LEAST ABSOLUTE DEVIATIONS LEARNING OF MULTIPLE TASKS

WEI XUE$^{1,2}$, WENSHENG ZHANG$^{2,3}$ AND GAOHANG YU$^4$

$^1$School of Computer Science and Technology, Anhui University of Technology
Maanshan 243032, China
$^2$School of Computer Science and Engineering, Nanjing University of Science and Technology
Nanjing 210094, China
$^3$Institute of Automation, Chinese Academy of Sciences
Beijing 100190, China
$^4$School of Mathematics and Computer Sciences, Gannan Normal University
Ganzhou 341000, China

(Communicated by Cheng-Chew Lim)

Abstract. In this paper, we propose a new multitask feature selection model based on least absolute deviations. However, due to the inherent nonsmoothness of $l_1$ norm, optimizing this model is challenging. To tackle this problem efficiently, we introduce an alternating iterative optimization algorithm. Moreover, under some mild conditions, its global convergence result could be established. Experimental results and comparison with the state-of-the-art algorithm SLEP show the efficiency and effectiveness of the proposed approach in solving multitask learning problems.

1. Introduction. Multitask feature selection (MTFS), which is aimed to learn explanatory features across multiple related tasks, has been successfully applied in various applications including character recognition [18], classification [21], medical diagnosis [23], and object tracking [2]. One key assumption behind many MTFS models is that all tasks are interrelated to each other. In this paper, we mainly consider regression problem, and in the problem setup, we assume that there are $k$ regression problems or “tasks” with all data coming from the same space. For each task, there are $m_j$ points. Hence, it consists of a dataset of $D = \bigcup_{j=1}^{k} D_j$, where $D_j = \{(x_j^i, y_j^i)\}_{i=1}^{m_j}$ are sampled from an underlying distribution $P_j$. $x_j^i \in \mathbb{R}^n$ denotes the $i$-th training sample for the $j$-th task, $y_j^i \in \mathbb{R}$ denotes the corresponding response, the superscript $i$ indexes the independent and identically distributed (i.i.d.) observations for each task, $m_j$ is the number of samples for the $j$-th task, and the total number of training samples is denoted by $m = \sum_{j=1}^{k} m_j$. The goal of MTFS is to learn $k$ decision functions $f_j|_{j=1}^{k}$ such that $f_j(x_j^i)$ approximates $y_j^i$. Typically, in multitask learning models, the decision function $f_j$ for the $j$-th task is assumed as a hyperplane parameterized by the model weight vector.
\( \mathbf{w}_j \in \mathbb{R}^n \). The objective of MTFS models is to learn a weight matrix \( W \in \mathbb{R}^{n \times k} \). To make the notation uncluttered, we express the weight matrix \( W \) into columnwise and rowwise vectors, i.e., \( W = [\mathbf{w}_1, \ldots, \mathbf{w}_k] = [\mathbf{w}^1; \ldots; \mathbf{w}^n] \). For convenience, let \( X_j = [\mathbf{x}^1_j, \ldots, \mathbf{x}^m_j]^T \in \mathbb{R}^{m_j \times n} \) denote the sample matrix for the \( j \)-th task, \( \mathbf{y}_j = [y^1_j, \ldots, y^m_j]^T \in \mathbb{R}^{m_j} \), and suppose that \( L_j(X_j \mathbf{w}_j, \mathbf{y}_j) \) be a loss function on the sample \( (X_j, \mathbf{y}_j) \) for task \( j \), a standard method to finding a sparse \( W \) is to solve the following \( l_1 \) minimization problem:

\[
\min_{W \in \mathbb{R}^{n \times k}} \frac{1}{k} \sum_{j=1}^{k} \frac{1}{m_j} \sum_{i=1}^{m_j} L_j(X_j \mathbf{w}_j, \mathbf{y}_j) + \mu \sum_{j=1}^{k} \| \mathbf{w}_j \|_1,
\]

where \( \mu > 0 \) is the regularization parameter used to balance the loss term and the regularization term.

Solving (1) leads to individual sparsity patterns for each \( \mathbf{w}_j \). Many methods have been proposed to select features globally with the use of variants of \( l_1 \) regularization term, or more specifically, by imposing a mixed \( l_2,1 \) matrix norm, \( \| W \|_{2,1} \triangleq \sum_{j=1}^{k} \| \mathbf{w}_j \|_2 \), and \( l_{\infty,1} \) matrix norm, \( \| W \|_{\infty,1} \triangleq \sum_{j=1}^{k} \| \mathbf{w}_j \|_{\infty} = \sum_{j=1}^{k} \max_{i} | w_{i,j} | \). As argued by [26], the advantages of the two matrix norms are that they can not only gain benefit from the \( l_1 \) norm which can promote sparse solutions, but also achieve group sparsity through the \( l_p \) norm. The \( l_{2,1} \) norm is essentially the sum of the \( l_2 \) norms of the rows, and the \( l_{\infty,1} \) norm penalizes the sum of maximum absolute values of each row. One appealing property of the two matrix norms is that they encourage multiple predictors from different tasks to share similar parameter sparsity patterns and discover solutions where only a few features are nonzero. One commonly used function for \( L_j(X_j \mathbf{w}_j, \mathbf{y}_j) \) is the squared loss, that is, \( \| X_j \mathbf{w}_j - \mathbf{y}_j \|_2^2 \), which can be viewed as a regression with Gaussian noise [14]. In this paper, we focus on the regression problem in the context of MTFS. Instead of choosing the conventional squared loss, we consider a nonsmooth loss function and present our MTFS model from the perspective of probability. The key contributions of this paper are highlighted as follows.

- We propose a new MTFS model within probabilistic framework and develop an iterative algorithm to solve it.
- We theoretically provide a convergence result of the developed algorithm.
- We conduct experiments on both synthetic data and real data to show the performance of the proposed approach.

The rest of this paper is organized as follows. In Section 2, we introduce the multitask learning formulation and present an optimization algorithm to solve the proposed model. In Section 3, we provide theoretical analysis of the algorithm. Experimental comparison and results are reported in Section 4. Finally, we conclude this paper with future work in Section 5.

2. Model formulation and algorithm. In this section, we first introduce our formulation for multitask learning and then present the solving process.

2.1. Model formulation. Given \( \mathbf{x}_j \in \mathbb{R}^n \), suppose that the corresponding output \( y_j \in \mathbb{R} \) for task \( j \) has a Laplacian distribution with a location parameter \( (\mathbf{w}_j^T \mathbf{x}_j) \) and a scale parameter \( \sigma_j > 0 \); that is to say, its probability density function is in the form:

\[
p(\mathbf{y}_j | \mathbf{w}_j, \mathbf{x}_j, \sigma_j) = \frac{1}{2\sigma_j} \exp\left(-\frac{|y_j - \mathbf{w}_j^T \mathbf{x}_j|}{\sigma_j}\right),
\]

(2)
Denote $\sigma = [\sigma_1, \ldots, \sigma_k] \in \mathbb{R}^k$, and assume that the data $\{A, y\}$ is drawn i.i.d. according to the distribution in (2), then the likelihood function can be written as

$$p(y|W, A, \sigma) = \prod_{k} \prod_{m} p(y^i_j|w_j, x^i_j, \sigma_j).$$

To capture the task relatedness, we impose the exponential prior on the $i$-th row of $W$, i.e.,

$$p(w^i|\delta^i) \propto \exp(-\|w^i\|_2^2\delta^i), \quad i = 1, 2, \ldots, n,$$  \hspace{2cm} (3)

where $\delta^i > 0$ is the so-called rate parameter. Denote $\delta = [\delta^1, \ldots, \delta^n] \in \mathbb{R}^n$ and assume that $w^1, \ldots, w^n$ are drawn i.i.d. according to (3), then we can express the prior on $W$ as $p(W|\delta) = \prod_{i=1}^n p(w^i|\delta^i)$. It follows that the posterior distribution of $W$ is $p(W|A, y, \sigma, \delta) \propto p(y|W, A, \sigma)p(W|\delta)$.

With the above likelihood and prior, we can obtain a maximum posterior solution of $W$ by solving the following optimization problem:

$$\min_{W \in \mathbb{R}^{n \times k}} \sum_{j=1}^k \frac{1}{\sigma_j} \|X_jw_j - y_j\|_1 + \sum_{i=1}^n \delta^i \|w^i\|_2.$$ \hspace{2cm} (4)

Clearly, the solution of the first term in (4) can be viewed as a Least Absolute Deviations (LAD) solution. Thus, we name (4) as LAD multitask feature selection model. For simplicity, we assume that $\sigma_j = \sigma$ for all $j$ and $\delta^i = \delta$ for all $i$. Let $\mu = \sigma \delta$, the proposed MTFS model based on LAD is given as follows:

$$\min_{W \in \mathbb{R}^{n \times k}} \sum_{j=1}^k \|X_jw_j - y_j\|_1 + \mu \sum_{i=1}^n \|w^i\|_2.$$ \hspace{2cm} (5)

2.2. Algorithm. In the following, we show how to solve the LAD multitask learning model (5). Actually, (5) can be rewritten as a matrix factorization problem with a sparsity penalty:

$$\min_{W \in \mathbb{R}^{n \times k}} \|X(W) - y\|_1 + \mu \|W\|_{2,1},$$ \hspace{2cm} (6)

where $X : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^m$ is a map defined by matric-vector multiplication based on each task, i.e., $X(W) = [X_1w_1; \ldots; X_kw_k] \in \mathbb{R}^m$, $y = [y_1; \ldots; y_k] \in \mathbb{R}^m$. By introducing an artificial variable $r \in \mathbb{R}^m$, (6) can be rewritten as

$$\min_{W \in \mathbb{R}^{n \times k}} \left\{ \|r\|_1 + \mu \|W\|_{2,1} : X(W) - y = r \right\},$$ \hspace{2cm} (7)

which is a convex optimization problem and can be solved by many methods. In this paper, we adopt an alternating direction method (ADM) that minimizes the following augmented Lagrange function:

$$\mathcal{L}(W, r, \lambda) = \|r\|_1 + \mu \|W\|_{2,1} - \lambda^T[X(W) - y - r] + \frac{\beta}{2} \|X(W) - y - r\|^2_2,$$ \hspace{2cm} (8)

where $\lambda \in \mathbb{R}^m$ is the Lagrange multiplier, $\beta > 0$ is the penalty parameter of the linear constraint in (7). The basic idea of ADM can date back to the work of Gabay and Mercier [8], which has been applied to many different fields such as image restoration [20, 28], quadratic programming [16], online learning [24], and background-foreground extraction [27]. Given $(r_k, \lambda_k)$, ADM generates the next iteration via

$$\begin{cases} W_{k+1} \leftarrow \arg \min_{W \in \mathbb{R}^{n \times k}} \mathcal{L}(W, r_k, \lambda_k), \\ r_{k+1} \leftarrow \arg \min_{r \in \mathbb{R}^m} \mathcal{L}(W_{k+1}, r, \lambda_k), \\ \lambda_{k+1} \leftarrow \lambda_k - \beta[X(W_{k+1}) - y - r_{k+1}]. \end{cases}$$ \hspace{2cm} (9)
We can see from (9) that at each iteration the main computation of ADM is solving two subproblems for \( W \) and \( r \).

Firstly, for \( r = r_k \) and \( \lambda = \lambda_k \), the minimizer \( W_{k+1} \) of (8) with respect to \( W \) is given by

\[
W_{k+1} \leftarrow \arg \min_{W \in \mathbb{R}^{n \times k}} \mathcal{L}(W, r_k, \lambda_k),
\]

\[
= \arg \min_{W \in \mathbb{R}^{n \times k}} \left\{ \mu \|W\|_2,1 - \lambda_k^r [\mathcal{X}(W) - y - r_k] + \frac{\beta}{2} \|\mathcal{X}(W) - y - r_k\|_2^2 \right\}, \tag{10}
\]

\[
= \arg \min_{W \in \mathbb{R}^{n \times k}} \left\{ \mu \|W\|_2,1 + \frac{\beta}{2} \|\mathcal{X}(W) - y - r_k - \lambda_k\|_2^2 \right\}.
\]

Instead of solving (10) exactly, we approximate it by

\[
\min_{W \in \mathbb{R}^{n \times k}} \mu \|W\|_2,1 + \frac{\beta}{2} \left( \left( G_k, W - W_k \right) + \frac{1}{2\tau} \|W - W_k\|_F^2 \right), \tag{11}
\]

where \( G_k = \mathcal{X}^r [\mathcal{X}(W_k) - y - r_k - \lambda_k/\beta] \) is the gradient of the quadratic term \( \frac{1}{2} \|\mathcal{X}(W) - y - r_k - \lambda_k/\beta\|_2^2 \) in (10) at \( W = W_k \), and \( \tau > 0 \) is a parameter\(^2\). The penalty term \( \|W - W_k\|_F^2 \) prevents \( W \) from walking far away from \( W_k \). It is not hard to see that (11) is equivalent to

\[
\min_{W \in \mathbb{R}^{n \times k}} \mu \|W\|_2,1 + \frac{\beta}{2\tau} \|W - (W_k - \tau G_k)\|_F^2. \tag{12}
\]

Let \( V = W_k - \tau G_k \) and \( v^i \) be the \( i \)-th row of \( V \), then the solution of (12) is in the form:

\[
(W_{k+1} =) \ W = \arg \min_{w^1, \ldots, w^n} \sum_{i=1}^n \left( \mu \|w^i\|_2 + \frac{\beta}{2\tau} \|w^i - v^i\|_2^2 \right), \tag{13}
\]

which means that the optimization problem (12) can be decomposed into \( n \) separate subproblems, namely,

\[
\min_{w^i \in \mathbb{R}^k} \mu \|w^i\|_2 + \frac{\beta}{2\tau} \|w^i - v^i\|_2^2, \quad i = 1, \ldots, n.
\]

According to [13], the closed-form solutions of (13) can be given explicitly as

\[
\hat{w}^i = v^i \left( 1 - \frac{\mu \tau}{\beta \|v^i\|_2} \right)_+, \quad i = 1, \ldots, n, \tag{14}
\]

where \((\cdot)_+ = \max(\cdot, 0)\). All the operations in (14) are performed componentwise. Therefore, the solution of (12) is

\[
W_{k+1} = \begin{bmatrix}
(W_k - \tau G_k)^1 \left( 1 - \frac{\mu \tau}{\beta \|W_k - \tau G_k\|_2} \right)_+ \\
\cdots \\
(W_k - \tau G_k)^n \left( 1 - \frac{\mu \tau}{\beta \|W_k - \tau G_k\|_2} \right)_+
\end{bmatrix}. \tag{15}
\]

\(^2\)Here, \( \mathcal{X}^r \) is a map operation defined by matric-vector multiplication based on each task; see [22] for example.
Secondly, given \((W_{k+1}, \lambda_k)\), minimizing (8) with respect to \(r\) is equivalent to
\[
\begin{align*}
\mathbf{r}_{k+1} & \leftarrow \arg \min_{r \in \mathbb{R}^m} \mathcal{L}(W_{k+1}, r, \lambda_k) \\
& = \arg \min_{r \in \mathbb{R}^m} \left\{ \|r\|_1 - \lambda_k^T [X(W_{k+1}) - y - r] + \frac{\beta}{2} \|X(W_{k+1}) - y - r\|_2^2 \right\} \\
& = \arg \min_{r \in \mathbb{R}^m} \left\{ \frac{\beta}{2} \|r - [X(W_{k+1}) - y - \frac{\lambda_k}{\beta}]\|_2^2 + \|r\|_1 \right\}.
\end{align*}
\] (16)

By using the soft thresholding operator \(S^3\), the solution of (16) can be written as
\[
\begin{align*}
\mathbf{r}_{k+1} &= S^3 \left( \lambda_k^T (X(W_{k+1}) - y - \frac{\lambda_k}{\beta}) \right). \\
&= S^3 \left( \mathbf{r} \right) \quad \text{for } \mathbf{r} \neq 0.
\end{align*}
\] (17)

Finally, the multiplier \(\lambda\) is updated as
\[
\lambda_{k+1} = \lambda_k - \beta [X(W_{k+1}) - y - \mathbf{r}_{k+1}] \\
= \lambda_k - \beta \mathbf{r}.
\] (18)

This is an inexact ADM since the \(W\)-subproblem and the \(r\)-subproblem are solved approximately, and we name this method LADL21 and outline it in Algorithm 1. The iteration procedure is repeated until the algorithm converges.

**Algorithm 1:** LADL21 – An efficient iterative algorithm to solve the optimization problem in (5).

1. **Input:** \(r_0, \lambda_0, \mu, \beta,\) and \(\tau\).
2. **while** “not converged”, **do**
   1. Compute \(W_{k+1}\) according to (15) for given \(r_k\) and \(\lambda_k\).
   2. Compute \(r_{k+1}\) according to (17) for given \(W_{k+1}\) and \(\lambda_k\).
   3. Update the multipliers \(\lambda_{k+1}\) according to (18).
3. **end while**

3. **Convergence analysis.** This section is devoted to establishing the convergence property of LADL21. We first show that the proposed LADL21 algorithm belongs to the framework of He et al. [11], which is designed to solve structured variational inequality (SVI) problems, and then the convergence result of LADL21 follows directly.

3.1. **Preparations.** We begin with some preparations for analyzing the convergence property. In order to show the global convergence of LADL21 clearly, we first turn to consider the SVI problem: finding a vector \(u_* \in \Omega\) such that
\[
(u - u_*)^T F(u_*) \geq 0, \forall u \in \Omega,
\] (19)

or equivalently, \(S^3_\mathbf{u}(v) = (v - u)_+ - (-v - u)_+\). Another formula, which shows that the soft thresholding operator is a shrinkage operator (i.e., moves a point toward zero), is
\[
S^3_\mathbf{u}(v) = (1 - \frac{v}{|v|})_+ v
\] for \(v \neq 0\).
where $\Omega$ is a nonempty closed convex subset of $\mathbb{R}^{a+b}$, $F$ is a mapping from $\mathbb{R}^{a+b}$ to itself,

$$u = \begin{pmatrix} s \\ t \end{pmatrix}, \quad F(u) = \begin{pmatrix} f(s) \\ g(t) \end{pmatrix}, \quad \Omega = \left\{ (s, t) : s \in S, t \in T, Ms + Nt = c \right\},$$

where $S$ and $T$ are given nonempty closed convex subsets of $\mathbb{R}^a$ and $\mathbb{R}^b$, respectively, $M \in \mathbb{R}^{l \times a}$ and $N \in \mathbb{R}^{l \times b}$ are given matrices, $c \in \mathbb{R}^l$ is a given vector, $f : S \to \mathbb{R}^a$ and $g : T \to \mathbb{R}^b$ are given monotone operators. By attaching a Lagrange multiplier vector $\lambda \in \mathbb{R}^l$ to the linear constraint $Ms + Nt = c$, one can obtain an equivalent form of (19):

$$z_\ast \in Z, (z - z_\ast)^T Q(z_\ast) \geq 0, \forall z \in Z,$$

(20)

where

$$z = \begin{pmatrix} s \\ t \\ \lambda \end{pmatrix}, \quad Q(z) = \begin{pmatrix} f(s) - M^T \lambda \\ g(t) - N^T \lambda \\ Ms + Nt - c \end{pmatrix}, \quad Z = S \times T \times \mathbb{R}^l.$$

To solve (20), the method proposed by He et al. [11] produces a new iteration $(s_{k+1}, t_{k+1}, \lambda_{k+1})$ via the following procedure given $(t_k, \lambda_k)$.

Firstly, $s_{k+1}$ is obtained by solving the following problem:

$$(s' - s)^T \left\{ f(s) - M^T [\lambda_k - H_k(Ms + Nt_k - c)] + R_k(s - s_k) \right\} \geq 0, \forall s' \in S. \quad (21)$$

Then, $t_{k+1}$ is produced by solving

$$(t' - t)^T \left\{ g(t) - N^T [\lambda_k - H_k(Ms_{k+1} + Nt - c)] + S_k(t - t_k) \right\} \geq 0, \forall t' \in T. \quad (22)$$

Finally, the multiplier is updated by

$$\lambda_{k+1} = \lambda_k - H_k(Ms_{k+1} + Nt_{k+1} - c). \quad (23)$$

Here $\{H_k\}$, $\{R_k\}$ and $\{S_k\}$ are sequences of both lower and upper bounded symmetric positive definite matrices. Under mild conditions, He et al. [11] established the convergence for their method.

3.2. Convergence result. Based on the analysis mentioned above, we now consider (9) in the SVI framework. For simplicity, let $\partial(\cdot)$ be the subgradient operator of a function. The optimal condition of problem (7) can be characterized by finding a vector $z_\ast = (W_\ast, r_\ast, \lambda_\ast) \in Z = \mathbb{R}^{n \times k} \times \mathbb{R}^m \times \mathbb{R}^m$ such that $\forall z' = (W', r', \lambda') \in Z$, then it holds that

$$\begin{cases} 
(W' - W_\ast, \partial(\mu\|W_\ast\|_{2,1}) - \lambda^* \lambda_\ast) \geq 0, \\
(r' - r_\ast)^T (\partial(\|r\|_1) + \lambda_\ast) \geq 0, \\
(\lambda' - \lambda_\ast)^T (\lambda(W_\ast) - y - r_\ast) \geq 0. 
\end{cases} \quad (24)$$

(24) is equivalent to

$$(z' - z_\ast)^T Q(z_\ast) \geq 0, \forall z' \in Z,$$

where

$$z = \begin{pmatrix} W \\ r \\ \lambda \end{pmatrix}, \quad Q(z) = \begin{pmatrix} \partial(\mu\|W\|_{2,1}) - \lambda^* \lambda \\ \partial(\|r\|_1) + \lambda \\ \lambda(W) - y - r \end{pmatrix}.$$

We consider the approximative $W-$subproblem in (12) for its optimal condition in SVI form:

$$\left\langle W' - W, \partial(\mu\|W\|_{2,1}) + \frac{2}{\tau} [W - (W_k - \tau G_k)] \right\rangle \geq 0,$$
or equivalently,
\[
\left\langle W' - W, \partial (\mu \| W \|_{2,1}) - X^*(\lambda_k - \beta [X(W) - y - r_k]) \right\rangle + \langle \beta \left( \frac{1}{\tau} I - X^* X \right), W - W_k \rangle \geq 0.
\]
Similarly, we have
\[
(r' - r)^T \left\{ \partial ([r]_1) - [\lambda_k - \beta (X(W_{k+1}) - y - r)] \right\} \geq 0.
\]
In the end, the multiplier is updated via
\[
\lambda_{k+1} = \lambda_k - \beta [X(W_{k+1}) - y - r_{k+1}].
\]
Comparing (25-27) with (21-23), one can see that our method can be seen as a special case of the framework of [11] provided that 
\[
H_k = \beta I, \quad R_k = \beta \left( \frac{1}{\tau} I - X^* X \right)
\]
and 
\[
S_k = \odot,
\]
where \(I\) is an identity matrix, \(\odot\) denotes a zero matrix. Hence, the main convergence property of Algorithm 1 can be stated as follows.

**Theorem 3.1.** Suppose that \(0 < \tau < 1 / \lambda_{\text{max}}(X^* X)\), then for any \(\beta > 0\), the sequence \(\{(W_k, r_k, \lambda_k)\}\) generated by Algorithm 1 from any starting point \((r_0, \lambda_0)\) converges to a solution of (7).

4. Experiments. In the following, we conduct experiments to demonstrate the performance of the proposed approach to solve MTFS problems. All experiments are implemented in Matlab. The compared algorithm is “SLEP” (short for “sparse learning with efficient projections”) [15]. SLEP is an open Matlab package which provides a set of learning algorithms for solving \(l_{2,1}\) regularized learning problems, whose functions are based on the work in [14]. In [14], the \(l_{2,1}\) regularized multi-task learning takes the form of
\[
\min_W \frac{1}{2} \sum_{j=1}^{k} \| X_j w_j - y_j \|_2^2 + \rho \sum_{i=1}^{n} \| w_i \|_2,
\]
where \(\rho > 0\) is the regularization parameter. Considering that the objective function in (28) is nonsmooth, the authors first transform (28) into the following equivalent constrained smooth convex optimization problem:
\[
\min_{W, u} \frac{1}{2} \sum_{j=1}^{k} \| X_j w_j - y_j \|_2^2 + \rho \sum_{i=1}^{n} u_i,
\]
where \(u = [u_1, \ldots, u_n]^T\) and \(\| w_i \|_2 \leq u_i, \forall i = 1, \ldots, n\), and then solve (29) via the Nesterov’s method. Parametric settings in SLEP will be specified when we discuss individual experiment.

4.1. Synthetic data. In this section, we use synthetic data to test the ability of our model and algorithm. We create the synthetic data by generating task parameters \(w_j\) from a Gaussian distribution with zero mean and covariance \(Cov\). The training and test datasets \(X_j\) are the Gaussian matrices whose elements are generated by Matlab function \(randn(m_j, n)\). The outputs are computed by
\[
y_j = X_j w_j + \omega,
\]
where $\varpi$ is zero-mean Gaussian noise with standard deviation 0.01. Let $W^*$ be the optimal solution, we use the relative error (RelErr) to measure the quality of $W^*$,

$$\text{RelErr} = \frac{\|W^* - W\|_F}{\|W\|_F}.$$ 

We terminate the two methods when the relative change (RelChg) between two consecutive iterations is less than a pre-set positive threshold $\epsilon$,

$$\text{RelChg} = \frac{\|W_{k+1} - W_k\|_F}{\|W_k\|_F} < \epsilon.$$ 

Firstly, let $Cov = diag\{1, 0.25, 0.1, 0.05, 0.01\}$ and $Cov = diag\{0.81, 0.64, 0.49, 0.36, 0.25, 0.16, 0.09, 0.04\}$, respectively, and to these we keep adding up to 20 irrelevant dimensions which are exactly zero. We set $k = 200$, $n = 10$, $m_j = 100$, $\forall j = 1, \ldots, k$, and $\epsilon = 1e^{-3}$. In SLEP method, we set $mFlag = 1$ and $lFlag = 1$, which means an adaptive line search is used. The other values of parameters are the same as the previous test. Figure 1 shows the decrease of relative errors as the number of iterations and CPU time increase. We can see from Figure 1 that the proposed method is faster than SLEP and obtains a better accuracy.

Secondly, considering that the number of samples, dimensions and tasks may also affect the performance of each method, we report the numerical results of
the compared algorithms with different numbers of samples, dimensions and tasks. In the test, we take \( \text{Cov} = \text{diag}\{1, 0.64, 0.49, 0.36, 0.25\} \), and Table 1 shows the simulation results. These results indicate that our method can obtain better quality solutions, which demonstrates the merit of the proposed approach.

Table 1. Comparison of the RelErr.

| (m,n,k)          | SLEP | LADL21 |
|------------------|------|--------|
| (10000,20,100)   | 0.0139 | 0.0025 |
| (20000,20,200)   | 0.0136 | 0.0017 |
| (30000,20,300)   | 0.0140 | 0.0015 |
| (40000,20,400)   | 0.0138 | 0.0016 |
| (10000,30,100)   | 0.0141 | 0.0016 |
| (20000,30,200)   | 0.0199 | 0.0016 |
| (30000,30,300)   | 0.0143 | 0.0017 |
| (40000,30,400)   | 0.0202 | 0.0018 |
| (10000,40,100)   | 0.0187 | 0.0018 |
| (20000,40,200)   | 0.0146 | 0.0018 |
| (30000,40,300)   | 0.0154 | 0.0020 |
| (40000,40,400)   | 0.0188 | 0.0021 |
| (10000,50,100)   | 0.0170 | 0.0020 |
| (20000,50,200)   | 0.0179 | 0.0022 |
| (30000,50,300)   | 0.0177 | 0.0022 |
| (40000,50,400)   | 0.0176 | 0.0024 |

4.2. School data. We now conduct experiment on the School data, which has been widely used in multitask learning [1, 3, 7, 14]. This data consists of the exam scores of 15362 students from 139 secondary schools in London from 1985 to 1987 with each sample containing 28 attributes. Here, each school is taken as “one task”. Hence, we have 139 regression tasks corresponding to predicting the student performance in each school. We randomly take 75% of each task’s data for training, and the rest for testing. We run each method with taking \( \mu = 1 \times 10^{-5} \) and 200 iterations, and examine the behavior of the training errors and testing errors when each method proceeds. The fidelity of training and testing data is measured by root mean squared error. Figure 2 shows the convergence behavior of the compared algorithms. These curves show that all the methods converge finally. From Figure 2, we can find that the errors obtained by LADL21 are slightly larger than SLEP at the beginning, while becoming smaller in the later iterations. In addition, Figure 2(b) and Figure 2(d) indicate that our method is faster than SLEP. Overall, we propose a valid method for multitask learning which is comparable to the state-of-the-art algorithm SLEP.

5. Conclusion. In this paper, we proposed a new multitask feature selection model and presented a solving algorithm for it. We derived closed-form solution to update the weight matrix, which ensures that the developed algorithm can work well. Numerical results illustrated that the proposed approach is effective and promising in solving multitask learning problems.

Since the proposed LADL21 algorithm is a batch learning algorithm, it has to retrain from scratch when a new sample comes in. So a future work associated with this work is to generalize LADL21 to an online situation, which may have less flops than that of batch learning. Secondly, the proposed MTFS model is based on a convex formulation, it is promising to design a nonconvex formulation for multitask learning via some new regularization functions; see [9] for example. Thirdly, the
hyper-parameter $\mu$ in the LADL21 algorithm is predefined, we can find the optimal solution of $\mu$ via a systematic search technique, such as bilevel optimization [5].

Acknowledgments. The authors would like to thank the associate editor and the anonymous reviewers for their constructive comments and suggestions which improved the paper greatly. The work of W. Xue and W. Zhang is supported by NSFC (Grant No. 61305018, 61432008, 61472423 and 61532006) and the Major Technologies R & D Special Program of Anhui, China (Grant No. 16030901060). The work of G. Yu is supported by NSFC (Grant No. 61262026, 11661007), NCET Programm of the Ministry of Education (NCET 13-0738), JGZX programm of Jiangxi Province (20112BCB23027), Natural Science Foundation of Jiangxi Province (20132BAB201-026), Science and Technology Programm of Jiangxi Education Committee (LDJH12-088).

REFERENCES

[1] A. Argyriou, T. Evgeniou and M. Pontil, Convex multi-task feature learning, *Mach. Learn.*, 73 (2008), 243–272.
[2] Y. Bai and M. Tang, Object tracking via robust multitask sparse representation, *IEEE Signal Process. Lett.*, 21 (2014), 909–913.
[3] B. Bakker and T. Heskes, Task clustering and gating for Bayesian multitask learning, *J. Mach. Learn. Res.*, 4 (2003), 83–99.
[4] S. Ben-David and R. Schuller, Exploiting task relatedness for multiple task learning, in Proc. Int. Conf. Learn. Theory, 2777 (2003), 567–580.
[5] K. P. Bennett, J. Hu, X. Ji, G. Kunapuli and J.-S. Pang, Model selection via bilevel optimization, in Proc. IEEE Int. Joint Conf. Neural Netw., (2006), 1922–1929.
[6] X. Chen, W. Pan, J. T. Kwok and J. G. Carbonell, Accelerated gradient method for multi-task sparse learning problem, in Proc. IEEE Int. Conf. Data Min., (2009), 746–751.
[7] T. Evgeniou, C. A. Micchelli and M. Pontil, Learning multiple tasks with kernel methods, *J. Mach. Learn. Res.*, 6 (2005), 615–637.

[8] D. Gabay and B. Mercier, A dual algorithm for the solution of nonlinear variational problems via finite-element approximations, *Comp. Math. Appl.*, 2 (1976), 17–40.

[9] P. Gong, J. Ye and C. Zhang, Multi-stage multi-task feature learning, in *Adv. Neural Inf. Process. Syst.*, (2012), 1988–1996.

[10] P. Gong, J. Ye and C. Zhang, Robust multi-task feature learning, *J. Mach. Learn. Res.*, 14 (2013), 2979–3010.

[11] B. He, L. Liao, D. Han and H. Yang, A new inexact alternating directions method for monotone variational inequalities, *Math. Program.*, 92 (2002), 103–118.

[12] Y. Hu, Z. Wei and G. Yuan, Inexact accelerated proximal gradient algorithms for matrix $l_{2,1}$-norm minimization problem in multi-task feature learning, *Stat., Optim. Inf. Comput.*, 2 (2014), 352–367.

[13] M. Kowalski, M. Szafranski and L. Ralaivola, Multiple indefinite kernel learning with mixed norm regularization, in *Proc. Int. Conf. Mach. Learn.*, (2009), 545–552.

[14] J. Liu, S. Ji and J. Ye, Multi-task feature learning via efficient $l_{2,1}$-norm minimization, in *Proc. Uncertainty Artif. Intell.*, (2009), 339–348.

[15] J. Liu, S. Ji and J. Ye, SLEP: Sparse learning with efficient projections, Available from: http://yelab.net/software/SLEP.

[16] Y. Lu, Y.-E. Ge and L-W. Zhang, An alternating direction method for solving a class of inverse semidefinite quadratic programming problems, *J. Ind. Manag. Optim.*, 12 (2016), 317–336.

[17] F. Nie, H. Huang, X. Cai and C. Ding, Efficient and robust feature selection via joint $l_{2,1}$-norms minimization, in *Adv. Neural Inf. Process. Syst.*, (2010), 1813–1821.

[18] G. Obozinski, B. Taskar and M. I. Jordan, Joint covariate selection and joint subspace selection for multiple classification problems, *Statist. Comput.*, 20 (2010), 231–252.

[19] A. Quattoni, X. Carreras, M. Collins and T. Darrell, An efficient projection for $l_{1,\infty}$ regularization, in *Proc. Int. Conf. Mach. Learn.*, (2009), 857–864.

[20] Z. Shen, Z. Geng and J. Yang, Image reconstruction from incomplete convolution data via total variation regularization, *Stat., Optim. Inf. Comput.*, 3 (2015), 1–14.

[21] S. Xiang, F. Nie, G. Meng, C. Fan and C. Zhang, Discriminative least squares regression for multiclass classification and feature selection, *IEEE Trans. Neural Netw. Learn. Syst.*, 23 (2012), 1738–1754.

[22] Y. Xiao, S.-Y. Wu and B. He, A proximal alternating direction method for $l_{2,1}$-norm least squares problem in multi-task feature learning, *J. Ind. Manag. Optim.*, 8 (2012), 1057–1069.

[23] T. Xiong, J. Bi, B. Rao and V. Cherkassky, Probabilistic joint feature selection for multi-task learning, in *Proc. SIAM Int. Conf. Data Min.*, (2007), 332–342.

[24] W. Xue and W. Zhang, Learning a coupled linearized method in online setting, *IEEE Trans. Neural Netw. Learn. Syst.*, 28 (2017), 438–450.

[25] W. Xue and W. Zhang, Online weighted multi-task feature selection, in *Proc. Int. Conf. Neural Inf. Process.*, (2016), 195–203.

[26] H. Yang, M. R. Lyu and I. King, Efficient online learning for multitask feature selection, *ACM Trans. Knowl. Discov. Data*, 7 (2013), 1–27.

[27] L. Yang, T. K. Pong and X. Chen, Alternating direction method of multipliers for a class of nonconvex and nonsmooth problems with applications to background/foreground extraction, *SIAM J. Imaging Sci.*, 10 (2017), 74–110, arXiv:1506.07029v4.

[28] G. Yu, W. Xue and Y. Zhou, A nonmonotone adaptive projected gradient method for primal-dual total variation image restoration, *Signal Process.*, 103 (2014), 242–249.

Received December 2015; 1st revision August 2016; final revision October 2016.

E-mail address: csxvue@ahut.edu.cn
E-mail address: wensheng.zhang@ia.ac.cn
E-mail address: maghyu@163.com