Learning Incoherent Sparse and Low-Rank Patterns from Multiple Tasks

Jianhui Chen
Arizona State University
Tempe, AZ 85287, U.S.A
jianhui.chen@asu.edu

Ji Liu
Arizona State University
Tempe, AZ 85287, U.S.A
ji.liu@asu.edu

Jieping Ye
Arizona State University
Tempe, AZ 85287, U.S.A
jieping.ye@asu.edu

ABSTRACT

We consider the problem of learning incoherent sparse and low-rank patterns from multiple tasks. Our approach is based on a linear multi-task learning formulation, in which the sparse and low-rank patterns are induced by a cardinality regularization term and a low-rank constraint, respectively. This formulation is non-convex; we convert it into its convex surrogate, which can be routinely solved via semidefinite programming for small-size problems. We propose to employ the general projected gradient scheme to efficiently solve such a convex surrogate; however, in the optimization formulation, the objective function is non-differentiable and the feasible domain is non-trivial. We present the procedures for computing the projected gradient and ensuring the global convergence of the projected gradient scheme. The computation of projected gradient involves a constrained optimization problem; we show that the optimal solution to such a problem can be obtained via solving an unconstrained optimization subproblem and an Euclidean projection subproblem. In addition, we present two projected gradient algorithms and discuss their rates of convergence. Experimental results on benchmark data sets demonstrate the effectiveness of the proposed multi-task learning formulation and the efficiency of the proposed projected gradient algorithms.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - Data Mining

General Terms

Algorithms

Keywords

Multi-task learning, sparse and low-rank patterns, trace norm

1. INTRODUCTION

In the past decade there has been a growing interest in the problem of multi-task learning (MTL) [13]. It has been applied successfully in many areas of data mining and machine learning [2, 3, 9, 10, 31, 38]. MTL aims to enhance the overall generalization performance of the resulting classifiers by learning multiple tasks simultaneously in contrast to single-task learning (STL) setting. A common assumption in MTL is that all tasks are intrinsically related to each other. Under such an assumption, the informative domain knowledge is allowed to be shared across the tasks, implying what is learned from one task is beneficial to another. This is particularly desirable when there are a number of related tasks but only a limited amount of training data is available for learning each task.

MTL has been investigated by many researchers from different perspectives. Hidden units of neural networks are shared among similar tasks [6, 13]; task relatedness are modeled using the common prior distribution in hierarchical Bayesian models [5, 29, 40, 41]; the parameters of Gaussian Process covariance are learned from multiple tasks [21]; kernel methods and regularization networks are extended to multi-task learning setting [16]; a convex formulation is developed for learning clustered tasks [19]; a shared low-rank structure is learned from multiple tasks [3, 15]. Recently, trace norm regularization has been introduced into the multi-task learning domain [1, 4, 20, 27, 28] to capture the task relationship via a shared low-rank structure of the model parameters, resulting in a tractable convex optimization problem [22].

In many real-world applications, the underlying predictive classifiers may lie in a hypothesis space of some low-rank structure [3], in which the multiple learning tasks can be coupled using a set of shared factors, i.e., the basis of a low-rank subspace [30]. For example, in natural scene categorization problems, images of different labels may share similar background of a low-rank structure; in collaborative filtering or recommender system, only a few factors contribute to an individual's tastes. On the other hand, multiple learning tasks may have sufficient differences and meanwhile the discriminative features for each task can be sparse. Thus learning an independent predictive classifier for each task and identifying the task-relevant discriminative features simultaneously may lead to improved performance and easily interpretable models.

In this paper, we consider the problem of learning incoherent sparse and low-rank patterns from multiple related tasks. We propose a linear multi-task learning formulation, in which the model parameter can be decomposed as a sparse component and a low-rank one. Specifically, we employ a cardinality regularization term to enforce the sparsity in the model parameter, identifying the essential discriminative feature for effective classification; meanwhile, we use a rank constraint to encourage the low-rank structure, capturing the underlying relationship among the tasks for improved generalization performance. The proposed multi-task learning formulation is non-convex and lead to an NP-hard optimization problem. We convert this formulation into its tightest convex surrogate,

which can be routinely solved via semi-definite programming. It is, however, not scalable to large scale data sets in practice. We propose to employ the general projected gradient scheme to solve the convex surrogate; however, in the optimization formulation, the objective function is non-differentiable and the feasible domain is non-trivial. We present the procedures for computing the projected gradient and ensuring the global convergence of the projected gradient scheme. The computation of projected gradient involves a constrained optimization problem; we show that the optimal solution to such a problem can be obtained via solving an unconstrained optimization subproblem and an Euclidean projection subproblem separately. In addition, we present two detailed algorithms based on the projected gradient scheme and discuss their rates of convergence. We conduct extensive experiments on real-world data sets. Our results demonstrate the effectiveness of the proposed multi-task learning formulation and also demonstrate the efficiency of the projected gradient algorithms.

The remainder of this paper is organized as follows: in Section 2 we propose the linear multi-task learning formulation; in Section 3 we present the general projected gradient scheme for solving the proposed multi-task learning formulation; in Section 4 we present efficient computational algorithms for solving the optimization problems involved in the iterative procedure of the projected gradient scheme; in Section 5 we present two detailed algorithms based on the projected gradient scheme and discuss their rates of convergence; we report the experimental results in Section 6 and the paper concludes in Section 7.

Notations For any matrix $A \in \mathbb{R}^{m \times n}$, let a_{ij} be the entry in the i-th row and j-th column of A; denote by $\|A\|_0$ the number of nonzero entries; let $\|A\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|$; let $\{\sigma_i(A)\}_{i=1}^{r}$ be the set of singular values in non-increasing order, where $r = \text{rank}(A)$; denote by $\|A\|_2 = \sigma_1(A)$ and $\|A\|_* = \sum_{i=1}^{r} \sigma_i(A)$ the operator norm and trace norm of A, respectively; let $\|A\|_\infty = \max_{i,j} |a_{ij}|$.

2. MULTI-TASK LEARNING FRAMEWORK

Assume that we are given m supervised (binary) learning tasks, where each of the learning tasks is associated with a predictor f_ℓ and a set of training data as $\{(x_i^\ell, y_i^\ell)\}_{i=1}^{n_\ell} \subset \mathbb{R}^d \times \{-1, +1\}$ ($\ell = 1, \cdots, m$). We focus on linear predictors as $f_\ell(x^\ell) = z_\ell^T x^\ell$, where $z_\ell \in \mathbb{R}^d$ is the weight vector for the ℓth learning task.

We assume that the m tasks are related using an incoherent rank-sparsity structure, that is, the transformation matrix can be decomposed as a sparse component and a low-rank component. Denote the transformation matrix by $Z = [z_1, \cdots, z_m] \in \mathbb{R}^{d \times m}$; Z is the summation of a sparse matrix $P = [p_1, \cdots, p_m] \in \mathbb{R}^{d \times m}$ and a low-rank matrix $Q = [q_1, \cdots, q_m] \in \mathbb{R}^{d \times m}$ given by

$$Z = P + Q, \tag{1}$$

as illustrated in Figure 1. The ℓ^0-norm (cardinality) [11], i.e., the number of non-zero entries, is commonly used to control the sparsity structure in the matrix; similarity, matrix rank [18] is used to encourage the low-rank structure. We propose a multi-task learning formulation with a cardinality regularization and a rank constraint given by

$$\min_{Z, P, Q \in \mathbb{R}^{d \times m}} \quad \sum_{\ell=1}^{m} \sum_{i=1}^{n_\ell} \mathcal{L}\left(z_\ell^T x_i^\ell, y_i^\ell\right) + \gamma \|P\|_0$$
$$\text{subject to} \quad Z = P + Q, \ \text{rank}(Q) \leq \tau, \tag{2}$$

where $\mathcal{L}(\cdot)$ denotes a smooth convex loss function, γ provides a trade-off between the sparse regularization term and the general
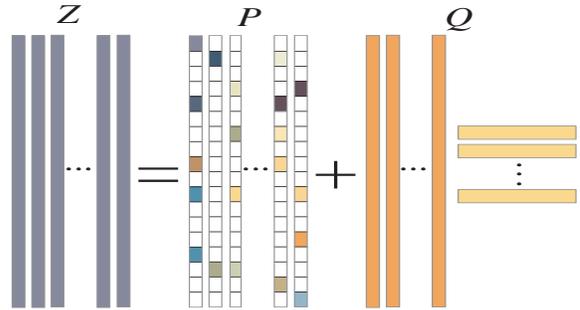

Figure 1: Illustration of the transformation matrix Z in Eq. (1), where P denotes the sparse component with the zero-value entries represented by white blocks, and Q denotes the low-rank component.

loss component, and τ explicitly specifies the upper bound of the matrix rank. Both γ and τ are non-negative and determined via cross-validation in our empirical studies.

The optimization problem in Eq. (2) is non-convex due to the non-convexity of the components $\|P\|_0$ and $\text{rank}(Q)$; in general solving such an optimization problem is NP-hard and no efficient solution is known. We consider a computationally tractable alternative by employing recently well-studied convex relaxation techniques [11].

Let the function $f : \mathbb{C} \to \mathbb{R}$, where $\mathbb{C} \subseteq \mathbb{R}^{d \times m}$. The convex envelope [11] of f on $\mathbb{C}$ is defined as the largest convex function g such that $g(\hat{Z}) \leq f(\hat{Z})$ for all $\hat{Z} \in \mathbb{C}$. The ℓ^1-norm has been known as the convex envelope of the ℓ^0-norm as [11]:

$$\|P\|_1 \leq \|P\|_0, \ \forall P \in \mathbb{C} = \{P \mid \|P\|_\infty \leq 1\}. \tag{3}$$

Similarly, trace norm (nuclear norm) has been shown as the convex envelop of the rank function as [17]:

$$\|Q\|_* \leq \text{rank}(Q), \ \forall Q \in \mathbb{C} = \{Q \mid \|Q\|_2 \leq 1\}. \tag{4}$$

Note that both the ℓ^1-norm and the trace-norm functions are convex but non-smooth, and they have been shown to be effective surrogates of the ℓ^0-norm and the matrix rank functions, respectively.

Based on the heuristic approximations in Eq. (3) and Eq. (4), we can replace the ℓ^0-norm with the ℓ^1-norm, and the rank function with the trace norm function in Eq. (2), respectively. Therefore, we can reformulate the multi-task learning formulation as:

$$\min_{Z, P, Q \in \mathbb{R}^{d \times m}} \quad \sum_{\ell=1}^{m} \sum_{i=1}^{n_\ell} \mathcal{L}\left(z_\ell^T x_i^\ell, y_i^\ell\right) + \gamma \|P\|_1$$
$$\text{subject to} \quad Z = P + Q, \ \|Q\|_* \leq \tau. \tag{5}$$

The optimization problem in Eq. (5) is the tightest convex relaxation of Eq. (2). Such a problem can be reformulated as a semi-definite program (SDP) [35], and then solved using many off-the-shelf optimization solvers such as SeDuMi [32]; however, SDP is computationally expensive and can only handle several hundreds of optimization variables. For simplicity, in this paper we assume that all of the m tasks share the same set of training data in Eq. (5), and the derivation below can be easily extended to the case where each learning task has a different set of training data.

Related Work: The formulation in Eq. (5) resembles the Alternating Structure Optimization algorithm (ASO) for multi-task learning proposed in [3]. However, they differ in several key aspects: (1) In ASO, the tasks are coupled using a shared low-dimensional structure induced by an orthonormal constraint, and the formulation in

ASO is non-convex and its convex counterpart cannot be easily obtained. Our formulation encourages the low-rank structure via a trace norm constraint and the resulting formulation is convex. (2) In ASO, in addition to a low-dimensional feature map shared by all tasks, the classifier for each task computes an independent high-dimensional feature map specific to each individual task, which is in general dense and does not lead to interpretable features. In our formulation, the classifier for each task constructs a sparse high-dimensional feature map for discriminative feature identification. (3) The alternating algorithm in ASO can only find a local solution with no known convergence rate. The proposed algorithm for solving the formulation in Eq. (5) finds a globally optimal solution and achieves the optimal convergence rate among all first-order methods. Note that recent works in [12, 14, 37] consider the problem of decomposing a given matrix into its underlying sparse component and low-rank component in a different setting: they study the theoretical condition under which such two components can be exactly recovered via convex optimization, i.e., the condition of guaranteeing to recover the sparse and low-rank components by minimizing a weighted combination of the trace norm and the ℓ^1-norm.

3. PROJECTED GRADIENT SCHEME

In this section, we propose to apply the general projected gradient scheme [11] to solve the constrained optimization problem in Eq. (5). Note that the projected gradient scheme belongs to the category of first-order methods and has demonstrated good scalability in many optimization problems [11, 25].

The objective function in Eq. (5) is non-smooth and the feasible domain is non-trivial. For simplicity, we denote Eq. (5) as

$$\min_{T} \quad f(T) + g(T)$$
$$\text{subject to} \quad T \in \mathcal{M}, \quad (6)$$

where the functions $f(T)$ and $g(T)$ are defined respectively as

$$f(T) = \sum_{\ell=1}^{m} \sum_{i=1}^{n_\ell} \mathcal{L}\left((p_\ell + q_\ell)^T x_i^\ell, y_i^\ell\right), \ g(T) = \gamma \|P\|_1,$$

and the set $\mathcal{M}$ is defined as

$$\mathcal{M} = \left\{ T \middle| T = \begin{pmatrix} P \\ Q \end{pmatrix}, P \in \mathbb{R}^{d \times m}, \|Q\|_* \leq \tau, Q \in \mathbb{R}^{d \times m} \right\}.$$

Note that $f(T)$ is a smooth convex function with Lipschitz continuous gradient L_f [8] as:

$$\|\nabla f(T_x) - \nabla f(T_y)\|_F \leq L_f \|T_x - T_y\|_F, \ \forall T_x, T_y \in \mathcal{M}, \ (7)$$

$g(T)$ is a non-smooth convex function, and $\mathcal{M}$ is a compact and convex set [8]; moreover, for any $L \geq L_f$, the following inequality holds [26]:

$$f(T_x) \leq f(T_y) + \langle T_x - T_y, \nabla f(T_y) \rangle + \frac{L}{2} \|T_x - T_y\|^2, \quad (8)$$

where $T_x, T_y \in \mathcal{M}$.

The projected gradient scheme computes the global minimizer of Eq. (6) via an iterative refining procedure. That is, given T_k as the intermediate solution of the kth iteration, we refine T_k as

$$T_{k+1} = T_k - t_k \mathcal{P}_k, \ \forall k, \quad (9)$$

where $\mathcal{P}_k$ and t_k denote the appropriate projected gradient direction and the step size, respectively. The computation of Eq. (9) depends on $\mathcal{P}_k$ and t_k; in the following subsections, we will present a procedure for estimating appropriate $\mathcal{P}_k$ and t_k, and defer the discussion of detailed projected gradient algorithms to Section 5.

Note that since $\mathcal{P}_k$ is associated with T_k and t_k, we denote $\mathcal{P}_k$ by $\mathcal{P}_{1/t_k}(T_k)$; the reason for using this notation will become clear from the following discussion.

3.1 Projected Gradient Computation

For any $L > 0$, we consider the construction associated with the smooth component $f(T)$ of the objective function in Eq. (6) as

$$f_L(S, T) = f(S) + \langle T - S, \nabla f(S) \rangle + \frac{L}{2} \|T - S\|_F^2,$$

where $S, T \in \mathbb{R}^{d \times m}$. It can be verified that $f_L(S, T)$ is strongly convex with respect to the variable T. Moreover, we denote

$$G_L(S, T) = f_L(S, T) + g(T), \quad (10)$$

where $g(T)$ is the non-smooth component of the objective function in Eq. (6). From the convexity in $g(T)$, $G_L(S, T)$ is strongly convex with respect to T. Since

$$\begin{aligned} G_L(S, T) = \ & f(S) - \frac{1}{2L} \|\nabla f(S)\|_F^2 \\ & + \frac{L}{2} \left\| T - \left(S - \frac{1}{L} \nabla f(S) \right) \right\|_F^2 + g(T), \end{aligned}$$

the global minimizer of $G_L(S, T)$ with respect to T can be computed as

$$\begin{aligned} T_{L,S} &= \arg \min_{T \in \mathcal{M}} G_L(S, T) \\ &= \arg \min_{T \in \mathcal{M}} \left(\frac{L}{2} \left\| T - \left(S - \frac{1}{L} \nabla f(S) \right) \right\|_F^2 + g(T) \right). (11) \end{aligned}$$

Therefore we can obtain the L-projected gradient [25] of f at S via

$$\mathcal{P}_L(S) = L(S - T_{L,S}). \quad (12)$$

It is obvious that $1/L$ can be seen as the step size associated with the projected gradient $\mathcal{P}_L(S)$ by rewritting Eq. (12) as $T_{L,S} = S - \mathcal{P}_L(S)/L$.

3.2 Step Size Estimation

From Eq. (12), the step size associated with $\mathcal{P}_L(S)$ is given by $1/L$. Denote the objective function in Eq. (6) as

$$F(T) = f(T) + g(T). \quad (13)$$

Theoretically, any step size $1/L$ satisfying $L \geq L_f$ guarantees the global convergence in the projected gradient based algorithms [25]. It follows from Eq. (8) that

$$F(T_{L,S}) \leq G_L(S, T_{L,S}), \ \forall L \geq L_f. \quad (14)$$

In practice we can estimate an appropriate L (hence the appropriate step size $1/L$) by ensuring the inequality in Eq. (14). By applying an appropriate step size and the associated projected gradient in Eq. (9), we can verify that [7, 25]

$$F(T) - F(T_{L,S}) \geq \langle T - S, \mathcal{P}_L(S) \rangle + \frac{1}{2L} \|\mathcal{P}_L(S)\|_F^2. \quad (15)$$

Moreover, by replacing S with T in Eq. (15), we have

$$F(T) - F(T_{L,T}) \geq \frac{1}{2L} \|\mathcal{P}_L(T)\|_F^2. \quad (16)$$

Note that the inequality in Eq. (15) characterizes the relationship of the objective values in Eq. (6) using T and its refined version via the procedure in Eq. (9).

4. EFFICIENT COMPUTATION

The projected gradient scheme requires to solve Eq. (11) for each iterative step given in Eq. (9). In Eq. (11), the objective function is non-smooth and the feasible domain set is non-trivial; we show that its optimal solution can be obtained by solving an unconstrained optimization problem and an Euclidean projection problem separately.

Denote T and S in Eq. (11) respectively as

$$T = \begin{pmatrix} T_P \\ T_Q \end{pmatrix}, \quad S = \begin{pmatrix} S_P \\ S_Q \end{pmatrix}.$$

Therefore the optimization problem in Eq. (11) can be expressed as

$$\min_{T_P, T_Q} \quad \frac{L}{2} \left\| \begin{pmatrix} T_P \\ T_Q \end{pmatrix} - \begin{pmatrix} \hat{S}_P \\ \hat{S}_Q \end{pmatrix} \right\|_F^2 + \gamma \|T_P\|_1$$
$$\text{subject to} \quad \|T_Q\|_* \leq \tau, \tag{17}$$

where $\hat{S}_P$ and $\hat{S}_Q$ can be computed respectively as

$$\hat{S}_P = S_P - \frac{1}{L} \nabla_P f(S), \; \hat{S}_Q = S_Q - \frac{1}{L} \nabla_Q f(S).$$

Note that $\nabla_P f(S)$ and $\nabla_Q f(S)$ denote the derivative of the smooth component $f(S)$ with respect to the variables P and Q, respectively. In our experiments, we focus on the least squares loss function, where the gradient of $f(T)$ with respect to P and Q can be expressed as

$$\nabla_P f(T) = \nabla_Q f(T) = 2 \left(XX^T (P + Q) - XY^T \right).$$

We can further rewrite Eq. (17) as

$$\min_{T_P, T_Q} \quad \beta \|T_P - \hat{S}_P\|_F^2 + \beta \|T_Q - \hat{S}_Q\|_F^2 + \gamma \|T_P\|_1$$
$$\text{subject to} \quad \|T_Q\|_* \leq \tau, \tag{18}$$

where $\beta = L/2$. Since T_P and T_Q are decoupled in Eq. (18), they can be optimized separately as presented in the following subsections.

4.1 Computation of T_P

The optimal T_P in Eq. (18) can be obtained by solving the following optimization problem:

$$\min_{T_P} \quad \beta \|T_P - \hat{S}_P\|_F^2 + \gamma \|T_P\|_1.$$

It is obvious that each entry of the optimal matrix T_P can be obtained by solving

$$\min_{\hat{t} \in \mathbb{R}} \quad \beta \|\hat{t} - \hat{s}\|^2 + \gamma |\hat{t}|, \tag{19}$$

where $\hat{s}$ denotes the entry in $\hat{S}_P$ corresponding to $\hat{t}$ in T_P. It is known [33] that the optimal $\hat{t}$ to Eq. (19) admits an analytical solution; for completeness, we present its proof in Lemma 4.1.

LEMMA 4.1. *The minimizer of Eq. (19) can be expressed as*

$$\hat{t}^* = \begin{cases} \hat{s} - \frac{\gamma}{2\beta} & \hat{s} > \frac{\gamma}{2\beta} \\ 0 & -\frac{\gamma}{2\beta} \leq \hat{s} \leq \frac{\gamma}{2\beta} \\ \hat{s} + \frac{\gamma}{2\beta} & \hat{s} < -\frac{\gamma}{2\beta} \end{cases}. \tag{20}$$

PROOF. Denote by $h(\hat{t})$ the objective function in Eq. (19), and by $\hat{t}^*$ the minimizer of $h(\hat{t})$. The subdifferential of $h(\hat{t})$ can be expressed as

$$\partial h(\hat{t}) = 2\beta(\hat{t} - \hat{s}) + \gamma \text{sgn}(\hat{t}),$$

where the function $\text{sgn}(\cdot)$ is given by

$$\text{sgn}(\hat{t}) = \begin{cases} \{1\} & \hat{t} > 0 \\ [-1, 1] & \hat{t} = 0 \\ \{-1\} & \hat{t} < 0 \end{cases}.$$

It is known that $\hat{t}^*$ minimizes $h(\hat{t})$ if and only if 0 is a subgradient of $h(\hat{t})$ at the point $\hat{t}^*$, that is,

$$0 \in 2\beta(\hat{t}^* - \hat{s}) + \gamma \text{sgn}(\hat{t}^*).$$

Since the equation above is satisfied with $\hat{t}^*$ defined in Eq. (20), we complete the proof of this lemma. $\square$

4.2 Computation of T_Q

The optimal T_Q in Eq. (18) can be obtained by solving the optimization problem as

$$\min_{T_Q} \quad \frac{1}{2} \|T_Q - \hat{S}_Q\|_F^2$$
$$\text{subject to} \quad \|T_Q\|_* \leq \tau, \tag{21}$$

where the constant $1/2$ is added into the objective function for convenient presentation. In the following theorem, we show that the optimal T_Q to Eq. (21) can be obtained via solving a simple convex optimization problem.

THEOREM 4.1. *Let $\hat{S}_Q = U\Sigma_S V^T \in \mathbb{R}^{d \times m}$ be the SVD of $\hat{S}_Q$, where $q = rank(\hat{S}_Q)$, $U \in \mathbb{R}^{d \times q}$, $V \in \mathbb{R}^{m \times q}$, and $\Sigma_S = diag(\varsigma_1, \cdots, \varsigma_q) \in \mathbb{R}^{q \times q}$. Let $\{\sigma_i\}_{i=1}^q$ be the minimizers of the following problem:*

$$\min_{\{\sigma_i\}_{i=1}^q} \quad \sum_{i=1}^q (\sigma_i - \varsigma_i)^2$$
$$\text{subject to} \quad \sum_{i=1}^q \sigma_i \leq \tau, \; \sigma_i \geq 0. \tag{22}$$

Denote $\Sigma = diag(\sigma_1, \cdots, \sigma_q) \in \mathbb{R}^{q \times q}$. Then the optimal solution to Eq. (21) is given by

$$T_Q^* = U\Sigma V^T.$$

PROOF. Assume that the optimal T_Q^* to Eq. (21) shares the same left and right singular vectors as $\hat{S}_Q$. Then the problem in Eq. (21) is reduced to the problem in Eq. (22). Thus, all that remains is to show that T_Q^* shares the same left and right singular vectors as $\hat{S}_Q$. Denote the Lagrangian function [11] associated with Eq. (21) as

$$H(T_Q, \lambda) = \frac{1}{2} \|T_Q - \hat{S}_Q\|_F^2 + \lambda(\|T_Q\|_* - \tau).$$

Since $\mathbf{0}$ is strictly feasible in Eq. (21), i.e., $\|0\|_* < \tau$, the Slater's condition [11] is satisfied and strong duality holds in Eq. (21). Let $\lambda^* \geq 0$ be the optimal dual variable [11] in Eq. (21). Therefore,

$$\begin{aligned} T_Q^* &= \arg\min_{T_Q} H(T_Q, \lambda^*) \\ &= \arg\min_{T_Q} \frac{1}{2} \|T_Q - \hat{S}_Q\|_F^2 + \lambda^* \|T_Q\|_*. \end{aligned}$$

Let $T_Q^* = U_T \Sigma_T V_T^T \in \mathbb{R}^{d \times m}$ be the SVD of T_Q^* and $r = rank(T_Q^*)$, where $U_T \in \mathbb{R}^{d \times r}$ and $U_T \in \mathbb{R}^{m \times r}$ are columnwise orthonormal, and $\Sigma_T \in \mathbb{R}^{r \times r}$ is diagonal consisting of non-zero singular values on the main diagonal. It is known [36] that the sub-differentials of $\|T_Q\|_*$ at T_Q^* can be expressed as

$$\begin{aligned} \partial \|T_Q^*\|_* &= \left\{ U_T V_T^T + D : D \in \mathbb{R}^{d \times m}, U_T^T D = 0, \right. \\ &\qquad \left. DV_T = 0, \|D\|_2 \leq 1 \right\}. \end{aligned} \tag{23}$$

On the other hand, we can verify that T_Q^* is optimal to Eq.(21) if and only if $\mathbf{0}$ is a subgradient of $H(T_Q, \lambda^*)$ at T_Q^*, that is,

$$\mathbf{0} \in \partial H(T_Q^*, \lambda^*) = T_Q^* - \hat{S}_Q + \lambda^* \partial \|T_Q^*\|_*. \quad (24)$$

Let $U_T^\perp \in \mathbb{R}^{d \times (d-m)}$ and $V_T^\perp \in \mathbb{R}^{m \times (m-r)}$ be the null space [18] of U_T and V_T, respectively. It follows from Eq. (23) that there exists a point $D_T = U_T^\perp \Sigma_d \left(V_T^\perp\right)^T$ such that

$$U_T V_T^T + D_T \in \partial \|T_Q^*\|_*$$

satisfies Eq. (24), and $\Sigma_d \in \mathbb{R}^{(d-m)\times(m-r)}$ is diagonal consisting of the singular values of D_T on the main diagonal. It follows that

$$
\begin{aligned}
\hat{S}_Q &= T_Q^* + \lambda^* \left(U_T V_T^T + D_T\right) \\
&= U_T \Sigma_T V_T^T + \lambda^* U_T V_T^T + \lambda^* U_T^\perp \Sigma_d \left(V_T^\perp\right)^T \\
&= U_T \left(\Sigma_T + \lambda^* I\right) V_T + U_T^\perp \left(\lambda^* \Sigma_d\right) \left(V_T^\perp\right)^T
\end{aligned}
$$

corresponds to the SVD of $\hat{S}_Q$. This completes the proof of this theorem. $\square$

Note that the problem in Eq. (22) is convex, and can be solved via an algorithm similar to the one in [23] proposed for solving the Euclidean projection onto the ℓ_1 ball.

5. ALGORITHMS AND CONVERGENCE

We present two algorithms based on the projected gradient scheme presented in Section 3 for solving the constrained convex optimization problem in Eq. (6), and discuss their rates of convergence. Note that the theorems in this section can be proved using standard techniques in [25, 26].

5.1 Projected Gradient Algorithm

We first present a simple projected gradient algorithm. Let T_k be the feasible solution point in the k-th iteration; the projected gradient algorithm refines T_k by recycling the following two steps: find a candidate $\hat{T}$ for the subsequent feasible solution point T_{k+1} via

$$\hat{T} = T_{L,T_k} = \arg\min_{T \in \mathcal{M}} G_L(T_k, T),$$

and meanwhile ensure the step size $\frac{1}{L}$ satisfying the condition

$$F(\hat{T}) \le G_L(T_k, \hat{T}).$$

Note that both T_k and $\hat{T}$ are feasible in Eq. (6). It follows from Eq. (16) that the solution sequence generated in the projected gradient algorithm leads to a non-increasing objective value in Eq. (6), that is,

$$F(T_{k-1}) \ge F(T_k), \ \forall k. \quad (25)$$

The pseudo-code of the projected gradient algorithm is presented in Algorithm 1, and its convergence rate analysis is summarized in Theorem 5.1. Note that the stopping criteria in line 11 of Algorithm 1 can be set as: the change of objective values in two successive steps are smaller than some pre-specified value (e.g., 10^{-5}).

THEOREM 5.1. *Let T^* be the global minimizer of Eq. (6); let L_f be the Lipschitz continuous gradient defined in Eq.(7). Denote by k the index of iteration, and by T_k the solution point in the kth iteration. Then Algorithm 1 converges at the rate of $\mathcal{O}(\frac{1}{k})$, i.e., for all $k \ge 1$, we have*

$$F(T_k) - F(T^*) \le \frac{\hat{L}}{2k} \|T_0 - T^*\|_F^2,$$

Algorithm 1 Projected Gradient Method
1: **Input:** $T_0, L_0 \in \mathbb{R}$, and max-iter.
2: **Output:** T.
3: **for** $i = 0, 1, \cdots$, max-iter **do**
4:   **while** (**true**)
5:     Compute $\hat{T} = T_{L_i, T_i}$ via Eq. (11).
6:     **if** $F(\hat{T}) \le G_{L_i}(T_i, \hat{T})$ **then** exit the loop.
7:       **else** update $L_i = L_i \times 2$.
8:     **end-if**
9:   **end-while**
10:   Update $T_{i+1} = \hat{T}$ and $L_{i+1} = L_i$.
11:   **if** stopping criteria satisfied **then** exit the loop.
12: **end-for**
13: Set $T = T_{i+1}$.

---

where $\hat{L} = \max\{L_0, 2L_f\}$, and L_0 and T_0 are the initial values of L_k and T_k in Algorithm 1, respectively.

5.2 Accelerated Projected Algorithm

The proposed projected gradient method Section 5.1 is simple to implement but converges slowly. We accelerate the projected gradient method using a scheme developed by Nesterov [26], which has been applied for solving various sparse learning formulations [22].

Algorithm 2 Accelerated Projected Gradient Method
1: **Input:** $T_0, L_0 \in \mathbb{R}$, and max-iter.
2: **Output:** T.
3: Set $T_1 = T_0$, $t_{-1} = 0$, and $t_0 = 1$.
4: **for** $i = 1, 2, \cdots$, max-iter **do**
5:   Compute $\alpha_i = (t_{i-2} - 1)/t_{i-1}$.
6:   Compute $S = (1 + \alpha_i)T_i - \alpha_i T_{i-1}$.
7:   **while** (**true**)
8:     Compute $\hat{T} = T_{L_i, S}$ via Eq. (11).
9:     **if** $F(\hat{T}) \le G_{L_i}(S, \hat{T})$ **then** exit the loop
10:       **else** update $L_i = L_i \times 2$.
11:     **end-if**
12:   **end-while**
13:   Update $T_{i+1} = \hat{T}$ and $L_{i+1} = L_i$.
14:   **if** stopping criteria satisfied **then** exit the loop.
15:   Update $t_i = \frac{1}{2}(1 + \sqrt{1 + 4t_{i-1}^2})$.
16: **end-for**
17: Set $T = T_{i+1}$.

We utilize two sequences of variables in the accelerated projected gradient algorithm: (feasible) solution sequence $\{T_k\}$ and searching point sequence $\{S_k\}$. In the i-th iteration, we construct the searching point as

$$S_k = (1 + \alpha_k)T_k - \alpha_k T_{k-1}, \quad (26)$$

where the parameter $\alpha_k > 0$ is appropriately specified as shown in Algorithm 2. Similar to the projected gradient method, we refine the feasible solution point T_{k+1} via the general step as:

$$\hat{T} = T_{L,S_k} = \arg\min_{T \in \mathcal{M}} G_L(S_k, T),$$

and meanwhile determine the step size by ensuring

$$F(\hat{T}) \le G_L(S_k, \hat{T}).$$

The searching point S_k may not be feasible in Eq. (6), which can be seen as a forecast of the next feasible solution point and hence

leads to the faster convergence rate in Algorithm 2. The pseudo-code of the accelerated projected gradient algorithm is presented in Algorithm 2, and its convergence rate analysis is summarized in the following theorem.

THEOREM 5.2. *Let T^* be the global minimizer of Eq. (6); let L_f be the Lipschitz continuous gradient defined in Eq.(7). Denote by k the index of iteration, and by T_k the solution point in the kth iteration. Then Algorithm 2 converges at the rate of $\mathcal{O}(\frac{1}{k^2})$, i.e., for all $k \geq 1$, we have*

$$F(T_{k+1}) - F(T^*) \leq \frac{2\hat{L}}{k^2}\|T_0 - T^*\|_F^2,$$

where $\hat{L} = \max\{L_0, 2L_f\}$, where L_0 and T_0 are the initial values of L_k and T_k in Algorithm 2.

Note that the convergence rate achieved by Algorithm 2 is optimal among the first-order methods [26].

6. EMPIRICAL EVALUATIONS

In this section, we evaluate the proposed multi-task learning formulation in comparison with other representative ones; we also conduct numerical studies on the proposed projected gradient algorithms. All algorithms are implemented in MATLAB, and the codes are available at the supplemental website[1].

Table 1: Statistics of the benchmark data sets.

Data Set	Sample Size	Dimension	Label	Type
Scene	2407	294	6	image
Yeast	2417	103	14	gene
MediaMill$_1$	8000	120	80	multimedia
MediaMill$_2$	8000	120	100	multimedia
References	7929	26397	15	text
Science	6345	24002	22	text

We employ six benchmark data sets in our experiments. One of them is *AR Face Data* [24]: we use its subset consisting of 1400 face images corresponding to 100 persons. Another three are LIB-SVM multi-label data sets[2]: for *Scene* and *Yeast*, we use the entire data sets; for *MediaMill*, we generate several subsets by randomly sampling 8000 data points with different numbers of labels. *References* and *Science* are Yahoo webpages data sets [34]: we preprocess the data sets following the same procedures in [15]. All of the benchmark data sets are normalized and their statistics are summarized in Table 1. Note that in our multi-task learning setting, each task corresponds to a label and we employ the least squares loss function for the following empirical studies.

6.1 Demonstration of Extracted Structures

We apply the proposed multi-task learning algorithm on the face images and then demonstrate the extracted sparse and low-rank structures. We use a subset of *AR Face Data* for this experiment. The original size of these images is 165×120; we reduce the size to 82×68.

We convert the face recognition problem into the multi-task learning setting, where one task corresponds to learning a linear classifier, i.e., $f_\ell(x) = (p_\ell + q_\ell)^T x$, for recognizing the faces of one person. By solving Eq. (5), we obtained p_ℓ (sparse structure) and q_ℓ (low-rank structure); we reshape p_ℓ and q_ℓ and plot them in Figure 2. We only plot p_1 and q_1 for demonstration. The first two plots
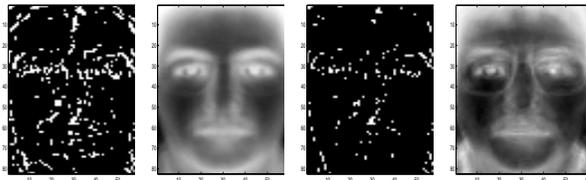
Figure 2: Extracted sparse (first and third plots) and low-rank (second and fourth plots) structures on AR face images with different sparse regularization and rank constraint parameters in Eq. (5): for the first two plots, we set $\gamma = 11, \tau = 0.08$; for the last two plots, we set $\gamma = 14, \tau = 0.15$.

in Figure 2 are obtained by setting $\gamma = 11, \tau = 0.08$ in Eq. (5): we obtain a sparse structure of 15.07% nonzero entries and a low-rank structure of rank 3; similarly, the last two plots are obtained by setting $\gamma = 14, \tau = 0.15$, we obtain a sparse structure of 5.35% nonzero entries and a low-rank structure of rank 7. We observe that the sparse structure identifies the important detailed facial marks, and the low-rank structure preserves the rough shape of the human face; we also observe that a large sparse regularization parameter leads to high sparsity (lower percentage of the non-zero entries) and a large rank constraint leads to structures of high rank.

6.2 Performance Evaluation

We compare the proposed multi-task learning formulation with other representative ones in terms of average Area Under the Curve (AUC), Macro F1, and Micro F1 [39]. The reported experimental results are averaged over five random repetitions of the data sets into training and test sets of the ratio $1 : 9$. In this experiment, we stop the iterative procedure of the algorithms if the change of the objective values in two consecutive iterations is smaller than 10^{-5} or the iteration numbers larger than 10^5. The experimental setup is summarized as follows:

1. **MixedNorm**: The proposed multi-task learning formulation with the least squares loss. The trace-norm constraint parameter is tuned in $\{10^{-2} \times i\}_{i=1}^{10} \cup \{10^{-1} \times i\}_{i=2}^{10} \cup \{2 \times i\}_{i=1}^p$, where $p = \lfloor k/2 \rfloor$ and k is the label number; the one-norm regularization parameter is tuned in $\{10^{-3} \times i\}_{i=1}^{10} \cup \{10^{-2} \times i\}_{i=2}^{10} \cup \{10^{-1} \times i\}_{i=2}^{10} \cup \{2 \times i\}_{i=1}^{10} \cup \{40 \times i\}_{i=1}^{20}$.

2. **OneNorm**: The formulation of the least squares loss with the one-norm regularization. The one-norm regularization parameter is tuned in $\{10^{-3} \times i\}_{i=1}^{10} \cup \{10^{-2} \times i\}_{i=2}^{10} \cup \{10^{-1} \times i\}_{i=2}^{10} \cup \{2 \times i\}_{i=1}^{10} \cup \{40 \times i\}_{i=1}^{20}$.

3. **TraceNorm**: The formulation of the least squares loss with the trace-norm constraint. The trace-norm constraint parameter is tuned in $\{10^{-2} \times i\}_{i=1}^{10} \cup \{10^{-1} \times i\}_{i=2}^{10} \cup \{2 \times i\}_{i=1}^p$, where $p = \lfloor k/2 \rfloor$ and k denotes the label number.

4. **ASO**: The alternating structure optimization algorithm [3]. The regularization parameter is tuned in $\{10^{-3} \times i\}_{i=1}^{10} \cup \{10^{-2} \times i\}_{i=2}^{10} \cup \{10^{-1} \times 2\}_{i=1}^{10} \cup \{2 \times i\}_{i=1}^{10} \cup \{40 \times i\}_{i=1}^{20}$; the dimensionality of the shared subspace is tuned in $\{2 \times i\}_{i=1}^p$, where $p = \lfloor k/2 \rfloor$ and k denotes the label number.

5. **IndSVM**: Independent support vector machines. The regularization parameter is tuned in $\{10^{-i}\}_{i=1}^3 \cup \{2 \times i\}_{i=1}^{50} \cup \{200 \times i\}_{i=1}^{20}$.

6. **RidgeReg**: Ridge regression. The regularization parameter is tuned in $\{10^{-3} \times i\}_{i=1}^{10} \cup \{10^{-2} \times i\}_{i=2}^{10} \cup \{10^{-1} \times 2\}_{i=1}^{10} \cup \{2 \times i\}_{i=1}^{10} \cup \{40 \times i\}_{i=1}^{20}$.

We present the averaged performance (with standard deviation) of the competing algorithms in Table 2. From Table 2, we have

Table 2: Average performance (with standard derivation) comparison of six competing algorithms on six data sets in terms of average AUC (top section), Macro F1 (middle section), and Micro F1 (bottom section). All parameters of the six methods are tuned via cross-validation, and the reported performance is averaged over five random repetitions.

|	Data	*Scene*	*Yeast*	*References*	*Science*	*MediaMill$_1$*	*MediaMill$_2$*
	MixedNorm	**91.602 ± 0.374**	**79.871 ± 0.438**	**77.526 ± 0.285**	**75.746 ± 1.423**	**72.571 ± 0.363**	**65.932 ± 0.321**
	OneNorm	87.846 ± 0.193	65.602 ± 0.842	75.444 ± 0.074	74.456 ± 1.076	70.453 ± 0.762	64.219 ± 0.566
Average	TraceNorm	90.205 ± 0.374	76.877 ± 0.127	71.259 ± 0.129	71.478 ± 0.293	69.469 ± 0.425	60.882 ± 1.239
AUC	ASO	86.258 ± 0.981	64.519 ± 0.633	75.960 ± 0.104	75.535 ± 1.591	71.067 ± 0.315	65.444 ± 0.424
	IndSVM	84.056 ± 0.010	64.601 ± 0.056	73.882 ± 0.244	70.220 ± 0.065	67.088 ± 0.231	57.437 ± 0.594
	RidgeReg	85.209 ± 0.246	65.491 ± 1.160	74.781 ± 0.556	69.177 ± 0.863	66.284 ± 0.482	56.605 ± 0.709
	MixedNorm	**60.602 ± 1.383**	**55.624 ± 0.621**	**37.135 ± 0.229**	**38.281 ± 0.011**	**9.706 ± 0.229**	**7.981 ± 0.011**
	OneNorm	55.061 ± 0.801	42.023 ± 0.120	36.579 ± 0.157	37.981 ± 0.200	8.579 ± 0.157	6.447 ± 0.133
Macro	TraceNorm	57.692 ± 0.480	52.400 ± 0.623	35.562 ± 0.278	36.447 ± 0.055	8.562 ± 0.027	6.765 ± 0.039
F1	ASO	56.819 ± 0.214	45.599 ± 0.081	34.462 ± 0.315	36.278 ± 0.183	8.023 ± 0.196	6.150 ± 0.023
	IndSVM	54.253 ± 0.078	38.507 ± 0.576	31.207 ± 0.416	35.175 ± 0.177	6.207 ± 0.410	5.175 ± 0.177
	RidgeReg	53.281 ± 0.949	42.315 ± 0.625	32.724 ± 0.190	35.066 ± 0.196	7.724 ± 0.190	5.066 ± 0.096
	MixedNorm	**64.392 ± 0.876**	**56.495 ± 0.190**	**59.408 ± 0.344**	52.619 ± 0.042	**61.426 ± 0.062**	**60.117 ± 0.019**
	OneNorm	59.951 ± 0.072	47.558 ± 1.695	58.798 ± 0.166	**52.733 ± 0.394**	60.594 ± 0.026	59.221 ± 0.39
Micro	TraceNorm	61.172 ± 0.838	54.172 ± 0.879	57.497 ± 0.130	49.124 ± 0.409	59.090 ± 0.117	58.317 ± 1.01
F1	ASO	59.015 ± 0.124	45.952 ± 0.011	55.406 ± 0.198	49.616 ± 0.406	59.415 ± 0.005	59.079 ± 1.72
	IndSVM	57.450 ± 0.322	52.094 ± 0.297	54.875 ± 0.185	48.574 ± 0.265	57.825 ± 0.272	56.525 ± 0.317
	RidgeReg	56.012 ± 0.144	46.743 ± 0.625	53.713 ± 0.213	47.454 ± 0.255	57.752 ± 0.210	56.982 ± 0.455
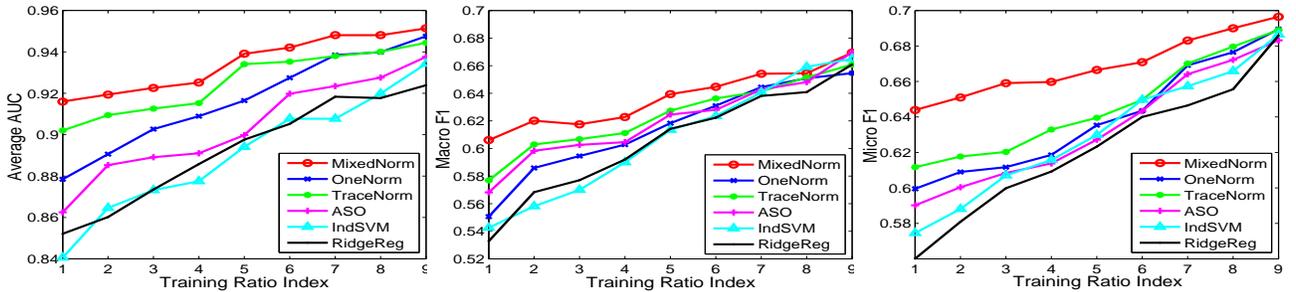
Figure 3: Performance comparison of six multi-task learning algorithms with different training ratios in terms of average AUC (left plot), Macro F1 (middle plot), and Micro F1 (right plot). The index on x-axis corresponds to the training ratio varying from 0.1 to 0.9.

the following observations: (1) MixedNorm achieves the best performance among the competing algorithms on all benchmark data sets in this experiment, which gives strong support for our rationale of improving the generalization performance by learning the sparse and low-rank patterns simultaneously from multiple tasks; (2) TraceNorm outperforms OneNorm on *Scene* and *Yeast* data sets, which implies that the shared low-rank structure may be important for image and gene classification tasks; meanwhile, OneNorm outperforms TraceNorm on *MediaMill* and yahoo webpage data sets, which implies that sparse discriminative features may be important for multimedia learning problems; (3) the multi-task learning algorithms in our experiments outperform SVM and RidgeReg, which verifies the effect of improved generalization performance via multi-task learning.

6.3   Sensitivity Study

We conduct sensitivity studies on the proposed multi-task learning formulation, and study how the training ratio and the task number affect its generalization performance.

Effect of the training ratio We use *Scene* data for this experiment. We vary the training ratio in the set $\{0.1 \times i\}_{i=1}^{9}$ and record the obtained generalization performance for each training ratio. The experimental results are depicted in Figure 3. We can observe that (1) for all of the compared algorithms, the resulting generalization performance improves with the increase of the training ra-

tio; (2) MixedNorm outperforms other competing algorithms in all cases in this experiment; (3) when the training ratio is small (e.g., smaller than 0.5), multi-task learning algorithms can significantly improve the generalization performance compared to IndSVM and RidgeReg; on the other hand, when the training ratio is large, all competing algorithms achieve comparable performance. This is consistent with previous observations that multi-task learning is most effective when the training size is small.

Effect of the task number We use *MediaMill* data for this experiment. We generate five data sets by randomly sampling 8000 data points with the task number set at $20, 40, 60, 80, 100$, respectively; for each data set, we set the training and test ratio at $1 : 9$ and record the average generalization performance of the multi-task learning algorithms over 5 random repetitions. The experimental results are depicted in Figure 4. We can observe that (1) for all of the compared algorithms, the achieved performance decreases with the increase of the task numbers; (2) MixedNorm outperforms or perform competitively compared to other algorithms with different task numbers; (3) all of the specific multi-task learning algorithms outperform IndSVM and RidgeReg. Note that the learning problem becomes more difficult as the number of the tasks increases, leading to decreased performance for both multi-task and single-task learning algorithms. We only present the performance comparison in terms of Macro/Micro F1; we observe a similar trend in terms of average AUC in the experiments.
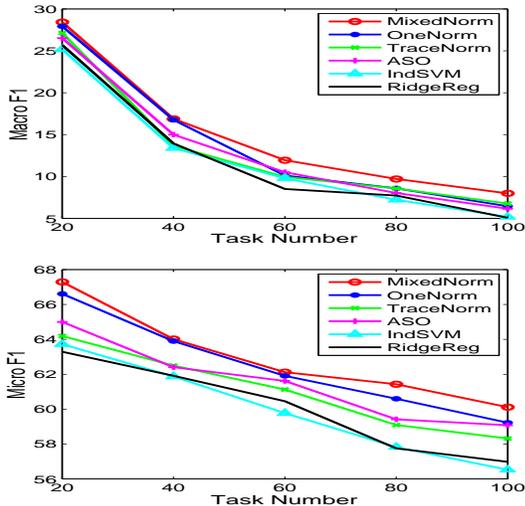
Figure 4: Performance comparison of the six competing multi-task learning algorithms with different numbers of tasks in terms of Macro F1 (top plot) and Micro F1 (bottom plot).

6.4 Comparison of PG and AG

We empirically compare the projected gradient algorithm (PG) in Algorithm 1 and the accelerated projected gradient algorithm (AG) in Algorithm 2 using *Scene* data. We present the comparison results of setting $\gamma = 1, \tau = 2$ and $\gamma = 6, \tau = 4$ in Eq. (5); for other parameter settings, we observe similar trends in our experiments.
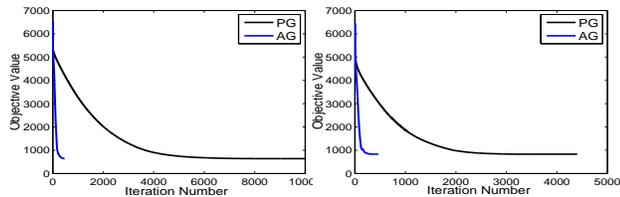

Figure 5: Convergence rate comparison between PG and AG: the relationship between the objective value of Eq. (5) and the iteration number (achieved via PG and AG, respectively). For the left plot, we set $\gamma = 1, \tau = 2$; for the right plot, we set $\gamma = 6, \tau = 4$.

Comparison on convergence rate We apply PG and AG for solving Eq. (5) respectively, and compare the relationship between the obtained objective values and the required iteration numbers. The experimental setup is as follows: we terminate the PG algorithm when the change of objective values in two successive steps is smaller than 10^{-5} and record the obtained objective value; we then use such a value as the stopping criterion in AG, that is, we stop AG when AG attains an objective value equal to or smaller than the one attained by PG. The experimental results are presented in Figure 5. We can observe that AG converges much faster than PG, and their respective convergence speeds are consistent with the theoretical convergence analysis in Section 5, that is, PG converges at the rate of $\mathcal{O}(1/k)$ and AG at the rate of $\mathcal{O}(1/k^2)$, respectively.

Comparison on computation cost We compare PG and AG in terms of computation time (in seconds) and iteration numbers (for attaining convergence) by using different stopping criteria $\{10^{-i}\}_{i=1}^{10}$.
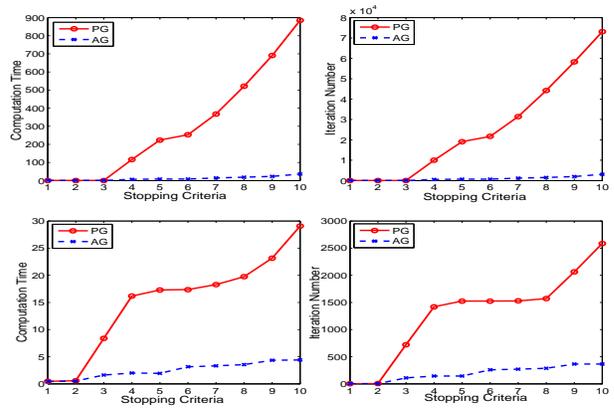

Figure 6: Comparison of PG and AG in terms of the computation time in seconds (left column) and iteration number (right column) with different stopping criteria. The x-axis indexes the stopping criterion from 10^{-1} to 10^{-10}. Note that we stop PG or AG when the change of the objective value in Eq. (5) is smaller than the value of stopping criteria. For the first row, we set $\gamma = 1, \tau = 2$; for the second row, we set $\gamma = 6, \tau = 4$.

We stop PG and AG if the stopping criterion is satisfied, that is, the change of the objective values in two successive steps is smaller than 10^{-i}. The experimental results are presented in Table 3 and Figure 6. We can observe from these results that (1) PG and AG require higher computation costs (more computation time and larger numbers of iterations) for a smaller value of the stopping criterion (higher accuracy in the optimal solution); (2) in general, AG requires lower computation costs than PG in this experiment; such an efficiency improvement is more significant when a smaller value is used in the stopping criterion.

Table 3: Comparison of PG and AG in terms of computation time (in seconds) and iteration number using different stopping criteria.

| stopping criteria | $\gamma = 1, \tau = 2$ | | | | $\gamma = 6, \tau = 4$ | | | |
| | iteration | | time | | iteration | | time | |
	PG	AG	PG	AG	PG	AG	PG	AG
10^{-1}	2	2	0.6	0.4	3	3	0.5	0.4
10^{-2}	4	4	0.6	0.4	5	4	0.6	0.5
10^{-3}	17	15	0.6	0.5	722	110	8.4	1.6
10^{-4}	9957	537	116.1	6.5	1420	144	16.2	1.9
10^{-5}	19103	683	223.7	8.3	1525	144	17.3	1.9
10^{-6}	21664	683	253.0	8.3	1525	259	17.4	3.1
10^{-7}	31448	1199	367.9	14.3	1527	271	18.3	3.3
10^{-8}	44245	1491	521.3	18.4	1570	287	19.7	3.5
10^{-9}	58280	1965	690.5	23.0	2062	365	23.1	4.2
10^{-10}	73134	3072	885.4	35.9	2587	365	29.1	4.4

7. CONCLUSION

We consider the problem of learning sparse and low-rank patterns from multiple related tasks. We propose a multi-task learning formulation in which the sparse and low-rank patterns are induced respectively by a cardinality regularization term and a low-rank constraint. The proposed formulation is non-convex; we convert it into its tightest convex surrogate and then propose to apply the general projected gradient scheme to solve such a convex surrogate. We present the procedures for computing the projected gradient and ensuring the global convergence of the projected gradient scheme. Moreover, we show the projected gradient can be obtained via solv-

ing two simple convex subproblems. Additionally, we present two detailed algorithms based on the projected gradient scheme and discuss their rates of convergence. Our experiments on benchmark data sets demonstrate the effectiveness of the proposed multi-task learning formulation and the efficiency of the proposed projected gradient algorithms. In the future, we plan to conduct a theoretical analysis on the proposed multi-task learning formulation and apply the proposed algorithm to other real-world applications.

Acknowledgment

8. REFERENCES

[1] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, 2009.

[2] R. K. Ando. BioCreative II gene mention tagging system at IBM Watson. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, 2007.

[3] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, 2005.

[4] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[5] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4:83–99, 2003.

[6] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[7] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Science*, 2:183–202, 2009.

[8] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar. *Convex Analysis and Optimization*. Athena Scientific, April 2003.

[9] J. Bi, T. Xiong, S. Yu, M. Dundar, and R. B. Rao. An improved multi-task learning approach with applications in medical diagnosis. In *ECML/PKDD*, 2008.

[10] S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer. Multi-task learning for HIV therapy screening. In *ICML*, 2008.

[11] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis. *Submitted for publication*.

[13] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[14] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky. Sparse and low-rank matrix decompositions. In *SYSID*, 2009.

[15] J. Chen, L. Tang, J. Liu, and J. Ye. A convex formulation for learning shared structures from multiple tasks. In *ICML*, 2009.

[16] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

[17] M. Fazel, H. Hindi, and S. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *ACL*, 2001.

[18] G. Gene and V. L. Charles. *Matrix computations*. Johns Hopkins University Press, 1996.

[19] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *NIPS*, 2008.

[20] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *ICML*, 2009.

[21] N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In *ICML*, 2004.

[22] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[23] J. Liu and J. Ye. Efficient euclidean projections in linear time. In *ICML*, 2009.

[24] A. Martinez and R. Benavente. The AR face database. Technical report, 1998.

[25] A. Nemirovski. *Efficient Methods in Convex Programming*. Lecture Notes, 1995.

[26] Y. Nesterov. *Introductory Lectures on Convex Programming*. Lecture Notes, 1998.

[27] G. Obozinski, B.Taskar, and M. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009.

[28] T. K. Pong, P. Tseng, S. Ji, and J. Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *Submitted to SIAM Journal on Optimization*, 2009.

[29] A. Schwaighofer, V. Tresp, and K. Yu. Learning gaussian process kernels via hierarchical bayes. In *NIPS*, 2004.

[30] A. Shapiro. Weighted minimum trace factor analysis. *Psychometrika*, 47:243–264, 1982.

[31] S. Si, D. Tao, and B. Geng. Bregman divergence-based regularization for transfer subspace learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:929–942, 2010.

[32] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, (11-12):653–625, 1998.

[33] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[34] N. Ueda and K. Saito. Single-shot detection of multiple categories of text using parametric mixture models. In *KDD*, 2002.

[35] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

[36] G. A. Watson. Characterization of the subdifferential of some matrix norms. *Linear Algebra and its Applications*, (170):33–45, 1992.

[37] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. In *NIPS*, 2009.

[38] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, 2007.

[39] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, 1997.

[40] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *ICML*, 2005.

[41] J. Zhang, Z. Ghahramani, and Y. Yang. Learning multiple related tasks using latent independent component analysis. In *NIPS*, 2005.