An $O(n\log(n))$ Algorithm for Projecting Onto the Ordered Weighted ℓ_1 Norm Ball

Damek Davis

June 26, 2015

Abstract The ordered weighted ℓ_1 (OWL) norm is a newly developed generalization of the Octogonal Shrinkage and Clustering Algorithm for Regression (OSCAR) norm. This norm has desirable statistical properties and can be used to perform simultaneous clustering and regression. In this paper, we show how to compute the projection of an n-dimensional vector onto the OWL norm ball in $O(n\log(n))$ operations. In addition, we illustrate the performance of our algorithm on a synthetic regression test.

1 Introduction

Sparsity is commonly used as a model selection tool in statistical and machine learning problems. For example, consider the following Ivanov regularized (or constrained) regression problem:

$$\underset{x\in\mathbf{R}^n}{\text{minimize}}\ \frac{1}{2}\|Ax-b\|^2 \qquad \text{subject to: } \|x\|_0 \leq \varepsilon. \tag{1.1}$$

where $m, n > 0$ are integers, $\varepsilon > 0$ is a real number, $A \in \mathbf{R}^{m\times n}$ and $b \in \mathbf{R}^m$ are given, and $\|x\|_0$ is the number of nonzero components of a vector $x \in \mathbf{R}^n$. Solving (1.1) yields the "best" predictor x with fewer than ε nonzero components. Unfortunately, (1.1) is nonconvex and NP hard [12]. Thus, in practice the following convex surrogate (LASSO) problem is solved instead (see e.g., [6]):

$$\underset{x\in\mathbf{R}^n}{\text{minimize}}\ \frac{1}{2}\|Ax-b\|^2 \qquad \text{subject to: } \|x\|_1 \leq \varepsilon \tag{1.2}$$

where $\|x\|_1 = \sum_{i=1}^n |x_i|$.

Recently, researchers have moved beyond the search for sparse predictors and have begun to analyze "group-structured" sparse predictors [1]. These models are motivated by a deficiency of (1.1) and (1.2): they yield a predictor with a small number of nonzero components, but they fail to identify and take into account

This work is supported in part by NSF grant DMS-1317602.

D. Davis
Department of Mathematics, University of California, Los Angeles
Los Angeles, CA 90025, USA
E-mail: damek@ucla.edu

similarities between features. In other words, group-structured predictors simultaneously cluster and select groups of features for prediction purposes. Mathematically, this behavior can be enforced by replacing the ℓ_0 and ℓ_1 norms in (1.1) and (1.2) with new regularizers. Typical choices for group-structured regularizers include the Elastic Net [19] (EN), Fused LASSO [15], Sparse Group LASSO [14], and Octogonal Shrinkage and Clustering Algorithm for Regression [5] (OSCAR). The EN and OSCAR regularizers have the benefit of being invariant under permutations of the components of the predictor and do not require prior specification of the desired groups of features (when a clustering is not known *a priori*). However, OSCAR has been shown to outperform EN regularization in feature grouping [5,18]. This has motivated the recent development of the ordered weighted ℓ_1 norm [4,16] (OWL) (see (2.1) below), which includes the OSCAR, ℓ_1, and ℓ_∞ norms as a special case.

Related work. Recently, the paper [17] investigated the properties of the OWL norm, discovered the atomic norm characterization of the OWL norm, and developed an $O(n\log(n))$ algorithm for computing its proximal operator (also see [4] for the computation of the proximal operator). Using the atomic characterization of the OWL norm, the paper [17] showed how to apply the Frank-Wolfe conditional gradient algorithm (CG) [8] to the Ivanov regularized OWL norm regression problem. However, when more complicated, and perhaps, nonsmooth data fitting and regularization terms are included in the Ivanov regularization model, the Frank-Wolfe algorithm can no longer be applied. If we knew how to quickly project onto the OWL norm ball, we could apply modern proximal-splitting algorithms [7], which can perform better than CG for OWL problems [17], to get a solution of modest accuracy quickly. Note that [17] proposes a root-finding scheme for projecting onto the OWL norm ball, but it is not guaranteed to terminate at an exact solution in a finite number of steps.

Contributions. The paper introduces an $O(n\log(n))$ algorithm and MATLAB code for projecting onto the OWL norm ball (Algorithm 1). Given a norm $f : \mathbf{R}^n \to \mathbf{R}_+$, computing the proximal map

$$\mathbf{prox}_f(z) := \operatorname*{arg\,min}_{x\in\mathbf{R}^n} f(x) + \frac{1}{2}\|x-z\|^2$$

can be significantly easier than evaluating the projection map

$$P_{\{x\in\mathbf{R}^n \mid f(x)\le\varepsilon\}}(z) := \operatorname*{arg\,min}_{f(x)\le\varepsilon} \frac{1}{2}\|x-z\|^2. \tag{1.3}$$

In this paper, we devise an $O(n\log(n))$ algorithm to project onto the OWL norm ball that matches the complexity (up to constants) of the currently best performing algorithm for computing the proximal operator of the OWL norm. The algorithm we present is the first known method that computes the projection in a finite number of steps, unlike the existing root-finding scheme [17], which only provides an approximate solution in finite time. In addition, using duality (see (2.4)) we immediately get an $O(n\log(n))$ algorithm for computing the proximity operator of the dual OWL norm (see (2.3)).

The main bottleneck in evaluating the proximity and projection operators of the OWL norm arises from repeated partial sortings and averagings. Unfortunately, this seems unavoidable because even evaluating the OWL norm requires sorting a (possibly) high-dimensional vector. This suggests that any OWL norm projection algorithm requires $\Omega(n\log(n))$ operations in the worst case.

Organization. The OWL norm is introduced in Section 2. In Section 2.1, we reduce the OWL norm projection to a simpler problem (Problem 2.1). In Section 2.2, we introduce crucial notation and properties for working with partitions. In Section 3, we introduce the 6 alternatives (Proposition 3.1), which directly lead to our main algorithm (Algorithm 1) and its complexity (Theorem 3.2). Finally, in Section 4, we illustrate the performance of our algorithm on a synthetic regression test.

2 Basic Properties and Definitions

We begin with the definition of the OWL norm.

Definition 2.1 (The OWL Norm) Let $n \geq 1$ and let $w \in \mathbf{R}^n_+$ satisfy $w_1 \geq w_2 \geq \cdots w_n \geq 0$ with $w \neq 0$. Then for all $z \in \mathbf{R}^n$, the OWL norm $\Omega_w : \mathbf{R}^n \to \mathbf{R}_+$ is given by

$$\Omega_w(x) := \sum_{i=1}^{n} w_i |x|_{[i]} \tag{2.1}$$

where for any $x \in \mathbf{R}^n$, the scalar $|x|_{[i]}$ is the i-th largest element of the magnitude vector $|x| := (|x_1|, \ldots, |x_n|)^T$. For all $\varepsilon > 0$, let $\mathcal{B}(w, \varepsilon) := \{x \in \mathbf{R}^n \mid \Omega_w(x) \leq \varepsilon\}$ be the closed OWL norm ball of radius ε.

Notice that when w is a constant vector, we have $\Omega_w \equiv w_1 \|\cdot\|_1$. On the other hand, when $w_1 = 1$ and $w_i = 0$ for $i = 2, \ldots, n$, we have $\Omega_w \equiv w_1 \|\cdot\|_\infty$. Finally, given nonnegative real numbers μ_1 and μ_2, for all $i \in \{1, \ldots, n\}$, define $w_i = \mu_1 + \mu_2(n - i)$. Then the OSCAR norm [4] is precisely:

$$\Omega_w(x) = \mu_1 \|x\|_1 + \mu_2 \sum_{i<j} \max\{|x_i|, |x_j|\}. \tag{2.2}$$

Note that Ω_w was originally shown to be a norm in [4,16]. The paper [16] also showed that the dual norm (in the sense of functional analysis) of Ω_w has the form

$$\Omega^*_w(x) = \max\{\tau_i \|x_{(i)}\|_1 \mid i = 1, \ldots, n\} \tag{2.3}$$

where $x \in \mathbf{R}^n$ and for all $1 \leq j \leq n$, $\tau_j = \left(\sum_{i=1}^{j} w_j\right)^{-1}$ and $x_{(j)} \in \mathbf{R}^j$ is a vector consisting of the j largest components of x (where size is measured in terms of magnitude). One interesting consequence of this fact is that for all $\gamma > 0$ and $z \in \mathbf{R}^n$, we have (from [2, Proposition 23.29])

$$\mathbf{prox}_{\gamma\Omega^*_w}(z) := \operatorname*{arg\,min}_{x \in \mathbf{R}^n} \left\{ \Omega^*_w(x) + \frac{1}{2\gamma}\|x - z\|^2 \right\} = z - \gamma P_{\mathcal{B}(w,1)}\left(\frac{1}{\gamma} z\right). \tag{2.4}$$

Thus, Algorithm 1 (below) also yields an $O(n\log(n))$ algorithm for evaluating $\mathbf{prox}_{\gamma\Omega^*_w}(z)$.

2.1 A Simplification of the OWL Norm Projection Problem

The following transformation (which is based on [17, Lemmas 2-4]) will be used as a preprocessing step in our algorithm. For convenience, we let $\odot : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}^n$ denote the componentwise vector product operator. Finally, for any $z \in \mathbf{R}^n$, let $\operatorname{sign}(z) \in \{-1, 1\}^n$ be the componentwise vector of signs of z (with the convention $\operatorname{sign}(0) = 1$).

Proposition 2.1 (Problem Reduction) *Let $z \in \mathbf{R}^n$, and let $Q(|z|)$ be the permutation matrix that sorts $|z|$ to be in nonincreasing order. Then*

$$P_{\mathcal{B}(w,\varepsilon)}(z) = \operatorname{sign}(z) \odot Q(|z|)^T P_{\mathcal{L}(w,\varepsilon)\cap\mathcal{T}}(Q(|z|)|z|)$$

where $\mathcal{L}(w, \varepsilon) := \{x \in \mathbf{R}^n \mid \langle w, x\rangle \leq \varepsilon\}$ and $\mathcal{T} := \{x \in \mathbf{R}^n \mid x_1 \geq x_2 \geq \cdots x_n \geq 0\}$.

Proof Note that $\Omega_w(\mathrm{sign}(z) \odot Q(|z|)x) = \Omega_w(x)$ for all $x \in \mathbf{R}^n$. Thus,

$$P_{\mathcal{B}(w,\varepsilon)}(Q(|z|)|z|) = P_{\mathcal{B}(w,\varepsilon)}(\mathrm{sign}(z) \odot Q(|z|)z) = \operatorname*{arg\,min}_{\Omega_w(x)\leq\varepsilon} \frac{1}{2}\left\|\mathrm{sign}(z) \odot Q(|z|)z - x\right\|^2$$
$$= \operatorname*{arg\,min}_{\Omega_w(x)\leq\varepsilon} \frac{1}{2}\left\|z - \mathrm{sign}(z) \odot Q(|z|)^T x\right\|^2 = \mathrm{sign}(|z|) \odot Q(|z|)P_{\mathcal{B}(w,\varepsilon)}(z).$$

Thus, we have shown that for general vectors $z \in \mathbf{R}^n$, we have $P_{\mathcal{B}(w,\varepsilon)}(z) = \mathrm{sign}(z) \odot Q(|z|)^T P_{\mathcal{B}(w,\varepsilon)}(Q(|z|)|z|)$. Finally, the result follows from the equality $P_{\mathcal{B}(w,\varepsilon)}(Q(|z|)|z|) = P_{\mathcal{L}(w,\varepsilon)\cap\mathcal{T}}(Q(|z|)|z|)$.

Thus, whenever $z \in \mathcal{T}$, projecting onto the OWL norm ball is equivalent to projecting onto the set intersection $\mathcal{L}(w,\varepsilon) \cap \mathcal{T}$:

$$P_{\mathcal{B}(w,\varepsilon)}(z) = \operatorname*{arg\,min}_{x\in\mathbf{R}^n} \frac{1}{2}\|x - z\|^2 \quad \text{subject to:} \quad \sum_{i=1}^n w_i x_i \leq \varepsilon \text{ and } x_1 \geq x_2 \geq \cdots x_n \geq 0.$$

Finally, we make one more reduction to the problem, which is based on the following simple lemma.

Lemma 2.1 *Let $z, w \in \mathcal{T}$ and suppose that $w \neq 0$. If $\langle z, w\rangle \leq \varepsilon$, then $P_{\mathcal{B}(w,\varepsilon)}(z) = z$. Otherwise, $\langle P_{\mathcal{B}(w,\varepsilon)}(z), w\rangle = \varepsilon$.*

We arrive at our final problem:

Problem 2.1 (Reduced Problem) Given $z \in \mathcal{T}$ such that $\langle z, w\rangle > \varepsilon$, find

$$x^* := \operatorname*{arg\,min}_{x\in\mathbf{R}^n} \frac{1}{2}\|x - z\|^2 \quad \text{subject to:} \quad \sum_{i=1}^n w_i x_i = \varepsilon \text{ and } x_1 \geq x_2 \geq \cdots x_n \geq 0 \tag{2.5}$$

Now define $H(w,\varepsilon) = \{x \in \mathbf{R}^n \mid \langle w, x\rangle = \varepsilon\}$. Then $x^* = P_{H(w,\varepsilon)\cap\mathcal{T}}(z)$.

The following proposition is a straightforward exercise in convex analysis.

Proposition 2.2 (KKT Conditions) *The point x^* satisfies Equation (2.5) if, and only if, there exists $\lambda^* \in \mathbf{R}_{++}$ and a vector $v^* \in \mathbf{R}^n_+$ such that*

1. $x^* \in \mathcal{T}$
2. $v_i^*(x_i^* - x_{i+1}^*) = 0$ *for* $1 \leq i < n$ *and* $v_n^* x_n = 0$;
3. $x_i^* = z_i - \lambda^* w_i + v_i^* - v_{i-1}^*$ *for* $1 \leq i \leq n$ *where* $v_0^* := 0$;
4. *and* $\langle x^*, w\rangle = \varepsilon$.

We now record the solution to (2.5) in the special case that w is the constant vector.

Proposition 2.3 (Projection Onto the Simplex [9]) *Let $\kappa > 0$ and let $\Delta(\kappa, n)$ denote the simplex $\{x \in \mathbf{R}^n \mid 0 \leq x \leq \kappa$ and $\sum_{i=1}^n x_i = \kappa\}$. Let $z, w \in \mathcal{T}$ and suppose that $w \neq 0$. In addition, suppose that $w_1 = w_2 = \cdots = w_n$. Then $x^* = P_{\Delta(\varepsilon/w_1,n)}(z)$ is the solution to Problem (2.5). In other words, we can replace the constraint $x \in \mathcal{T}$ with $x \in \mathbf{R}^n_+$ in Problem (2.5). Furthermore, $x^* = \max\{z - \lambda, 0\}$ where*

$$\lambda := \frac{\sum_{i=1}^K z_i - \varepsilon/w_1}{K} \qquad \text{and} \qquad K := \max\left\{k \mid \frac{\sum_{i=1}^k z_i - \varepsilon/w_1}{k} < z_k\right\}.$$

2.2 Partitions

Define $\mathcal{P}_n$ to be the set of partitions of $\{1,\dots,n\}$ built entirely from intervals of integers. For example, when $n=5$, the partition $\mathcal{G} := \{\{1,2\},\{3\},\{4,5\}\}$ is an element of $\mathcal{P}_5$, but $\mathcal{G}' := \{\{1,3\},\{2,4,5\}\}$ is not an element of $\mathcal{P}_5$ because $\{1,3\}$ and $\{2,4,5\}$ are not intervals. For two partitions $\mathcal{G}_1,\mathcal{G}_2 \in \mathcal{P}_n$, we say that

$$\mathcal{G}_1 \preccurlyeq \mathcal{G}_2 \text{ if for all } G_1 \in \mathcal{G}_1, \text{ there exists } G_2 \in \mathcal{G}_2 \text{ with } G_1 \subseteq G_2.$$

Note that if $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $\mathcal{G}_2 \preccurlyeq \mathcal{G}_1$, then $\mathcal{G}_1 = \mathcal{G}_2$. In addition, we have the following fact:

Lemma 2.2 *Let $\mathcal{G}_1,\mathcal{G}_2 \in \mathcal{P}_n$. If $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ and $|\mathcal{G}_1| = |\mathcal{G}_2|$, then $\mathcal{G}_1 = \mathcal{G}_2$.*

Suppose that we partition a vector $z \in \mathbf{R}^n$ into g maximal groups of nondecreasing components

$$z = (\underbrace{z_1,\dots,z_{n_1}}_{G_1(z)},\underbrace{z_{n_1+1},\dots,z_{n_2}}_{G_2(z)},\dots,\underbrace{z_{n_{g-1}+1},\dots,z_{n_g}}_{G_g(z)})^T$$

where $z_{n_j} > z_{n_j+1}$ for all $1 \le j \le g-1$, and inside the each group, z is a nondecreasing list of numbers (i.e., $z_k \le z_{k+1}$ whenever $k, k+1 \in G_j(z)$ for some $j \in \{1,\cdots,g\}$). Note that g can be 1, in which case we let $n_0 = 1$. We let

$$\mathcal{G}(z) := \{G_1(z),\dots,G_g(z)\} \in \mathcal{P}_n. \tag{2.6}$$

For example, for $z := (1,4,5,1,3)^T$, we have $\mathcal{G}(z) = \{\{1,2,3\},\{4,5\}\}$, $g=2$, $G_1(z) = \{1,2,3\}$, and $G_2(z) = \{4,5\}$. Note that when $z \in \mathcal{T}$, the vector z is constant within each group.

For simplicity, whenever x^* is a solution to (2.5), we define

$$\mathcal{G}^* := \mathcal{G}(x^*). \tag{2.7}$$

Finally, for simplicity, we will also drop the dependence of the groups on z: $G_i := G_i(z)$.

For any vector $z \in \mathbf{R}^n$ and any partition $\mathcal{G} = \{G_1,\dots,G_g\} \in \mathcal{P}_n$, define an averaged vector: for all $j = 1,\dots,g$ and $i \in G_j$, let

$$(z_{\mathcal{G}})_i := \frac{1}{|G_j|}\sum_{k\in G_j} z_k. \tag{2.8}$$

For example, for $z := (1,4,5,1,3)^T$ and $\mathcal{G} := \{\{1,2\},\{3,4\},\{5\}\}$, we have $z_{\mathcal{G}} = (5/2,5/2,3,3,5)^T$. Note that $z_{\mathcal{G}} \in \mathcal{T}$ whenever $z \in \mathcal{T}$.

The following proposition will allow us to repeatedly apply transformations to the vectors z and w without changing the solution to (2.5).

Proposition 2.4 (Increasing Partitions) *Let $z, w \in \mathcal{T}$ and suppose that $w \neq 0$.*

1. *Suppose that $\lambda^* \ge \lambda$ (where λ^* is as in Proposition 2.2). Then we have*

$$\mathcal{G}(z) \preccurlyeq \mathcal{G}(z-\lambda w) = \mathcal{G}(z_{\mathcal{G}(z-\lambda w)}) \preccurlyeq \mathcal{G}^*.$$

2. *We have $x^* = P_{H(w_{\mathcal{G}},\varepsilon)\cap\mathcal{T}}(z_{\mathcal{G}})$ whenever $\mathcal{G} \preccurlyeq \mathcal{G}^*$.*

Proof See Appendix A. □

3 The Algorithm

The following proposition is the workhorse of our algorithm. It provides a set of 6 alternatives, three of which give a solution when true; the other three allow us to update the vectors z and w so that $\mathcal{G}(z)$ strictly decreases in size, while keeping x^* fixed. Clearly, the size of this partition must always be greater than 0, which ensures that our algorithm terminates in a finite number of steps.

Proposition 3.1 (The 6 Alternatives) *Let $z, w \in \mathcal{T}$. Suppose that $w \neq 0$, that $\langle w, z\rangle > \varepsilon$, and $w = w_{\mathcal{G}(z)}$. Let*

$$r := \min\left\{ \frac{z_i - z_{i+1}}{w_i - w_{i+1}} \mid i = 1, \ldots, n-1 \right\}$$

where we use the convention that $0/0 = \infty$. Define

$$\lambda_0 := \frac{\left(\sum_{\{i|z_i > z_n\}} z_i w_i\right) - \varepsilon}{\sum_{\{i|z_i > z_n\}} w_i^2} \qquad \text{and} \qquad \lambda_1 := \frac{\langle z, w\rangle - \varepsilon}{\|w\|^2}.$$

Then $\lambda^* \geq \lambda_1$ (where λ^* is as in Proposition 2.2).

Let $n' := \min\left(\{k \mid z_k - \lambda_0 w_k < 0\} \cup \{n+1\}\right)$. Then one of the following mutually exclusive alternatives must hold:

1. *If $r = \infty$, we have $x^* = P_{\Delta(\varepsilon/w_1, n)}(z)$.*
2. *If $\lambda_1 > r$, then $\lambda^* \geq \lambda_1 > r$.*
3. *If $\lambda_1 \leq r < \infty$ and $z_n - \lambda_1 w_n \geq 0$, then $x^* = z - \lambda_1 w$.*
4. *If $\lambda_1 \leq r < \infty$, $z_n - \lambda_1 w_n < 0$ and $\lambda_0 > r$, then $\lambda^* \geq \lambda_0 > r$.*
5. *If $\lambda_1 \leq r < \infty$, $z_n - \lambda_1 w_n < 0$, $\lambda_0 \leq r$, and $n' \leq n$ with $z_{n'} = z_n$, then $x^* = \max\{z - \lambda_0 w, 0\}$.*
6. *If $\lambda_1 \leq r < \infty$, $z_n - \lambda_1 w_n < 0$, $\lambda_0 \leq r$, and $n' < n$ with $z_{n'} \neq z_n$, then $\mathcal{G}_0 \preccurlyeq \mathcal{G}^*$ where $\mathcal{G}_0 = \{G \in \mathcal{G}(z) \mid \max(G) < n'\} \cup \{\{n', \ldots, n\}\}$.*
7. *It cannot be the case that $\lambda_1 \leq r < \infty$, $z_n - \lambda_1 w_n < 0$, $\lambda_0 \leq r$, and $n' = n+1$.*

In addition, whenever $\lambda^* \geq \lambda \geq r$, we have $\mathcal{G}(z - \lambda w) \preccurlyeq \mathcal{G}^*$ and $|\mathcal{G}(z_{\mathcal{G}(z-\lambda w)})| \leq |\mathcal{G}(z)| - 1$. Similarly, when 6 holds, we have $\mathcal{G}_0 \in \mathcal{P}_n$, $\mathcal{G}(z) \preccurlyeq \mathcal{G}_0 = \mathcal{G}(z_{\mathcal{G}_0}) \preccurlyeq \mathcal{G}^*$, and $|\mathcal{G}(z_{\mathcal{G}_0})| \leq |\mathcal{G}(z)| - 1$. In particular, if $\mathcal{G}(z) = \mathcal{G}^*$, then at least one of steps 1, 3, and 5 will not fail.

Proof See Appendix B. □

We are now ready to present our algorithm. It repeatedly transforms the vectors z and w after checking whether Proposition 3.1 yields a solution to (2.5). Note that we assume the input is sorted and nonnegative. Thus to project onto the OWL ball with Algorithm 1, the preprocessing in Proposition 2.1 must be applied first. Please see Appendix C for an example of Algorithm 1.

Algorithm 1 (Algorithm to solve (2.5)) *Let $z \in \mathcal{T}$, $w \in \mathcal{T}\backslash\{0\}$, and $\varepsilon \in \mathbf{R}_{++}$.*

Initialize:

1. *$w \leftarrow w_{\mathcal{G}(z)}$;*

Repeat:

1. *Computation:*
   (a) *$r \leftarrow \min\left\{ \frac{z_i - z_{i+1}}{w_i - w_{i+1}} \mid i = 1, \ldots, n-1 \right\}$ (where $0/0 = \infty$);*

(b) Define

$$\lambda_0 \leftarrow \frac{\sum_{\{i|z_i>z_n\}} z_i w_i - \varepsilon}{\sum_{\{i|z_i>z_n\}} w_i^2} \qquad \textit{and} \qquad \lambda_1 \leftarrow \frac{\langle z, w\rangle - \varepsilon}{\|w\|^2};$$

(c) $n' \leftarrow \min\left(\{k \mid z_k - \lambda_0 w_k < 0\} \cup \{n+1\}\right)$*;*
(d) $\mathcal{G}_0(z) \leftarrow \{G \in \mathcal{G}(z) \mid \max(G) < n'\} \cup \{\{n', \ldots, n\}\}$

2. Tests:

(a) If $\langle z, w\rangle \leq \varepsilon$*, set*
i. $x^* \leftarrow z$*.*
Exit;

(b) If $r = \infty$*, set*
i. $x^* \leftarrow P_{\Delta(\varepsilon/w_1,n)}(z)$*.*
Exit;

(c) If $\lambda_1 > r$*, set*
i. $z \leftarrow z_{\mathcal{G}(z-\lambda_1 w_0)}$*;*
ii. $w \leftarrow w_{\mathcal{G}(z-\lambda_1 w_0)}$*;*
Go to step 1.

(d) If $\lambda_1 \leq r < \infty$ *and* $z_n - \lambda_1 w_n \geq 0$*, set*
i. $x^* \leftarrow z - \lambda_1 w$
Exit;

(e) If $\lambda_1 \leq r < \infty$*,* $z_n - \lambda_1 w_n < 0$ *and* $\lambda_0 > r$*, set*
i. $z \leftarrow z_{\mathcal{G}(z-\lambda_0 w_0)}$*;*
ii. $w \leftarrow w_{\mathcal{G}(z-\lambda_0 w_0)}$*;*
Go to step 1.

(f) If $\lambda_1 \leq r < \infty$*,* $z_n - \lambda_1 w_n < 0$*,* $\lambda_0 \leq r$*, and* $n' \leq n$ *with* $z_{n'} = z_n$*, set*
i. $x^* \leftarrow \max\{z - \lambda_0 w, 0\}$*.*
Exit;

(g) If $\lambda_1 \leq r < \infty$*,* $z_n - \lambda_1 w_n < 0$*,* $\lambda_0 \leq r$*, and* $n' < n$ *with* $z_{n'} \neq z_n$*, set*
i. $z \leftarrow z_{\mathcal{G}_0}$*;*
ii. $w \leftarrow w_{\mathcal{G}_0}$*;*
Go to step 1.

Output: x^**.*

With the previous results, the following theorem is almost immediate.

Theorem 3.1 *Algorithm 1 converges to* x^* *in at most* n *outer loops.*

Proof By Proposition 2.4, $x^* = P_{H(w,\varepsilon)\cap\mathcal{T}}(z) = P_{H(w_{\mathcal{G}(z)},\varepsilon)\cap\mathcal{T}}(z_{\mathcal{G}(z)}) = P_{H(w_{\mathcal{G}(z)},\varepsilon)\cap\mathcal{T}}(z)$, so we can assume that $w = w_{\mathcal{G}(z)}$ from the start. Furthermore, throughout this process z and w are updated to maintain that $\mathcal{G}(z) \preccurlyeq G^*$, and so we can apply Proposition 3.1 at every iteration. In particular, Proposition 3.1 implies that during every iteration of Algorithm 1, z and w must pass exactly one test. If tests 2a, 2b, 2d, or 2f are passed, the algorithm terminates with the correct solution. If tests 2c, 2e, or 2g are passed, then we update z and w, and the set $\mathcal{G}(z)$ decreases in size by at least one. Because $1 \leq |\mathcal{G}(z)| \leq n$, this process must terminate in at most n outer loops. □

The naive implementation of Algorithm 1 has worst case complexity bounded above by $O(n^2\log(n))$ because we must continually sort the ratios in Step 1a and update the vectors z and w through averaging

in Algorithm 1. However, it is possible to keep careful track of $\lambda_0, \lambda_1, r, z$, and w and get an $O(n\log(n))$ implementation of Algorithm 1. In order to prove this, we need to use a data-structure that is similar to a relational database.

Theorem 3.2 (Complexity of Algorithm 1) *There is an $O(n\log(n))$ implementation of Algorithm 1.*

Proof The key idea is to introduce a data structure $T_{\mathcal{G}} = \{t_{G_1}, \ldots, t_{G_g}\}$ consisting of 5-tuples, one for each group in a given partition $\mathcal{G} = \{G_1, \ldots, G_g\}$:

$$\forall i \in \{1, \ldots, g\} \qquad t_{G_i} := (r_{G_i}, \min(G_i), \max(G_i), S(G_i, z), S(G_i, w)) \in \mathbf{R} \times \mathbf{N} \times \mathbf{N} \times \mathbf{R}^2$$

where for any vector $x \in \mathbf{R}^n$, we let $S(G, x) = \sum_{i \in G} x_i$, and the ratios r_G are defined by

$$\forall i \in \{1, \ldots, g-1\} \qquad r_{G_i} := \frac{\frac{S(G_i,z)}{|G_i|} - \frac{S(G_{i+1},z)}{|G_{i+1}|}}{\frac{S(G_i,w)}{|G_i|} - \frac{S(G_{i+1},w)}{|G_{i+1}|}}, \qquad \text{and} \qquad r_{G_g} = \infty.$$

Notice that $S(G, z) = S(G, z_{\mathcal{G}})$ and $S(G, w) = S(G, w_{\mathcal{G}})$. We assume that the data structure $T_{\mathcal{G}}$ maintains 2 ordered-set views of the underlying tuples t_G, one of which is ordered by r_G, and another that is ordered by $\min(G)$. We also assume that the data structure allows us to convert iterators between views in constant time. This ensures that we can find the position of t_G with $G \in \arg\min\{r_G \mid G \in \mathcal{G}\}$ in the view ordered by r_G in time $O(\log(|\mathcal{G}|)$ and convert this to an iterator (at the tuple t_G) in the view ordered by $\min(G)$ in constant time. We also assume that the "delete," "find," and "insert," operations have complexity $O(\log(|\mathcal{G}|))$. We note that this functionality can be implemented with the Boost Multi-Index Containers Library [11].

Now, the first step of Algorithm 1 is to build the data structure $T_{\mathcal{G}(z)}$, which requires $O(n\log(n))$ operations. The remaining steps of the algorithm simply modify $T_{\mathcal{G}(z)}$ by merging and deleting tuples. Suppose that Algorithm 1 terminates in K steps for some $K \in \{1, \ldots, n\}$. For $i = 1, \ldots, K$, let $\mathcal{G}_i$ be partition at the current iteration, and let $m_i = |\mathcal{G}_i|$. Notice that for $i < K$, we have $\mathcal{G}_i \preccurlyeq \mathcal{G}_{i+1}$, so we get $\mathcal{G}_{i+1}$ by merging groups in $\mathcal{G}_i$, and $m_i > m_{i+1}$. Finally, we also maintain two numbers throughout the algorithm: $I_{\mathcal{G}_i} = \langle z_{\mathcal{G}_i}, w_{\mathcal{G}_i}\rangle$ and $N_{\mathcal{G}_i} = \|w_{\mathcal{G}_i}\|^2$. Given $I_{\mathcal{G}_i}$ and $N_{\mathcal{G}_i}$, we can compute λ_1 and λ_0 in constant time.

Now fix $i \in \{1, \ldots, K-1\}$. Suppose that we get from iteration i to $i+1$ through one of the updates $\mathcal{G}_{i+1} = \mathcal{G}(z_{\mathcal{G}_i} - \lambda_1 w_{\mathcal{G}_i})$ or $\mathcal{G}_{i+1} = \mathcal{G}(z_{\mathcal{G}_i} - \lambda_0 w_{\mathcal{G}_i})$. We note that each of these updates to $T_{\mathcal{G}_i}$ can be performed in at most $O((m_i - m_{i+1})\log(m_i))$ steps because we call at most $O(m_i - m_{i+1})$ "find", "insert", "delete", and "merge" operations on the structure $T_{\mathcal{G}_i}$ to get $T_{\mathcal{G}_{i+1}}$, and at most $O(m_i - m_{i+1})$ modifications to the variables $I_{\mathcal{G}_i}$ and $N_{\mathcal{G}_i}$ to get $I_{\mathcal{G}_{i+1}}$ and $N_{\mathcal{G}_{i+1}}$. Likewise, it is easy to see that modifications of the form $\mathcal{G}_{i+1} \leftarrow \mathcal{G}_0(z_{\mathcal{G}_i})$ can be implemented to run in $O((m_i - m_{i+1})\log(m_i))$ time.

Therefore, the total complexity of Algorithm 1 is

$$\begin{aligned} O\left(n\log(n) + \sum_{i=1}^{K}(m_i - m_{i+1})\log(m_i)\right) &= O\left(n\log(n) + \sum_{i=1}^{K}(m_i - m_{i+1})\log(n)\right) \\ &= O(n\log(n)). \end{aligned}$$

□

4 Numerical Results

In this section we present some numerical experiments to demonstrate the utility of the OWL norm and test our C++ implementation and MATLAB MEX file wrapper.

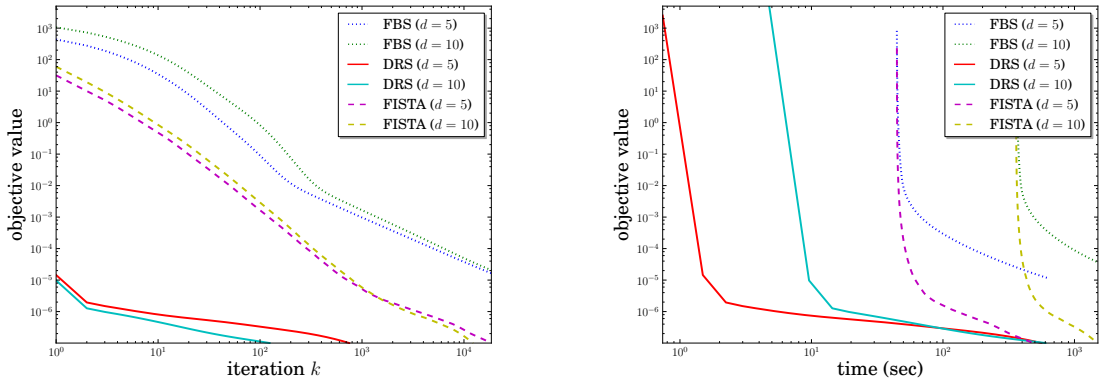

Fig. 4.1: We solve Problem (4.1) for $d = 5, 10$ with Douglas-Rachford splitting (DRS) [10], Forward-Backward splitting (FBS) [13], and an accelerated forward-backward splitting method (dubbed FISTA [3]). Note that the optimal objective value is 0 because $\varepsilon = \Omega_w(x_{\text{true}})$. In Figure 4.1b, there is a delay in the FBS and FISTA methods due to an initial investment in computing $\|A\|$, which is quite expensive. The test was run on a PC with 32GB memory and an Intel i5-3570 CPU with Ubuntu 12.04 and Matlab R2011b.

4.1 Synthetic Regression Test

We adopt and slightly modify the experimental set up of [17, Section V.A]. We choose an integer $d \geq 1$, and generate a vector

$$x_{\text{true}} := (\underbrace{0,\ldots 0}_{150d}, \underbrace{3,\ldots,3}_{50d}, \underbrace{0,\ldots 0}_{250d}, \underbrace{-4,\ldots,-4}_{50d}, \underbrace{0,\ldots 0}_{250d}, \underbrace{6,\ldots,6}_{50d}, \underbrace{0,\ldots 0}_{250d})^T \in \mathbf{R}^{1000d}.$$

We generate a random matrix $A = [A_1, \ldots, A_{1000d}] \in \mathbf{R}^{1000d \times 1000d}$ where the columns $A_i \in \mathbf{R}^{1000d}$ follow a multivariate Gaussian distribution with $\text{cov}(A_i, A_j) = .8^{|i-j|}$ after which the columns are standardized and centered. Then we generate a measurement vector $b = Ax_{\text{true}} + \nu$ where ν is Gaussian noise with variance .01. Next we generate w with OSCAR parameters $\mu_1 = 10^{-3}$ and μ_2^{-5} (See Equation (2.2)). Finally, we set $\varepsilon = \Omega_w(x_{\text{true}})$.

To test our implementation, we solve the regression problem

$$\underset{x \in \mathbf{R}^n}{\text{minimize}} \frac{1}{2}\|Ax - b\|^2 \qquad \text{subject to: } \Omega_w(x) \leq \varepsilon. \tag{4.1}$$

with three different proximal splitting algorithms. We plot the results in Figure 4.1.

4.2 Standalone Tests

In Table 4.1 we display the timings for our MATLAB MEX implementation of Algorithm 1. Note that solutions to (4.1) can be quite sparse (although usually not as sparse as solutions to (1.2)). Thus, the iterates generated by algorithms that solve (4.1), such as those applied in Figure 4.1, are sparse as well. Thus, we test our implementation on high-dimensional vectors of varying sparsity levels.

Density	length n			
	10^3	10^4	10^5	10^6
100%	3.6e-04	5.1e-03	6.8e-02	1.6
50%	2.1e-04	3.1e-03	3.8e-02	8.3e-01
25%	1.1e-04	1.6e-03	2.0e-02	3.7e-01
10%	5.6e-05	8.5e-04	1.0e-02	1.4e-01

Table 4.1: Average timings in seconds (over 100 runs) for random Gaussian vectors with different density levels (measured in percentage of nonzero entries). The test was run on a PC with 32GB memory and an Intel i5-3570 CPU with Ubuntu 12.04 and Matlab R2011b.

5 Conclusion

In this paper, we introduced an $O(n\log(n))$ algorithm to project onto the OWL norm ball. Previously, there was no algorithm to compute this projection in a finite number of steps. We also evaluated our algorithm with a synthetic regression test. A C++ implementation of our algorithm with a MEX wrapper, is available at the authors' website.

Acknowledgement

We thank Professor Wotao Yin for reading this draft and suggesting several improvements, Professor Robert Nowak for introducing us to the OWL norm, Zhimin Peng for discussing code implementation issues with us, and Professor Mário Figueiredo for trying out our code.

References

1. Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Structured Sparsity through Convex Optimization. Statist. Sci. **27**(4), 450–468 (2012). DOI 10.1214/12-STS394
2. Bauschke, H.H., Combettes, P.L.: Convex Analysis and Monotone Operator Theory in Hilbert Spaces, 1st edn. Springer Publishing Company, Incorporated (2011)
3. Beck, A., Teboulle, M.: A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. SIAM Journal on Imaging Sciences **2**(1), 183–202 (2009)
4. Bogdan, M., Berg, E.v.d., Su, W., Candes, E.: Statistical estimation and testing via the sorted L1 norm. arXiv preprint arXiv:1310.1969 (2013)
5. Bondell, H.D., Reich, B.J.: Simultaneous Regression Shrinkage, Variable Selection, and Supervised Clustering of Predictors with OSCAR. Biometrics **64**(1), 115–123 (2008)
6. Candes, E., Tao, T.: Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies? Information Theory, IEEE Transactions on **52**(12), 5406–5425 (2006). DOI 10.1109/TIT.2006.885507
7. Combettes, P.L., Pesquet, J.C.: Proximal splitting methods in signal processing. In: Fixed-point algorithms for inverse problems in science and engineering, pp. 185–212. Springer (2011)
8. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quarterly **3**(1-2), 95–110 (1956)
9. Held, M., Wolfe, P., Crowder, H.: Validation of subgradient optimization. Mathematical Programming **6**(1), 62–88 (1974)
10. Lions, P.L., Mercier, B.: Splitting Algorithms for the Sum of Two Nonlinear Operators. SIAM Journal on Numerical Analysis **16**(6), 964–979 (1979)

11. Muñoz, J.M.L.: The Boost Multi-Index Containers Library. C/C++ Users Journal **22**(9), 6 (2004)
12. Natarajan, B.K.: Sparse Approximate Solutions to Linear Systems. SIAM Journal on Computing **24**(2), 227–234 (1995)
13. Passty, G.B.: Ergodic Convergence to a Zero of the Sum of Monotone Operators in Hilbert Space. Journal of Mathematical Analysis and Applications **72**(2), 383–390 (1979)
14. Simon, N., Friedman, J., Hastie, T., Tibshirani, R.: A Sparse-Group Lasso. Journal of Computational and Graphical Statistics **22**(2), 231–245 (2013)
15. Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., Knight, K.: Sparsity and smoothness via the fused lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(1), 91–108 (2005)
16. Zeng, X., Figueiredo, M.A.: Decreasing Weighted Sorted ℓ_1 Regularization. IEEE Signal Processing Letters **21**, 1240–1244 (2014)
17. Zeng, X., Figueiredo, M.A.: The Ordered Weighted ℓ_1 Norm: Atomic Formulation, Dual Norm, and Projections. arXiv preprint arXiv:1409.4271v4 (2014)
18. Zhong, L., Kwok, J.: Efficient Sparse Modeling With Automatic Feature Grouping. Neural Networks and Learning Systems, IEEE Transactions on **23**(9), 1436–1447 (2012)
19. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(2), 301–320 (2005)

Appendix

A Proof of Proposition 2.4

First we prove a simple fact that we will use throughout the following proofs. Intuitively, it states that $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ if, and only if, $\mathcal{G}_2$ does not split groups in $\mathcal{G}_1$.

Lemma A.1 (Equivalent conditions for nested partitions) *Let $\mathcal{G}_1, \mathcal{G}_2 \in \mathcal{P}_n$. Then $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$ if, and only if, for every $i \in \{1,\dots,n\}$ such that there exists a group $G_1 \in \mathcal{G}_1$ with $i, i+1 \in G_1$, there exists a group G_2 such that $i, i+1 \in G_2$.*

Proof (Proof of lemma) $\Longrightarrow$: This direction is clear by definition of $\preccurlyeq$.

$\Longleftarrow$: Suppose that $G_1 = \{i_1,\dots,i_k\} \in \mathcal{G}_1$ for some $k \geq 1$. If $|G_1| = 1$, the partition property implies there exists $G_2 \in \mathcal{G}_2$ containing G_1. Suppose $|G_1| > 1$. For each i_j with $j = 1,\dots,k-1$, there exists $G_2^j \in \mathcal{G}_2$ with $i_j, i_{j+1} \in G_2^j$. Notice that each of the adjacent G_2^j sets intersect: $i_j \in G_2^{j-1} \cap G_2^j$ for $j = 2,\dots,k-1$. Thus, by the partition property, all G_2^j are the same and hence, $G_1 \subseteq G_2^j$ for any such j. Thus, $\mathcal{G}_1 \preccurlyeq \mathcal{G}_2$. □

Part 1: Let $i \in \{1,\dots,n\}$. Suppose that $z_i = z_{i+1}$. Then $z_i - z_{i+1} = 0 \leq \lambda(w_i - w_{i+1})$, i.e., $z_i - \lambda w_i \leq z_{i+1} - \lambda w_{i+1}$. Therefore, by Lemma A.1, we have $\mathcal{G}(z) \preccurlyeq \mathcal{G}(z - \lambda w)$.

Next, suppose that $z_i - \lambda w_i \leq z_{i+1} - \lambda w_{i+1}$ where z_i is not necessarily equal to z_{i+1}. Then i and $i+1$ are in the same group in $\mathcal{G}(z - \lambda w)$. Thus, by Equation (2.8), we have $(z_{\mathcal{G}(z-\lambda w)})_i = (z_{\mathcal{G}(z-\lambda w)})_{i+1}$. Therefore, by Lemma A.1, we have $\mathcal{G}(z - \lambda w) \preccurlyeq \mathcal{G}(z_{\mathcal{G}(z-\lambda w)})$. Conversely, suppose that $(z_{\mathcal{G}(z-\lambda w)})_i = (z_{\mathcal{G}(z-\lambda w)})_{i+1}$, but $z_i - \lambda w_i > z_{i+1} - \lambda w_{i+1}$. Then i and $i+1$ are not in the same group in $\mathcal{G}(z - \lambda w)$ and, in particular, $z_i - z_{i+1} > \lambda(w_i - w_{i+1}) \geq 0$. Thus, because $z \in \mathcal{T}$, we have $(z_{\mathcal{G}(z-\lambda w)})_i \geq z_i > z_{i+1} \geq (z_{\mathcal{G}(z-\lambda w)})_{i+1}$, which is a contradiction. Therefore, by Lemma A.1, we have $\mathcal{G}(z - \lambda w) \succcurlyeq \mathcal{G}(z_{\mathcal{G}(z-\lambda w)})$, and so $\mathcal{G}(z - \lambda w) = \mathcal{G}(z_{\mathcal{G}(z-\lambda w)})$.

Finally, suppose that there exists $i \in \{1,\dots,n-1\}$ such that $z_i - \lambda w_i \leq z_{i+1} - \lambda w_{i+1}$. Then by Proposition 2.2,

$$x_i^* - x_{i+1}^* := (z_i - \lambda^* w_i) - (z_{i+1} - \lambda^* w_{i+1}) + 2v_i^* - (v_{i-1}^* + v_{i+1}^*).$$

If $x_i^* \neq x_{i+1}^*$, then $2v_i^* = 0$ so the expression on the left is nonpositive, which is a contradiction. Thus, $x_i^* = x_{i+1}^*$. Therefore, by Lemma A.1, we have $\mathcal{G}(z_{\mathcal{G}(z-\lambda w)}) = \mathcal{G}(z - \lambda w) \preccurlyeq \mathcal{G}^*$.

Part 2: Note that

$$\langle w_{\mathcal{G}}, x^* \rangle = \sum_{G \in \mathcal{G}} \sum_{i \in G} x_i^* \left(\frac{1}{|G|} \sum_{j \in G} w_j \right) = \langle w, x^* \rangle = \varepsilon$$

because x^* is constant along each group G. Thus, $x^* \in H(w_{\mathcal{G}}, \varepsilon) \cap \mathcal{T}$. Let $x^0 = P_{H(w_{\mathcal{G}},\varepsilon)\cap\mathcal{T}}(z_{\mathcal{G}})$. We will show that $x^* = x^0$. Indeed, $\mathcal{G} = \mathcal{G}(z_{\mathcal{G}}) \preccurlyeq \mathcal{G}(x^0)$ and

$$\langle w, x^0\rangle = \sum_{G\in\mathcal{G}}\sum_{i\in G} x_i^0\left(\frac{1}{|G|}\sum_{j\in G} w_j\right) = \langle w_{\mathcal{G}}, x^0\rangle = \varepsilon$$

because x^0 is constant along each group. Therefore, $x^0 \in H(w,\varepsilon)\cap\mathcal{T}$. In addition, for all $G\in\mathcal{G}$, we have $x_i^0 = x_j^0$ for all $i,j\in G$; let x^G denote x_i^0 for any $i\in G$. Therefore,

$$\begin{aligned}\|z-x^*\|^2 \le \|z-x^0\|^2 = \sum_{G\in\mathcal{G}}\sum_{i\in G}\left(z_i - x^G\right)^2 &= \sum_{G\in\mathcal{G}}\left(\frac{1}{2|G|}\sum_{i,j\in G}(z_i-z_j)^2 + \sum_{i\in G}\left(z_{\mathcal{G}}-x^G\right)^2\right)\\ &\le \sum_{G\in\mathcal{G}}\left(\frac{1}{2|G|}\sum_{i,j\in G}(z_i-z_j)^2 + \sum_{i\in G}(z_{\mathcal{G}}-x_i^*)^2\right)\\ &= \|z-x^*\|^2\end{aligned}$$

Thus, $\|z-x^*\| = \|z-x^0\|$, so by the uniqueness of the projection, we have $x^0 = x^*$.

B Proof of Proposition 3.1

First note that because $\langle w, x^*\rangle = \varepsilon$, Proposition 2.2 implies that

$$\lambda^* = \frac{\sum_{i=1}^n z_i w_i - \varepsilon + \sum_{i=1}^{n-1} v_i^*(w_i - w_{i+1}) + v_n^* w_n^*}{\|w\|^2} \ge \lambda_1. \tag{B.1}$$

because $\sum_{i=1}^{n-1} v_i^*(w_i - w_{i+1}) + v_n^* w_n^* \ge 0$.

Part 1: Suppose $r=\infty$. Then w is a constant vector. Thus, the result follows from Proposition 2.3.

Part 2: Suppose that $\lambda_1 > r$. Then $\lambda^* > r$ by Equation (B.1).

Part 3: Suppose that $\infty > r \ge \lambda_1$ and $z_n - \lambda_1 w_n \ge 0$. Then $z - \lambda_1 w \in \mathcal{T}$ and $x^0 = z - \lambda_1 w$ satisfies the conditions of Proposition 2.2 with $v^* = 0$ and $\lambda^* = \lambda_1$. Thus, $x^* = z - \lambda_1 w$.

Part 4: Suppose that $\infty > r \ge \lambda_1$, $z_n - \lambda_1 w_n < 0$, and $\lambda_0 > r$. Then, $z_n - \lambda^* w_n \le z_n - \lambda_1 w_n < 0$. From $x_n^* = z_i - \lambda^* w_n + v_n^* - v_{i-1}^* < v_n^*$, and $v_n^* x_n^* = 0$, we have $x_n^* = 0$. Next, because $\mathcal{G}(z) \preccurlyeq \mathcal{G}^*$, we have $\{i \mid z_i = z_n\} \subseteq \{i \mid x_i^* = x_n^*\} = \{i \mid x_i^* = 0\}$ and so $\{i \mid z_i > z_n\} \supseteq \{i \mid x_i^* > 0\}$. Let $k_0 = \max\{i \mid z_i > z_n\}$. Therefore, from $\sum_{\{i|z_i>z_n\}} x_i^* w_i = \sum_{\{i|x_i^*>0\}} x_i^* w_i = \varepsilon$ and Proposition 2.2, we have

$$\lambda^* = \frac{\sum_{\{i|z_i>z_n\}} z_i w_i - \varepsilon + \sum_{\{i|z_i>z_n\}} v_i^*(w_i - w_{i+1}) + v_{k_0} w_{k_0+1}}{\sum_{\{i|z_i>z_n\}} w_i^2} \ge \lambda_0 > r \tag{B.2}$$

where we use the bound $\sum_{\{i|z_i>z_n\}} v_i^*(w_i - w_{i+1}) + v_{k_0} w_{k_0+1} \ge 0$. Notice that $x_n^* = 0$ and the first inequality in Equation (B.2) holds whether or not $\lambda_0 > r$: we just need $\infty > r \ge \lambda_1$ and $z_n - \lambda_1 w_n < 0$. We will use this fact in Part 6 below.

Part 5: Suppose that $\infty > r \ge \lambda_1$, $z_n - \lambda_1 w_n < 0$, $r \ge \lambda_0$, $n' \le n$ and $z_{n'} = z_n$. Then $\max\{z - \lambda_0 w, 0\} \in \mathcal{T}$. In addition, we have $\langle w, \max\{z-\lambda_0 w, 0\}\rangle = \varepsilon$ by the choice of λ_0. We will now define a vector $v \in \mathbf{R}_+^n$ recursively: If $z_i > z_n$, set $v_i = 0$; otherwise set $v_i = v_{i-1} - (z_i - \lambda_0 w_i)$. We can satisfy the optimality conditions of Proposition 2.2 with $\lambda^* = \lambda_0$ and $v^* = v$. Thus, $x^* = \max\{z - \lambda_0 w, 0\}$.

Part 6: Suppose that $\infty > r \ge \lambda_1$, $z_n - \lambda_1 w_n < 0$, $r \ge \lambda_0$, $n' < n$ and $z_{n'} \ne z_n$. From the proof of Part 4 we have $z_k - \lambda^* w_k \le z_k - \lambda_0 w_k < 0$ for all $k = n', n'+1, \ldots, n$ (from $\lambda^* \ge \lambda_0$) and $x_n^* = 0$. Suppose that $x_{n'}^* \ne x_n^* = 0$. Let $n'' = \min\{k \mid x_k^* = 0\}$. Then $n''-1 \ge n' \ge 1$. Thus because $x_{n''-1}^* \ne x_{n''}^* = 0$, we have $v_{n''-1}^* = 0$ and $x_{n''-1}^* = z_{n''-1} - \lambda^* w_{n''-1} - v_{n''-2}^* < 0$ (where we let $v_{n''-2}^* = 0$ if $n'' = 2$). This is a contradiction because $x^* \in \mathcal{T}$. Thus, $x_{n'}^* = x_{n'+1}^* = \ldots = x_n^* = 0$. If $n' = 1$, then we see that $\mathcal{G}(z) \preccurlyeq \mathcal{G}_0 \preccurlyeq \mathcal{G}^*$. Furthermore, if $n' > 1$, then we claim that $n'-1$ and n' are not in the same group in $\mathcal{G}(z)$, i.e., that $z_{n'-1} \ne z_{n'}$. Indeed, if $z_{n'-1} = z_{n'}$, then $w_{n'-1} = w_{n'}$ and hence, $z_{n'-1} - \lambda_0 w_{n'-1} = z_{n'} - \lambda_0 w_{n'} < 0$, which is a contradiction.

Thus, this argument has shown that $\mathcal{G}(z) = \{G \in \mathcal{G}(z) \mid \max(G) < n'\} \cup \{G \in \mathcal{G}(z) \mid \min(G) \geq n'\}$ and there exists $G_2 \in \mathcal{G}^*$ with $\{n', \dots, n\} \subseteq G_2$. Note that the first of these identities implies that $\mathcal{G}_0 \in \mathcal{P}_n$. Let us now prove the claimed nestings: $\mathcal{G}(z) \preccurlyeq \mathcal{G}_0 = \mathcal{G}(z_{\mathcal{G}_0}) \preccurlyeq \mathcal{G}^*$.

1. $(\mathcal{G}(z) \preccurlyeq \mathcal{G}_0)$: Suppose that $G \in \mathcal{G}(z)$. If $\max(G) < n'$, then $G \in \mathcal{G}_0$. If $\min(G) \geq n'$, then $G \subseteq \{n', \dots, n\} \in \mathcal{G}_0$. Thus, $\mathcal{G}(z) \preccurlyeq \mathcal{G}_0$.
2. $(\mathcal{G}_0 = \mathcal{G}(z_{\mathcal{G}_0}))$: The identity follows because

$$(z_{\mathcal{G}_0})_i = \begin{cases} z_i & \text{if } i < n'; \\ \frac{1}{n-n'+1} \sum_{i=n'}^{n} z_i & \text{if } i \geq n'. \end{cases}$$

3. $(\mathcal{G}_0 \preccurlyeq \mathcal{G}^*)$: Suppose that $G \in \mathcal{G}_0$. If $\max(G) < n'$, it follows that $G \in \mathcal{G}(z)$ and hence by Part 1 of Proposition 2.4, there is a $G_2 \in \mathcal{G}^*$ with $G \subseteq G_2$. If $\min(G) \geq n'$, then $G = \{n', \dots, n\}$ and there exists $G_2 \in \mathcal{G}^*$ with $G \subseteq G_2$. Therefore, $G_0 \preccurlyeq \mathcal{G}^*$.

Finally, note that $|\mathcal{G}(z_{\mathcal{G}_0})| = |\mathcal{G}_0| \leq |\mathcal{G}(z)| - 1$ because $z_{n'} \neq z_n$ implies that $\{G \in \mathcal{G}(z) \mid \min(G) \geq n'\} \subseteq \mathcal{G}(z)$ contains at least two distinct groups that are both contained in $\{n', \dots, n\} \in \mathcal{G}_0$.

Part 7: Suppose that $\infty > r \geq \lambda_1$, $z_n - \lambda_1 w_n < 0$, $r > \lambda_0$, and $n' = n + 1$. Then $z_n - \lambda_0 w_n \geq 0$. Thus, $\lambda_1 > \lambda_0$ and

$$\begin{aligned} \lambda_1 \left(\sum_{\{i|z_i > z_n\}} w_i^2 + \sum_{\{i|z_i = z_n\}} w_i^2 \right) &= \left(\sum_{\{i|z_i > z_n\}} z_i w_i + \sum_{\{i|z_i = z_n\}} z_i w_i \right) - \varepsilon \\ &< \left(\sum_{\{i|z_i > z_n\}} z_i w_i - \varepsilon \right) + \sum_{\{i|z_i = z_n\}} \lambda_1 w_i^2 \\ &= \lambda_0 \left(\sum_{\{i|z_i > z_n\}} w_i^2 \right) + \lambda_1 \sum_{\{i|z_i = z_n\}} w_i^2. \end{aligned}$$

where the strict inequality follows from $z_n < \lambda_1 w_n$. Thus, $\lambda_1 < \lambda_0$, which is a contradiction.

The final conclusions of the proposition are simple consequence of Lemma 2.2, Proposition 2.4, and the 6 alternatives.

C An Example

In this section, we project the point $z_0 = (3, 2, 1, -1, 2)$ onto the OWL ball of radius $\varepsilon = 1$ with weights $w_0 = (5, 4, 3, 1, 1)$.

- **Preprocessing.**
  - Set $s := \operatorname{sign}(z_0) = (1, 1, 1, -1, 1)^T$;
  - Set $z := Q(|z_0|)|z_0| = (3, 2, 2, 1, 1)^T$;
  - Set $\mathcal{G}(z) \leftarrow \{\{1\}, \{2, 3\}, \{4, 5\}\}$;
  - Set $w := (w_0)_{\mathcal{G}(z)} = (5, 7/2, 7/2, 1, 1)^T$;
- **Iteration 1.**
  - Set
$$r \leftarrow \min\left\{ \frac{3-2}{5-7/2}, \infty, \frac{2-1}{7/2-1}, \infty \right\} = \frac{2}{5};$$
  - Set
$$\lambda_0 \leftarrow \frac{28}{49.5} \qquad \text{and} \qquad \lambda_1 \leftarrow \frac{31}{51.5};$$
  - Set $n' \leftarrow 6$;
  - Set $\mathcal{G}_0(z) \leftarrow \mathcal{G}(z)$;
  - **Test 2c passed:** $\lambda_1 = 31/51.5 > 2/5 = r$;
    - Set $\mathcal{G}(z - \lambda_1 w) \leftarrow \{\{1\}, \{2, 3, 4, 5\}\}$;
    - Set $z \leftarrow z_{\mathcal{G}(z-\lambda_1 w)} = (3, 3/2, 3/2, 3/2, 3/2)^T$;
    - Set $w \leftarrow w_{\mathcal{G}(z-\lambda_1 w)} = (5, 9/4, 9/4, 9/4, 9/4)^T$;
- **Iteration 2.**

  - Set
$$r \leftarrow \min\left\{\frac{3-3/2}{5-9/4}, \infty, \infty, \infty\right\} = \frac{12}{22};$$
  - Set
$$\lambda_0 \leftarrow \frac{14}{25} \qquad \text{and} \qquad \lambda_1 \leftarrow \frac{27.5}{45.25};$$
  - Set $n' \leftarrow 6$;
  - Set $\mathcal{G}_0(z) \leftarrow \mathcal{G}(z) = \{\{1\}, \{2,3,4,5\}\}$;
  - **Test 2c passed:** $\lambda_1 = 27.5/45.25 > 12/22 = r$;
    - Set $\mathcal{G}(z - \lambda_1 w) \leftarrow \{\{1,2,3,4,5\}\}$;
    - Set $z \leftarrow z_{\mathcal{G}(z-\lambda_1 w)} = (9/5, 9/5, 9/5, 9/5, 9/5)^T$;
    - Set $w \leftarrow w_{\mathcal{G}(z-\lambda_1 w)} = (14/5, 14/5, 14/5, 14/5, 14/5)^T$;
- **Iteration 3.**
  - Set $r = \infty$;
  - **Test 2b passed:** (We use Proposition 2.3 to finish.)
    - Set $\lambda = 121/70$;
    - Set $x^* = \max\{z - \lambda, 0\} = (1/14, 1/14, 1/14, 1/14, 1/14)^T$;
- **Undo preprocessing.**
  - Set $x_0^* = s \odot Q(|z_0|)^T x^* = (1/14, 1/14, 1/14, -1/14, 1/14)^T$;
- **Terminate.**
  - We have $P_{\mathcal{B}(w_0,\varepsilon)}(z_0) = x_0^*$.

  Notice that x_0^* satisfies $\Omega_{w_0}(x_0^*) = 1$ because $\sum_{i=1}^5 (w_0)_i = 14$.