



Published in final edited form as:

ACM Trans Knowl Discov Data. 2012 February 1; 5(4): 22. doi:10.1145/2086737.2086742.

Learning Incoherent Sparse and Low-Rank Patterns from Multiple Tasks

Jianhui Chen,
Arizona State University

Ji Liu, and
Arizona State University

Jieping Ye
Arizona State University

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. We also present two projected gradient algorithms and analyze their rates of convergence in details. In addition, we illustrate the use of the presented projected gradient algorithms for the proposed multi-task learning formulation using the least squares loss. Experimental results on a collection of real-world data sets demonstrate the effectiveness of the proposed multi-task learning formulation and the efficiency of the proposed projected gradient algorithms.

Keywords

Multi-task learning; Low-rank and sparse patterns; Trace norm

1. INTRODUCTION

In the past decade there has been a growing interest in the problem of multi-task learning (MTL) [Caruana 1997]. MTL aims to enhance the overall generalization performance of the resulting classifiers by learning multiple tasks simultaneously in contrast to the single-task learning (STL) setting. It has been applied successfully in many areas of data mining and

© 2000 ACM

Author's address: J. Chen, J. Liu and J. Ye, Center for Evolutionary Medicine and Informatics, The Biodesign Institute, and Computer Science and Engineering, School of Computing, Informatics, and Decision System Engineering, Arizona State University, Tempe, AZ 85287, jianhui.chen@asu.edu, ji.liu@asu.edu, jieping.ye@asu.edu..

General Terms: Algorithms

machine learning [Ando 2007; Ando and Zhang 2005; Bi et al. 2008; Bickel et al. 2008; Si et al. 2010; Xue et al. 2007]. A common assumption in MTL is that all tasks are intrinsically related to each other. Under such an assumption, certain information 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 [Caruana 1997]; task relatedness are modeled using the common prior distribution in hierarchical Bayesian models [Bakker and Heskes 2003; Schwaighofer et al. 2004; Yu et al. 2005; Zhang et al. 2005]; the parameters of Gaussian Process covariance are learned from multiple tasks [Lawrence and Platt 2004]; kernel methods and regularization networks are extended to the multi-task learning setting [Evgeniou et al. 2005]; a convex formulation is developed for learning clustered tasks [Jacob et al. 2008; Zhou et al. 2011]; a shared low-rank structure is learned from multiple tasks [Ando and Zhang 2005; Chen et al. 2009]; the relatedness of multiple tasks is modeled using various structured sparsity penalties such as the tree-guided group lasso and the graph-guided fused lasso [Kim and Xing 2010; Chen et al. 2010; Chen et al. 2010]. Recently, trace norm regularization has been introduced into the multi-task learning domain [Abernethy et al. 2009; Argyriou et al. 2008; Ji and Ye 2009; Obozinski et al. 2010; Pong et al. 2010] to capture the task relationship via a shared low-rank structure of the model parameters, resulting in a tractable convex optimization problem [Liu et al. 2009b].

In many real-world applications involving multiple (related) tasks, the underlying predictive hypothesis structure [Ando and Zhang 2005; Shapiro 1982] may be complex, e.g., the predictive classifiers may admit a low-rank structure as well as a (entry-wise) sparse structure. For example, for the problems of reconstructing natural scene images via the multi-scale approximation approaches [Gonzalez and Woods 2002], the images usually have sparse representations (corresponding to a low-rank structure) in the form of a linear combination of certain bases; meanwhile, the images obtained under different illumination conditions or using different equipments may have only a small (sparse) sets of discriminative features (corresponding to a sparse structure). Analogously, for the problem of predicting disease progression in different geographic locations, the underlying progression patterns for all locations may share a set of basis factors, while the progression pattern for a specific location is also distinguished from other locations by a sparse set of features. Moreover in collaborative filtering or recommender system, a few factors contribute to the common taste of the individuals, while each individual's taste also relies on the personal inclination. Therefore it is desirable to utilize both the low-rank structure (representing the tasks relatedness) and the sparse structure (representing the task-wise discriminative patterns) for improved generalization performance and easily interpretable models for certain applications, e.g., the aforementioned real-world applications.

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 component. 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 leads 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 the 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. We also present two algorithms based on the projected gradient scheme and analyze their rates of convergence in details. In addition, we present an example of the proposed multi-task learning formulation using the least squares loss and illustrate the use of the presented projected gradient based algorithms in this case. We conduct extensive experiments on a collection of 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 algorithms based on the projected gradient scheme and analyze their rates of convergence in details; in Section 6 we present a concrete example on the use of the projected gradient based algorithms for the proposed multi-task learning formulation using the least squares loss; we report the experimental results in Section 7 and the paper concludes in Section 8.

Notations

For any matrix $A \in \mathbb{R}^{m \times n}$, let a_{ij} be the (i, j) th entry of A ; denote by $\|A\|_0$ the number of nonzero entries in A ; 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 of A in non-increasing order, where $r = \text{rank}(A)$; denote by $\|A\|_2 = \sqrt{\lambda_1(A)}$ and $\|A\|_* = \sum_{i=1}^r \sigma_i(A)$ the operator norm and the trace norm of A , respectively; let $\|A\|_{\max} = \max_{i,j} |a_{ij}|$.

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

$\{(x_i^\ell, y_i^\ell)\}_{i=1}^{n_\ell} \subset \mathbb{R}^d \times \{-1, +1\}$ ($\ell=1, \dots, m$). 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 ℓ th learning task.

2.1 Proposed Formulation

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 $Z = [z_1, \dots, z_m] \in \mathbb{R}^{d \times m}$, Z is the summation of a sparse matrix $P = [p_1, \dots, p_m] \in \mathbb{R}^{d \times m}$ and a low-rank matrix $Q = [q_1, \dots, q_m] \in \mathbb{R}^{d \times m}$ given by

$$Z = P + Q, \quad (1)$$

as illustrated in Figure 1. The ℓ_0 -norm (cardinality) [Boyd and Vandenberghe 2004], i.e., the number of non-zero entries, is commonly used to control the sparsity structure in the

matrix; similarity, matrix rank [Golub and Van Loan 1996] 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

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

where $\mathcal{L}(\cdot)$ denotes a smooth convex loss function, γ provides a trade-off between the sparse regularization term and the general 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 [Boyd and Vandenberghe 2004].

Define the function $f: \mathbb{C} \rightarrow \mathbb{R}$, where $\mathbb{C} \subseteq \mathbb{R}^{d \times m}$. The convex envelope [Boyd and Vandenberghe 2004] 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 to be the convex envelope of the ℓ_0 -norm as [Boyd and Vandenberghe 2004]:

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

Similarly, the trace norm (nuclear norm) has been shown to be the convex envelope of the rank function as [Fazel et al. 2001]:

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

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 replace the rank function with the trace norm function in Eq. (2), respectively. Therefore, we can reformulate the multi-task learning formulation as:

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

Note that in Eqs. (2) and (5), the transformation matrix is denoted by Z and it is the superposition of an entry-wise sparse matrix P and a low-rank matrix Q . 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) [Vandenberghe and Boyd 1996], and solved using many off-the-shelf optimization solvers such as SeDuMi [Sturm 2001]; however, SDP is computationally expensive and can only handle several hundreds of optimization variables. Since the proposed MTL formulation in Eq. (5) is convex, it admits a globally optimal solution; however, its (globally) optimal solution may not be unique, as the uniqueness of the solution is dependent on the involved training data, e.g., whether the resulting optimization formulation is strictly convex.

2.2 Related Work

Our multi-task learning formulation in Eq. (5) resembles the Alternating Structure Optimization algorithm (ASO) proposed in [Ando and Zhang 2005]. 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. 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 optimization 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.

Our formulation is also closely related to the concept of dirty models for multi-task learning [Jalali et al. 2010]. Specifically in [Jalali et al. 2010] a dirty model is proposed for the joint learning of multiple linear regression functions via a composite structured sparsity regularization; this model expresses the transformation matrix (of multiple regression functions) as a superposition of two matrix parameters and regularizes them respectively via the $\ell_{1,1}$ -norm and the $\ell_{1,\infty}$ -norm, in order to perform biased statistical estimation tailored to certain types of data.

Recent works in [Candès et al. 2009; Chandrasekaran et al. 2009; Wright et al. 2009] 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 [Boyd and Vandenberghe 2004] to solve the constrained optimization problem in Eq. (5). The projected gradient scheme is a first-order method; it requires only the computation of the first-order information and hence it is suitable for large scale data analysis. This type of methods has recently attracted intense interest in machine learning and data mining areas [Bach et al. 2011; Nemirovski 1995; Liu et al. 2009b]. Specifically it has been employed for solving mathematical formulations arising from different multi-task learning scenarios [Chen et al. 2009; Zhang et al. 2010; Liu et al. 2009a].

In Eq. (5), the objective function is non-smooth and the feasible domain is associated with a trace norm constraint. For simplicity, we denote the optimization in Eq. (5) as

$$\begin{aligned} \min_T \quad & f(T) + g(T) \\ \text{subject to} \quad & T \in \mathcal{M}, \end{aligned} \quad (6)$$

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

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

and the set \mathcal{M} is defined as

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

Note that $f(T)$ is a smooth convex function with a Lipschitz constant L_f [Bertsekas et al. 2003] as:

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

$g(T)$ is a non-smooth convex function, and \mathcal{M} is a compact and convex set [Bertsekas et al. 2003]. It is known that the smallest Lipschitz constant L_f in Eq. (7), i.e., $L_f = \min L_{f_i}$ is called the best Lipschitz constant for the function $f(T)$; moreover, for any $L \geq L_f$ the following inequality holds [Nesterov 1998]:

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

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

The projected gradient scheme computes the global minimizer of Eq. (6) via an iteratively updating procedure. That is, given T_k as the intermediate solution of the k th iteration, we update T_k as

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

where \mathcal{P}_k and t_k denote the appropriate projected gradient direction and the step size, respectively. The appropriate choice of \mathcal{P}_k and t_k is key to the global convergence of the projected gradient scheme. The computation in 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 based algorithms to Section 5. Note that since the determination of \mathcal{P}_k is associated with T_k and t_k , we denote \mathcal{P}_k by $\mathcal{P}_{1/t_k}(T_k)$, and the reason will become clear from the following discussion. It is worth pointing out that throughout this paper we focus on using a smooth convex loss function in the proposed multi-task learning formulation as described in Eq. (2) and hence the objective function (of the proposed multi-task learning formulation) can be expressed as the combination of a smooth component and a non-smooth component as in Eq. (6); for this case the projected gradient scheme is applicable; when the loss function is non-smooth convex, we can resort to the subgradient algorithms [Bertsekas 1999].

3.1 Projected Gradient Computation

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

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

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 of $g(T)$, $G_L(S, T)$ is strongly convex with respect to T . Since

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

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

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

We can then obtain the projected gradient of f at S via

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

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

$$T_{L,S} = S - \frac{1}{L} \mathcal{P}_L(S). \quad (13)$$

Note that if the inequality $f(T_{L,S}) - f_L(S, T_{L,S})$ is satisfied, $\mathcal{P}_L(S)$ is called the L -gradient of f at S [Nemirovski 1995].

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

$$F(T) = f(T) + g(T). \quad (14)$$

Theoretically, any step size $1/L$ of the value L larger than the best Lipschitz constant L_f guarantees the global convergence in the projected gradient based algorithms [Nemirovski 1995]. It follows from Eq. (8) that

$$F(T_{L,S}) \leq G_L(S, T_{L,S}), \quad \forall L \geq L_f. \quad (15)$$

In practice we can estimate an appropriate L (hence the appropriate step size $1/L$) by ensuring the inequality in Eq. (15). By applying an appropriate step size and the associated projected gradient in Eq. (9), we can verify an important inequality [Beck and Teboulle 2009; Nemirovski 1995], as summarized in the following lemma.

LEMMA 3.1. Let L_f be the Lipschitz continuous gradient associated with the function $f(T)$ as defined in Eq. (7). Let $S \in \mathbb{R}^{d \times m}$, and let $T_{L,S}$ be the minimizer of $G_L(S, T)$ as defined in Eq. (11). Then if $L \geq L_f$, the following inequality holds

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

for any $T \in \mathcal{M}$.

PROOF. Following from the convexity of $f(\cdot)$ and $g(\cdot)$, we have

$$f(T) \geq f(S) + \langle T - S, \nabla f(S) \rangle \quad (17)$$

$$g(T) \geq g(T_{L,S}) + \langle T - T_{L,S}, \partial g(T_{L,S}) \rangle, \quad (18)$$

where $g(T_{L,S})$ denotes the subgradient [Nesterov 1998] of $g(\cdot)$ at $T_{L,S}$. It is well known that T minimizes $G_L(S, T)$ (with respect to the variable T) if and only if $\mathbf{0}$ is a subgradient of $G_L(S, T)$ at T , that is,

$$\mathbf{0} \in \mathbf{L}(T_{L,S} - S) + \nabla f(S) + \partial g(T_{L,S}). \quad (19)$$

From Eqs. (10), (14), (17) and (18), we have

$$\begin{aligned} F(T) - G_L(S, T_{L,S}) &= (f(T) + g(T)) - (f_L(S, T_{L,S}) + g(T_{L,S})) \\ &\geq \langle T - T_{L,S}, \nabla f(S) + \partial g(T_{L,S}) \rangle - \frac{L}{2} \|S - T_{L,S}\|_F^2 \\ &= -L \langle T - T_{L,S}, T_{L,S} - S \rangle - \frac{L}{2} \|S - T_{L,S}\|_F^2 \\ &= \langle T - S, \mathcal{P}_L(S) \rangle + \frac{1}{2L} \|\mathcal{P}_L(S)\|_F^2, \end{aligned}$$

where the second and third equalities follow from Eqs. (19) and (12), respectively. This completes the proof of this lemma.

By replacing S with T in Eq. (16), we have

$$F(T) - F(T_{L,T}) \geq \frac{1}{2L} \|\mathcal{P}_L(T)\|_F^2. \quad (20)$$

Note that the inequality in Eq. (16) characterizes the relationship of the objective values in Eq. (6) using T and its updated version via the procedure in Eq. (9).

4. EFFICIENT COMPUTATION

The projected gradient scheme requires to solve Eq. (11) at each iterative step. In Eq. (11), the objective function is non-smooth and the feasible domain set is associated with a trace norm constraint; 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

$$\begin{aligned} \min_{T_P, T_Q} \quad & \frac{L}{2} \left\| \begin{pmatrix} T_P \\ T_Q \end{pmatrix} - \begin{pmatrix} \hat{S}_P \\ \hat{S}_Q \end{pmatrix} \right\|_F^2 + \gamma \|T_P\|_1 \\ \text{subject to} \quad & \|T_Q\|_* \leq \tau, \end{aligned} \quad (21)$$

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

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

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. We can further rewrite Eq. (21) as

$$\begin{aligned} \min_{T_P, T_Q} \quad & \beta \|T_P - \hat{S}_P\|_F^2 + \beta \|T_Q - \hat{S}_Q\|_F^2 + \gamma \|T_P\|_1 \\ \text{subject to} \quad & \|T_Q\|_* \leq \tau, \end{aligned} \quad (22)$$

where $\beta = L/2$. Since T_P and T_Q are decoupled in Eq. (22), they can be optimized separately as presented in the following subsections.

4.1 Computation of T_P

The optimal T_P can be obtained by solving the following optimization problem:

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

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

$$\min_{\hat{t} \in} \beta |\hat{t} - \hat{s}|^2 + \gamma |\hat{t}|. \quad (23)$$

Note that \hat{s} denotes an entry in \hat{S}_P corresponding to t in T_P from the same location. It is known [Tibshirani 1996] that the optimal t to Eq. (23) admits an analytical solution; for completeness, we present its proof in Lemma 4.1.

LEMMA 4.1. *The minimizer of Eq. (23) can be expressed as*

$$\hat{t}^* = \begin{cases} \hat{s} - \frac{\gamma}{2\beta} & \hat{s} > \frac{\gamma}{2\beta} \\ 0 & -\frac{\gamma}{2\beta} \leq \hat{s} \leq \frac{\gamma}{2\beta} \\ \hat{s} + \frac{\gamma}{2\beta} & \hat{s} < -\frac{\gamma}{2\beta} \end{cases}. \quad (24)$$

PROOF. Denote by $h(\hat{t})$ the objective function in Eq. (23), 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 t^* minimizes $h(t)$ if and only if 0 is a subgradient of $h(t)$ at the point t^* , that is,

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

Since the equation above is satisfied with t^* defined in Eq. (24), we complete the proof of this lemma.

4.2 Computation of T_Q

The optimal T_Q can be obtained by solving the optimization problem:

$$\begin{aligned} \min_{T_Q} \quad & \frac{1}{2} \|T_Q - \hat{S}_Q\|_F^2 \\ \text{subject to} \quad & \|T_Q\|_* \leq \tau, \end{aligned} \quad (25)$$

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

$$\begin{aligned} \min_{\{\sigma_i\}_{i=1}^q} \quad & \sum_{i=1}^q (\sigma_i - \varsigma_i)^2 \\ \text{subject to} \quad & \sum_{i=1}^q \sigma_i \leq \tau, \quad \sigma_i \geq 0. \end{aligned} \quad (26)$$

Denote $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_q) \in \mathbb{R}^{q \times q}$. Then the optimal solution to Eq. (25) is given by

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

PROOF. Assume that the optimal T_Q^* to Eq. (25) shares the same left and right singular vectors as \hat{S}_Q . Then the problem in Eq. (25) is reduced to the problem in Eq. (26). 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 [Boyd and Vandenberghe 2004] associated with Eq. (25) 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. (25), i.e., $\mathbf{0} \in \text{int}(\mathcal{C})$, the Slater's condition [Boyd and Vandenberghe 2004] is satisfied and strong duality holds in Eq. (25). Let $\lambda^* \geq 0$ be the optimal dual variable [Boyd and Vandenberghe 2004] in Eq. (25). Therefore,

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

Let $T_Q^* = U_T \Sigma_T V_T^T \in \mathbb{R}^{d \times m}$ be the SVD of T_Q^* and $r = \text{rank}(T_Q^*)$, where $U_T \in \mathbb{R}^{d \times r}$ and $V_T \in \mathbb{R}^{m \times r}$ 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 [Watson 1992] that the subdifferentials of $\|T_Q\|_*$ at T_Q^* can be expressed as

$$\partial \|T_Q^*\|_* = \left\{ U_T V_T^T + D : D \in \mathbb{R}^{d \times m}, \quad U_T^T D = 0, \quad D V_T = 0, \quad \|D\|_2 \leq 1 \right\}. \quad (27)$$

On the other hand, we can verify that T_Q^* is optimal to Eq.(25) if and only if $\mathbf{0}$ is a subgradient of $H(T_Q, \lambda^*)$ at T_Q^* , that is,

$$\mathbf{0} \in \partial H(T_Q^*, \lambda^*) = \mathbf{T}_Q^* - \widehat{S}_Q + \lambda^* \partial \|T_Q^*\|_*. \quad (28)$$

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 [Golub and Van Loan 1996] of U_T and V_T , respectively. It follows from Eq. (27) that there exists a point

$D_T = U_T^\perp \Sigma_d (V_T^\perp)^T$ such that

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

satisfies Eq. (28), 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

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

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

Note that the optimization problem in Eq. (26) is convex, and can be solved via an algorithm similar to the one in [Liu and Ye 2009] 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 in Section 3 for solving the constrained convex optimization problem in Eq. (6), and analyze their rates of convergence using techniques in [Nemirovski 1995; Nesterov 1998].

5.1 Projected Gradient Algorithm

We first present a simple projected gradient algorithm. Let T_k be the feasible solution point at the k -th iteration; the projected gradient algorithm updates T_k by recycling the following two steps: find a candidate 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

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

Note that both T_k and T are feasible in Eq. (6). It follows from Eq. (20) that the solution sequence generated in the projected gradient algorithm leads to a non-increasing objective value in Eq. (6), that is,

$$F(T_{k-1}) \geq F(T_k), \quad \forall k. \quad (29)$$

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 criterion in line 11 of Algorithm 1 can be set as follows: 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 best Lipschitz continuous gradient defined in Eq.(7). Denote by k the index of the iteration, and by T_k the solution point at the k th iteration of Algorithm 1. Then we have

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

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

PROOF. It follows from Eq. (12) we have

$$T_{i+1} = T_{L_i, T_i} = T_i - \frac{1}{L_i} \mathcal{P}_{L_i}(T_i).$$

Moreover, from Eq. (16), we have

$$\begin{aligned} -\varepsilon_{i+1} &\geq \langle T^* - T_i, \mathcal{P}_{L_i}(T_i) \rangle + \frac{1}{2L_i} \|\mathcal{P}_{L_i}(T_i)\|_F^2 \\ &= \frac{L_i}{2} \left(-\|T_i\|_F^2 + \|T_{i+1}\|_F^2 + 2 \langle T^*, T_i - T_{i+1} \rangle \right), \end{aligned} \quad (30)$$

where $\varepsilon_{i+1} = F(T_{i+1}) - F(T^*)$. Moving $L_i/2$ to the left side in Eq. (30) and summing such a reformulation from $i = 0$ to $i = k$, we have

$$\begin{aligned}
\sum_{i=0}^k \frac{2}{L_i} \varepsilon_{i+1} &\leq \|T_0\|_F^2 - \|T_{k+1}\|_F^2 + 2 \langle T^*, T_{k+1} - T_0 \rangle \\
&= \|T_0 - T^*\|_F^2 - \|T_{k+1} - T^*\|_F^2 \\
&\leq \|T_0 - T^*\|_F^2.
\end{aligned}$$

Since $L_i \geq L_{i-1}$ from line 7 in algorithm 1, and $\alpha_i \leq \alpha_{i-1}$ from Eq. (29) for all i , we have

$$\varepsilon_{k+1} \leq \frac{L_k}{2(k+1)} \|T_0 - T^*\|_F^2.$$

Moreover, it can be verified that $L_0 \leq L_k \leq 2L_f$ for all k . This completes the proof of this theorem.

5.2 Accelerated Projected Algorithm

The proposed projected gradient method in Section 5.1 is simple to implement but converges slowly. We improve the projected gradient method using a scheme developed by Nesterov [Nesterov 1998], which has been applied for solving various sparse learning formulations [Liu et al. 2009b].

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

$$S_k = (1 + \alpha_k) T_k - \alpha_k T_{k-1}, \quad (31)$$

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

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

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 best Lipschitz continuous gradient defined in Eq.(7). Denote by k the index of the iteration, and by T_k the solution point at the k th iteration of Algorithm 2. Then we have

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

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

PROOF. Denote $\epsilon_i = F(T_i) - F(T^*)$. Setting $T = T_i$, $S = S_i$ and $L = L_i$ in Eq. (16), we have

$$\epsilon_i - \epsilon_{i+1} \geq \left\langle T_i - S_i, \mathcal{P}_{L_i}(S_i) \right\rangle + \frac{1}{2L_i} \|\mathcal{P}_{L_i}(S_i)\|_F^2, \quad (32)$$

where the left side of the inequality above follows from

$$T_{i+1} = T_{L_i, S_i} = \arg \min_{T \in \mathcal{M}} G_{L_i}(S_i, T).$$

Similarly, setting $T = T^*$, $S = S_i$ and $L = L_i$ in Eq. (16), we have

$$-\epsilon_{i+1} \geq \left\langle T^* - S_i, \mathcal{P}_{L_i}(S_i) \right\rangle + \frac{1}{2L_i} \|\mathcal{P}_{L_i}(S_i)\|_F^2. \quad (33)$$

Multiplying Eq. (32) by $t_{i-1} - 1$ and summing it with Eq. (33), we have

$$(t_{i-1} - 1)\epsilon_i - t_{i-1}\epsilon_{i+1} \geq \left\langle (t_{i-1} - 1)(T_i - S_i) + T^* - S_i, \mathcal{P}_{L_i}(S_i) \right\rangle + \frac{t_{i-1}}{2L_i} \|\mathcal{P}_{L_i}(S_i)\|_F^2. \quad (34)$$

Moreover, multiplying Eq. (34) by t_{i-1} , we have

$$t_{i-2}^2\epsilon_i - t_{i-1}^2\epsilon_{i+1} \geq \frac{1}{2L_i} \|t_{i-1}\mathcal{P}_{L_i}(S_i)\|_F^2 + \left\langle t_{i-1}\mathcal{P}_{L_i}(S_i), (t_{i-1} - 1)(T_i - S_i) + T^* - S_i \right\rangle. \quad (35)$$

where the left side is obtained via the equation

$$t_{i-1}^2 - t_{i-1} = t_{i-2}^2$$

from the line 15 in Algorithm 2. On the other hand, it follows from Eq. (12) we have

$$\mathcal{P}_{L_i}(S_i) = L_i(S_i - T_{L_i, S_i}) = L_i(S_i - T_{i+1}). \quad (36)$$

From Eq. (31) and the line 5 in Algorithm 2, we have

$$t_{i-1}S_i = t_{i-1}T_i + (t_{i-2} - 1)(T_i - T_{i-1}). \quad (37)$$

Denote

$$C_{i-2} = t_{i-2}T_i - (t_{i-2} - 1)T_{i-1} - T^*. \quad (38)$$

From Eqs. (36), (37) and (38), we can verify that

$$t_{i-1}\mathcal{P}_{L_i}(S_i) = t_{i-1}L_i(S_i - T_{i+1}) = L_i(C_{i-2} - C_{i-1}). \quad (39)$$

Moreover, we have

$$\begin{aligned} (t_{i-1} - 1)(T_i - S_i) + T^* - S_i &= (t_{i-1} - 1)T_i + T^* - t_{i-1}S_i \\ &= -t_{i-2}T_i + (t_{i-2} - 1)T_{i-1} + T^* = -C_{i-2}. \end{aligned} \quad (40)$$

Substituting Eqs. (39) and (40) into Eq. (35), we obtain

$$\begin{aligned} \|C_{i-1}\|_F^2 - \|C_{i-2}\|_F^2 &\leq \frac{2}{L_i} (t_{i-2}^2 \varepsilon_i - t_{i-1}^2 \varepsilon_{i+1}) \\ &\leq \frac{2}{L_{i-1}} t_{i-2}^2 \varepsilon_i - \frac{2}{L_i} t_{i-1}^2 \varepsilon_{i+1}. \end{aligned} \quad (41)$$

Summing Eq. (41) from $i = 1$ to $i = k$, we have

$$\|C_{k-1}\|_F^2 - \|C_{-1}\|_F^2 \leq \frac{2}{L_0} t_{-1}^2 \varepsilon_1 - \frac{2}{L_k} t_{k-1}^2 \varepsilon_{k+1}.$$

Therefore, we have

$$\begin{aligned} \frac{2}{L_k} t_{k-1}^2 \varepsilon_{k+1} &\leq \|C_{-1}\|_F^2 - \|C_{k-1}\|_F^2 + \frac{2}{L_0} t_{-1}^2 \varepsilon_1 \\ &\leq \|C_{-1}\|_F^2 + \frac{2}{L_0} t_{-1}^2 \varepsilon_1 = \|T_0 - T^*\|^2, \end{aligned} \quad (42)$$

where the equality follows from $t_{-1} = 0$ in Algorithm 2. From line 15 in Algorithm 2, we have

$$2t_i = 1 + \sqrt{1 + 4t_{i-1}^2} \geq 2t_{i-1} + 1. \quad (43)$$

Summing Eq. (43) from $i = 1$ to $i = k$, we have

$$t_k \geq \frac{1}{2} (k+1), \quad \forall k. \quad (44)$$

Substituting Eq. (44) into Eq. (42), we complete the proof.

The proof of Theorem 5.2 uses standard techniques in [Nemirovski 1995; Nesterov 1998] yet with simplification in several aspects for easy understanding. Note that the convergence rate achieved by Algorithm 2 is optimal among all first-order methods [Nesterov 1998; Nemirovski 1995].

6. EXAMPLE: LEARNING SPARSE AND LOW-RANK PATTERNS WITH LEAST SQUARES LOSS

In this section, we present a concrete example of learning the sparse and low-rank patterns from multiple tasks, i.e., the MTL formulation in Eq. (5) using the least squares loss function; we also illustrate the use of the PG and AG algorithms in this case. Mathematically, the specific MTL formulation can be expressed as

$$\begin{aligned} \min_{P, Q} & \quad \|(P+Q)^T X - Y\|_F^2 + \gamma \|P\|_1 \\ \text{subject to} & \quad \|Q\|_* \leq \tau, \end{aligned} \quad (45)$$

where $X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d \times n}$, and $Y = [y_1, y_2, \dots, y_n] \in \mathbb{R}^{m \times n}$. For simplicity in Eq. (45) we assume that all of the m tasks share the same set of training data, and the derivation below can be easily extended to the case where each learning task has a different set of training data.

6.1 Efficient Computation for the Key Component

The computation of Eq. (11) is involved at each iteration of the projected gradient scheme. For the specific MTL formulation in Eq. (45), given the intermediate solution pair $\{P_i, Q_i\}$ at the i -th iteration, the subsequent solution pair $\{P_{i+1}, Q_{i+1}\}$ can be obtained via

$$\begin{aligned} \min_{\hat{P}, \hat{Q}} \quad & \frac{L_i}{2} \|\hat{P} - \tilde{P}_i\|_F^2 + \frac{L_i}{2} \|\hat{Q} - \tilde{Q}_i\|_F^2 + \gamma \|\hat{P}\|_1 \\ \text{subject to} \quad & \|\hat{Q}\|_* \leq \tau, \end{aligned} \quad (46)$$

where L_i specifies the step size of the i -th iteration. The optimal P and Q to Eq. (46) can be obtained via solving two separate problems as below.

Computation of P The optimal P can be obtained via solving

$$\min_{\hat{P}} \quad \frac{L_i}{2} \|\hat{P} - \tilde{P}_i\|_F^2 + \gamma \|\hat{P}\|_1. \quad (47)$$

Based on the results in Section 4.1, the optimization problem in Eq. (47) can be further decomposed into entry-wise subproblems in the form of Eq. (23), which admits an analytical solution (Lemma 4.1).

Computation of Q The optimal Q can be obtained via solving

$$\begin{aligned} \min_{\hat{Q}} \quad & \|\hat{Q} - \tilde{Q}_i\|_F^2 \\ \text{subject to} \quad & \|\hat{Q}\|_* \leq \tau. \end{aligned} \quad (48)$$

Based on the results in Section 4.2, the optimal solution to Eq. (48) can be obtained via the following two steps:

- Compute the SVD of $\tilde{Q}_i = U_{Q_i} \Sigma_{Q_i} V_{Q_i}^T$, where $\text{rank}(\tilde{Q}_i) = q$, $U_{Q_i} \in \mathbb{R}^{d \times q}$, $V_{Q_i} \in \mathbb{R}^{m \times q}$, and $\Sigma_{Q_i} = \text{diag}(\hat{\varsigma}_1, \dots, \hat{\varsigma}_q) \in \mathbb{R}^{q \times q}$.
- Compute the optimal solution $\{\sigma_i^*\}_{i=1}^q$ to the following problem

$$\begin{aligned} \min_{\{\sigma_i\}_{i=1}^q} \quad & \sum_{i=1}^q (\sigma_i - \hat{\varsigma}_i)^2 \\ \text{subject to} \quad & \sum_{i=1}^q \sigma_i \leq \tau, \quad \sigma_i \geq 0. \end{aligned}$$

The optimal Q can be constructed as $\hat{Q} = U_{Q_i} \Sigma_Q V_{Q_i}^T$, where $\Sigma_Q = \text{diag}(\sigma_1^*, \dots, \sigma_q^*)$.

6.2 Estimation of the Lipschitz Constant

An appropriate step size $1/L$ in Eq. (13) is important for the global convergence of the projected gradient based algorithms and its value can be estimated via many sophisticated line search schemes [Boyd and Vandenberghe 2004] in general. In Algorithm 1 (lines 6 ~ 7) and Algorithm 2 (lines 9 ~ 10), the value of L is updated until the inequality in Eq. (15) is satisfied; however, this updating procedure may incur overhead cost in the computation.

Denote the smooth component of the objective function in Eq. (45) by

$$f(P, Q) = \|(P+Q)^T X - Y\|_F^2. \quad (49)$$

It can be verified that any Lipschitz constant L_f of the function $f(P, Q)$ can satisfy Eq. (15). Note that the gradient of $f(P, Q)$ with respect to P and Q can be expressed as

$$\nabla_P f(P, Q) = \nabla_Q f(P, Q) = 2 \left(X X^T (P+Q) - X Y^T \right).$$

To avoid the computational cost of estimating the Lipschitz constant for $f(P, Q)$, we directly estimate its best value (the smallest Lipschitz constant), as summarized in the following lemma.

LEMMA 6.1. *Given $X \in \mathbb{R}^{d \times n}$ and $Y \in \mathbb{R}^{m \times n}$, the best Lipschitz constant L_f of the function $f(P, Q)$ in Eq. (49) is no larger than $2 \sigma_X^2$, where σ_X denotes the largest singular value of X .*

PROOF. For arbitrary $P_x, P_y, Q \in \mathbb{R}^{d \times m}$, we have

$$\begin{aligned} \hat{L}_P &= \frac{\|\nabla_{P_x} f(P_x, Q) - \nabla_{P_y} f(P_y, Q)\|_F}{\|P_x - P_y\|_F} = \frac{\|2 X X^T (P_x - P_y)\|_F}{\|P_x - P_y\|_F} \\ &\leq \frac{2 \sigma_X^2 \|P_x - P_y\|_F}{\|P_x - P_y\|_F} = 2 \sigma_X^2. \end{aligned} \quad (50)$$

Similarly, for arbitrary $P, Q_x, Q_y \in \mathbb{R}^{d \times m}$, we have

$$\hat{L}_Q = \frac{\|\nabla_{Q_x} f(P, Q_x) - \nabla_{Q_y} f(P, Q_y)\|_F}{\|Q_x - Q_y\|_F} \leq 2 \sigma_X^2. \quad (51)$$

Therefore it follows from Eq. (7) that

$$\hat{L}_f \leq \max(\hat{L}_P, \hat{L}_Q) = 2 \sigma_X^2. \quad (52)$$

This completes the proof.

6.3 Main Algorithms

The pseudo-codes of the PG and AG algorithms for solving Eq. (45), i.e., Eq. (5) with the least squares loss, are presented in Algorithm 3 and Algorithm 4 respectively. The main difference between PG and AG lies in the construction of S_{P_i} and S_{Q_i} : in line 4 of Algorithm 3, S_{P_i} and S_{Q_i} are set as the pair of feasible points from the previous iteration; in line 6 of Algorithm 4, S_{P_i} and S_{Q_i} are set as the a linear combination of the feasible points from the previous and the current iterations, which are not necessarily feasible in Eq. (45). The different construction leads to different rates of convergence, i.e., $\mathcal{O}(1/k)$ in Algorithm 3 and $\mathcal{O}(1/k^2)$ in Algorithm 4.

7. EMPIRICAL EVALUATIONS

In this section, we evaluate the proposed multi-task learning formulation (MixedNorm) in comparison with other seven representative methods; we also conduct numerical studies on the proposed projected gradient based algorithms. The employed optimization algorithms are implemented in MATLAB.

We employ six benchmark data sets in our experiments. One of them is *AR Face Data* [Martinez and Benavente 1998]: we use a subset consisting of 1400 face images corresponding to 100 persons. The other three are LIBSVM multi-label data sets¹: 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 [Ueda and Saito 2002]: we preprocess the data sets following the same procedures in [Chen et al. 2009]. All of the benchmark data sets are normalized and their statistics are summarized in Table I. 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.

7.1 Demonstration of Extracted Structures

We apply the proposed multi-task learning algorithm on synthetic data and face data to demonstrate the extracted sparse and low-rank structures.

Demonstration on Synthetic Data—We apply the proposed multi-task learning algorithm on a synthetic data and also demonstrate the extracted sparse and low-rank structures. The synthetic data is constructed as follows: set the task number $m = 30$, the training sample size of each task $n_\ell = 60$, and the training sample dimensionality $d = 100$; generate the entries of the training data (of the ℓ -th task) $X^\ell = [x_1^\ell, \dots, x_{n_\ell}^\ell] \in \mathbb{R}^{d \times n_\ell}$ from $\mathcal{N}(0, 25)$; generate the entries of the sparse component $P = [p_1, \dots, p_k]$ (of size $k \times d$) from $\mathcal{N}(0, 100)$ and then set the entries of its first 10 rows and its first 30 columns as zero; generate the entries of the low-rank component $Q = [q_1, \dots, q_k]$ (of size $k \times d$) from $\mathcal{N}(0, 9)$ and then set its smallest 20 singular values as zero; construct the response vector of each task as $y^\ell = X^{\ell T} (p_\ell + q_\ell) + \delta^\ell \in \mathbb{R}^{n_\ell}$, where the entries in δ^ℓ are generated from $\mathcal{N}(0, 1)$. By this construction, we have 30 related tasks, where each task has 60 training samples of feature dimensionality 100; moreover, the transformation matrix for all 30 tasks consists of a sparse component as well as a low-rank component.

In the left column of Figure 2, we present the ground truth of the sparse and low-rank structures; in the right column we present the extracted sparse and low-rank structures via solving Eq. (45) with $\lambda = 150$ and $\mu = 50$. From the first row of the plots in Figure 2, we can observe that the extracted sparse structure (right plot) is similar to the ground truth of the sparse structure (left plot), i.e., most of the entries in the first 10 columns and the first 30 rows of the sparse component are zero; moreover from the second row of the plots, we can observe that the patterns of the extracted low-rank structure (right plot) is similar to the patterns of the ground truth of the low-rank structure (left plot), i.e., they have a similar number of non-zeros singular values. Our observation empirically demonstrates the potential of the proposed multi-task learning formulation for extracting sparse and low-rank structures (of multiple tasks).

Demonstration on AR Face Data—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 82×60 . We convert the face recognition problem into a multi-task learning problem, 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. (45), we obtained p_ℓ (sparse structure) and q_ℓ (low-rank structure); we reshape p_ℓ and q_ℓ and plot them in Figure 3. We only plot p_1 and q_1 for demonstration. The first two plots in Figure 3 are obtained by setting $\lambda = 11$, $\mu = 0.08$ in Eq.

¹<http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/multilabel/>

(45): 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 $\alpha = 14$, $\beta = 0.15$, 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 larger sparse regularization parameter leads to higher sparsity (lower percentage of the non-zero entries) and a larger rank constraint leads to structures of higher rank.

7.2 Performance Evaluation

We compare the proposed multi-task learning formulation with other seven representative algorithms including the multi-task relationship learning (MTRL) algorithm [Zhang and Yeung 2010], the multi-task learning with a sparse matrix-normal penalty (MTL[α & β]) algorithm [Zhang and Schneider 2010], the alternating structure optimization (ASO) algorithm [Ando and Zhang 2005], the least squares with trace norm regularization (TraceNorm) [Ji and Ye 2009; Pong et al. 2010], the least squares with one norm regularization (OneNorm) [Tibshirani 1996], the independent support vector machines (IndSVM) [Schölkopf and Smola 2002], and the ridge regression (RidgeReg). Note that in [Zhang and Schneider 2010], the authors propose various configurations of their MTL learning formulation; in our experiments, we focus on the most general one, i.e., MTL(α & β), in which the two covariance matrices are automatically learned.

In this experiment, we employ Average AUC (the averaged area under the curve), Macro F1 (the global calculation of F1 regardless of the tasks), and Micro F1 (the averaged F1 scores of all tasks) as the performance measures [Yang and Pedersen 1997]. The reported experimental results are averaged over five random repetitions of the data sets into training and test sets of a 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 below. Note that in our experiments, OneNorm, RidgeReg, and IndSVM represent the single task learning (STL) algorithms; they are employed as baseline algorithms; the comparison among MixedNorm, OneNorm, and TraceNorm is used to verify the effectiveness of the combination of OneNorm and TraceNorm.

1. **MixedNorm** (the proposed multi-task learning formulation with the least squares loss) The trace-norm constraint parameter is tuned in

$$\left\{10^{-2} \times i\right\}_{i=1}^{10} \cup \left\{10^{-1} \times i\right\}_{i=2}^{10} \cup \left\{2 \times i\right\}_{i=1}^p, \text{ where } p = \lfloor k/2 \rfloor \text{ and } k \text{ is the label number; the one-norm regularization parameter is tuned in}$$

$$\left\{10^{-3} \times i\right\}_{i=1}^{10} \cup \left\{10^{-2} \times i\right\}_{i=2}^{10} \cup \left\{10^{-1} \times i\right\}_{i=2}^{10} \cup \left\{2 \times i\right\}_{i=1}^{10} \cup \left\{40 \times i\right\}_{i=1}^{20}.$$

2. **MTRL** (the multi-task relationship learning formulation) The two regularization parameters (α_1 and α_2) in MTRL are tuned in the range

$$\left\{10^{-3} \times i\right\}_{i=1}^{10} \cup \left\{10^{-2} \times i\right\}_{i=2}^{10} \cup \left\{10^{-1} \times i\right\}_{i=2}^{10} \cup \left\{2 \times i\right\}_{i=1}^{10} \cup \left\{40 \times i\right\}_{i=1}^{20}.$$

3. **MTL(α & β)** (the multi-task learning formulation with a sparse matrix-normal penalty) Following [Zhang and Schneider 2010], we set $\lambda_\Omega = \lambda_\Sigma = \hat{\lambda}$. The two regularization parameters α and $\hat{\lambda}$ are tuned in the range

$$\left\{10^{-3} \times i\right\}_{i=1}^{10} \cup \left\{10^{-2} \times i\right\}_{i=2}^{10} \cup \left\{10^{-1} \times i\right\}_{i=2}^{10} \cup \left\{2 \times i\right\}_{i=1}^{10} \cup \left\{40 \times i\right\}_{i=1}^{20}.$$

4. **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.

5. **ASO** (the alternating structure optimization algorithm) 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.

6. **OneNorm** (the formulation of the least squares loss with the one-norm regularization) The one-norm regularization parameter is tuned in

$\{10^{-3} \times i\}_{i=1}^{10} \cup \{10^{-2} \times i\}_{i=2}^{10} \cup \{10^{-1} \times i\}_{i=2}^{10} \cup \{2 \times i\}_{i=1}^{10} \cup \{40 \times i\}_{i=1}^{20}$

7. **IndSVM** (independent support vector machines) The regularization parameter is

tuned in $\{10^{-i}\}_{i=1}^3 \cup \{2 \times i\}_{i=1}^{50} \cup \{200 \times i\}_{i=1}^{20}$.

8. **RidgeReg** (ridge regression) The regularization parameter is tuned in

$\{10^{-3} \times i\}_{i=1}^{10} \cup \{10^{-2} \times i\}_{i=2}^{10} \cup \{10^{-1} \times 2\}_{i=1}^{10} \cup \{2 \times i\}_{i=1}^{10} \cup \{40 \times i\}_{i=1}^{20}$.

The averaged performance (with standard deviation) of the competing algorithms are presented in Table II and Table III. We have 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 web-page 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 or perform competitively compared to OneNorm, IndSVM and RidgeReg, which verifies the effect of improved generalization performance via multi-task learning.

7.3 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 4. We can observe that (1) for all compared algorithms, the resulting generalization performance improves with the increase of the training ratio; (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 5 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 5. We can observe that (1) for all compared algorithms, the achieved performance decreases with the increase of the task numbers; (2) MixedNorm outperforms or performs competitively compared to other algorithms with different task numbers; (3) all 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 the average AUC in the experiments.

7.4 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 with $\epsilon = 1$, $\epsilon = 2$ and $\epsilon = 6$, $\eta = 4$ in Eq. (45); for other parameter settings, we observe similar trends in our experiments.

Comparison on convergence rate—We apply PG and AG for solving Eq. (45) 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 6. 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 $\{10^{-i}\}_{i=1}^{10}$. 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 IV and Figure 7. 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.

7.5 Discussion on Computation Cost of MixedNorm

The proposed multi-task learning formulation (MixedNorm) is formulated as a mathematical program which minimizes a non-smooth objective function subject to a trace norm constraint. We employ the projected gradient based algorithms to solve MixedNorm (described in Algorithms 3 and 4). From the pseudo-codes, we can observe that the key component for solving MixedNorm lies in the computation of Eq. (46), i.e., Eq. (11) with the least squares loss. Moreover, it follows from Section 6.1 that the dominant computation cost for solving Eq. (46) includes an SVD operation as well as an singular value projection.

To illustrate the trade-off between the computation cost and the generalization performance, we conduct a comparison among MixedNorm, OneNorm, RidgeReg, and IndSVM in terms

of the obtained average AUC and the required computation time (in seconds). From Table V, we can observe that MixedNorm requires more computation time; however it leads to significantly better performance in terms of average AUC. It is worth pointing out that our algorithms are implemented in Matlab, while the LIBSVM package [Chang and Lin 2011] (for solving IndSVM) is implemented in C and highly optimized as well.

7.6 Automated Annotation of the Gene Expression Pattern Images

We apply the proposed multi-task learning formulation for the automated annotation of the *Drosophila* gene expression pattern images from the FlyExpress² database. Note that for this biological application, we focus on comparing the performance of MixedNorm and its special cases, i.e., OneNorm and TraceNorm; we also use the single task learning algorithms, i.e., OneNorm, IndSVM, and RidgeReg, as baselines. The *Drosophila* gene expression pattern images capture the spatial and temporal dynamics of gene expression and hence facilitate the explication of the gene functions, interactions, and networks during *Drosophila* embryogenesis [Fowlkes et al. 2008; Lécuyer et al. 2007]. To provide text-based pattern searching, the gene expression pattern images are annotated manually using a structured controlled vocabulary (CV) in small groups based on the genes and the developmental stages as shown in Table VI. However, with a rapidly increasing number of gene expression pattern images, it is desirable to design computational approaches to automate the CV annotation process.

We preprocess the *Drosophila* gene expression pattern images (of the standard size 128×320) from the FlyExpress database following the procedures in [Ji et al. 2009]. The *Drosophila* images are from 16 specific stages, which are then grouped into 6 stage ranges (1 ~ 3, 4 ~ 6, 7 ~ 8, 9 ~ 10, 11 ~ 12, 13 ~ 16). We manually annotate the image groups (based on the genes and the developmental stages) using the structured CV terms. Each image group is then represented as a feature vector based on the bag-of-words and the soft-assignment sparse coding. Note that the SIFT features [Lowe 2004] are extracted from the images with the patch size set at 16×16 and the number of visual words in sparse coding set at 2000. The first stage range only contains 2 CV terms and we do not report the performance for this stage range. For other stage ranges, we consider the top 10 and 20 CV terms that appear most frequently in the image groups and treat the annotation of each CV term as one task. We generate 10 subsets for this experiment, and randomly partition each subset into training and test sets using the ratio 1 : 9. Note that the parameters in the competing algorithms are tuned following the experimental setting in Section 7.2.

We report the averaged AUC (Avg. AUC), Macro F1 (Mac. F1), and Micro F1 (Mic. F1) over 10 random repetitions in Table VII (for 10 CV terms) and Table VIII (for 20 CV terms), respectively. We can observe that MixedNorm achieves the best performance among the five algorithms on all subsets. In particular, MixedNorm outperforms its special cases, i.e., OneNorm and TraceNorm; MixedNorm also outperforms other single-task learning algorithms: IndSVM and RidgeReg. The experimental results demonstrate the effectiveness of learning the sparse and low-rank patterns from multiple tasks for improved generalization performance.

8. 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

²<http://www.flyexpress.net/>

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 that the projected gradient can be obtained via solving two simple convex subproblems. We also present two detailed projected gradient based algorithms and analyze their rates of convergence. Additionally, we illustrate the use of the presented projected gradient algorithms for the proposed multi-task learning formulation using the least squares loss. Our experiments 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.

Acknowledgments

This work was supported by NSF IIS-0812551, IIS-0953662, CCF-1025177, and NIH LM010730.

REFERENCES

- Abernethy J, Bach F, Evgeniou T, Vert J-P. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*. 2009; 10:803–826.
- Ando RK. BioCreative II gene mention tagging system at IBM Watson. *Proceedings of the Second BioCreative Challenge Evaluation Workshop*. 2007
- Ando RK, Zhang T. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*. 2005; 6:1817–1853.
- Argyriou A, Evgeniou T, Pontil M. Convex multi-task feature learning. *Machine Learning*. 2008; 73(3):243–272.
- Bach, F.; Jenatton, R.; Mairal, J.; Obozinski, G. Convex optimization with sparsity-inducing norms in optimization for machine learning. In: Sra, S.; Nowozin, S.; Wright, SJ., editors. *Optimization for Machine Learning*. MIT Press; 2011.
- Bakker B, Heskes T. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*. 2003; 4:83–99.
- Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal of Imaging Science*. 2009; 2:183–202.
- Bertsekas DP. *Nonlinear Programming*. Athena Scientific. 1999
- Bertsekas DP, Nedic A, Ozdaglar AE. *Convex Analysis and Optimization*. Athena Scientific. 2003
- Bi J, Xiong T, Yu S, Dundar M, Rao RB. An improved multi-task learning approach with applications in medical diagnosis. *ECML*. 2008
- Bickel S, Bogojeska J, Lengauer T, Scheffer T. Multi-task learning for HIV therapy screening. *ICML*. 2008
- Boyd, S.; Vandenberghe, L. *Convex Optimization*. Cambridge University Press; 2004.
- Candés EJ, Li X, Ma Y, Wright J. Robust principal component analysis? *Journal of ACM*. 2009; 58(1):1–37.
- Caruana R. Multitask learning. *Machine Learning*. 1997; 28(1):41–75.
- Chandrasekaran V, Sanghavi S, Parrilo PA, Willsky AS. Sparse and low-rank matrix decompositions. *SYSID*. 2009
- Chang C-C, Lin C-J. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*. 2011; 2:27:1–27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Chen J, Tang L, Liu J, Ye J. A convex formulation for learning shared structures from multiple tasks. *ICML*. 2009
- Chen X, Kim S, Lin Q, Carbonell JG, Xing EP. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *CoRR abs/1005.3579*. 2010

- Chen X, Lin Q, Kim S, Carbonell JG, Xing EP. An efficient proximal-gradient method for single and multi-task regression with structured sparsity. CoRR abs/1005.4717v3. 2010
- Chen X, Pan W, Kwok JT, Carbonell JG. Accelerated gradient method for multi-task sparse learning problem. ICDM. 2009
- Evgeniou T, Micchelli CA, Pontil M. Learning multiple tasks with kernel methods. Journal of Machine Learning Research. 2005; 6:615–637.
- Fazel M, Hindi H, Boyd S. A rank minimization heuristic with application to minimum order system approximation. ACC. 2001
- Fowlkes CC, Hendriks CLL, Keränen SV, Weber GH, Ruübel O, Huang M-Y, Cha Toor S, Depace AH, Simirenko L, Henriquez C, Beaton A, Weiszmann R, Celniker S, Hamann B, Knowles DW, Biggin MD, Eisen MB, Malik J. A quantitative spatiotemporal atlas of gene expression in the drosophila blastoderm. Cell. 2008; 133(2):364–374. [PubMed: 18423206]
- Golub, GH.; Van Loan, CF. Matrix computations. Johns Hopkins University Press; 1996.
- Gonzalez, RC.; Woods, RE. Digital Image Processing. Prentice Hall; 2002.
- Jacob L, Bach F, Vert J-P. Clustered multi-task learning: A convex formulation. NIPS. 2008
- Jalali A, Ravikumar P, Sanghavi S, Ruan C. A dirty model for multi-task learning. NIPS. 2010
- Ji S, Ye J. An accelerated gradient method for trace norm minimization. ICML. 2009
- Ji S, Yuan L, Li Y-X, Zhou Z-H, Kumar S, Ye J. Drosophila gene expression pattern annotation using sparse features and term-term interactions. KDD. 2009
- Kim S, Xing EP. Tree-guided group lasso for multi-task regression with structured sparsity. ICML. 2010
- Lawrence ND, Platt JC. Learning to learn with the informative vector machine. ICML. 2004
- Lécuyer E, Yoshida H, Parthasarathy N, Alm C, Babak T, Cerovina T, Hughes TR, Tomancak P, Krause HM. Global analysis of mRNA localization reveals a prominent role in organizing cellular architecture and function. Cell. 2007; 131(1):174–187. [PubMed: 17923096]
- Liu J, Ji S, Ye J. Multi-task feature learning via efficient $2,1$ -norm minimization. UAI. 2009a
- Liu, J.; Ji, S.; Ye, J. SLEP: Sparse Learning with Efficient Projections. Arizona State University; 2009b.
- Liu J, Ye J. Efficient euclidean projections in linear time. ICML. 2009
- Lowe DG. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision. 2004; 60(2):91–110.
- Martinez, A.; Benavente, R. The AR face database. Tech. rep. 1998.
- Nemirovski, A. Efficient Methods in Convex Programming. Lecture Notes; 1995.
- Nesterov, Y. Introductory Lectures on Convex Programming. Lecture Notes; 1998.
- Obozinski G, Taskar B, Jordan M. Joint covariate selection and joint subspace selection for multiple classification problems. Statistics and Computing. 2010; 20(2):231–252.
- Pong TK, Tseng P, Ji S, Ye J. Trace norm regularization: Reformulations, algorithms, and multi-task learning. SIAM Journal on Optimization. 2010; 20:6.
- Schölkopf, B.; Smola, AJ. Learning with kernels : support vector machines, regularization, optimization, and beyond. The MIT Press; 2002.
- Schwaighofer A, Tresp V, Yu K. Learning gaussian process kernels via hierarchical bayes. NIPS. 2004
- Shapiro A. Weighted minimum trace factor analysis. Psychometrika. 1982; 47(3):243–264.
- Si S, Tao D, Geng B. Bregman divergence-based regularization for transfer subspace learning. IEEE Transactions on Knowledge and Data Engineering. 2010; 22(7):929–942.
- Sturm JF. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optimization Methods and Software. 2001; 11-12:653–625.
- Tibshirani R. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, Series B. 1996; 58:267–288.
- Ueda N, Saito K. Single-shot detection of multiple categories of text using parametric mixture models. KDD. 2002
- Vandenberghe L, Boyd S. Semidefinite programming. SIAM Review. 1996; 38(1):49–95.

- Watson GA. Characterization of the subdifferential of some matrix norms. *Linear Algebra and its Applications*. 1992; 170:33–45.
- Wright J, Peng Y, Ma Y, Ganesh A, Rao S. Robust principal component analysis: Exact recovery of corrupted low-rank matrices by convex optimization. *NIPS*. 2009
- Xue Y, Liao X, Carin L, Krishnapuram B. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*. 2007; 8:35–63.
- Yang Y, Pedersen JO. A comparative study on feature selection in text categorization. *ICML*. 1997
- Yu K, Tresp V, Schwaighofer A. Learning Gaussian processes from multiple tasks. *ICML*. 2005
- Zhang J, Ghahramani Z, Yang Y. Learning multiple related tasks using latent independent component analysis. *NIPS*. 2005
- Zhang X, Saha A, Vishwanathan SVN. Regularized risk minimization by nesterov's accelerated gradient methods: Algorithmic extensions and empirical studies. *CoRR abs/1011.0472*. 2010
- Zhang Y, Schneider J. Learning multiple tasks with a sparse matrix-normal penalty. *NIPS*. 2010
- Zhang Y, Yeung D-Y. A convex formulation for learning task relationship in multi-task learning. *UAI*. 2010
- Zhou J, Chen J, Ye J. Clustered multi-task learning via alternating structure optimization. *NIPS*. 2011

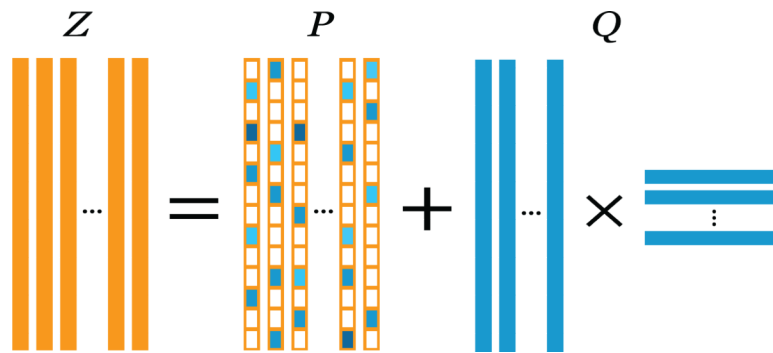
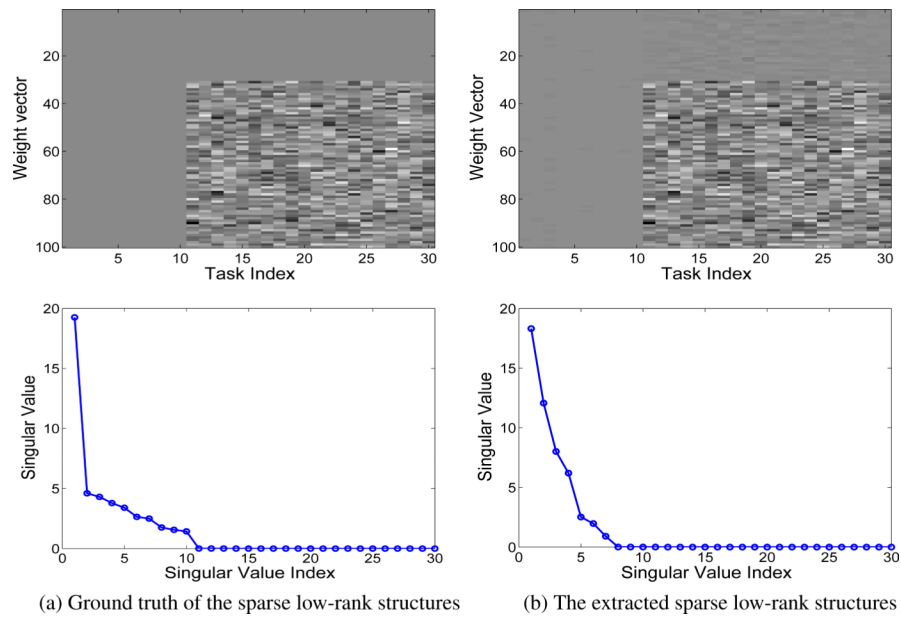


Fig. 1. Illustration of the transformation matrix Z in Eq. (1), where P denotes the sparse component with the zero-value entries represented by white blocks, and Q denotes the low-rank component.

**Fig. 2.**

Demonstration of the sparse and low-rank structures: the left column corresponds to the ground truth (of the sparse structure and the low-rank structure), and the right column corresponds to the extracted (sparse and low-rank) structures from Eq. (45); the first row shows the sparse components, and the second row shows the singular values of the low-rank components.

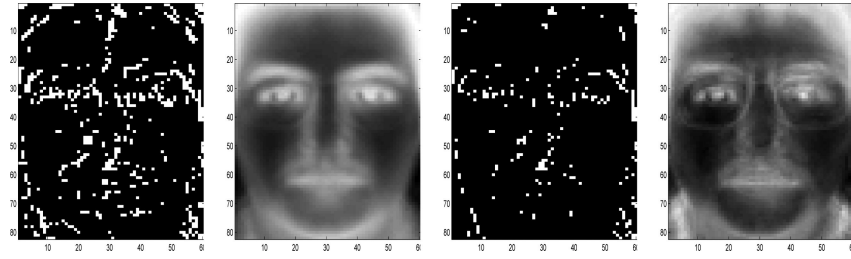


Fig. 3.

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. (45): for the first two plots, we set $\lambda = 11$, $\mu = 0.08$; for the last two plots, we set $\lambda = 14$, $\mu = 0.15$.

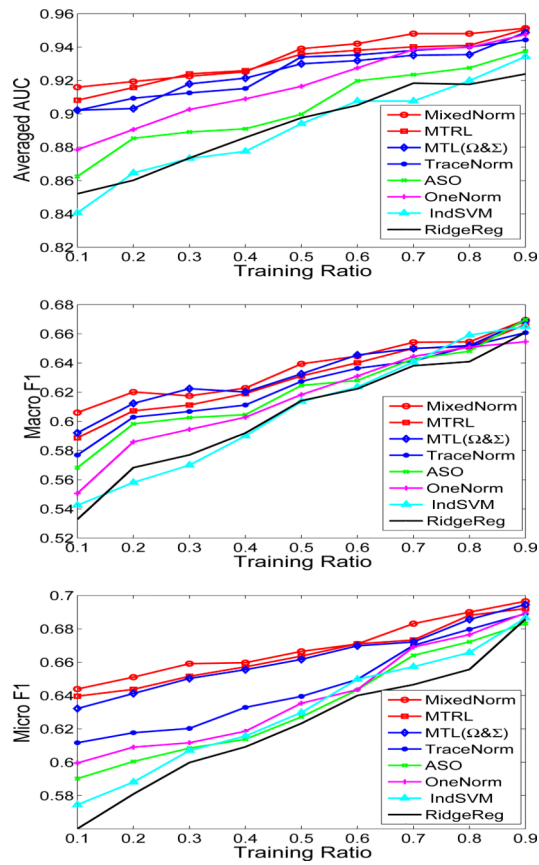


Fig. 4. Performance comparison of eight multi-task learning algorithms with different training ratios in terms of average AUC (top plot), Macro F1 (middle plot), and Micro F1 (bottom plot). The index on the x -axis corresponds to the training ratio varying from 0.1 to 0.9.

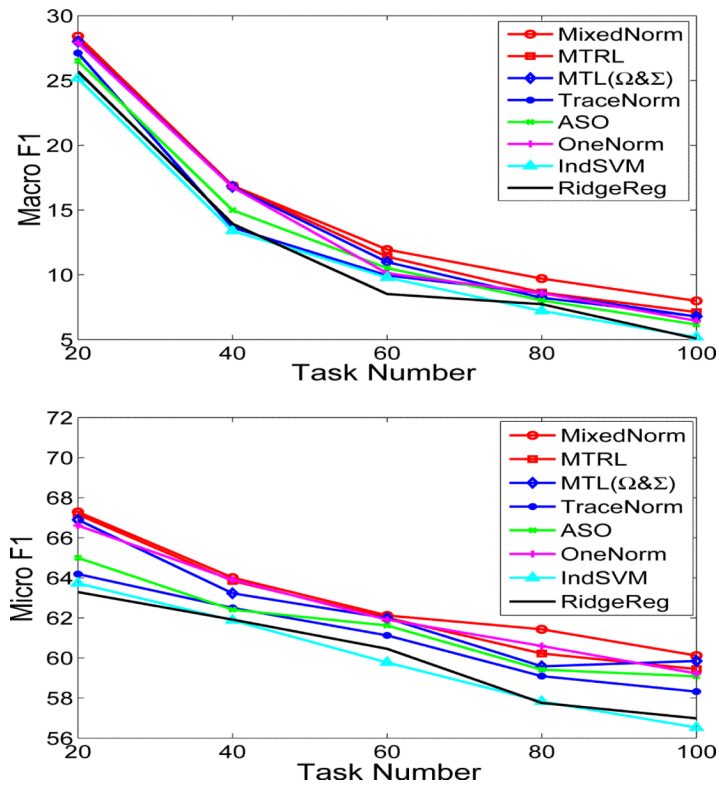


Fig. 5. Performance comparison of the eight competing multi-task learning algorithms with different numbers of tasks in terms of Macro F1 (top plot) and Micro F1 (bottom plot).

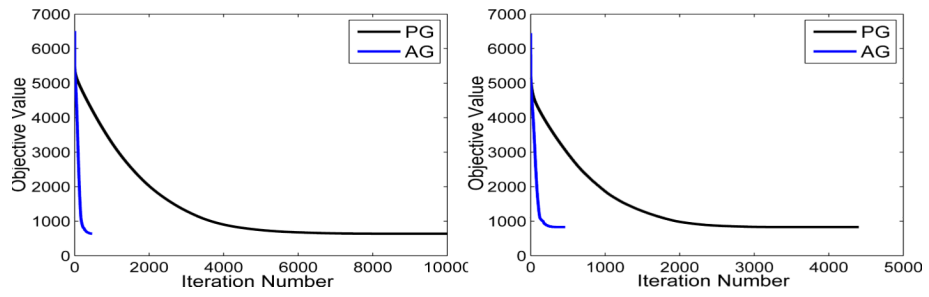


Fig. 6.

Convergence rate comparison between PG and AG: the relationship between the objective value of Eq. (45) and the iteration number (achieved via PG and AG, respectively). For the left plot, we set $\alpha = 1$, $\beta = 2$; for the right plot, we set $\alpha = 6$, $\beta = 4$.

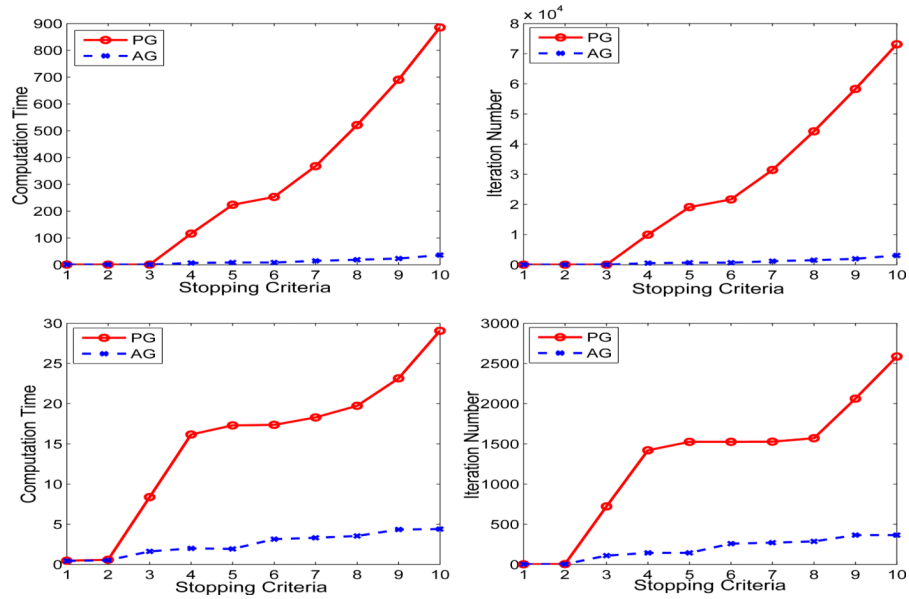


Fig. 7. Comparison of PG and AG in terms of the computation time in seconds (left column) and the iteration number (right column) with different stopping criteria. The x-axis indexes the stopping criterion from 10^{-1} to 10^{-10} . Note that we stop PG or AG when the change of the objective value in Eq. (45) is smaller than the value of the stopping criterion. For the first row, we set $\epsilon = 1$, $\delta = 2$; for the second row, we set $\epsilon = 6$, $\delta = 4$.

Table I

Statistics of the benchmark data sets.

Data Set	Sample Size	Dimension	Task Number	Type
Face	1400	19800	100	image
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

Table II

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

Data (n, d, m)	Scene (2407, 294, 6)	Yeast (2417, 103, 14)	References (7929, 26397, 15)	
Average AUC	MixedNorm	91.602 ± 0.374	79.871 ± 0.438	77.526 ± 0.285
	MTRL	90.821 ± 0.512	78.437 ± 0.610	75.133 ± 0.410
	MTL(&)	90.217 ± 1.139	78.515 ± 0.393	76.249 ± 0.277
	TraceNorm	90.205 ± 0.374	76.877 ± 0.127	71.259 ± 0.129
	ASO	86.258 ± 0.981	64.519 ± 0.633	75.960 ± 0.104
	OneNorm	87.846 ± 0.193	65.602 ± 0.842	75.444 ± 0.074
	IndSVM	84.056 ± 0.010	64.601 ± 0.056	73.882 ± 0.244
	RidgeReg	85.209 ± 0.246	65.491 ± 1.160	74.781 ± 0.556
Macro F1	MixedNorm	60.602 ± 1.383	55.624 ± 0.621	37.135 ± 0.229
	MTRL	58.873 ± 0.814	53.913 ± 0.785	36.492 ± 0.575
	MTL(&)	59.212 ± 0.671	54.854 ± 0.803	36.218 ± 0.157
	TraceNorm	57.692 ± 0.480	52.400 ± 0.623	35.562 ± 0.278
	ASO	56.819 ± 0.214	45.599 ± 0.081	34.462 ± 0.315
	OneNorm	55.061 ± 0.801	42.023 ± 0.120	36.579 ± 0.157
	IndSVM	54.253 ± 0.078	38.507 ± 0.576	31.207 ± 0.416
	RidgeReg	53.281 ± 0.949	42.315 ± 0.625	32.724 ± 0.190
Micro F1	MixedNorm	64.392 ± 0.876	56.495 ± 0.190	59.408 ± 0.344
	MTRL	63.958 ± 0.324	55.127 ± 0.922	58.112 ± 0.322
	MTL(&)	63.219 ± 0.769	54.235 ± 0.318	58.118 ± 1.246
	TraceNorm	61.172 ± 0.838	54.172 ± 0.879	57.497 ± 0.130
	ASO	59.015 ± 0.124	45.952 ± 0.011	55.406 ± 0.198
	OneNorm	59.951 ± 0.072	47.558 ± 1.695	58.798 ± 0.166
	IndSVM	57.450 ± 0.322	52.094 ± 0.297	54.875 ± 0.185
	RidgeReg	56.012 ± 0.144	46.743 ± 0.625	53.713 ± 0.213

Table III

Average performance comparison of eight competing algorithms on three data sets in terms of average AUC, Macro F1, and Micro F1. See the caption of Table II for detailed explanations.

Data (n, d, m)	<i>Science</i> (6345, 24002,22)	<i>MediaMill</i> ₁ (8000, 120, 80)	<i>MediaMill</i> ₂ (8000, 120, 100)	
Average AUC	MixedNorm	75.746 ± 1.423	72.571 ± 0.363	65.932 ± 0.321
	MTRL	73.389 ± 0.417	71.447 ± 0.185	64.980 ± 0.522
	MTL(&)	75.165 ± 0.439	70.625 ± 0.570	64.303 ± 0.241
	TraceNorm	71.478 ± 0.293	69.469 ± 0.425	60.882 ± 1.239
	ASO	75.535 ± 1.591	71.067 ± 0.315	65.444 ± 0.424
	OneNorm	74.456 ± 1.076	70.453 ± 0.762	64.219 ± 0.566
	IndSVM	70.220 ± 0.065	67.088 ± 0.231	57.437 ± 0.594
	RidgeReg	69.177 ± 0.863	66.284 ± 0.482	56.605 ± 0.709
Macro F1	MixedNorm	38.281 ± 0.011	9.706 ± 0.229	7.981 ± 0.011
	MTRL	36.320 ± 0.421	8.625 ± 0.507	7.112 ± 0.303
	MTL(&)	38.198 ± 0.348	8.225 ± 0.108	6.785 ± 0.624
	TraceNorm	36.447 ± 0.055	8.562 ± 0.027	6.765 ± 0.039
	ASO	36.278 ± 0.183	8.023 ± 0.196	6.150 ± 0.023
	OneNorm	37.981 ± 0.200	8.579 ± 0.157	6.447 ± 0.133
	IndSVM	35.175 ± 0.177	6.207 ± 0.410	5.175 ± 0.177
	RidgeReg	35.066 ± 0.196	7.724 ± 0.190	5.066 ± 0.096
Micro F1	MixedNorm	52.619 ± 0.042	61.426 ± 0.062	60.117 ± 0.019
	MTRL	51.212 ± 0.165	60.217 ± 0.311	59.443 ± 0.508
	MTL(&)	52.612 ± 1.192	59.578 ± 0.631	59.848 ± 1.140
	TraceNorm	49.124 ± 0.409	59.090 ± 0.117	58.317 ± 1.010
	ASO	49.616 ± 0.406	59.415 ± 0.005	59.079 ± 1.720
	OneNorm	52.733 ± 0.394	60.594 ± 0.026	59.221 ± 0.390
	IndSVM	48.574 ± 0.265	57.825 ± 0.272	56.525 ± 0.317
	RidgeReg	47.454 ± 0.255	57.752 ± 0.210	56.982 ± 0.455

Table IV

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

stopping criteria	$\epsilon = 1, \epsilon = 2$				$\epsilon = 6, \epsilon = 4$			
	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

Table V

Comparison of the proposed multi-task learning algorithm and the representative single task learning algorithm in terms of average AUC and computation time (in seconds). Note that data size corresponds to sample size, feature dimensionality, and task number.

Scene			Reference	
Data Size	$2406 \times 294 \times 6$		$7929 \times 26397 \times 15$	
Measures	Average AUC	Computation Time	Average AUC	Computation Time
MixedNorm	91.602	0.5460	77.526	159.868
OneNorm	87.846	0.4836	75.444	141.257
RidgeReg	85.209	0.2496	65.491	80.200
IndSVM	84.056	0.2946	73.882	5.479

Table VI

Sample images and the associated controlled vocabulary (CV) terms in FlyExpress database.

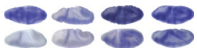
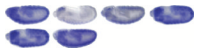
Stage Range	7 ~ 8	11 ~ 12
Gene	Pfx	Ran
Images Groups		
CV Terms	anterior endoderm anlage dorsal ectoderm primordium head mesoderm primordium P4 mesectoderm primordium posterior endoderm primordium P2 procephalic ectoderm anlage trunk mesoderm primordium P2 ventral ectoderm primordium P2 ventral nerve cord anlage visual anlage	anterior midgut primordium brain primordium posterior midgut primordium ventral nerve cord primordium

Table VII

Performance comparison of five competing algorithms for the gene expression pattern images annotation (10 CV terms) 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. Note that n , d , and m denote the sample size, dimensionality, and term (task) number, respectively.

Stage Range (n, d, m)		4 ~ 6 (925, 2000, 10)	7 ~ 8 (797, 2000, 10)	9 ~ 10 (919, 2000, 10)	11 ~ 12 (1622, 2000, 10)	13 ~ 16 (2228, 2000, 10)
Avg. AUC	MixedNorm	75.44 ± 0.87	75.55 ± 0.42	77.18 ± 0.50	83.82 ± 0.93	85.54 ± 0.25
	OneNorm	74.98 ± 0.12	73.80 ± 0.55	75.80 ± 0.24	82.78 ± 0.27	84.77 ± 0.20
	TraceNorm	73.04 ± 0.79	74.06 ± 0.46	76.71 ± 0.72	81.77 ± 1.10	83.64 ± 0.27
	IndSVM	71.00 ± 0.53	72.13 ± 0.70	73.58 ± 0.48	79.01 ± 0.58	82.06 ± 1.04
	RidgeReg	72.46 ± 0.15	72.51 ± 0.82	73.10 ± 0.38	80.83 ± 0.67	82.02 ± 0.15
Mac. F1	MixedNorm	43.71 ± 0.32	48.31 ± 0.56	53.11 ± 0.56	61.11 ± 0.58	61.81 ± 0.40
	OneNorm	42.24 ± 0.14	47.40 ± 0.23	51.04 ± 0.10	59.36 ± 0.60	61.02 ± 0.10
	TraceNorm	41.38 ± 0.36	46.51 ± 0.67	51.13 ± 0.95	61.05 ± 0.78	60.15 ± 0.45
	IndSVM	40.88 ± 0.49	46.73 ± 0.51	50.28 ± 0.65	59.82 ± 0.83	59.62 ± 0.94
	RidgeReg	41.65 ± 0.45	46.91 ± 0.94	50.69 ± 0.77	59.46 ± 0.95	60.59 ± 0.79
Mic. F1	MixedNorm	46.98 ± 0.90	62.73 ± 0.93	63.46 ± 0.07	69.31 ± 0.37	67.13 ± 0.41
	OneNorm	44.55 ± 0.38	60.02 ± 0.56	61.78 ± 0.10	68.54 ± 0.17	66.30 ± 0.55
	TraceNorm	43.88 ± 0.73	61.29 ± 0.78	61.33 ± 1.04	68.68 ± 0.27	66.37 ± 0.26
	IndSVM	42.05 ± 0.61	60.09 ± 0.78	60.57 ± 0.75	67.08 ± 0.99	65.95 ± 0.80
	RidgeReg	43.63 ± 0.41	59.95 ± 0.75	60.59 ± 0.66	66.87 ± 0.11	65.67 ± 1.10

Table VIII

Performance comparison of five competing algorithms for the gene expression pattern images annotation (20 CV terms).

Stage Range (n, d, m)		4 ~ 6 (1023, 2000, 20)	7 ~ 8 (827, 2000, 20)	9 ~ 10 (1015, 2000, 20)	11 ~ 12 (1940, 2000, 20)	13 ~ 16 (2476, 2000, 20)
Avg. AUC	MixedNorm	76.27 ± 0.53	72.03 ± 0.63	73.97 ± 1.10	82.27 ± 0.42	82.16 ± 0.16
	OneNorm	75.13 ± 0.03	70.95 ± 0.14	72.49 ± 1.00	81.73 ± 0.36	81.03 ± 0.08
	TraceNorm	74.69 ± 0.39	69.43 ± 0.46	71.59 ± 0.79	81.53 ± 0.16	80.88 ± 1.10
	IndSVM	73.82 ± 0.78	69.74 ± 0.19	70.84 ± 0.85	80.86 ± 0.56	79.94 ± 0.19
	RidgeReg	74.66 ± 1.44	70.77 ± 0.62	69.36 ± 1.44	80.40 ± 0.43	78.29 ± 0.42
Mac. F1	MixedNorm	31.90 ± 0.11	31.13 ± 0.68	32.28 ± 1.13	43.48 ± 0.39	43.44 ± 0.60
	OneNorm	30.48 ± 0.12	30.07 ± 0.56	30.50 ± 1.13	41.89 ± 0.24	42.64 ± 0.47
	TraceNorm	29.22 ± 0.31	30.24 ± 0.78	31.28 ± 0.54	42.07 ± 0.67	41.11 ± 0.52
	IndSVM	29.47 ± 0.46	28.85 ± 0.62	30.03 ± 1.68	41.63 ± 0.58	40.80 ± 0.66
	RidgeReg	28.92 ± 1.24	28.76 ± 0.95	29.94 ± 1.84	41.51 ± 0.39	40.84 ± 0.40
Mic. F1	MixedNorm	42.50 ± 0.63	57.04 ± 0.13	57.37 ± 0.71	61.97 ± 0.51	56.75 ± 0.40
	OneNorm	40.80 ± 0.48	56.55 ± 0.22	56.82 ± 0.04	60.59 ± 0.32	55.87 ± 0.11
	TraceNorm	41.26 ± 1.16	56.47 ± 0.27	55.37 ± 0.38	59.27 ± 0.93	54.08 ± 0.51
	IndSVM	39.24 ± 0.82	55.40 ± 0.15	55.75 ± 1.70	58.33 ± 0.53	53.61 ± 0.36
	RidgeReg	38.46 ± 0.41	56.08 ± 0.46	54.23 ± 0.85	59.13 ± 0.67	53.75 ± 0.31

Algorithm 1

Projected Gradient (PG) Method

```

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

```

Algorithm 2

Accelerated Projected Gradient (AG) Method

```

1: Input:  $T_0, L_0 \in \mathbf{R}$ , and max-iter.
2: Output:  $T$ .
3: Set  $T_1 = T_0, L_1 = 0$ , and  $t_0 = 1$ .
4: for  $i = 1, 2, \dots, \text{max-iter}$  do
5:   Compute  $\alpha_i = (t_{i-2} - 1) / t_{i-1}$ .
6:   Compute  $S = (1 + \alpha_i) T_{i-1} - \alpha_i T_{i-2}$ .
7:   while (true)
8:     Compute  $T = T_{L_i, S}$  via Eq. (11).
9:     if  $F(T) \leq G_{L_i}(S, T)$  then exit the loop
10:    else update  $L_i = L_i \times 2$ .
11:    end-if
12:  end-while
13:  Update  $T_{i+1} = T$  and  $L_{i+1} = L_i$ .
14:  if the stopping criterion is satisfied then exit the loop.
15:  Update  $t_i = \frac{1}{2} \left( 1 + \sqrt{1 + 4t_{i-1}^2} \right)$ .
16: end-for
17: Set  $T = T_{i+1}$ .

```

Algorithm 3

Projected Gradient Algorithm (PG) for Solving Eq. (45)

-
- 1: **Input:** $P_0, Q_0, L = 2 \frac{2}{X}$, and max-iter.
 - 2: **Output:** P, Q .
 - 3: **for** $i = 0, 1, \dots, \text{max-iter}$ **do**
 - 4: Set $L_i = L, S_{P_i} = P_i, S_{Q_i} = Q_i$
 - 5: Compute $P_i = S_{P_i} - P f(P, Q)|_{P=S_{P_i}, Q=S_{Q_i}}$
 - 6: $Q_i = S_{Q_i} - Q f(P, Q)|_{P=S_{P_i}, Q=S_{Q_i}}$
 - 7: Compute P via Eq. (47) and Q via Eq. (48).
 - 8: Set $P_{i+1} = P, Q_{i+1} = Q$.
 - 9: **if** the stopping criterion is satisfied **then** exit the loop.
 - 10: **end-for**
 - 11: Set $P = P_{i+1}, Q = Q_{i+1}$.
-

Algorithm 4

Accelerated Projected Gradient Algorithm (AG) for Solving Eq. (45)

-
- 1: **Input:** $P_0, Q_0, L = 2 \frac{2}{X}$, and max-iter.
 - 2: **Output:** P, Q .
 - 3: Set $P_1 = P_0, Q_1 = Q_0, t_1 = 0$ and $t_0 = 1$.
 - 4: **for** $i = 1, 2, \dots, \text{max-iter}$ **do**
 - 5: Compute $\beta_i = (t_{i-2} - 1)/t_{i-1}$.
 - 6: Set $L_i = L, S_{P_i} = (1 + \beta_i)P_i - \beta_i P_{i-1}, S_{Q_i} = (1 + \beta_i)Q_i - \beta_i Q_{i-1}$.
 - 7: Compute $P_i = S_{P_i} - \rho f(P, Q)|_{P=S_{P_i}, Q=S_{Q_i}}$
 - 8: $Q_i = S_{Q_i} - \rho f(P, Q)|_{P=S_{P_i}, Q=S_{Q_i}}$
 - 9: Compute P via Eq. (47), and Q via Eq. (48).
 - 10: Set $P_{i+1} = P, Q_{i+1} = Q$.
 - 11: **if** the stopping criterion is satisfied **then** exit the loop.
 - 12: Update $t_i = \frac{1}{2} \left(1 + \sqrt{1 + 4t_{i-1}^2} \right)$.
 - 13: **end-for**
 - 14: Set $P = P_{i+1}, Q = Q_{i+1}$.
-