Department of Computer Science, University of British Columbia
Technical Report TR-2008-09, June 2008 (revised July 31, 2008)

GROUP SPARSITY VIA LINEAR-TIME PROJECTION

EWOUT VAN DEN BERG, MARK SCHMIDT, MICHAEL P. FRIEDLANDER, AND KEVIN MURPHY*

Abstract. We present an efficient spectral projected-gradient algorithm for optimization subject to a group ℓ_1-norm constraint. Our approach is based on a novel linear-time algorithm for Euclidean projection onto the ℓ_1- and group ℓ_1-norm constraints. Numerical experiments on large data sets suggest that the proposed method is substantially more efficient and scalable than existing methods.

1. Introduction. Parameter estimation with sparsity-promoting regularization is a topic of substantial interest to the machine learning community. Perhaps the most successful approach for promoting sparsity in the parameter vector is ℓ_1-regularization, which is the cornerstone of the widely-used least-absolute shrinkage and selection (LASSO) and basis pursuit denoising (BPDN) models [8, 22].

In the LASSO approach, the aim is to minimize the least-squares objective of a linear regression model subject to a bound on the ℓ_1-norm of the coefficients; this corresponds to a Gaussian response likelihood with independent Laplace priors on the parameters. More generally, given a (possibly nonconvex) loss-function $f(x)$ and a positive regularization parameter τ, the nonlinear LASSO problem is given by

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad \|x\|_1 \leq \tau. \tag{1.1}$$

The ℓ_1-norm constraint encourages sparsity in x for sufficiently small τ. This type of constraint has appealing properties regarding variable selection consistency [25], sample complexity [18], and generalization error [15]. In the emerging field of compressed sensing, ℓ_1-regularization can be used to exactly recover signals from sparsely sampled data [6, 11]. Because of these appealing properties, there is substantial interest in developing large-scale solvers for ℓ_1-regularized optimization problems; some recent examples of large-scale ℓ_1-solvers include [3, 13].

The useful properties of ℓ_1-regularization do not immediately carry over to problems where there does not exist a direct mapping between model variables and optimization parameters. An important example is problems with complex-valued variables, where penalizing real and imaginary components separately fails to recognize the pairing of real and imaginary optimization parameters to the same model variables. In contrast, the ℓ_1-norm for complex-valued vectors recognizes the explicit pairing between real and imaginary components. Fig. 1.1 shows two complex-valued examples that contrast penalizing components with penalizing groups.

In this paper, we consider the general *group* ℓ_1-norm regularization problem, where ℓ_1-regularization is applied to the norms of subsets of variables. Let the n-vector x be partitioned into g disjoint groups denoted by σ_i, $i = 1, \ldots, g$. We define the $(1, p)$-group norm of x as

$$\|x\|_{1,p} := \sum_{i=1}^{g} \|x_{\sigma_i}\|_p, \quad \text{for} \quad p \geq 1, \tag{1.2}$$

*Department of Computer Science, University of British Columbia, Vancouver V6T 1Z4, B.C., Canada ({ewout78,schmidtm,mpf,murphyk}@cs.ubc.ca). Van den Berg is corresponding author.
This work was supported by the NSERC Collaborative Research and Development Grant 334810-05. Submitted to NIPS on June 6, 2008.

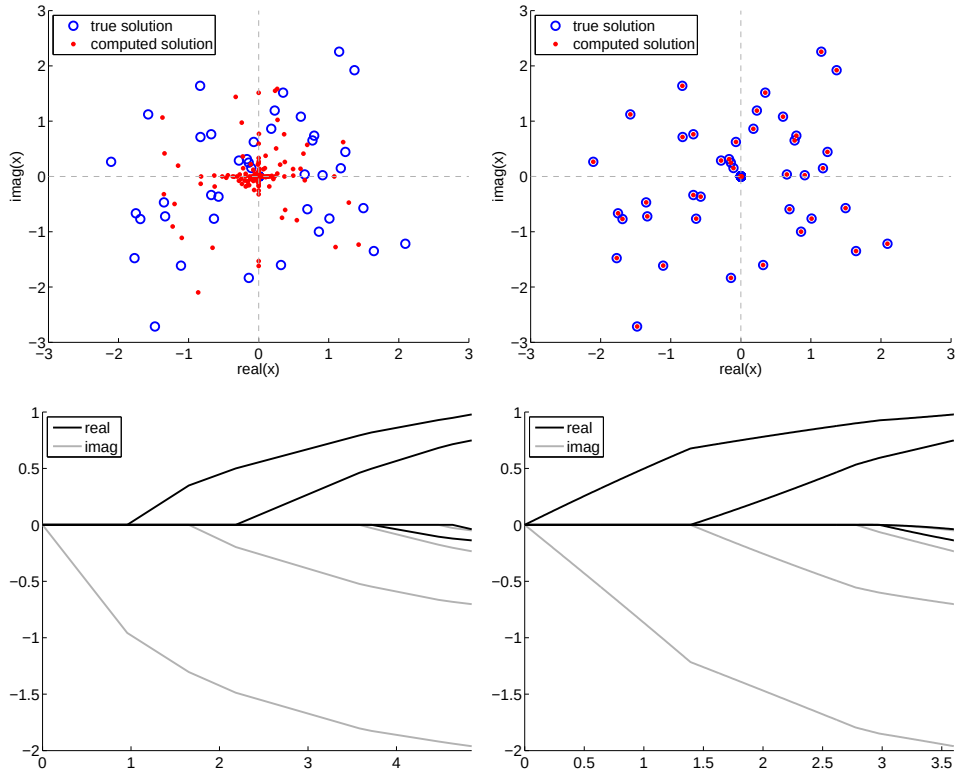

Fig. 1.1: Top: coefficients of linear model estimated with ℓ_1-norm penalization of (left) real and imaginary components, and (right) complex modulus using group ℓ_1. Bottom: regularization paths for a linear model with four complex variables using (left) ℓ_1-norm penalization of the real and imaginary components, and (right) group ℓ_1-norm penalization.

where x_{σ_i} denotes the subvector of x indexed by σ_i.

We focus on the nonlinear group Lasso problem [24], where the ℓ_1-norm in the nonlinear Lasso problem (1.1) is replaced with the more general $(1,2)$-group norm:

$$\underset{x}{\text{minimize}} \quad f(x) \quad \text{subject to} \quad \|x\|_{1,2} \equiv \sum_{i=1}^{g} \|x_{\sigma_i}\|_2 \leq \tau. \tag{1.3}$$

With a suitable choice of the index sets σ_i, this reduces to the usual ℓ_1-norm of complex vectors. The constraint on the group norm $\|x\|_{1,2}$ leads to sparsity in terms of groups. This has led to a variety of applications. For example, Yuan and Lin [24] use the $(1,2)$-group norm to achieve sparsity in terms of categorical attributes encoded as a set of indicator variables. Grouped variables also arise in multi-response linear regression and multinomial logistic regression, where each input feature has one parameter associated with each target vector [23]. Natural variable groupings are also present in linear models that are augmented with higher-order interactions [17], and in multiple kernel learning [1]. Finally, group sparsity has been used to learn the graph structure of multi-class Markov random fields and conditional random fields [16, 20].

In this paper, we present a large-scale algorithm for solving (1.3). The solver is based on the spectral projected-gradient (SPG) algorithm [4], which is also the basis for several state-of-the-art solvers for large-scale ℓ_1-regularized problems [13, 3]. The method crucially depends on the ability to project iterates onto the $(1,2)$-group norm, i.e., the following projection operator must be applied at each iteration:

$$P_\tau^{1,2}(c) := \Big\{ \arg\min_x \ \|c - x\|_2 \ \text{ subject to } \ \|x\|_{1,2} \leq \tau \Big\}. \tag{1.4}$$

Note that when $\sigma_i = \{i\}$ and $g = n$, the $(1,2)$-group projection reduces to the projection onto the (usual) ℓ_1-ball with radius τ; we denote this projection by $P_\tau^1(c)$.

We develop a linear-time algorithm for computing (1.4). We begin our development by describing a linear-time algorithm for computing $P_\tau^1(c)$ (§4.2) and then generalize this algorithm to $(1,2)$-group projection (§4.3). Experiments on data with a large number of parameters demonstrate the ability of these new algorithms to scale to large problems.

2. Previous work. The group ℓ_1-norm regularization problem is non-differentiable whenever the norm of a group is zero. Various approaches have been proposed to address this non-differentiability. Meier el al. [17] address the non-differentiability by using non-smooth optimality conditions and block coordinate descent. Unfortunately, this strategy scales poorly with the number of non-zero groups. Smoothing approximations are used in [20], but these approximations become highly ill-conditioned near the points of non-differentiability, substantially slowing convergence. Simila and Tikka [21] formulate the problem as a differentiable objective with second-order cone constraints, and propose a primal-dual interior-point method; however, this method scales poorly to large problems because large linear systems (that involve the Hessian matrix) must be solved at each iteration. Existing path-following methods also require Hessian information and consequently do not scale well to large problems either [19, 24]. In contrast, the projected-gradient algorithm described by [14] has low-memory requirements and scales well, but it typically exhibits slow convergence, and requires an expensive iteration in order to solve (1.4). Other authors [20, 23] have noted the difficulty in solving (1.3), and have instead used the $(1,\infty)$-group norm as the regularizing function.

3. Spectral projected gradient. The nonlinear group Lasso problem (1.3) with convex f can be solved using projected-gradient (PG) methods. For nonconvex loss-functions, PG methods can be used to find a local optimum.

Given a current iterate $x^{(k)}$, PG methods search along the projected-gradient path

$$x(\alpha) = P\big(x^{(k)} - \alpha \nabla f(x^{(k)})\big), \tag{3.1}$$

where P gives the projection onto the convex feasible set; for the particular case (1.3), the projection is described by (1.4). The step length $\alpha \in (0, \alpha_{\max}]$ is determined with a backtracking line search along the projection arc to ensure sufficient descent of the objective function (e.g., using an Armijo condition). After finding an appropriate α, the next iterate is defined by $x^{(k+1)} := x(\alpha)$. This scheme is repeated until a suitable stopping criteria is satisfied.

The PG algorithm above is often implemented with $\alpha_{\max} = 1$, and the resulting method is essentially steepest descent, which is known to have poor convergence rates. The SPG method relaxes the PG method in two ways. First, the iterates do not have to monotonically decrease the objective function; instead, sufficient decrease is only

required relative to a fixed number of recent iterations. Second, the parameter $\alpha_{\max}$ is chosen as the two-point approximation to the quasi-Newton secant equations:

$$\alpha_{\max} = \frac{s^{(k)T} s^{(k)}}{s^{(k)T} y^{(k)}}, \quad \text{with } s^{(k)} := x^{(k)} - x^{(k-1)} \text{ and } y^{(k)} := \nabla f(x^{(k)}) - \nabla f(x^{(k-1)});$$

see [2]. Even though the SPG algorithm is simple to implement and has low-memory requirements, it is competitive with more elaborate and extensively tuned optimization algorithms [4].

It is important to note that although SPG is oblivious to the shape of the feasible set, its efficiency heavily depends on the efficiency of computing the projection in (3.1). In some cases, calculating the projection can have a similar cost to solving the original problem. Hence, the SPG method is mainly used for problems where the projection can be computed efficiently; an important case is the convex set defined by bound-constraints on the variables. For example, Figueiredo et al. [13] use the SPG method to efficiently solve a bound-constrained formulation of the LASSO problem.

4. Projection algorithms. In this section, we develop a linear-time algorithm for group projection. We motivate our linear-time algorithm by describing an $\mathcal{O}(n \log n)$ algorithm for projecting onto the (usual) ℓ_1-ball (§4.1), and then show how to reduce this to linear time (§4.2). We then describe how to leverage this algorithm to compute the group projection (1.4) in linear time (§4.3).

4.1. ℓ_1-norm projection in $\mathcal{O}(\mathbf{n} \log \mathbf{n})$ time. In the special case of singleton groups, (1.4) reduces to projection onto the ℓ_1-ball

$$P_\tau^1 := \Big\{ \arg\min_x \ \|c - x\|_2 \ \text{ subject to } \ \|x\|_1 \le \tau \Big\}. \tag{4.1}$$

Ignoring the case $\|c\|_1 \le \tau$ which has the trivial solution $x = c$, there exists for each τ a λ such that

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|c - x\|_2^2 + \lambda \|x\|_1, \tag{4.2}$$

has the same solution as (4.1). The solution of this penalized formulation is given directly by applying (componentwise) the soft-thresholding operator [7, §III]

$$S_\lambda(c) = \operatorname{sgn}(c) \cdot \max\{0, |c| - \lambda\}. \tag{4.3}$$

The signum function $\operatorname{sgn}(c) := c/\|c\|$ is also defined componentwise, and by convention the elements of $\operatorname{sgn}(0)$ can be chosen arbitrarily between -1 and 1. Thus, if we can find the λ that makes (4.1) and (4.2) equivalent, then both can be solved using (4.3). This comes down to finding λ such that $\|S_\lambda(c)\|_1 = \tau$. The remainder of this section is devoted to developing a method for finding such λ.

To simplify the discussion, let a_i, $i = 1, \ldots, n$, be the absolute values of c in decreasing order. It is convenient to define $a_{n+1} = 0$ and to define the function

$$\phi(\lambda) := \|S_\lambda(c)\|_1. \tag{4.4}$$

It can be verified from (4.3) that $\phi(\lambda)$ is monotonically decreasing in λ from $\phi(a_{n+1}) = \phi(0) = \|c\|_1$ to $\phi(a_1) = 0$. Therefore, there exists an integer k such that

$$\phi(a_k) \le \tau < \phi(a_{k+1}). \tag{4.5}$$

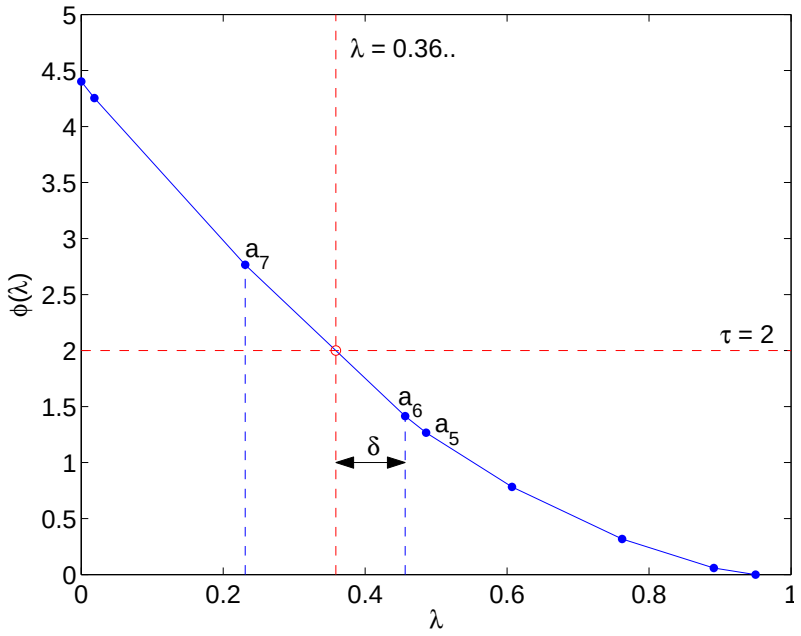

Fig. 4.1: Plots of $\phi(\lambda)$ for a random c of length 8 showing the relationship between τ, λ, and δ. The points indicate locations where $\lambda = a_i$.

Suppose that k is given. Then it remains to find $\delta \geq 0$ such that $\phi(a_k - \delta) = \tau$. It follows from the definition of ϕ that

$$\phi(a_i) = \sum_{j=1}^{n} \max\{0, a_j - a_i\} = \sum_{j=1}^{i} (a_j - a_i) = \Big(\sum_{j=1}^{i} a_j\Big) - i \cdot a_i. \tag{4.6}$$

For a δ such that $0 \leq \delta \leq a_i - a_{i+1}$, it similarly holds that

$$\phi(a_i - \delta) = \sum_{j=1}^{n} \max\{0, a_j - (a_i - \delta)\} = \sum_{j=1}^{i} (a_j - a_i + \delta) = \delta i + \phi(a_i). \tag{4.7}$$

Given k, we find λ by solving $\phi(a_k - \delta) = \tau$ for δ, which by (4.7) gives $\delta = (\tau - \phi(a_k))/k$, and hence $\lambda = a_k - \delta$. Fig. 4.1 graphically illustrates the relationship between τ, λ, k, and δ.

To find the projection x we can thus proceed as follows: (i) sort the absolute values of c to get the vector a; (ii) find the value of k such that (4.5) is satisfied, (iii) compute δ and λ as given above, (iv) compute the final projection $x := S_\lambda(c)$.

This approach was independently developed by Candès and Romberg [5] and Daubechies et al [10]. Because of the sorting step, the overall time complexity of these algorithms is $\mathcal{O}(n \log n)$; all subsequent steps can be implemented in $\mathcal{O}(n)$. We conclude this section by noting that the signs of c are relevant only in determining the final solution $S_\lambda(c)$. This observation forms the basis of group projection and allows the above algorithm to assume, without loss of generality, that $c \geq 0$.

4.2. ℓ_1-norm projection in linear time. In order to derive an $\mathcal{O}(n)$ projection algorithm[1] it suffices to find an $\mathcal{O}(n)$ algorithm for the first two steps used to determine the k satisfying (4.5). The proposed algorithm takes advantage of two crucial ideas. First, given i such that $\phi(a_i)$ does not satisfy (4.5), then $\phi(a_i)$ tells us whether $k < i$ or $k > i$. Second, after we compute $\phi(a_i)$ we can compute $\phi(a_j)$ for any j more economically by reusing intermediate results. To simplify the presentation, we will assume that c is positive and that there are no ties between non-zero elements (removing these restrictions is straightforward). Further, as a notational convenience we continue to denote by a_i the ith largest value of c, although the algorithm never explictly constructs this vector.

We first consider evaluating $\phi(a_i)$ given some a_i which we shall call the *pivot*. By (4.6) we can evaluate $\phi(a_i)$ by first partitioning c into $p_{\text{low}} := \{j : c_j > a_i\}$ and $p_{\text{high}} := \{j : c_j < a_i\}$, computing $s := a_i + \sum_{j \in p_{\text{low}}} c_j$, and finally setting $\phi(a_i) = s - i \cdot a_i$. This requires $\mathcal{O}(n)$ time.

By considering two cases, we now show how to efficiently compute $\phi(a_m)$ for some $m \neq i$ after computing $\phi(a_i)$. If $m < i$, then $a_m > a_i$, and all elements in p_{high} are irrelevant allowing us to compute $\phi(a_m)$ based only on p_{low}. This reduces the evaluation cost to $\mathcal{O}(i)$. For $m > i$, we first use p_{high} to define the new set $p_{low} = \{j \in p_{high} : c_j > a_m\}$. We then update $s := s + a_m + \sum_{j \in p_{\text{low}}} c_j$ and use (4.6) to find $\phi(a_m) = s - m \cdot a_m$. In this case, the computation complexity of evaluating $\phi(a_m)$ is $\mathcal{O}(n - i)$.

Considering both cases, the largest reduction in computing $\phi(a_m)$, given $\phi(a_i)$, is achieved if i is chosen such that a_i is the (upper) median of c. Furthermore, the median value a_i of the n-vector c can be computed in $\mathcal{O}(n)$ using the *median-of-medians* algorithm [9].

The last property we use is that $\phi(a_i)$ is non-decreasing in i. Therefore, given some a_i that does not satisfy (4.5), we can reduce our search for k to one of p_{low} or p_{high}. Each iteration of the algorithm is thus summarized as follows: (i) set the pivot to the upper-median of the remaining elements, (ii) partition around the pivot, (iii) test condition (4.5), and (iv) reduce the search to one of the two partitions. Algorithm 1 explicitly outlines the procedure. The number of elements remaining at each iteration is at most $\lfloor n/2^k \rfloor$, while the iteration cost when r elements remain is in $\mathcal{O}(r)$. If ξ is the constant associated with the iteration cost, we thus have the runtime bounded above by

$$\sum_{k=0}^{\infty} \xi \frac{n}{2^k} = \xi n \sum_{k=0}^{\infty} \frac{1}{2^k} = 2\xi n = \mathcal{O}(n).$$

Unfortunately, the constant factor associated with the median-of-medians algorithm is high [9]. In our implementation, we choose the pivot as a random remaining value rather than the upper-median. This no longer guarantees a worst-case linear runtime, but achieves an *expected* linear runtime [9]. In particular, the runtime is linear with respect to the expectation over possible permutations of the rank-orderings of the vector (the actual distribution of the data is not relevant).

4.3. Group ℓ_1-norm projection in linear time. In this section, we detail how the *group* ℓ_1-norm projection (1.4) can be computed by projecting the individual group ℓ_2

[1] A similar algorithm was independently proposed recently by Duchi et al. [12]

Algorithm 1: Linear-time projection of n-vector c onto ℓ_1-ball of radius τ

```
if ||c||_1 ≤ τ then return c;
Initialize: p := c,  s := 0;
while 1 do
    k := ⌈length(p)/2⌉;
    a_k := upper-median(p);
    p_high := find(p < a_k);
    p_low := find(p > a_k);
    s_low := ‖p_low‖_1 + a_k;
    φ(a_k) := s + s_low − k · a_k ;
    if φ(a_k) < τ then
        p := p_low;
    else
        a_{k+1} := max(p_high);
        φ(a_{k+1}) := s + s_low − k a_{k+1};
        if φ(a_{k+1}) < τ then break
        p := p_high;
        s := s + s_low;
λ := a_k − (τ − φ(a_k))/k;
return max(c − λ, 0);
```

norms onto the ℓ_1-norm ball of radius τ. By defining the *group signum* as

$$\operatorname{sgn}(x_\sigma) = \frac{x_\sigma}{\|x_\sigma\|_2},$$

with the entries of sgn(0) arbitrarily between -1 and 1, we have the following result:

THEOREM 4.1. *The solution of the projection problem* (1.4) *is given by*

$$x_{\sigma_i} = \operatorname{sgn}(c_{\sigma_i}) \cdot w_i, \qquad \text{for } i = 1, \ldots, g, \tag{4.8}$$

where $v_i = \|c_{\sigma_i}\|_2$ *and* $w = P_\tau^1(v)$ *is the projection of* v *onto the (usual)* ℓ_1*-ball with radius* τ*.*

The proof of the theorem is given in Appendix A. The proof is based on a version of (4.2) where the ℓ_1-norm is replaced by the $(1,2)$-group norm. It is possible to show that the solution of this problem can be obtained by applying the specially defined soft-thresholding operator

$$x_{\sigma_i} = \operatorname{sgn}(c_{\sigma_i}) \cdot \max\left\{0,\ \|c_{\sigma_i}\|_2 - \lambda\right\}, \quad \text{for } i = 1, \ldots, g. \tag{4.9}$$

We then show that the λ in the penalty formulation of (1.4) depends only on $v_i = \|c_{\sigma_i}\|_2$, and that (4.9) can be computed using $w = P_\tau^1(v)$ by applying (4.8).

With the projection algorithm given in §4.2, we can compute w in linear time. Because v and x can also be computed in linear time, it follows that the $(1,2)$-group projection can be computed in linear time.

5. Results. We compared the scalability of our randomized projection algorithm against two approaches that require sorting the input vector c: a low-level C implementation of quicksort representing the approach of [5, 10], and a more recent heap-based method proposed in [3] that only requires sorting in the worst case.

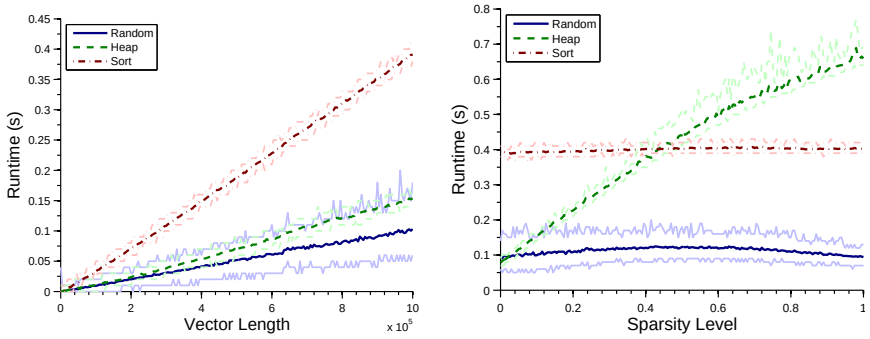

Fig. 5.1: Runtimes for different (a) vector lengths and (b) sparsity levels.

In Fig. 5.1(a) we plot the longest, shortest, and average projection time over one hundred trials against the length of c with τ chosen such that the number of non-zero entries in the solution is 10% of the length. Part (b) of the same figure shows how the projection time changes for a vector of fixed length 10^6, but with varying sparsity levels. Our randomized projection algorithm has larger deviations in runtime, but dominates sorting for any non-trivial problem size and dominates the heap-based method if the sparsity level is above approximately 10%.

We next assessed the performance of SPG in solving the group Lasso problem. We compared to all first-order (i.e., no Hessian information required) methods that we are aware of: the block coordinate descent (BCD) algorithm of [17], the PG algorithm [14], and the multi-quadric smoothing approximation [20]. BCD and the smoothing method use slightly different, yet equivalent, problem formulations and it is possible to convert between them. We applied the methods to a sparse recovery scenario where group-sparse solutions are sought for minimizing $f(x) = \frac{1}{2}\|b - Ax\|_2^2$, with a 9,600×65,536 matrix A consisting of randomly restricted rows of a discrete cosine transform matrix, and $b = Ax_0 + s$, where x_0 is a group-sparse vector with 512 groups of 128 entries and eight non-zero groups, and additive noise $s_i \sim \mathcal{N}(0, 10^{-3})$. Fig. 5.2(a) shows box plots of the required number of function evaluations for convergence over 50 problem instances. In all instances, BCD reached the imposed maximum of 5,000 function evaluation. Part (b) plots the number of function evaluations required as a function of τ for a single problem instance.

The second problem set we applied the solvers to was joint structure and parameter learning in a conditional random field with 4,950 groups (half of which are relevant) and 165,000 variables following the experimental set-up in [20]. Fig. 5.2(c) shows the value of the objective function against the number of function evaluations for each method. As in the case of a quadratic objective, the SPG method was the most efficient by a substantial margin.

Following the principle of reproducible research, we have made the software used in these experiments available on-line at: `http://www.cs.ubc.ca/~mpf/?n=Gsparse`.

6. Discussion. We have presented an efficient algorithm for solving large-scale non-linear group ℓ_1-norm regularization problems based on the SPG algorithm. We have shown that the complexity of computing the Euclidean projection on the ℓ_1-ball is $\mathcal{O}(n)$, and shown how efficient linear-time projection algorithms can be extended to

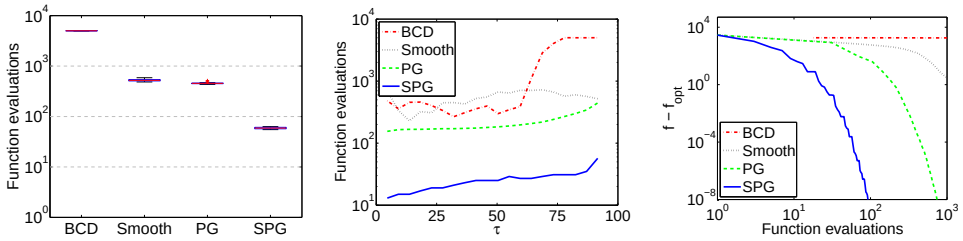

Fig. 5.2: Evaluation of group-ℓ_1 solvers for sparse recovery, (a) average number of function evaluations, (b) evaluations versus τ, and (c) CRF structure learning, showing function evaluations on log scale against accuracy of solution.

the case of grouped variables.

Although we did not explicitly discuss it, SPG is well-suited to the task of computing regularization paths by solving the group ℓ_1-norm problem for related values of τ. In particular, the number of iterations can be reduced by warm-starting the optimization for a fixed τ with the solution obtained with a nearby τ. Furthermore, because in our formulation the regularization parameter does not affect the objective (nor its gradient), the two-point quasi-Newton approximation remains valid after changing τ. Hence both the parameter *and* gradient vectors for a related value of τ can be used in warm-starting the SPG method.

Finally, note that we have exclusively considered the deterministic optimization scenario. However, group sparsity is also relevant in scenarios where stochastic approximation methods are employed. Because our algorithm allows linear-time projection, the cost of moving along the projected stochastic gradient direction to enforce group sparsity is only slightly more expensive than moving along the unconstrained stochastic gradient direction.

Appendix A. Group projection. In this section we prove Theorem 4.1, given in section 4.3.

Proof. Recall that the n-vector x is partitioned into g groups with pairwise disjoint index sets σ_i, $i = 1, \ldots, g$ whose union is $\{1, \ldots, n\}$. We represent x as an g-vector $\widetilde{x} = (\widetilde{x}_1, \widetilde{x}_2, \cdots, \widetilde{x}_g)$, where each $\widetilde{x}_k = (x_j)_{j \in \sigma_k}$ denotes the tuple formed by x_{σ_k}, the components of x belonging to group k. This is somewhat akin to complex numbers where each tuple consists of the real and imaginary part of the corresponding complex number, except that we allow for nonuniform tuple sizes. With this notation we extend the signum function as

$$\operatorname{sgn}(\widetilde{x})_k = \operatorname{sgn}(\widetilde{x}_k) = \frac{\widetilde{x}_k}{|\widetilde{x}_k|}, \quad \text{with} \quad |\widetilde{x}_k| = \Big(\sum_{j \in \sigma_k} x_j^2 \Big)^{1/2} = \|x_{\sigma_k}\|_2. \tag{A.1}$$

It can be verified that his extended signum function satisfies the usual properties that

$$\operatorname{sgn}(\alpha \widetilde{x}) = \operatorname{sgn}(\widetilde{x}) \quad \text{for all } \alpha > 0, \quad \text{and} \quad \|\operatorname{sgn}(\widetilde{x}_k)\|_2 \leq 1. \tag{A.2}$$

We adopt the convention that sgn(0) can be taken to be any vector satisfying the

second property. Finally, with the p-norm defined as

$$\|\widetilde{x}\|_p = \Big(\sum_{k=1}^{\ell} |\widetilde{x}_k|^p\Big)^{1/p} = \|x\|_{p,2}.$$

it can be shown that $\nabla \frac{1}{2}\|\widetilde{x}\|_2^2 = \widetilde{x}$, and $\nabla\|\widetilde{x}\|_1 = \operatorname{sgn}(\widetilde{x})$.

We now apply these results to derive the optimality conditions for the group projection leading to the extended soft-thresholding operator. Using $\|\widetilde{x}\|_2 = \|x\|_2$, and $\|\widetilde{x}\|_1 = \|x\|_{1,2}$ we can solve

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad \tfrac{1}{2}\|c-x\|_2^2 + \lambda\|x\|_{1,2} \quad \text{or} \quad \underset{\widetilde{x}}{\text{minimize}} \quad \tfrac{1}{2}\|\widetilde{c}-\widetilde{x}\|_2^2 + \lambda\|\widetilde{x}\|_1. \tag{A.3}$$

A vector $\widetilde{x}$ is a solution of this problem if and only if it satisfies

$$\nabla(\tfrac{1}{2}\|\widetilde{x}-\widetilde{c}\|_2^2 + \lambda\|\widetilde{x}\|_1) = \widetilde{x} - \widetilde{c} + \lambda\operatorname{sgn}(\widetilde{x}) = 0. \tag{A.4}$$

We claim that the $\widetilde{x}$ satisfying this condition, and hence giving the solution of (A.3), is given by

$$\widetilde{x}_k = S_\lambda(\widetilde{c})_k = S_\lambda(\widetilde{c}_k) = \begin{cases} \operatorname{sgn}(\widetilde{c}_k)(|\widetilde{c}_k| - \lambda) & \text{if } |\widetilde{c}_k| > \lambda; \\ 0 & \text{otherwise.} \end{cases} \tag{A.5}$$

To check this we separately consider the two cases $|\widetilde{c}| > \lambda$ and $|\widetilde{c}| \le \lambda$. In the case where $|\widetilde{c}_k| > \lambda$, substitute $\widetilde{x}_k = S_\lambda(\widetilde{c}_k)$ into (A.4) and use the first property in (A.2) to obtain:

$$\begin{aligned} \widetilde{x}_k - \widetilde{c}_k + \lambda\operatorname{sgn}(\widetilde{x}_k) &= \operatorname{sgn}(\widetilde{c}_k)(|\widetilde{c}_k| - \lambda) - \widetilde{c}_k + \lambda\operatorname{sgn}(\operatorname{sgn}(\widetilde{c}_k)(|\widetilde{c}_k| - \lambda)) \\ &= \operatorname{sgn}(\widetilde{c}_k)(|\widetilde{c}_k| - \lambda) - \widetilde{c}_k + \lambda\operatorname{sgn}(\widetilde{c}_k) \\ &= \operatorname{sgn}(\widetilde{c}_k)\cdot|\widetilde{c}_k| - \widetilde{c}_k = 0. \end{aligned}$$

In case $|\widetilde{c}_k| \le \lambda$, we substitute $\widetilde{x} = 0$, giving $\widetilde{c} = \lambda\operatorname{sgn}(0)$. But because $\operatorname{sgn}(0)$ is arbitrary, we can choose it as $(1/\lambda)\widetilde{c}_k$, which clearly satisfies the condition.

Assuming $\|\widetilde{c}\|_1 > \tau$, it remains to be shown how to find λ giving $\phi(|\widetilde{c}_k|) = \|S_\lambda(\widetilde{c}))_1\| = \tau$, based on $v_i = |\widetilde{c}_i|$. Defining $\mathcal{I}_k = \{j \mid |\widetilde{c}_j| > |\widetilde{c}_k|\} = \{j \mid v_j > v_k\}$, and noting that $|\operatorname{sgn}(\widetilde{c}_i)| = 1$ we have

$$\begin{aligned} \phi(|\widetilde{c}_k|) &= \sum_{i\in\mathcal{I}_k} |\operatorname{sgn}(\widetilde{c}_i)(|\widetilde{c}_i| - |\widetilde{c}_k|)| = \sum_{i\in\mathcal{I}_k} |(\operatorname{sgn}(\widetilde{c}_i)|\widetilde{c}_i|)| - \sum_{i\in\mathcal{I}_k} |(\operatorname{sgn}(\widetilde{c}_i)|\widetilde{c}_k|)| \\ &= \sum_{i\in\mathcal{I}_k} |\operatorname{sgn}(\widetilde{c}_i)|\cdot|\widetilde{c}_i| - \sum_{i\in\mathcal{I}_k} |\operatorname{sgn}(\widetilde{c}_i)|\cdot|\widetilde{c}_k| \\ &= \sum_{i\in\mathcal{I}_k} |\widetilde{c}_i| - |\mathcal{I}_k|\cdot|\widetilde{c}_k| = \sum_{i\in\mathcal{I}_k} v_i - |\mathcal{I}_k|\cdot v_k. \end{aligned}$$

But this is exactly the setting of equation (4.6). Similarly rewriting (4.7) completes the equivalence for finding λ. The last part of the theorem constructs x from v and $w = P_\tau(v)$. In group notation this step can be derived as

$$\begin{aligned} \widetilde{x}_i &= \operatorname{sgn}(\widetilde{c}_i)\cdot w_i \\ &= \operatorname{sgn}(\widetilde{c}_i)\cdot\operatorname{sgn}(v_i)\cdot\max\{0, v_i - \lambda\} \\ &= \operatorname{sgn}(\widetilde{c}_i)\cdot\max\{0, v_i - \lambda\} \\ &= \operatorname{sgn}(\widetilde{c}_i)\cdot\max\{0, |\widetilde{c}_i| - \lambda\}, \end{aligned}$$

where we used (A.1) and the fact that $\text{sgn}(|\widetilde{c}_i|) = 1$. This exactly coincides with (A.5), as required. □

REFERENCES

[1] F. R. Bach, *Consistency of the group lasso and multiple kernel learning*, 2008. To appear in *J. Mach. Learn. Res.*

[2] J. Barzilai and J. M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.

[3] E. van den Berg and M. P. Friedlander, *Probing the Pareto frontier for basis pursuit solutions*, Tech. Rep. TR-2008-01, Department of Computer Science, University of British Columbia, Vancouver, January 2008. To appear in *SIAM J. Sci. Comp.*

[4] E. G. Birgin, J. M. Martínez, and M. Raydan, *Nonmonotone spectral projected gradient methods on convex sets*, SIAM J. Optim., 10 (2000), pp. 1196–1211.

[5] E. J. Candès and J. Romberg, *Practical signal recovery from random projections*, in Computational Imaging III, Proc. SPIE Conf., vol. 5914, March 2005, pp. 76–86.

[6] E. J. Candès, J. Romberg, and T. Tao, *Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information*, IEEE Trans. Inform. Theory, 52 (2006), pp. 489–509.

[7] A. Chambolle, R. DeVore, N.-Y. Lee, and B. Lucier, *Nonlinear wavelet image processing: variational problems, compression, and noise removal through wavelet shrinkage*, IEEE Trans. Image Proc., 7 (1998), pp. 319–335.

[8] S. S. Chen, D. L. Donoho, and M. A. Saunders, *Atomic decomposition by basis pursuit*, SIAM J. Sci. Comput., 20 (1998), pp. 33–61.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachussetts, second ed., September 2001.

[10] I. Daubechies, M. Fornasier, and I. Loris, *Accelerated projected gradient method for linear inverse problems with sparsity constraints*, J. Fourier Anal. Appl., (2007). To appear.

[11] D. L. Donoho, *Compressed sensing*, IEEE Trans. Inform. Theory, 52 (2006), pp. 1289 – 1306.

[12] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra, *Efficient projections onto the l1-ball for learning in high dimensions*, International Conference on Machine Learning, (2008).

[13] M. Figueiredo, R. Nowak, and S. J. Wright, *Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems*, IEEE Trans. on Selected Topics in Sig. Proc., 1 (2007), pp. 586–597.

[14] Y. Kim, J. Kim, and Y. Kim, *Blockwise sparse regression*, Statistica Sinica, 16 (2006), pp. 375–390.

[15] B. Krishnapuram, L. Carin, M. Figueiredo, and A. Hartemink, *Learning sparse Bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds*, IEEE. Trans. Pattern. Anal. Mach. Intell., (2005).

[16] S.-I. Lee, V. Ganapathi, and D. Koller, *Efficient structure learning of Markov networks using L1-regularization*, Neural Information Processing Systems, (2006).

[17] L. Meier, S. van de Geer, and P. Buhlmann, *The group lasso for logistic regression*, J. R. Statist. Soc. B, 70 (2008), pp. 53–71.

[18] A. Ng, *Feature selection, L1 vs. L2 regularization, and rotational invariance*, International Conference on Machine Learning, (2004).

[19] M.-Y. Park and T. Hastie, *Regularization path algorithms for detecting gene interactions*, tech. rep., Department of Statistics, Stanford University, 2006.

[20] M. Schmidt, K. Murphy, G. Fung, and R. Rosales, *Structure learning in random fields for heart motion abnormality detection*, Inter. Conf. Comp. Vision Pattern Recog., (2008). To appear.

[21] T. Simila and J. Tikka, *Input selection and shrinkage in multiresponse linear regression*, Computational Statistics and Data Analysis, 52 (2007), pp. 406–422.

[22] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, J. R. Statist. Soc. B, 58 (1996), pp. 267–288.

[23] B. Turlach, W. Venables, and S. Wright, *Simultaneous variable selection*, Technometrics, 47 (2005), pp. 349–363.

[24] M. Yuan and Y. Lin, *Model selection and estimation in regression with grouped variables*, J. R. Statist. Soc. B, 68 (2006), pp. 49–67.

[25] P. Zhao and B. Yu, *On model selection consistency of Lasso*, J. Mach. Learn. Res., 7 (2007), pp. 2541–2567.