FaRSA for ℓ_1-regularized convex optimization: local convergence and numerical experience

Tianyi Chen, Frank E. Curtis & Daniel P. Robinson

To cite this article: Tianyi Chen, Frank E. Curtis & Daniel P. Robinson (2018) FaRSA for ℓ_1-regularized convex optimization: local convergence and numerical experience, Optimization Methods and Software, 33:2, 396-415, DOI: 10.1080/10556788.2017.1415336

To link to this article: https://doi.org/10.1080/10556788.2017.1415336

Published online: 19 Jan 2018.

Optimization Methods & Software, 2018
Vol. 33, No. 2, 396–415, https://doi.org/10.1080/10556788.2017.1415336

FaRSA for ℓ_1-regularized convex optimization: local convergence and numerical experience

Tianyi Chen[a], Frank E. Curtis[b] and Daniel P. Robinson[a]*

[a]*Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, MD, USA*
[b]*Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA*

(*Received 9 May 2017; accepted 5 December 2017*)

FaRSA is a new method for minimizing the sum of a differentiable convex function and an ℓ_1-norm regularizer. The main features of the method include: (i) an evolving set of indices corresponding to variables that are predicted to be nonzero at a solution; (ii) subproblems that only need to be solved in a reduced space to lessen per-iteration computational costs; (iii) conditions that determine how accurately each subproblem must be solved, which allow conjugate gradient or coordinate descent techniques to be employed; (iv) a computationally practical condition that dictates when the subspace explored by the current subproblem should be updated; and (v) a reduced proximal gradient step that ensures a sufficient decrease in the objective function when it is decided that the index set that holds the nonzero variables should be expanded. We proved global convergence of the method and demonstrated its performance on a set of model prediction problems with a Matlab implementation. Here, we introduce an enhanced subproblem termination condition that allows us to prove that the iterates converge locally at a superlinear rate. Moreover, we present the details of our publicly available C implementation along with extensive numerical comparisons to other state-of-the-art solvers.

Keywords: nonlinear optimization; convex optimization; sparse optimization; active-set methods; reduced-space methods; subspace minimization; model prediction

AMS Subject Classification: 49J52; 62–07; 65K05; 90C25; 90C30

1. Introduction

In [6], we proposed a *fast reduced-space algorithm* (FaRSA) for solving ℓ_1-norm regularized convex optimization problems of the form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ F(x), \quad \text{where } F(x) := f(x) + \lambda\|x\|_1, \tag{1}$$

$f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable convex function, and $\lambda > 0$ is a weighting parameter. Problems of the form (1) are used extensively in statistics, signal processing, and machine learning applications. A popular setting is binary classification using logistic regression, although such problems also arise when performing multi-class logistic regression and profit regression. Instances of (1) also surface when using LASSO or elastic-net formulations to perform data analysis and discovery.

*Corresponding author. Email: daniel.p.robinson@gmail.com

The main contributions in [6] were that we introduced FaRSA and established its global convergence properties. We also provided preliminary numerical results that indicated its potential to be a state-of-the-art algorithm. In this paper, we analyse the local convergence properties of FaRSA, and provide detailed and extensive numerical experiments showing that our FaRSA implementation is competitive with, and often outperforms, state-of-the-art software for solving problems of the form (1).

1.1 *Literature review and contributions*

There exist many first-order methods, i.e. ones that only use first-order derivatives of f, for solving problem (1); popular examples include ISTA [8], FISTA [2], and SpaRSA [15]. These methods have proved quite useful in practice because of their simplicity and good performance when solving problems arising in a variety of interesting applications. Nonetheless, first-order methods are somewhat limited in terms of their reliability when it comes to solving ill-conditioned problems. Such poor conditioning can often be overcome by employing second-order derivative information.

The method considered here is a second-order method, i.e. it uses second-order derivatives of the objective function. Second-order methods can roughly be split into the two classes of proximal-Newton [4,12,14,16] and orthant-based [1,3,13] methods. Proximal-Newton methods solve problem (1) by minimizing a sequence of subproblems formed as the sum of a quadratic approximation to f and the nonsmooth ℓ_1-norm regularizer. For example, the state-of-the-art software LIBLINEAR, which implements the solver newGLMNET [16], uses a coordinate descent (CD) algorithm to approximately minimize each piecewise quadratic subproblem. On the other hand, orthant-based methods minimize smooth quadratic approximations to (1) over a sequence of orthants in $\mathbb{R}^n$. For example, OBA [13] computes a sequence of orthant predictions and subspace minimization steps using the conjugate gradient (CG) algorithm. Yet another class of methods for solving (1) involves not approaching the problem directly, but rather introducing and solving a smooth bound-constrained reformulation from which a solution of (1) can be recovered. The bound-constrained reformulation can be solved, e.g. using active-set techniques such as in the ASA-CG software [11].

The following summarizes the key features of FaRSA.

(i) FaRSA is an active-set line search method that utilizes reduced-space subproblems for which only approximate solutions are needed.
(ii) Convergence is achieved by combining a new projected backtracking line search procedure and a mechanism for determining when the predicted set of variables that are zero at a solution should be updated.
(iii) A general set of conditions is used to determine how accurately each subproblem should be solved. The conditions allow for various subproblem solvers to be used, such as CG (as in OBA and ASA-CG) or CD (as in LIBLINEAR).

The key contributions of this paper are as follows. First, using enhanced subproblem termination conditions, we show that the iterates computed by FaRSA converge locally at a superlinear rate. As the analysis will reveal, this is a nontrivial task due to how the reduced space subproblems are constructed and the step computation is performed. Second, we present the details of our publicly available C implementation of FaRSA along with the results of extensive numerical comparisons to the state-of-the-art solvers LIBLINEAR and ASA-CG. These results show that FaRSA has superior performance on most of our test problems, which are taken from the LIBSVM repository [5].

1.2 *Notation and assumptions*

Let $\mathcal{I} \subseteq \{1, 2, \ldots, n\}$ denote an index set of variables. For any $v \in \mathbb{R}^n$, we let $[v]_{\mathcal{I}}$ denote the subvector of v consisting of elements of v with indices in $\mathcal{I}$. Similarly, for any symmetric matrix $M \in \mathbb{R}^{n\times n}$, we let $[M]_{\mathcal{I}\mathcal{I}}$ denote the submatrix of M consisting of the rows and columns of M that correspond to the index set $\mathcal{I}$. If, in addition, the index set $\mathcal{I}$ satisfies $[x]_i \neq 0$ for all $i \in \mathcal{I}$, then we let $\nabla_{\mathcal{I}} F(x)$ and $\nabla^2_{\mathcal{I}\mathcal{I}} F(x)$ denote the vector of first derivatives and matrix of second derivatives of F at x, respectively, corresponding to the elements in $\mathcal{I}$. Similarly, we refer to the ith component of the gradient of f at x as $\nabla_i f(x)$. For any vector v, we let $\text{sgn}(v)$ denote the vector of the same length as v whose ith component is 0 when $[v]_i = 0$, is 1 when $[v]_i > 0$, and is -1 when $[v]_i < 0$. For any vector v, we let $\|v\|_1$ and $\|v\|$ denote its ℓ_1-norm and ℓ_2-norm, respectively. For any matrix M, we let $\|M\|$ denote its vector-induced ℓ_2-norm.

The following assumption, used in [6], ensures global convergence for FaRSA.

Assumption 1.1 *The function $f : \mathbb{R}^n \to \mathbb{R}$ is convex, twice continuously differentiable, and bounded below. The gradient function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is Lipschitz continuous on the level set $\mathcal{L} := \{x \in \mathbb{R}^n : F(x) \leq F(x_0)\}$ with Lipschitz constant L_g. The Hessian function $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is uniformly positive definite and bounded on $\mathcal{L}$, i.e. there exist positive constants $\theta_{\min}$ and $\theta_{\max}$ such that*

$$\theta_{\min}\|v\|^2 \leq v^{\mathrm{T}}\nabla^2 f(x) v \leq \theta_{\max}\|v\|^2 \quad \textit{for all } (x, v) \in \mathcal{L} \times \mathbb{R}^n.$$

As a consequence of Assumption 1.1, there is a unique solution to problem (1) that we denote by $x_* \in \mathbb{R}^n$. Our local convergence analysis requires the following additional assumption on the Hessian function in a neighbourhood about x_*.

Assumption 1.2 *The Hessian function $\nabla^2 f : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ is Lipschitz continuous in a neighbourhood of x_* with Lipschitz constant L_H.*

Being a reduced-space method, FaRSA relies on determining the zero and nonzero components of the unique solution x_*. To this end, FaRSA uses the index sets

$$\mathcal{I}^0(x) := \{i \in \mathcal{I} : [x]_i = 0\}, \quad \mathcal{I}^+(x) := \{i \in \mathcal{I} : [x]_i > 0\}, \quad \text{and} \quad \mathcal{I}^-(x) := \{i \in \mathcal{I} : [x]_i < 0\}.$$

We call $\mathcal{I}^0(x)$ the set of *zero variables*, $\mathcal{I}^+(x)$ the set of *positive variables*, $\mathcal{I}^-(x)$ the set of *negative variables*, and $\mathcal{I}^\pm(x) := \mathcal{I}^-(x) \cup \mathcal{I}^+(x)$ the set of *nonzero variables* at x.

It follows from the optimality conditions for (1) that $|\nabla_i f(x_*)| \leq \lambda$ for all $i \in \mathcal{I}^0(x_*)$. Our local analysis needs x_* to satisfy the following stronger condition.

Assumption 1.3 *At the solution x_*, one finds $|\nabla_i f(x_*)| < \lambda$ for all $i \in \mathcal{I}^0(x_*)$.*

Assumption 1.3 is a strict complementarity assumption. In particular, if one uses the continuous reformulation of problem (1) that minimizes $f(x) + \lambda e^T(u + v)$ over (x, u, v) subject to the constraints $x = u - v$ and $(u, v) \geq 0$, then Assumption 1.3 is equivalent to assuming that the Lagrange multipliers (z_u, z_v) associated with $(u, v) \geq 0$, which must be nonnegative, satisfy $([z_v]_i, [z_u]_i) > 0$ for all $i \in \mathcal{I}^0(x_*)$.

2. Review of FaRSA

In this section, we review FaRSA. Although we give a complete statement of the algorithm and briefly describe the important steps of the method, we direct the reader to [6] for additional discussion that motivates each component of the entire algorithm. For our purposes here, we focus on those aspects that are more relevant for this manuscript.

The sets $\mathcal{I}^0$, $\mathcal{I}^+$, and $\mathcal{I}^-$ introduced in Section 1.2 are used to define the following functions that correspond to the zero and nonzero variables at a given $x \in \mathbb{R}^n$:

$$[\beta(x)]_i := \begin{cases} \nabla_i f(x) + \lambda & \text{if } i \in \mathcal{I}^0(x) \text{ and } \nabla_i f(x) + \lambda < 0, \\ \nabla_i f(x) - \lambda & \text{if } i \in \mathcal{I}^0(x) \text{ and } \nabla_i f(x) - \lambda > 0, \\ 0 & \text{otherwise,} \end{cases}$$

whose nonzero elements can only arise from zero variables at x, and

$$[\phi(x)]_i := \begin{cases} 0 & \text{if } i \in \mathcal{I}^0(x), \\ \min\{\nabla_i f(x) + \lambda, \max\{[x]_i, \nabla_i f(x) - \lambda\}\} & \text{if } i \in \mathcal{I}^+(x), \\ \max\{\nabla_i f(x) - \lambda, \min\{[x]_i, \nabla_i f(x) + \lambda\}\} & \text{if } i \in \mathcal{I}^-(x), \end{cases}$$

whose nonzero elements can only arise from nonzero variables at x. If x were a solution to problem (1), then for any $i \in \mathcal{I}^0(x)$ it holds from the optimality conditions for problem (1) that $|\nabla_i f(x)| \le \lambda$. Therefore, the size of $[\beta(x)]_i$ indicates how far variable i—currently with value *zero*—is from being optimal. That is, the norm of the vector $\beta(x)$ is a measure of optimality for the zero variables, i.e. for those in $\mathcal{I}^0(x)$. On the other hand, for any $i \in \mathcal{I}^+(x) \cup \mathcal{I}^-(x)$ it holds from the optimality conditions that $\nabla_i f(x) + \text{sgn}([x]_i)\lambda = 0$. Therefore, the size of $[\phi(x)]_i$ indicates how far that *nonzero* variable is from being optimal. Thus, the norm of the vector $\phi(x)$ is a measure of optimality for the nonzero variables, i.e. for those in $\mathcal{I}^+(x) \cup \mathcal{I}^-(x)$. For purposes of motivation, we include the following result from [6], which shows that β and ϕ together represent a valid optimality measure.

LEMMA 2.1 *Let $\mathcal{S}$ be an infinite set of positive integers such that $\{x_k\}_{k\in\mathcal{S}} \to \bar{x}$. Then, $\bar{x}$ is a solution to* (1) *if and only if $\{\beta(x_k)\}_{k\in\mathcal{S}} \to 0$ and $\{\phi(x_k)\}_{k\in\mathcal{S}} \to 0$. Consequently, $\bar{x}$ is a solution to* (1) *if and only if $\|\beta(\bar{x})\| = \|\phi(\bar{x})\| = 0$.*

FaRSA is stated as Algorithm 1. It is written the same way as in [6], except for an additional condition in line 11. To optimize the reduced-space subproblem defined through choosing the index set $\mathcal{I}_k$ in lines 8 and 15, the algorithm makes use of a quadratic model of the objective function (see line 11) given by

$$m_k(d) := g_k^T d + \tfrac{1}{2} d^T H_k d,$$

where $g_k = \nabla_{\mathcal{I}_k} F(x_k)$ and $H_k = \nabla^2_{\mathcal{I}_k\mathcal{I}_k} F(x_k)$. (We remark that the gradient and Hessian of F at x_k with respect to elements in $\mathcal{I}_k$ is well defined since $\mathcal{I}_k$ is a subset of indices for which $[\phi(x_k)]_i \neq 0$, which is in turn a subset of indices corresponding to nonzero elements of x_k.) FaRSA also makes use of two line search subroutines, stated as Algorithms 2 and 3, the former of which employs the following projection operator dependent on x_k:

$$[\text{Proj}(y\,; x_k)]_i := \begin{cases} \max\{0, [y]_i\} & \text{if } \mathcal{I}^+(x_k), \\ \min\{0, [y]_i\} & \text{if } \mathcal{I}^-(x_k), \\ 0 & \text{if } \mathcal{I}^0(x_k). \end{cases}$$

FaRSA computes a sequence of iterates $\{x_k\}$. During each iteration, the index sets $\mathcal{I}^0(x_k)$, $\mathcal{I}^+(x_k)$, and $\mathcal{I}^-(x_k)$ are identified and then used to define $\beta(x_k)$ and $\phi(x_k)$. From line 4 of Algorithm 1, we can see that when both $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$ are less than a prescribed tolerance

Algorithm 1 FaRSA for solving problem (1).

```
1:  Input: $x_0$
2:  Constants: $\{\eta_\phi, \eta_\beta\} \subset (0,1]$, $\xi \in (0,1)$, $\eta \in (0, \frac{1}{2})$, $\{\gamma, \epsilon\} \subset (0, \infty)$, and $r \in [1,2]$.
3:  for $k = 0, 1, 2, \ldots$ do
4:      if $\max\{\|\beta(x_k)\|, \|\phi(x_k)\|\} \le \epsilon$ then
5:          Return the (approximate) solution $x_k$ of problem (1).
6:      end if
7:      if $\|\beta(x_k)\| \le \gamma \|\phi(x_k)\|$ then                    $\triangleright\ k \in \mathcal{S}_\phi$
8:          Choose any $\mathcal{I}_k \subseteq \{i : [\phi(x_k)]_i \neq 0\}$ such that $\|[\phi(x_k)]_{\mathcal{I}_k}\| \ge \eta_\phi \|\phi(x_k)\|$.
9:          Set $H_k \leftarrow \nabla^2_{\mathcal{I}_k \mathcal{I}_k} F(x_k)$ and $g_k \leftarrow \nabla_{\mathcal{I}_k} F(x_k)$.
10:         Compute the reference direction
                $$d_k^R \leftarrow -\alpha_k g_k, \quad \text{where} \quad \alpha_k \leftarrow \|g_k\|^2/(g_k^T H_k g_k).$$
11:         Choose $\mu_k \in (0,1]$ and compute any $\bar{d}_k \approx \text{argmin}\ m_k(d)$ that satisfies
                $$g_k^T \bar{d}_k \le g_k^T d_k^R, \tag{2}$$
                $$m_k(\bar{d}_k) \le m_k(0), \quad \text{and} \tag{3}$$
                $$\|H_k \bar{d}_k + g_k\| \le \min\{\mu_k \|g_k\|^r, \|g_k\|^2\}. \tag{4}$$
12:         Set $[d_k]_{\mathcal{I}_k} \leftarrow \bar{d}_k$ and $[d_k]_i \leftarrow 0$ for $i \notin \mathcal{I}_k$.
13:         Use Algorithm (2) to compute $x_{k+1} \leftarrow \text{LINESEARCH\_}\phi(x_k, d_k, \mathcal{I}_k, \eta, \xi)$.
14:     else                                                              $\triangleright\ k \in \mathcal{S}_\beta$
15:         Choose any $\mathcal{I}_k \subseteq \{i : [\beta(x_k)]_i \neq 0\}$ such that $\|[\beta(x_k)]_{\mathcal{I}_k}\| \ge \eta_\beta \|[\beta(x_k)]\|$.
16:         Set $[d_k]_{\mathcal{I}_k} \leftarrow -[\beta(x_k)]_{\mathcal{I}_k}$ and $[d_k]_i \leftarrow 0$ for $i \notin \mathcal{I}_k$.
17:         Use Algorithm (3) to compute $x_{k+1} \leftarrow \text{LINESEARCH\_}\beta(x_k, d_k, \eta, \xi)$.
18:     end if
19: end for
```

$\epsilon > 0$, the vector x_k is returned as an approximate solution to (1), as justified by Lemma 2.1. Otherwise, it proceeds in one of two ways depending on the relative sizes of $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$. We describe these cases next, and remark that a similar description is given in [6] and repeated here for convenience.

The relationship $\|\beta(x_k)\| \le \gamma \|\phi(x_k)\|$ indicates that substantial progress towards optimality can be achieved by reducing F over the current set of nonzero variables at x_k, which is the goal of lines 8–13. In line 8, the subset $\mathcal{I}_k$ of variables is chosen so that the norm of $\phi(x_k)$ over the chosen subset of variables is proportional to the norm of $\phi(x_k)$ over the full set of variables. This allows control over the size of the subproblem. Also note that the reduced space subproblem of minimizing $m_k(d)$ over $d \in \mathbb{R}^{|\mathcal{I}_k|}$ is aimed at minimizing F over the variables in $\mathcal{I}_k$. The global convergence analysis in [6] does not require an exact minimizer of m_k, but merely requires the computation of any direction $\bar{d}_k$ that satisfies (2) and (3) with the reference direction d_k^R computed in line 10 by minimizing m_k along the steepest decent direction. The first condition imposes how much descent is required of the search direction $\bar{d}_k$, and the second condition ensures that the model is reduced at least as much as a zero step. We highlight that in this paper we include the additional third condition (4) since it allows us to establish local superlinear convergence rates. (The $\|g_k\|^2$ quantity appearing in the right-hand side of (4) may be replaced by $\|\bar{d}_k\|^2$ without affecting the convergence theory, although a minor change to the

Algorithm 2 A line search procedure for computing x_{k+1} when $k \in \mathcal{S}_\phi$.

```
1: procedure $x_{k+1} = \text{LINESEARCH\_}\phi(x_k, d_k, \mathcal{I}_k, \eta, \xi)$
2:     Set $j \leftarrow 0$ and $y_0 \leftarrow \text{Proj}(x_k + d_k\,; x_k)$.
3:     while $\text{sgn}(y_j) \neq \text{sgn}(x_k)$ do
4:         if $F(y_j) \le F(x_k)$ then
5:             return $x_{k+1} \leftarrow y_j$.                              ▷ $k \in \mathcal{S}_\phi^{\text{ADD}}$
6:         end if
7:         Set $j \leftarrow j + 1$ and then $y_j \leftarrow \text{Proj}(x_k + \xi^j d_k\,; x_k)$.
8:     end while
9:     if $j \neq 0$ then
10:        Set $\alpha_B \leftarrow \text{argsup}\,\{\alpha > 0 : \text{sgn}(x_k + \alpha d_k) = \text{sgn}(x_k)\}$.
11:        Set $y_j \leftarrow x_k + \alpha_B d_k$.
12:        if $F(y_j) \le F(x_k) + \eta\alpha_B \nabla_{\mathcal{I}_k} F(x_k)^{\text{T}}[d_k]_{\mathcal{I}_k}$ then
13:            return $x_{k+1} \leftarrow y_j$.                              ▷ $k \in \mathcal{S}_\phi^{\text{ADD}}$
14:        end if
15:    end if
16:    loop
17:        if $F(y_j) \le F(x_k) + \eta\xi^j \nabla_{\mathcal{I}_k} F(x_k)^{\text{T}}[d_k]_{\mathcal{I}_k}$ then
18:            return $x_{k+1} \leftarrow y_j$.                              ▷ $k \in \mathcal{S}_\phi^{\text{SD}}$
19:        end if
20:        Set $j \leftarrow j + 1$ and then $y_j \leftarrow x_k + \xi^j d_k$.
21:    end loop
22: end procedure
```

Algorithm 3 A line search procedure for computing x_{k+1} when $k \in \mathcal{S}_\beta$.

```
1: procedure $x_{k+1} = \text{LINESEARCH\_}\beta(x_k, d_k, \eta, \xi)$
2:     Set $j \leftarrow 0$ and $y_0 \leftarrow x_k + d_k$.
3:     while $F(y_j) > F(x_k) - \eta\xi^j \|d_k\|^2$ do
4:         Set $j \leftarrow j + 1$ and then $y_j \leftarrow x_k + \xi^j d_k$.
5:     end while
6:     return $x_{k+1} \leftarrow y_j$.
7: end procedure
```

proof of Lemma 3.6 is required. We prefer using the quantity $\|g_k\|^2$ because then only the left-hand-side changes while the subproblem is being solved.) Importantly, all three conditions are satisfied by the exact solution to $H_k \bar{d}_k = -g_k$, and so are satisfied asymptotically by a convergent iterative algorithm such as CG or CD. Once $\bar{d}_k$ is computed, the search direction d_k in the full space is calculated by filling elements that correspond to the index set $\mathcal{I}_k$ with the elements from $\bar{d}_k$, and setting the complementary set of variables to zero (see line 12). Next, Algorithm 2 is called in line 13 to perform a (non-standard) backtracking projected line search. The while-loop that starts in line 3 of Algorithm 2 checks whether the trial point y_j decreases the objective function F relative to its value at x_k when $\text{sgn}(y_j) \neq \text{sgn}(x_k)$. If the line search terminates in this while-loop, then it means that at least one nonzero component of x_k has become zero at $x_{k+1} = y_j$. Since the dimension of the reduced space will therefore be reduced during the next iteration (provided line 7 of Algorithm 1 tests true), the procedure only requires $F(x_{k+1}) \le F(x_k)$ instead of a more traditional sufficient decrease condition such as the Armijo condition. If line 9 of Algorithm 2 is reached, the trial iterate y_j satisfies $\text{sgn}(y_j) = \text{sgn}(x_k)$, meaning that the trial iterate has entered the same orthant inhabited by x_k. Once this occurs, the method could perform a standard backtracking Armijo line search as indicated in the loop starting at line 16. However, to guarantee global convergence, the method first checks whether the largest step along d_k that stays in the same orthant as x_k (see lines 10 and 11) satisfies the Armijo condition in line 12, which makes our procedure a non-standard backtracking procedure. If Algorithm 2 terminates in either line 5 or line 13, then at least one nonzero component of x_k will be zero at x_{k+1}, which we indicate by saying $k \in \mathcal{S}_\phi^{\mathrm{ADD}} \subseteq \mathcal{S}_\phi$. On the other hand, if Algorithm 2 terminates in line 18, then x_{k+1} and x_k are in the same orthant and sufficient decrease in F was achieved (i.e. the Armijo condition in line 17 was satisfied), which we indicate by saying that $k \in \mathcal{S}_\phi^{\mathrm{SD}} \subseteq \mathcal{S}_\phi$.

When $\|\beta(x_k)\| > \gamma\|\phi(x_k)\|$, it is likely that progress towards optimality is best achieved by freeing at least one variable that is currently set to zero, which is the purpose of lines 15–17. Since $\|\beta(x_k)\|$ is relatively large, in line 15 of Algorithm 1 a subset $\mathcal{I}_k$ of variables is chosen such that the norm of $\beta(x_k)$ over that subset of variables is at least proportional to the norm of $\beta(x_k)$ over the full set of variables. This allows control over the number of zero variables that become nonzero during the iteration. The components of $\beta(x_k)$ that correspond to $\mathcal{I}_k$ are used to define the search direction d_k in line 16, which is itself used as an input to Algorithm 3 (a standard backtracking Armijo line search) when called in line 17 to obtain x_{k+1}. If a unit step length is taken, i.e. if $x_{k+1} = x_k + d_k$, then x_{k+1} can be interpreted as a *reduced ISTA step* in the space of variables indexed by $\mathcal{I}_k$ (see the appendix in [6]).

Before proceeding to our local convergence analysis, we remark that the global convergence guarantees established for FaRSA in [6] under Assumption 1.1 still hold for our slightly modified version of FaRSA given by Algorithm 1. This follows because the only difference between the two is that Algorithm 1 requires a more accurate subproblem solution since it demands that the extra condition (4) is satisfied in line 11. Of course, this makes sense because condition (4) has been introduced for the purpose of establishing local convergence rates, an aspect not considered in [6]. The precise global convergence result for algorithm FaRSA under Assumption 1.1 is as follows.

THEOREM 2.2 *Let x_* be the unique solution to problem* (1). *If ϵ in the finite termination condition in line* 4 *of Algorithm 1 is zero, then either*:

(i) *there exists an iteration k such that $x_k = x_*$; or*
(ii) *infinitely many iterates $\{x_k\}$ are computed yielding*

$$\lim_{k\to\infty} x_k = x_*, \quad \lim_{k\to\infty} \varphi(x_k) = 0, \quad \text{and} \quad \lim_{k\to\infty} \beta(x_k) = 0.$$

3. Local convergence analysis

In this section, we prove a superlinear local convergence rate guarantee for FaRSA. For the analysis, we assume in this section that case (i) of Theorem 2.2 does not occur and that $\epsilon = 0$ in Algorithm 1. Together, these assumptions imply that an infinite sequence of iterates $\{x_k\}_{k\geq 0}$ is computed by Algorithm 1.

A feature of FaRSA that we will show is that the iterates themselves reveal the variables that are zero, negative, and positive at the solution x_* after a finite number of iterations. Establishing this fact requires the following constants:

$$0 < \delta_1 := \frac{1}{2} \min_{i\in\mathcal{I}^{\pm}(x_*)} |[x_*]_i|$$

$$\text{and} \quad 0 < \delta_2 := \frac{1}{2} \min_{i\in\mathcal{I}^{0}(x_*)} (\lambda - |\nabla_i f(x_*)|).$$

We remark that δ_2 is positive as a consequence of Assumption 1.3.

The following result shows that the activities at x_k agree with the activities at x_* for certain components when x_k is sufficiently close to x_*. (Throughout, the word *activities* refers to the partition of a given vector into its positive, negative, and zero variables.)

LEMMA 3.1 *If* $\|x_k - x_*\| \le \delta_1/2$, *then*

$$sgn([x_k]_i) = sgn([x_*]_i) \quad \textit{for all } i \in (\mathcal{I}^0(x_*) \cap \mathcal{I}^0(x_k)) \cup \mathcal{I}^{\pm}(x_*).$$

Proof The conclusion holds for $i \in \mathcal{I}^0(x_*) \cap \mathcal{I}^0(x_k)$ from the definition of $\mathcal{I}^0$. To obtain the conclusion for $i \in \mathcal{I}^{\pm}(x_*)$, observe that since $\|x_k - x_*\| \le \delta_1/2$ by assumption, it follows from the definition of δ_1, the triangle inequality, and basic norm inequalities that

$$\begin{aligned}\text{for all } i \in \mathcal{I}^-(x_*)\text{: } [x_k]_i &= [x_k]_i - [x_*]_i + [x_*]_i \le |[x_k - x_*]_i| - 2\delta_1 \\ &\le \|x_k - x_*\| - 2\delta_1 \le \delta_1/2 - 2\delta_1 = -(3/2)\delta_1. \end{aligned} \tag{5}$$

Similarly,

$$\begin{aligned}\text{for all } i \in \mathcal{I}^+(x_*)\text{: } [x_k]_i &= [x_k]_i - [x_*]_i + [x_*]_i \ge -|[x_k - x_*]_i| + 2\delta_1 \\ &\ge -\|x_k - x_*\| + 2\delta_1 \ge -\delta_1/2 + 2\delta_1 = (3/2)\delta_1. \end{aligned} \tag{6}$$

The fact that the conclusion holds for $i \in \mathcal{I}^{\pm}(x_*)$ follows from (5) and (6). ■

We now give a bound on the magnitudes of the components of the gradient of f for elements in $\mathcal{I}^0(x_*)$ and prove that $\beta(x_k) = 0$ when x_k is sufficiently close to x_*.

LEMMA 3.2 *If* $\|x_k - x_*\| \le \min\{\delta_1/2, \delta_2/L_g\}$, *then the following hold:*

(i) $|\nabla_i f(x_k)| < \lambda - \delta_2$ *for all* $i \in \mathcal{I}^0(x_*)$,
(ii) $\beta(x_k) = 0$, *and*
(iii) $k \in \mathcal{S}_\phi$.

Proof It follows from the triangle inequality, basic norm inequalities, Lipschitz continuity of ∇f, and the assumption of the lemma that

$$\begin{aligned}|\nabla_i f(x_k)| - |\nabla_i f(x_*)| &\le |\nabla_i f(x_k) - \nabla_i f(x_*)| \\ &\le \|\nabla f(x_k) - \nabla f(x_*)\| \\ &\le L_g\|x_k - x_*\| \le L_g(\delta_2/L_g) = \delta_2 \quad \text{for all } i \in \{1, \ldots, n\}.\end{aligned}$$

It follows from this string of inequalities that

$$|\nabla_i f(x_k)| \le |\nabla_i f(x_*)| + \delta_2 \quad \text{for all } i \in \{1, \ldots, n\}. \tag{7}$$

Defining $c := \max_{i \in \mathcal{I}^0(x_*)} |\nabla_i f(x_*)|$, the definition of δ_2 implies that

$$\delta_2 = \frac{\lambda - c}{2}.$$

Combining this with (7) and the definition of c shows that

$$|\nabla_i f(x_k)| \le |\nabla_i f(x_*)| + \delta_2 \le c + \frac{\lambda - c}{2} = \frac{\lambda + c}{2} = \lambda - \delta_2 \quad \text{for all } i \in \mathcal{I}^0(x_*),$$

which proves part (i). Now, to prove part (ii), observe that we can only have $[\beta(x_k)]_i \ne 0$ if $i \in \mathcal{I}^0(x_k)$, where $\mathcal{I}^0(x_k) = (\mathcal{I}^0(x_*) \cap \mathcal{I}^0(x_k)) \cup (\mathcal{I}^{\pm}(x_*) \cap \mathcal{I}^0(x_k))$. However, from Lemma 3.1 and the assumption of the lemma, we have that $\mathcal{I}^{\pm}(x_*) \cap \mathcal{I}^0(x_k) = \emptyset$, while for $i \in \mathcal{I}^0(x_*) \cap \mathcal{I}^0(x_k)$ we have from part (i) that $[\beta(x_k)]_i = 0$. Thus, overall, $\beta(x_k) = 0$, as desired. Finally, part (iii) follows from part (ii) and line 7 of Algorithm 1. ■

Using the previous result allows us to prove the following lemma, which establishes that the activities of the iterates eventually do not change.

Lemma 3.3 *There exists an iteration $\bar{k}$ such that the following hold for all $k \geq \bar{k}$:*

(i) $\|x_k - x_*\| \leq \min\{\delta_1/2, \delta_2/L_g\}$,
(ii) $k \in \mathcal{S}_\phi^{\mathrm{SD}}$, *and*
(iii) $\mathcal{I}^0(x_k) = \mathcal{I}^0(x_{\bar{k}})$, $\mathcal{I}^+(x_k) = \mathcal{I}^+(x_{\bar{k}})$, *and* $\mathcal{I}^-(x_k) = \mathcal{I}^-(x_{\bar{k}})$.

Proof First, the fact that (i) holds for all sufficiently large k follows from Theorem 2.2(ii). Second, to prove that (ii) holds for sufficiently large k, observe that (i) and Lemma 3.2 imply that $k \in \mathcal{S}_\phi$ for all sufficiently large k. The fact that there exists an iteration $\bar{k}$ such that $k \in \mathcal{S}_\phi^{\mathrm{SD}}$ for all $k \geq \bar{k}$ now follows because (a) $\mathcal{S}_\phi = \mathcal{S}_\phi^{\mathrm{SD}} \cup \mathcal{S}_\phi^{\mathrm{ADD}}$, (b) for each $k \in \mathcal{S}_\phi^{\mathrm{ADD}}$ at least one nonzero component of x_k becomes zero at x_{k+1} and no components that were zero at x_k become nonzero at x_{k+1}, and (c) for each $k \in \mathcal{S}_\phi^{\mathrm{SD}}$ the activities at x_k are the same as the activities at x_{k+1}. Finally, using (ii) and the fact that the activities at x_k are the same as the activities at x_{k+1} for $k \in \mathcal{S}_\phi^{\mathrm{SD}}$ shows that (iii) holds. ■

Our next goal is to establish that the activities at x_k for $k \geq \bar{k}$ are the same as the activities at the solution x_*. We require the following straightforward lemma.

Lemma 3.4 *For $\bar{k}$ defined in Lemma 3.3, it follows that*

$$\lim_{\bar{k} \leq k \to \infty} \|g_k\| = 0 \quad \text{and} \quad \lim_{\bar{k} \leq k \to \infty} \|\bar{d}_k\| = 0.$$

Proof We first observe from Lemma 3.3(ii) that $k \in \mathcal{S}_\phi^{\mathrm{SD}}$ for all $k \geq \bar{k}$. Thus, it follows from [6, proof of Lemma 9] that

$$F(x_{k+1}) \leq F(x_k) - \min\left\{\frac{\eta}{\theta_{\max}}, \frac{\eta\xi(1-\eta)\theta_{\min}^2}{2\theta_{\max}^3}\right\}\|g_k\|^2 \quad \text{for all } k \geq \bar{k}.$$

This inequality and the fact that F is bounded below as a consequence of Assumption 1.1 allows us to deduce that $\lim_{k\to\infty}\|g_k\| = 0$, which is the first desired limit. Combining the first limit with [6, Lemma 8] and $\mathcal{S}_\phi^{\mathrm{SD}} \subseteq \mathcal{S}_\phi$ allows us to conclude that $\lim_{k\to\infty}\|\bar{d}_k\| = 0$, which completes the proof. ■

We can now prove a finite active set identification result. For this, since we already know from Lemma 3.3(iii) that the activities of the sequence $\{x_k\}$ do not change for $k \geq \bar{k}$, all that remains is to show that they agree with the activities of x_*.

Lemma 3.5 *If we choose $\mathcal{I}_k = \{i \in \mathcal{I} : [\phi(x_k)]_i \neq 0\}$ in line 8 of Algorithm 1 for all $k \geq \bar{k}$ with $\bar{k}$ defined in Lemma 3.3, then it holds that*

$$\mathcal{I}^0(x_k) = \mathcal{I}^0(x_*), \quad \mathcal{I}^+(x_k) = \mathcal{I}^+(x_*), \quad \textit{and} \quad \mathcal{I}^-(x_k) = \mathcal{I}^-(x_*) \quad \textit{for all } k \geq \bar{k}.$$

Proof We first observe from Lemma 3.4 and the choice for g_k in Algorithm 1 that

$$0 = \lim_{\bar{k} \le k \to \infty} \|\nabla_{\mathcal{I}_k} F(x_k)\| = \lim_{\bar{k} \le k \to \infty} \|[\nabla f(x_k) + \lambda \zeta]_{\mathcal{I}_k}\|, \tag{8}$$

where

$$[\zeta]_i := \begin{cases} 1 & \text{if } i \in \mathcal{I}^+(x_{\bar{k}}), \\ -1 & \text{if } i \in \mathcal{I}^-(x_{\bar{k}}), \\ \text{arbitrary} & \text{if } i \in \mathcal{I}^0(x_{\bar{k}}). \end{cases}$$

(We remind the reader that from the choice of $\mathcal{I}_k$ in the statement of this lemma, the definition of ϕ, and Lemma 3.3(iii) that $\mathcal{I}_k = \{i \in \mathcal{I} : [\phi(x_k)]_i \neq 0\} \subseteq \mathcal{I}^\pm(x_k) = \mathcal{I}^\pm(x_{\bar{k}})$ for all $k \ge \bar{k}$. In particular, this means that $\nabla_{\mathcal{I}_k} F(x_k)$ is well defined in (6).)

Next, as an intermediate result, we claim that $\mathcal{I}_k \cap \mathcal{I}^0(x_*) = \emptyset$ for all sufficiently large k. For a proof by contradiction, suppose that there exists an infinite subsequence of iteration numbers $\mathcal{K}$ and an index j such that $j \in \mathcal{I}_k \cap \mathcal{I}^0(x_*)$ for all $k \in \mathcal{K}$. It follows from $j \in \mathcal{I}^0(x_*)$ that $[x_*]_j = 0$. On the other hand, from $j \in \mathcal{I}_k$ for all $k \in \mathcal{K}$ and (8), it follows that $0 = \lim_{\bar{k} \le k \in \mathcal{K}} [\nabla f(x_k) + \lambda \zeta]_j = [\nabla f(x_*) + \lambda \zeta]_j$, which implies that $|\nabla_j f(x_*)| = \lambda$. This violates Assumption 1.3 since $j \in \mathcal{I}^0(x_*)$, which means that we must conclude that $\mathcal{I}_k \cap \mathcal{I}^0(x_*) = \emptyset$ for all sufficiently large k, as claimed.

Next, to prove that $\mathcal{I}^0(x_*) \subseteq \mathcal{I}^0(x_k)$ for all $k \ge \bar{k}$, let $j \in \mathcal{I}^0(x_*)$. Since we have shown that $\mathcal{I}_k \cap \mathcal{I}^0(x_*) = \emptyset$ for all sufficiently large k, we can conclude that $j \notin \mathcal{I}_k$ for all sufficiently large k. This may be combined with $k \in \mathcal{S}_\phi^{\text{SD}}$ for all $k \ge \bar{k}$ (recall Lemma 3.3(ii)) and line 12 of Algorithm 1 to conclude that $[d_k]_j = 0$ for all sufficiently large k. Since the update procedure for Algorithm 1 is given by $x_{k+1} = x_k + \alpha_k d_k$ for some positive α_k, we can conclude that $[x_{k+1}]_j = [x_k]_j$ for all sufficiently large k. However, since we also know from Theorem 2.2(ii) that $\{x_k\} \to x_*$, we can see that $[x_k]_j = [x_*]_j = 0$ for all sufficiently large k. Finally, if we observe that the activities of $\{x_k\}$ do not change for $k \ge \bar{k}$ (see Lemma 3.3(iii)), we can conclude the stronger statement that $[x_k]_j = 0$ for all $k \ge \bar{k}$, i.e. that $j \in \mathcal{I}^0(x_k)$ for all $k \ge \bar{k}$. Overall, we have established that $\mathcal{I}^0(x_*) \subseteq \mathcal{I}^0(x_k)$ for all $k \ge \bar{k}$.

To prove the opposite inclusion, i.e. that $\mathcal{I}^0(x_k) \subseteq \mathcal{I}^0(x_*)$ for all $k \ge \bar{k}$, suppose that $j \in \mathcal{I}^0(x_{\hat{k}})$ for some $\hat{k} \ge \bar{k}$. It follows from Lemma 3.3(iii) that $j \in \mathcal{I}^0(x_k)$ for all $k \ge \bar{k}$, meaning that $[x_k]_j = 0$ for all $k \ge \bar{k}$. Then, if we use the fact that $\{x_k\} \to x_*$ (from Theorem 2.2(ii)), we must conclude that $[x_*]_j = 0$, i.e. that $j \in \mathcal{I}^0(x_*)$. Thus, we have shown that $\mathcal{I}^0(x_k) \subseteq \mathcal{I}^0(x_*)$ for all $k \ge \bar{k}$. Combining this with the conclusion from the previous paragraph yields $\mathcal{I}^0(x_k) = \mathcal{I}^0(x_*)$ for all $k \ge \bar{k}$, which is the first desired result.

We next prove that $\mathcal{I}^+(x_k) = \mathcal{I}^+(x_*)$ for all $k \ge \bar{k}$. The inclusion $\mathcal{I}^+(x_*) \subseteq \mathcal{I}^+(x_k)$ for all $k \ge \bar{k}$ follows from Lemmas 3.1 and 3.3(i). Thus, it remains to prove the opposite inclusion that $\mathcal{I}^+(x_k) \subseteq \mathcal{I}^+(x_*)$ for all $k \ge \bar{k}$. For this purpose, suppose that $j \in \mathcal{I}^+(x_{\hat{k}})$ for some $\hat{k} \ge \bar{k}$, which because of Lemma 3.3(iii) implies that $j \in \mathcal{I}^+(x_k)$ for all $k \ge \bar{k}$, i.e. that $[x_k]_j > 0$ for all $k \ge \bar{k}$. Since $\{x_k\} \to x_*$ (from Theorem 2.2), this means that $[x_*]_j \ge 0$ so that

$$j \in \mathcal{I}^0(x_*) \cup \mathcal{I}^+(x_*) = \mathcal{I}^0(x_k) \cup \mathcal{I}^+(x_*) \quad \text{for all } k \ge \bar{k}; \tag{9}$$

here, we have used the already-proved fact that $\mathcal{I}^0(x_k) = \mathcal{I}^0(x_*)$ for all $k \ge \bar{k}$. Since from above we know that $j \in \mathcal{I}^+(x_k)$ for all $k \ge \bar{k}$, and that by construction $\mathcal{I}^+(x_k) \cap \mathcal{I}^0(x_k) = \emptyset$, we can conclude from (9) that $j \in \mathcal{I}^+(x_*)$. Thus, we have shown that $\mathcal{I}^+(x_k) \subseteq \mathcal{I}^+(x_*)$ for all $k \ge \bar{k}$. Since we have established both inclusions, we have proved that $\mathcal{I}^+(x_k) = \mathcal{I}^+(x_*)$ for all $k \ge \bar{k}$, as claimed.

The final result, namely that $\mathcal{I}^-(x_k) = \mathcal{I}^-(x_*)$ for all $k \ge \bar{k}$, may be proved using the same approach as in the previous paragraph. ■

Before giving a local rate of convergence result for Algorithm 1, we must establish that the unit stepsize is accepted for sufficiently large k.

Lemma 3.6 *The iterate update computed by Algorithm* 1 *is given by*

$$x_{k+1} = x_k + d_k \quad \textit{for all sufficiently large } k,$$

where d_k is defined in line 12 *using $\bar{d}_k$ computed to satisfy the conditions in line* 11.

Proof Lemma 3.3(ii), Lemma 3.4, and line 12 of Algorithm 1 give $\lim_{k\to\infty}\|d_k\| = 0$. Combining this with Lemma 3.5 and $\{x_k\} \to x_*$ (see Theorem 2.2) shows that $x_k + d_k$ is in the same orthant as x_k for all sufficiently large k, i.e.that $\operatorname{sgn}(x_k + d_k) = \operatorname{sgn}(x_k)$ for all sufficiently large k. Since we know that $k \in \mathcal{S}_\phi^{\mathrm{SD}}$ for all sufficiently large k (see Lemma 3.3(ii)), we know that Algorithm 1 calls the line search Algorithm 2 for all sufficiently large k. Moreover, since $\operatorname{sgn}(x_k + d_k) = \operatorname{sgn}(x_k)$ for all sufficiently large k, Algorithm 2 will find that $j = 0$ when line 9 is reached, which means that the condition in line 17 will first be checked. In what follows, we proceed to show that this condition in line 17 will indeed be satisfied with $j = 0$, thereby showing that $x_{k+1} = x_k + d_k$ and completing the proof of the lemma.

To prove that the condition in line 17 of Algorithm 2 is satisfied with $j = 0$, we first notice from lines 10 and 11 of Algorithm 1, Assumption 1.1, the definition of H_k, the interlacing property of the eigenvalues of submatrices, and [6, Lemma 8] that

$$|{g_k}^{\mathrm{T}}\bar{d}_k| \geq |{g_k}^{\mathrm{T}} d_k^R| = \frac{\|g_k\|^4}{g_k^{\mathrm{T}} H_k g_k} \geq \frac{\|g_k\|^4}{\|H_k\|\,\|g_k\|^2} \geq \frac{\|g_k\|^2}{\theta_{\max}}. \tag{10}$$

Next, from the Mean Value Theorem, the fact that F is twice continuously differentiable on the interval $[x_k, x_k + d_k]$, the definitions of g_k and $\bar{d}_k$, and the fact that the second derivatives of the regularizer are zero for $i \in \mathcal{I}_k$, there exists $z_k \in [x_k, x_k + d_k]$ such that

$$\begin{aligned} F(x_k + d_k) &= F(x_k) + \nabla_{\mathcal{I}_k} F(x_k)^{\mathrm{T}} [d_k]_{\mathcal{I}_k} + \tfrac{1}{2} [d_k]_{\mathcal{I}_k}{}^{\mathrm{T}} \nabla^2_{\mathcal{I}_k\mathcal{I}_k} f(z_k) [d_k]_{\mathcal{I}_k} \\ &= F(x_k) + {g_k}^{\mathrm{T}}\bar{d}_k + \tfrac{1}{2}\bar{d}_k{}^{\mathrm{T}} \nabla^2_{\mathcal{I}_k\mathcal{I}_k} f(z_k)\bar{d}_k. \end{aligned}$$

Combining this with Assumption 1.2, Theorem 2.2(ii), $\lim_{k\to\infty}\|d_k\| = 0$, line 11 in Algorithm 1, and the fact that $\|\bar{d}_k\| \leq (2/\theta_{\min})\|g_k\|$ (see [6, Lemma 8]) gives

$$\begin{aligned} F(x_k + d_k) - F(x_k) - \frac{1}{2}{g_k}^{\mathrm{T}}\bar{d}_k &= \frac{1}{2}({g_k}^{\mathrm{T}}\bar{d}_k + \bar{d}_k{}^{\mathrm{T}}\nabla^2_{\mathcal{I}_k\mathcal{I}_k} f(z_k)\bar{d}_k) \\ &= \frac{1}{2}({g_k}^{\mathrm{T}}\bar{d}_k + \bar{d}_k{}^{\mathrm{T}} H_k \bar{d}_k) + \frac{1}{2}(\bar{d}_k{}^{\mathrm{T}}(\nabla^2_{\mathcal{I}_k\mathcal{I}_k} f(z_k) - H_k)\bar{d}_k) \\ &\leq \frac{1}{2}\bar{d}_k{}^{\mathrm{T}}(H_k\bar{d}_k + g_k) + \frac{1}{2}L_H\|\bar{d}_k\|^3 \\ &\leq \frac{1}{2}\|\bar{d}_k\|\,\|H_k\bar{d}_k + g_k\| + \frac{1}{2}L_H\|\bar{d}_k\|^3 \\ &\leq \frac{1}{2}\|\bar{d}_k\|\,\|g_k\|^2 + \frac{2L_H}{\theta_{\min}^2}\|\bar{d}_k\|\,\|g_k\|^2 \\ &= \left(\frac{1}{2} + \frac{2L_H}{\theta_{\min}^2}\right)\|\bar{d}_k\|\,\|g_k\|^2 \quad \text{for all sufficiently large } k. \end{aligned} \tag{11}$$

Since $\eta \in (0, \frac{1}{2})$, we know from Lemma 3.4 and Assumption 1.1 that

$$\theta_{\max}\left(\frac{1}{2}+\frac{2L_H}{\theta_{\min}^2}\right)\|\bar{d}_k\| \le \frac{1}{2}(1-2\eta) \quad \text{for all sufficiently large } k. \tag{12}$$

It now follows from (11), (10), (12), and $g_k^{\mathrm{T}}\bar{d}_k < 0$ that

$$\begin{aligned} F(x_k+d_k)-F(x_k) &\le \tfrac{1}{2}{g_k}^{\mathrm{T}}\bar{d}_k + \left(\frac{1}{2}+\frac{2L_H}{\theta_{\min}^2}\right)\|\bar{d}_k\|\|g_k\|^2 \\ &\le \frac{1}{2}{g_k}^{\mathrm{T}}\bar{d}_k + \theta_{\max}\left(\frac{1}{2}+\frac{2L_H}{\theta_{\min}^2}\right)\|\bar{d}_k\||{g_k}^{\mathrm{T}}\bar{d}_k| \\ &\le \frac{1}{2}{g_k}^{\mathrm{T}}\bar{d}_k + \frac{1}{2}(1-2\eta)|{g_k}^{\mathrm{T}}\bar{d}_k| \\ &= \frac{1}{2}(1-(1-2\eta)){g_k}^{\mathrm{T}}\bar{d}_k \\ &= \eta {g_k}^{\mathrm{T}}\bar{d}_k \quad \text{for all sufficiently large } k, \end{aligned} \tag{13}$$

which after rearrangement and use of the definitions of g_k and $\bar{d}_k$ implies that

$$F(x_k+d_k) \le F(x_k) + \eta\nabla_{\mathcal{I}_k}F(x_k)^{\mathrm{T}}[d_k]_{\mathcal{I}_k}. \tag{14}$$

Therefore, we have shown that the condition in line 17 is satisfied when $j=0$ for all sufficiently large k, thereby completing the proof of this lemma. ■

We showed in Lemma 3.5 that, after a finite number of iterations, the signs of the variables at x_k and x_* are the same. Moreover, Lemma 3.6 shows that for sufficiently large k, the iteration for Algorithm 1 takes the form $x_{k+1} = x_k + d_k$, where d_k is defined in line 12 using $\bar{d}_k$ computed to satisfy the conditions in line 11. Therefore, the iterations of Algorithm 1 in the space of nonzero variables are exactly those of an inexact Newton method for computing a zero of ∇f restricted to the components in the set $\mathcal{I}^{\pm}(x_*)$. Consequently, the next result follows from [7, Theorem 3.3].

LEMMA 3.7 *Let us define*

$$\bar{x}_k := [x_k]_{\mathcal{I}^{\pm}(x_*)} \quad \textit{and} \quad \bar{x}_* := [x_*]_{\mathcal{I}^{\pm}(x_*)}.$$

Suppose that we choose $\mathcal{I}_k = \{i \in \mathcal{I} : [\phi(x_k)]_i \neq 0\}$ in line 8 of Algorithm 1 for all sufficiently large k. Then, if either $r \in (1, 2]$, or $r = 1$ and $\{\mu_k\} \to 0$, then $\{\bar{x}_k\} \to \bar{x}_*$ at a superlinear rate. In particular, if we choose $r = 2$, then the rate is quadratic.

We now state our final rate-of-convergence result.

THEOREM 3.8 *Suppose that $\mathcal{I}_k = \{i \in \mathcal{I} : [\phi(x_k)]_i \neq 0\}$ in line 8 of Algorithm 1 for all sufficiently large k. Then, if either $r \in (1, 2]$, or $r=1$ and $\{\mu_k\} \to 0$, then $\{x_k\} \to x_*$ at a superlinear rate. In particular, if we choose $r = 2$, then the rate is quadratic.*

Proof The result follows from Lemma 3.7 and since $[x_k]_{\mathcal{I}^0(x_*)} = [x_*]_{\mathcal{I}^0(x_*)} = 0$ for all $k \ge \bar{k}$ (with $\bar{k}$ defined in Lemma 3.3) as a result of Lemma 3.5. ■

4. Numerical performance

We present results when employing an implementation of FaRSA to solve a collection of ℓ_1-norm regularized logistic regression problems of the form (1) with

$$f(x) := \frac{1}{N}\sum_{\ell=1}^{N} \log(1 + \mathrm{e}^{-y_\ell d_\ell^{\mathrm{T}} x}), \tag{15}$$

where $d_\ell \in \mathbb{R}^n$ is the ℓth data vector, N is the number of data vectors in the dataset, $y_\ell \in \{-1, 1\}$ is the class label for the ℓth data vector, and $\lambda = 1/N$ is the weighting parameter in (1). We refer to $D \in \mathbb{R}^{N\times n}$, whose ℓth row is d_ℓ^{T} , as the data matrix. Such problems arise in the context of model prediction, making the design of advanced optimization algorithms that efficiently solve them paramount in big data applications.

4.1 *Implementation details*

We have developed a C-library implementation of FaRSA that is based on Algorithm 1. Although our software allows for other convex functions to be used—in which case, the user must supply subroutines for computing $f(x)$, $\nabla f(x)$, and $\nabla^2 f(x)$—it includes specialized subroutines for solving (1) when f is the logistic loss (13). The special features of this implementation are described later in this subsection. Our implementation may be obtained from our Github repository https://github.com/daniel-p-robinson/FaRSA.

The default parameter values for FaRSA, which are also the values used in our numerical experiments for this paper, have been chosen based on a rough tuning procedure that resulted in the following. We use $\gamma = 1$ in Step 7 of Algorithm 1 so that no preference is given to whether the kth iteration is in $\mathcal{S}_\beta$ or $\mathcal{S}_\phi$. In Step 8, we use $\mathcal{I}_k \equiv \{i : [\phi(x_k)]_i \neq 0\}$, which is equivalent to choosing $\eta_\phi = 1$. However, our choice for $\mathcal{I}_k$ in Step 15 is based on taking the largest (in absolute value) 80% (rounded up to the nearest integer) of the entries from β, which means that $\|[\beta(x_k)]_{\mathcal{I}_k}\| \geq \|\beta(x_k)\|_\infty \geq (1/\sqrt{n})\|\beta(x_k)\|_2$ so that the required condition in Step 15 holds with $\eta_\beta = 1/\sqrt{n}$. For the line search procedures stated as Algorithm 2 and Algorithm 3, which are called in Steps 13 and 17 of Algorithm 1, we use $\eta = 10^{-2}$ and $\xi = 0.5$. As an initial solution estimate we use $x_0 = 0$, and we terminate if x_k satisfies

$$\max\{\|\beta(x_k)\|, \|\phi(x_k)\|\} \leq \epsilon \max\{1, \|\beta(x_0)\|, \|\phi(x_0)\|\}$$

with $\epsilon = 10^{-6}$, which is a relative stopping condition based on Step 4 of Algorithm 1.

Our strategy for computing the search direction d_k when $k \in \mathcal{S}_\phi$ is the one that we found to yield the best practical performance. As is well known, the CG algorithm is a popular and effective choice for computing an approximate solution to the linear system $H_k\bar{d} = -g_k$ associated with the data in Step 9 of Algorithm 1. Our strategy is based on monitoring the CG iterates and terminating when certain conditions are satisfied. In order to describe these conditions for the kth outer iteration, let $d_{j,k}$ denote the jth CG iterate, $r_{j,k} = \|H_k d_{j,k} + g_k\|$ denote the jth CG residual, and $v_{j,k}$ denote the number of entries in the vector $[x_k]_{\mathcal{I}_k} + d_{j,k}$ that fall into a different orthant than $[x_k]_{\mathcal{I}_k}$. We decide to terminate the CG method when any of the following hold:

$$v_{j,k} \geq \min\{10^7, 0.25|\mathcal{I}_k|\}; \text{ or} \tag{16a}$$

$$\|d_{j,k}\| \geq \delta_{k,\phi} := \max\{10^{-3}, \min\{10^3, 10\|x_{k_\phi(k)+1} - x_{k_\phi(k)}\|\}\}; \text{ or} \tag{16b}$$

$$r_{j,k} \leq \max\{10^{-1}\min\{r_{0,k}, r_{0,k}^2\}, 10^{-12}\}; \text{ or} \tag{16c}$$

$$r_{j,k} \le \max\{10^{-1} r_{0,k}, 10^{-12}\} \text{ and either } v_{j,k-1} > 0 \text{ or } \|\beta(x_{k-1})\| > \epsilon; \tag{16d}$$

where $k_\phi(k) := \max\{\bar{k} : \bar{k} \in \mathcal{S}_\phi \text{ and } \bar{k} < k\}$ and $\epsilon = 10^{-6}$. When the inequality in (16a) holds, it indicates that a relatively large number of entries in $[x_k]_{\mathcal{I}_k} + d_{j,k}$ are in a different orthant than $[x_k]_{\mathcal{I}_k}$. In this case, it makes sense to terminate CG since the exact solution is likely to be in a different orthant. In the second termination condition (16b), the value $\delta_{k,\phi}$ plays a role similar to a trust region constraint for trust region methods; in particular, the value $\|x_{k_\phi(k)+1} - x_{k_\phi(k)}\|$ is the size of the step computed during the last iteration in $\mathcal{S}_\phi$. When the inequality in (16b) tests true, the size of the CG trial iterate $d_{j,k}$ is deemed to be relatively large. In this case it makes sense to terminate CG because it is unlikely that x_k is 'near' a minimizer to the optimization problem. (We remind the reader of the well-known property that, if initialized with the zero vector, the iterates produced by CG are monotonically increasing in terms of Euclidean norm.) In some sense, the best termination outcome is (16c), which requires (ignoring the numerical safeguard value of 10^{-12}) that the residual of the Newton linear system is sufficiently reduced. In particular, condition (16c) aims to achieve a quadratic decrease in the residual (confer with Step 4 of Algorithm 1) as a minimizer of the optimization problem is approached, which allows for the superlinear rate of convergence that we established in Theorem 3.8. Finally, we also allow for termination if the conditions in (16d) hold. These conditions include a relaxed requirement on the reduction of the CG residual as compared to (16c), which we allow since when either $v_{j,k-1} > 0$ or $\|\beta(x_{k-1})\| > \epsilon$, we expect that x_k is not near a minimizer. For context, note that Lemma 3.2 shows that $\beta(x_k)$ would be zero when x_k is close enough to a minimizer x_*, and Lemma 3.5 shows that all iterates corresponding to sufficiently large iteration numbers would lie in the same orthant as x_*. Overall, the relaxed requirement on the CG residual in (16d) aims to avoid excessive computation.

In order to compute the CG iterates $\{d_{j,k}\}_{j\ge 0}$, we must be able to perform Hessian-vector products. In order to obtain the derivatives of f, let us define $\tau(\zeta) := 1/(1+e^{-\zeta})$. We can now observe that the following relationship holds:

$$1 - \tau(\zeta) = 1 - \frac{1}{1+e^{-\zeta}} = \frac{e^{-\zeta}}{1+e^{-\zeta}} =: \sigma(-\zeta), \tag{17}$$

which may be used to show that the gradient of the logistic function f in (15) is

$$\nabla f(x) = -\frac{1}{N}\sum_{\ell=1}^{N}[1 - \tau(y_\ell d_\ell{}^{\mathrm{T}} x)] y_\ell d_\ell = -\frac{1}{N}\sum_{\ell=1}^{N}\sigma(-y_\ell d_\ell{}^{\mathrm{T}} x) y_\ell d_\ell = -\frac{1}{N}(v(x)^{\mathrm{T}} D)^{\mathrm{T}}$$

with

$$[v(x)]_\ell := \sigma(-y_\ell d_\ell{}^{\mathrm{T}} x) y_\ell.$$

The second derivative of f may be derived by letting $d_\ell(i)$ denote the ith element of d_ℓ, using $y_\ell^2 = 1$ because $y_\ell \in \{-1, 1\}$ for all $\ell \in \{1, 2, \ldots, N\}$, and (17) to obtain

$$[\nabla^2 f(x)]_{ij} = \frac{\mathrm{d}}{\mathrm{d}x_j}[\nabla_i f(x)] = \frac{1}{N}\sum_{\ell=1}^{N}\sigma(-y_\ell d_\ell{}^{\mathrm{T}} x)[1 - \sigma(-y_\ell d_\ell{}^{\mathrm{T}} x)] d_\ell(j) d_\ell(i)).$$

Using this fact, we now observe that the second-derivative matrix may be written as

$$\nabla^2 f(x) = (1/N) D^{\mathrm{T}} E(x) D, \tag{18}$$

where $E(x)$ is a diagonal matrix with entries

$$[E(x)]_{\ell\ell} := \sigma(-y_\ell d_\ell{}^{\mathrm{T}} x)[1 - \sigma(-y_\ell d_\ell{}^{\mathrm{T}} x)] \quad \text{for all } \ell \in \{1, 2, \ldots, N\}.$$

Using (18) and the definition of the matrix H_k in Step 9 of Algorithm 1, it follows that the required Hessian-vector products take the form

$$H_k s = [\nabla^2_{\mathcal{I}_k \mathcal{I}_k} f(x_k)]s = (1/N)D(:,\mathcal{I}_k)^{\mathrm{T}} E(x) D(:,\mathcal{I}_k)s = (1/N)(w(x)^T D(:,\mathcal{I}_k))^{\mathrm{T}},$$

where

$$[w(x)]_\ell := [E(x)]_{\ell\ell}[D(:,\mathcal{I}_k)s]_\ell.$$

Therefore, each Hessian-vector product proceeds by first multiplying s by a subset of the data matrix (namely $D(:,\mathcal{I}_k)$), followed by N scalar multiplications to obtain the vector $w(x)$, followed by inner-products of $w(x)$ with $|\mathcal{I}_k|$ columns from the data matrix, and a final scaling by $1/N$. It follows that the overall efficiency of computing each Hessian-vector product depends on the size of the set $\mathcal{I}_k$ and the sparsity of D.

4.2 *Computational set-up and datasets used*

Our tests were conducted using the cluster in the Computational Optimization Research Laboratory at the Lehigh University (COR@L) with two-core 2.0GHz AMD CPUs and 30GB of main memory. The datasets used were obtained from the LIBSVM repository in the following manner. First, we excluded all datasets that were for classification of more than two items (i.e. datasets with greater than two labels). Among the remaining binary classification datasets, we removed *criteo*, *kdd2010(algebra)*, *kdd2010(bridge to algebra)*, *kdd2010 raw version (bridge to algebra)*, *kdd2012*, *webspam*, and *splice-site* because they caused an out-of-memory error for FaRSA. (All of these data sets, except *kdd2010(algebra)* and *kdd2010(bridge to algebra)*, also gave an out-of-memory error for the other codes we tested; see Section 4.3. For the remaining two datasets, namely *kdd2010(algebra)* and *kdd2010(bridge to algebra)*, the other codes both reached the maximum allowed iterations. The extra memory required by FaRSA is mostly due to the storage of the reduced-Hessian matrix needed for the reduced-space ϕ-step calculations.) Finally, for the adult datasets (*a1a–a9a*) and webpage datasets (*w1a–w8a*), we only used the largest instances, namely *a9a* and *w8a*. This left us with the 30 datasets summarized in Table 1. (In the table, density refers to the number of nonzero entries in the data matrix D divided by $N \times n$.) Since the LIBSVM data sets seem to be updated periodically, we are happy to share the data set instances that we used upon request.

4.3 *Numerical experiments*

We first compare the performance of FaRSA with the state-of-the-art C++/C solvers LIBLINEAR and ASA-CG on the datasets in Table 1. The LIBLINEAR solver [16] is an implementation of a proximal-Newton method that uses coordinate blockwise minimization to approximately solve its subproblems. Algorithm ASA-CG [11] is a successful bound-constrained optimization method that we use to solve a smooth bound-constrained reformulation of problem (1). Specifically, we use ASA-CG to solve the following:

$$\underset{u\in\mathbb{R}^n, v\in\mathbb{R}^n}{\text{minimize}}\, f(u-v) + \lambda e^{\mathrm{T}}(u+v) \;\text{ subject to }\; u \geq 0,\; v \geq 0,$$

where f is the logistic function defined in (15). For both LIBLINEAR and ASA-CG, we used their default parameter settings. For all three algorithms, we used their native termination conditions with a termination tolerance value of $\epsilon = 10^{-6}$.

The results of our tests on the datasets from Section 4.2 are presented in Table 2. In order to obtain reliable timing information, we report under the column ‘Time (seconds)’ the average

Table 1. Subset of binary classification datasets from the LIBSVM repository.

Dataset	N	n	Density (%)	Size (MB)
fourclass	862	2	100.000	0.024
svmguide1	3089	4	99.701	0.104
cod-rna	59,535	8	100.000	4.451
breast-cancer	683	10	100.000	0.078
australian	690	14	87.443	0.068
skin–nonskin	245,057	3	98.267	6.430
splice	1000	60	100.000	0.517
heart	270	13	96.239	0.026
german.numer	1000	24	100.000	0.273
diabetes	768	8	100.000	0.072
madelon	2000	500	100.000	7.428
w8a	49,749	300	3.883	4.926
a9a	32,561	123	11.276	2.222
liver-disorders	345	6	100.000	0.014
sonar	208	60	99.992	0.149
ijcnn1	49,990	22	59.091	7.386
svmguide3	1243	22	84.335	0.165
gisette	6000	5000	12.971	47.398
real-sim	72,309	20,958	0.245	86.191
mushrooms	8124	112	18.750	0.839
covtype.binary	581,012	54	21.998	59.581
rcv1.binary	20,242	47,236	0.157	34.849
colon-cancer	62	2000	100.000	1.647
leukaemia	34	7129	100.000	3.723
duke-breast-cancer	38	7129	100.000	3.721
news20	19,996	1,355,191	0.034	133.524
avazu-app.tr	12,642,186	999,990	0.002	2306.000
HIGGS	11,000,000	28	92.112	7387.000
url_combined	2,396,130	3,231,961	0.004	2058.000
SUSY	5,000,000	18	98.820	2301.000

CPU time over 10 trials for each solver on each dataset. The value of the objective function $F(x) = f(x) + \lambda\|x\|_1$ at the final iterate is stated under the column 'F-final'. For all three methods, we allowed a maximum of 12 hours of computing time. We indicate any instance for which the 12 hours was reached before the termination condition was satisfied by writing 'max time'. (We did not have any situations in which, for a particular dataset, a solver reached the time limit for some trials and not for others.)

To facilitate a comparison between the three algorithms, for each dataset the shortest required computing time is noted with red font, the second shortest with blue font, and the longest with black font. Although the different solvers employ their own distinct termination conditions, they achieve the same final objective function values (to five digits of accuracy) on most of the datasets. Notable exceptions for which convergence for all methods occurs are datasets *breast-cancer*, *skin-nonskin*, *covtype.binary*, and *leukaemia* although they do not show any clear preference for one algorithm. Finally, although the maximum time limit is reached by LIBLINEAR and ASA-CG on datasets *avazu-app.tr* and *url_combined* they still manage to find comparable final objective values, thus indicating that the termination condition plays an important role for these two datasets.

Table 2 shows that FaRSA generally performs better than LIBLINEAR and ASA-CG on these datasets. In fact, FaRSA achieves the best computational time on 19 datasets and achieves the second best computational time on 11 of the datasets. The second best performer is LIBLINEAR, which achieves the best computational time on 10 datasets and achieves the second best computational time on 7 of the datasets. In terms of reliability, FaRSA is the best since it succeeds

Table 2. The required computing time and final objective function value for algorithms FaRSA, LIBLINEAR, and ASA-CG on the datasets listed in Table 1.

	Time (seconds)			F-final		
Dataset	FaRSA	LIBLINEAR	ASA-CG	FaRSA	LIBLINEAR	ASA-CG
fourclass	0.0077	0.0093	0.0083	0.53305	0.53305	0.53305
svmguide1	0.0177	0.0197	0.0836	0.32350	0.32350	0.32350
cod-rna	1.2096	10.4215	3.5237	0.19312	0.19312	0.19312
breast-cancer	0.0136	0.0074	0.0712	0.15478	0.14626	0.14626
australian	0.0138	0.0336	0.0162	0.33669	0.33669	0.33669
skin-nonskin	1.5154	4.3514	0.7119	0.00085	0.35153	0.00079
splice	0.0098	0.0110	0.0187	0.50671	0.50671	0.50671
heart	0.0072	0.0037	0.0073	0.38025	0.38025	0.38025
german.numer	0.0704	0.0351	0.3066	0.48005	0.48005	0.48005
diabetes	0.0222	0.0139	0.1416	0.60913	0.60913	0.60913
madelon	21.1252	30496.0909	903.6747	0.51685	0.51685	0.51685
w8a	1.6514	1.8244	9.0901	0.12206	0.12206	0.12206
a9a	3.2802	20.8732	18.2233	0.32428	0.32428	0.32428
liver-disorders	0.0077	0.0086	0.0086	0.65047	0.65047	0.65047
sonar	0.0126	0.0193	0.0314	0.47238	0.47238	0.47238
ijcnn1	0.8214	1.2669	3.2924	0.18461	0.18461	0.18461
svmguide3	0.0301	0.0324	0.1065	0.50362	0.50362	0.50362
gisette	23.7832	2.8383	284.7169	0.00013	0.00015	0.00012
real-sim	6.1680	3.8144	19.2361	0.14125	0.14125	0.14126
mushrooms	0.1090	0.1141	0.2377	0.00123	0.01014	0.00123
covtype.binary	0.2859	9767.2581	7.9744	0.00000	0.51367	0.00000
rcv1.binary	1.4968	0.9301	7.7141	0.19252	0.19252	0.19252
colon-cancer	0.0580	0.0214	0.1908	0.20047	0.20047	0.20047
leukaemia	0.1671	0.0638	0.6760	0.17995	0.20112	0.17995
duke-breast-cancer	0.1140	0.0621	0.6071	0.19326	0.19326	0.19326
news20	6.9871	13.91598	119.2786	0.32348	0.32348	0.32348
avazu-app.tr	22,433.9274	max time	max time	0.30009	0.30011	0.30071
HIGGS	789.9433	11641.3749	5022.4552	0.63771	0.63771	0.63771
url_combined	14,845.3728	max time	max time	0.02624	0.01812	0.03203
SUSY	467.3623	15373.2348	2744.7625	0.46300	0.46300	0.46300

on every dataset, whereas LIBLINEAR and ASA-CG fail on (the same) two datasets, namely *avazu-app.tr* and *url_combined*.

Next, we investigate the empirical local convergence of the iterates produced by FaRSA within the context predicted by Lemma 3.2(ii) and Theorem 3.8. In Table 3, we present the evolution of $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$ over the final 5 iterations. The column titled 'final' gives the values for $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$ at the final iterate, the column titled 'final-1' gives the values for $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$ at the penultimate iterate, and so on.

In Table 3 we can observe for the majority of data sets (22 out of 30) that the vector $\beta(x_k)$ is eventually equal to zero as predicted by Lemma 3.2(ii). At the point that $\beta(x_k)$ becomes zero, it is also the case that the value for $\|\phi(x_k)\|$ starts decreasing at a superlinear rate for a majority of the data sets (18/22), which is expected because of the superlinear convergence of $\{x_k\}$ as predicted by Theorem 3.8. For the other 4 datasets we can see that FaRSA converges on *covtype.binary* in a single iteration, and exhibits linear convergence for the data sets *skin-nonskin*, *w8a*, *mushrooms*.

Table 3. The values of $\|\beta(x_k)\|$ and $\|\phi(x_k)\|$ for the last 5 iterations.

Dataset	Measure	final-4	final-3	final-2	final-1	final
fourclass	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	1.49e+01	2.01e+00	1.18e−01	5.97e−04	1.79e−08
svmguide1	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	5.57e−01	1.03e−01	1.06e−01	3.47e−03	2.71e−05
cod-rna	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	9.16e−04	3.04e−03	5.66e−03	1.26e−03	4.88e−06
breast-cancer	$\|\beta(x_k)\|$	1.87e−01	5.51e−02	5.46e−02	1.38e−02	1.38e−02
	$\|\phi(x_k)\|$	9.04e−01	1.50e+02	2.55e−01	2.69e+01	7.96e−02
australian	$\|\beta(x_k)\|$	2.12e−03	1.69e−03	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	2.35e−03	1.71e−04	1.45e−03	1.20e−04	4.91e−07
skin-nonskin	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	2.08e−03	8.73e−04	3.63e−04	1.49e−04	5.61e−05
splice	$\|\beta(x_k)\|$	1.00e−03	1.05e−03	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	1.82e−03	1.33e−04	3.89e−04	2.57e−05	2.22e−09
heart	$\|\beta(x_k)\|$	2.33e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	2.16e−02	3.11e−02	5.80e−03	2.61e−04	5.40e−07
german.numer	$\|\beta(x_k)\|$	9.79e−04	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	7.62e−04	3.88e−03	5.37e−04	4.89e−05	2.20e−07
diabetes	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	2.94e−01	1.93e−02	1.03e−02	9.43e−05	1.83e−07
madelon	$\|\beta(x_k)\|$	6.13e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	6.92e−01	2.22e−02	2.10e−03	2.24e−05	1.46e−08
w8a	$\|\beta(x_k)\|$	3.69e−06	2.74e−06	1.73e−06	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	1.80e−05	1.14e−05	1.49e−06	4.09e−06	4.10e−07
a9a	$\|\beta(x_k)\|$	0.00e+00	5.61e−05	5.18e−06	0.00e+00	5.90e−08
	$\|\phi(x_k)\|$	7.76e−05	2.18e−04	1.08e−05	1.05e−06	1.02e−07
liver-disorder	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	2.40e+00	2.28e−01	1.38e−02	2.46e−05	2.41e−11
sonar	$\|\beta(x_k)\|$	7.16e−06	8.32e−04	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	1.05e−03	9.23e−03	6.90e−04	5.56e−05	6.28e−08
ijcnn1	$\|\beta(x_k)\|$	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	4.53e−03	1.35e−03	1.30e−04	1.36e−06	1.50e−10
svmguide3	$\|\beta(x_k)\|$	9.85e−05	4.84e−05	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	6.17e−04	2.78e−05	8.45e−05	4.85e−06	6.67e−09
gisette	$\|\beta(x_k)\|$	7.40e−03	7.77e−03	8.28e−03	1.31e−02	2.65e−05
	$\|\phi(x_k)\|$	1.13e−02	1.13e−02	1.16e−02	1.51e−02	1.14e−03
real-sim	$\|\beta(x_k)\|$	5.84e−06	5.49e−06	0.00e+00	2.00e−09	1.62e−08
	$\|\phi(x_k)\|$	6.77e−06	3.75e−06	6.52e−06	6.23e−06	5.97e−07
mushrooms	$\|\beta(x_k)\|$	4.76e−06	0.00e+00	4.53e−06	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	4.51e−05	4.53e−05	9.51e−06	8.98e−06	3.62e−07
covtype.binary	$\|\beta(x_k)\|$	–	–	–	2.15e+03	0.00e+00
	$\|\phi(x_k)\|$	–	–	–	0.00e+00	5.42e−06
rcv1.binary	$\|\beta(x_k)\|$	1.46e−05	0.00e+00	1.92e−06	4.71e−07	5.28e−07
	$\|\phi(x_k)\|$	1.31e−05	1.92e−05	1.74e−05	2.14e−06	2.04e−07
colon-cancer	$\|\beta(x_k)\|$	3.93e−04	6.02e−05	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	6.03e−05	8.38e−04	8.12e−05	7.61e−06	6.05e−09
leukaemia	$\|\beta(x_k)\|$	5.26e−05	4.08e−05	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	5.61e−05	3.16e−06	1.75e−04	1.32e−05	3.20e−08
duke-breast-cancer	$\|\beta(x_k)\|$	4.62e−04	4.24e−04	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	5.80e−04	1.60e−05	7.44e−04	3.60e−05	1.84e−07
news20	$\|\beta(x_k)\|$	2.33e−05	1.35e−08	2.24e−07	1.84e−07	1.60e−07
	$\|\phi(x_k)\|$	6.41e−06	2.34e−05	2.15e−05	1.78e−06	1.49e−07
avazu-app.tr	$\|\beta(x_k)\|$	1.01e−06	7.99e−08	2.41e−06	1.24e−06	8.60e−08
	$\|\phi(x_k)\|$	4.83e−07	1.01e−06	3.72e−06	4.60e−06	4.16e−07
HIGGS	$\|\beta(x_k)\|$	5.62e−04	5.58e−04	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	6.22e−04	6.13e−05	1.12e−04	9.03e−06	6.90e−09
url_combined	$\|\beta(x_k)\|$	6.75e−07	1.06e−06	8.41e−07	1.32e−05	6.21e−07
	$\|\phi(x_k)\|$	1.77e−06	2.85e−06	1.91e−06	1.55e−05	1.46e−06
SUSY	$\|\beta(x_k)\|$	4.99e−05	3.47e−05	0.00e+00	0.00e+00	0.00e+00
	$\|\phi(x_k)\|$	2.17e−04	6.77e−06	1.26e−04	7.68e−06	5.04e−08

This led us to use FaRSA on these three datasets with a tighter tolerance of 10^{-12} in order to better observe the asymptotic convergence; a superlinear rate of convergence of the iterates was then observed. A similar test was performed with the tolerance 10^{-12} on the 8 data sets for which $\beta(x_k)$ was not equal to zero at the final iterate in Table 3. The result of this test showed that eventually $\beta(x_k)$ was equal to zero, again as predicted by Lemma 3.2(ii), for all the datasets except *avazu-app.tr* and *url_combined*, which had final $\beta(x_k)$ values on the order of 10^{-14} and 10^{-7}, respectively.

5. Conclusions and final comments

In this paper, we proved a superlinear convergence result (see Theorem 3.8) for the solver FaRSA (see Algorithm 1) that was introduced with a Matlab implementation in [6] for solving ℓ_1-regularized convex optimization problems. The general framework of FaRSA involves an iterative procedure for identifying the optimization variables that are zero at a minimizer. To accelerate the convergence of the iterates, FaRSA integrates subspace minimization calculations based on the CG method. The decision of when and how to update the subspaces being probed is based on easily computable quantities that aim to quantify the potential gain of additional optimization over the current subspace and the potential gain of choosing a new subspace.

Crucial to the efficiency of FaRSA is a reliable strategy for determining when to terminate the CG algorithm when used to compute a search direction in the subspace. For this purpose, we introduced the set of CG termination conditions (16). The numerical results that we obtained on a subset of the LIBSVM binary-classification datasets for minimizing an ℓ_1-regularized logistic function showed that our new implementation of FaRSA (now written in C) usually requires less computing time when compared to the state-of-the-art solvers LIBLINEAR and ASA-CG, and is more reliable. It should also be mentioned that FaRSA has a greater memory requirement than the other methods primarily because of the memory needed for our reduced space ϕ-step calculations.

Finally, some additional discussion on Assumption 1.1 is appropriate. In particular, the local convergence analysis that we present for FaRSA requires f to be strongly convex. Interestingly, Yue *et al.* [17] also established local superlinear convergence of a regularized Newton method for the more general class of convex functions. This was accomplished by choosing the size of the regularization to be proportional to the distance to the solution set; similar ideas for general nonlinear programming can also be found in [9,10]. We suspect that similar ideas could be incorporated into the FaRSA framework and would allow us to prove a superlinear convergence result without having to make a strong convexity assumption. We leave this claim as an open conjecture.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the U.S. National Science Foundation [CCF-1618717] and the U.S. Department of Energy [DE-SC0010615].

ORCID

Frank E. Curtis http://orcid.org/0000-0001-7214-9187
Daniel P. Robinson http://orcid.org/0000-0003-0251-4227

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*, ACM, New York, NY, 2007, pp. 33–40. Available at https://dl.acm.org/citation.cfm?id = 1273501.

[2] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci. 2(1) (2009), pp. 183–202.

[3] R.H. Byrd, G.M. Chin, J. Nocedal, and F. Oztoprak, *A family of second-order methods for convex ℓ_1-regularized optimization*, Math. Program. 159(1-2) (2016), pp. 435–467.

[4] R.H. Byrd, J. Nocedal and F. Oztoprak, *An inexact successive quadratic approximation method for ℓ_1-regularized optimization*, Math. Program. 157(2) (2016), pp. 375–396.

[5] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intel. Syst. Technol. 2 (2011), pp. 27:1–27:27. Available at http://www.csie.ntu.edu.tw/ ~ cjlin/libsvm.

[6] T. Chen, F.E. Curtis and D.P. Robinson, *A reduced-space algorithm for minimizing ℓ_1-regularized convex functions*, SIAM J. Optim. 27 (2016), pp. 1583–1610.

[7] Dembo R.S., Eisenstat S.C. and T. Steihaug, *Inexact newton methods*, SIAM J. Numer. Anal. 19(2) (1982), pp. 400–408.

[8] D.L. Donoho, *De-noising by soft-thresholding*, IEEE Trans. Inf. Theory 41(3) (1995), pp. 613–627.

[9] P.E. Gill, V. Kungurtsev and D.P. Robinson, *A stabilized SQP method: Global convergence*, IMA J. Numer. Anal. 37(1) (2017), pp. 407–443.

[10] P.E. Gill, V. Kungurtsev and D.P. Robinson, *A stabilized SQP method: Superlinear convergence*, Math. Program. 163 (2017), pp. 1–42.

[11] W.W. Hager and H. Zhang, *A new active set algorithm for box constrained optimization*, SIAM J. Optim. 17(2) (2006), pp. 526–557.

[12] C.-J. Hsieh, I.S. Dhillon, P.K. Ravikumar and M.A. Sustik, *Sparse inverse covariance matrix estimation using quadratic approximation*, in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, eds., Curran Associates, Inc., 2011, pp. 2330–2338. Available at http://papers.nips.cc/paper/4266-sparse-inverse-covariance-matrix-estimation-using-quadratic-approximation.pdf.

[13] N. Keskar, J. Nocedal, F. Öztoprak and A. Waechter, *A second-order method for convex ℓ_1-regularized optimization with active-set prediction*, Optim. Method Softw. 31(3) (2016), pp. 605–621.

[14] K. Scheinberg and X. Tang, *Practical inexact proximal quasi-newton method with global complexity analysis*, Math. Program. 160(1/2) (2016), pp. 495–529.

[15] S.J. Wright, R.D. Nowak and M.A.T. Figueiredo, *Sparse reconstruction by separable approximation*, IEEE Trans. Signal Process. 57(7) (2009), pp. 2479–2493.

[16] G.-X. Yuan, C.-H. Ho and C.-J. Lin, *An improved GLMNET for ℓ_1-regularized logistic regression*, J. Mach. Learn. Res. 13(1) (2012), pp. 1999–2030.

[17] M.-C. Yue, Z. Zhou and A.M.-C. So, *Inexact regularized proximal Newton method: provable convergence guarantees for non-smooth convex minimization without strong convexity*, preprint (2016). Available at arXiv:1605.07522.