INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 15, Number 1-2, Pages 154–169

A FIXED-POINT PROXIMITY APPROACH TO SOLVING THE SUPPORT VECTOR REGRESSION WITH THE GROUP LASSO REGULARIZATION

ZHENG LI, GUOHUI SONG, AND YUESHENG XU

In Memory of the Professor Ben-yu Guo

Abstract. We introduce an optimization model of the support vector regression with the group lasso regularization and develop a class of efficient two-step fixed-point proximity algorithms to solve it numerically. To overcome the difficulty brought by the non-differentiability of the group lasso regularization term and the loss function in the proposed model, we characterize its solutions as fixed-points of a nonlinear map defined in terms of the proximity operators of the functions appearing in the objective function of the model. We then propose a class of two-step fixed-point algorithms to solve numerically the optimization problem based on the fixed-point equation. We establish convergence results of the proposed algorithms. Numerical experiments with both synthetic data and real-world benchmark data are presented to demonstrate the advantages of the proposed model and algorithms.

Key words. Two-step fixed-point algorithm, proximity operator, group lasso, support vector machine, ADMM

1. Introduction

The support vector machine (SVM) has been widely used in many applications including text/image recognition [8, 35], face detection [29], bioinformatics [4, 6], since its introduction in [13]. In general, we could consider SVM in two main categories [15, 31, 36]: support vector classification (SVC) [16, 18] and support vector regression (SVR) [2, 32, 33]. The standard ℓ^2-norm SVC aims at finding the best hyperplane that has the largest distance to the nearest points of each class. It turns out that this hyperplane is determined by a small fraction of the training points that are called the support vectors. The standard ℓ^2-norm SVR performs in an analogical way. It maximizes the margin from the hyperplane to the nearest points to get the best fitting hyperplane. Similarly, this hyperplane is also determined by only a small subset of the training points. In this paper we shall focus on SVR.

For the purpose of promoting sparsity of the support vectors, the SVM with the ℓ^1-norm regularizer [31, 36, 38] was put forward. It is well received that the ℓ^1-norm regularizer produces sparse solutions [34]. In particular, the ℓ^1-SVM has been proven to be advantageous when there are redundant noise features [38] and to have shorter training time than the standard ℓ^2-SVM [20]. A natural extension of the ℓ^1-norm regularization is the group lasso regularization that could be viewed as a group-wise ℓ^1-norm. It has been shown in [19, 26, 37] that group lasso regularization overwhelms the ℓ^1-norm regularization when the optimal variable has the group structure. The group lasso regularization performs better when the regression problem has the prior information with group structure [14, 26, 37]. On the other hand, applications

Received by the editors February 2, 2017 and, in revised form, April 12, 2017.
2000 *Mathematics Subject Classification.* 65N30.

with cluster structure have been observed in practice [10, 25]. Therefore, in this paper we shall consider the SVM model with the group lasso regularization.

The main challenge of solving the SVM model with the group lasso regularization comes from the non-differentiability of the SVM loss functions and the group lasso regularization term. A popular technique [9, 11] is to solve a smooth approximation of the original model instead. However, it may bring an extra approximation error term and thus we prefer solving the original model rather than a smooth approximation.

The goal of this paper is to develop numerical algorithms of solving the original SVR model with the group lasso regularization. Specifically, we shall employ the techniques of proximity operators to construct a two-step fixed-point proximity algorithm. We point out that fixed-point proximity algorithms have been popular in solving non-differentiable optimization models in image processing [21, 22, 27, 28] and machine learning [1, 23, 24]. We shall first characterize solutions of the non-differential model as fixed-points of certain nonlinear map defined in terms of the proximity operator of the convex functions involved in the objective function. We then employ a matrix splitting technique to derive a class of two-step algorithms to compute the fixed points.

The rest of this paper is organized as follows. In Section 2, we introduce the optimization model of the group lasso regularized SVR. In Section 3, we characterize solutions of the proposed model as the fixed-points of a nonlinear map defined in terms of the proximity operators of the convex functions appearing in the objective function. We develop a class of two-step proximity algorithms for computing the fixed-points and present its convergence analysis in Section 4. We demonstrate the performance of the proposed model and algorithms in Section 5 through numerical experiments with both synthetic data and real-world benchmark data. We draw a conclusion in Section 6.

2. SVR with Group Lasso Regularization

In this section, we shall introduce the model of the SVR with group lasso regularization. To this end, we first recall the models of the standard ℓ^2-norm SVR (ℓ^2-SVR) and the variant ℓ^1-norm SVR (ℓ^1-SVR).

We start with the notation used throughout this paper. We denote by $\mathbb{R}^m$ the usual m-dimensional Euclidean space and define

$$\mathbb{R}^m_+ := \{\boldsymbol{x} \in \mathbb{R}^m : x_i \geq 0,\ 1 \leq i \leq m\}.$$

For a positive integer $m \in \mathbb{N}$, we set $\mathbb{N}_m := \{1, 2, \ldots, m\}$. The standard inner product is defined for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ by

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \sum_{i \in \mathbb{N}_m} x_i y_i.$$

For $p \in \mathbb{N}_2$, we define the ℓ^p norm for $\boldsymbol{x} \in \mathbb{R}^m$ by

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}.$$

We next recall the SVR models. Given instances $\{(\boldsymbol{x}_i, y_i) : i \in \mathbb{N}_m\} \subseteq \mathbb{R}^n \times \mathbb{R}$, the standard ℓ^2-norm soft margin SVR aims at finding the best hyperplane that has the largest margin to the nearest training points. This leads to the following optimization problem

$$
\begin{aligned}
&\min \quad \left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i\in\mathbb{N}_m}(\xi_i+\xi_i^*) : \boldsymbol{w}\in\mathbb{R}^m,\ \boldsymbol{\xi},\boldsymbol{\xi}^*\in\mathbb{R}_+^m,\ b\in\mathbb{R}\right\} \\
&\text{subject to} \quad \langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b - y_i \le \epsilon + \xi_i, \\
&\qquad\qquad\quad y_i - \langle \boldsymbol{w},\boldsymbol{x}_i\rangle - b \le \epsilon + \xi_i^*, \quad i\in\mathbb{N}_m,
\end{aligned}
\tag{1}
$$

where $\epsilon > 0$ is a prescribed real number. The desired determined function f is given by

$$f(\boldsymbol{x}) = \langle \boldsymbol{w},\boldsymbol{x}\rangle + b, \text{ for } \boldsymbol{x}\in\mathbb{R}^n.$$

By the theory of Lagrangian multipliers, the solution $\boldsymbol{w}$ of problem (1) has the form

$$\boldsymbol{w} = \sum_{i\in\mathbb{N}_m}\alpha_i\boldsymbol{x}_i, \text{ for some } \alpha_i\in\mathbb{R}_+,$$

and only a small fraction of $\alpha_i, i\in\mathbb{N}_m$ are non-zero. The training point $\boldsymbol{x}_i$ corresponding to the non-zero parameter α_i is called support vector. By defining ϵ-insensitive loss function [36]

$$\widetilde{L}_\epsilon(\boldsymbol{w},\boldsymbol{x}_i,y_i,b) := \max\big\{|\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b - y_i| - \epsilon, 0\big\},$$

problem (1) has an equivalent unconstrained form [31]:

$$\min \quad \left\{\frac{1}{2}\|\boldsymbol{w}\|_2^2 + \frac{C}{m}\sum_{i\in\mathbb{N}_m}\widetilde{L}_\epsilon(\boldsymbol{w},\boldsymbol{x}_i,y_i,b) : \boldsymbol{w}\in\mathbb{R}^m,\ b\in\mathbb{R}\right\}.$$

The notion of kernels [13, 31, 36] was introduced to handle the nonlinear problem by implicitly mapping the inputs into high-dimensional feature spaces and replacing the inner product with the kernel evaluation. Therefore, when a kernel function $K(\cdot,\cdot)$ is given on $\mathbb{R}^m\times\mathbb{R}^m$, and the standard ℓ^2-SVR performs on the corresponding feature space, the optimization problem is as follows:

$$\min \quad \left\{\frac{1}{2}\sum_{i\in\mathbb{N}_m}\sum_{j\in\mathbb{N}_m}\alpha_i\alpha_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) + \frac{C}{m}\sum_{i\in\mathbb{N}_m}L_\epsilon(\boldsymbol{\alpha},\boldsymbol{x}_i,y_i,b) : \boldsymbol{\alpha}\in\mathbb{R}^m,\ b\in\mathbb{R}\right\},$$

where the loss function

$$L_\epsilon(\boldsymbol{\alpha},\boldsymbol{x}_i,y_i,b) := \max\left\{\left|\sum_{j\in\mathbb{N}_m}\alpha_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) + b - y_i\right| - \epsilon, 0\right\}, \tag{2}$$

and the prediction function f has the form

$$f(\boldsymbol{x}) = \sum_{j\in\mathbb{N}_m}\alpha_j K(\boldsymbol{x}_j,\boldsymbol{x}) + b.$$

In order to further promote sparsity of the support vectors and use the liner combination of the training points as a representation of the solution, SVR with the ℓ^1 norm regularizer [36, 31, 38] is put forward by using a different regularizer, that is, the ℓ^1-norm of the coefficient $\boldsymbol{\alpha}\in\mathbb{R}^m$, as

$$\min \quad \left\{\|\boldsymbol{\alpha}\|_1 + C\sum_{i\in\mathbb{N}_m}L_\epsilon(\boldsymbol{\alpha},\boldsymbol{x}_i,y_i,b) : \boldsymbol{\alpha}\in\mathbb{R}^m,\ b\in\mathbb{R}\right\}. \tag{3}$$

The ℓ^1-SVR (3) is advantageous when there are redundant noise features [38]. By redundant noise features, we mean that the dictionary of basis functions has redundant basis functions. Usually, it has shorter training time than the standard ℓ^2-SVM (1) [20]. However, when the data set has a cluster structure, that is, the variable in problem (3) has a group sparse property, the ℓ^1-norm regularization might not generate a group sparse solution in general. This requires a model to take advantage of the cluster structure in the data set.

We next introduce the SVR with group lasso regularization which serves this purpose. Suppose that the m-dimensional variable can be divided into l disjoint groups $G_j, j \in \mathbb{N}_l$. For $\boldsymbol{\alpha} \in \mathbb{R}^m$ we define

$$\boldsymbol{\alpha}_{G_i} := (\alpha_j : j \in G_i).$$

The group lasso regularized SVR can be written as

$$\text{(4)} \qquad \min \quad \left\{ \sum_{i \in \mathbb{N}_l} \delta_i \|\boldsymbol{\alpha}_{G_i}\|_2 + C \sum_{i \in \mathbb{N}_m} L_\epsilon(\boldsymbol{\alpha}, \boldsymbol{x}_i, y_i, b) : \boldsymbol{\alpha} \in \mathbb{R}^m, \ b \in \mathbb{R} \right\},$$

where $\delta_i > 0, i \in \mathbb{N}_l$ are prescribed parameters. Note that the group lasso is the sum of the ℓ^2-norm of the variable groups, and it would promote solutions that preserve the structure information, or more precisely, the group sparsity [14, 19, 26, 37]. We also remark that both the group lasso term and the loss function in (4) are non-differentiable, which brings challenges to solve this model numerically. A popular approach is to use some differentiable approximations [11] of the group lasso term, or use the squared ϵ-sensitive loss function [31] instead of solving the approximate smooth models. However, this might bring extra approximation errors to the original model and we prefer solving the original model in this paper. In what follows, we refer to the proposed group lasso model (4) as GL-SVR.

We next rewrite problem (4) in a compact form to facilitate the development of our algorithms. To this end, we define for any $\boldsymbol{s} \in \mathbb{R}^{m+1}$

$$\text{(5)} \qquad \varphi_g(\boldsymbol{s}) := \sum_{i \in \mathbb{N}_{l+1}} \delta_i \|\boldsymbol{s}_{G_i}\|_2.$$

Here, G_i's and δ_i's are given groups and parameters for $i \in \mathbb{N}_l$, and for $i = l+1$, we set $G_i = \{m+1\}$ and $\delta_i = 0$. We also define for any $\boldsymbol{s} \in \mathbb{R}^m$

$$\text{(6)} \qquad \psi_{\epsilon,\boldsymbol{y}}(\boldsymbol{s}) := C \sum_{i \in \mathbb{N}_m} (|s_i - y_i| - \epsilon|)_+,$$

where $|t|_+ := \max\{|t|, 0\}, t \in \mathbb{R}$. Let $\boldsymbol{u} \in \mathbb{R}^{m+1}$ be the vector coupling the variables $\boldsymbol{\alpha} \in \mathbb{R}^m$ and $b \in \mathbb{R}$ in (4) as $\boldsymbol{u} := \begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix}$, K be the kernel matrix defined by

$$\mathsf{K} := [K(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j \in \mathbb{N}_m},$$

$\mathbf{1}$ be the $m \times 1$ vector of all ones, and B be the $m \times (m+1)$ matrix defined by

$$\text{(7)} \qquad \mathsf{B} := [\mathsf{K} \ \mathbf{1}].$$

It follows from a direct computation that the GL-SVR model (4) is equivalent to

$$\text{(8)} \qquad \min\{\varphi_g(\boldsymbol{u}) + \psi_{\epsilon,\boldsymbol{y}}(\mathsf{B}\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{R}^{m+1}\}.$$

We observe from the above formulation that both φ_g and $\psi_{\epsilon,\boldsymbol{y}}$ are non-differentiable functions, and this results in the computational difficulty of solving this model. However, though they are non-differentiable, we shall show in Section 4 that their proximity operators have closed form, which makes it amenable to develop proximity algorithms for it.

3. A Characterization of the Solution

In this section, we shall characterize the solutions of model (8) as fixed-points of the proximity operators of the functions appearing in the objective function. It will enable us to develop the proximity algorithms in Section 4. To this end, we first review several necessary notions and results.

We begin by recalling the notions of the proximity operator. We denote by $\Gamma_0(\mathbb{R}^d)$ the class of all lower semi continuous convex functions $f : \mathbb{R}^m \to (-\infty, +\infty]$ such that

$$\mathrm{dom}(f) := \{\boldsymbol{x} \in \mathbb{R}^m : f(\boldsymbol{x}) < +\infty\} \neq \emptyset.$$

The proximity operator of a function $f \in \Gamma_0(\mathbb{R}^n)$ is defined for $\boldsymbol{z} \in \mathbb{R}^m$ by

$$\mathrm{prox}_f(\boldsymbol{z}) := \mathrm{argmin}\{\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{z}\|^2 + f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^m\}.$$

This operator has many good mathematical properties, see [30] for a survey. In particular, its connection to the subdifferential of a convex function plays a crucial role in developing the fixed-point chracterization of the solution of (8). We next review the definition of the subdifferential. The subdifferential of a function $f \in \Gamma_0(\mathbb{R}^m)$ at $\boldsymbol{z} \in \mathbb{R}^m$ is defined by

$$\partial f(\boldsymbol{z}) := \{\boldsymbol{y} : \boldsymbol{y} \in \mathbb{R}^m \text{ and } f(\boldsymbol{x}) \geq f(\boldsymbol{z}) + \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{z}\rangle \text{ for all } \boldsymbol{x} \in \mathbb{R}^m\}.$$

A relation [3, 27] between the proximity operator and the subdifferential may be described for $f \in \Gamma_0(\mathbb{R}^m)$ and $\boldsymbol{z} \in \mathbb{R}^m$ as

$$\boldsymbol{x} \in \partial f(\boldsymbol{z}) \text{ if and only if } \boldsymbol{z} = \mathrm{prox}_f(\boldsymbol{x} + \boldsymbol{z}). \tag{9}$$

It follows immediately from (9) that

$$\boldsymbol{x} \in \partial f(\boldsymbol{z}) \text{ if and only if } \boldsymbol{x} = (\mathsf{I} - \mathrm{prox}_f)(\boldsymbol{x} + \boldsymbol{z}), \tag{10}$$

where I is the identity matrix.

We are now ready to present a characterization of solutions of problem (8). The proof of the following theorem originates from [27]. We outline it for the convenience of the reader.

Theorem 1. *A vector $\boldsymbol{u} \in \mathbb{R}^{m+1}$ is a minimizer of problem* (8) *if and only if there exist $\lambda, \beta > 0$ and $\boldsymbol{q} \in \mathbb{R}^m$ such that*

$$\begin{aligned} \boldsymbol{u} &= \mathrm{prox}_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u} - \frac{C\beta}{\lambda}\mathsf{B}^\top \boldsymbol{q}) \\ \boldsymbol{q} &= (\mathsf{I} - \mathrm{prox}_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}})(\mathsf{B}\boldsymbol{u} + \boldsymbol{q}). \end{aligned} \tag{11}$$

Proof. We first show the necessity. Suppose that $\boldsymbol{u}$ is a minimizer of (8). It follows from Fermat's rule (see [3], chap. 16) and the chain rule [3] that

$$\boldsymbol{0} \in \partial(\varphi_g(\boldsymbol{u}) + \psi_{\epsilon,\boldsymbol{y}}(\mathsf{B}\boldsymbol{u})) = \partial_{\varphi_g}(\boldsymbol{u}) + \mathsf{B}^T\partial_{\psi_{\epsilon,\boldsymbol{y}}}(\mathsf{B}\boldsymbol{w}).$$

Since both φ_g and $\psi_{\epsilon,\boldsymbol{y}}$ are convex and the subdifferential of a convex function is a nonempty set [3], for any positive numbers λ and β, there exist

$$\boldsymbol{p} \in \partial_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u}) \text{ and } \boldsymbol{q} \in \partial_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}}(\mathsf{B}\boldsymbol{u}) \tag{12}$$

such that

$$\boldsymbol{0} = \lambda\boldsymbol{p} + C\beta\mathsf{B}^T\boldsymbol{q}. \tag{13}$$

Since

$$\boldsymbol{q} \in \partial_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}}(\mathsf{B}\boldsymbol{u}),$$

the second equality of (11) follows from the relation (10) between the proximity operator and the subdifferential of a convex function. Moreover, from (13) and the first inclusion of (12), we obtain $\frac{C\beta}{\lambda}\mathsf{B}^\top \boldsymbol{q} \in \partial_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u})$, which together with (9) implies the first equality of (11).

We next show the sufficiency. It follows from (11) and the relations (9) and (10) that

$$-\frac{C\beta \mathsf{B}^T \boldsymbol{q}}{\lambda} \in \partial_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u}) \quad \text{and} \quad \boldsymbol{q} \in \partial_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}}(\mathsf{B}\boldsymbol{u}),$$

which imply that

$$\boldsymbol{0} = -C\beta \mathsf{B}^T \boldsymbol{q} + C\beta \mathsf{B}^T \boldsymbol{q} \in \partial_{\varphi_g}(\boldsymbol{u}) + \mathsf{B}^T \partial_{\psi_{\epsilon,\boldsymbol{y}}}(\mathsf{B}\boldsymbol{u}).$$

That is,

$$0 \in \partial(\varphi_g(\boldsymbol{u}) + \psi_{\epsilon,\boldsymbol{y}}(\mathsf{B}\boldsymbol{u})).$$

By Fermat's rule, $\boldsymbol{u}$ is a minimizer of (8). □

We observe from the above Theorem that the minimization problem (8) is transferred into a fixed point problem. This equivalent reformulation brings convenience in both designing numerical algorithms and conducting the convergence analysis as we shall see in Section 4.

For the simplicity of presentation, we rewrite the characterization (11) into a compact form by coupling the two equations together. We define a vector $\boldsymbol{v} \in \mathbb{R}^{2m+1}$ coupling the vectors $\boldsymbol{u} \in \mathbb{R}^{m+1}$ and $\boldsymbol{q} \in \mathbb{R}^m$ as

$$\boldsymbol{v} := \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{q} \end{pmatrix},$$

and an operator $T : \mathbb{R}^{2m+1} \to \mathbb{R}^{2m+1}$ coupling the proximity operator of the function $\frac{1}{\lambda}\varphi_g$ and the operator of $\mathsf{I} - \operatorname{prox}_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}}$ as for any $\boldsymbol{v} \in \mathbb{R}^{2m+1}$

$$T(\boldsymbol{v}) := \begin{pmatrix} \operatorname{prox}_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u}) \\ \mathsf{I} - \operatorname{prox}_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}}(\boldsymbol{q}) \end{pmatrix}. \tag{14}$$

Let

$$\mathsf{P} := \frac{\lambda}{C\beta}\mathsf{I}. \tag{15}$$

The characterization (11) can be reformulated as

$$\boldsymbol{v} = T \circ \mathsf{E}(\boldsymbol{v}), \tag{16}$$

where

$$\mathsf{E} := \begin{bmatrix} \mathsf{I} & -\mathsf{P}^{-1}\mathsf{B}^\top \\ \mathsf{B} & \mathsf{I} \end{bmatrix}. \tag{17}$$

4. A Class of Two-Step Fixed-Point Proximity Algorithms

In this section, we shall develop efficient algorithms to solve the fixed-point problem (16). In particular, we first show that due to the expansiveness of the matrix E, the algorithm generated by directly applying the Picard iteration on equation (16) may not be convergent. We shall then introduce a matrix splitting technique to derive a two-step iteration scheme and prove its convergence, following the approach developed in [21]. Moreover, we shall also show that the two-step iterative scheme will speed up the convergence through numerical experiments in Section 5.

We first study the Picard iteration algorithm of solving the fixed-point equation (16) directly. Given the matrix B, the positive parameters C, λ and β. Choose $\boldsymbol{u}^0$

and $\boldsymbol{q}^0$ as the initial points. Let $\boldsymbol{v}^0 := (\boldsymbol{w}^0, \boldsymbol{q}^0)^\top$, T be defined by (14), and E be defined by (17). The Picard iterative sequence $\{\boldsymbol{v}^k\}_{k\in\mathbb{N}}$ of $T \circ \mathsf{E}$ is generated by the following iteration

$$\boldsymbol{v}^{k+1} = T \circ \mathsf{E}(\boldsymbol{v}^k). \tag{18}$$

We point out that the convergence of the above sequence depends on whether the operator $T \circ \mathsf{E}$ is *firmly nonexpansive*. We next give a brief review of the definition and some properties of firmly nonexpansive operators. We denote by $\mathbb{S}_+$ the set of symmetric positive definite matrices. An operator S is called *firmly nonexpansive* with respect to $\mathsf{R} \in \mathbb{S}_+$ if

$$\|S(\boldsymbol{v}_1) - S(\boldsymbol{v}_2)\|_{\mathsf{R}}^2 \le \langle S(\boldsymbol{v}_1) - S(\boldsymbol{v}_2), \mathsf{R}(\boldsymbol{v}_1 - \boldsymbol{v}_2)\rangle.$$

Here, $\|\boldsymbol{x}\|_{\mathsf{R}} := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle_{\mathsf{R}}}$ is the norm induced by the weighted inner product defined by $\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathsf{R}} := \langle \boldsymbol{x}, \mathsf{R}\boldsymbol{y}\rangle$. It has been shown in [5] that the sequence $\{S^k \boldsymbol{w}^0 : k \in \mathbb{N}\}$ converges to a fixed-point of S for any initial $\boldsymbol{w}^0$ when S is firmly nonexpansive with respect to a certain positve definite matrix R.

It has been proved in [12] that the proximity operators are firmly nonexpansive, that is, the operator T is firmly nonexpansive. If E were also nonexpansive, then the composition $T \circ \mathsf{E}$ would be nonexpansive [5]. However, as we can see in the following result, the matrix E in (16) is *not* nonexpansive. The proof of the following proposition originates from [21]. We also outline it for the convenience of the reader.

Proposition 1. *If* E *is the operator defined in* (17)*,*

$$\|\mathsf{E}\|_{\mathsf{R}} := \sup\{\|\mathsf{E}\boldsymbol{v}\|_{\mathsf{R}}, \|\boldsymbol{v}\|_{\mathsf{R}} = 1\},$$

and

$$\mathsf{R} := \begin{pmatrix} \mathsf{P} & 0 \\ 0 & \mathsf{I} \end{pmatrix}, \tag{19}$$

where P *is defined by* (15)*, then* $\|\mathsf{E}\|_{\mathsf{R}} > 1$ *and* E *is not nonexpansive.*

Proof. We show the desired result by a direct computation of $\|\mathsf{E}\|_{\mathsf{R}}$. For any $\boldsymbol{v} = (\boldsymbol{u}, \boldsymbol{q}) \in \mathbb{R}^{2m+1}$ with $\|\boldsymbol{v}\|_{\mathsf{R}}^2 = 1$, it follows from the definition of E in (17) that

$$\|\mathsf{E}\boldsymbol{v}\|_{\mathsf{R}}^2 = \|\boldsymbol{u}\|_{\mathsf{P}}^2 - 2\langle \boldsymbol{u}, \mathsf{P}^{-1}\mathsf{B}^T\boldsymbol{q}\rangle_{\mathsf{P}} + \|\mathsf{P}^{-1}\mathsf{B}^T\boldsymbol{q}\|_{\mathsf{P}}^2 + \|\boldsymbol{q}\|_2^2 + 2\langle \boldsymbol{q}, \mathsf{B}\boldsymbol{u}\rangle + \|\mathsf{B}\boldsymbol{u}\|_2^2.$$

Note that $\|\boldsymbol{v}\|_{\mathsf{R}}^2 = \|\boldsymbol{u}\|_{\mathsf{P}}^2 + \|\boldsymbol{q}\|_2^2 = 1$, and $\langle \boldsymbol{u}, \mathsf{P}^{-1}\mathsf{B}^T\boldsymbol{q}\rangle_{\mathsf{P}} = \langle \boldsymbol{q}, \mathsf{B}\boldsymbol{u}\rangle$. It follows that

$$\|\mathsf{E}\boldsymbol{v}\|_{\mathsf{R}}^2 = 1 + \|\mathsf{P}^{-1}\mathsf{B}^T\boldsymbol{q}\|_{\mathsf{P}}^2 + \|\mathsf{B}\boldsymbol{u}\|_2^2.$$

Since B is non-singular, there exists a non-zero vector $\boldsymbol{v} = (\boldsymbol{u}, \boldsymbol{q})$ with $\|\boldsymbol{v}\|_{\mathsf{R}} = 1$ such that

$$\|\mathsf{P}^{-1}\mathsf{B}^T\boldsymbol{q}\|_{\mathsf{P}}^2 + \|\mathsf{B}\boldsymbol{u}\|_2^2 > 0.$$

Therefore, by the definition of $\|\mathsf{E}\|_{\mathsf{R}}$, we have that $\|\mathsf{E}\|_{\mathsf{R}} > 1$. □

We point out that the Picard iteration (18) may not yield a convergent sequence since E is not non-expansive. To overcome this difficulty, we shall split the expansive matrix E to derive an equivalent fixed point formulation of a non-expansive operator.

To this end, we first show how to split the matrix E to obtain a two-step iterative scheme. We choose appropriate matrices $\mathsf{M}_1, \mathsf{M}_2$ (which would be specified later in this section) and decompose the matrix E as

$$\mathsf{E} = (\mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2) + \mathsf{M}_1 + \mathsf{M}_2.$$

Equation (16) can then be rewritten as

$$\boldsymbol{v} = T \circ ((\mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2)\boldsymbol{v} + \mathsf{M}_1\boldsymbol{v} + \mathsf{M}_2\boldsymbol{v}). \tag{20}$$

Instead of using the Picard iteration (18), we consider the following iteration

$$\boldsymbol{v}^{k+1} = T \circ ((\mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2)\boldsymbol{v}^{k+1} + \mathsf{M}_1\boldsymbol{v}^k + \mathsf{M}_2\boldsymbol{v}^{k-1}). \tag{21}$$

We observe from the above iterative scheme that it is an implicit scheme. However, one can choose M_1 and M_2 such that $\mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2$ is strictly block upper (or lower) triangular and it would lead to an explicit iterative scheme, since $\boldsymbol{v}$ has two blocks $\boldsymbol{w}$ and $\boldsymbol{y}$. We also observe that the above iteration is a two-step scheme that makes each iteration more efficient and speeds up the overall convergence, as we can see from the numerical experiments in Section 5.

We next make specific choices of the matrices M_1 and M_2 to split the expansive matrix E. Namely, we choose

$$\mathsf{M}_1 := \begin{bmatrix} \mathsf{I} & (1-\theta)\mathsf{P}^{-1}\mathsf{B}^T \\ (1+\theta)\mathsf{B} & \mathsf{I} \end{bmatrix}, \quad \mathsf{M}_2 := \begin{bmatrix} 0 & 0 \\ -\theta\mathsf{B} & 0 \end{bmatrix}, \tag{22}$$

where θ is a constant to be specified later in convergence analysis. Substituting M_1 and M_2 into iterative scheme (21), we have the following iterative scheme

$$\begin{aligned} \boldsymbol{q}^{k+1} &= (\mathsf{I} - \operatorname{prox}_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}})(\boldsymbol{q}^k + \mathsf{B}(\boldsymbol{u}^k + \theta(\boldsymbol{u}^k - \boldsymbol{u}^{k-1}))) \\ \boldsymbol{u}^{k+1} &= \operatorname{prox}_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u}^k - \frac{C\beta}{\lambda}\mathsf{B}^\top(\boldsymbol{q}^{k+1} + (1-\theta)(\boldsymbol{q}^{k+1} - \boldsymbol{q}^k))). \end{aligned} \tag{23}$$

It can be directly observed that the above iteration scheme has an explicit form. We are now ready to present a two-step fixed-point proximity algorithm for solving GL-SVR.

Algorithm 1 Two-step Fixed-Point Proximity Algorithm (TFP^{2}A)

```
Given: the matrix $\mathsf{B}$, the positive parameters $C$, $\theta$, $\lambda$ and $\beta$.
Initialization: $\boldsymbol{u}^0$, and $\boldsymbol{q}^0$.
repeat
    Step 1: $\boldsymbol{q}^{k+1} = (\mathsf{I} - \operatorname{prox}_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}})(\boldsymbol{q}^k + \mathsf{B}(\boldsymbol{u}^k + \theta(\boldsymbol{u}^k - \boldsymbol{u}^{k-1})))$
    Step 2: $\boldsymbol{u}^{k+1} = \operatorname{prox}_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u}^k - \frac{C\beta}{\lambda}\mathsf{B}^\top(\boldsymbol{q}^{k+1} + (1-\theta)(\boldsymbol{q}^{k+1} - \boldsymbol{q}^k)))$
until "convergence"
```

We remark that compared with the original Picard iteration scheme (18), the proposed TFP^{2}A splits the expansive operator, and results in a nonexpansive iterative scheme, as we shall see in the convergence analysis to be presented later. We also remark that both proximity operators in TFP^{2}A can be explicitly calculated.

Efficient implementation of Algorithm 1 requires the availability of closed forms of the proximity operator of the functions $\psi_{\epsilon,\boldsymbol{y}}$ and φ_g. We first compute the proximity operator of the function $\psi_{\epsilon,\boldsymbol{y}}$. To do this, we define a function $\phi_\epsilon : \mathbb{R}^m \to \mathbb{R}$ as for any $\boldsymbol{z} \in \mathbb{R}^m$

$$\phi_\epsilon(\boldsymbol{z}) := \sum_{i\in\mathbb{N}_m} (|z_i| - \epsilon)_+. \tag{24}$$

Proposition 2. *If ϕ_ϵ is defined by (24), then for any $\boldsymbol{z} \in \mathbb{R}^n$ and $\beta > 0$, we have that if $\epsilon \geq \frac{C}{2\beta}$,*

$$(\operatorname{prox}_{\frac{1}{\beta}\phi_\epsilon}(\boldsymbol{z}))_j = \begin{cases} z_j - \frac{C}{\beta}, & \textit{if } z_j \geq \epsilon + \frac{C}{\beta} \\ \epsilon, & \textit{if } \epsilon \leq z_j < \epsilon + \frac{C}{\beta} \\ z_j, & \textit{if } \epsilon - \frac{C}{\beta} \leq z_j < \epsilon \\ z_j + \frac{C}{\beta}, & \textit{if } -\epsilon \leq z_j < \epsilon - \frac{C}{\beta} \\ -\epsilon, & \textit{if } -\epsilon - \frac{C}{\beta} \leq z_j < -\epsilon \\ z_j + \frac{C}{\beta}, & \textit{if } z_j < -\epsilon - \frac{C}{\beta} \end{cases}, \quad j \in \mathbb{N}_n, \tag{25}$$

if $\epsilon < \frac{C}{2\beta}$,

$$(\operatorname{prox}_{\frac{1}{\beta}\phi_\epsilon}(\boldsymbol{z}))_j = \begin{cases} z_j - \frac{C}{\beta}, & \textit{if } z_j \geq \epsilon + \frac{C}{\beta} \\ \epsilon, & \textit{if } \epsilon \leq z_j < \epsilon + \frac{C}{\beta} \\ z_j, & \textit{if } -\epsilon \leq z_j < \epsilon \\ -\epsilon, & \textit{if } -\epsilon - \frac{C}{\beta} \leq z_j < -\epsilon \\ z_j + \frac{C}{\beta}, & \textit{if } z_j < -\epsilon - \frac{C}{\beta} \end{cases}, \quad j \in \mathbb{N}_n. \tag{26}$$

Proof. Note that the proximity operator of ϕ_ϵ can be computed component-wise. For each $1 \leq j \leq n$, we have

$$(\operatorname{prox}_{\frac{1}{\beta}\phi_\epsilon}(\boldsymbol{z}))_j = \operatorname{argmin}\left\{\frac{1}{2}(x_j - z_j)^2 + \frac{C}{\beta}(|x_j| - \epsilon)_+ : x_j \in \mathbb{R}\right\}. \tag{27}$$

Let

$$f(x_j) := \frac{1}{2}(x_j - z_j)^2 + \frac{1}{\beta}(x_j)_+ \ \text{ and } \ t := \operatorname{argmin} f(x_j).$$

When $\epsilon \geq \frac{C}{2\beta}$, we compute t in cases $z_j \geq \epsilon + \frac{C}{\beta}$, $\epsilon \leq z_j < \epsilon + \frac{C}{\beta}$, $\epsilon - \frac{C}{\beta} \leq z_j < \epsilon$, $-\epsilon \leq z_j < \epsilon - \frac{C}{\beta}$, $-\epsilon - \frac{C}{\beta} \leq z_j < -\epsilon$ and $z_j < -\epsilon - \frac{C}{\beta}$. For the first case $z_j \geq \epsilon + \frac{C}{\beta}$, when $x_j \geq \epsilon$, we have that

$$f(x_j) = \frac{1}{2}(x_j - z_j)^2 + \frac{C}{\beta}(x_j - \epsilon).$$

Since $z_j - \frac{C}{\beta} \geq \epsilon$, we have that the minimizer of $f(x_j)$ on $[\epsilon, \infty)$ is $z_j - \frac{C}{\beta}$ and $f(\epsilon) \geq f(z_j - \frac{C}{\beta})$. When $-\epsilon \leq x_j < \epsilon$, we have that

$$f(x_j) = \frac{1}{2}(x_j - z_j)^2.$$

Since $z_j > \epsilon$, $f(x_j)$ decreases on $[-\epsilon, \epsilon)$ and it follows that $f(-\epsilon) > f(\epsilon)$. When $x_j < -\epsilon$, then we have

$$f(x_j) = \frac{1}{2}(x_j - z_j)^2 + \frac{C}{\beta}(-x_j - \epsilon).$$

Since $z_j + \frac{C}{\beta} > \epsilon$, $f(x_j)$ decreases on $(-\infty, -\epsilon)$. Therefore, the minimizer of $f(x_j)$ on $\mathbb{R}$ is $z_j - \frac{C}{\beta}$. The minimizer of the other cases can be computed in a similar way.

On the other hand, when $\epsilon < \frac{C}{2\beta}$, we can obtain equation (26) by a similar computation as above. □

To derive the proximity operator of function $\psi_{\epsilon,\boldsymbol{y}}$, we recall a fact in [30]. Suppose that $f, g \in \Gamma_0(\mathbb{R}^m)$. If $f(\boldsymbol{x}) = g(\boldsymbol{x} + \boldsymbol{a})$ for any $\boldsymbol{x}, \boldsymbol{a} \in \mathbb{R}^m$, then

$$\operatorname{prox}_f(\boldsymbol{x}) = \operatorname{prox}_g(\boldsymbol{x} + \boldsymbol{a}) - \boldsymbol{a}. \tag{28}$$

Note that for any $\boldsymbol{z} \in \mathbb{R}^m$, $\psi_{\epsilon,\boldsymbol{y}}(\boldsymbol{z}) = \phi_\epsilon(\boldsymbol{z} - \boldsymbol{y})$, where $\boldsymbol{y}$ is a vector consisting of the labels $y_i, i \in \mathbb{N}_m$. It follows from (28) that

$$\operatorname{prox}_{\psi_{\epsilon,\boldsymbol{y}}}(\boldsymbol{z}) = \operatorname{prox}_{\phi_\epsilon}(\boldsymbol{z} - \boldsymbol{y}) + \boldsymbol{y}.$$

We next compute the operator of function φ_g.

Proposition 3. *If φ_g is defined by* (24)*, then for any $\boldsymbol{z} \in \mathbb{R}^{m+1}$ and $\lambda > 0$, we have that*

$$(29) \qquad (\operatorname{prox}_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{z}))_{G_j} = \max\left\{\|\boldsymbol{z}_{G_j}\|_2 - \frac{\delta_j}{\lambda}, 0\right\} \frac{\boldsymbol{z}_{G_j}}{\|\boldsymbol{z}_{G_j}\|_2}.$$

Proof. It suffices to compute the proximity operator at $\boldsymbol{z}$ group-wise, since the groups of the variable $\boldsymbol{z}$ are non-overlapped. Note that for each group, we need to compute a proximity operator of the ℓ^2-norm at the group of the variable. And it has been shown in [28] that for any $s \in \mathbb{R}^d$ and $\lambda > 0$, the proximity operator of $\frac{1}{\lambda}\|\boldsymbol{s}\|_2$ is

$$(30) \qquad \operatorname{prox}_{\frac{1}{\lambda}\|\cdot\|_2}(\boldsymbol{s}) = \max\left\{\|\boldsymbol{s}\|_2 - \frac{1}{\lambda}, 0\right\} \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|_2}.$$

Therefore, for each group $G_j, j \in \mathbb{N}_{l+1}$, by replacing the parameter $\frac{1}{\lambda}$ by $\frac{\delta_j}{\lambda}$ in (30), we have equation (29). □

The rest of this section is devoted to convergence analysis of the proposed TFP^{2}A. To this end, we first review the definition of weakly firmly nonexpansive introduced originally in [21].

Suppose that for any $\boldsymbol{y}, \boldsymbol{z} \in \mathbb{R}^{2m+1}$ there exists $\boldsymbol{x} \in \mathbb{R}^{2m+1}$ such that

$$(31) \qquad \boldsymbol{x} = T(\mathsf{E}_0\boldsymbol{x} + \mathsf{M}_1\boldsymbol{y} + \mathsf{M}_2\boldsymbol{z}),$$

where $\mathsf{E}_0 = \mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2$ and $\mathsf{E}, \mathsf{M}_1, \mathsf{M}_2$ are defined by (17) and (22). Let $\mathcal{M} := \{M_1, M_2\}$. We define a mapping $T_{\mathcal{M}} : \mathbb{R}^{4m+2} \to \mathbb{R}^{2m+1}$ as

$$(32) \qquad T_{\mathcal{M}} : (\boldsymbol{y}, \boldsymbol{z}) \to \{\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^d, (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \text{ satisfies equation (31)}\}.$$

We say an operator $T_{\mathcal{M}} : \mathbb{R}^{2d} \to \mathbb{R}^d$ is *weakly firmly nonexpansive* with respect to a matrix set $\mathcal{M} := \{\widetilde{\mathsf{M}_1}, \widetilde{\mathsf{M}_2}\}$ if for any $(\boldsymbol{x}_i, \boldsymbol{u}_i, \boldsymbol{z}_i) \in \operatorname{gra}(T_{\mathcal{M}})$, the graph of $T_{\mathcal{M}}$, for $i = 1, 2$,

$$(33) \quad \langle \boldsymbol{x}_2 - \boldsymbol{x}_1, (\widetilde{\mathsf{M}_1} + \widetilde{\mathsf{M}_2})(\boldsymbol{x}_2 - \boldsymbol{x}_1)\rangle \le \langle \boldsymbol{x}_2 - \boldsymbol{x}_1, \widetilde{\mathsf{M}_1}(\boldsymbol{u}_2 - \boldsymbol{u}_1) + \widetilde{\mathsf{M}_2}(\boldsymbol{z}_2 - \boldsymbol{z}_1)\rangle.$$

We first show that the mapping in TFP^{2}A is weakly firmly nonexpansive mapping and then employ the results in [21] to derive the convergence analysis of TFP^{2}A.

Lemma 1. *If $T_{\mathcal{M}}$ is an operator defined by* (32) *with the set $\mathcal{M} = \{\mathsf{M}_1, \mathsf{M}_2\}$ defined by* (22)*, then $T_{\mathcal{M}}$ is continuous and weakly firmly nonexpansive with respect to $\mathcal{M}$.*

Proof. We first show the weakly firmly nonexpansivity of the operator $T_{\mathcal{M}}$. Recalling the definition of the operator $T_{\mathcal{M}}$, we obtain that for any $(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{z}_i) \in \operatorname{gra}(T_{\mathcal{M}})$,

$$\boldsymbol{x}_i = T((\mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2)\boldsymbol{x}_i + \mathsf{M}_1\boldsymbol{y}_i + \mathsf{M}_2\boldsymbol{z}_i), i = 1, 2.$$

Since the operator T defined by (14) is firmly nonexpansive, we have that

$$(34) \ \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|^2 \le \langle \boldsymbol{x}_2 - \boldsymbol{x}_1, (\mathsf{E} - \mathsf{M}_1 - \mathsf{M}_2)(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \mathsf{M}_1(\boldsymbol{y}_2 - \boldsymbol{y}_1) + \mathsf{M}_2(\boldsymbol{z}_2 - \boldsymbol{z}_1)\rangle.$$

By the definition of the matrix E in (17), we have that

$$(35) \qquad \mathsf{E} = \mathsf{I} + \mathsf{R}^{-1}\mathcal{S}_{\mathsf{B}}, \quad \text{where } \mathsf{R} \text{ is defined by (19)}, \ \mathcal{S}_{\mathsf{B}} := \begin{pmatrix} 0 & -\mathsf{B}^\top \\ \mathsf{B} & 0 \end{pmatrix}.$$

Substituting equality (35) into inequality (34) and noticing the fact that

$$\langle \boldsymbol{x}_2 - \boldsymbol{x}_1, \mathcal{S}_{\mathsf{B}}(\boldsymbol{x}_2 - \boldsymbol{x}_1)\rangle = 0,$$

we have the desired inequality (33), which means $T_{\mathcal{M}}$ is weakly firmly nonexpansive.

The continuity of $T_{\mathcal{M}}$ follows from the continuity of the operator T in (14), and this ends the proof. □

We are now ready to present the main convergence result of TFP^{2}A.

Theorem 2. *Suppose* B *is the matrix defined in* (7)*. If* $\theta \in \mathbb{R}$ *and positive constants* C*,* λ*,* β *satisfy*

$$\frac{\beta(1-\theta)^2}{\lambda} < \frac{1}{C\|\mathsf{B}\|_2^2}, \tag{36}$$

and

$$\frac{\max\{\frac{C\beta}{\lambda}, 1\}}{1 - |1-\theta|\sqrt{\frac{C\beta}{\lambda}}\|\mathsf{B}\|_2}|\theta|\|\mathsf{B}\|_2 < \frac{1}{2}, \tag{37}$$

then the sequence $\{\boldsymbol{u}^k\}_{k\in\mathbb{N}}$ *generated by TFP*^{2}*A converges to a solution of problem* (8)*.*

Proof. By Lemma 1, we have that $T_{\mathcal{M}}$ is weakly firmly nonexpansive. Since

$$\mathsf{P} = \frac{\lambda}{C\beta}\mathsf{I},$$

it follows from a direct computation from (36) and (37) that

$$|1-\theta|\|\mathsf{B}\mathsf{P}^{-\frac{1}{2}}\|_2 < 1,$$

and

$$\frac{\max\{\|\mathsf{P}^{-1}\|_2, 1\}}{1 - |1-\theta|\|\mathsf{B}\mathsf{P}^{-\frac{1}{2}}\|_2}|\theta|\|\mathsf{B}\|_2 < \frac{1}{2}.$$

This implies that $\widetilde{\mathsf{H}} := \mathsf{R}(\widetilde{\mathsf{M}_1} + 2\widetilde{\mathsf{M}_2})$ is symmetric positive definite and

$$\left\|\widetilde{\mathsf{H}}^{-\frac{1}{2}}\mathsf{R}\widetilde{\mathsf{M}_2}\widetilde{\mathsf{H}}^{-\frac{1}{2}}\right\|_2 < \frac{1}{2},$$

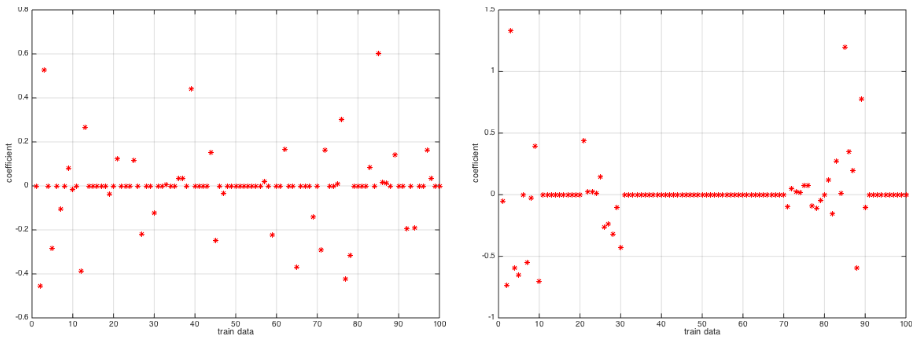
where R are defined by (19). By Theorem 4.6 in [21], the sequence $\{\boldsymbol{v}^k\}$ generated by (21) converges to a fixed point $\boldsymbol{v}^*$ of $T_{\mathcal{M}}$, that is,

$$\boldsymbol{v}^* = T_{\mathcal{M}}(\boldsymbol{v}^*, \boldsymbol{v}^*).$$

We let $\boldsymbol{v}^* = (\boldsymbol{u}^*, \boldsymbol{q}^*)$. It follows that $\boldsymbol{u}^*$ is a solution of problem (8). Since $\{\boldsymbol{v}^k\}$ converges to $\boldsymbol{v}^*$, we also have $\{\boldsymbol{u}^k\}$ converges to $\boldsymbol{u}^*$, which finishes the proof. □

5. Numerical Experiments

In this section, we present numerical results to demonstrate the advantages of the proposed GL-SVR model and the TFP^{2}A algorithm. Specifically, we first conduct a numerical experiment to show that on a simulation data set with group structure, the proposed model is more effective than the standard ℓ^1-norm SVR. We further compare TFP^{2}A with ADMM on two real-world benchmark data sets to show the efficiency of TFP^{2}A. All the numerical experiments are implemented on a personal computer with a 2.6 GHz Intel Core i5 CPU and an 8G RAM memory.



(a) ℓ^1-SVR (b) GL-SVR

FIGURE 1. The result of training ℓ^1-SVR and GL-SVR.

5.1. Simulation Data. The simulation data set contains 100 instances as training data and 100 instances as testing data. They are generated randomly on the domain $[0,1] \times [0,1]$. We denote the whole data set as $\mathcal{X}_{whole}$ and the set of the first 100 training instances as $\mathcal{X}_{train}$. The labels of the instances in $\mathcal{X}_{whole}$ are generated by a group sparse kernel combination of the instances in $\mathcal{X}_{train}$. That is, for each $\boldsymbol{x}_i \in \mathcal{X}_{whole}$, $i \in \mathbb{N}_{200}$, we generate the corresponding label y_i as

$$y_i = \sum_{j \in \mathbb{N}_{100}} \alpha_j K(\boldsymbol{x}_j, \boldsymbol{x}_i) + b,$$

where $\boldsymbol{x}_j \in \mathcal{X}_{train}, j \in \mathbb{N}_{100}$, and the coefficients $\alpha_j, j \in \mathbb{N}_{100}$ are divided into 10 groups. α_j in odd groups are randomly set as 1 or -1, and α_j in even groups are set as 0. Here, we choose Gaussian kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) := \exp(-g\|\boldsymbol{x} - \boldsymbol{y}\|^2), \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^2$$

as the base kernel with parameter $g = 1$, and offset $b = -0.5$.

We compare the performance of ℓ^1-SVR (3) and GL-SVR (4) on this simulation data set. We apply the proximity algorithm proposed in [23] to solve the ℓ^1-SVR model, and use TFP^{2}A to solve GL-SVR (4). We use the same Gaussian kernel with parameter $g = 1$ and the same model parameter $\epsilon = 0.01$. And parameters of the algorithms are tuned to approximately achieve the best performance for each model, while maintaining the same sparsity of each solution in order to be fair. We compare the testing mean squared error (MSE) and the number of support vectors of the two models. The numerical result is presented in Table 1. We further visualize the estimated coefficients derived from the two models in Figure 1. The estimated coefficients of the two models are illustrated in Figures 1. Clearly, in the left figure, the solution of ℓ^1-SVR is globally sparse; while in the right figure, the solution of GL-SVR is sparse in groups.

TABLE 1. Comparison of ℓ^1-SVM and GL-SVM in the mean squared error (MSE) and numbers of support vectors (SVs).

Simulation	MSE	SVs
ℓ^1-SVR	3.68×10^{-4}	41
GL-SVR	$1.30\ \times10^{-4}$	40

We observe from Table 1 and Figure 1 that the solution of Gl-SVR achieves sparsity in groups and has a smaller MSE than ℓ^1-SVR does, when the data set has a group structure.

5.2. Real World Data. We next compare TFP^{2}A with ADMM for solving GL-SVR on two real world data sets from [7]. To this end, we first describe an ADMM algorithm for solving GL-SVR model (8), followed by a detailed discussion on the comparison of the proposed TFP^{2}A and ADMM from the multi-step point of view.

We now describe the ADMM algorithm for solving problem (8). Given the matrix B, the positive parameters C, μ and γ, we choose $\boldsymbol{u}^0$, $\boldsymbol{z}^0$, and $\boldsymbol{x}^0$ as initial points and define the iteration scheme as below. For $k = 0, 1, \ldots$, we generate $\boldsymbol{u}^{k+1}$, $\boldsymbol{z}^{k+1}$, and $\boldsymbol{x}^{k+1}$ from $\boldsymbol{u}^k$, $\boldsymbol{z}^k$, and $\boldsymbol{x}^k$ via the alternating iteration

$$\begin{aligned} \boldsymbol{z}^{k+1} &= \operatorname{prox}_{\gamma\psi_{\epsilon,\boldsymbol{y}}}(\mathsf{B}\boldsymbol{u}^k + \boldsymbol{x}^k) \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \mathsf{B}\boldsymbol{u}^k - \boldsymbol{z}^{k+1} \\ \boldsymbol{u}^{k+1} &= \operatorname{prox}_{\mu\varphi_g}(\boldsymbol{u}^k - \frac{\mu}{\gamma}\mathsf{B}^T(\mathsf{B}\boldsymbol{u}^k - \boldsymbol{z}^{k+1} + \boldsymbol{x}^{k+1})). \end{aligned} \tag{38}$$

The algorithm parameters μ and γ are chosen to satisfy

$$0 < \mu \leq \frac{\gamma}{\|\mathsf{B}\|_2^2}$$

to ensure convergence of the algorithm. We remark that this scheme follows from the augmented Lagrangian and a linearized technique, see [17, 30] for more details.

For convenience of understanding the difference between TFP^{2}A and ADMM, we reformulate the above ADMM iteration scheme (38) as follows. Let

$$\boldsymbol{q}^k := \boldsymbol{x}^k, \quad \gamma := \frac{1}{C\beta}, \quad \mu := \frac{1}{\lambda}.$$

It follows from a direct computation that (38) is equivalent to

$$\begin{aligned} \boldsymbol{q}^{k+1} &= (\mathsf{I} - \operatorname{prox}_{\frac{1}{C\beta}\psi_{\epsilon,\boldsymbol{y}}})(\mathsf{B}\boldsymbol{u}^k + \boldsymbol{q}^k) \\ \boldsymbol{u}^{k+1} &= \operatorname{prox}_{\frac{1}{\lambda}\varphi_g}(\boldsymbol{u}^k - \frac{C\beta}{\lambda}\mathsf{B}^\top(2\boldsymbol{q}^{k+1} - \boldsymbol{q}^k)). \end{aligned}$$

We further write it in a compact form by coupling the two equations together. Introducing $\boldsymbol{v}^k := (\boldsymbol{u}^k, \boldsymbol{q}^k)^\top$,

$$\bar{\mathsf{E}}_0 := \begin{bmatrix} 0 & -2\mathsf{P}^{-1}\mathsf{B} \\ 0 & 0 \end{bmatrix}, \quad \bar{\mathsf{M}}_1 := \begin{bmatrix} \mathsf{I} & \mathsf{P}^{-1}\mathsf{B}^\top \\ \mathsf{B} & \mathsf{I} \end{bmatrix}, \quad \bar{\mathsf{M}}_2 := \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

the above iteration scheme is equivalently rewritten as

$$\boldsymbol{v}^{k+1} = T\left(\bar{\mathsf{E}}_0\boldsymbol{v}^{k+1} + \bar{\mathsf{M}}_1\boldsymbol{v}^k + \bar{\mathsf{M}}_2\boldsymbol{v}^{k-1}\right),$$

where T is as defined in (16).

It can be directly observed that the above iteration scheme is the same as (23) with $\theta = 0$. Moreover, in the above ADMM scheme, the matrix $\bar{\mathsf{M}}_2$ is zero, which means that the information $\boldsymbol{v}^{k-1}$ is not used for updating $\boldsymbol{v}^{k+1}$. That is, ADMM is a one-step iteration method, while TFP^{2}A is a two-step iteration method by choosing the parameter θ appropriately. In general, the more information is used in each iteration, the faster the algorithm converges. We shall further confirm the advantages of TFP^{2}A through presenting several numerical results on two real-world benchmark data sets.

The first data set is “Housing” with 506 instances and each instance has 13 features. We use 300 instances as training data and the other 206 as testing data.

The second data set is "Mg" with 1385 instances and each instance has 6 features. We set 1000 instances as training data and the other 385 as testing data. We use the same Gaussian kernel and the same regularized parameters C, ϵ, and $\delta_i, i \in \mathbb{N}_m$ for both algorithms. The stopping criterion is set to be the relative error between the successive iterations less than a given tolerance, which we set as 10^{-7} in this experiment. In each algorithm, the parameters are tuned to approximately achieve the best prediction performance. We present the comparisons of MSE on testing data, the iteration numbers, and computational time for training of TFP^{2}A and ADMM in Table 2.

TABLE 2. Comparison of ADMM and TFP^{2}A in MSE, iteration numbers and training time. For "Housing" and "Mg", the parameter θ of TFP^{2}A is set as 1.3 and 1.6, respectively.

	Housing			Mg		
	MSE	iteration	time	MSE	iteration	time
ADMM	50.80	2059	1.60s	0.04	714	9.94s
TFP^{2}A	50.74	648	0.47s	0.04	238	2.61s

We remark that both algorithms have similar MSE since they are essentially solving the same model. However, TFP^{2}A requires a much shorter training time and less iterations than ADMM in both data sets.

6. Conclusions

We introduce the group lasso regularized SVR model and develop a novel two-step fixed-point proximity algorithm to solve it. We establish the convergence result of the proposed two-step fixed-point proximity algorithm. We perform numerical experiments on both synthetic data sets and real-world benchmark data sets to test the proposed model and algorithm. The numerical results demonstrate that the proposed GL-SVR performs better than the standard ℓ^1-SVR when the underlying data set has the group sparse structure, and the proposed algorithm is more computationally efficient than ADMM on the two real-world benchmark data sets.

Acknowledgments

This research is supported in part by the Special Project on High-performance Computing under the National Key R&D Program (No. 2016YFB0200602), by the Natural Science Foundation of China under grants 11471013, 91530117 and 11371333, and by the US National Science Foundation under grant DMS-1522332 and DMS-1521661.

References

[1] A. Argyriou, C. A. Micchelli, M. Pontil, L. Shen, and Y. Xu, Efficient first order methods for linear composite regularizers, arXiv preprint arXiv:1104.1436, (2011).

[2] D. Basak, S. Pal, and D. C. Patranabis, Support vector regression, Neural Information Processing-Letters and Reviews, 11 (2007), pp. 203–224.

[3] H. H. Bauschke and P. L. Combettes, Convex Analysis and Monotone Operator Theory in Hilbert Spaces, Springer, New York, 2011.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik, A training algorithm for optimal margin classifiers, in Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, New York, USA, 1992, ACM, pp. 144–152.

[5] C. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, Inverse Problems, 20 (2004).

[6] E. Byvatov and G. Schneider, Support vector machine applications in bioinformatics., Applied Bioinformatics, 2 (2002), pp. 67–77.

[7] C. Chang and C. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, 2 (2011), pp. 27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] O. Chapelle, P. Haffner, and V. N. Vapnik, Support vector machines for histogram-based image classification, IEEE Transactions on Neural Networks, 10 (1999), pp. 1055–1064.

[9] O. Chapelle and S. S. Keerthi, Multi-class feature selection with support vector machines, in Proceedings of the American Statistical Association, 2008.

[10] S. Chatterjee, A. Banerjee, S. Chatterjee, and A. R. Ganguly, Sparse group lasso for regression on land climate variables, in 2011 IEEE 11th International Conference on Data Mining Workshops, 2011.

[11] X. Chen, Smoothing methods for nonsmooth, nonconvex minimization, Mathematical Programming, 134 (2012), pp. 71–99.

[12] P. L. Combettes and V. R. Wajs, Signal recovery by proximal forward-backward splitting, Multiscale Modeling & Simulation, 4 (2005), pp. 1168–1200.

[13] C. Cortes and V. Vapnik, Support-vector networks, Machine Learning, 20 (1995), pp. 273–297.

[14] J. Friedman, T. Hastie, and R. Tibshirani, A note on the group lasso and a sparse group lasso, arXiv preprint arXiv:1001.0736, (2010).

[15] S. R. Gunn et al., Support vector machines for classification and regression, ISIS Technical Report, 14 (1998).

[16] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning, 46 (2002), pp. 389–422.

[17] T. Hastie, R. Tibshirani, and M. Wainwright, Statistical Learning with Sparsity: The Lasso and Generalizations, CRC Press, 2015.

[18] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al., A practical guide to support vector classification, (2003).

[19] L. Jacob, G. Obozinski, and J.-P. Vert, Group lasso with overlap and graph lasso, in Proceedings of the 26th Annual International Conference on Machine Learning, ACM, 2009, pp. 433–440.

[20] Y. Koshiba and S. Abe, Comparison of l1 and l2 support vector machines, in Proceedings of the International Joint Conference on Neural Networks, vol. 3, IEEE, 2003, pp. 2054–2059.

[21] Q. Li, L. Shen, Y. Xu, and N. Zhang, Multi-step proximity algorithms for solving a class of convex optimization problems, Advances in Computational Mathematics, 41 (2014), pp. 387–422.

[22] Q. Li, Y. Xu, and N. Zhang, Two-step fixed-point proximity algorithms for multi-block separable convex problems, Journal of Scientific Computing, (2016), pp. 1–25.

[23] Z. Li, G. Song, and Y. Xu, Fixed-point proximity algorithms for solving sparse machine learning models, Preprint, (2017).

[24] Z. Li, Q. Ye, and Y. Xu, Sparse support vector machines in reproducing kernel banach spaces, Invited paper in a book to be published in Springer, Submitted, (2016).

[25] S. Ma, X. Song, and J. Huang, Supervised group lasso with applications to microarray data analysis, BMC Bioinformatics, 8 (2007).

[26] L. Meier, S. Van De Geer, and P. Bühlmann, The group lasso for logistic regression, Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70 (2008), pp. 53–71.

[27] C. A. Micchelli, L. Shen, and Y. Xu, Proximity algorithms for image models: denoising, Inverse Problems, 27 (2011).

[28] C. A. Micchelli, L. Shen, Y. Xu, and X. Zeng, Proximity algorithms for the l1/tv image denoising model, Advances in Computational Mathematics, 38 (2013), pp. 401–426.

[29] E. Osuna, R. Freund, and F. Girosi, Training support vector machines: An application to face detection, in Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97), Washington, D.C., USA, 1997, IEEE Computer Society, pp. 130–136.

[30] N. Parikh and S. Boyd, Proximal algorithms, Foundations and Trends in Optimization, 1 (2014), pp. 127–239.

[31] B. Schölkopf and A. J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond, MIT Press, Cambridge, Mass, 2002.

[32] A. Smola and V. Vapnik, Support vector regression machines, Advances in Neural Information Processing Systems, 9 (1997), pp. 155–161.

[33] A. J. Smola and B. Schölkopf, A tutorial on support vector regression, Statistics and Computing, 14 (2004), pp. 199–222.

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

[35] S. Tong and D. Koller, Support vector machine active learning with applications to text classification, Journal of Machine Learning Research, 2 (2002), pp. 45–66.

[36] V. Vapnik, Statistical Learning Theorey, Wiley, New York, 1998.

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

[38] J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani, 1-norm support vector machines, Advances in Neural Information Processing Systems, 16 (2004), pp. 49–56.

Guangdong Province Key Lab of Computational Science, School of Mathematics, Sun Yat-sen University, Guangzhou 510275, P. R. China
E-mail: `li_zheng_2011@163.com`

Department of Mathematics, Clarkson University, Potsdam, New York 13699, USA
E-mail: `gsong@clarkson.edu`

School of Data and Computer Science, Guangdong Province Key Lab of Computational Science, Sun Yat-sen University, Guangzhou 510275, P. R. China, and Department of Mathematics and Statistics, Old Dominion University, Norfolk, Virginia, USA. All correspondence should be sent to this author.
E-mail: `y1xu@odu.edu`