Efficient Bregman Projections onto the Permutahedron and Related Polytopes

Cong Han Lim
Department of Computer Sciences,
University of Wisconsin-Madison

Stephen J. Wright
Department of Computer Sciences,
University of Wisconsin-Madison

Abstract

The problem of projecting onto the permutahedron $\mathcal{PH}(c)$—the convex hull of all permutations of a fixed vector c—under a uniformly separable Bregman divergence is shown to be reducible to the Isotonic Optimization problem. This allows us to employ known fast algorithms to improve on several recent results on Bregman projections onto permutahedra. In addition, we present a new algorithm `MergeAndPool` that have better complexity when the number of distinct entries d in the vector c is small, the simplex being one such example, with $c = (1, 0, 0, \ldots, 0)^T$ and $d = 2$. `MergeAndPool` runs in $O(n \log d)$ for certain popular Bregman divergence measures and requires $O((n \log d) \log \frac{U}{\epsilon})$ to find ϵ-close solutions for general uniformly separable Bregman divergences, where U is a bound on the width of the interval containing the dual solution components. These estimates matches or improves best known bounds for all Bregman projection problems onto various permutahedra, including recent results for projection onto the simplex. The same complexity bounds apply to signed permutahedra, a class that includes the ℓ_1-ball as a special case. In summary, this work describes a fast unified approach to this well-known class of problems.

1 Introduction

The permutahedron induced by vector $c \in \mathbb{R}^n$ is the convex hull of the set of all permutations of c. When $c = (n, n-1, \ldots, 1)$, we obtain the convex hull of all ranking vectors, and when $c = (1, 0, \ldots, 0)$, we obtain the probability simplex. A related set is the signed permutahedron, obtained by taking the convex hull of all combinations of permutations and sign-flips of a vector c (for $c = (1, 0, \ldots, 0)$, we get the ℓ_1-ball). In recent years, there has been significant work on algorithms for projecting onto such sets under the Euclidean norm and other Bregman divergences, which includes works on regularization for sparse optimization (Duchi et al., 2008; Zeng and Figueiredo, 2014; Negrinho and Martins, 2014) and online learning (Suehiro et al., 2012; Ailon et al., 2014).

Given a vector $z \in \mathbb{R}^n$, uniformly separable Bregman divergence $\Delta_\Phi = \sum \Delta_\phi$, and permutahedron $\mathcal{PH}(c)$, the problem we address in this paper is:

$$\operatorname*{arg\,min}_{x \in \mathcal{PH}(c)} \sum_{i=1}^{n} \Delta_\phi(x_i, z_i).$$

This problem can be reduced to the *Isotonic Optimization* problem by sorting the entries of the vector z, and taking the dual of a simplified problem, to obtain

$$\min_{y \in \mathbb{R}^n} \sum_{i=1}^{n} f_i(y_i) \quad \text{such that } y_1 \le y_2 \le \ldots \le y_n,$$

Best et al. (2000) demonstrate that the canonical `PAV` algorithm for Isotonic Regression (Barlow, 1972) solves this problem and requires up to n solves of a certain pooling subproblem (referred to below as `PoolV`), which involves unconstrained minimization over a partial sum of the n terms in the full summation above. The cost of obtaining this minimizer depends on the choice of Bregman divergence. For certain Bregman divergences, like Euclidean distance, unnormalized relative entropy (Suehiro et al., 2012), or KL-divergence (Krichene et al., 2015), `PoolV` can be computed in an "incremental" fashion that allows re-use of the work from previous computations, leading to an overall cost of just $O(n)$ to compute all minimizers necessary for

Appearing in Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51. Copyright 2016 by the authors.

Type	Incremental `PoolV`	General Uniformly Separable Bregman Divergences
Previously Reported: Permutahedron	ℓ_2: $O(n\log n)$ (Negrinho and Martins, 2014) Unnormalized Relative Entropy: $O(n^2)$ (Suehiro et al., 2012)	$O(n^3 \log \frac{U}{\epsilon})$ (Ailon et al., 2014)
Previously Reported: Simplex ($d=2$) (($d=3$) for capped)	ℓ_2: $O(n)$ (Brucker, 1984) KL-divergence: $O(n)$ (Krichene et al., 2015) ℓ_2, Capped: $O(n\log n)$(Wang and Lu, 2015)	$O(n \log \frac{U}{\epsilon})$ (Krichene et al., 2015)
Known Isotonic Optimization Algorithms	`PAV`: $O(n\log n)$ (Best et al., 2000)	`PAV`: $O(n^2 \log \frac{nU}{\epsilon})$ `ScalingPAV`: $O(n \log \frac{nU}{\epsilon})$ (Ahuja and Orlin, 2001)
This Work [1]	`MergeAndPool`: $O(n\log d)$	`MergeAndPool`: $O((n\log d)\log \frac{U}{\epsilon})$

Table 1: Reported running time results for Bregman projections onto permutahedra and other related polytopes vs. results from using known `PAV`-type algorithms vs. our work. The running times stated for the algorithms factor in the time needed for the initial sort. For general Bregman divergences, U denotes the gap between an upper and lower bound for the solution to the dual problem, and solution obtained is ϵ-close in ℓ_∞-norm.

the algorithm. For other uniformly-separable Bregman divergences, an ϵ-close solution (in the ℓ_∞-norm) to Isotonic Optimization can be solved in $O(n\log\frac{U}{\epsilon})$ time using `ScalingPAV` (Ahuja and Orlin, 2001), where U denotes the gap between an upper and lower bound on all values of the dual solution. After factoring in the cost of the initial sorting step, the total cost of this approach is either $O(n\log n)$ for Bregman divergences with incremental `PoolV` steps and $O(n\log\frac{nU}{\epsilon})$ for other uniformly separable Bregman divergences.

For the simplex, it has been noted that we can avoid the initial sorting step and compute the projection in just $O(n)$ time (Brucker, 1984; Krichene et al., 2015) for incremental `PoolV` and $O(n\log\frac{U}{\epsilon})$ for other uniformly separable Bregman divergences. We identify the key feature that allows the projection onto the simplex to be significantly easier than projection onto the permutahedron: *the number d of distinct entries in the vector c*. The permutahedron has $d=n$, whereas the simplex has only $d=2$.

A key contribution of this paper is the introduction of an algorithm `MergeAndPool` that is able to adapt smoothly to the number of distinct entries in the base vector c. `MergeAndPool` achieves $O(n\log d)$ for incremental `PoolV` and $O((n\log d)\log\frac{U}{\epsilon})$ for other uniformly separable Bregman divergences respectively. See Table 1 for details.

We begin by covering the preliminaries and notation in Section 2. We then describe the reduction of the projection problem to the Isotonic Optimization problem and show the relation between the projection problem for permutahedra and signed permutahedra in Section 3. Section 4 introduces the `PoolV` operation and describes when Bregman divergences have the incremental `PoolV` property. Finally, we cover the algorithms in Sections 5 (incremental `PoolV`) and 6 (general uniformly separable Bregman divergences). In the supplementary material, we include omitted proofs and experiments illustrating scaling effects as predicted by the theory.

[1]In the full version of this paper, we introduce a variant of the `ScalingPAV` algorithm that achieves $O(n\log\frac{dU}{\epsilon})$ for general uniformly separable Bregman divergences.

Prior Work. Several works have tackled related problems, but we study it with an explicit focus on the projection onto permutahedron step. Bregman projections onto permutahedra to our knowledge was first explicitly tackled in the online learning literature by Yasutake et al. (2011), who demonstrate a $O(n^2)$ algorithm for Euclidean projection. The same framework was used to show a $O(n^2)$ algorithm for unnormalized relative entropy (Suehiro et al., 2012) and a $O(n^2\tau(n))$ algorithm for the binary relative entropy (Ailon et al., 2014), where $\tau(n)$ is the cost of computing the minimizer of a pooling subproblem associated with binary relative entropy. (Using a binary search process, $\tau(n)$ is $O(n\log\frac{U}{\epsilon})$ for an ϵ-close solutions).

Efficient algorithms for the general Isotonic Optimization problem were introduced by Best et al. (2000) and Ahuja and Orlin (2001). Special cases such as Isotonic Regression (where f_i are Euclidean distance functions (Barlow, 1972; Brucker, 1984)) have been studied for some time. The reduction of Euclidean projection onto the permutahedron to the Isotonic Regression problem has been explicitly pointed out by Zeng and Figueiredo (2014); Negrinho and Martins (2014).

Projection onto the simplex and related polytopes has been studied independently of the work on the permutahedron. A linear-time algorithm for Euclidean projection onto the simplex was first provided by Brucker (1984), and efficient algorithms for general uniformly separable Bregman divergences were recently studied by Krichene et al. (2015). The latter have complexities $O(n)$ for KL-divergences and $O(n \log \frac{U}{\epsilon})$ for general uniformly separably Bregman divergences. Wang and Lu (2015) demonstrate a $O(n \log n)$ algorithm for the capped simplex (which add an ℓ_∞ bound to the standard simplex).

The relationship between projecting onto the permutahedron and signed permutahedron has been known in the submodularity research community (e.g. Bach (2013)), and more recently has been noted for Euclidean projections by Zeng and Figueiredo (2014); Negrinho and Martins (2014), and for the Euclidean projection between the simplex and ℓ_1-ball by Duchi et al. (2008).

Consequences for online learning over ranking polytopes. For the online learning problems studied by Suehiro et al. (2012); Ailon et al. (2014), the authors propose online learning algorithms based on the Follow-The-Regularized-Leader (FTRL) approach (see Hazan (2012) for example). In these algorithms the projection step has been the bottleneck, taking $O(n^2)$ or more time while the rest of the algorithm takes $O(n \log n)$ time. Using the $O(n \log n)$ `PAV` algorithm for projection under unnormalized relative entropy proves that the FTRL approach by Suehiro et al. (2012) achieves both running time and optimal regret bound of the later work by Ailon (2014). By applying `ScalingPAV`, we also speed up the projection step for binary relative entropy (Ailon et al., 2014) and hence speed up the overall algorithm significantly. The new algorithms that adapt smoothly to the number of distinct entries in the vertices of the permutahedra also improve the running time for online learning over truncated permutahedra (Suehiro et al., 2012).

2 Preliminaries

Permutahedra and Norm-balls. Let $c = (c_1, c_2, \ldots, c_n)$ denote a vector in $\mathbb{R}^n$ where $c_1 \geq c_2 \geq \ldots \geq c_n$. We will use n throughout the paper to refer to the length of the permutation vectors, and use $[k]$ to denote the interval of indices $\{1, 2, \ldots, k\}$. Let $\mathcal{P}(c)$ be the set of all permutations of vector c.

Definition 2.1. *The* permutahedron $\mathcal{PH}(c)$ induced by *c, the convex hull of* $\mathcal{P}(c)$*, is*

$$\left\{ x \in \mathbb{R}^n \;\middle|\; \sum_{i \in S} x_i \leq \sum_{i=1}^{|S|} c_i \;\; \forall S \subset [n], \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} c_i \right\},$$

For $c \geq 0$*, the* signed permutahedron $\mathcal{SPH}(c)$ *is* $\{x \in \mathbb{R}^n \mid (|x_1|, |x_2|, \ldots, |x_n|) \in \mathcal{PH}(c)\}$*, which is equivalent to*

$$\left\{ x \in \mathbb{R}^n \;\middle|\; \sum_{i \in S} |x_i| \leq \sum_{i=1}^{|S|} c_i \;\; \forall S \subset [n] \right\}.$$

The term "permutahedron" is generally used in the literature for the special case of $\mathcal{PH}((n, n-1, \ldots, 1))$ in the definition above. The probability simplex is $\mathcal{PH}((1, 0, \ldots, 0))$. We can also express the capped simplex—the probability simplex with the additional restriction $\|x\|_\infty \leq \tau$, for some $\tau \in (0,1)$—as $\mathcal{PH}((\tau, \ldots, \tau, 1 - k\tau, 0, \ldots, 0))$, where k is chosen such that $1 - k\tau \in [0, \tau)$, and τ is repeated k times at the start of c. The signed permutahedron naturally defines an atomic norm, and can be shown to be the dual norm of the recently introduced SLOPE or OWL norms (Bogdan et al., 2013; Zeng and Figueiredo, 2014). The set $\mathcal{SPH}((1, 0, \ldots, 0)^T)$ is the ℓ_1-ball.

For the rest of the paper we will use x to denote a feasible point for the permutahedron and z to denote an arbitrary point in $\mathbb{R}^n$ which we wish to project onto this set.

Bregman Divergence. Let $\Phi : S \to \mathbb{R}$ be a continuously-differentiable strictly convex function on the closed set $S \subseteq \mathbb{R}^n$. The *Bregman divergence* Δ_Φ associated with Φ is a function on pairs of points $v, w \in S$ where

$$\Delta_\Phi(v, w) = \Phi(v) - \Phi(w) - \langle \nabla \Phi(w), v - w \rangle.$$

The function Φ is *separable* if we can express $\Phi(v)$ as the sum of univariate functions $\Phi(v) = \sum_{i=1}^n \phi_i(v_i)$, and Φ is *uniformly separable* if all $\phi_i = \phi$ are the same function. When Φ is uniformly separable, we can write $\Delta_\Phi(v, w)$ as $\sum_{i=1}^n \Delta_\phi(v_i, w_i)$. For the remainder of the paper, we assume that all Bregman divergences that we work with are uniformly separable.

Other Notation. We will use x' or y' to denote the optimal solution to their corresponding optimization problems. Given a set of indices $S \subset [n]$, we use v_S to denote the subvector of v corresponding to S. Many of the intervals described in this paper refer to intervals of indices.

3 Reduction to Isotonic Optimization

In this section, we will extend results for relation between Euclidean projection onto the permutahedron and the Isotonic Regression problem (as noted by Zeng and Figueiredo (2014); Negrinho and Martins (2014)) to any uniformly separable Bregman divergence.

Recall that given vector $z \in \mathbb{R}^n$ to project, a uniformly separable Bregman divergence Δ_Φ, and permutahedron $\mathcal{PH}(c)$, the projection problem is

$$\arg\min_{x \in \mathcal{PH}(c)} \Delta_\Phi(x, z) = \sum_{i=1}^{n} \Delta_\phi(x_i, z_i). \tag{1}$$

This problem has exponentially many linear constraints, but it can be solved in polynomial time. We could use standard convex optimization techniques on a compact extended formulation of the permutahedron (Goemans, 2015), use results for polymatroids or base polyhedron of submodular functions (e.g. Groenevelt (1991); Fujishige (1984)), or deploy specialized algorithms for this problem. Suehiro et al. (2012) observed that for uniformly separable Bregman divergences, we can reduce the problem to one with significantly fewer constraints. The following lemma shows that the ordering of the solution is the same as the ordering of elements in z.

Lemma 3.1. *(Suehiro et al., 2012) Let x' be the projection of z onto the permutahedron under a uniformly separable Bregman divergence Δ_Φ. Suppose $z_1 \geq z_2 \geq \ldots \geq z_n$. Then we have $x'_1 \geq x'_2 \geq \ldots \geq x'_n$.*

We can now sort the input z and restrict our attention to only the constraints that can be active under that ordering. Our problem (1) simplifies as follows:

$$\begin{aligned} &\arg\min_{x} \ \sum_{i=1}^{n} \Delta_\phi(x, z) \\ &\text{such that } \sum_{i=1}^{k} x_i \leq \sum_{i=1}^{k} c_i \text{ for } k \in [n-1], \\ &\qquad\qquad \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} c_i. \end{aligned} \tag{2}$$

This formulation only has linearly many constraints. By the strict convexity of ϕ, this problem has exactly one optimal solution.

Let $d_i(x_i) = \Delta_\phi(x_i, z_i)$ and let d_i^* denote the Legendre-Fenchel dual of d_i. The dual to problem (2) is

$$\begin{aligned} &\min_{y \in \mathbb{R}^n} \ \sum_{i=1}^{n} (d_i^*(y_i) - y_i c_i) \\ &\text{such that } y_1 \leq y_2 \leq \ldots \leq y_n. \end{aligned} \tag{3}$$

(We provide the derivation in the appendix.) From a dual solution y', we can recover x via the following relation:

$$\nabla\phi(x_i) = y'_i + \nabla\phi(z_i). \tag{4}$$

Problem (3) is an *Isotonic Optimization* problem, which has been studied in greater generality by Best et al. (2000) and Ahuja and Orlin (2001), who consider the form

$$\begin{aligned} &\min_{y \in \mathbb{R}^n} \ \sum_{i=1}^{n} f_i(y_i) \\ &\text{such that } y_1 \leq y_2 \leq \ldots \leq y_n, \end{aligned} \tag{5}$$

where each f_i is a convex function. Throughout this paper we will refer to (5) as the dual problem and the original problem (2) as the primal problem. The Isotonic Optimization problem is generalization of the well-known Isotonic Regression problem (where the f_i are Euclidean distance functions) and the algorithms described in the paper will tackle this form.

3.1 Projection onto Signed Permutahedron and Norm Balls

We can reduce the problem of projecting onto a signed permutahedron to the dual problem (5). This is a special case of a known result relating separable optimization problems over the base polyhedron of a submodular function (e.g. permutahedron) and the corresponding symmetric submodular polyhedron (e.g. signed permutahedron). See Proposition 8.9 of Bach (2013)). We will provide another derivation of this result to keep the presentation consistent.

We will assume that the convex function ϕ is *sign-invariant* (that is, $\phi(u) = \phi(-u)$).

The following variant of Lemma 3.1 applies here.

Lemma 3.2. *Let x' be the projection of z onto the permutahedron under a uniformly separable Bregman divergence defined by a sign-invariant ϕ. Then $\operatorname{sgn}(x'_i) = \operatorname{sgn}(z_i)$ for all i. Furthermore, if $|z_1| \geq |z_2| \geq \ldots \geq |z_n|$, we have $|x'_1| \geq |x'_2| \geq \ldots \geq |x'_n|$.*

The previous lemma and the fact that ϕ is sign-invariant allows us to assume without loss of generality that $z \geq 0$. The projection subproblem is identical to (2) except that the final equality constraint is replaced by a $\leq$ inequality. The dual is the same as (3) with the addition of a nonnegativity constraint $y_n \leq 0$, that is,

$$\begin{aligned} &\min_{y \in \mathbb{R}^n} \ \sum_{i=1}^{n} f_i(y_i) \\ &\text{such that } y_1 \leq y_2 \leq \ldots \leq y_n \leq 0, \end{aligned} \tag{6}$$

We can solve (6) by truncating the solution of (5).

Theorem 3.3. *Let $y^A \in \mathbb{R}^n$ be an optimal solution to problem (5). We get an optimal solution to problem (6) by truncating the positive values of y^A to zero.*

4 Pooling and Bregman Divergences

The algorithms for the Isotonic Optimization problem will rely on our ability to compute the following function efficiently:

Definition 4.1. *Given strictly convex univariate functions f_i, the function $\texttt{PoolV}_f$ is defined on each set S as follows:*

$$\texttt{PoolV}_f(S) = \arg\min_{\gamma} \sum_{i \in S} f_i(\gamma). \tag{7}$$

We let $\texttt{PoolV}_{\phi,z}$ denote the $\texttt{PoolV}_f$ obtained by setting $f_i(\gamma)$ to $d_i^*(\gamma) - \gamma c_i$, where $d_i(x_i) = \Delta_\phi(x_i, z_i)$.

The following lemma follows directly from the strict convexity of the functions f_i and will be useful in subsequent sections.

Lemma 4.2. *Let S_1, S_2 be disjoint sets such that $\texttt{PoolV}_f(S_1) < \texttt{PoolV}_f(S_2)$. Then*

$$\texttt{PoolV}_f(S_1) < \texttt{PoolV}_f(S_1 \cup S_2) < \texttt{PoolV}_f(S_2).$$

We now analyze the cost of computing $\texttt{PoolV}_{\phi,z}(S)$ for different Bregman projections. The minimizer of $\sum_{i \in S} f_i(\gamma) = \sum_{i \in S}(d_i^*(\gamma) - \gamma c_i)$ is unique due to the strict convexity of ϕ. It satisfies the following optimality condition, obtained by setting the derivative of $\sum_{i \in S} f_i(\gamma)$ to zero.

Lemma 4.3. $\texttt{PoolV}_{\phi,z}(S)$ *satisfies*

$$\sum_{i \in S} (\nabla\phi)^{-1}(\gamma + \nabla\phi(z_i)) = \sum_{i \in S} c_i. \tag{8}$$

Bregman divergences with incremental $\texttt{PoolV}$. The following property provides a sufficient condition for the Isotonic Optimization algorithms presented here to find the exact solution quickly. This allows us to give a unified presentation to the results on individual Bregman divergences, such as the work on unnormalized relative entropy (Suehiro et al., 2012) and Euclidean distance (Yasutake et al., 2011).

Definition 4.4. *Let $S \subset [n]$ be a set of indices. We say that a Bregman divergence ϕ has the* incremental $\texttt{PoolV}$ *property if we can compute $\texttt{PoolV}_{\phi,z}(S)$ in $O(1)$ time given the values $\sum_{i \in S} c_i$, $\sum_{i \in S} z_i$, and $|S|$.*

The reason for calling them *incremental* is as follows: For two disjoint sets S_1 and S_2, if we have stored $\sum_{i \in S_k} c_i$, $\sum_{i \in S_k} z_i$, and $|S_k|$ for each $k \in \{1, 2\}$, then we can compute in $O(1)$ time these values for the set $S_3 = S_1 \cup S_2$.

We now describe some Bregman divergences with the incremental $\texttt{PoolV}$ property, and use Lemma 4.3 to provide closed-form expressions for $\texttt{PoolV}_{\phi,z}$.

Euclidean distance: $\phi(u) = \frac{1}{2}u^2$, $\nabla\phi(u) = (\nabla\phi)^{-1}(u) = u$. We thus obtain

$$\texttt{PoolV}_{\phi,z}(S) = \sum_{i \in S}(c_i - z_i)/|S|.$$

Unnormalized relative entropy: $\phi(u) = u \ln u - u$, $\nabla\phi(u) = \ln u$, and $(\nabla\phi)^{-1}(u) = e^u$. Therefore

$$\texttt{PoolV}_{\phi,z}(S) = \ln \frac{\sum_{i \in S} c_i}{\sum_{i \in S} z_i}.$$

KL divergence: Krichene et al. (2015) study the following generalization of KL divergence, obtained from $\phi(u) = (u + \epsilon)\ln(u + \epsilon)$ for $\epsilon \geq 0$. (KL divergence has $\epsilon = 0$.) We have $\nabla\phi(u) = \ln(u + \epsilon) + 1$ and $(\nabla\phi)^{-1}(u) = e^{u-1} - \epsilon$, which implies

$$\texttt{PoolV}_{\phi,z}(S) = \ln \frac{\sum_{i \in S}(c_i + \epsilon)}{\sum_{i \in S}(z_i + \epsilon)} - 1.$$

General Uniformly Separable Bregman Divergences. For these divergences (e.g. binary relative entropy), we may be unable to compute $\texttt{PoolV}_{\phi,z}$ exactly, but we can obtain an approximate solution to problem (5) by solving $\texttt{PoolV}_{\phi,z}$ approximately. Details are provided in Section 6.

5 Pooling Algorithms for Incremental PoolV: PAV and MergeAndPool

5.1 Pool Adjacent Violators (PAV) algorithm

We can solve the dual problem (5) using the well-known *Pool Adjacent Violators* (PAV) algorithm; see Algorithm 1.

Algorithm 1 PAV Algorithm

```
  Input: Convex functions $f_i : \mathbb{R} \to \mathbb{R}$ for $i \in [n]$
  $P \leftarrow \{\{i\} \mid i \in [n]\}$
  $y_i \leftarrow \texttt{PoolV}_f(\{i\})$ for all $i \in [n]$
  while $\exists$ indices $i, i+1 \in L$ where $y_i > y_{i+1}$ do
    Let $I_{(i)}$ and $I_{(i+1)}$ be the intervals in $P$ containing indices $i$ and $i+1$ respectively
    Remove $I_{(i)}$, $I_{(i+1)}$ from $P$ and add $I_{(i)} \cup I_{(i+1)}$.
    $y_{I_{(i)} \cup I_{(i+1)}} \leftarrow \texttt{PoolV}_f(I_{(i)} \cup I_{(i+1)})$
  end while
  return $y$
```

Theorem 5.1. *(Best et al., 2000) PAV terminates within n iterations with a dual feasible solution y.*

The cost of PAV depends on how the adjacent violators are found and the cost of $\texttt{PoolV}_{\phi,z}$. For Bregman divergences with the incremental $\texttt{PoolV}_{\phi,z}$ property, it takes $O(1)$ time to obtain $\texttt{PoolV}_{\phi,z}$ in each iteration. All that is left to show is that we can find adjacent violators efficiently. In the appendix, we provide a description of the linked-list implementation of PAV that does so.

Theorem 5.2. *Given a sorted vector z, we can solve problem (2) for uniformly separable Bregman divergences with incremental* PoolV *using* PAV *in time $O(n)$.*

Factoring in the initial sorting step, the overall cost of the algorithm to solve problem (1) is $O(n \log n)$.

Example for Euclidean projection. Given a vector z that has been sorting in descending order, we first initialize vector y as $c - z$. The PAV algorithm now picks any pair of adjacent terms y_i, y_{i+1} such that $y_i > y_{i+1}$ (i.e. adjacent violators), takes the corresponding intervals $I_{(i)}$ and $I_{(i+1)}$, and sets the y_k terms for $k \in I_{(i)} \cup I_{(i+1)}$ to the average of these terms.

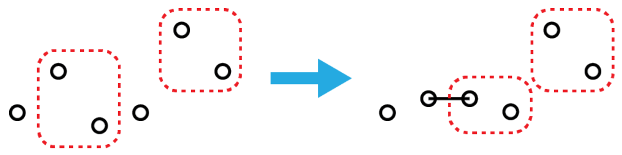

Figure 1: One iteration of the PAV algorithm. The circles represent y values, the dotted boxes represent adjacent violators, and the horizontal line represents an interval I.

5.2 MergeAndPool: Divide-and-Conquer Pooling for Incremental PoolV

We introduce MergeAndPool (Algorithm 2), a divide-and-conquer algorithm for the dual problem (5) that merges intervals in a bottom-up manner. Unlike the PAV algorithm, which requires a sorted input, the MergeAndPool algorithm can easily be modified to take advantage of the number of unique entries in the vector c for our permutahedron $\mathcal{PH}(c)$. We will first describe the algorithm for the case where c has n distinct entries. For MergeAndPool, the input functions f_i have to be strictly convex, as is true for Bregman divergences.

In each outer loop of the MergeAndPool algorithm, it makes a pass over all the elements of y, performing Merge on pairs of adjacent intervals of terms. The result of each call to $\texttt{Merge}_f$ is a single interval of terms that is sorted in the correct order. More precisely, we give as input to the $\texttt{Merge}_f$ subroutine two adjacent intervals I_1, I_2 and a vector $y_{I_1 \cup I_2}$, where y_{I_1} and y_{I_2} are optimal solutions to problem (5) when restricted to their respective intervals. Figure 2 illustrates the

Algorithm 2 MergeAndPool Algorithm for sorted z

```
Input:  $n$ strictly convex functions $f_i : \mathbb{R} \to \mathbb{R}$
$P \leftarrow \{\{i\} \mid i \in [n]\}$
$y_i \leftarrow \texttt{PoolV}_f(\{i\})$ for all $i \in [n]$
while $|P| > 1$ do
  $Q \leftarrow P,\ P \leftarrow \emptyset$
  while $|Q| > 1$ do
    $I_1, I_2 \leftarrow$ first two intervals in $Q$
    $y_{I_1 \cup I_2} \leftarrow \texttt{Merge}_f(I_1, I_2, y_{I_1 \cup I_2})$
    $Q \leftarrow Q \setminus \{I_1, I_2\}$ , $P \leftarrow P \cup \{I_1 \cup I_2\}$
  end while
  $P \leftarrow P \cup Q$
end while
return $y$
```

input and output of Merge and Figure 3 gives a high-level overview of the MergeAndPool algorithm.

Algorithm 3 $\texttt{Merge}_f$ subroutine

```
Input:  Adjacent intervals $I_1, I_2$ and $y_{I_1 \cup I_2}$
$\mathcal{Y} \leftarrow$ all $y_i$ values in $y_{I_1 \cup I_2}$ {Search Range}

{true $\texttt{PoolV}_f(S)$ value is in $[\min(\mathcal{Y}), \max(\mathcal{Y})]$}
while $|\mathcal{Y}| > 2$ do
  $\gamma_{\text{test}} \leftarrow \lceil |\mathcal{Y}|/2 \rceil$th smallest value in $\mathcal{Y}$
  $S_{\text{test}} \leftarrow \{i \in I_1 \mid y_i \geq \gamma_{\text{test}}\} \cup \{i \in I_2 \mid y_i \leq \gamma_{\text{test}}\}$
  if $\gamma_{\text{test}} < \texttt{PoolV}_f(S_{\text{test}})$ then
    $\mathcal{Y} \leftarrow \{y \in \mathcal{Y} \mid y \geq \gamma_{\text{test}}\}$
  else
    $\mathcal{Y} \leftarrow \{y \in \mathcal{Y} \mid y \leq \gamma_{\text{test}}\}$
  end if
end while

for $\gamma_{\text{test}} \in \{\min(\mathcal{Y}), \max(\mathcal{Y})\}$ do
  $S_{\text{test}} \leftarrow \{i \in I_1 \mid y_i \geq \gamma_{\text{test}}\} \cup \{i \in I_2 \mid y_i \leq \gamma_{\text{test}}\}$
  if $\gamma_{\text{test}} = \texttt{PoolV}_f(S_{\text{test}})$ then
    $y_{S_{\text{test}}} \leftarrow \texttt{PoolV}_f(S_{\text{test}})$
    return $y_{I_1 \cup I_2}$
  end if
end for
$S \leftarrow \{i \in I_1 \mid y_i > \min(\mathcal{Y})\} \cup \{i \in I_2 \mid y_i < \max(\mathcal{Y})\}$
$y_S \leftarrow \texttt{PoolV}_f(S)$
return $y_{I_1 \cup I_2}$
```

The following lemma about the output of $\texttt{Merge}_f$ is the key to proving the correctness of MergeAndPool. The proof of this lemma relies on the correctness of the PAV algorithm, and will demonstrate that any pooling decision that $\texttt{Merge}_f$ makes is one that the PAV algorithm *can* make in the process of solving the same problem.

Lemma 5.3. *Consider adjacent intervals I_1, I_2 and vector y, where y_{I_1} and y_{I_2} are the optimal solution to dual problem (5) when restricted to only the in-*

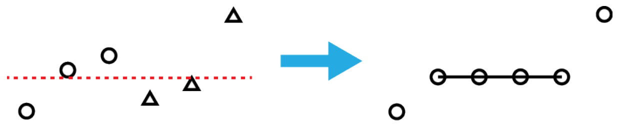
Figure 2: The input and output of the `Merge` algorithm. The circles and triangles represent y values corresponding to different intervals. The algorithm uses binary search to find the correct γ value (represented here by the red dotted line), then pools elements in the left interval that are above γ with elements in the right interval that are below γ.

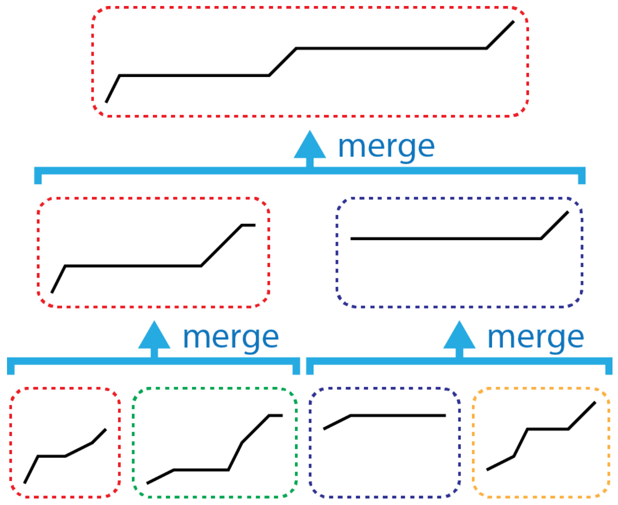

Figure 3: A high-level look at the `MergeAndPool` algorithm. Each box corresponds to one interval and the black lines inside represent the y values. The algorithm starts from the bottom, and each row represents one iteration of the outer loop.

dices in I_1 and I_2, respectively. The output $y_{I_1 \cup I_2}$ of `Merge`$_f(I_1, I_2, y_{I_1 \cup I_2})$ *gives the optimal solution to problem (5) when restricted to the indices in $I_1 \cup I_2$.*

Theorem 5.4. `MergeAndPool` *returns the solution y' to the dual problem (5).*

The outer loop of `MergeAndPool` is executed $O(\log n)$ times. We can show that a call to `Merge`$_f$ takes $O(|I_1| + |I_2|)$ time for Bregman divergences with the incremental `PoolV`$_{\phi,z}$ property by noting that the work needed halves with each iteration of the main loop.

Proposition 5.5. *The running time of* `MergeAndPool` *is $O(n \log n)$ for uniformly separable Bregman divergences ϕ with incremental* `PoolV`$_{\phi,z}$ *cost.*

5.3 `MergeAndPool` and Handling Repeat Entries in c

The results in the previous section match the best known results for the Euclidean projection onto the permutahedron and signed permutahedron when we factor in the cost of the initial sort, but for the simplex and ℓ_1-ball, one can avoid performing an initial sorting step and can compute the projection in just $O(n)$ time. We now describe the necessary modification to `MergeAndPool`.

When handling vector c with repeat entries, we will write c as $(c_{(1)}, \dots, c_{(1)}, c_{(2)}, \dots, c_{(2)}, c_{(3)}, \dots, c_{(d)})$, where $c_{(1)} > c_{(2)} > \dots > c_{(d)}$. Let n_i denote the number of $c_{(i)}$ entries present in c.

Our approach will rely on the fact that selecting a particular ranked item from a unsorted vector can be done in $O(n)$ time. This can be done in deterministic time by using the canonical "median of medians" algorithm or in expected time by a random pivot strategy.

Theorem 5.6. *(Blum et al., 1973) Selecting the ith largest element element in an unsorted list of length n can be done in $O(n)$ time.*

The next result follows from applying this fast selection algorithm recursively.

Lemma 5.7. *We can sort the entries of vector z into d groups in $O(n \log d)$ time such that the ith group has n_i elements and for each z_j in group i and z_k in group $i+1$, we have $z_j \geq z_k$.*

To solve the projection problem (1) when we have a vector c with repeat entries, we begin by applying the partial sort of Lemma 5.7 to our input vector z. We now modify `MergeAndPool` to *initialize partition P with the corresponding d groups of indices*, instead of just singleton sets. To prove that this strategy results in the correct output, we note that the `Merge`$_f$ subroutine does not require the z vector to be fully sorted. In fact, we can permute any of the elements within each interval I_1, I_2, and `Merge`$_f$ will still return the same result. The only things that is required are that the two intervals I_1, I_2 are adjacent and in the right order, and that the input y_{I_1}, y_{I_2} values are correct when considering the problem restricted to their respective intervals. Let $I_1, I_2, \dots, I_d$ denote the intervals of indices corresponding to the d groups. The following lemma implies that the initialization of each y_i to `PoolV`$_{\phi,z}(i)$ results in the correct y_{I_k} for the dual problem (5) over just the I_k indices for all $k \in [d]$.

Lemma 5.8. *For two indices i and j, if $c_i = c_j$ and $z_i \geq z_j$, then* `PoolV`$_{\phi,z}(\{i\}) \leq$ `PoolV`$_{\phi,z}(\{j\})$.

The modified `MergeAndPool` requires only $O(\log d)$ iterations of the outer loop, and the cost of each call to the $\texttt{Merge}_f$ is still $O(|I_1| + |I_2|)$ due to Theorem 5.6.

Theorem 5.9. *We can compute the projection x' onto the permutahedron $\mathcal{PH}(c)$ under any incremental uniformly separable Bregman divergence in time $O(n \log d)$.*

6 Handling General Uniformly Separable Bregman Divergences

Definition 6.1. *We say a vector $\hat{y}$ is an ϵ-close (in ℓ_∞-norm) solution to a problem if there is a solution y' to the problem such that $\|\hat{y} - y'\|_\infty < \epsilon$.*

Our goal in this section is to describe algorithms for general uniformly separable Bregman divergences that give us an ϵ-close solution for the dual problem. Let $l, u \in \{\epsilon k \mid k \in \mathbb{Z}\}$ denote lower and upper bounds on all the values of the solution y' to problem (5), and let $U = u - l$. Best et al. (2000) and Ahuja and Orlin (2001) provide two ways of obtaining an ϵ-close solution on the lattice $\mathcal{L} = \{\epsilon k \mid k \in \mathbb{Z}\}^n$.

The first method is to use `PAV`. We can replace functions f_i with continuous piecewise linear functions f_i^ϵ that interpolate between the f_i values at the points in $\{\epsilon k \mid k \in \mathbb{Z}\}$. If each term in the sum of the objective function can be evaluated in $O(1)$ time, we can use binary search to find a minimizer of $\sum_{i \in S} f_i(\gamma)$ in $O(|S| \log \frac{U}{\epsilon})$. This results in a $O(n^2 \log \frac{U}{\epsilon})$ approach.

One can do better. For a given vector $y \in \mathbb{R}^n$, let $\mathcal{L}(y)$ denote the vector obtained by rounding down each term in y to the nearest term in $\{\epsilon k \mid k \in \mathbb{Z}\}$. Ahuja and Orlin (2001) introduced an efficient scaling variant of the `PAV` algorithm:

Theorem 6.2. *(Ahuja and Orlin, 2001) Let y' denote the solution to problem (5). We can find $\mathcal{L}(y')$ in $O(n \log \frac{U}{\epsilon})$ time using* `ScalingPAV`.

For the projection problem, this result requires the points to be sorted, resulting in an overall complexity of $O(n \log \frac{nU}{\epsilon})$. We can further modify `MergeAndPool` to obtain a faster algorithm for small values of d.

By replacing the subroutine $\texttt{Merge}_f$ with ϵ-$\texttt{Merge}_f$ (Algorithm 4) and initializing the points y with their rounded down `PoolV` values, `MergeAndPool` will return an ϵ-close solution to problem (5). ϵ-$\texttt{Merge}_f$ works by binary searching over $\mathcal{L}$ instead of the y values. The correctness of this subroutine can be established from the proof of Lemma 5.3 and the fact that the gradient shows if $\texttt{PoolV}_f(S_{\text{test}})$ is higher/lower than γ_{test}.

If each ∇f_i can be evaluated in $O(1)$ time, then the complexity required in each iteration of the loop requires $O(|I_1| + |I_2|)$ time, leading to an complexity of $O((|I_1| + |I_2|) \log \frac{U}{\epsilon})$ for each call to ϵ-$\texttt{Merge}_f$. This leads to the following theorem:

Algorithm 4 ϵ-$\texttt{Merge}_f$ subroutine

```
Input: Adjacent intervals $I_1, I_2$ and $y_{I_1 \cup I_2}$
$\mathcal{Y} \leftarrow \{\epsilon k \mid k \in \mathbb{Z}\} \cap [l, u]$
while $|\mathcal{Y}| > 2$ do
  $\gamma_{\text{test}} \leftarrow \lceil |\mathcal{Y}|/2 \rceil$th smallest value in $\mathcal{Y}$
  $S_{\text{test}} \leftarrow \{i \in I_1 \mid y_i \geq \gamma_{\text{test}}\} \cup \{i \in I_2 \mid y_i < \gamma_{\text{test}}\}$
  if $\sum_{i \in S_{\text{test}}} \nabla f_i(\gamma_{\text{test}}) = 0$ then
    $y_S \leftarrow \gamma_{\text{test}}$
    return $y_{I_1 \cup I_2}$
  else if $\sum_{i \in S_{\text{test}}} \nabla f_i(\gamma_{\text{test}}) < 0$ then
    $\mathcal{Y} \leftarrow \{v \in \mathcal{Y} \mid v \geq \gamma_{\text{test}}\}$
  else
    $\mathcal{Y} \leftarrow \{v \in \mathcal{Y} \mid v \leq \gamma_{\text{test}}\}$
  end if
end while

$\gamma_{\text{test}} \leftarrow \max(\mathcal{Y})$
$S_{\text{test}} \leftarrow \{i \in I_1 \mid y_i \geq \gamma_{\text{test}}\} \cup \{i \in I_2 \mid y_i < \gamma_{\text{test}}\}$
if $\sum_{i \in S_{\text{test}}} \nabla f_i(\gamma_{\text{test}}) = 0$ then
  $y_{S_{\text{test}}} \leftarrow \gamma_{\text{test}}$
else
  $S \leftarrow \{i \in I_1 | y_i \geq \min(\mathcal{Y})\} \cup \{i \in I_2 | y_i < \min(\mathcal{Y})\}$
  $y_S \leftarrow \min(\mathcal{Y})$
end if
return $y_{I_1 \cup I_2}$
```

Theorem 6.3. *Let y' denote the solution to problem (5). We can find $\mathcal{L}(y')$ in $O((n \log d) \log \frac{U}{\epsilon})$ time using* `MergeAndPool`.

The same technique for handling repeat entries in c for $\texttt{Merge}_{\phi,z}$ also apply here, hence the complexity bound for the projection problem remains $O((n \log d) \log \frac{U}{\epsilon})$. This result improves on Ahuja and Orlin (2001) when $(\frac{U}{\epsilon})^{\log d - 1} = o(n)$, and matches the $O(n \log \frac{U}{\epsilon})$ algorithm (Krichene et al., 2015) for the simplex case. In the full version, we will describe a modification of `ScalingPAV` that achieves a better complexity of $O(n \log \frac{dU}{\epsilon})$.

7 Acknowledgements

We thank Okan Akalin and Ravi Ganti for helpful comments and suggestions on the presentation of this paper. We thank the anonymous referees for feedback that improved the paper and its presentation. Lim's work on this project was supported by NSF Award DMS-1216318 and Award EM02076 from Exxon Mobil Corp. Wright's work supported by NSF Awards DMS-1216318 and IIS-1447449, ONR Award N00014-13-1-0129, AFOSR Award FA9550-13-1-0138, and Subcontract 3F-30222 from Argonne National Laboratory.

References

Ahuja, R. K. and Orlin, J. B. (2001). A Fast Scaling Algorithm for Minimizing Separable Convex Functions Subject to Chain Constraints. *Operations Research*, 49(5):784–789.

Ailon, N. (2014). Improved Bounds for Online Learning Over the Permutahedron and Other Ranking Polytopes. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 29–37.

Ailon, N., Hatano, K., and Takimoto, E. (2014). Bandit online optimization over the permutahedron. In Auer, P., Clark, A., Zeugmann, T., and Zilles, S., editors, *Algorithmic Learning Theory*, volume 8776 of *Lecture Notes in Computer Science*, pages 215–229. Springer International Publishing.

Bach, F. (2013). Learning with submodular functions: A convex optimization perspective. *Foundations and Trends in Machine Learning*, 6(2-3):145–373.

Barlow, R. (1972). *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression*. J. Wiley.

Best, M. J., Chakravarti, N., and Ubhaya, V. A. (2000). Minimizing Separable Convex Functions Subject to Simple Chain Constraints. *SIAM Journal on Optimization*, 10(3):658–672.

Blum, M., Floyd, R. W., Pratt, V., Rivest, R. L., and Tarjan, R. E. (1973). Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461.

Bogdan, M., van den Berg, E., Su, W., and Candes, E. (2013). Statistical estimation and testing via the sorted L1 norm. *arXiv:1310.1969*.

Brucker, P. (1984). An O(n) algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l1-ball for learning in high dimensions. In McCallum, A. and Roweis, S., editors, *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, pages 272–279. Omnipress.

Fujishige, S. (1984). Submodular systems and related topics. In Korte, B. and Ritter, K., editors, *Mathematical Programming at Oberwolfach II*, pages 113–131. Springer, Berlin, Heidelberg.

Goemans, M. (2015). Smallest compact formulation for the permutahedron. *Mathematical Programming, Series A*, 153(1):5–11.

Groenevelt, H. (1991). Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *European Journal of Operational Research*, 54(2):227–236.

Hazan, E. (2012). The convex optimization approach to regret minimization. In Sra, S., Nowozin, S., and Wright, S. J., editors, *Optimization for Machine Learning*, pages 287–304. MIT press.

Krichene, W., Krichene, S., and Bayen, A. (2015). Efficient bregman projections onto the simplex. In *2015 IEEE 54rd Annual Conference on Decision and Control (CDC)*.

Negrinho, R. and Martins, A. (2014). Orbit regularization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3221–3229. Curran Associates, Inc.

Suehiro, D., Hatano, K., Kijima, S., Takimoto, E., and Nagano, K. (2012). Online prediction under submodular constraints. In Bshouty, N., Stoltz, G., Vayatis, N., and Zeugmann, T., editors, *Algorithmic Learning Theory*, volume 7568 of *Lecture Notes in Computer Science*, pages 260–274. Springer Berlin Heidelberg.

Wang, W. and Lu, C. (2015). Projection onto the capped simplex. *arXiv:1503.01002*.

Yasutake, S., Hatano, K., Kijima, S., Takimoto, E., and Takeda, M. (2011). Online linear optimization over permutations. In Asano, T., Nakano, S.-i., Okamoto, Y., and Watanabe, O., editors, *Algorithms and Computation*, volume 7074 of *Lecture Notes in Computer Science*, pages 534–543. Springer Berlin Heidelberg.

Zeng, X. and Figueiredo, M. A. T. (2014). The Ordered Weighted ℓ_1 Norm: Atomic Formulation, Projections, and Algorithms. *arXiv:1409.4271*.