Fast projections onto $\ell_{1,q}$-norm balls for grouped feature selection

Suvrit Sra

MPI for Intellingent Systems
72076 Tübingen, Germany

Abstract. Joint sparsity is widely acknowledged as a powerful structural cue for performing feature selection in setups where variables are expected to demonstrate "grouped" behavior. Such grouped behavior is commonly modeled by Group-Lasso or Multitask Lasso-type problems, where feature selection is effected via $\ell_{1,q}$-mixed-norms. Several particular formulations for modeling groupwise sparsity have received substantial attention in the literature; and in some cases, efficient algorithms are also available. Surprisingly, for *constrained* formulations of fundamental importance (e.g., regression with an $\ell_{1,\infty}$-norm constraint), highly scalable methods seem to be missing. We address this deficiency by presenting a method based on spectral projected-gradient (SPG) that can tackle $\ell_{1,q}$-constrained convex regression problems. The most crucial component of our method is an algorithm for projecting onto $\ell_{1,q}$-norm balls. We present several numerical results which show that our methods attain up to 30X speedups on large $\ell_{1,\infty}$-multitask lasso problems. Even more dramatic are the gains for just the $\ell_{1,\infty}$-projection subproblem: we observe almost three orders of magnitude speedups compared against the currently standard method.

1 Introduction

Sparsity offers powerful structural information that enables recovering unknown, high-dimensional vectors robustly. Consequently, sparsity has been intensively studied in signal processing, machine learning, and statistics, where it has played a key role in numerous algorithms and applications. The associated literature has grown too large, and a summary will be futile, so we refer the reader to [2, 24, 30, 31] as starting points.

Typically sparsity constrained problems that arise in machine learning and statistics may be written as instances of the following general problem:

$$\min_{\boldsymbol{w}} \quad \mathcal{L}(\boldsymbol{w}) + \lambda R(\boldsymbol{w}), \tag{1}$$

where $\mathcal{L}$ is differentiable, convex loss-function, $\lambda > 0$ is a scalar, while R is a convex (possibly nonsmooth) regularizer that models sparsity. Alternatively, one can consider the following constrained formulation of (1):

$$\min_{\boldsymbol{w}} \quad \mathcal{L}(\boldsymbol{w}) \quad \text{s.t.} \quad R(\boldsymbol{w}) \leq \gamma, \tag{2}$$

for an appropriate scalar $\gamma > 0$. We focus on the latter formulation, especially because it is particularly amenable to a simple first-order optimization procedure. We note another benefit that can make formulation (2) attractive: several variants of gradient-projection [6] remain applicable even when $\mathcal{L}$ is not convex; mere differentiability suffices. Moreover, theoretical analysis of (2) is simpler because the constraint ensures that we are minimizing over a compact set.

Amongst the dizzying number of variants of (2) that have been studied the literature, we consider in this paper a particular family: *groupwise sparsity*. Here, the regularizer R selects (or ignores) entire groups of variables simultaneously, e.g. in multitask learning [12,13,19,26], or in Group-Lasso [3,36,37]. One practical way to induce groupwise sparsity is to let R be a mixed-norm, defined as follows (for a more general definition, see [39]). Let a vector $\boldsymbol{w} \in \mathbb{R}^d$ be partitioned into the subvectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_G$, where $\boldsymbol{w}_g \in \mathbb{R}^{d_g}$ for $1 \leq g \leq G$, and let $q \geq 1$. Then, the $\ell_{1,q}$ *mixed-norm* for $\boldsymbol{w}$ is given by

$$\|\boldsymbol{w}\|_{1,q} = \sum\nolimits_{g=1}^{G} \|\boldsymbol{w}_g\|_q. \tag{3}$$

The most common variants of (3) are $\|\cdot\|_{1,2}$ and $\|\cdot\|_{1,\infty}$; the former is often used in Group-Lasso [37], while the latter arises in compressed sensing [35] and multitask Lasso [19]. Less common are versions with $1 < q < \infty$, see e.g., [18, 29, 38]—though both work with penalized formulations. Also note that if $0 < q < 1$, then (3) yields a nonconvex, quasinorm, while for $q = 0$ and $q = 1$, $\|\cdot\|_{1,q}$ totally decouples, thus losing the grouping effect of mixed-norms.

Other authors have considered overlapping versions of (3), i.e., where subvectors $\boldsymbol{w}_g$ might not be disjoint. However, unless the subvectors have a special structure [15, 22, 24], the resulting mixed-norm can be computationally very expensive. Because our aim is on developing fast algorithms, we focus on the non-overlapping version (3), especially because this version is widely applicable and enjoys numerous applications [5, 11, 12, 14, 16, 19, 26, 34, 36].

Contributions. Before beginning our theoretical discussion, we enlist the key contributions of this paper here for the reader's convenience:

1. An SPG-based algorithm for convex, sparsity-constrained regression problems (such as lasso, multitask lasso, group lasso, etc.).
2. A root-finding procedure grounded in convex-duality, which is crucial to making the SPG-based algorithm practical. As a byproduct we also obtain an efficient method to tackle operators for ℓ_{∞,q^*}-mixed norms.
3. Experimental validation and illustration of our methods on large-scale multitask lasso, leading to a highly competitive method for it.

2 Algorithm and Theory

The simplest method to solve (2) is perhaps *gradient-projection* [6], where starting with some $\boldsymbol{w}^0$, one iterates

$$\boldsymbol{w}^{t+1} = \mathrm{proj}(\boldsymbol{w}^t - \alpha^t \nabla \mathcal{L}(\boldsymbol{w}^t), \gamma, q), \quad t = 0, 1, \ldots, \tag{4}$$

where proj is the *projection operator*, defined as

$$\text{proj}(\boldsymbol{v}, \gamma, q) := \text{argmin}_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{u}\|_{1,q} \leq \gamma. \tag{5}$$

To implement (4) efficiently, there are three obvious, key components: (i) the gradient $\nabla\mathcal{L}$; (ii) the *step-size* $\alpha^t > 0$; and (iii) the projection operator (5). Almost all methods require computation of the gradient (or an approximation).

Let us begin by considering stepsize computation. There exist several classical strategies for computing the step-size α, for example, by exact minimization, by backtracking, Armijo line-search, and so on [6]. However, in general, these strategies are more expensive than absolutely necessary, and although scalable, may even lead to slowly converging algorithm [6, 16, 33].

A fairly recent, and powerful alternative is offered by the *spectral step-sizes* of Barzilai and Borwein [4] (BB), which avoid line-search by providing closed-form formulae for choosing α^t. These stepsizes are

$$\alpha_{BB1} := \frac{\boldsymbol{y}_t^T\boldsymbol{y}_t}{\boldsymbol{s}_t^T\boldsymbol{y}_t}, \quad \text{or} \quad \alpha_{BB2} := \frac{\boldsymbol{y}_t^T\boldsymbol{s}_t}{\boldsymbol{s}_t^T\boldsymbol{s}_t}, \tag{6}$$

where $\boldsymbol{y}_t = \boldsymbol{w}^t - \boldsymbol{w}^{t-1}$, and $\boldsymbol{s}_t = \nabla\mathcal{L}(\boldsymbol{w}^t) - \nabla\mathcal{L}(\boldsymbol{w}^{t-1})$. Directly setting α^t to one of the choices in (6) leads to a gradient-projection method whose convergence is not guaranteed. However, often this substitution displays strong empirical performance. Thus, there have been numerous attempts at using (6) in conjunction with the iteration (4), and one of the most well-known amongst them is the spectral-projected gradient (SPG) method [7].

SPG essentially substitutes (6) in (4) (using a safeguard to ensure bounded and nonzero stepsizes), and thus leverages the strong empirical performance offered by BB stepsizes [4, 7, 9, 33]. SPG ensures global convergence by invoking a nonmontone line search strategy that allows the objective value to occasionally increase, while maintaining some bookkeeping information that allows construction of a descending subsequence.

2.1 Efficiently Computing Projections

The second component of (4) is a core element of our algorithm: projection onto the $\ell_{1,q}$-norm ball. Since every iteration of (4) calls for a projection, it is critical to implement it efficiently. But before we describe the details of our projection algorithm, we first prove a useful duality result.

Lemma 1 (Dual-Norm). *Let $q \geq 1$, and let q^* be its* conjugate exponent*, i.e., it satisfies $1/q + 1/q^* = 1$. Then, the norm $\|\cdot\|_{\infty,q^*}$ is dual to $\|\cdot\|_{1,q}$.*

Proof. By definition, the norm dual to (3) is given by [32]:

$$\|\boldsymbol{u}\|_* := \sup_{\boldsymbol{w}} \ \{\langle \boldsymbol{u}, \boldsymbol{w}\rangle \mid \|\boldsymbol{w}\|_{1,q} \leq 1\}. \tag{7}$$

Consider the inequality

$$\langle \boldsymbol{u}, \boldsymbol{w}\rangle = \sum\nolimits_{g=1}^{G} \langle \boldsymbol{u}_g, \boldsymbol{w}_g\rangle \leq \sum\nolimits_{g=1}^{G} \|\boldsymbol{u}_g\|_{q^*}\|\boldsymbol{w}_g\|_q, \tag{8}$$

which follows directly from Hölder's inequality. Now introduce vectors, $\boldsymbol{\xi} = [\|\boldsymbol{u}_g\|_{q^*}]$, and $\boldsymbol{\psi} = [\|\boldsymbol{w}_g\|_q]$, and apply Hölder's inequality again to obtain

$$\langle \boldsymbol{\xi}, \boldsymbol{\psi} \rangle \leq \|\boldsymbol{\xi}\|_\infty \|\boldsymbol{\psi}\|_1 = \|\boldsymbol{u}\|_{\infty,q^*} \|\boldsymbol{w}\|_{1,q}, \tag{9}$$

so that from Definition (7) we conclude that $\|\boldsymbol{u}\|_* \leq \|\boldsymbol{u}\|_{\infty,q^*}$. To prove that $\|\boldsymbol{u}\|_* = \|\boldsymbol{u}\|_{\infty,q^*}$, we now show that for each $\boldsymbol{u}$, we can find a $\boldsymbol{w}$ satisfying $\|\boldsymbol{w}\|_{1,q} = 1$, and for which $\langle \boldsymbol{u}, \boldsymbol{w} \rangle = \|\boldsymbol{u}\|_{\infty,q^*}$. To that end, let g^* be any index in the set $\left\{\text{argmax}_{1 \leq g \leq G} \|\boldsymbol{u}_g\|_{q^*}\right\}$. To simplify notation let $\boldsymbol{z} = \boldsymbol{u}_{g^*}$ and $\boldsymbol{y} = \boldsymbol{w}_{g^*}$, and then set $\boldsymbol{w}_g = 0$ for all g except for g^* for which

$$(\boldsymbol{w}_{g^*})_i = \boldsymbol{y}_i = \frac{\text{sgn}(z_i)|z_i|^{q^*-1}}{\|\boldsymbol{z}\|_{q^*}^{q^*-1}}. \tag{10}$$

Now observe that the inner-product $\langle \boldsymbol{u}, \boldsymbol{w} \rangle$ satisfies

$$\begin{aligned}\langle \boldsymbol{u}, \boldsymbol{w} \rangle &= \sum\nolimits_g \langle \boldsymbol{u}_g, \boldsymbol{w}_g \rangle = \langle \boldsymbol{z}, \boldsymbol{y} \rangle = \frac{1}{\|\boldsymbol{z}\|_{q^*}^{q^*-1}} \sum\nolimits_i z_i y_i \\ &= \frac{1}{\|\boldsymbol{z}\|_{q^*}^{q^*-1}} \sum\nolimits_i |z_i|^{q^*} = \|\boldsymbol{z}\|_{q^*} = \|\boldsymbol{u}\|_{\infty,q^*},\end{aligned}$$

and that the norm $\|\boldsymbol{w}\|_{1,q} = \|\boldsymbol{w}_{g^*}\|_q = \|\boldsymbol{y}\|_q$ satisfies

$$\begin{aligned}\|\boldsymbol{y}\|_q &= \big(\sum\nolimits_i |y_i|^q\big)^{1/q} = \big(\sum\nolimits_i |z_i|^{q(q^*-1)} / \|\boldsymbol{z}\|_{q^*}^{q(q^*-1)}\big)^{1/q} \\ &= \big(\sum\nolimits_i |z_i|^{q^*} / \|\boldsymbol{z}\|_{q^*}^{q^*}\big)^{1/q} = 1,\end{aligned}$$

which complete the proof of the theorem. □

Note: As the reader may already guess, the proof above also extends to showing the ℓ_{p^*,q^*}-norm is dual to the $\ell_{p,q}$-norm; but we omit details for brevity.

2.2 Projections via proximity

Often, for computing $\text{proj}(\boldsymbol{v}, \gamma, q)$ it proves beneficial to consider the closely related *proximity operator*:

$$\text{prox}(\boldsymbol{v}, \theta, q) := \text{argmin}_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \theta \|\boldsymbol{u}\|_{1,q}. \tag{11}$$

In general, computing (11) can be simpler—for instance, with $q = 1$, (11) is merely the soft-thresholding operator [10], and with $q = \infty$ efficient methods already exist [11, 21]. Interestingly, for $q = 2$ both (11) and (5) are equally easy [5] to compute. The key reason why one may expect (11) to be simpler is because with non-overlapping variable groups, it separates into G independent ℓ_q-norm proximity subtasks. We now show how to leverage this separability.

The idea is simple and could be regarded as well-known [27]. But exploiting it effectively requires some care, as we show below. Let $L(\boldsymbol{u}, \theta)$ be the Lagrangian

to (5), and let the dual optimal be denoted by θ^*. Then, assuming strong-duality, the primal optimal $\boldsymbol{u}^*$ that solves (5) is obtained by computing

$$\boldsymbol{u}^*(\theta^*) = \operatorname{argmin}_{\boldsymbol{u}} \; L(\boldsymbol{u}, \theta^*) := \operatorname{argmin}_{\boldsymbol{u}} \; \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \theta^*(\|\boldsymbol{u}\|_{1,q} - \gamma). \tag{12}$$

But computing (12) requires the knowledge of the optimal θ^*, which in turn depends on u^*. How do we break this circularity? The almost obvious, but really crucial observation here is that the optimal θ^* can be computed by solving a single nonlinear equation. Let us see how.

Note that if $\|\boldsymbol{v}\|_{1,q} \leq \gamma$, then $\boldsymbol{u}^* = \boldsymbol{v}$ is the optimal solution. So we assume that $\|\boldsymbol{v}\|_{1,q} > \gamma$; in this case, the optimal θ^* satisfies $\|\boldsymbol{u}(\theta^*)\|_{1,q} = \gamma$. Define now

$$\boldsymbol{u}(\theta) := \operatorname{prox}(\boldsymbol{v}, \theta, q), \tag{13}$$

and consider the nonlinear function

$$g(\theta) = -\gamma + \|\boldsymbol{u}(\theta)\|_{1,q}, \tag{14}$$

from which the optimal θ^* can be obtained by solving $g(\theta) = 0$. Toward solving this equation, Lemma 2 proves useful.

Lemma 2. *Let $g(\theta)$ be defined by (14). There exists an interval $[0, \theta_{\max}]$ on which $g(\theta)$ is monotonically decreasing, and differs in sign at the endpoints.*

Proof. First, observe that by our assumption on $\|\boldsymbol{v}\|_{1,q}$, the inequality $g(0) = -\gamma + \|\boldsymbol{v}\|_{1,q} > 0$ holds.

Next, notice that for $\theta' \geq \|\boldsymbol{v}\|_{\infty,q^*}$, the solution $\boldsymbol{u}(\theta') = 0$. To see why, suppose $\theta' \geq \|\boldsymbol{v}\|_{\infty,q^*}$ but $\boldsymbol{u}(\theta') \neq 0$. Then, $\frac{1}{2}\|\boldsymbol{u}(\theta) - \boldsymbol{v}\|_2^2 + \theta'\|\boldsymbol{u}(\theta)\|_{1,q} < \frac{1}{2}\|\boldsymbol{v}\|_2^2$. Since $\|\cdot\|_2^2$ is strictly convex, for all $\boldsymbol{u}$ we have the inequality $\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 - \|\boldsymbol{v}\|_2^2 > -2\langle \boldsymbol{v}, \boldsymbol{u}\rangle$. Combining the two inequalities we obtain $\theta' < \langle \boldsymbol{v}, \boldsymbol{u}(\theta)\rangle / \|\boldsymbol{u}(\theta)\|_{1,q}$. Hence, if $\theta' \geq \sup_{\boldsymbol{u}\neq 0} \langle \boldsymbol{v}, \boldsymbol{u}\rangle / \|\boldsymbol{u}\|_{1,q} = \|\boldsymbol{v}\|_{\infty,q^*}$ (see (7)), then $\boldsymbol{u}(\theta)^*$ must equal 0. This argument implies that $g(\theta') = -\gamma < 0$, allowing us to select $\theta_{\max} = \theta'$.

Finally, to establish monotonicity of $g(\theta)$, merely recognize that $g(\theta)$ is the derivative of the (concave) *dual function* $\inf_{\boldsymbol{u}} L(\boldsymbol{u}, \theta)$. □

Since g changes sign in the interval $[0, \theta_{\max}]$, is continuous, and monotonic, it has a unique root in $[0, \theta_{\max}]$. We can compute this root θ^* to ϵ-accuracy using bisection in $O(\log(\theta_{\max}/\epsilon))$ iterations. In practice, we *do not use* plain bisection, but invoke a more powerful root-finder that combines bisection, inverse quadratic interpolation, and the secant method. Algorithm 1 provides pseudocode for computing the projection (5) based on the above ideas.

2.3 Computing the proximity operators

Now that we have reduced $\ell_{1,q}$-projections to a sequence of proximity computations, a few words about the associated proximity operators are in order. Depending on the choice of q, these operators can range from trivial to complicated. However, for all the computations, the key benefit arises from $\ell_{1,q}$ being a

Input: Subroutine for computing $\text{prox}(\boldsymbol{v}, \theta, q)$; vector $\boldsymbol{v}$; scalar $\gamma > 0$
Output: $\boldsymbol{u}^* = \text{argmin}_{\boldsymbol{u}} \|\boldsymbol{u} - \boldsymbol{v}\|_2$, s.t. $\|\boldsymbol{u}\|_{1,q} \leq \gamma$
if $\|\boldsymbol{v}\|_{1,q} \leq \gamma$ **then**
  return $\boldsymbol{u}^* = \boldsymbol{v}$
else
  Define $g(\theta) := -\gamma + \|\text{prox}(\boldsymbol{v}, \theta, q)\|_{1,q}$;
  Compute root-bracket $(\theta_{\min}, \theta_{\max}) = (0, \|\boldsymbol{v}\|_{\infty,q^*})$;
  Compute root $\theta^* = \text{FindRoot}(g(\theta), \theta_{\min}, \theta_{\max})$;
end
return $\boldsymbol{u}^* = \text{prox}(\boldsymbol{v}, \theta^*, q)$

Algorithm 1: Projection via proximity using root-finding

sum of independent terms over the G groups (recall that $\boldsymbol{v} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_G]$). Owing to this separability, the computation (13) decomposes into G independent, proximity subtasks, one for each $\boldsymbol{v}_g$, for $1 \leq g \leq G$.

We reiterate that unlike [21] or other authors who solve $\ell_{1,q}$-norm penalized regressions, our setup (for general q) is harder: we solve an $\ell_{1,q}$-**constrained** regression. This requires computing $\boldsymbol{u}(\theta)$ for several values of θ.

To simplify notation, in the sequel we use $\boldsymbol{u}(\theta)$, $\boldsymbol{v}$, and $\text{prox}(\boldsymbol{v}, \theta, q)$ to refer to an arbitrary ***single group*** of variables, so that the proximity task at hand is

$$\boldsymbol{u}(\theta) := \text{argmin}_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \theta\|\boldsymbol{u}\|_q, \quad q \geq 1. \tag{15}$$

First, we mention existing proximity algorithms for solving (15), and then, we present some new results in Section 2.3.1.

For $q = 1$, (15) reduces to the well-known *soft-thresholding* operation [10]:

$$\boldsymbol{u}(\theta) = \text{sgn}(\boldsymbol{v}) \odot \max(|\boldsymbol{v}| - \theta, 0), \tag{16}$$

where $\odot$ represents elementwise multiplication. For $q = 2$, the solution is again available in closed form (see e.g., [8, 11]), and is given by

$$\boldsymbol{u}(\theta) = \max(1 - \theta\|\boldsymbol{v}\|_2^{-1}, 0)\boldsymbol{v}. \tag{17}$$

For $q = \infty$, the solution is slightly more involved than (16) and (17), but can be obtained via the *Moreau decomposition* [8], which for $\text{prox}(\boldsymbol{v}, \theta, \infty)$ implies that

$$\text{prox}(\boldsymbol{v}, \theta, \infty) = \boldsymbol{v} - \text{proj}(\boldsymbol{v}, \theta, 1). \tag{18}$$

The projection $\text{proj}(\boldsymbol{v}, \theta, 1)$ is very well-studied: it is projection onto the ℓ_1-norm ball [17, 23, 25]. This choice of q arises most notably in grouped feature selection in multitask learning settings [19, 26, 28].

For $q > 1$ different from the choices above, the proximity problems are significantly harder. Fortunately, efficient proximity algorithms for ℓ_q-norms were recently developed in [20, 21]. These algorithms use nested root-finding subroutines: one for solving single variable nonlinear equations; one for performing bisection over a parameter akin to θ. But unlike the cases with $q \in \{1, 2, \infty\}$, due to the highly nonlinear derivatives, one can obtain only an approximate solution to the proximity operator.

2.3.1. Proximity operators for $\|\cdot\|_{\infty,q^*}$: We now leverage the machinery developed in Sections 2.1 and 2.2 to obtain fast proximity operators for ℓ_{∞,q^*}-norms, almost as a byproduct. The first step is to invoke convex duality.

Proposition 1. *Let $\|\cdot\|$ be any norm, and $\|\cdot\|_*$ its corresponding dual norm; also let $\gamma > 0$. Then, the following two problems are dual to each other:*

$$\min_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 + \gamma\|\boldsymbol{u}\|, \tag{19}$$

$$\max_{\boldsymbol{z}} \quad -\tfrac{1}{2}\|\boldsymbol{z}\|_2^2 + \boldsymbol{z}^T\boldsymbol{v}, \quad \textit{s.t. } \|\boldsymbol{z}\|_* \leq \gamma. \tag{20}$$

Moreover, if $\boldsymbol{z}^*$ is the optimal dual solution, then the corresponding primal is given by $\boldsymbol{u}^* = \boldsymbol{v} - \boldsymbol{z}^*$.

Proof. Follows from [32, Theorem 31.5] (essentially Moreau's decomposition).

Proposition 1 instantly yields the following useful corollary.

Corollary 1. *A problem dual to the ℓ_{∞,q^*}-proximity operator*

$$\min_{\boldsymbol{u}} \quad \tfrac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2 + \gamma\|\boldsymbol{u}\|_{\infty,q},$$

is the $\ell_{1,q}$-norm projection task (q is conjugate to q^*):

$$\min_{\boldsymbol{z}} \quad \tfrac{1}{2}\|\boldsymbol{z}-\boldsymbol{v}\|_2^2, \quad \textit{s.t.} \quad \|\boldsymbol{z}\|_{1,q} \leq \gamma. \tag{21}$$

Proof. Lemma 1 tells us that $\|\cdot\|_{1,q^*}$ is dual to $\|\cdot\|_{\infty,q}$. Consequently, Proposition 1 implies the result (21) (after completing squares).

Corollary 1 allows computing ℓ_{∞,q^*}-norm proximity by replacing it with the corresponding $\ell_{1,q}$-norm projection, which in turn can be solved by Algorithm 1.

3 Experimental results

In the discussion above we presented the following two main algorithms:

(i) The SPG based method iteration (4) for solving (2);
(ii) Algorithm 1, a method for computing the $\ell_{1,q}$-norm projection (5).

As previously mentioned, for $q = 1$, the projections reduce to the classical ℓ_1-norm ball problem, while for $q = 2$, already highly efficient methods are available [5]. The case $1 < q < \infty$ seems to be largely unstudied, while for $q = \infty$, recently an efficient method was proposed [28]. Thus, we begin our numerical results by evaluating our projection algorithms for computing projections onto the $\ell_{1,\infty}$-norm ball.

Note: It is not one of the aims of this paper to compare different MTL formulations (depending on different choices of q). Our main aim is to show scalability that comes from having fast proximity operators, which we highlight by using them as subroutines in a larger problem. The subroutines themselves can be used in any proximal splitting method that solves $\ell_{1,q}$-norm constrained problems. However, a longer version of this paper will include more detailed experimental evaluation.

$\frac{\gamma}{\|\boldsymbol{V}\|_{1,\infty}}$	QP_{time} (s)	**FP_{time} (s)**	**Speedup**	$\text{QP}_{\text{accuracy}}$	$\text{FP}_{\text{accuracy}}$	QP_{obj}	FP_{obj}
0.01	22719.76	**28.26**	**804.09**	7.76E-09	**7.28E-12**	6.8778E+03	6.8778E+03
0.05	20165.31	**24.08**	**837.38**	7.33E-09	**7.09E-11**	6.1387E+03	6.1387E+03
0.10	17064.19	**23.80**	**716.93**	5.69E-09	**5.82E-11**	5.2850E+03	5.2850E+03
0.20	11491.00	**24.74**	**464.40**	8.32E-09	**2.47E-10**	3.8133E+03	3.8133E+03
0.30	7046.42	**24.89**	**283.11**	4.98E-09	**4.44E-10**	2.6484E+03	2.6484E+03
0.40	3933.99	**29.69**	**132.52**	1.64E-08	**8.59E-10**	1.7656E+03	1.7656E+03
0.50	1982.77	**31.03**	**63.90**	9.30E-09	**5.82E-11**	1.1263E+03	1.1263E+03
0.60	905.22	**31.33**	**28.89**	1.56E-09	**7.13E-10**	6.8445E+02	6.8445E+02
0.70	380.78	**29.41**	**12.95**	**4.21E-09**	6.29E-09	3.9254E+02	3.9254E+02

Table 1. Runtime and accuracy for QP and FP on a $50,000 \times 10,000$ matrix $\boldsymbol{V}$.

3.1 Projection onto the $\ell_{1,\infty}$-ball

For ease of comparison, we use the notation of [28], which seems to be the currently standard method. In their notation, the projection task is to solve

$$\min_{\boldsymbol{W}} \quad \tfrac{1}{2}\|\boldsymbol{W}-\boldsymbol{V}\|_{\text{F}}^2, \quad \text{s.t.} \quad \sum\nolimits_{i=1}^{d} \|\boldsymbol{w}^i\|_\infty \leq \gamma, \tag{22}$$

where $\boldsymbol{W}$ is a $d \times n$ matrix, and $\boldsymbol{w}^i$ denotes the ith row of $\boldsymbol{W}$.

In our comparisons below, we refer to [28]'s algorithm, which was available as C code[1], as 'QP'; our fast projection method is called 'FP'. We show numerical results for a representative large-sized problem, and to stress-test both QP and FP we show results on a large dense matrix. In particular, we use a matrix $\boldsymbol{V} \in \mathbb{R}^{50{,}000 \times 1000}$ having entries drawn following $\mathcal{N}(0,1)$. Note that the matrix $\boldsymbol{V}$ has a total of 50 million nonzero entries.

We compute the optimal $\boldsymbol{W}^*$, as γ varies from $0.01\|\boldsymbol{V}\|_{1,\infty}$ (more sparse; difficult) to $0.7\|\boldsymbol{V}\|_{1,\infty}$ (less sparse; easy) settings. Table 1 presents the associated running times, objective function values, and accuracies (accuracy is measured by the constraint violation: $|\lambda - \|\boldsymbol{W}^*\|_{1,\infty}|$, for an estimated $\boldsymbol{W}^*$).

Minor numerical differences between both algorithms are due to unavoidable floating-point round-off errors. Also noteworthy is the fact that although QP is an "exact" method, and FP is based on root-finding, the latter ends up obtaining solutions of higher accuracy than QP. The results indicate the tremendous advantages that our method offers for large-scale data, where it vastly outperforms the competition.

3.2 Projection onto $\ell_{1,q}$-balls

Our next experiment shows running time results for computing projection onto $\ell_{1,q}$ balls; in our experiment we selected $q \in \{1.5, 2.5, 3, 5\}$. Here too, we use matrix-based groups and solve

$$\min_{\boldsymbol{W}} \quad \|\boldsymbol{W}-\boldsymbol{V}\|_{\text{F}}^2, \quad \text{s.t.} \ \sum\nolimits_i \|\boldsymbol{w}^i\|_q. \tag{23}$$

For each value of q, the plots in Figure 1 also show the running times requires as the parameter γ is varied. From these plots the we observe four main points: (i)

[1] *http://www.lsi.upc.edu/∼aquattoni/CodeToShare/*

the runtimes seem to be largely independent of the value of γ; (ii) for relatively small q, the projection times are approximately same; and (iii) for larger q (here $q = 5$), the projection times increase dramatically.

Moreover, from the actual running times it is apparent our projection code scales linearly with the data size. For example, the matrix corresponding to the second bar plot has 25 times more parameters than the first plot, and the runtimes reported in the second plot are approximately 25–30 times higher.

The running times in Figure 1 suggest that although the running times scale linearly, a single $\ell_{1,q}$-norm projection still takes nontrivial effort. Thus, even though our $\ell_{1,q}$-projection method is relatively fast, currently we can recommend it only for small and medium-scale regression problems.

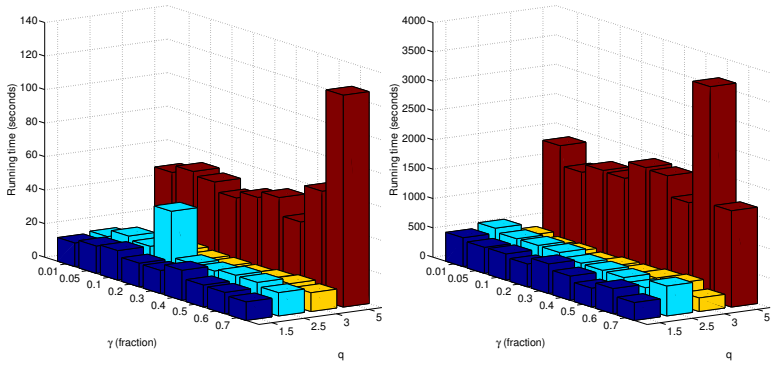

Fig. 1. Running times for $\ell_{1,q}$-norm projections as scalars q and ratios $\gamma/\|\boldsymbol{V}\|_{1,q}$ vary. The left plot is on a 1000×100 matrix, while the right one is on a 5000×500 matrix.

4 Multitask Lasso with $\ell_{1,\infty}$-constraint

Multitask Lasso (MTL) [19, 36] is a typical grouped feature selection problem, where important features are separated from less important ones by using information shared across multiple tasks. The feature selection is effected by a sparsity promoting mixed-norm, usually the $\ell_{1,\infty}$-norm [19].

MTL is setup is as follows. Let $\boldsymbol{X}_j \in \mathbb{R}^{m_j \times d}$ be the data matrix for task j, where $1 \leq j \leq n$. MTL seeks a parameter matrix $\boldsymbol{W} \in \mathbb{R}^{d \times n}$, each column of which corresponds to a task, which are regularized across the same feature by applying the mixed-norm over the rows $\boldsymbol{w}^i$ $(1 \leq i \leq d)$ of $\boldsymbol{W}$. This leads to a "grouped" feature selection, because if $\|\boldsymbol{w}^i\|_\infty = 0$, then the entire row $\boldsymbol{w}^i$ is eliminated (i.e., feature i is removed). A common formulation for MTL is

$$\min_{\boldsymbol{w}_1,\ldots,\boldsymbol{w}_n} \quad \mathcal{L}(\boldsymbol{W}) := \sum\nolimits_{j=1}^{n} \tfrac{1}{2}\|\boldsymbol{y}_t - \boldsymbol{X}_j\boldsymbol{w}_j\|_2^2, \quad \text{s.t.} \quad \sum\nolimits_{i=1}^{d} \|\boldsymbol{w}^i\|_\infty \leq \gamma, \quad (24)$$

Name	(m, d, n)	#nonzeros	RAM required
M1	(10, 5000, 20)	10^6	7.63MB
M2	(100, 5000, 20)	10^7	76.29MB
M3	(100, 10000, 100)	10^8	0.75GB
M4	(800, 15000, 300)	$3.6 \cdot 10^9$	26.822GB

Table 2. Details of simulation data used for MTL. For simplicity, all matrices X_j (for each task $1 \leq j \leq n$), were chosen to have size $m \times d$.

where the $\boldsymbol{y}_t$ are the dependent variables, and $\gamma > 0$ is a sparsity-tuning parameter. Notice that the loss-function combines the different tasks (over columns of $\boldsymbol{W}$), but the overall problem does not decompose into separable problems because the mixed-norm constrained is over the *rows* of $\boldsymbol{W}$.

4.1 Simulation results for MTL

In our first set of results (Tables 2, 3) we report running time comparisons between two different invocations of an SPG-based method for solving (24), once with QP as the projection method and once with FP—we call the corresponding solvers SPG_{QP}, and SPG_{FP}.

We show simulation results on small, medium, large, and very large-scale data matrices. The key aim of our experiments is to offer strong evidence showing that for data with a large number of features, or for data having large size, SPG_{FP} vastly outstrips SPG_{QP}.

We only compare SPG based implementations because SPG offers a simple, yet highly competitive framework for solving *constrained* convex problems. The other efficient MTL algorithms that are available, e.g., [16, 21], solve the simpler, *penalized* version of the problem. Moreover, even if we were to use a method such as PQN [33] that can handle mixed-norm constraints, the final performance of potential PQN+QP or PQN+FP combinations would exhibit trends, similar to those reported in Table 3. This is so, because of the vastly different costs of projection incurred by QP and FP. We also note in passing that methods such as block-coordinate descent that are popular for several lasso-type problems, do not scale well to the large MTL problems that we consider.

The running times shown in Table 3 suggest that SPG_{FP} is a valuable choice for solving large-scale MTL problems. Note that both SPG_{QP} and SPG_{FP} find exactly (up to roundoff error) the same solution, as both of them just perform an equal number of SPG iterations and $\ell_{1,\infty}$-projections. The difference lies in the speed of the overall execution.

4.2 MTL results on real-world data

Now we show running time results comparing SPG_{QP} against SPG_{FP} on a real-world dataset. These results corroborate the claims of the previous section, and indicate that the powerful speedups observed on simulated data also carry over to realistic data.

Dataset	#projs	proj_{QP}	SPG_{QP}	proj_{FP}	SPG_{FP}	Speedup
M1	171	822.2	826.3	26.58	**31.7**	**26.06**
M2	31	121.9	125.9	5.6	**11.4**	**11.04**
M3	29	2433.9	2474.6	31.6	**84.2**	**29.40**
M4	16	735.2	1054.2	6.0	**321.5**	**3.28**

Table 3. Running time (in seconds) comparisons for four different (synthetic) MTL datasets. The MTL problems were solved to an accuracy of 10^{-5}, and the total number of projections required to reach this accuracy are reported under the column '#projs'. The columns 'proj_{QP}' and 'proj_{FP}', report the total time spent by the SPG_{QP} and SPG_{FP} methods, respectively, for the $\ell_{1,\infty}$-projections alone. The total time taken by SPG_{QP} and SPG_{FP} is reported under columns with the same name. For SPG_{QP} the cost of projection dominates the total runtime, while SPG_{FP} this is not the case.

Also noteworthy is the observation that the speedups attained become more significant with increasing data dimensionality. Thus, for many machine learning and statistics datasets, the acceleration offered by our fast projection algorithms can be advantageous.

For our experiments with real-world data, we run MTL on a subset of the CMU Newsgroups dataset[2]. This data corresponds to 5 feature selection tasks based on data taken from the following newsgroups: computer, politics, science, recreation, and religion. The feature selection tasks are spread over the matrices $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_5$, each of size 2907×53975, while the dependent variables $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_5$ correspond to class labels.

We note that for timing experiments, the exact details of the data are not that critical, except the fact that the number of features (53975) is large; indeed much larger than in our simulations. For such a large number of features, based on the results of Table 1, one can already anticipate that SPG_{FP} will strongly outperform the competition.

(γ-fraction)	#projs	proj_{QP}	SPG_{QP}	proj_{FP}	SPG_{FP}	Speedup
0.01	85	6322.2	6806.6	96.2	**507.6**	**13.41**
0.1	94	9335.2	9759.5	92.6	**649.5**	**15.03**
0.2	97	4209.8	4746.2	112.8	**554.3**	**8.56**
0.3	99	4516.6	4951.2	109.7	**514.9**	**9.62**

Table 4. Running time (in seconds) comparisons for 4 different runs of MTL on the CMU newsgroups data. Here, γ-fraction (cf. Table 1) indicates sparsity-level; low γ means high sparsity, and consequently difficult optimization. The other columns are the same as in Table 3.

[2] Original available from: *http://www.cs.cmu.edu/∼textlearning/*. We obtained the reduced, pre-processed version from the authors of [16].

5 Conclusions

In this paper we took a careful look at projections onto $\ell_{1,q}$-mixed norm balls. This projection arises as a key step (particularly for $q = 2, \infty$) in groupwise feature selection problems, such as multitask lasso or group lasso. We first presented a simple spectral projected gradient (SPG)-based method for solving convex regression problems with $\ell_{1,q}$-norm constraints. We chose SPG as it offers a simple to use method that displays strong empirical performance. This performance, though, in mixed-norm regression problems depends on the projection step being cheap. Thus, to handle projections efficiently, we presented a generic root-finding algorithm.

Our numerical results highlighted our root-finding method, specialized to the $\ell_{1,\infty}$-norm, by comparing it against the state-of-the-art method of [28]. The speedups observed were of almost three orders of magnitude. Building on these speedups, we showed how SPG combined with our projection method leads to an effective multitask lasso algorithm. On both simulated and real-world data our numerical results indicate the added efficiency afforded by methods.

At this juncture, several directions of future work are open. The most challenging amongst them is the development of a projection based method that can outperform the well-established SPG method. Additional avenues of work include extending some of our ideas to tackle more complex structured sparsity inducing norms [1, 24].

References

1. Bach, F.: Structured sparsity-inducing norms through submodular functions. In: NIPS (2010)
2. Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Convex optimization with sparsity-inducing norms. In: Sra, S., Nowozin, S., Wright, S.J. (eds.) Optimization for Machine Learning. MIT Press (2011)
3. Bach, F.R.: Consistency of the Group Lasso and Multiple Kernel Learning. J. Mach. Learn. Res. 9, 1179–1225 (2008)
4. Barzilai, J., Borwein, J.M.: Two-Point Step Size Gradient Methods. IMA Journal of Numerical Analysis 8(1), 141–148 (1988)
5. van den Berg, E., Schmidt, M., Friedlander, M.P., Murphy, K.: Group sparsity via linear-time projection. Tech. Rep. TR-2008-09, Univ. British Columbia (Jun 2008)
6. Bertsekas, D.P.: Nonlinear Programming. Athena Scientific, second edn. (1999)
7. Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone Spectral Projected Gradient Methods on Convex Sets. SIAM J. Opt. 10(4), 1196–1211 (2000)
8. Combettes, P.L., Pesquet, J.: Proximal Splitting Methods in Signal Processing. arXiv:0912.3522v4 (May 2010)
9. Dai, Y.H., Fletcher, R.: Projected Barzilai-Borwein Methods for Large-scale Box-constrained Quadratic Programming. Numerische Mathematik 100(1), 21–47 (2005)
10. Donoho, D.: Denoising by soft-thresholding. IEEE Tran. Inf. Theory 41(3), 613–627 (2002)
11. Duchi, J., Singer, Y.: Online and Batch Learning using Forward-Backward Splitting. JMLR (Sep 2009)

12. Evgeniou, T., Micchelli, C., Pontil, M.: Learning multiple tasks with kernel methods. J. Mach. Learn. Res. 6, 615–637 (2005)
13. Evgeniou, T., Pontil, M.: Regularized multi-task learning. In: KDD (2004)
14. Friedman, J., Hastie, T., Tibshirani, R.: A note on the group lasso and a sparse group lasso. arXiv:1001.0736v1 [math.ST] (Jan 2010)
15. Jenatton, R., Mairal, J., Obozinski, G., Bach, F.: Proximal Methods for Sparse Hierarchical Dictionary Learning. In: ICML (2010)
16. Kim, D., Sra, S., Dhillon, I.S.: A scalable trust-region algorithm with application to mixed-norm regression. In: Int. Conf. Machine Learning (ICML) (2010)
17. Kiwiel, K.: On Linear-Time Algorithms for the Continuous Quadratic Knapsack Problem. Journal of Optimization Theory and Applications 134, 549–554 (2007)
18. Kowalski, M.: Sparse regression using mixed norms. Applied and Computational Harmonic Analysis 27(3), 303 – 324 (2009)
19. Liu, H., Palatucci, M., Zhang, J.: Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery. In: Int. Conf. Machine Learning (Jun 2009)
20. Liu, J., Ji, S., Ye, J.: SLEP: Sparse Learning with Efficient Projections. Arizona State University (2009), http://www.public.asu.edu/ jye02/Software/SLEP
21. Liu, J., Ye, J.: Efficient L1/Lq Norm Regularization. arXiv:1009.4766v1 (2010)
22. Liu, J., Ye, J.: Moreau-Yosida Regularization for Grouped Tree Structure Learning. In: NIPS (2010)
23. Liu, J., Ye, J.: Efficient Euclidean projections in linear time. In: ICML (Jun 2009)
24. Mairal, J., Jenatton, R., Obozinski, G., Bach, F.: Network Flow Algorithms for Structured Sparsity. In: NIPS (2010)
25. Michelot, C.: A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$. J. Optim. Theory Appl. 50(1), 195–200 (1986)
26. Obonzinski, G., Taskar, B., Jordan, M.: Multi-task feature selection. Tech. rep., UC Berkeley (Jun 2006)
27. Patriksson, M.: A survey on a classic core problem in operations research. Tech. Rep. 2005:33, Chalmers University of Technology and Göteborg University (Oct 2005)
28. Quattoni, A., Carreras, X., Collins, M., Darrell, T.: An Efficient Projection for $\ell_{1,\infty}$ Regularization. In: ICML (2009)
29. Rakotomamonjy, A., Flamary, R., Gasso, G., Canu, S.: $\ell_p - \ell_q$ penalty for sparse linear and sparse multiple kernel multi-task learning. Tech. Rep. hal-00509608, Version 1, INSA-Rouen (2010)
30. Rice, U.: Compressive sensing resources. *http://dsp.rice.edu/cs* (Oct 2010)
31. Rish, I., Grabarnik, G.: Sparse modeling: ICML 2010 tutorial. Online (Jun 2010)
32. Rockafellar, R.T.: Convex Analysis. Princeton Univ. Press (1970)
33. Schmidt, M., van den Berg, E., Friedlander, M., Murphy, K.: Optimizing Costly Functions with Simple Constraints: A Limited-Memory Projected Quasi-Newton Algorithm. In: AISTATS (2009)
34. Similä, T., Tikka, J.: Input selection and shrinkage in multiresponse linear regression. Comp. Stat. & Data Analy. 52(1), 406 – 422 (2007)
35. Tropp, J.A.: Algorithms for simultaneous sparse approximation, Part II: Convex relaxation. Signal Proc. 86(3), 589–602 (2006)
36. Turlach, B.A., Venables, W.N., Wright, S.J.: Simultaneous Variable Selection. Technometrics 27, 349–363 (2005)
37. Yuan, M., Lin, Y.: Model Selection and Estimation in Regression with Grouped Variables. Tech. Rep. 1095, Univ. of Wisconsin, Dept. of Stat. (2004)

38. Zhang, Y., Yeung, D.Y., Xu, Q.: Probabilistic Multi-Task Feature Selection. In: NIPS (2010)
39. Zhao, P., Rocha, G., Yu, B.: The composite absolute penalties family for grouped and hierarchical variable selection. Ann. Stat. 37(6A), 3468–3497 (2009)