Fast Projection onto the Capped Simplex with Applications to Sparse Regression in Bioinformatics

Anonymous Author(s)
Affiliation
Address
email

Abstract

We consider the problem of projecting a vector onto the so-called k-capped simplex, which is a hyper-cube cut by a hyperplane. For an n-dimensional input vector with bounded elements, we found that a simple algorithm based on Newton's method is able to solve the projection problem to high precision with a complexity roughly about O(n), which has a much lower computational cost compared with the existing sorting-based methods proposed in the literature. We provide a theory for partial explanation and justification of the method.

We demonstrate that the proposed algorithm can produce a solution of the projection problem with high precision on large scale datasets, and the algorithm is able to significantly outperform the state-of-the-art methods in terms of run time (about 6-8 times faster than commercial software with respect to CUP time for a vector with 1 million variables or even more).

We further illustrate the effectiveness of the proposed algorithm on solving sparse regression in a bioinformatics problem. Empirical results on the GWAS dataset (with 1,500,000 single-nucleotide polymorphisms) show that, when using the proposed method to accelerate the Projected Quasi-Newton (PQN) method, the accelerated PQN algorithm is able to handle huge-scale regression problem and it is more efficient (about 3-6 times faster) than the current state-of-the-art methods.

1 Introduction

The k-capped simplex [15] is defined as $\Delta_k := \left\{ \mathbf{x} \in \mathbb{R}^n \;\middle|\; \mathbf{x}^\top \mathbf{1} \leq k,\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{1} \right\}$, where $k \in \mathbb{R}$ is an input parameter, $\mathbf{0}$ and $\mathbf{1}$ are vectors of zeros and vector of ones in $\mathbb{R}^n$, respectively (resp.), and the sign $\leq$ is taken element-wise. Geometrically, for some $k > 0$, the set Δ_k represents an n-dimensional hyper-cube $[\mathbf{0}, \mathbf{1}]^n$ with a corner chopped off by the half-space $\mathbf{x}^\top \mathbf{1} \leq k$. If the half-space constraint $\mathbf{x}^\top \mathbf{1} \leq k$ in Δ_k is replaced by the hyper-plane constraint $\mathbf{x}^\top \mathbf{1} = k$, for some $k > 0$, this set represents the slice of the hyper-cube by a hyper-plane, and we denote this set as $\Delta_k^=$. In this paper, we consider the problem of projection onto the capped simplex:

$$\min_{\mathbf{x}} \ : \ \left\{ \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 \text{ s.t. } \mathbf{x} \in \Delta_k \text{ (or } \Delta_k^=) \right\}. \tag{1}$$

Remark 1. ***(On the input*** k***)*** *In this paper, we assume $0 < k < \infty$ is properly selected such that the set Δ_k (and $\Delta_k^=$) has a nonempty interior, and therefore the projection is feasible. I.e., we assume $\left\{ \mathbf{p} \in \mathbb{R}^n \mid \mathbf{p}^\top \mathbf{1} \leq k \right\} \cap \left\{ \mathbf{q} \in \mathbb{R}^n \mid \mathbf{0} \leq \mathbf{q} \leq \mathbf{1} \right\} \neq \varnothing$ (and similarly for $\Delta_k^=$). In this setting, a unique global minimizer $\mathbf{x}^*$ of Problem(1) exists, since the function $\|\mathbf{x} - \mathbf{y}\|_2^2$ is coercive. The value of k is closely related to the root of a piecewise-linear equation that plays a critical role in the solution of Problem(1), see discussion in § 2.*

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Problem (1) generalizes the projection onto the probability simplex [4, 5] which has many applications. In this paper, we consider the application in sparse regression in bioinformatics, see § 4.

To the best of our knowledge, for an input vector $\mathbf{y} \in \mathbb{R}^n$, the state-of-the-art algorithm [15] for solving the Projection (1) has a time complexity of $\mathcal{O}(n^2)$. We found that a simple heuristic algorithm based on Newton's method is able to solve the same problem to high precision with a complexity of $\mathcal{O}(n)$ on average, based on exploiting the 2nd-order information of a minimization problem on the Lagrangian multiplier of the Projection (1); see § 2.3.

Contributions We proposed a simple algorithm based on Newton's method, as an alternative to the sorting-based method to solving Problem (1). We first convert the problem into a scalar minimization problem, then we show that such scalar problem can be solved by Newton's iteration, see Theorem 1. We then show that the proposed method is superior to the current state-of-the-art method in terms of run time. For a vector with dimension 10^6, the proposed method is about 6-8 times faster than a commercial solver. Furthermore, we give numerical results to show the effectiveness of the proposed method on solving sparse regression in a bioinformatics problem on a real-world dataset with a data size of 1,500,000.

Scope, limitation, and impact We emphasize that the proposed method is not aimed at replacing all the well-established sorting-based methods in the literature, but to provide an alternative method for solving the projection problem in a special case. To be specific, we found that, when the input vector $\mathbf{y}$ has bounded elements (i.e., $|y_i| \leq \alpha$, for α up to 1000), the proposed algorithm of this paper is able to solve the projection problem much faster than the existing methods in the literature. It is important to note that, when the bound α becomes very huge (say a million), the proposed method sometimes converges slower than existing methods. Although limited in the scope of application, however, we argue that the proposed method is still very useful for applications that the input vectors are bounded. Using bioinformatics as an example, on an ordinary PC, we are able to reduce the run time on identifying the genetic factors that contribute to dental caries (i.e., tooth decay) from half-hour to just 3 minutes; see § 4.

Paper organization In § 2, we give the background material, show the main theorem, and discuss the proposed algorithm. Next, in § 3, we show experimental results to verify the efficiency of the proposed algorithm on solving the projection problem. Then, in § 4, we illustrative the effectiveness of the proposed method on solving sparse regression in a large bioinformatics dataset. We conclude the paper in § 5 and provide supplementary material in the appendix.

2 Projection onto the capped simplex

In this section, we discuss how to solve Problem (1). By introducing a Lagrangian multiplier γ, we convert (1) to a scalar problem in the form of $\min_\gamma \omega(\gamma)$. Then, we provide a theorem for characterizing the function $\omega(\gamma)$. More specifically, we show that $\omega(\gamma)$ is n-smooth, i.e., the Lipschitz constant of the slope $\omega'(\gamma)$ grows linearly as the dimension n, which makes the gradient method ineffective to solve the problem due to very small stepsize when the input vector $\mathbf{y}$ has many variables. This motivates us to switch to the 2nd-order method, i.e., Newton's method, to solve the scalar problem. We finish this section by discussing the properties of Newton's update for solving the scalar problem $\min_\gamma \omega(\gamma)$.

2.1 Reformulate the projection problem to a scalar minimization problem

Consider the case of projection onto $\Delta_k^{=}$[1]. Problem (1) is equivalent to Problem (2): by introducing a Lagrangian multiplier γ corresponds to the equality constraint, we form a partial Lagrangian $\mathcal{L}(\mathbf{x}, \gamma)$, and arrive at the min-max problem (2), where we keep the inequality constraint on $\mathbf{x}$.

$$\min_{\mathbf{0} \leq \mathbf{x} \leq \mathbf{1}} \max_{\gamma \in \mathbb{R}} : \left\{ \mathcal{L}(\mathbf{x}, \gamma) = \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|^2 + \gamma(\mathbf{x}^\top \mathbf{1} - k) \right\}, \tag{2}$$

[1]For projection onto Δ_k, we can use the same approach by adding nonnegativity constraint $\gamma \geq 0$ in (2).

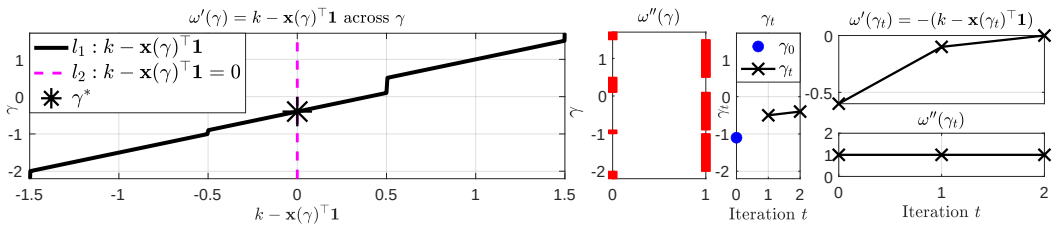

Figure 1: Illustration on a toy example with $\mathbf{y} = [0.1, 1.5, -1] \in \mathbb{R}^3, k = 1.5, \gamma_0 = -1.1$. The line l_1 intersects the piecewise-liner function l_2 at γ^*, which is the root of l_2. The core idea of the sorting-based methods is to find such root by sorting the elements of $\mathbf{y}$ and try each linear segment of l_1 to find the root. The sorting-method has a $\mathcal{O}(3)$ cost on the toy example. The proposed method (Algorithm 1) solves the problem in 2 iterations, hence it gains speed-up. Note that l_1, l_2 may not intersect each other for some $(\mathbf{y}, k)$, in this case, l_2 has no root and all the sorting-based methods do not work. However, the proposed algorithm can still produce an approximate solution.

Closed-form solution on $\mathbf{x}$ Let γ^* be the optimal γ in (2), then the minimizer $\mathbf{x}^*$ of (2) is

$$\mathbf{x}^*(\gamma^*) = \underset{\mathbf{0} \leq \mathbf{x} \leq \mathbf{1}}{\operatorname{argmin}} \left\{ \mathcal{L}(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{2} x_i^2 + (\gamma^* - y_i) x_i \right\} \overset{(4)}{=} \min \left\{ \mathbf{1}, [\mathbf{v}(\gamma^*)]_+ \right\}, \tag{3}$$

where $[\,\cdot\,]_+ = \max\{\,\cdot\,, 0\,\}$, both $\max, \min$ in (3) are taken element-wise, and $\mathbf{v}$ is defined as

$$\mathbf{v}(\gamma) = \mathbf{y} - \gamma \mathbf{1}. \tag{4}$$

As stated in the introduction, under a sufficiently large $k > 0$, such $\mathbf{x}^*$ exists in the relative interior of $\Delta_k^{=}$, which implies Slater's condition is true and thereby strong duality holds, i.e., we have the first equality sign in (3). Furthermore, we can swap the order of the min-max in (2) since it has a saddle point. By swapping the order of min-max, and using (3), we convert Problem (2) to a scalar minimization problem on γ:

$$\min_{\gamma \in \mathbb{R}} \; : \; \left\{ \omega(\gamma) := -\mathcal{L}\Big(\min \left\{ \mathbf{1}, [\mathbf{v}(\gamma)]_+ \right\}, \gamma \Big) \right\}, \tag{5}$$

where the negative sign comes from flipping the maximization on γ to minimization. Now, it is clear that if we can solve (5), we can construct the solution $\mathbf{x}^*$ as in (3) to solve Problem (1).

Many papers [4, 5, 15] in the literature follow such a direction on solving the projection: they solve Problem (5) using the sorting-based technique to find the rootof the piecewise-linear function $\omega'(\gamma) = 0$ (see Theorem 1 for the expression of ω').

In this paper, we solve (5) by Newton's method. It is important to note that we can solve the same problem using gradient descent, but we found that the gradient method has a slow convergence, because $\omega(\gamma)$ can have a huge smoothness constant (see Theorem 1), and thus the feasible stepsize for the gradient step such that the function value will decrease is very very small, which contributes to the slow convergence. We also observed that empirically the proposed method of this paper (see Algorithm 1) yields a much faster convergence (about 6-8 times faster) than the sorting-based methods, especially when the input vector $\mathbf{y}$ has a huge dimension; see § 3 for the experimental results.

Before we proceed, we illustrate the idea of solving the projection on a toy example, see Fig.1 and the caption therein.

2.2 Derivatives and the huge smoothness constant

Now we talk about the derivatives of $\omega(\gamma)$ with respect to (w.r.t.) γ. First, we show that $\omega(\cdot)$ is continuously differentiable and we derive its 1st and 2nd order derivatives. Then we show that the Lipschitz constant L of $\omega'(\cdot)$ is n, which is the dimension of the input vector $\mathbf{y} \in \mathbb{R}^n$. With a huge L, we observed that gradient descent has a slow convergence on minimizing $\gamma(\gamma)$, hence we propose to use the 2nd-order method, i.e., Newton's method, to solve (2); see § 2.3.

We now present the main theorem in this paper, which is about the derivatives of $\omega(\gamma)$.

Theorem 1. *The function $\omega(\gamma)$ is convex and twice differentiable with*

$$\omega'(\gamma) = k - \min\Big(\mathbf{1}, [\mathbf{v}(\gamma)]_+\Big)^\top \mathbf{1} \quad \textit{is } n\textit{-Lipschitz}, \quad \textit{and} \quad \omega''(\gamma) = \sum_{i=1}^{n} I_{0<v_i<1}, \tag{6}$$

where I is a indicator function that $I_A = 1$ if A is true and equals to 0 otherwise.

Remark 2. *It is important to note that the inequality signs in $0 < v_i < 1$ in the theorem are strict.*

Now we present two lemmas to prove the Theorem 1. For the first lemma, we first recall two useful equations: for all $a, b \in \mathbb{R}$,

$$\max\{a, b\} = \frac{1}{2}\Big(a + b + |a - b|\Big) \quad \text{and} \quad \min\{a, b\} = \frac{1}{2}\Big(a + b - |a - b|\Big). \tag{*}$$

Lemma 1. *$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have $\Big\| \min\big(\mathbf{1}, [\mathbf{x}]_+\big) - \min\big(\mathbf{1}, [\mathbf{y}]_+\big)\Big\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2$.*

Proof. By triangle inequality and (*), for all $a, b \in \mathbb{R}$, we have $\big|\min(a, 1) - \min(b, 1)\big| < \big|a - b\big|$ and $\big|[a]_+ - [b]_+\big| < \big|a - b\big|$. Therefore,

$$\big\| \min(\mathbf{1}, [\mathbf{x}]_+) - \min(\mathbf{1}, [\mathbf{y}]_+)\big\|_2 \leq \big\|[\mathbf{x}]_+ - [\mathbf{y}]_+\big\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2. \tag{7}$$

□

The second lemma is related to the notion of optimal value function [3, Section 4].

Lemma 2. *[Theorem 4.1 [3]] Let $\mathcal{X}$ be a metric space and $\mathcal{U}$ be a normed space. Suppose that for all $\mathbf{x} \in \mathcal{X}$, the function $\mathcal{L}(\mathbf{x}, \cdot)$ is differentiable and that $\mathcal{L}(\mathbf{x}, \gamma)$ and $D_\gamma \mathcal{L}(\mathbf{x}, \gamma)$ (the partial derivative of $\mathcal{L}(\mathbf{x}, \gamma)$ w.r.t. γ) are continuous on $\mathcal{X} \times \mathcal{U}$. Let Φ be a compact subset of $\mathcal{X}$. Then the optimal value function $\phi(\gamma) := \inf_{\mathbf{x}\in\Phi} \mathcal{L}(\mathbf{x}, \gamma)$, is (Hadamard) directionally differentiable. In addition, if $\forall \gamma \in \mathcal{U}$, $\mathcal{L}(\cdot, \gamma)$ has a unique minimizer $\mathbf{x}(\gamma)$ over Φ, then $\phi(\gamma)$ is differentiable at γ and the slope of $\phi(\gamma)$ w.r.t. γ is given by $\phi'(\gamma) = D_\gamma \mathcal{L}(\mathbf{x}(\gamma), \gamma)$.*

Now we are at the position to prove Theorem 1.

Proof of Theorem 1. To prove the differentiability of $\omega(\gamma)$, we apply Lemma 2 with $\mathcal{X} = \mathbb{R}^n, \mathcal{U} = \mathbb{R}$ and $\Phi = \{\mathbf{x} \in \mathcal{X} : \mathbf{0} \leq x \leq \mathbf{1}\}$. Immediately we have i. $\mathcal{L}(\mathbf{x}, \cdot)$ is differentiable; ii. $\mathcal{L}(\mathbf{x}, \gamma)$ and $D_\gamma \mathcal{L}(\mathbf{x}, \gamma) = \mathbf{x}^\top \mathbf{1} - k$ are continuous on $\mathcal{X} \times \mathcal{U}$; iii. Φ is a compact subset of $\mathcal{X}$; and iv. $\forall \gamma \in \mathcal{U}$, $\mathcal{L}(\mathbf{x}, \gamma)$ has a unique minimizer $\mathbf{x}(\gamma) = \min\Big(\mathbf{1}, \max\big(\mathbf{y} - \gamma\mathbf{1}, \mathbf{0}\big)\Big)$ over Φ. The last one follows that $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \gamma) = \mathbf{0}$ has a unique solution and further indicates that

$$\mathbf{x}(\gamma) = \min\Big(\mathbf{1}, \max\big(\mathbf{y} - \gamma\mathbf{1}, \mathbf{0}\big)\Big) = \operatorname*{argmin}_{\mathbf{x}\in\Phi} \mathcal{L}(\mathbf{x}, \gamma).$$

Applying Lemma 2 gives

$$\phi(\gamma) = \inf_{\mathbf{x}\in\Phi} \mathcal{L}(\mathbf{x}, \gamma) = \mathcal{L}\Big(\min\Big(\mathbf{1}, \max\big(\mathbf{y} - \gamma\mathbf{1}, \mathbf{0}\big)\Big), \gamma\Big)$$

is differentiable, and the slope is

$$\phi'(\gamma) = \min\Big(\mathbf{1}, \max\big(\mathbf{y} - \gamma\mathbf{1}, \mathbf{0}\big)\Big)^\top \mathbf{1} - k = -\omega'(\gamma). \tag{8}$$

We now show the derivative on $\omega'(\gamma)$ w.r.t. γ is the number of element of the vector $\mathbf{v} := \mathbf{y} - \gamma\mathbf{1}$ that is between 0 and 1. Let S_0 be the set of indices i that $v_i \leq 0$, let S_1 be the set of indices i that $v_i \geq 1$, and let S be the set of indices i that $0 < v_i < 1$. Now, as $\phi'(\gamma) \overset{(8)}{=} \sum_i \min\Big(1, \max\big(v_i, 0\big)\Big) - k$, expressing ϕ' using the sets S_0, S_1, S gives

$$\phi'(\gamma) = \left(\sum_{i\in S_0} 0 + \sum_{i\in S_1} 1 + \sum_{i\in S}(y_i - \gamma)\right) - k = |S_1| + \left(\sum_{i\in S} y_i\right) - |S|\gamma - k, \tag{9}$$

which gives $\phi'(\gamma) = -|S|\gamma$+constant, hence, $\omega''(\gamma) = -\phi''(\gamma) = -(-|S|)$ which gives (6).

It is clear that $\mathcal{L}(x, \gamma)$ in (2) is convex in $\mathbf{x}$ and the constraint set is closed and convex. For $(\mathbf{y}, k)$ that the feasible set for $\mathbf{x}$ is nonempty, by von Neumann Lemma [14], there exists a saddle point for Problem (2), as a result, $\phi(\gamma)$ is concave and $\omega(\gamma) = -\phi(\gamma)$ is convex.

Lastly, we show the Lipschitz constant of $\omega'(\gamma)$ is n: let $\mathbf{v}(\gamma) = \mathbf{y} - \gamma\mathbf{1}$, then

$$\begin{aligned}
\left|\omega'(\gamma_1) - \omega'(\gamma_2)\right| &= \left\| \min\left(\mathbf{1}, \left[\mathbf{v}(\gamma_1)\right]_+\right)^\top \mathbf{1} - \min\left(\mathbf{1}, \left[\mathbf{v}(\gamma_2)\right]\right)^\top \mathbf{1} \right\|_2 \\
&\leq \|\mathbf{1}\|_2 \cdot \left\| \min\left(\mathbf{1}, [\mathbf{v}(\gamma_1)]_+\right) - \min\left(\mathbf{1}, [\mathbf{v}(\gamma_2)]_+\right) \right\|_2 \\
&\overset{\text{Lemma 1}}{\leq} \sqrt{n}\|\mathbf{v}(\gamma_1) - \mathbf{v}(\gamma_2)\|_2 \ = \sqrt{n}\|\gamma_1\mathbf{1} - \gamma_2\mathbf{1}\|_2 = n|\gamma_1 - \gamma_2|.
\end{aligned}$$

□

2.3 Solving the projection by Newton's method

Now it is clear that to solve Problem (1), we solve the scalar minimization problem (5), and we proposed to use Newton's method to solve it, see Algorithm 1.

Algorithm 1: Newton's method for solving Problem (1) onto $\Delta_k^=$

Input: a nonzero vector $\mathbf{y}$, and initialize $\gamma_0 \in (\max(y) - 1, \max(y))$;
if $\mathbf{x} \in \Delta_k^=$ **then**
  return $\mathbf{x}^* = \mathbf{x}$.
else
  for $t = 1, ..., T$ **do**
    $\mathbf{v} = \mathbf{y} - \gamma_{t-1}\mathbf{1}$;
    Compute $\omega'(\gamma_{t-1}) = k - \min\left(\mathbf{1}, [\mathbf{v}]_+\right)^\top \mathbf{1}$ and $\omega''(\gamma_{t-1}) = \sum_{i=1}^{n} I_{0<v_i<1}$;
    $\gamma_t = \gamma_{t-1} - \dfrac{\omega'(\gamma_{t-1})}{\omega''(\gamma_{t-1})}$;
  end
  return $\mathbf{x}^* = \min(\mathbf{1}, [\mathbf{v}]_+)$
end

Before we discuss the case of projection onto Δ_k, we first give some remarks about Algorithm 1.

Quantify the performance of the algorithm: feasibility gap As the goal of solving Problem (1) is to find a vector inside the k-crapped simplex $\Delta_k^=$, hence we can define the feasibility gap as

$$\text{dist}(\mathbf{x}, \Delta_k^=)^2 := \text{dist}\Big(\mathbf{x}, \big\{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{u}^\top\mathbf{1} = k\big\}\Big)^2 + \text{dist}\Big(\mathbf{x}, \big\{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{0} \leq \mathbf{u} \leq \mathbf{1}\big\}\Big)^2.$$

By the definition of the algorithm, $\mathbf{x}^*$ is always inside the interval $[\mathbf{0}, \mathbf{1}]$, so we can simplify the feasibility gap as $\text{dist}\big(\mathbf{x}, \big\{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{u}^\top\mathbf{1} = k\big\}\big) = |\mathbf{x}^\top\mathbf{1} - k|$. For the toy example in Fig.1, we can see that such a gap is monotonically decreasing for the proposed algorithm.

Range for γ^* and $\omega''(\gamma^*)$ for general problem In general, the minimizer γ^* can take any value in $[-\infty, +\infty]$ (with ∞ included) for any $(\mathbf{y}, k)$. Also, by definition, $\omega''(\gamma^*)$ can take any value in $[0, n]$. Hence, cases like $\omega''(\gamma^*) = 0$ and/or $\gamma^* \in \{\pm\infty\}$ are possible. For example, suppose $k = n$, then for any bounded $\mathbf{y} \in \mathbb{R}^n$, one can find that a root of $\omega'(\gamma) = \omega''(\gamma) = 0$ is $\gamma = -\infty$. Furthermore, if the parameter k is too large (or too small) such that l_1 and l_2 in Fig.1 do not come cross each other, then it is possible for the algorithm to produce γ^* with $\omega''(\gamma^*) = 0$ and/or $\gamma* \in \{\pm\infty\}$.

On k, and the feasible range of γ In practice, k is a user-defined value that is related to the sparsity of the desired solution vector, so k is usually (much) smaller than n, therefore we focus on the case $k \ll n$ (in § 4, we will illustrate the experimental result on solving bioinformatics problems on real-world datasets, where we used $k \ll n$). In the following, we discuss γ, assuming $0 < k \ll n$ is

selected such that the Problem (1) is feasible to solve. Now we use the toy example in Fig.1 as an illustration. As Problem (1) is feasible, we have l_1 and l_2 in Fig.1 intersect at a $\gamma^* \notin \{\pm\infty\}$ (in the toy example, γ^* is unique). Now for the update of γ in Algorithm 1 to work, we need $\omega''(\gamma) \neq 0$. Notice that $\omega''(\gamma) = 0 \iff \{v_j \leq 0 \text{ or } v_j \geq 1\}\ \forall j$, i.e., the "bad region" such that $\omega''(\gamma) = 0$ is the intersection of all $\mathcal{I}_j \coloneqq \{\gamma \leq y_j - 1 \ \text{ or } \ y_j \leq \gamma\}$. Therefore, the feasible range for γ such that $\omega''(\gamma) \neq 0$ is the complement of such bad region, i.e,

$$\gamma \ \in \ \mathbb{R} \backslash \bigcap_j \mathcal{I}_j \ \subset \ \Big] \min(y_i) - 1, \max(y_i) \Big[, \tag{10}$$

where $]a, b[$ denotes the open interval $a < x < b$, and $\min(y_i), \max(y_i)$ are the smallest and largest value in $\mathbf{y}$, resp.. We can make use of this to design the initialization of γ_0. For example, we can simply set $\gamma_0 \in [\min(y_i) - 1 + \delta, \min(y_i) - \delta]$ for a small constant $\delta > 0$, which has the cost of $\mathcal{O}(1)$. Or, we can search γ_0 within the interval (10), with a worst-case complexity of $\mathcal{O}(n \log n)$, by sorting all the n elements of y_i and trying all the n feasible intervals. Refer to the toy example in Fig.1, we see that once γ_0 is initialized within the feasible range, the algorithm will produce a sequence $\{\gamma_t\}_{t \in \mathbb{N}}$ that converges to the minimizer γ^*. The convergence is guaranteed based on the convergence theory of Newton's method. Furthermore, if $\gamma^* \neq \infty$, the sequence $\{\gamma_t\}_{t \in \mathbb{N}}$ will not diverges to infinity since all $\omega''(\gamma) \neq 0$. We have the following corollary.

Corollary 2. *If $\gamma^* \neq \infty$ is the minimizer of Problem* (5)*, then $\omega''(\gamma^*) \neq 0$.*

Proof. As $\omega'(\gamma^*) = 0$, then $\gamma^* \overset{(8)}{=} \frac{k - |S_1| + \sum_{i \in S_{y_i}}}{|S|} \neq 0$, hence $|S| \neq 0$ and $\omega''(\gamma^*) \neq 0$. □

Computational cost The cost of the Algorithm 1 mainly comes from line 8, which has a per-iteration cost of $\mathcal{O}(n)$. The total cost inside the loop is $T\mathcal{O}(n)$. Theoretically speaking, it can take Newton's method many iterations to converge, but empirically, we observed that the algorithm usually converges in just a few iterations, so T is small. In comparison, it costs $\mathcal{O}(n^2)$ for the method proposed in [15] for solving the root of the piecewise-linear equation $\omega'(\gamma) = 0$.

Comparisons with existing methods To the best of our knowledge, all existing works solve the projection problem by finding the root of $\omega'(\gamma) = 0$ [4, 5, 15][2], in which they solve such a piecewise-linear equation by sorting-based technique. As these methods do solve $\omega'(\gamma) = 0$ exactly, so when the projection problem is feasible, they will eventually produce an exact result. In contrast, the proposed method is an iterative numerical algorithm, so it is possible that, due to numerical errors, the proposed method does not produce an exact solution but only an approximate solution. However, we argue that, for application with an input vector that has a huge dimension, the proposed method is good enough.

If the equation $\omega'(\gamma) = 0$ turns out to have no root for the input pair $(\mathbf{y}, k)$, in this case, all the sorting-based methods will fail, but the proposed algorithm is still able to produce an approximate solution, which can be useful in practice.

Lastly, as the proposed method has a lower computational cost, the method is very suitable for the projection of high dimensional vectors, see § 3. The proposed method also finds application on large-scale bioinformatics problems, see § 4 for details.

The case of projection onto Δ_k If we want to project onto Δ_k, the same algorithm can be used with the following twists: i. The feasibility gap becomes

$$\operatorname{dist}\Big(\mathbf{x}, \big\{\mathbf{u} \in \mathbb{R}^n \mid \mathbf{u}^\top \mathbf{1} \leq k\big\}\Big) = \begin{cases} 0 & \text{if } \mathbf{x}^\top \mathbf{1} \leq k \\ \mathbf{x}^\top \mathbf{1} - k & \text{else} \end{cases}, \text{and}$$

ii. γ has to be nonnegative, which can be done simply as $\gamma = [\gamma]_+$. Note that now it is more possible for $\omega''(\gamma) = 0$, since the safe range (10) may become empty after $\gamma = [\gamma]_+$, for some input vector $\mathbf{y}$.

[2]Note that the works [4, 5] do not tackle Problem (1) but they are focusing on solving the projection on to unit simplex. However, a similar approach based on their methods can be derived to solve Problem (1).

Table 1: Running time in seconds (in mean$\pm$std) of different methods on various sizes of $\mathbf{y}$.

Method	5×10^1	10^2	10^3	10^4
quadprog	0.0128 ± 0.01058	0.0034 ± 0.0095	0.1419 ± 0.0183	-
C++	**0.00003± 0.00007**	**0.00004± 0.00004**	0.0011 ± 0.0014	0.10 ± 0.129
Gurobi	0.002 ± 0.0048	0.0022 ± 0.0021	0.0041 ± 0.0022	0.02 ± 0.0032
Proposed	0.00013 ± 0.00014	0.00016 ± 0.00065	**0.00039± 0.00038**	**0.0039± 0.0037**
Method	10^5	10^6	10^7	10^8
quadprog	-	-	-	-
C++	10.3985 ± 13.703	-	-	-
Gurobi	0.2413 ± 0.00117	2.8229 ± 0.0955	32.26 ± 0.4953	-
Proposed	**0.0406± 0.0377**	**0.4723 ± 0.3629**	**4.2067 ± 2.5795**	**14.925 ± 2.9893**

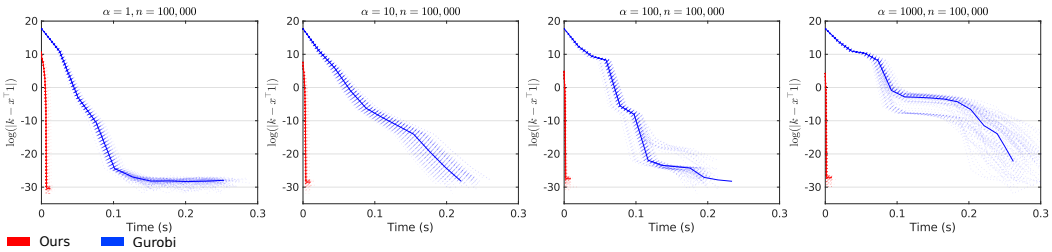

Figure 2: Comparison between the proposed projection method and the Gurobi projection for $\alpha \in \{1, 10, 100, 1000\}$. The thick curves are the median of the results of over 100 experiments on 100 datasets. The figure shows the superior performance of the proposed method in terms of run time.

3 Experiment on comparing the projection algorithms

Now we show the experimental results on two datasets to demonstrate the efficiency of the proposed projection algorithm. The proposed projection algorithm is implemented in MATLAB[3], and is compared with the MATLAB command quadprog, which is a MATLAB general-purpose solver for the quadratic program, the C++ implementation of [15], and a state-of-the-art commercial solver Gurobi [7], which uses the interior-point method [9] to solve the projection problem. All experiments were conducted on a PC with an Intel Xeon CUP (3.7GHz) and 32GB memory.

The result of Table 1: fast convergence of the proposed method across data size We compared the methods on the simulation data used in [15]. We generate vector $\mathbf{y} \in \mathbb{R}^n$ with y_i uniformly sampled from $[-0.5, 0.5]$. We pick k as an integer between 1 and n by random, and we pick the dimension $n \in \{50, 10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8\}$. We repeat the experiments 100 times, and the comparison results are shown in Table 1. We see that as n increases, the efficiency of the proposed algorithm becomes more and more significant. For large n, many methods take more than 60 seconds (denoted as '-' in Table 1). The results confirmed that the proposed projection method is the most efficient one among these methods. In the following experiments, we only compare the proposed algorithm and the commercial solver Gurobi, as they are the two most efficient methods.

The result of Fig. 2: fast convergence of the proposed method across range We compare the performance of the methods on solving the projection problem w.r.t. different range of the elements of $\mathbf{y}$. Here, we generate $\mathbf{y} \in \mathbb{R}^n$ uniformly from $[-\alpha, +\alpha]$ with $n = 10^5$ and $\alpha = \{1, 10, 100,\ 1,000\}$. As shown in Fig. 2, the proposed algorithm converges much faster than the commercial solver. The figure also shows that the performance of the proposed method is not sensitive to α.

[3]The code is in the appendix.

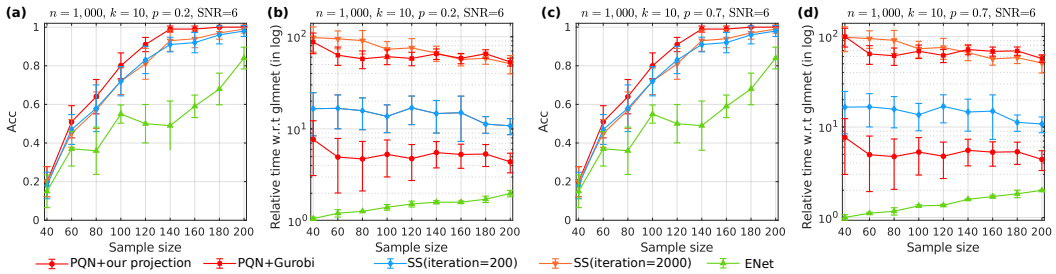

Figure 3: Comparison between different methods. The results are the average over 100 datasets, and all the error bars are in mean±std. From left to right, the sub-figures are: (a) The Acc values between different methods when $n = 10^3$, $k = 10$, $p = 0.2$, and SNR= 6; (b) The computation time of the algorithms in (a); (c) The Acc values between different methods when p changed to 0.7; and (d) The computation time of the algorithms in (c).

4 Application of the projection in Sparse Least-square Regression

An application of the proposed projection method is solving the cardinality-constrained ℓ_2^2 regularized regression [8]:

$$\min_{\|\mathbf{w}\|_0 \le k} \left\{ F(\mathbf{w}) := \sum_{i=1}^{n} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 + \frac{1}{2}\rho \|\mathbf{w}\|_2^2 \right\}, \tag{P^*}$$

where $\left\{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \mathbb{R}\right\}_{i=1}^m$ is a collection of m samples of data point with label. The sparsity-inducing constraint $\|\mathbf{w}\|_0 \le k$ enforces $\mathbf{w} \in \mathbb{R}^n$ to be at most k-sparse (with k non-zero elements). A recent work [10] studied the Boolean relaxation of P^*, which gives the following problem

$$\min_{\mathbf{u} \in \Delta_k} \left\{ G(\mathbf{u}) := \mathbf{y}^\top \left(\frac{1}{\rho} \mathbf{X} \mathbf{D}(\mathbf{u}) \mathbf{X}^\top + \mathbf{I} \right)^{-1} \mathbf{y} \right\}, \tag{P_{BR}}$$

where $\mathbf{u} \in \Delta_k$ means $\mathbf{u}$ has to be inside the k-capped simplex. Both the original work of [10] and several follow-up papers [1, 2] showed that the solution of the Boolean relaxation P_{BR} empirically outperforms the solution of other sparse estimation methods on recovering the sparse features, such as lasso [13] and elastic net [16], especially when the sample size is small and the feature dimension is huge. These empirical results encourage us to develop an efficient and scalable algorithm for solving P_{BR}, which in turn motivates the study of the projection problem (1).

Now we show extensive experimental results to demonstrate the efficiency of the proposed method (Algorithm 1) on accelerating the Projected Quasi-Newton (PQN) method [11] on solving P_{BR} on big data. To be specific, we use both a simulation dataset and a real-world dataset with various problem sizes. We compare with a state-of-the-art method, which uses a subgradient method [2] (denoted as SS) implemented in Julia to solve a min-max problem that is equivalent to P_{BR}. In addition, we compare with the classical elastic net (ENet) [16] as a reference. For a fair comparison between methods that are implemented in different programming languages, we report the computational time for each algorithm relative to the time needed to compute a lasso estimator with Glmnet [6, 12] on the same programming language, and on the same dataset. Glmnet is a package that fits the data to a generalized linear model, based on a penalized maximum likelihood estimation method. Glmnet has been implemented in various programming languages, such as Julia, R, and MATLAB.

Sparse Regression on Simulation Data First, we conduct experiments on simulation data generated as follows: let $[m] = \{1, 2, \ldots, m\}$, we draw $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}_n, \mathbf{\Sigma}),\ i \in [m]$ independently from a n-dimensional normal distribution with zero mean and co-variance matrix $\mathbf{\Sigma}$. We randomly sample a weight vector $\mathbf{w}^{\text{true}} \in \{-1, 0, +1\}$ with exactly k_{true} non-zero coefficients. Then we generate noise vector $\epsilon \in \mathbb{R}^m$, where $\epsilon_i,\ i \in [m]$ drawn independently from a normal distribution scaled according to a chosen signal-to-noise ratio (SNR) defined as $\sqrt{\text{SNR}} = \|\mathbf{X}\mathbf{w}^{\text{true}}\| / \|\epsilon\|$. Finally, we form $\mathbf{Y} = \mathbf{X}\mathbf{w}^{\text{true}} + \epsilon$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top$. We evaluate the performance on simulation data using accuracy defined as $\text{Acc}(\mathbf{w}) := \dfrac{\left|\{i : w_i \neq 0, w_i^{\text{true}} \neq 0\}\right|}{\left|\{i : w_i^{\text{true}} \neq 0\}\right|}$.

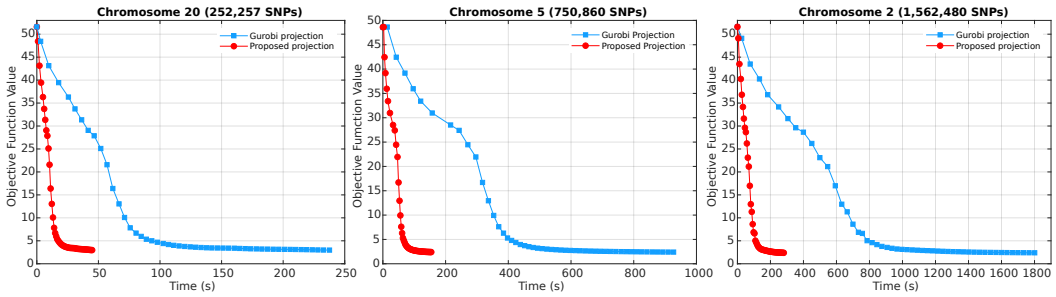

Figure 4: Convergence comparison between PQN with the proposed projection algorithm and PQN with Gurobi projection. From left to right, the sub-figures are: (a) Comparison on chromosome 20; (b)Comparison on chromosome 5; and (c) Comparison on chromosome 2.

For the first simulation data, we generate 100 datasets by setting $\mathbf{\Sigma}_{ij} = p^{|i-j|}$ to a Toeplitz covariance matrix with $p = 0.2$, $n = 1,000$, $k_{\text{true}} = 20$, and SNR$= 6$. As shown in Fig. 3 (a), PQN + the proposed projection and PQN + Gurobi have exactly the same performance in in terms of the Acc value, which is expected since the only difference between these methods is the projection algorithm. In addition, the two methods converge fast to Acc$= 1$, outperforming all the other methods. Fig. 3 (b) shows that PQN + the proposed projection is the most efficient algorithm for solving $P_{\boldsymbol{BR}}$ among all the tested methods.

The second simulation data is generated similarly to the first one. The only difference here is that $p = 0.7$, indicting the higher correlation between the features, meaning that now it is harder to recover the true support of $\mathbf{w}^{\text{true}}$. As shown in Fig. 3 (c), PQN + the proposed projection and PQN + Gurobi again have the same performance, and they still outperform all the other methods. Again, PQN + the proposed projection is the most efficient algorithm for solving the problem here.

For the results on comparing the convergence between PQN and SS, as well as the details of all the parameter settings for all the methods, are in the appendix.

Sparse Regression on GWAS Data Lastly, we conducted experiments to evaluate the efficiency of the proposed projection algorithm on a Genome-Wide Association Study (GWAS) dataset. The data was obtained from the dbGaP web site, under the dbGaP Study Accession phs000095.v3.p1[4]. This study is part of the Gene Environment Association Studies initiative with the goal to identify novel genetic factors that contribute to dental caries through large-scale genome-wide association studies of well-characterized families and individuals at multiple sites in the United States. The data we used in this experiment contains 227 Caucasians with 11,626,696 Single-nucleotide polymorphisms (SNPs) for all chromosomes. Here, the goal is to identify the important SNPs by performing a sparse regression. We solve the Boolean-relaxed sparse regression problem $P_{\boldsymbol{BR}}$ on chromosome 20, 5, and 2, which have 252,257 SNPs, 750,860 SNPs, and 1,562,480 SNPs, resp..

Fig. 4 shows the comparison result between PQN + the projection and PQN + Gurobi. We find that PQN + the proposed projection outperforms in all three problems (with different numbers of SNPs) in terms of computational time. In addition, PQN + the proposed projection and PQN + Gurobi converge to the same objective function value, indicating the correctness of the proposed projection algorithm.

5 Conclusion

We propose to use Newton's method to solve the problem of projecting a vector onto the k-capped simplex. On the theory side, we transform the problem to a scalar minimization problem and provide a theorem to characterize the cost function of such a problem. On the practical side, we show that the proposed method outperforms existing methods on solving the projection problem, and show that it can be used to accelerate the method on solving sparse regressions in bioinformatics on a huge dataset.

[4] `https://www.ncbi.nlm.nih.gov/projects/gap/cgi-bin/study.cgi?study_id=phs000095.v3.p1`

References

[1] D. Bertsimas and B. V. Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics*, 48(1):300 – 323, 2020.

[2] D. Bertsimas, J. Pauphilet, and B. V. Parys. Sparse Regression: Scalable Algorithms and Empirical Performance. *Statistical Science*, 35(4):555 – 578, 2020.

[3] J. F. Bonnans and A. Shapiro. Optimization problems with perturbations: A guided tour. *SIAM Review*, 40(2):228–264, 1998.

[4] L. Condat. Fast projection onto the simplex and the l1 ball. *Mathematical Programming*, 158(1):575–585, 2016.

[5] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279, 2008.

[6] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[7] L. Gurobi Optimization. Gurobi optimizer reference manual, 2021.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[9] Y. Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

[10] M. Pilanci, M. J. Wainwright, and L. El Ghaoui. Sparse learning via boolean relaxations. *Mathematical Programming*, 151(1):63–87, 2015.

[11] M. Schmidt, E. Berg, M. Friedlander, and K. Murphy. Optimizing costly functions with simple constraints: A limited-memory projected quasi-newton algorithm. In *AISTATS 2009*, pages 456–463. PMLR, 16–18 Apr 2009.

[12] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13, 2011.

[13] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[14] J. v. Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[15] W. Wang and C. Lu. Projection onto the capped simplex. *arXiv preprint arXiv:1503.01002*, 2015.

[16] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See § 1 and also the discussion in § 2.3
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See § 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See § 2.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplementary materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]