Semismooth Newton Algorithm for Efficient Projections onto $\ell_{1,\infty}$-norm Ball

Dejun Chu [1] **Changshui Zhang** [1] **Shiliang Sun** [2] **Qing Tao** [3]

Abstract

The structured sparsity-inducing $\ell_{1,\infty}$-norm, as a generalization of the classical ℓ_1-norm, plays an important role in jointly sparse models which select or remove simultaneously all the variables forming a group. However, its resulting problem is more difficult to solve than the conventional ℓ_1-norm constrained problem. In this paper, we propose an efficient algorithm for Euclidean projection onto $\ell_{1,\infty}$-norm ball. We tackle the projection problem via semismooth Newton algorithm to solve the system of semismooth equations. Meanwhile, exploiting the structure of the Jacobian matrix via LU decomposition yields an equivalent algorithm which is proved to terminate after a finite number of iterations. Empirical studies demonstrate that our proposed algorithm outperforms the existing state-of-the-art solver and is promising for the optimization of learning problems with the $\ell_{1,\infty}$-norm ball constraint.

1. Introduction

Sparse statistical model (Tibshirani et al., 2015) seeks only a small amount of non-zero parameters to describe data, which not only makes the prediction more interpretable, but also leads to significant computational advantages. It has emerged as a powerful tool in machine learning, signal processing and statistics. Typically sparse learning model can be cast as the following regularized loss function minimization problem

$$\min_{\boldsymbol{W}} \ \mathcal{L}(\boldsymbol{W}) + \lambda \mathcal{R}(\boldsymbol{W}) \tag{1}$$

where $\mathcal{L}(\boldsymbol{W})$ is known as the surrogate convex loss function, $\lambda > 0$ is an appropriate regularization parameter and $\mathcal{R}(\boldsymbol{W})$ represents the sparsity-inducing regularization (Bach et al., 2012). By Lagrangian duality theory (Bertsekas et al., 2003), problem (1) is equivalent under mild conditions to the following constrained optimization problem

$$\min_{\boldsymbol{W}} \ \mathcal{L}(\boldsymbol{W}), \quad s.t. \ \mathcal{R}(\boldsymbol{W}) \leq C \tag{2}$$

where C is a user-specified parameter. It is well known that the regularization parameter λ in the problem (1) is usually estimated via an external procedure such as cross-validation. In contrast, it is easier to find the parameter C in the constrained problem (2) from the fact that C has a much more intuitive meaning than λ in many situations. One straightforward solution to the problem (2) is the projected gradient algorithm (PGA). Therefore an efficient projection step will play a crucial role in the whole procedure since every iteration of PGA will call for a projection. However, in many cases developing an efficient projection algorithm comparable with the computational cost of each gradient update in the PGA is still a challenge. In this paper, we propose an efficient projection algorithm that converges in a finite number of iterations for the problem (2) when the constraint is the $\ell_{1,\infty}$-norm ball.

As a general extension of ℓ_1-norm, $\ell_{1,q}$-norms with $q \in (1, \infty]$ promote group sparsity, i.e., the solution is not only sparse as that to the classical ℓ_1-norm, but also the pre-specified group variables can be selected or removed simultaneously. It has been shown (Turlach et al., 2005; Yuan & Lin, 2006; Roth & Fischer, 2008; Huang et al., 2010) that $\ell_{1,q}$-norms using group structure can yield more reliable estimation than the standard ℓ_1-norm when the data have underlying group structure, rather than just sparsity. In practice, the settings of $q = 2$ and $q = \infty$ are the most popular choices. While $\ell_{1,2}$-norm has been studied in detail (Yuan & Lin, 2006; Roth & Fischer, 2008), which will induce the group Lasso model if combined with the least-squares loss function, we focus on its counterpart $\ell_{1,\infty}$-norm in this paper.

[1]Institute for Artificial Intelligence, Tsinghua University (THUAI), State Key Lab of Intelligent Technologies and Systems, Beijing National Research Center for Information Science and Technology (BNRist), Department of Automation, Tsinghua University, Beijing, P. R. China [2]School of Computer Science and Technology, East China Normal University, Shanghai, P. R. China [3]Army Academy of Artillery and Air Defense, Hefei, P. R. China. Correspondence to: Changshui Zhang <zcs@mail.tsinghua.edu.cn>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

The $\ell_{1,\infty}$-norm projection arises in a variety of applications, such as variable selection (Turlach et al., 2005; Masaeli et al., 2010; Xiang et al., 2013; Hernández-Lobato et al., 2015), multitask learning (Liu et al., 2009a;b; Rakotomamonjy et al., 2011), nonnegative matrix factorization (Kim et al., 2012), ranking (Rakotomamonjy, 2012), and computer vision (Jia et al., 2010; Zhang et al., 2013), etc. However, unlike the $\ell_{1,2}$-norm, the projection onto the $\ell_{1,\infty}$-norm ball is not easy to tackle and lacks a closed-form solution. The time required by the projection onto the $\ell_{1,\infty}$-norm ball is greater than the time cost of gradient update at each iteration, which has become the bottleneck of the PGA method with $\ell_{1,\infty}$-norm constraint and thus weakens the competitiveness of $\ell_{1,\infty}$-norm (Vogt & Roth, 2010).

The reliable workhorse to solve the $\ell_{1,\infty}$ projection was proposed by Quattoni et al. (2009), which maps the $\ell_{1,\infty}$ projection problem to a feasibility problem. It takes $O(dm \log dm)$ time to find the jointly sparse solution via sorting all the $dm + 1$ breakpoints associated with data matrix $\boldsymbol{A} \in \mathbb{R}^{d\times m}$. The alternative scheme is to cast the projection into the univariate root-finding procedure (Sra, 2011; 2012; Su et al., 2012). Due to the lack of exploitation of the special structure of the $\ell_{1,\infty}$-norm constraint, generally the bisection method can be invoked which finds the root with ϵ-accuracy in $O(\log(\delta/\epsilon))$ iterations where δ represents the width of the interval containing the root. Following this idea, Chau et al. (2019) just recently employed a Newton based method to find the root. While the Newton procedure can converge faster than the sorting-based method, it is somehow disappointing to observe in the experiments of Chau et al. (2019) that the acceleration effect is gradually declining when the data size increases. In fact, although Newton's method has fewer iterations than its counterpart, it needs to solve multiple ℓ_1-norm projection problems accurately at each iteration. Overall, it takes $O(dm)$ iterations with arithmetic complexity $O(dm^2)$ of each step such that its advantages, compared with the sorting-based method, gradually decline when m becomes large.

The main contribution of this paper is that we develop an efficient semismooth Newton method for computing the $\ell_{1,\infty}$-norm ball projection after at most $O(dm)$ iterations. More specifically, we reformulate the projection problem to semismooth system and employ Newton's method to find the solution. Furthermore, we exploit the structure of the Jacobian matrix via LU decomposition and obtain an equivalent algorithm whose time complexity can be bounded theoretically in finite steps. Meanwhile, an important appeal of the proposed algorithm is its linear time iteration complexity $O(dm)$, while it also enjoys the comparable iteration numbers to the recent Newton's method (Chau et al., 2019). The experimental results show that our method outperforms the state-of-the-art solver.

The paper is organized as follows. We first introduce the notation and cast the $\ell_{1,\infty}$-norm ball projection as a solving problem of semismooth equations in Section 2. We then gain from a proper use of the special structure of the Jacobian matrix and propose the semismooth Newton method in Section 3. The characteristics of Newton iterations are formally analysed and as a by-product an equivalent algorithm is guaranteed to converge in a finite number of iterations in Section 4. The experimental results are reported in Section 5, after which the conclusion is given in Section 6.

2. Notation and Problem Formulation

Vectors are denoted by bold lower case letters and matrices by upper case ones. We use x_i to denote the i-th component of the vector $\boldsymbol{x} \in \mathbb{R}^d$ and $x_{[i]}$ as the i-th smallest component of $\boldsymbol{x}$, that is, $x_{[1]} \leq x_{[2]} \leq \ldots \leq x_{[d]}$.

Assume the parameter matrix $\boldsymbol{W} \in \mathbb{R}^{d\times m}$ with rows $\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_d^\top$, where $\boldsymbol{w}_i \in \mathbb{R}^m$ for $i = 1, 2, \ldots, d$ is the i-th row of $\boldsymbol{W}$. We consider the Frobenius norm of matrix $\boldsymbol{W}$ as $\|\boldsymbol{W}\|_F = (\sum_{i=1}^d \sum_{j=1}^m W_{i,j}^2)^{1/2}$ and define the $\ell_{1,\infty}$-norm of $\boldsymbol{W}$ as $\|\boldsymbol{W}\|_{1,\infty} = \sum_{i=1}^d \max_j |W_{i,j}|$. Thus the Euclidean projection with respect to $\boldsymbol{A}$ onto the $\ell_{1,\infty}$-norm ball can be formulated as

$$\min_{\boldsymbol{W}} \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{A}\|_F^2, \quad s.t. \sum_{i=1}^d \max_j |W_{i,j}| \leq C \tag{3}$$

where C is a bound on $\|\boldsymbol{W}\|_{1,\infty}$.

It is easy to show that each non-zero component of the optimal solution $\boldsymbol{W}$ of problem (3) shares the sign of its counterpart in $\boldsymbol{A}$ (Duchi et al., 2008). Thus, we may assume without loss of generality that $A_{i,j} \geq 0$ for all i, j. Furthermore, it can also be assumed that the constraint in (3) is active, i.e. $\|\boldsymbol{A}\|_{1,\infty} \geq C$. Otherwise, there is a trivial optimal $\boldsymbol{W} = \boldsymbol{A}$. Based on these two assumptions, the above problem can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{\mu}} \quad & \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{A}\|_F^2 \\ s.t. \quad & 0 \leq W_{i,j} \leq \mu_i, \ \forall i, j \\ & \sum_{i=1}^d \mu_i = C. \end{aligned} \tag{4}$$

Introducing Lagrange multiplier $\tilde{\theta}$ for the equality constraint $\sum_{i=1}^d \mu_i = C$, while keeping the box constraint $0 \leq W_{i,j} \leq \mu_i$ intact, we define the Lagrange dual function associated with (4) as

$$\begin{aligned} &\min_{\boldsymbol{W},\boldsymbol{\mu}} \tilde{\mathcal{L}}(\boldsymbol{W}, \boldsymbol{\mu}; \tilde{\theta}) = \frac{1}{2}\|\boldsymbol{W} - \boldsymbol{A}\|_F^2 + \tilde{\theta}\big(\sum_{i=1}^d \mu_i - C\big) \\ &s.t.\ 0 \leq W_{i,j} \leq \mu_i, \ \forall i, j. \end{aligned} \tag{5}$$

Combining the KKT conditions with box constraints on the variable $\boldsymbol{W}$, we obtain the primal optimal

$$W_{i,j} = \min\left(\max\left(A_{i,j}, 0\right), \mu_i\right) = \min\left(A_{i,j}, \mu_i\right). \quad (6)$$

Using this expression to substitute $W_{i,j}$ in (4), we can write the equivalent optimization problem with only $\boldsymbol{\mu}$ as

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{m}\max\left(A_{i,j}-\mu_i, 0\right)^2 \\ s.t. \quad & \sum_{i=1}^{d}\mu_i = C \\ & \mu_i \geq 0,\ i = 1, \ldots, d. \end{aligned} \quad (7)$$

Now we will focus on the problem (7), since the optimal solution $W_{i,j}$ depends on the variable μ_i from the closed-form expression (6). To form the Lagrangian similarly, we introduce multiplier θ for the equality constraint, and multipliers $\gamma_i \geq 0$ for the d inequality constraints, and obtain

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mu}; \lambda) = & \frac{1}{2}\sum_{i=1}^{d}\sum_{j=1}^{m}\max\left(A_{i,j}-\mu_i, 0\right)^2 \\ & + \theta\left(\sum_{i=1}^{d}\mu_i - C\right) - \sum_{i=1}^{d}\gamma_i\mu_i. \end{aligned} \quad (8)$$

Applying the zero gradient condition with respect to the variable μ_i, we obtain

$$\gamma_i + \sum_{j}\max\left(A_{i,j}-\mu_i, 0\right) = \theta,$$

where we can conclude that $\theta \geq 0$ as the two terms on the left side of the equation are both nonnegative. Furthermore, we distinguish two cases with respect to μ_i separately. We first consider the case $\mu_i > 0$. By the complementary relaxation condition, the optimal Lagrange multiplier γ_i must be zero. Therefore,

$$\sum_{j=1}^{m}\max\left(A_{i,j}-\mu_i, 0\right) = \theta. \quad (9)$$

Alternatively, we obtain $\theta \geq \sum_{j=1}^{m} A_{i,j}$ in which case $\mu_i = 0$.

We note that given a constant $\theta > 0$, the equation (9) is the transformation from the classical ℓ_1-norm ball projection problem (Liu & Ye, 2009; Gong et al., 2011; Condat, 2014). However, $\ell_{1,\infty}$-norm ball projection problem is more difficult than the projection onto the ℓ_1-norm ball. In the latter problem, not only θ is a user-specified constant but the equation (9) is just a univariate root-finding problem. In contrast, a major difficulty that arises in solving the $\ell_{1,\infty}$ projection problem stems from the fact that θ is an unknown Lagrange multiplier. More importantly, there is also a d-dimension variable $\boldsymbol{\mu}$ that should satisfy the equality constraint in (7).

One straightforward approach is to build a univariate equation like the ℓ_1-norm ball projection problem. We consider the i-th row of matrix $\boldsymbol{A}$. It can be shown from the analysis that every $\mu_i \in [0, \max_j A_{i,j}]$ should satisfy the equation (9) in which the multiplier $\theta \in [0, \sum_{j=1}^{m} A_{i,j}]$. In this case we can derive the following closed-form expression for μ_i which is denoted by $\tilde{\mu}_i(\theta)$:

$$\tilde{\mu}_i(\theta) = \begin{cases} \frac{\sum_{j\in\mathcal{I}(\mu_i)} A_{i,j} - \theta}{|\mathcal{I}(\mu_i)|}, & \text{if } 0 \leq \theta < \sum_{j=1}^{m} A_{i,j} \\ 0, & \text{if } \theta \geq \sum_{j=1}^{m} A_{i,j} \end{cases} \quad (10)$$

where the index set

$$\mathcal{I}(\mu_i) = \{j \mid A_{i,j} \geq \mu_i, j = 1, \ldots, m\}$$

and $|\mathcal{I}(\mu_i)|$ denotes the cardinality of the set $\mathcal{I}(\mu_i)$.

We substitute $\tilde{\mu}_i(\theta)$ into the equality constraint of (7), and then obtain a univariate equation for θ,

$$s(\theta) = 0 \quad (11)$$

where

$$s(\theta) = \sum_{i=1}^{d}\tilde{\mu}_i(\theta) - C. \quad (12)$$

Evaluating the derivative of $s(\theta)$ with respect to θ, we get

$$s'(\theta) = -\sum_{i=1}^{d}\frac{1}{|\mathcal{I}(\mu_i)|}. \quad (13)$$

Proposition 1 *Let $s(\theta)$ be defined by (12). Then $s(\theta)$ is convex, strictly monotonically decreasing with $dm+1$ breakpoints at most, and the equation (11) has unique root on the interval $[0, \max_i \sum_{j=1}^{m} A_{i,j}]$.*

The proof is given in the supplementary material. Figure 1 illustrates a toy example which builds upon a 2×3 matrix $\boldsymbol{A}$. Hence, there are 7 breakpoints in the function $s(\theta)$.

Based on the root-finding idea, we can employ not only the primary bisection method (Sra, 2011; 2012; Su et al., 2012) but more sophisticated algorithms to solve the equation (12). For instance, we can sort all the $d(m+1)$ breakpoints in all the nonsmooth functions $\tilde{\mu}_i(\theta)$ for $i = 1, \ldots, d$ (Quattoni et al., 2009) or apply linear-time median-finding algorithm (Duchi et al., 2008) to search the root for the univariate equation about θ. Newton's method (Cominetti et al., 2014; Davis et al., 2016) can also be used to tackle the equation (11) which is the main contribution of the recent work (Chau

et al., 2019). It is well known that Newton's method enjoys fast convergence rate. However, we should note that each Newton step relies on the accurate solution μ_i to (9). In other words, we should solve d ℓ_1-norm ball projection problems at each iteration which will severely hinder the total performance of Newton's method.

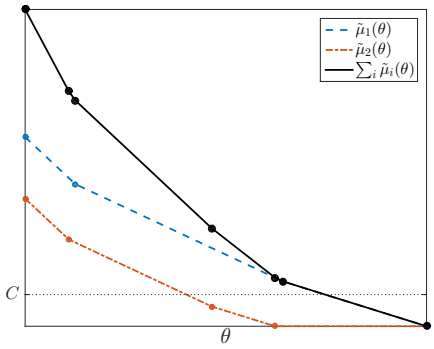

Figure 1. Geometric interpretation of the functions $\tilde{\mu}_1(\theta)$, $\tilde{\mu}_2(\theta)$ and their summation. The horizontal dashed line shows the bound C and all the breakpoints are indicated as solid circles.

3. Semismooth Newton Method for $\ell_{1,\infty}$-norm Ball Euclidean Projection

Before proposing our semismooth Newton method, we give an overview of semismoothness. Semismoothness was first introduced for functionals by Mifflin (1977). Qi & Sun (1993) extended the definition to vector-value functions and developed the semismooth Newton method to solve the nonsmooth equations. We note that the notion of semismoothness is based on Clarke's generalized Jacobian (Clarke, 1990). Important examples of semismooth functions include convex functions, smooth functions and piecewise linear functions (Mifflin, 1977). For more details we refer to the book (Izmailov & Solodov, 2014) and the survey (Qi & Sun, 1999).

3.1. Semismooth Equations

Suppose the variable $\boldsymbol{x} = (\mu_1, \mu_2, \ldots, \mu_d, \theta)^\top$. Combining the equations (9) and the equality constraint in (7), we consider the system

$$F(\boldsymbol{x}) = \mathbf{0}, \tag{14}$$

where function $F : \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ is defined by

$$F(\boldsymbol{x}) = \begin{pmatrix} \sum_j \max\{A_{1,j} - \mu_1, 0\} - \theta \\ \vdots \\ \sum_j \max\{A_{d,j} - \mu_d, 0\} - \theta \\ -\mu_1 - \mu_2 - \ldots - \mu_d + C \end{pmatrix}.$$

Before attempting to find the solution of (14), it is important to verify that each component $F_i(\boldsymbol{x})$ of $F(\boldsymbol{x})$ is convex. Since the convex function is a special case of semismooth function (Mifflin, 1977), we can conclude $F(\boldsymbol{x})$ is semismooth without elaborating on these mathematical concepts. The semismoothness is the hallmark of the applicability of semismooth Newton algorithm.

3.2. Semismooth Newton Method

To solve the semismooth equation (14), we design a semismooth Newton algorithm by exploiting the underlying second-order structure information. The key of our method is to find the Newton direction. Specifically, for the current iteration of $\boldsymbol{x}^{(t)}$ we solve the following equation

$$J(\boldsymbol{x}^{(t)})\boldsymbol{v} = -F(\boldsymbol{x}^{(t)}),$$

where $J(\boldsymbol{x}^{(t)})$ is an arbitrary element of the generalized Jacobian of $F(\boldsymbol{x}^{(t)})$. Generally speaking, it is not always straightforward to compute the element of the generalized Jacobians. Fortunately, in our case the i-th row of $J(\boldsymbol{x}^{(t)})$ is just a subgradient of $F_i(\boldsymbol{x})$ at $\boldsymbol{x}^{(t)}$ because each component $F_i(\boldsymbol{x})$ is convex (Clarke, 1990).

Assuming that the Jacobian $J(\boldsymbol{x}^{(t)})$ is nonsingular, the semismooth Newton method iterates as

$$\boldsymbol{v}^{(t)} = -J^{-1}(\boldsymbol{x}^{(t)})F(\boldsymbol{x}^{(t)}), \tag{15}$$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{v}^{(t)}. \tag{16}$$

Computing a subgradient of each row of F with respect to μ_i and θ, we obtain the generalized Jacobian

$$J(\boldsymbol{x}) = \begin{pmatrix} -|\mathcal{I}(\mu_1)| & 0 & \ldots & 0 & -1 \\ 0 & -|\mathcal{I}(\mu_2)| & \ddots & \vdots & -1 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & \ldots & 0 & -|\mathcal{I}(\mu_d)| & -1 \\ -1 & -1 & \ldots & -1 & 0 \end{pmatrix}.$$

The main disadvantage of Newton's method is the storage cost of the Jacobian and the computational cost of Newton step, which for large scale problems can be prohibitively expensive. Fortunately, instead of calculating the inverse of generalized Jacobian J directly, we can exploit the symmetric structure of sparse J to substantially reduce the cost. The following lemma establishes the nonsingularity of the generalized Jacobian J and the unique LU factorization of J.

Lemma 1 *Suppose $\mu_i \in [0, \max_j A_{i,j}]$ for $i = 1, 2, \ldots, d$. Then the generalized Jacobian J is nonsingular. Furthermore, the LU factorization of J exits and is unique.*

Proof: It is easy to verify that there exist a lower triangular matrix L and an upper triangular matrix U such that

$$J = LU,$$

where

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ \frac{1}{|\mathcal{I}(\mu_1)|} & \frac{1}{|\mathcal{I}(\mu_2)|} & \cdots & \frac{1}{|\mathcal{I}(\mu_d)|} & 1 \end{pmatrix}$$

and

$$U = \begin{pmatrix} -|\mathcal{I}(\mu_1)| & \cdots & 0 & -1 \\ 0 & \ddots & 0 & \vdots \\ \vdots & \cdots & -|\mathcal{I}(\mu_d)| & -1 \\ 0 & \cdots & 0 & \sum_{i=1}^{d} \frac{1}{|\mathcal{I}(\mu_i)|} \end{pmatrix}.$$

On the other hand, since $\mu_i \in [0, \max_j A_{i,j}]$, then $|\mathcal{I}(\mu_i)| \geq 1$ and the determinant of matrix J

$$\det(J) = (-1)^d \sum_{i=1}^{d} \frac{1}{|\mathcal{I}(\mu_i)|} \prod_{i=1}^{d} |\mathcal{I}(\mu_i)| \neq 0.$$

Thus, the Jacobian J is nonsingular which concludes that the LU factorization is unique. ■

From the proof of Lemma 1, one can see that the semismooth Newton iteration (15)-(16) is well-defined. Moreover, once the Jacobian J has been factored into the product of L and U, we can take advantage of these two sparse triangular matrices to find the Newton step $\boldsymbol{v}$ via two steps, i.e.,

$$L\boldsymbol{z} = -F,\ U\boldsymbol{v} = \boldsymbol{z}, \tag{17}$$

where $\boldsymbol{z} \in \mathbb{R}^{d+1}$ is a temporary variable. First we solve the lower triangular system by forward substitution to obtain

$$z_i = -F_i,\ i = 1, \ldots, d \tag{18}$$

$$z_{d+1} = -F_{d+1} - \sum_{i=1}^{d} \frac{z_i}{|\mathcal{I}(\mu_i)|}. \tag{19}$$

Similarly, solve the upper triangular system by back substitution to obtain the Newton step $\boldsymbol{v}$ as

$$v_{d+1} = \frac{z_{d+1}}{\sum_{i=1}^{d} 1/|\mathcal{I}(\mu_i)|} \tag{20}$$

$$v_i = -\frac{z_i + v_{d+1}}{|\mathcal{I}(\mu_i)|},\ i = 1, \ldots, d. \tag{21}$$

To compute the Newton step takes the linear time complexity $O(dm)$, which is much more efficient than directly calculating the inverse matrix in (15). We describe the proposed semismooth Newton procedure in Algorithm 1.

It is worth noting that if $\theta \geq \sum_j A_{i,j}$, we conclude $\mu_i = 0$ which means that μ_i contributes nothing to $s(\theta)$. Thus the i-th equation in (14) should be deleted and the row index set $\mathcal{G}$ is reduced. The Jacobian $J(\boldsymbol{x})$ will also be reduced accordingly. In a practical implementation, the stopping criterion can be checked after the Newton step is computed.

Algorithm 1 Semismooth Newton Algorithm for $\ell_{1,\infty}$-norm Ball Projection

Input: $\boldsymbol{A} \in \mathbb{R}^{d \times m}, C > 0, \epsilon > 0, t = 0$.
if $\|\boldsymbol{A}\|_{1,\infty} \leq C$ **then**
  $\boldsymbol{W} = \boldsymbol{A}$ and return.
end if
Initialize $\theta^{(0)} \in [0, \max_i \sum_{j=1}^{m} A_{i,j}]$ and compute $\mu_i^{(0)}$ to satisfy the equation (9).
repeat
  Compute $\mathcal{G} = \{i \mid \theta^{(t)} \leq \sum_{j=1}^{m} A_{i,j}\}$.
  Update $\mathcal{I}(\mu_i^{(t)}) = \left\{j \mid A_{i,j} \geq \mu_i^{(t)}, i \in \mathcal{G}\right\}$.
  Compute $F(\boldsymbol{x}^{(t)})$ via (14).
  Compute $\boldsymbol{v}^{(t)}$ via (18)-(21).
  Update $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{v}^{(t)}$.
  $t = t + 1$.
until stopping criterion is satisfied, i.e., $\|\boldsymbol{v}^{(t)}\| \leq \epsilon$.
$\boldsymbol{\mu} = \max(\boldsymbol{\mu}, 0)$.
$W_{i,j} = \min(A_{i,j}, \mu_i)$ for all i, j.
Output: $\boldsymbol{W}$

Although the semismooth Newton algorithm enjoys fast convergence speed, if the initial estimate is too far from the root of the equation (14), the generated sequence may converge slowly or even diverge. In general, choosing a proper initial guess relies on a shrewd insight into the underlying problem. From the equation (14) and Figure (1), while we can randomly initialize $\theta^{(0)} \in [0, \max_i \sum_j A_{i,j}]$, $\mu_i^{(0)}$ should satisfy the equation (9). A simple and effective strategy is setting $\mu_i^{(0)} = \max_j A_{i,j}$ and $\theta^{(0)} = 0$.

3.3. Convergence Analysis

Using semismoothness and nonsingularity conditions, generalized Newton's methods have been proved locally and superlinearly convergent for solving systems of nonsmooth equations (Qi & Sun, 1993). As $F(\boldsymbol{x})$ in the system (14) is a semismooth function, our algorithm enjoys the following superlinear convergence rate.

Theorem 1 *Suppose $\boldsymbol{x}^*$ is the solution of (14), $F : \mathbb{R}^{d+1} \to \mathbb{R}^{d+1}$ is locally Lipschitz continuous and semismooth at $\boldsymbol{x}^*$, and the Jacobian $J(\boldsymbol{x})$ is nonsingular. Let $\{\boldsymbol{x}^t\}$ be the sequence generated by Algorithm 1. Then the semismooth Newton iteration (15)-(16) is well-defined and $\{\boldsymbol{x}^t\}$ converges to the solution $\boldsymbol{x}^*$ Q-superlinearly.*

The proof is similar to that of Qi & Sun (1993, Th. 3.2) for semismooth functions and is omitted here. In the next sec-

tion, we will show that the equivalent variant of Algorithm 1 terminates after a finite number of iterations.

4. The Equivalent Variant of Algorithm 1

In order to better understand why our method is superior to the existing Newton based method (Chau et al., 2019), we need to further delve into Algorithm 1.

4.1. Equivalent Semismooth Newton Iteration

Proposition 2 *Assume* $|\mathcal{I}(\mu_i^{(t)})| \geq 1$ *for* $i = 1, 2, \ldots, d$. *Then* $v_{d+1}^{(t)}$ *is the Newton step for* $s(\theta)$ *at* $\theta^{(t)}$.

The proof is given in the supplementary material. According to the Proposition 2, we can see that the iteration of variable θ follows the Newton iteration

$$\theta^{(t+1)} = \theta^{(t)} + v_{d+1}^{(t)} = \theta^{(t)} - \frac{s(\theta^{(t)})}{s'(\theta^{(t)})}. \tag{22}$$

Now let us consider the update of variable $\boldsymbol{\mu}$ in (16) via the Newton step (21). It follows that

$$\begin{aligned}
\mu_i^{(t+1)} &= \mu_i^{(t)} + v_i^{(t)},\ i = 1, \ldots, d \\
&= \mu_i^{(t)} + \frac{F_i^{(t)}}{|\mathcal{I}(\mu_i^{(t)})|} + \frac{s(\theta^{(t)})}{|\mathcal{I}(\mu_i^{(t)})| s'(\theta^{(t)})} \\
&= \left(\sum_{j \in \mathcal{I}(\mu_i^{(t)})} A_{i,j} - \left(\theta^{(t)} - \frac{s(\theta^{(t)})}{s'(\theta^{(t)})} \right) \right) \Big/ |\mathcal{I}(\mu_i^{(t)})| \\
&= \frac{\sum_{j \in \mathcal{I}(\mu_i^{(t)})} A_{i,j} - \theta^{(t+1)}}{|\mathcal{I}(\mu_i^{(t)})|}.
\end{aligned} \tag{23}$$

In fact from the derivation given above, one can see that $\mu_i^{(t)}$ is updated after the Newton iteration of $\theta^{(t)}$, which is presented as Algorithm 2. The whole procedure can give important insights into the Newton iteration (16) of Algorithm 1. At step t, given $\theta^{(t)}$ we first compute $\tilde{\mu}_i(\theta^{(t)})$ via (10). It follows from (22) that $\theta^{(t)}$ is updated by the Newton step.

Significantly different from the previous method (Chau et al., 2019), Algorithm 2 updates the variable μ_i via (23) with low cost at each step. More precisely, the former needs to use an iterative algorithm to calculate each μ_i accurately to satisfy the equation (9) with overall arithmetic complexity $O(m^2)$, while the latter update of μ_i is essentially a one-step computation with cost $O(m)$ in the ℓ_1-norm ball projection problem (9). There is no doubt that the iterative algorithm for the accurate μ_i of (9) is less efficient than our update strategy. This is the key reason why the acceleration effect of Newton's method declines when the data size increases in Chau et al. (2019). However, an important question follows: why does our simple update work? We will conduct an extensive proof in the next section.

Algorithm 2 The Equivalent Algorithm for $\ell_{1,\infty}$-norm Ball Euclidean Projection

Input: $\boldsymbol{A} \in \mathbb{R}^{d \times m}, C > 0, \epsilon > 0, t = 0$.
if $\|\boldsymbol{A}\|_{1,\infty} \leq C$ **then**
  $\boldsymbol{W} = \boldsymbol{A}$ and return.
end if
Initialize $\mu_i^{(0)}$ and $\theta^{(0)}$, which satisfy $\sum_{j=1}^m \max(A_{i,j} - \mu_i^{(0)}, 0) \geq \theta^{(0)}$ and $s(\theta^{(0)}) \geq 0$.
repeat
  Compute $\mathcal{G} = \{i \mid \theta^{(t)} \leq \sum_{j=1}^m A_{i,j}\}$.
  Update $\mathcal{I}(\mu_i^{(t)}) = \left\{ j \mid A_{i,j} \geq \mu_i^{(t)}, i \in \mathcal{G} \right\}$.
  Compute $\tilde{\mu}_i(\theta^{(t)}) = \frac{\sum_{j \in \mathcal{I}(\mu_i^{(t)})} A_{i,j} - \theta^{(t)}}{|\mathcal{I}(\mu_i^{(t)})|}$.
  Update $\theta^{(t+1)} = \theta^{(t)} - \frac{s(\theta^{(t)})}{s'(\theta^{(t)})}$.
  Update $\mu_i^{(t+1)} = \frac{\sum_{j \in \mathcal{I}(\mu_i^{(t)})} A_{i,j} - \theta^{(t+1)}}{|\mathcal{I}(\mu_i^{(t)})|},\ i \in \mathcal{G}$.
  $t = t + 1$.
until stopping criterion is satisfied, i.e., $s(\theta^{(t)}) \leq \epsilon$.
$\boldsymbol{\mu} = \max(\boldsymbol{\mu}, 0)$.
$W_{i,j} = \min(A_{i,j}, \mu_i)$ for all i, j.
Output: $\boldsymbol{W}$

4.2. Convergence Theorem in Finite Steps

Algorithm 2 tackles the $\ell_{1,\infty}$-norm ball projection as a univariate equation problem $s(\theta) = 0$. Hence we first pay attention to the update of $\theta^{(t)}$ and the following proposition holds.

Proposition 3 *Suppose* $\theta^{(t)}$ *lies between two breakpoints, i.e.,* $\theta^{(t)} \in (\Theta_{[j-1]}, \Theta_{[j]}]$. *Assume* $s(\Theta_{[j]}) > 0$. *There holds*

$$\theta^{(t)} \leq \Theta_{[j]} < \theta^{(t+1)}.$$

All subsequent proofs except for Theorem 2 will be given in the supplementary material.

Proposition 4 *Assume* $\mu_i^{(t)}$ *is updated via (23) for* $t \geq 0$. *Then we have*

$$\sum_{i=1}^d \mu_i^{(t+1)} - C = 0.$$

Lemma 2 *Assume that* $\mu_i^{(t)} \in [0, \max_j A_{i,j}], \theta^{(t)} \geq 0$ *and the following two inequalities hold: (i)* $\sum_{j=1}^m \max\left(A_{i,j} - \mu_i^{(t)}, 0\right) \geq \theta^{(t)}$, *(ii)* $s(\theta^{(t)}) \geq 0$,

then it can be obtained that

$$\sum_{j=1}^{m} \max\left(A_{i,j} - \mu_i^{(t+1)}, 0\right) \geq \theta^{(t+1)}.$$

According to the above Lemma 2, we can obtain the following corollary.

Corollary 1 *Assume that* $\sum_{j=1}^{m} \max\left(A_{i,j} - \mu_i^{(t)}, 0\right) \geq \theta^{(t)}$. *Then we can obtain*

$$\tilde{\mu}_i(\theta^{(t)}) \geq \mu_i^{(t)}.$$

Building upon the above analysis of the equivalent Algorithm 2, we give the following convergence result.

Theorem 2 *Algorithm 2 converges to the root of equation* $s(\theta) = 0$ *in* $O(dm)$ *steps at most.*

Proof: We denote the breakpoint set by $\boldsymbol{\Theta} = \{\Theta_1, \Theta_2, \ldots, \Theta_{dm+1}\}$. Similarly, $\Theta_{[j]}$ represents the j-th small element of the set $\boldsymbol{\Theta}$.

From the assumption, we have the initialization $\theta^{(0)} \in (\Theta_{[j-1]}, \Theta_{[j]}]$ and $s(\theta^{(0)}) > 0$. If $s(\Theta_{[j]}) \leq 0$, we will find the root immediately. Otherwise, with Proposition 3 we have

$$\theta^{(0)} \leq \Theta_{[j]} < \theta^{(1)}.$$

If we can prove $s(\theta^{(1)}) > 0$, it means that each iteration will jump over one breakpoint at least. Thus, Algorithm 2 will terminate in $O(dm)$ steps due to the fact that only $dm + 1$ breakpoints exist at most. Indeed,

$$s(\theta^{(1)}) = \sum_i \tilde{\mu}_i(\theta^{(1)}) - C = \sum_i \tilde{\mu}_i(\theta^{(1)}) - \mu_i^{(1)} \geq 0.$$

The second equation comes from Proposition 4 and the last inequality comes from Corollary 1. ■

In the iterative process of the algorithm, it is necessary to ensure $|\mathcal{I}(\mu_i)| \geq 1$ so that the problem is well-defined. Indeed, it should be pointed out that one conclusion can be implicitly obtained in the above proof that $\theta^{(t+1)} \geq \theta^{(t)}$. According to the update formula (23), it can be shown that

$$\begin{aligned}\mu_i^{(t+1)} &= \frac{\sum_{j \in \mathcal{I}_i(\mu_i^{(t)})} A_{i,j} - \theta^{(t+1)}}{|\mathcal{I}(\mu_i^{(t)})|} \\ &\leq \frac{|\mathcal{I}(\mu_i^{(t)})| \max_j A_{i,j} - \theta^{(t+1)}}{|\mathcal{I}(\mu_i^{(t)})|} \leq \max_j A_{i,j}.\end{aligned}$$

Therefore, $|\mathcal{I}(\mu_i^{(t)})| \geq 1$ is always true during the iteration.

5. Experiments

In this section, we evaluate the proposed semismooth Newton algorithm (SNA) for computing projections onto the $\ell_{1,\infty}$-norm ball. In particular, we first make the running time comparison on synthetic data against the benchmark and the state-of-the-art methods: the sorting-based method (SBM)[1] (Quattoni et al., 2009) and recent Newton-based root-finding method (NRFM)[2] (Chau et al., 2019). Then we show a notable application to the multitask Lasso problem with $\ell_{1,\infty}$-norm constraint (MTLC) on real-world data. Equipped with our proposed projection algorithm, the projected gradient descent method can solve the MTLC problem more efficiently. For the fairness of the experimental comparison, all the algorithms are implemented in C language with a MATLAB interface and run on 3.1-GHz Intel Core i5 MacBook Pro with 16GB memory. The source codes of SBM and NRFM are available on the authors' websites, and our implementation will be released publicly on the github[3] for reproducing the results.

5.1. Projection onto the $\ell_{1,\infty}$-norm Ball

We first study the relationship between the running time and the bound parameter C in the problem (3). We randomly generate a data matrix $\boldsymbol{A} \in \mathbb{R}^{10,000 \times 10,000}$ from the normal distribution $\mathcal{N}(\mathbf{0}, \mathrm{diag}(\mathbf{1}))$, which means that there are 100 millions of non-zeros elements in matrix $\boldsymbol{A}$. The optimal $\boldsymbol{W}^*$ is computed via three different projection algorithms, i.e., SBM, NRFM and our SNA. The bound parameter C varies from $0.01\|\boldsymbol{A}\|_{1,\infty}$ to $0.5\|\boldsymbol{A}\|_{1,\infty}$ as listed in the left column of Table 1. We choose the tolerance $\epsilon = 10^{-10}$ in SNA. Table 1 presents the average errors and running times over 10 randomly generated tests. The errors are measured by the constraint violation, i.e., $|C - \|\boldsymbol{W}^*\|_{1,\infty}|$ for an optimal point $\boldsymbol{W}^*$. The errors in Table 1 reveal that SNA performs comparable to the state-of-the-art NRFM, and both represent several orders of magnitude better than SBM. The speedup with respect to SBM is shown for NRFM and SNA respectively. It is demonstrated that the proposed SNA always outperforms its counterparts in all scenarios without sacrificing the accuracy.

We observe that the time cost does not vary very much with respect to C for each method from Table 1. Thus, in our next two comparisons we set $C = 0.01\|\boldsymbol{A}\|_{1,\infty}$ to investigate the scalability of the projection algorithms. We first randomly generate a data matrix $\boldsymbol{A} \in \mathbb{R}^{d \times m}$ while setting $m = 10^3$ and varying $d \in \{10^3, 5 \times 10^3, 10^4, 5 \times 10^4, 10^5\}$. Similarly, we set $d = 10^3$ and varying $m \in$

[1] https://www.lsi.upc.edu/~aquattoni/CodeToShare/L1InfProjection.tar.gz

[2] https://sites.google.com/a/pucp.edu.pe/grchau/projects/l1inf

[3] https://github.com/djchu/projontol1inf

Table 1. Running times (s) and accuracy on a $10,000 \times 10,000$ matrix $\boldsymbol{A}$.

$\frac{C}{\|\boldsymbol{A}\|_{1,\infty}}$	SBM (QUATTONI ET AL., 2009)		NRFM (CHAU ET AL., 2019)			PROPOSED SNA		
	ERROR	TIME	ERROR	TIME	SPEEDUP	ERROR	TIME	SPEEDUP
0.01	1.058×10^{-5}	35.434	2.069×10^{-11}	10.337	3.428	1.478×10^{-12}	2.851	12.431
0.05	8.550×10^{-6}	35.060	2.069×10^{-11}	10.718	3.302	3.183×10^{-12}	3.536	9.915
0.1	6.406×10^{-6}	34.850	4.547×10^{-12}	10.999	3.169	1.182×10^{-11}	4.833	7.211
0.2	3.325×10^{-6}	33.876	1.273×10^{-11}	11.783	2.875	5.457×10^{-11}	5.489	6.171
0.3	1.543×10^{-6}	33.363	5.639×10^{-11}	12.310	2.710	3.820×10^{-11}	4.602	7.250
0.4	6.328×10^{-7}	32.950	1.819×10^{-11}	12.987	2.537	9.459×10^{-11}	3.751	8.784
0.5	2.267×10^{-7}	32.617	2.547×10^{-11}	13.705	2.380	6.548×10^{-11}	3.251	10.032

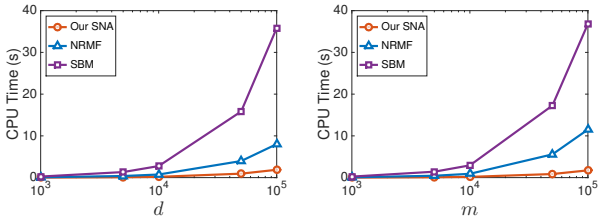

Figure 2. Time versus data dimensions and tasks. The left figure displays the result for varying dimensions and fixed tasks $m = 1000$. The right figure displays the result for varying tasks and fixed dimensions $d = 1000$.

$\{10^3, 5\times10^3, 10^4, 5\times10^4, 10^5\}$ in the second test problem. In both comparisons, the data entries are also sampled from the normal distribution $\mathcal{N}(\mathbf{0}, \text{diag}(\mathbf{1}))$.

The computational results are demonstrated in Figure 2. We can observe from the left figure that with a fixed number of m, the time cost of the three methods increases with the number of dimensions d. Moreover, both recent NRFM and our SNA run faster than the SBM, of which SNA performs best. The situation is slightly different in the right figure with a fixed number of data dimensions. Although the time overhead of all the competitors increases as the number of m increases, the efficiency advantage of NRFM is gradually decreasing compared to SBM, which is consistent with our previous theoretical analysis. In contrast, the proposed method performs quite stably and runs fastest.

5.2. Application to MTLC Model on Real-world Dataset

The multitask Lasso problem aims to fit multiple regression tasks jointly and encourage group-wise sparsity across related tasks via a sparsity promoting mixed-norm, such as the $\ell_{1,\infty}$-norm (Liu et al., 2009a). In order to study the potential of our proposed algorithm in machine learning, we apply the PGA method to solve the multitask Lasso problem with $\ell_{1,\infty}$-norm constraint, which can be cast as the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{W}} \quad & \mathcal{L}(\boldsymbol{W}) = \frac{1}{2}\sum_{i=1}^{n} \|\boldsymbol{y}_i - \boldsymbol{X}_i \tilde{\boldsymbol{w}}_i\|_2^2 \\ s.t. \quad & \|\boldsymbol{W}\|_{1,\infty} \le C \end{aligned} \tag{24}$$

where $\boldsymbol{X}_i \in \mathbb{R}^{n_i \times d}$ denotes the data matrix for the i-th task, $\boldsymbol{y}_i \in \mathbb{R}^{n_i \times 1}$ contains the associating responses, and $\tilde{\boldsymbol{w}}_i$ is the i-th column of parameter matrix $\boldsymbol{W}$.

PGA method solves the constrained optimization problem (24) via the iteration

$$\boldsymbol{W}_{k+1} = \mathcal{P}_{\ell_{1,\infty}}\left(\boldsymbol{W}_k - \lambda_k \nabla \mathcal{L}(\boldsymbol{W}_k)\right),$$

where λ_k is the stepsize, and the operator $\mathcal{P}_{\ell_{1,\infty}}$ performs projection onto the $\ell_{1,\infty}$-norm ball. In the experiment, we apply the BB line search scheme (Barzilai & Borwein, 1988) for the stepsize λ_k. The PGA process is terminated when the stopping criteria $\|\boldsymbol{W}_{k+1} - \boldsymbol{W}_k\|_F \le 10^{-8}$ is satisfied before the maximum iteration number (set to 1000) is reached.

Since our focus is evaluating the efficiency of the projection algorithm from an optimization perspective, we report the running time results of projection steps on a widely used real-world dataset, i.e., School[4]. The School data is from the Inner London Education Authority and it contains the test scores of 15,362 students from 139 secondary schools. Each student is a sample described by 27 features. In total, we are given 139 tasks (schools) to predict the exam scores.

Table 2 reports the total running time of projection steps with different bound parameters C which are listed in the leftmost column. Initialized by the same $\boldsymbol{W}_0 = \mathbf{0}$, both NRFM and the proposed SNA outperform the sorting-based method SBM. Compared with SBM, SNA speeds up faster than NRFM except in the only case $C = 0.01d$. Meanwhile, it can be shown that the speedup effect in Table 2 is not as good as that in Table 1. This is mainly because the dimensionality of the real-world dataset School is not high. As shown in Figure 2, the higher the dimensionality of

[4] https://ttic.uchicago.edu/~argyriou/code

Table 2. Projection times (s) on dataset School.

C/d	SBM	NRFM		OUR SNA	
	TIME	TIME	SPEEDUP	TIME	SPEEDUP
0.01	0.017	0.002	8.755	0.002	7.688
0.05	0.459	0.116	3.944	0.053	8.717
0.1	0.442	0.137	3.224	0.057	7.724
0.5	0.434	0.172	2.528	0.065	6.712
1.0	0.389	0.172	2.199	0.067	5.617

the problem is, the more obvious the acceleration effect becomes.

6. Conclusion

We proposed an efficient semismooth Newton algorithm for projection onto $\ell_{1,\infty}$-norm ball and obtained an equivalent variant by using LU decomposition to exploit the structural information of the Jacobian matrix. The theoretical analysis concludes that our algorithm not only converges after a finite number of steps, but also enjoys the linear time iteration cost. Experiments on both artificial data and real-world dataset show that our method outperforms the state-of-the-art solvers for the $\ell_{1,\infty}$-norm projection. Therefore, with the fast convergence of the semismooth Newton method, we are able to solve the challenging large scale $\ell_{1,\infty}$-norm projection problem efficiently as the classical ℓ_1-norm.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments. We also thank the authors of the comparative methods for providing the code. This research was supported by the NSFC Project (No. 61673179 and No. 61673394), the Fundamental Research Funds for the Central Universities, Anhui Provincial Natural Science Foundation (No. 1908085MF193) and Beijing Academy of Artificial Intelligence (BAAI).

References

Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

Barzilai, J. and Borwein, J. M. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988.

Bertsekas, D. P., Nedi, A., Ozdaglar, A. E., et al. *Convex Analysis and Optimization*. Athena Scientific, 2003.

Chau, G., Wohlberg, B., and Rodriguez, P. Efficient projection onto the $\ell_{\infty,1}$ mixed-norm ball using a Newton root search method. *SIAM Journal on Imaging Sciences*, 12 (1):604–623, 2019.

Clarke, F. H. *Optimization and Nonsmooth Analysis*. SIAM, 1990.

Cominetti, R., Mascarenhas, W. F., and Silva, P. J. A Newton's method for the continuous quadratic knapsack problem. *Mathematical Programming Computation*, 6(2): 151–169, 2014.

Condat, L. Fast projection onto the simplex and the l_1 ball. *Mathematical Programming Series A*, 2014.

Davis, T. A., Hager, W. W., and Hungerford, J. T. An efficient hybrid algorithm for the separable convex quadratic knapsack problem. *ACM Transactions on Mathematical Software*, 42(3):22, 2016.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. Efficient projections onto the l_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 272–279. ACM, 2008.

Gong, P., Gai, K., and Zhang, C. Efficient Euclidean projections via piecewise root finding and its application in gradient projection. *Neurocomputing*, 74(17):2754–2766, 2011.

Hernández-Lobato, D., Hernández-Lobato, J. M., and Ghahramani, Z. A probabilistic model for dirty multi-task feature selection. In *International Conference on Machine Learning*, pp. 1073–1082, 2015.

Huang, J., Zhang, T., et al. The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978–2004, 2010.

Izmailov, A. F. and Solodov, M. V. *Newton-Type Methods for Optimization and Variational Problems*. Springer, 2014.

Jia, Y., Salzmann, M., and Darrell, T. Factorized latent spaces with structured sparsity. In *Advances in Neural Information Processing Systems*, pp. 982–990, 2010.

Kim, J., Monteiro, R. D., and Park, H. Group sparsity in nonnegative matrix factorization. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 851–862. SIAM, 2012.

Liu, H., Palatucci, M., and Zhang, J. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 649–656. ACM, 2009a.

Liu, H., Wasserman, L., and Lafferty, J. D. Nonparametric regression and classification with joint sparsity constraints. In *Advances in Neural Information Processing Systems*, pp. 969–976, 2009b.

Liu, J. and Ye, J. Efficient Euclidean projections in linear time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 657–664. ACM, 2009.

Masaeli, M., Yan, Y., Cui, Y., Fung, G., and Dy, J. G. Convex principal feature selection. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp. 619–628. SIAM, 2010.

Mifflin, R. Semismooth and semiconvex functions in constrained optimization. *SIAM Journal on Control and Optimization*, 15(6):959–972, 1977.

Qi, L. and Sun, D. A survey of some nonsmooth equations and smoothing Newton methods. In *Progress in Optimization*, pp. 121–146. Springer, 1999.

Qi, L. and Sun, J. A nonsmooth version of Newton's method. *Mathematical Programming*, 58(1-3):353–367, 1993.

Quattoni, A., Carreras, X., Collins, M., and Darrell, T. An efficient projection for $\ell_{1,\infty}$ regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 857–864. ACM, 2009.

Rakotomamonjy, A. Sparse support vector infinite push. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 339–346. Omnipress, 2012.

Rakotomamonjy, A., Flamary, R., Gasso, G., and Canu, S. l_p-l_q penalty for sparse linear and sparse multiple kernel multitask learning. *IEEE Transactions on Neural Networks*, 22(8):1307–1320, 2011.

Roth, V. and Fischer, B. The group-lasso for generalized linear models: uniqueness of solutions and efficient algorithms. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 848–855. ACM, 2008.

Sra, S. Fast projections onto l1,q-norm balls for grouped feature selection. In *Machine Learning and Knowledge Discovery in Databases*, pp. 305–317. Springer, 2011.

Sra, S. Fast projections onto mixed-norm balls with applications. *Data Mining and Knowledge Discovery*, 25(2): 358–377, 2012.

Su, H., Yu, W., and Li, F. Efficient Euclidean projections onto the intersection of norm balls. In *Proceedings of the 29th International Conference on Machine Learning*, volume 1, pp. 433–440. International Machine Learning Society., 2012.

Tibshirani, R., Wainwright, M., and Hastie, T. *Statistical Learning with Sparsity: the Lasso and Generalizations*. Chapman and Hall/CRC, 2015.

Turlach, B. A., Venables, W. N., and Wright, S. J. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

Vogt, J. E. and Roth, V. The group-lasso: $\ell_{1,\infty}$ regularization versus $\ell_{1,2}$ regularization. In Goesele, M., Roth, S., Kuijper, A., Schiele, B., and Schindler, K. (eds.), *Proceedings of the 32nd DAGM Conference on Pattern Recognition*, pp. 252–261, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

Xiang, S., Tong, X., and Ye, J. Efficient sparse group feature selection via nonconvex optimization. In *International Conference on Machine Learning*, pp. 284–292, 2013.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (1):49–67, 2006.

Zhang, T., Ghanem, B., Liu, S., and Ahuja, N. Robust visual tracking via structured multi-task sparse learning. *International Journal of Computer Vision*, 101(2):367–383, 2013.