k = 0 \text{ and } x_i^{k+1} < 0\}, \quad V_k^N = \{i \in N^k \,|\, x_i^k = 0 \text{ and } x_i^{k+1} > 0\}, \tag{4.1}$$

and determine whether the set V_k is non-empty, where

$$V_k = V_k^P \cup V_k^N.$$

If V_k is not empty, all variables $i \in V_k$ are removed from their respective free sets and reassigned to the active set A^k. A step of the algorithm is then re-computed using the new sets P^k, N^k, A^k, thus holding the indices $i \in V_k$, at zero. This corrective procedure is repeated until all predictions for zero-valued variables are correct, i.e., until V_k is empty. The implementation of this strategy within the BAS method is described in Algorithm 4.1.

Algorithm 4.1: Corrected Block Active-Set Algorithm for the Quadratic L1 Problem (3.8)

Initialize x^0, w^0, and the index sets P^{-1}, N^{-1}, A^{-1} as in Algorithm 3.1.
for $k = 0, \cdots$, until the optimality conditions (3.11) are satisfied:

1. Compute the index sets: P^k, N^k, A^k:

$$\begin{aligned} P^k &= \{i \in P^{k-1} : x_i^k \geq 0\} \cup \{i \in A^{k-1} : w_i^k \leq -\mu\}, && \text{(4.2a)} \\ N^k &= \{i \in N^{k-1} : x_i^k \leq 0\} \cup \{i \in A^{k-1} : w_i^k \geq \mu\}, && \text{(4.2b)} \\ A^k &= \{i \in A^{k-1} : w_i^k \in (-\mu, \mu)\} \cup \{i \in P^{k-1} : x_i^k < 0\} && \text{(4.2c)} \\ &\qquad \cup \{i \in N^{k-1} : x_i^k > 0\}. \end{aligned}$$

2. **Repeat** until $V_k = \emptyset$:

   2a) Compute the iterate (x^{k+1}, w^{k+1}) satisfying

$$\begin{aligned} x_i^{k+1} &= 0 && \forall\ i \in A^k, && \text{(4.3a)} \\ [Qx^{k+1} + q + \mu e]_i &= 0 && \forall\ i \in P^k, && \text{(4.3b)} \\ [Qx^{k+1} + q - \mu e]_i &= 0 && \forall\ i \in N^k, && \text{(4.3c)} \\ w_i^{k+1} &= [Qx^{k+1} + q]_i. && && \text{(4.3d)} \end{aligned}$$

   2b) Compute V_k^P, V_k^N by (4.1); set $V_k = V_k^P \cup V_k^N$.

   2c) $A^k \leftarrow A^k \cup V_k$.

   2d) $P^k \leftarrow P^k \setminus V_k^P, \quad N^k \leftarrow N^k \setminus V_k^N$.

   End repeat
end(for)

We note that the `repeat` loop in Algorithm 4.1 is finite since at each round, elements are subtracted from P^k or N^k, and these sets are finite.

We now show that Algorithm 4.1 will take a nonzero step from any iterate that is not a solution of problem (3.8). For convenience, we partition the variables according to their

transitions:

$$\begin{aligned} R^k &:= \{i : i \in (P^{k-1} \cap P^k) \cup (N^{k-1} \cap N^k)\} \\ S^k &:= \{i : i \in (P^{k-1} \cup N^{k-1}) \cap A^k\} \\ T^k &:= \{i : i \in A^{k-1} \cap (P^k \cup N^k)\} \\ W^k &:= \{i : i \in (A^{k-1} \cap A^k)\} \end{aligned}$$

and, by (4.2),

$$R^k \cup S^k \cup T^k \cup W^k = \{1 \dots n\}. \tag{4.4}$$

In addition it is convenient to define

$$s_i^k = \begin{cases} 1, & \text{if } i \in P^k \\ 0, & \text{if } i \in A^k \\ -1, & \text{if } i \in N^k. \end{cases} \tag{4.5}$$

Theorem 4.1 *Let $\{x^k\}$ be the iterates generated by Algorithm 4.1 when applied to problem* (3.8)*, where Q is symmetric and positive definite. Then, if x^k is not a stationary point for problem* (3.8)*, we have that $x^{k+1} \neq x^k$. Furthermore, either $S^k \neq \emptyset$, which means some nonzero variables are moved to zero, or $S^k = \emptyset$ in which case we have*

$$(Qx^k + q + s^k\mu)^T(x^{k+1} - x^k) = \sum_{i \in T^k} ([Qx^k]_i + q_i + s_i^k\mu)(x^{k+1} - x^k)_i < 0. \tag{4.6}$$

Proof. Let us assume that x^k is not optimal. We consider two cases at iteration k of Algorithm 4.1.

Case 1. Suppose that the set $S^k = (P^{k-1} \cup N^{k-1}) \cap A^k$ is nonempty at the beginning of Step 2 of Algorithm 4.1. Consider some $j \in S^k$; by (4.2c), we must have $x_j^k \neq 0$. Since these variables are not removed from A^k by the `repeat` loop, then by (4.3a), it follows that $x_j^{k+1} = 0 \neq x_j^k$, which implies $x^{k+1} \neq x^k$.

Case 2. Suppose that the set $S^k = (P^{k-1} \cup N^{k-1}) \cap A^k$ is empty at the beginning of Step 2. It follows by (4.2) that $P^{k-1} \subseteq P^k$ and $N^{k-1} \subseteq N^k$; therefore, $R^k = P^{k-1} \cup N^{k-1}$, which implies using (4.5) that $s_i^k = s_i^{k-1}$ for all $i \in R^k$. Given this stability of sign, it follows from the computation of step 2a of Algorithm 4.1 that

$$[Qx^k]_i + q_i + s_i^k\mu = [Qx^k]_i + q_i + s_i^{k-1}\mu = 0, \text{ for all } i \in R^k. \tag{4.7}$$

Let us consider the indices $i \in A^{k-1}$. Since $R^k = P^{k-1} \cup N^{k-1}$, the last two optimality conditions in (3.11) are satisfied, and since x^k is assumed to be non-stationary, it follows that at the beginning of Step 2, there must exist indices $i \in A^{k-1}$ such that

$$|w_i^k| = |[Qx^k]_i + q_i| > \mu; \tag{4.8}$$

moreover these indices will be initially assigned to $N^k \cup P^k$ by Step 1. Therefore, the set $T^k = A^{k-1} \cap (P^k \cup N^k)$ is nonempty at the beginning of Step 2.

We will now show that the set T^k is also nonempty at termination of the `repeat` loop in Step 2, which we have previously shown to be finite. We note that at each iteration of the `repeat` loop, x^{k+1} is computed as the minimizer of the strictly convex quadratic program

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \tfrac{1}{2}x^TQx + (q + s^k\mu)^Tx \\ &\text{subject to } x_i = 0 : i \in A^k, \end{aligned} \tag{4.9}$$

where s^k is defined by (4.5). Furthermore, since by (4.8) x^k is not a minimizer of the quadratic, $x^{k+1} - x^k$ is a strict descent direction for the quadratic, i.e.,

$$(Qx^k + q + s^k\mu)^T(x^{k+1} - x^k) = \sum_{i=1}^{n}([Qx^k]_i + q_i + s_i^k\mu)(x^{k+1} - x^k)_i < 0.$$

Given the assumption $S^k = (P^{k-1} \cup N^{k-1}) \cap A^k = \emptyset$, it follows that $A^k \subseteq A^{k-1}$, and $W^k = A^k$. This implies:

$$x_i^{k+1} = x_i^k = 0 \quad \text{for } i \in W^k. \tag{4.10}$$

Therefore, since (4.7) holds for $i \in R^k, S^k = \emptyset$, and $(x^{k+1} - x^k)_i = 0$ for $i \in W^k$, it follows from (4.4) that only terms in $T^k = A^{k-1} \cap (N^k \cup P^k)$ contribute to the sum; thus

$$(Qx^k + q + s^k\mu)^T(x^{k+1} - x^k) = \sum_{i\in T^k}([Qx^k]_i + q_i + s_i^k\mu)(x^{k+1} - x^k)_i < 0. \tag{4.11}$$

Now suppose that at some iteration of the `repeat` loop in Step 2, $|[Qx^k]_i + q_i| \geq \mu$ for some $i \in V_k$. If $i \in A^{k-1} \cap P^k$, then $[Qx^k]_i + q_i + s_i^k\mu \leq 0$, but since $i \in V^k$, then we must have $x_i^{k+1} < 0$; as a result, that term i in (4.11) is nonnegative. If $i \in N^k \cap A^{k-1}$, a similar argument implies that term i is nonnegative in that case. So by (4.11) there must exist a negative term in the sum. This negative term must correspond to some $i \in T^k \setminus V^k = (A^{k-1} \cap (N^k \cup P^k)) \setminus V^k$, for which $[Qx^k]_i + q_i + s_i^k\mu \neq 0$. Since this index remains in T^k at the next iteration, x^k is still not optimal for (4.9), and thus (4.11) holds. It follows that (4.11) holds at the end of step 2, and that $x^{k+1} \neq x^k$.

□

This result can be used to show that the Corrected BAS algorithm (unlike the original BAS algorithm) generates descent directions at each iteration.

Corollary 4.2 *Under the conditions of Theorem 4.1, if x^k is not a stationary point for problem* (3.8)*, the direction $d^k = x^{k+1} - x^k$, generated by Algorithm 4.1, is a descent direction for the objective function φ, i.e., the directional derivative satisfies $D\varphi(x^k; d^k) < 0$.*

Proof. The directional derivative of φ at x^k in the direction d^k is given by

$$D\varphi(x^k; d^k) = \sum_{i=1}^{n} (Qx^k + q)_i d_i^k + \sum_{i:x_i^k>0} \mu d_i^k - \sum_{i:x_i^k<0} \mu d_i^k + \sum_{i:x_i^k=0} \mu |d_i^k|. \tag{4.12}$$

As the indices $i \in W^k$ have $d_i^k = x_i^{k+1} - x_i^k = 0$, they have zero contribution to the sum (4.12). We analyze the individual contributions from the remaining subsets.

$i \in R^k$: First consider indices i where $x_i^k \neq 0$. By (4.2), if $i \in P^{k-1} \cap P^k$ then $x_i^k > 0$; if $i \in N^{k-1} \cap N^k$ then $x_i^k < 0$. Thus, the contribution to the directional derivative (4.12) is

$$\big((Qx^k + q)_i + \mu \operatorname{sign}(x_i^k)\big)(x^{k+1} - x^k)_i = 0,$$

by (4.3b) and (4.3c). Next consider the indices i where $x_i^k = 0$. If $i \in P^{k-1} \cap P^k$, by the corrective mechanism described in Step 2 of Algorithm 4.1, we have $x_i^{k+1} \geq 0$ and therefore, $d_i^k \geq 0$. By (4.12) the contribution to the directional derivative is given by

$$\big((Qx^k + q)_i + \mu\big)x_i^{k+1} = 0,$$

where the equality follows from $i \in P^{k-1}$ and (4.3*b*) at the previous step. A similar argument holds for $i \in N^{k-1} \cap N^k$. Therefore, the contribution of indices $i \in R^k$ to the sum (4.12) is zero.

$i \in S^k$: If $i \in P^{k-1} \cap A^k$, then, given (4.3b) at the previous iterate, it follows that $(Qx^k + q)_i = -\mu$. Since $i \in A^k$, then $x_i^k < 0$, and the contribution to the directional derivative is given by

$$(Qx^k + q)_i - \mu)(0 - x^k)_i < 0.$$

A similar argument shows that the contribution of indices $i \in N^{k-1} \cap A^k$ is also negative. Therefore, we can conclude that the contribution of any index $i \in S^k$ to the directional derivative is negative.

$i \in T^k$: If $i \in A^{k-1} \cap P^k$, we know that $x_i^k = 0$, and the corrective mechanism described in Step 2 of Algorithm 4.1 yields $x_i^{k+1} \geq 0$. Furthermore, given that the index was assigned to P^k from A^{k-1}, it follows from (4.2) that $(Qx^k + q)_i + \mu \leq 0$, and therefore

$$\big((Qx^k + q)_i + \mu\big)(x^{k+1} - 0)_i \leq 0.$$

The analysis for the case $i \in A^{k-1} \cap N^k$ yields the same result.

Thus, the only nonzero contributions to $D\varphi(x^k; d^k)$ are those found in sets S^k and T^k. If S^k is nonempty, those terms are negative and $D\varphi(x^k; d^k) < 0$. If S^k is empty, then by Theorem 4.1 it follows that (4.6) holds, and since the loop in step 2 ensures $|d_i^k| = s_i^k (x^{k+1} - x^k)_i$, by (4.12), we have:

$$D\varphi(x^k; d^k) = \sum_{i \in T^k} ([Qx^k]_i + q_i + s_i^k \mu)(x^{k+1} - x^k)_i < 0. \tag{4.13}$$

□

4.1 A Noncycling Variant

Algorithm 4.1 is not globally convergent, and in fact may be subject to cycling behavior, although it is extremely rare to see this in practice. In this section, we describe a strategy for making Algorithm 4.1 globally convergent based on the safeguarding procedure proposed in [19], and employed in [20] in the context of regularized linear regression. If the cardinality of indices violating (3.11) fails to decrease for several iterations, a backup procedure is invoked that ensures decrease of this cardinality by restricting the changes in the sets in (3.11) to one variable per iteration. A modification of Algorithm 4.1 that incorporates this safeguarding strategy to provide global convergence is given in Algorithm 4.2.

Algorithm 4.2: A Noncycling Variant of Algorithm 4.1

Initialize x^0, w^0, and the index sets P^{-1}, N^{-1}, A^{-1} as in Algorithm 3.1. Choose an integer t_{max} and set $card \leftarrow n+1$.

for $k = 0, \cdots$, until the stopping conditions (3.11) are satisfied:

1. $W^k = \{i \in A^{k-1} : w_i^k \notin [-\mu, \mu]\} \cup \{i \in P^{k-1} : x_i^k < 0\} \cup \{i \in N^{k-1} : x_i^k > 0\}$.
2. **If** $|W^k| < card$, **then** set $t \leftarrow 0$; $card \leftarrow |W^k|$;

   Else set $t \leftarrow t+1$;

   Endif
3. **If** $t \leq t_{max}$, **then** Perform Step 1 and Step 2 of Algorithm 4.1.

   Else Set $j = \max\{i : i \in W^k\}$

   If $j \in P^{k-1}$, set $P^k = P^{k-1} \setminus \{j\}$; $N^k = N^{k-1}$; $A^k = A^{k-1} \cup \{j\}$.
   If $j \in N^{k-1}$, set $P^k = P^{k-1}$; $N^k = N^{k-1} \setminus \{j\}$; $A^k = A^{k-1} \cup \{j\}$.
   If $j \in A^{k-1}$ and $w^k > \mu$, set $P^k = P^{k-1}$; $N^k = N^{k-1} \cup \{j\}$; $A^k = A^{k-1} \setminus \{j\}$.
   If $j \in A^{k-1}$ and $w^k < -\mu$, set $P^k = P^{k-1} \cup \{j\}$; $N^k = N^{k-1}$; $A^k = A^{k-1} \setminus \{j\}$.
   Compute the iterate (x^{k+1}, w^{k+1}) satisfying (4.3).

   Endif

end(for)

Another approach we could use for ensuring convergence in the general convex case is to incorporate a line search in Algorithm 4.1, which is feasible since by Corollary 4.2 the algorithm produces descent directions. However, such a strategy changes the fundamentally discrete character of the algorithm, in which each iterate is the minimizer of the quadratic defined by the objective in a specified orthant. Therefore, we will defer consideration of

this approach to a future study. We stress that failures of Algorithm 4.1 are so rare (unlike those of the basic BAS method) that backup strategies are likely to be invoked very rarely in practice.

5 An Orthant Based Method with Corrections (OBM-COR)

The correction mechanism, described by the `repeat` loop in Step 2 of Algorithm 4.1, can be applied to methods other than the BAS algorithm. In particular, it could be employed in orthant-based methods, or in the second-order ISTA method described in Section 2. In this section, we study the use of the correction mechanism within an orthant-based method, which shares many similarities with the Corrected BAS strategy.

A distinctive feature of orthant based methods is that they employ the smooth quadratic model (2.14) of the objective ϕ, minimize it over the space of free variables, and ensure the new iterate is constrained to the current orthant face Ω_k. To describe the implementation of the corrective mechanism within an orthant based method, we first express the orthant face Ω_k, in terms of the index sets of Algorithm 4.1. We make the following assignments:

$$\xi_i^k < 0 \to i \in N^k, \quad \xi_i^k > 0 \to i \in P^k, \quad \text{and} \quad \xi_i^k = 0 \to i \in A^k, \tag{5.1}$$

where ξ^k is defined by (2.12).

The first step of the orthant-based method with correction (OBM-COR) is to identify the orthant face Ω_k, using (2.12), and the corresponding index sets P^k, N^k and A^k, through (5.1). The algorithm then performs Step 2 of Algorithm 4.1. (Note that the update (4.3a)-(4.3d) is equivalent to solving (2.14), for a particular setting of (P^k, N^k, A^k)). Upon termination of the `repeat` loop, the resultant iterate, x^{k+1}, is then projected onto the current orthant face, to yield the point

$$\hat{x}_i^{k+1} = \begin{cases} x_i^{k+1} & \text{if } (i \in P^k \wedge x_i^{k+1} > 0) \vee (i \in N^k \wedge x_i^{k+1} < 0) \\ 0 & \text{otherwise.} \end{cases} \tag{5.2}$$

It is interesting to compare the OBM-COR and Corrected BAS methods, as the principal difference is in the projection. If an iterate x_k has been generated by OBM-COR, then the sets defined by (5.1) are almost identical with those defined by Step 1 of Algorithm 4.1, the only difference being at the discontinuities where $x_i^k = 0$ and $|[Qx^k + k]_i| = \mu$. However, at the termination of the `repeat` loop in Step 2, OBM-COR will project onto the chosen orthant, while Corrected BAS will take the full step.

To illustrate the effects of this projection, consider the case $x_i^{k+1} < 0$ for $i \in P^k$, which by (5.2) yields $\hat{x}_i^{k+1} = 0$. As this index can be placed in N^{k+1} at the next iteration, the OBM-COR method allows faster changes to the index sets in comparison to the BAS method, which would instead place $i \in A^{k+1}$ at the beginning of the next iteration and hold the variable at zero.

6 Numerical Experiments

We coded the semi-smooth Newton method of Section 2 in MATLAB, with the values of τ and δ being supplied as input parameters. While numerous methods could be tested using this semi-smooth Newton framework, our focus is on evaluating the effectiveness of the corrective mechanism proposed in this paper, in both block active-set and orthant-based methods. The investigation of alternate methods generated by other choices of τ and δ, such as the second-order iterative soft-thresholding (ISTA) method, is postponed to a future paper.

We first compare the new block active-set method (Algorithm 4.1), against the original block active-set method (Algorithm 3.1) on quadratic ℓ_1-regularized problems of the form (3.8). We generated 1000 test problems where the Hessian matrix Q was created using the MATLAB sparse matrix generator, `sprandsym`, with a density factor of 25%. One half of the test problems had a moderate condition number of Q (in the order of 10^4), and the other half had a condition number of magnitude 10^7. The vector q was randomly generated such that its entries were of a similar magnitude to the eigenvalues of Q, and approximately half of its entries were positive, and half negative. The regularization parameter μ was randomly generated in the interval $[2.5, n/3]$.

In Table 6.1, we compare the robustness of the two methods on the set of test problems described above. We also report the average number of correction steps, i.e., the number of executions of the `repeat` loop in Algorithm 4.1, per outer iteration. The effectiveness of the correction mechanism in improving the robustness of the iteration is evident. We note that when Algorithm 3.1 failed, it exhibited cycling, as discussed in [4]. We also ran Algorithm 4.2 (with $t_{max} = 10$) on these test problems, and the backup strategy was never invoked. Therefore, the performance of Algorithms 4.1 and 4.2 was identical.

Table 6.1: Comparison of the BAS and Corrected BAS methods over 1000 test problems

Conditioning of Matrix Q	**Algorithm 3.1**	**Algorithm 4.1**	
	Number of Failures	Number of Failures	Average corr/iter
Moderate	10 (out of 500)	0	0.92283
High	120 (out of 500)	0	1.40629
Total	130 Failures	0 Failures	1.1646

Figure 6.1 gives a comparison of CPU times of the two methods using the logarithmic performance profiles described in [9]. That figure also reports the performance of the orthant based method with correction (OBM-COR) discussed in section 5. We observe that the correction mechanism is effective in both methods, and that the OBM-COR method is somewhat more efficient in this tests, possibly due to the use of the projection at the end of the step. While both the Corrected BAS and OBM-COR methods are more robust than the BAS algorithm, this stability is obtained at the expense of increased computational cost. However, savings in computation can be achieved through the selective use of corrections, inexact steps [15, 17], and by updating matrix factorizations. Such enhancements require careful consideration and a sophisticated implementation, and will be the subject of a future

study.

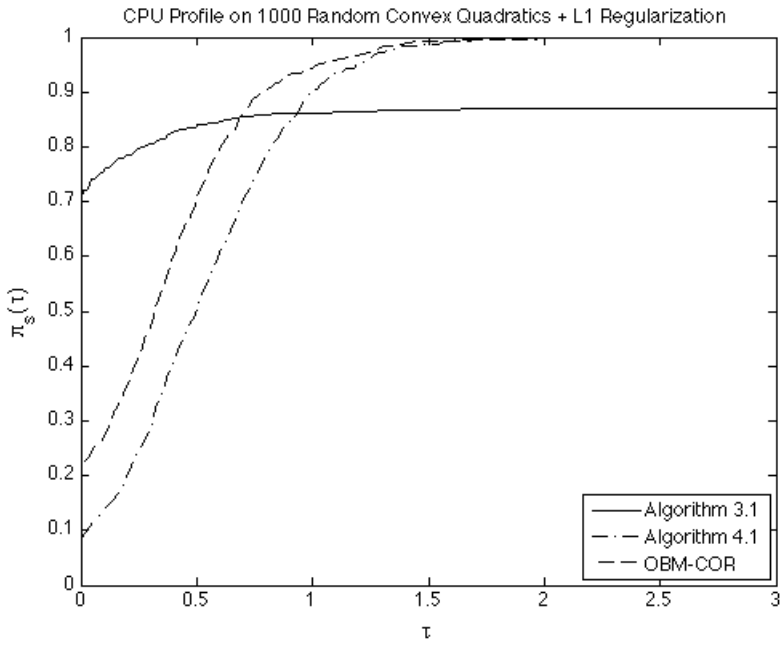


Figure 6.1: CPU performance profiles for Algorithm 3.1, Algorithm 4.1 and the OBM-COR method

6.1 General Convex Problems with ℓ_1 Regularization

We now study the use of a block active-set method in the solution of the general convex problem (1.1). We follow a Newton-like approach, similar to successive quadratic programming, in which at each iteration, a step is computed as the minimizer of the model

$$m^k(d) = f(x^k) + \nabla f(x^k)^T d + \tfrac{1}{2} d^T B^k d + \mu \|x^k + d\|_1, \tag{6.1}$$

where B^k is either $\nabla^2 f(x^k)$ or an approximation to it. We minimize problem (6.1) using the Corrected BAS method, and denote its solution by d^k. The new iterate is given by $x^{k+1} = x^k + \alpha^k d^k$, where $\alpha^k \in \{1, \frac{1}{2}, \frac{1}{4}, \cdots\}$ is the largest value such that the following Armijo condition is met:

$$\phi(x^k + \alpha^k d^k) \leq \phi(x^k) + \gamma \alpha^k \nabla f(x^k)^T d^k + \gamma \mu (\|x^k + \alpha^k d^k\|_1 - \|x^k\|_1), \tag{6.2}$$

where $\gamma = 0.0001$. We refer to this approach as the Newton-CBAS method.

It is easy to see that this method generates a descent direction for the true objective (1.1), since it is evident that the directional derivatives satisfy $D\phi(x^k; d^k) = Dm^k(0; d^k)$, and the convexity of the model m^k and the minimization property yield $Dm^k(0; d^k) \leq m^k(d^k) - m^k(0) < 0$, see also [21].

We compare the performance of the Newton-CBAS method for problem (1.1), with that of the well known orthant-based OWL-QN method [2]. Since the latter uses a limited memory BFGS approximation, we defined B^k in (6.1) in the same manner, to make the comparison as fair as possible. The test problem is generated using the StatLog Landstat Satellite dataset from the UCI machine learning repository [13]. We formulate this problem as a multi-class logistic function, where $f(x)$ is a negative log-likelihood function of the form

$$f(x) = -\frac{1}{N}\sum_{h=1}^{N} \log \frac{\exp(x_{y_h}^T z_h)}{\sum_{i\in\mathcal{C}} \exp(x_i^T z_h)}. \tag{6.3}$$

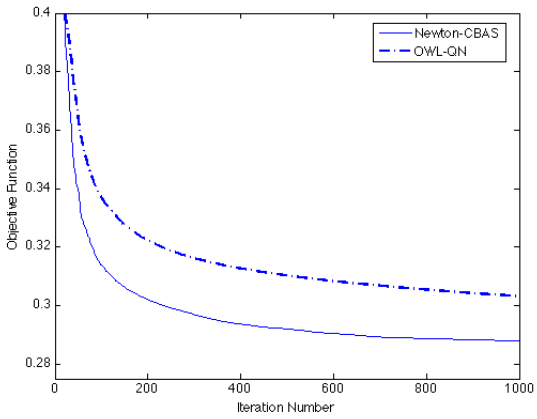
In this expression, N is the number of training points, $\mathcal{C}$ denotes the set of all class labels; $y_h \in \mathcal{C}$ is the class label associated with data point h; z_h is the feature vector for data point h. In solving the sparse classification problem for the StatLog Landstat Satellite dataset, we expand the feature space for every data point by taking the products of all possible pairs of features, thus increasing the feature set from 36 to 1296 features. As the number of classes $|C| = 6$, the dimension of the problem is $n = \textit{number of classes} \times \textit{number of features} = 7776$. We utilize a memory of 5 for the limited memory BFGS approximation, and set the regularization parameter to be $\mu = 1/n$.

In Figure 6.2, we plot the objective function value ϕ and the optimality error $\|F\|_1$, as a function of the iteration number, where F is defined in (2.2). It is clear that the Newton-CBAS method outperforms OWL-QN in both of measures. We note that in this problem the cost of the iterations in the two methods is dominated by the function evaluations.

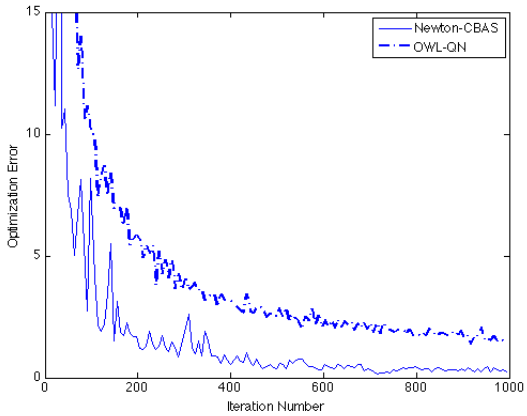
7 Final Remarks

We presented a semi-smooth Newton framework for generating a variety of two-phase methods for solving the ℓ_1 regularized convex problem (1.1). While we have explored only a few of the methods that can be generated by this framework, some of the remaining alternatives, such as the second-order ISTA algorithm, deserve further consideration. In addition to its generality and flexibility, the semi-smooth Newton framework allows us to develop variants in which the subspace phase is solved inexactly [15], although the exploration of that topic will be the subject of a future study.

We proposed a novel corrective mechanism for the block active set (BAS) method that significantly improves the robustness of the iteration. Although the proposed algorithm can cycle, failures are so rare that it is easy to incorporate safeguarding mechanisms that do not interfere with the efficiency of the approach. We described one such procedure that ensures the (non-monotone) decrease in the cardinality of the infeasibility set. A different safeguarding procedure that builds on the descent property given in Corollary 4.2 will be developed in a future study, in conjunction with the inexact subspace solve mentioned above, with the aim of reducing the computational costs associated with the corrective mechanism proposed in this paper. We have shown that this mechanism is also effective in an orthant-based method, and it would be of interest to explore it in the context of a second-order ISTA method.



(a) Objective function ϕ



(b) Optimality error $\|F\|_1$

Figure 6.2: *StatLog Landstat Satellite Dataset*: Newton-CBAS vs. OWL-QN

Acknowledgments The authors would like to thank Michael Hintermüller and Daniel Robinson for several useful conversations on the subject of this paper.

8 Appendix: Derivation of the Block Active-Set (BAS) Algorithm for Problem (3.8)

By introducing auxiliary variables u, v, such that

$$x = u - v \quad \text{and} \quad u, v \geq 0, \tag{8.1}$$

we can reformulate problem (3.8) as a bound constrained problem of the form (3.1):

$$\begin{aligned} \min_{u,v \in \mathbb{R}^n} \ & \tfrac{1}{2}(u-v)^T Q(u-v) + q^T(u-v) + \mu(u+v)^T e \\ \text{s.t.} \quad & u, v \geq 0, \end{aligned} \tag{8.2}$$

where $e \in \mathbb{R}^n$ represents a vector with entries of all ones. In this smooth reformulation, the variable u is used to represent the positive components of x, and v the negative components. The KKT conditions for (8.2) are

$$Q(u-v) + q + \mu e - y = 0, \tag{8.3}$$

$$-(Q(u-v) + q) + \mu e - z = 0, \tag{8.4}$$

$$y^T u = 0, \qquad z^T v = 0, \tag{8.5}$$

$$u, v \geq 0, \qquad y, z \geq 0, \tag{8.6}$$

where y and z are the Lagrange multipliers for the nonnegative constraints of u and v respectively.

Given a primal-dual iterate (u^k, v^k, y^k, z^k), the BAS algorithm (3.5)-(3.7) begins with the specification of the index sets, which for the case of problem (8.2) take the form:

A_k^u : is the set of indices of u^k that will be set to zero,

$\mathcal{I}_k^u$: is the set of indices of u^k that will be allowed to move; $\mathcal{I}_k^u = \{1, \cdots, n\} \setminus A_k^u$,

A_k^v : is the set of indices of v^k that will be set to zero,

$\mathcal{I}_k^v$: is the set of indices of v^k that will be allowed to move; $\mathcal{I}_k^v = \{1, \cdots, n\} \setminus A_k^v$.

The method is specified in Algorithm 8.1, where we have assumed the initial variables u^0, v^0 satisfy the complementarity conditions

$$0 \leq u_i^0 \perp v_i^0 \geq 0 \quad \forall\ i. \tag{8.7}$$

Algorithm 8.1: BAS Algorithm for the Bound Constrained Problem (8.2)

Initialize: Choose an initial iterate u^0, v^0 that satisfies (8.7).
Set y^0 and z^0 such that

$$\begin{aligned}
&\text{If } u_i^0 > 0: && \text{set: } y_i^0 = 0, z_i^0 > 0;\\
&\text{If } v_i^0 > 0: && \text{set: } y_i^0 > 0, z_i^0 = 0;\\
&\text{If } u_i^0 = v_i^0 = 0: && \text{set } y_i^0, z_i^0 \text{ such that (8.3) and (8.4) are satisfied.}
\end{aligned}$$

for $k = 0, \cdots$ until the optimality conditions (8.3)-(8.6) are met:

1. Compute the index sets:

$$A_k^u = \{i : u_i^k < 0\} \cup \{i : u_i^k = 0 \text{ and } y_i^k > 0\} \qquad \mathcal{I}_k^u = [A_k^u]^c, \tag{8.8}$$
$$A_k^v = \{i : v_i^k < 0\} \cup \{i : v_i^k = 0 \text{ and } z_i^k > 0\} \qquad \mathcal{I}_k^v = [A_k^v]^c, \tag{8.9}$$

   where $[A]^c$ indicates the complement of set A.

2. Compute the iterate $(u^{k+1}, v^{k+1}, y^{k+1}, z^{k+1})$ using the following procedure:

   2.1 Set the variables in the active set, and the multipliers of variables in the inactive set, to zero:

$$u_i^{k+1} = 0 \qquad \forall\ i \in A_k^u, \tag{8.10}$$
$$v_i^{k+1} = 0 \qquad \forall\ i \in A_k^v, \tag{8.11}$$
$$y_i^{k+1} = 0 \qquad \forall\ i \in \mathcal{I}_k^u, \tag{8.12}$$
$$z_i^{k+1} = 0 \qquad \forall\ i \in \mathcal{I}_k^v. \tag{8.13}$$

   2.2 Compute the remaining variables so as to satisfy (8.3) and (8.4), i.e.,

$$Q(u^{k+1} - v^{k+1}) + q + \mu e - y^{k+1} = 0, \tag{8.14}$$
$$-(Q(u^{k+1} - v^{k+1}) + q) + \mu e - z^{k+1} = 0. \tag{8.15}$$

End(for)

Algorithm 8.1 preserves complementarity in u and v. To see this, note that equations (8.10)-(8.13), together with the initialization, yield primal-dual complementarity at every iteration k, i.e.,

$$u_i^k y_i^k = 0 \quad \text{and} \quad v_i^k z_i^k = 0, \qquad \forall\ i. \tag{8.16}$$

By adding the equations (8.14) and (8.15), we see that $y^{k+1} + z^{k+1} = 2\mu e$. This implies that for each index i, we will have either $y_i^{k+1} > 0$ or $z_i^{k+1} > 0$. Thus, by (8.8), (8.9) and (8.16), either $i \in A_k^u$ or $i \in A_k^v$. As a result, for each i, either $i \notin \mathcal{I}_k^u$ or $i \notin \mathcal{I}_k^v$, since $\mathcal{I}_k^u = [A_k^u]^c$,

and hence

$$\mathcal{I}_k^u \cap \mathcal{I}_k^v = \emptyset \quad \forall\ k \geq 0. \tag{8.17}$$

Therefore, at every iteration k either u_i^k or v_i^k are zero, for all i.

As a result, it suffices to keep track of the positive and negative components of a single vector x, thus reducing the number of variables by half. Given (8.1), we can similarly define a single multiplier vector,

$$w^k := Q(u^k - v^k) + q = Qx^k + q. \tag{8.18}$$

so that by (8.14) and (8.15) we have

$$y_i^{k+1} = w_i^{k+1} + \mu, \qquad z_i^{k+1} = -w_i^{k+1} + \mu. \tag{8.19}$$

Next, we define index sets corresponding to the single variable x. To do so, we first make some observations about the transitions of the indices of the vectors u, v in Algorithm 8.1. Let us define

$$\begin{aligned} P^{k-1} &= \mathcal{I}_{k-1}^u : \text{ the set of indices of } x^k \text{ we predict are positive,} \\ N^{k-1} &= \mathcal{I}_{k-1}^v : \text{ the set of indices of } x^k \text{ we predict are negative,} \\ A^{k-1} &= A_{k-1}^u \cup A_{k-1}^v : \text{ the set of indices of } x^k \text{ that are set to zero.} \end{aligned} \tag{8.20}$$

We analyze the following cases.

1. Let us consider an index $i \in A^{k-1}$. By (8.10) and (8.11), it follows that $u_i^k = v_i^k = 0$. Therefore, one of the following conditions holds:

   If $y_i^k > 0 \wedge z_i^k > 0 :$ then, by (8.8), (8.9), $i \in A^k$.
   If $y_i^k = w_i^k + \mu \leq 0 :$ then, by (8.19), $z_i^k > 0$. From (8.8) and (8.9), we have $i \in P^k$.
   If $z_i^k = -w_i^k + \mu \leq 0 :$ then, by similar arguments, $i \in N^k$.

2. Let us consider an index $i \in P^{k-1}$. From (8.17), we have $i \in A_{k-1}^v$. By (8.11), (8.12) and (8.19), it follows that $v_i^k = 0$, $y_i^k = 0$ and $z_i^k > 0$ respectively. Therefore, from (8.9), $i \in A_k^v$, and by (8.8), it follows that:

$$i \in \begin{cases} A^k & \text{if } u_i^k < 0 \\ P^k & \text{if } u_i^k \geq 0. \end{cases}$$

3. Let us consider $i \in N^{k-1}$. From (8.17), we have $i \in A_{k-1}^u$. By (8.10), (8.13) and (8.19), it follows that $u_i^k = 0$, $z_i^k = 0$ and $y_i^k > 0$ respectively. Therefore, $i \in A_k^u$, and we have that

$$i \in \begin{cases} A^k & \text{if } v_i^k < 0 \\ N^k & \text{if } v_i^k \geq 0. \end{cases}$$

This analysis demonstrates that, for the case $i \in P^{k-1} \cup N^{k-1}$, the multipliers $y_i^k, z_i^k \geq 0$ are always feasible, and that indices cannot transition from being inactive in v to being inactive in u, or vice versa. As a result, transitions between the index sets P and N over consecutive iterations are prohibited in the BAS method.

Based on this fact, and the observations made in the three cases above, we can redefine the index sets in (8.20) in terms of the x variable,

$$\begin{aligned} P^k &= \{i \in P^{k-1} : x_i^k \geq 0\} \cup \{i \in A^{k-1} : w_i^k \leq -\mu\}, \\ N^k &= \{i \in N^{k-1} : x_i^k \leq 0\} \cup \{i \in A^{k-1} : w_i^k \geq \mu\}, \\ A^k &= \{i \in A^{k-1} : w_i^k \in (-\mu, \mu)\} \cup \{i \in P^{k-1} : x_i^k < 0\} \cup \{i \in N^{k-1} : x_i^k > 0\}. \end{aligned} \tag{8.21}$$

To express (8.10)-(8.15) in terms of the x variable alone, we first note that (8.10) and (8.11) can be replaced by

$$x_i^{k+1} = 0 \qquad \forall\ i \in A^k.$$

To compute the remaining components of x^{k+1}, that is $i \in P^k \cup N^k$, we substitute (8.12) and (8.13) into the equations (8.14) and (8.15) to obtain

$$\begin{aligned} [Qx_{k+1} + q + \mu e]_i &= 0 \qquad \forall\ i \in P^k, \\ [Qx_{k+1} + q - \mu e]_i &= 0 \qquad \forall\ i \in N^k. \end{aligned}$$

In summary, the description of the BAS algorithm applied to problem (3.8) is given in Algorithm 3.1.

References

[1] M. Aganagić. Newton's method for linear complementarity problems. *Mathematical Programming*, 28(3):349–362, 1984.

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

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] I. Ben Gharbia and J.C. Gilbert. Nonconvergence of the plain newton-min algorithm for linear complementarity problems with a P-matrix. *Mathematical Programming*, pages 1–16, 2010.

[5] M. Bergounioux, K. Ito, and K. Kunisch. Primal-dual strategy for constrained optimal control problems. *SIAM Journal on Control and Optimization*, 37(4):1176–1194, 1999.

[6] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221–246, 1982.

[7] R. Byrd, G.M. Chin, J. Nocedal, and Y. Wu. Sample size selection in optimization methods for machine learning. Technical report, Optimization Center Report 2011/3, Northwestern University, 2011.

[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(11):1413–1457, 2004.

[9] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A*, 91:201–213, 2002.

[10] D.L. Donoho. De-noising by soft-thresholding. *Information Theory, IEEE Transactions on*, 41(3):613–627, 1995.

[11] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

[12] F. Facchinei and J.S. Pang. *Finite-dimensional variational inequalities and complementarity problems*, volume 1. Springer Verlag, 2003.

[13] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[14] S. Hansen and J. Nocedal. Second-order methods for L_1 regularized problems in machine learning. Technical report, Optimization Center Report 2011/3, Northwestern University, 2011. To appear in ICAASP 2012.

[15] M. Hintermuller and M. Hinze. A SQP-semismooth Newton-type algorithm applied to control of the instationary Navier-Stokes system subject to control constraints. *SIAM Journal on Optimization*, 16(4):1177–1200, 2006.

[16] M. Hintermüller, K. Ito, and K. Kunisch. The primal-dual active set strategy as a semismooth Newton method. *SIAM J. Optim.*, 13(3):865–888, 2003.

[17] M. Hintermüller and G. Stadler. An infeasible primal-dual algorithm for total bounded variation-based inf-convolution-type image restoration. *SIAM Journal on Scientific Computing*, 28(1):1–23, 2007.

[18] 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.

[19] J. J. Júdice and F. M. Pires. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers and Operations Research*, 21(5):587–596, 1994.

[20] J. Kim and H. Park. Fast active-set-type algorithms for l_1-regularized linear regression. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

[21] Z. Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible direction methods: a general approach. *Annals of Operations Research*, 46:157–178, 1993.

[22] R. Mifflin. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

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

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[25] L. Qi. Convergence analysis of some algorithms for solving nonsmooth equations. *Mathematics of Operations Research*, 18(1):227 – 244, 1993.

[26] L. Qi and J. Sun. A nonsmooth version of newton's method. *Mathematical Programming*, 58:353–367, 1993. 10.1007/BF01581275.

[27] A. Ruszczynski. *Nonlinear optimization*. Princeton university press, 2011.

[28] S. Sra, S. Nowozin, and S.J. Wright. *Optimization for Machine Learning*. Mit Pr, 2011.

[29] M. Ulbrich. *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*, volume 11. MOS-SIAM Series on Optimization, 2011.

[30] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization and continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857, 2010.

[31] S.J. Wright, R.D. Nowak, and M.A.T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[32] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *The Journal of Machine Learning Research*, 11:2543–2596, 2010.