Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 192

Generalized Proximity and Projection with Norms and Mixed-norms

Suvrit Sra[1]

May 5, 2010

This Technical Report has been approved by:

Director at MPIK

Postdoc at MPIK

Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. 192

Generalized Proximity and Projection with Norms and Mixed-norms

Suvrit Sra[1]

May 5, 2010

[1] MPI für biologische Kybernetik

This report is available in PDF–format via anonymous ftp at ftp://ftp.kyb.tuebingen.mpg.de/pub/mpi-memos/pdf/projectionsTR.pdf.
The complete series of Technical Reports is documented at: http://www.kyb.tuebingen.mpg.de/techreports.html

Generalized Proximity and Projection with Norms and Mixed-norms

Suvrit Sra

Abstract. We discuss generalized proximity operators (GPO) and their associated generalized projection problems. On inputs of size n, we show how to efficiently apply GPOs and generalized projections for *separable* norms and distance-like functions to accuracy ϵ in $O(n\log(1/\epsilon))$ time. We also derive projection algorithms that run theoretically in $O(n\log n\log(1/\epsilon))$ time but can for suitable parameter ranges empirically outperform the $O(n\log(1/\epsilon))$ projection method. The proximity and projection tasks are either separable, and solved directly, or are reduced to a single root-finding step. We highlight that as a byproduct, our analysis also yields an $O(n\log(1/\epsilon))$ (weakly linear-time) procedure for Euclidean projections onto the $\ell_{1,\infty}$-norm ball; previously only an $O(n\log n)$ method was known. We provide empirical evaluation to illustrate the performance of our methods, noting that for the $\ell_{1,\infty}$-norm projection, our implementation is more than two orders of magnitude faster than the previously known method.

---

1 Introduction

The focus of this paper is on the following two related problems:

$$\text{(Generalized proximity operator)} \qquad \min_{\boldsymbol{x}} \quad F(\boldsymbol{x},\boldsymbol{y})+\lambda\|\boldsymbol{x}\|, \tag{GPO}$$

$$\text{(Generalized projection problem)} \qquad \min_{\boldsymbol{x}} \quad F(\boldsymbol{x},\boldsymbol{y}) \quad \text{s.t.} \quad \|\boldsymbol{x}\|\le\gamma, \tag{GPP}$$

where $\lambda,\gamma>0$, and $F:\mathcal{X}\times\mathcal{Y}\to\mathbb{R}_+$, where $\mathcal{X},\mathcal{Y}\subseteq\mathbb{R}^n$, is a "distance-like" function defined as

$$F(\boldsymbol{x},\boldsymbol{y})=\sum\nolimits_{i=1}^{n} f_i(x_i,y_i). \tag{1.1}$$

Each $f_i:\mathcal{X}_i\times\mathcal{Y}_i\to\mathbb{R}_+$ (for $\mathcal{X}_i,\mathcal{Y}_i\subseteq\mathbb{R}$) is a differentiable, strictly convex function (over $\mathcal{X}_i$), that satisfies the key "distance-like" property

$$f_i(x,y)\ge 0, \quad \text{with equality if and only if} \quad x=y. \tag{1.2}$$

Notable examples of (1.2) are: ℓ_p^p *functions*, $f_i(x,y)=(1/p_i)|x-y|^{p_i}$ $(p_i>1)$; *Bregman divergences*, $f_i(x,y)=\varphi_i(x)-\varphi_i(y)-\varphi_i'(y)(x-y)$, where φ_i is a differentiable, strictly convex function [1, 2]. For the special case $f_i(x,y)=0.5|x-y|^2$, (GPO) reduces to a (Euclidean) proximity operator for $\lambda\|\cdot\|$ [3], while (GPP) reduces to a Euclidean projection onto the $\|\cdot\|$-norm ball; this explains our choice of nomenclature.

1.1 Related Work

Efficiently solving separable convex optimization problems such as (GPO) and (GPP) has long captured the attention of many researchers. For a comprehensive listing of references we refer the reader to the tech-report by Patriksson [7].

2 Efficiently solving (GPO)

The main difficulty in solving (GPO) or (GPP) comes from the nonsmooth term $\|\boldsymbol{x}\|$. When this norm is separable, then Lemma 1 below (which holds for arbitrary norm though) can prove helpful for simplifying the problem.

Lemma 1. *If $\boldsymbol{x}^*$ is the optimal solution to* (GPO) *or* (GPP)*, then: (i) $x_i^*y_i\ge 0$, and (ii) $|x_i^*|\le|y_i|$.*

Proof. **(i)**: We describe the proof only for the (GPP) setup, because for an appropriate $\gamma(\lambda) > 0$, (GPO) can be recast as a (GPP) problem. The proof is by contradiction; so we assume that there exists an index j for which $x_j^* y_j < 0$. Now choose a vector $\hat{\boldsymbol{x}} = \boldsymbol{x}^*$, except for $\hat{x}_j = 0$. Then, the difference $F(\boldsymbol{x}^*, \boldsymbol{y}) - F(\hat{\boldsymbol{x}}, \boldsymbol{y})$ is

$$\sum\nolimits_i \big(f_i(x_i^*, y_i) - f_i(\hat{x}_i, y_i)\big) = f_j(x_j^*, y_j) - f_j(0, y_j) > x_j^* f_j'(0, y_j), \tag{2.1}$$

where the inequality follows as f_j is strictly convex. If $x_j^* > 0$, then $y_j < 0$, whereby the strict monotonicity of f_j' implies that $f_j'(y_j, y_j) < f_j'(0, y_j)$. Property (1.1) then allows us to conclude $0 < f_j'(0, y_j)$, and thereby $x_j^* f_j'(0, y_j) > 0$. A similar argument for $x_j^* < 0$ yields $x_j^* f_j'(0, y_j) > 0$. But these inequalities imply that (2.1) is positive, which contradicts the optimality of $\boldsymbol{x}^*$ since $\hat{\boldsymbol{x}}$ is also feasible.
(ii): This part also follows by contradiction, using $\hat{\boldsymbol{x}} = \boldsymbol{x}^*$ except for $\hat{x}_j = y_j$. □

Remark 1: It is easy to see that Lemma 1 also holds for constraints of the form $h(\boldsymbol{x}) \leq \gamma$, where h is a non-decreasing function that satisfies $h(\boldsymbol{x}) \leq h(\boldsymbol{z})$, whenever $\|\boldsymbol{x}\| \leq \|\boldsymbol{z}\|$ for some norm or quasi-norm.
Remark 2: It is interesting to note that Lemma 1 also holds for the ℓ_0-norm.[1]

2.1 GPO with the ℓ_1-norm

Here we consider (GPO) with $\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_1$. In this case we need to compute the GPO

$$\min\nolimits_{\boldsymbol{x}} \quad F(\boldsymbol{x}, \boldsymbol{y}) + \lambda \|\boldsymbol{x}\|_1. \tag{2.2}$$

Lemma 1 suggests that (2.2) may be rewritten as

$$\min\nolimits_{\boldsymbol{x}} \quad F(\boldsymbol{x}, \boldsymbol{y}) + \lambda \boldsymbol{s}^T \boldsymbol{x}, \quad \text{s.t.} \quad s_i x_i \geq 0 \quad \text{for } 1 \leq i \leq n, \tag{2.3}$$

where $s_i = \operatorname{sgn}(y_i)$. Problem (2.3) separates into n problems of the form

$$\min_{s_i x_i \geq 0} \quad f_i(x_i, y_i) + \lambda s_i x_i. \tag{2.4}$$

To solve (2.4), we distinguish between two cases: $x_i^* = 0$, and $s_i x_i^* > 0$. Which one of these two is chosen for x_i^* depends on which leads to a lower objective. The following simple lemma makes this choice explicit.

Lemma 2. *Let $\hat{x}_i(\lambda)$ solve $f_i'(x_i, y_i) + \lambda s_i = 0$, where $s_i = \operatorname{sgn}(y_i)$. Then, the solution to (2.4) is given by*

$$x_i^*(\lambda) = \begin{cases} 0, & \text{if} \quad |f_i'(0, y_i)| \leq \lambda, \\ \hat{x}_i(\lambda), & \text{if} \quad |f_i'(0, y_i)| > \lambda. \end{cases} \tag{2.5}$$

Proof. Assuming the second case in (2.5) holds, by direct calculation we see that $\hat{x}_i(\lambda)$ is the desired solution. We merely need to verify the first case. Thus, assume $|f_i'(0, y_i)| \leq \lambda$ and divide the analysis into two cases: (i) $f_i'(0, y_i) \geq 0$ and (ii) $f_i'(0, y_i) < 0$. For Case (i), Property (1.1) implies that $y_i \leq 0$, so that any potentially optimal $\hat{x}_i$ must be ≤ 0, whereby $f_i'(0, y_i) \leq \lambda \implies \hat{x}_i f_i'(0, y_i) \geq \hat{x}_i \lambda$. But since f_i is convex, inequality $\hat{x}_i f_i'(0, y_i) < f_i(\hat{x}_i, y_i) - f_i(0, y_i)$ holds, which implies that $f_i(\hat{x}_i, y_i) - \hat{x}_i \lambda = f_i(\hat{x}_i, y_i) + |\hat{x}_i|\lambda > f_i(0, y_i)$. Hence, $x_i^* = 0$ must hold. When $f_i'(0, y_i) < 0$, we can similarly argue that $x_i^* = 0$, completing the proof. □

Remark: For the Euclidean case $f_i(x, y) = (1/2)(x - y)^2$, equation (2.5) reduces to the familiar *soft-thresholding*: $x_i^*(\lambda) = \operatorname{sgn}(y_i) \max(|y_i| - \lambda, 0)$,

2.2 GPO with the ℓ_∞-norm

Now we consider $\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_\infty$; here we need to solve

$$\min\nolimits_{\boldsymbol{x}} \quad F(\boldsymbol{x}, \boldsymbol{y}) + \lambda \|\boldsymbol{x}\|_\infty. \tag{2.6}$$

Using Lemma 1 and introducing an auxiliary variable $t = \|\boldsymbol{x}\|_\infty$, we rewrite (2.6) as

$$\min\nolimits_{\boldsymbol{x}, t} \quad F(\boldsymbol{x}, \boldsymbol{y}) + \lambda t, \quad \text{s.t.} \quad 0 \leq s_i x_i \leq t, \tag{2.7}$$

[1] The ℓ_0-norm $\|\boldsymbol{x}\|_0$ which counts the number of nonzero entries in a vector $\boldsymbol{x}$, is actually not really a norm because it fails to satisfy the *positive homogeneity* property: $\|\alpha \boldsymbol{x}\| = |\alpha| \|\boldsymbol{x}\|$.

where $s_i = \text{sgn}(v_i)$ as before. The Lagrangian for (2.7) is

$$L(\boldsymbol{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = F(\boldsymbol{x}, \boldsymbol{y}) + \lambda t - \sum\nolimits_i \alpha_i s_i x_i + \sum\nolimits_i \beta_i (s_i x_i - t), \tag{2.8}$$

from which we get the optimality conditions ($1 \le i \le n$):

$$f_i'(x_i, y_i) - \alpha_i s_i + \beta_i s_i = 0, \quad \lambda - \sum\nolimits_i \beta_i = 0, \quad \alpha_i (s_i x_i) = 0, \quad \beta_i (s_i x_i - t) = 0, \quad \alpha_i, \beta_i \ge 0. \tag{2.9}$$

If the optimal $t = t^*$ were known, we could use the conditions (2.9) to compute the optimum x_i^* via

$$x_i^*(t^*) = s_i t^* \quad \text{if } |y_i| \ge t^*; \quad x_i^* = y_i \quad \text{otherwise.} \tag{2.10}$$

The following lemma makes this claim explicit.

Lemma 3. *The optimality conditions* (2.9) *are satisfied by the solution* (2.10).

Proof. Consider the three simple cases: (i) $y_i = 0$; (ii) $|y_i| < t^*$; and (iii) $|y_i| \ge t^*$. *Case (i)*: If $y_i = 0$, then Lemma 1-(ii) implies that $x_i = 0$.
Case (ii): If $|y_i| < t^*$, then setting $x_i^* = y_i$ satisfies (2.9) ($\alpha_i = \beta_i = 0$).
Case (iii): If $|y_i| \ge t^*$, then we have two possibilities. Either $s_i x_i < t^*$, or $s_i x_i = t^*$. For the former case $\beta_i = 0$, and we have two further possibilities: $x_i = 0$ or $|x_i| > 0$. If $|x_i| > 0$, then $\alpha_i = 0$, and (2.9) implies that $f_i'(x_i, y_i) = 0$, a contradiction because $x_i \ne y_i$. If $x_i = 0$, then (2.9) implies that $f_i'(x_i, y_i) = \alpha_i s_i$, which again cannot hold because both sides differ in signs. Thus, the only possibility that remains is $s_i x_i = t^*$. □

We now show how compute the optimum t^*, by "searching" for it. To this end, recall the optimality condition $\lambda - \sum_i \beta_i = 0$. To contribute to this sum β_i must be positive, which occurs when $s_i x_i = t$, which itself occurs when $|y_i| \ge t$. In such a case (2.9) implies $\beta_i = -s_i f_i'(x_i, y_i)$. Thus, we define the function

$$g(t) = \lambda - \sum\nolimits_i \beta_i = \lambda + \sum\nolimits_{i:|y_i| \ge t} s_i f_i'(x_i(t), y_i), \tag{2.11}$$

where $x_i(t)$ is obtained by (2.10) with t instead of t^*. This functions is a (piecewise continuous) monotonically increasing function of t with $g(0) = \lambda - \|f'(0, \boldsymbol{y})\|_1 < 0$, where $f'(0, \boldsymbol{y}) = [f_1'(0, y_1), \ldots, f_n'(y_n)]$; also note that $g(\|\boldsymbol{y}\|_\infty) = \lambda > 0$. The former inequality holds, otherwise $\|f'(0, \boldsymbol{y})\|_1 \le \lambda$ implies $\boldsymbol{x}^* = 0$ to be the (trivial) solution. Thus, we conclude that g is invertible, and has a unique root in the range $[0, \|\boldsymbol{y}\|_\infty]$, which can be found in (effectively) linear time using bisection. Note this unique root, say $\hat{t}$, must equal t^*, because $g(t^*) = 0$ as $x_i(t^*) = s_i t^*$.

For certain parameter ranges we can exploit the structure of $g(t)$ to narrow down its root t^* faster. To that end, notice that g is piecewise continuous, and its pieces are defined by the 'breakpoints' $|y_1|, |y_2|, \ldots, |y_n|$. If we sort the breakpoints in decreasing order, say $\pi_1 \ge \pi_2 \ge \cdots \pi_n$, then we just need to find an index ρ for which $g'(|\pi_\rho|) > 0$ but $g'(|\pi_{\rho+1}|) < 0$; this ρ then yields the desired interval bracketing t^*. The details are summarized in Algorithm 1, wherein the call to ROOT refers to any root-finder (e.g., `fzero` in MATLAB) that is invoked to compute t^* once an interval ($[\pi_\rho, \pi_{\rho+1}]$) has been determined.

Input: Vector $\boldsymbol{y}$; scalar $\lambda > 0$
Output: t^*
$\boldsymbol{\pi} \leftarrow \text{SORT}(|y_i|, \downarrow, 1 \le i \le n)$;
$\rho_l \leftarrow 1, \rho_h \leftarrow n$;
while $\rho_l \le \rho_h$ **do**
  $\rho \leftarrow \lfloor (\rho_l + \rho_h)/2 \rfloor$
  if $g'(\pi_\rho) > 0$ **then**
    $\rho_h \leftarrow \rho - 1$
  else
    $\rho_l \leftarrow \rho + 1$
  end
end
if $S \ge 0$ **then** $\rho \leftarrow \rho - 1$;
$t^* \leftarrow \text{ROOT}(g'(t), [\pi_\rho, \pi_{\rho+1}])$;

Algorithm 1: Binary-search for ρ.

Note that sorting step takes $O(n \log n)$ time, while the binary search requires $O(\log n)$ steps, each of which costs $O(\rho)$ time: so overall cost is $O(n \log n)$. Further, we observe that if $f_i'(x_i, y_i)$ is linear, then Algorithm 1 can be reimplemented as an $O(n)$ median-finding algorithm.

2.3 GPO with the $\ell_{1,\infty}$ mixed-norm

First we need to define the $\ell_{1,\infty}$ *mixed-norm*.

Definition 4 (Mixed-norm)**.** Let matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$. We *define* the $\ell_{1,\infty}$ mixed-norm of $\boldsymbol{X}$ as the ℓ_1-norm of the ℓ_∞-norms of the rows. Thus, letting $\boldsymbol{x}^i$ denote the i-th row of $\boldsymbol{x}$, we define[2] the $\ell_{1,\infty}$ norm as

$$\|\boldsymbol{X}\|_{1,\infty} = \sum\nolimits_{i=1}^{d} \|\boldsymbol{x}^i\|_\infty \tag{2.12}$$

With Definition (2.12) in hand, we see that the $\ell_{1,\infty}$-norm GPO is

$$\min_{\boldsymbol{X}} \quad F(\boldsymbol{X}, \boldsymbol{Y}) + \lambda\|\boldsymbol{X}\|_{1,\infty}. \tag{2.13}$$

Since both the norm and the objective function are separable, this problem splits into d independent subproblems:

$$\min \quad \sum\nolimits_{i=1}^{d} F(\boldsymbol{x}^i, \boldsymbol{y}^i) + \lambda \sum\nolimits_{i=1}^{d} \|\boldsymbol{x}^i\|_\infty, \tag{2.14}$$

each of which can be then solved either in $O(n)$ time (bisection) or via Algorithm 1 in $O(n \log n)$ time. Thus, the overall GPO problem can be solved in $O(dn)$ time (bisection) or $O(dn \log n)$ (Algorithm 1).

2.4 Box-constrained proximity

In some situations, one may need to consider a box-constrained version of the proximity operation (GPO). Here the optimization problem is

$$\min\nolimits_{\boldsymbol{x}} \quad F(\boldsymbol{x}, \boldsymbol{y}) + \lambda\|\boldsymbol{x}\|, \quad \text{s.t.} \quad \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \tag{2.15}$$

where $\boldsymbol{l}$, $\boldsymbol{u} \in \mathbb{R}^n$, and inequalities are componentwise. An important special case of (2.15) is the *trust-region* subproblem [5]

$$\min\nolimits_{\boldsymbol{x}} \quad \tfrac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \quad \text{s.t.} \quad \|\boldsymbol{x}\|_\infty \le \Delta. \tag{2.16}$$

This problem is a special case because the constraint $\|\boldsymbol{x}\|_\infty \le \Delta$ can be rewritten as $-\Delta \le x_i \le \Delta$ for $1 \le i \le n$.

Because of separability, one sees that Problem (2.15) is solved by following obvious steps:

1. Compute unconstrained solution $\boldsymbol{x}$ via (2.5);
2. Ensure feasibility by setting $\boldsymbol{x} \leftarrow \min(\max(\boldsymbol{x}, \boldsymbol{l}), \boldsymbol{u})$

Example 5 (ℓ_1-proximity)**.** For (2.15) with $\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_1$, we obtain the solution

$$x_i^* = \min(\max(\hat{x}_i, l_i), u_i),$$

where $\hat{x}_i = \operatorname{sgn}(y_i)\max(|y_i| - \lambda, 0)$.

3 Efficiently solving (GPP)

Now we come to the generalized projection problems. These seem to be slightly more difficult than their proximity counterparts (intuitively because GPP can be cast as a GPO task, if we knew the "true" Lagrange multiplier). However, it turns out that one can still solve GPP efficiently. Details follow.

3.1 GPP with ℓ_1-norm

Lemma 1 provides one method for easily tackling the non-differentiability of the constraints, and provides a method to reduce the problem to a standard setup where one only needs to compute a single Lagrange multiplier.

Notice that if $\|\boldsymbol{y}\|_1 \le \gamma$, then $\boldsymbol{x} = \boldsymbol{y}$ solves (GPO), so we assume that $\|\boldsymbol{y}\|_1 > \gamma$. Then, the optimal $\boldsymbol{x}^*$ must satisfy $\|\boldsymbol{x}^*\|_1 = \gamma$, or equivalently, by Lemma 1, $\boldsymbol{s}^T\boldsymbol{x}^* = \gamma$, and $s_i x_i^* \ge 0$. Under these constraints, and assuming strong-duality, the optimal $\boldsymbol{x}^*$ is given by

$$\boldsymbol{x}^* = \quad \operatorname{argmin}\nolimits_{\forall i, s_i x_i \ge 0} \quad L(\boldsymbol{x}, \theta^*) = F(\boldsymbol{x}, \boldsymbol{y}) + \theta^*(\boldsymbol{s}^T\boldsymbol{x} - \gamma), \tag{3.1}$$

[2] Definition (2.12) can be generalized in numerous ways, e.g., by (i) computing an ℓ_p norm of ℓ_q norms (ii) letting each $\boldsymbol{x}^i$ be of a different length; (iii) computing a possibly different ℓ_q-norm for each $\boldsymbol{x}^i$; and (iv) grouping components of $\boldsymbol{x}$ into arbitrary overlapping vectors and then computing mixed-norms. Generalization (iii) is discussed in [9] for studying ℓ_p-nested symmetric distributions, while (iv) is discussed in [11] under the name Composite Absolute Penalties.

where $L(\boldsymbol{x}, \theta)$ is the partial Lagrangian, and θ^* the optimal dual-solution. Since $F(\boldsymbol{x}, \boldsymbol{y})$ is separable, (3.12) reduces to n sub-problems of the form

$$x_i^* = \text{argmin}_{s_i x_i \geq 0} \, f_i(x_i, y_i) + \theta^* s_i x_i, \qquad \text{for } 1 \leq i \leq n. \tag{3.2}$$

Hence, if we knew θ^*, we could solve (3.2) by simply invoking (2.5) with $\lambda = \theta^*$. Thus, all that remains is to compute the optimum value θ^*. For this we invoke the constraint $\|\boldsymbol{x}^*\|_1 = \sum_i s_i x_i^* = \gamma$, and use it to define the function[3]

$$g'(\theta) = -\gamma + \sum\nolimits_{i=1}^{n} s_i x_i^*(\theta), \tag{3.3}$$

Now define $\theta_{\max} = \max_{1 \leq i \leq n} |f'(0, y_i)|$, and consider the interval $[0, \theta_{\max}]$. On this interval g' is continuous and also changes sign since $g'(0) = \|\boldsymbol{y}\|_1 - \gamma > 0$ and $g'(\theta_{\max}) = -\gamma < 0$. Thus, g' has a root in this interval. Moreover, since g' is monotonic, this root is unique. Let this root be $\hat{\theta}$. But the solution (2.5) implies that even $g'(\theta^*) = 0$. Thus, θ^* must equal $\hat{\theta}$, where the latter can be computed using a root-finding method for (3.3).

Complexity: Assuming the root-finding procedure takes I_R iterations (e.g., $I_R = \log(\theta_{\max}/\epsilon)$ for bisection) to solve (3.3) within ϵ accuracy, the overall complexity of computing $\boldsymbol{x}^*$ is easily seen to be $O(I_R \cdot n)$: theoretically *linear* in the problem size, assuming I_R to be constant.

Binary-search: Can we further speed up the method suggested in the previous section? The answer is 'yes,' because the empirical performance of a root-finding method can be significantly improved if we can narrow down the interval to which θ^* belongs. In fact we already have all the ingredients for determining such an interval thanks to our identification of the solution via (2.5) and (3.3). The general idea has appeared previously in several contexts[4], though, to our knowledge, *only* for differentiable constraints. Below we explain the details of the faster method that is based on deriving a tighter interval bracketing θ^*.

To determine an interval bracketing θ^* we exploit the structure of $g'(\theta)$. This function is piecewise continuous with 'breakpoints' $\theta_1, \ldots, \theta_n$, which, as per (2.5) are given by $\theta_i = |f_i'(0, y_i)|$, for $1 \leq i \leq n$. But $g'(\theta)$ is also monotonically decreasing, so if we sort the breakpoints in decreasing order, then we just need to find an index ρ for which $g'(\theta_\rho) < 0$ but $g'(\theta_{\rho+1}) > 0$; this ρ then yields the desired interval $(\theta_{\rho+1}, \theta_\rho)$ containing θ^*. The details are summarized in Algorithm 2, wherein the call to ROOT refers to any root-finder (e.g., `fzero` in MATLAB) that is invoked to compute $\theta^* \in (\theta_{\rho+1}, \theta_\rho)$.

Input: Vector $\boldsymbol{y}$; scalar $\gamma > 0$;
Output: θ^*
$\boldsymbol{\theta} \leftarrow \text{SORT}(|f_i'(0, y_i)|, \downarrow, 1 \leq i \leq n)$;
$\rho_l \leftarrow 1$, $\rho_h \leftarrow n$;
while $\rho_l \leq \rho_h$ **do**
  $\rho \leftarrow \lfloor (\rho_l + \rho_h)/2 \rfloor$
  if $g'(\theta_\rho) > 0$ **then**
    $\rho_h \leftarrow \rho - 1$
  else
    $\rho_l \leftarrow \rho + 1$
  end
end
if $S \geq 0$ **then** $\rho \leftarrow \rho - 1$;
$\theta^* \leftarrow \text{ROOT}[g'(\theta), (\theta_{\rho+1}, \theta_\rho)]$;

Algorithm 2: Binary-search for ρ

3.2 GPP with ℓ_∞-norm

GPP with ℓ_∞ requires solving

$$\min \quad F(\boldsymbol{x}, \boldsymbol{y}) \quad \text{s.t.} \quad \|\boldsymbol{x}\|_\infty \leq \gamma. \tag{3.4}$$

Once again Lemma 1 allows us to replace (3.4) by

$$\min \quad F(\boldsymbol{x}, \boldsymbol{y}) \quad \text{s.t.} \quad 0 \leq s_i x_i \leq \gamma. \tag{3.5}$$

But we have already used the machinery needed for solving (3.5), namely when solving (2.7), whereby

$$x_i^* = s_i \gamma, \quad \text{if} \quad |y_i| > \gamma, \quad \text{and} \quad x_i^* = y_i, \quad \text{otherwise}. \tag{3.6}$$

[3] This function is merely the derivative of the dual function $g(\theta) = -\theta\gamma + \sum_i \min_{s_i x_i \geq 0} f_i(x_i, y_i)$.

[4] In fact, the author "discovered" this general idea himself while writing this paper, but fortunately before submitting the paper, recognized the "discovery" to be a rediscovery thanks to the useful survey by Patriksson [7].

3.3 GPP with $\ell_{1,\infty}$-norm

The $\ell_{1,\infty}$ GPP setup is slightly more involved. Previously somewhat complicated approaches have been proposed for the Euclidean projection [8]. We present a much simpler, and more general approach, which essentially follows from a combination of the ℓ_∞-norm GPO with our ℓ_1-norm GPP solution. Specific details are outlined below.

For the $\ell_{1,\infty}$-norm GPP, we have as input a matrix $\boldsymbol{Y} \in \mathbb{R}^{d\times n}$, and the task is to solve

$$\min \quad F(\boldsymbol{X}, \boldsymbol{Y}) \quad \text{s.t.} \quad \|\boldsymbol{X}\|_{1,\infty} \leq \gamma. \tag{3.7}$$

As before, we assume $\|\boldsymbol{Y}\|_{1,\infty} > \gamma$, so that the inequality constraint gets replaced by the equality $\|\boldsymbol{X}\|_{1,\infty} = \gamma$. Now introduce the Lagrangian $L(\boldsymbol{X}, \theta)$ so that we have (assuming strong-duality)

$$\boldsymbol{X}^* = \operatorname{argmin}_{\boldsymbol{X}} \quad L(\boldsymbol{X}, \theta^*) = F(\boldsymbol{X}, \boldsymbol{Y}) + \theta^*(\|\boldsymbol{X}\|_{1,\infty} - \gamma), \tag{3.8}$$

where θ^* is the optimal value of the dual variable corresponding to the $\ell_{1,\infty}$ constraint. Note that for a given value θ, we can compute a corresponding optimal solution $\boldsymbol{X}(\theta)$ by solving

$$\boldsymbol{X}(\theta) = \operatorname{argmin}_{\boldsymbol{X}} \quad F(\boldsymbol{X}, \boldsymbol{Y}) + \theta\|\boldsymbol{X}\|_{1,\infty}.$$

This problem is nothing but the $\ell_{1,\infty}$-norm version of GPO (2.13), and can be thus solved efficiently. All that remains is to then determine the optimal value θ^*. This can be done, as for the ℓ_1-GPP case by defining an associated (derivative of the dual) function

$$g'(\theta) = -\gamma + \|\boldsymbol{X}(\theta)\|_{1,\infty}. \tag{3.9}$$

It is easy to see that g is a piecewise continuous and monotonically decreasing function that changes sign in the interval $[0, \theta_{\max}]$ (since $g(0) = \|\boldsymbol{Y}\|_{1,\infty} - \gamma > 0$, and $g(\theta_{\max}) = -\gamma < 0$), so it must have a unique root in this interval. This root can be found by bisection, which requires $I_R = O(\log \theta_{\max}/\epsilon)$ iterations to converge to a solution of accuracy ϵ. Since computing $\boldsymbol{X}(\theta)$ requires $O(nd)$ time, the overall projection can be computed linear (in the input size) time.

What remains to be computed is the upper bound $\theta_{\max}$. For this, consider an arbitrary subproblem of (3.8)

$$\boldsymbol{x}(\theta) = \operatorname{argmin}_{\boldsymbol{x}} \quad F(\boldsymbol{x}, \boldsymbol{y}) + \theta\|\boldsymbol{x}\|_\infty. \tag{3.10}$$

For (3.10) too, we follow an argument similar to Lemma 2 and conclude that the optimal solution $\boldsymbol{x}^*(\theta)$ satisfies

$$\boldsymbol{x}^*(\theta) = 0, \quad \text{if} \quad \|f'(0, \boldsymbol{y})\|_1 \leq \theta, \tag{3.11}$$

which implies that

$$\theta_{\max} = \max_{1\leq i\leq d} \|f'(0, \boldsymbol{y}^i)\|_1, \tag{3.12}$$

where $\boldsymbol{y}^i$ denotes the i-th row of the matrix $\boldsymbol{Y}$.

3.3.1 Implication for the Euclidean case

Our analysis above has a direct implication for the Euclidean case. Previously, Quattoni et al. [8] presented a nontrivial algorithm based on median-finding and merge-sort to compute the Euclidean projection

$$\min \quad \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \|\boldsymbol{X}\|_{1,\infty} \leq \gamma. \tag{3.13}$$

Their algorithm had a running time complexity of $O(nd \log nd)$ for input matrices $\boldsymbol{Y} \in \mathbb{R}^{d\times n}$. Our analysis above immediately yields a (weakly) *linear-time* algorithm. We conjecture that a strongly linear time algorithm does not exist for this problem.[5]

Our observation is not only theoretically interesting, because previously reported algorithms for (3.13) were not linear, but also because it immediately leads to actual improvement in running times. The improvement is also nontrivial, especially when viewed in light of the linear running time required by the GPO version. From a computational perspective, even more important is the decomposition offered by our approach. The overall

[5] Our conjecture is grounded in the observation made in [10, §7.2].

projection task is reduced to a $I_R d$ (recall I_R is number of bisection iterations) calls to individual projection tasks of the form

$$\min \quad \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \theta\|\boldsymbol{x}\|_\infty, \tag{3.14}$$

whose solution can be computed using *Moreau's decomposition* (see [3] for a proof). Formally, the solution $\boldsymbol{x}^*$ to (3.14) is given by

$$\boldsymbol{x}^* = \boldsymbol{y} - \hat{\boldsymbol{x}}(\theta), \tag{3.15}$$

where $\hat{\boldsymbol{x}}(\theta)$ denotes the ℓ_1-norm projection: $P_{\|\cdot\|_1 \le \theta}(\boldsymbol{y})$. This projection can be computed by using the specialized root-finding procedure of [6], or simply the median-finding algorithm of [4]. We note in passing that one could also invoke a "pegging" method (see [7]), as for appropriate parameter ranges it has been noted to be the fastest.

4 Further extensions

Numerous useful extensions to both (GPO) and (GPP) can be considered. We list them below in roughly increasing order of difficulty:

1. using general, differentiable, (separable) convex functions (this extension is minor);
2. deriving variable fixing (aka pegging) methods (slightly tedious);
3. accelerating the root-finding steps (challenging); and
4. allowing non-separable norms: e.g., ℓ_2-norm, nested-norms [9], or composite absolute penalties [11] (hard).

5 Numerical Results

We show numerical results for the following two problems:

1. ℓ_1-norm generalized projection; and
2. $\ell_{1,\infty}$-norm Euclidean projections.

Notice that ℓ_1-GPO is a subproblem of ℓ_1-GPP, while $\ell_{1,\infty}$-GPP reduces to solving ℓ_∞-GPO problems: thus, for these problems we omit verbose experimental details. Further since computing ℓ_∞-norm generalized proximity (Algorithm 1) is similar to ℓ_1-GPP (Algorithm 2), we present results only for the latter. Furthermore, to highlight the merits of our approach for $\ell_{1,\infty}$-projections, we compare it against the $\ell_{1,\infty}$ (Euclidean) projection algorithm of Quattoni et al. [8].

Experimental setup. For all our experiments with non-Euclidean functions, we tested our algorithms with the following choices of F (suitably extended to matrices when needed):

1. ℓ_4^4 *distance*, $F(\boldsymbol{x},\boldsymbol{y}) = \sum_i (1/4)|x_i - y_i|^4$—note, since $F(-\boldsymbol{x},-\boldsymbol{y}) = F(\boldsymbol{x},\boldsymbol{y})$, if we wish we may assume $\boldsymbol{y} \ge 0$ without loss of generality;
2. *Bregman divergence* generated by $(1/4)|x_i|^4$, so $F(\boldsymbol{x},\boldsymbol{y}) = \sum_i (1/4)(|x_i|^4 - |y_i|^4) - \operatorname{sgn}(y_i)|y_i|^3(x_i - y_i)$—in this case too, $F(-\boldsymbol{x},-\boldsymbol{y}) = F(\boldsymbol{x},\boldsymbol{y})$;
3. *Bregman divergence* generated by
$$\varphi(x) = \begin{cases} (1/5)x^5, & x \ge 0, \\ (1/3)|x|^3 & x < 0. \end{cases}$$
Note that in this case $F(-\boldsymbol{x},-\boldsymbol{y}) \ne F(\boldsymbol{x},\boldsymbol{y})$.

For all our tests, we used MATLAB to implement Algorithm 2, as well as for associated the root-finding.

5.1 ℓ_1-GPP

We show numerical results of solving the ℓ_1-GPP problem

$$\min \quad F(\boldsymbol{x},\boldsymbol{y}) \quad \text{s.t.} \quad \|\boldsymbol{x}\|_1 \le \gamma.$$

5.1.1 Experiment 1: Scalability

We first show running time results for computing the generalized projections for $\boldsymbol{y} \in \mathbb{R}^n$, where n of ranges from 10^1 to 10^7 (in powers of 10). For this experiment, we generated random vectors $\boldsymbol{y}$ with entries drawn from the normal distribution $\mathcal{N}(0, 1)$. To simulate various sparsity level requirements on the solution, we used $\gamma = \alpha\|\boldsymbol{y}\|_1$, where $\alpha \in \{.01, .05, j/10 : 1 \leq j \leq 9\}$.

Figure 1 shows the time to compute projections for solving (3.3) with each of the three choices of $F(\boldsymbol{x}, \boldsymbol{y})$; the computation was done using MATLAB's `fzero` function, hereafter called A0. The plots show the (expected) linear scaling as the problem size is varied; the runtimes increase with increasing γ, which is also easily explained because larger γ means more nonzero entries in $\boldsymbol{x}^*$, whereby computing $g'(\theta)$ becomes more expensive. In all graphs, each of the plotted lines shows computation times averaged over 5 runs.

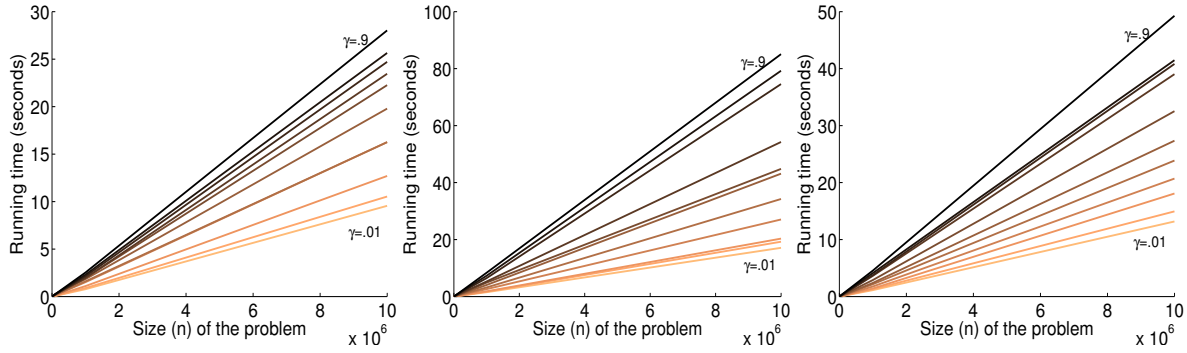

Figure 1: Total time to compute projection using `fzero` on (3.3) with $[0, \theta_{\max}]$ as the interval. The three plots show running times for the three different choices of $F(\boldsymbol{x}, \boldsymbol{y})$ as n increases from 10^1 to 10^7 and as γ is varied ($\gamma = .01\|\boldsymbol{y}\|_1$ for the lowest line, going up to $\gamma = .9\|\boldsymbol{y}\|_1$ for the uppermost line). The plot shows that the runtimes scale linearly and increase with increasing γ.

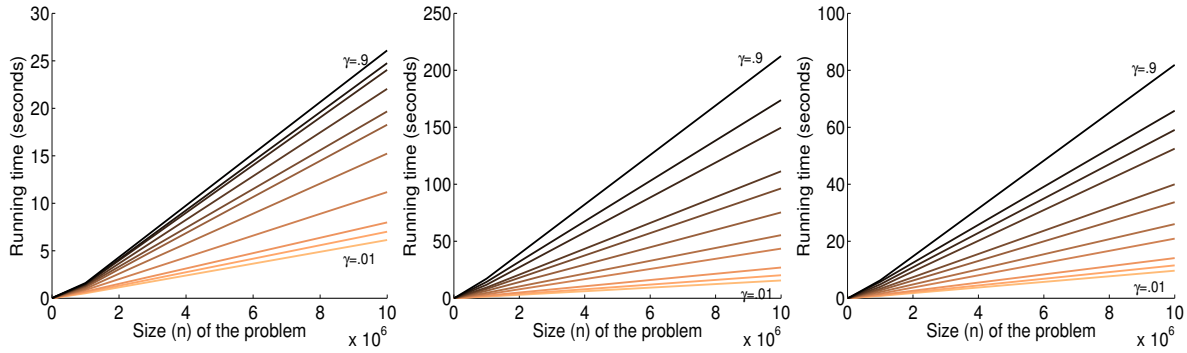

Figure 2: Total time to compute projection using Algorithm 2 for the same setup as in Figure 1. The plots show an empirical linear runtime, and with increasing γ the running times worsen.

Figure 2 reports the same experiment as in Figure 1, but this time with Algorithm 2, hereafter called A1. We observe that on a broad range of γ values, for the first function (first plot), A1 is as fast as A0, while for the other two choices of F (second and third plots) and larger values of γ, A1 takes about twice as long as A0. This behavior is expected, as the binary-search requires computing the function $g'(\theta_\rho)$ several times, and when the optimal ρ is large, a high overhead is incurred. The immediate question then is: when is A1 useful? If we look closely at the plots in Figures 1 and 2, we see that for small values of γ, A1 indeed outperforms A0. Let us take a closer look at this difference.

5.1.2 Experiment 2: Effect of tighter interval

We now retain the experimental setting of the previous section, but proceed to look more closely at the impact of a tighter interval used by Algorithm 2. We used $\gamma = \alpha\|\boldsymbol{y}\|_1$, where $\alpha \in \{.01j : 1 \leq j \leq 20\}$. From the plots one sees that for the first function A1 runs approximately twice as fast as A0 across the entire range of γ. For the

second and third function, A1 runs faster than A0 for $\gamma \leq .1\|\boldsymbol{y}\|_1$. Somewhat surprisingly, for $n = 10^7$ (second row, last plot), the overhead incurred by A1 makes it run slower than A0.

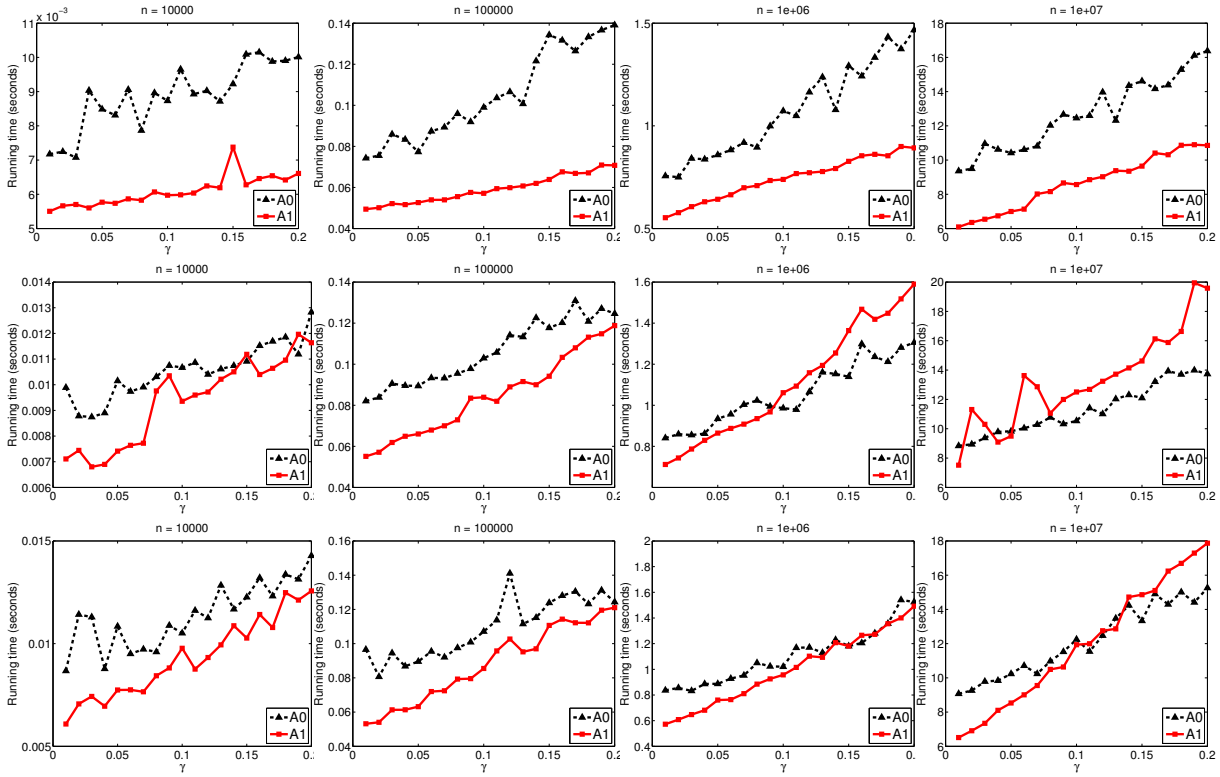

Figure 3: Running time comparisons for small values of γ between A0 and A1, where A0 refers to root-finding with $[0, \theta_{\max}]$ as the interval, and A1 refers to Algorithm 2. The values of γ range from $.01\|\boldsymbol{y}\|_1$ to $.2\|\boldsymbol{y}\|_1$. The three rows of the plot correspond to the three different choices of $F(\boldsymbol{x}, \boldsymbol{y})$.

5.2 Euclidean projections onto the $\ell_{1,\infty}$-ball

Here we compare the projection algorithm of Quattoni et al. [8] against our method from Section 3.3.1. Our implementation used an adaption of the median-finding algorithm of Duchi et al. [4] as a subroutine. We note that our algorithm could be further sped up, since in [6] the median-based approach is reported to be slower than a bisection-based method.

We obtained the $\ell_{1,\infty}$-projection code of Quattoni et al. [8] from the first author's webpage; this code is in C (MATLAB mex-file), and we refer to it as AP in the experiments below. We implemented our projection code in C (as a MATLAB mex-file), and this is referred to as SP in our experiments below.

We report results on the following two *dense* matrices:

1. $\boldsymbol{Y}_1 \in \mathbb{R}^{500\times 5000}$ with varying γ; and
2. $\boldsymbol{Y}_2 \in \mathbb{R}^{10000\times 3000}$ with varying γ.

Matrix $\boldsymbol{Y}_1$ is more favorable to AP (as the number of groups (rows) is much smaller than the number of columns), while the matrix $\boldsymbol{Y}_2$ stresses both AP and SP owing to its large number of groups (rows). In fact, $\boldsymbol{Y}_2$ is particularly challenging for SP, as SP is iterative and requires solving 10,000 subproblems as each major iteration.

Tables 1 and 2 present running time and accuracy results (accuracy is defined as the constraint violation: $|\gamma - \|\boldsymbol{X}\|_{1,\infty}|$, where $\boldsymbol{X}$ is the estimated solution). We remark that the corresponding objective function values attained by AP and SP generally agreed with each other to a relative error of 10^{-8} (the worst agreement had relative error of 10^{-3}).

From Table 1 it is clear why we called $\boldsymbol{Y}_1$ to be favorable to AP: this algorithm is at worst 10 times slower than our method, and at best slightly faster, though in regimes where sparsity is *low*. Furthermore, AP yields solutions

$\frac{\gamma}{\|\boldsymbol{Y}_1\|_{1,\infty}}$	AP Time (s)	SP Time (s)	Speedup	AP accuracy	SP accuracy
.01	17.87	1.71	10.42	3.89E-11	1.78E-14
.05	15.90	1.60	9.93	3.62E-11	2.71E-12
.10	13.54	1.60	8.44	3.65E-11	2.27E-13
.20	9.41	1.79	5.26	1.51E-11	3.41E-13
.30	6.26	1.96	3.20	5.46E-12	2.16E-12
.40	4.18	1.97	2.12	6.71E-12	4.55E-13
.50	3.03	2.15	1.41	2.05E-12	5.24E-10
.60	2.49	2.51	0.99	6.14E-12	2.05E-12
.70	2.24	2.69	0.83	7.05E-12	4.55E-13

Table 1: Running time and accuracy $|\gamma - \|\boldsymbol{X}\|_{1,\infty}|$ comparison between AP and SP on $\boldsymbol{Y}_1$

less accurate than SP, though both of them satisfy the constraint to high accuracy. Hence, for all practical purposes, even for the "favorable" case, we recommend using SP.

Table 2 highlights the setting where the AP method suffers, namely, the setting with a very large number of groups (rows) for the mixed-norm. Here, AP is at worst 161 times slower, while being overall significantly less accurate than SP. Thus, when the number of groups is large, SP is an even better choice. However, we also note that from both tables, it seems that for the non-sparse setting, AP competes with SP. But overall, for the typical regimes of interest, SP should be used for computing the projections.

$\frac{\gamma}{\|\boldsymbol{Y}_2\|_{1,\infty}}$	AP Time (s)	SP Time (s)	Speedup	AP accuracy	SP accuracy
0.01	3812.35	24.13	157.97	4.50E-09	1.48E-12
0.05	3344.85	20.79	160.90	4.82E-09	5.91E-12
0.10	2784.16	19.09	145.84	5.19E-09	1.46E-11
0.20	1807.52	21.22	85.19	3.25E-09	3.64E-12
0.30	1052.59	21.65	48.62	2.66E-09	3.27E-09
0.40	551.83	23.68	23.30	1.25E-09	7.28E-12
0.50	266.14	25.88	10.28	6.66E-10	2.87E-09
0.60	122.04	32.43	3.76	1.42E-09	4.73E-11
0.70	60.30	30.23	1.99	4.62E-10	6.95E-10

Table 2: Running time and accuracy ($|\gamma - \|\boldsymbol{X}\|_{1,\infty}|$) comparison between AP and SP on $\boldsymbol{Y}_2$

6 Conclusions

We discussed generalized proximity and projection operators. We derived efficient root-finding procedures for applying these operators for ℓ_1, ℓ_∞, and $\ell_{1,\infty}$-norms. While analyzing the $\ell_{1,\infty}$-case we also noted that our approach can be used to speed up the Euclidean projection too. Our experiments exhibited the behavior of our generalized ℓ_1-norm proximity algorithm. The practical benefit of our approach was then further highlighted by comparing it against the $\ell_{1,\infty}$-norm projection algorithm of [8], where we observed speedups of up to 160 times. Several aspects in which our work could be extended remain open, and we will explore some of these in the future.

References

[1] L. M. Bregman. The relaxation method of finding the common point of convex sets and its applications to the solution of problems in convex programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967. 1

[2] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997. 1

[3] P. L. Combettes and V. R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling and Simulation*, 4(4):1168–1200, 2005. 1, 7

[4] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient Projections onto the ℓ_1-Ball for Learning in High Dimensions. In *ICML*, 2008. 7, 9

[5] D. Kim, S. Sra, and I. S. Dhillon. A scalable trust-region algorithm with application to mixed-norm regression. In *Int. Conf. Machine Learning (ICML)*, 2010. *Submitted*. 4

[6] J. Liu and J. Ye. Efficient euclidean projections in linear time. In *Int. Conf. Machine Learning*, pages 657–664, 2009. 7, 9

[7] M. Patriksson. A survey on a classic core problem in operations research. Technical Report 2005:33, Department of Mathematical Sciences, Chalmers University of Technology and Göteborg University, Oct. 2005. 1, 5, 7

[8] A. Quattoni, X. Carreras, M. Collins, and T. Darrell. An Efficient Projection for $\ell_{1,\infty}$ Regularization. In *ICML*, 2009. 6, 7, 9, 10

[9] F. Sinz, E. Simoncelli, and M. Bethge. Hierarchical modeling of local image features through lp nested symmetric distributions. In *Adv. Neural Inf. Proc. Syst.*, 2009. 4, 7

[10] J. Yu, S. Vishwanathan, S. Günter, and N. N. Schraudolph. A Quasi-Newton Approach to Nonsmooth Convex Optimization Problems in Machine Learning. *J. Mach. Learn. Res.*, 11:1145–1200, Mar. 2010. 6

[11] P. Zhao, G. Rocha, and B. Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Stat.*, 37(6A):3468–3497, 2009. 4, 7