A Regularization Approach to Learning Task Relationships in Multi-Task Learning

Yu Zhang, Department of Computer Science, Hong Kong Baptist University
Dit-Yan Yeung, Department of Computer Science and Engineering, Hong Kong University of Science and Technology

Multi-task learning is a learning paradigm which seeks to improve the generalization performance of a learning task with the help of some other related tasks. In this paper, we propose a regularization approach to learning the relationships between tasks in multi-task learning. This approach can be viewed as a novel generalization of the regularized formulation for single-task learning. Besides modeling positive task correlation, our approach, called *multi-task relationship learning* (MTRL), can also describe negative task correlation and identify outlier tasks based on the same underlying principle. By utilizing a matrix-variate normal distribution as a prior on the model parameters of all tasks, our MTRL method has a jointly convex objective function. For efficiency, we use an alternating method to learn the optimal model parameters for each task as well as the relationships between tasks. We study MTRL in the symmetric multi-task learning setting and then generalize it to the asymmetric setting as well. We also discuss some variants of the regularization approach to demonstrate the use of other matrix-variate priors for learning task relationships. Moreover, to gain more insight into our model, we also study the relationships between MTRL and some existing multi-task learning methods. Experiments conducted on a toy problem as well as several benchmark data sets demonstrate the effectiveness of MTRL as well as its high interpretability revealed by the task covariance matrix.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database Applications—*Data mining*

General Terms: Algorithms

Additional Key Words and Phrases: Multi-Task Learning; Regularization Framework; Task Relationship

1. INTRODUCTION

Multi-task learning [Caruana 1997; Baxter 1997; Thrun 1996] is a learning paradigm which seeks to improve the generalization performance of a learning task with the help of some other related tasks. This learning paradigm has been inspired by human learning activities in that people often apply the knowledge gained from previous learning tasks to help learn a new task. For example, a baby first learns to recognize human faces and later uses this knowledge to help it learn to recognize other objects. Multi-task learning can be formulated under two different settings: *symmetric* and *asymmetric* [Xue et al. 2007].

Email: yuzhang@comp.hkbu.edu.hk, dyyeung@cse.ust.hk

While symmetric multi-task learning seeks to improve the performance of all tasks simultaneously, the objective of asymmetric multi-task learning is to improve the performance of some target task using information from the source tasks, typically after the source tasks have been learned using some symmetric multi-task learning method. In this sense, asymmetric multi-task learning is related to *transfer learning* [Pan and Yang 2010], but the major difference is that the source tasks are still learned simultaneously in asymmetric multi-task learning but they are learned independently in transfer learning.

Major advances have been made in multi-task learning over the past decade, although some preliminary ideas actually date back to much earlier work in psychology and cognitive science. Multi-layered feedforward neural networks provide one of the earliest models for multi-task learning. In a multi-layered feedforward neural network, the hidden layer represents the common features for data points from all tasks and each unit in the output layer usually corresponds to the output of one task. Similar to the multi-layered feedforward neural networks, multi-task feature learning [Argyriou et al. 2008] also learns common features for all tasks but under the regularization framework. Unlike these methods, the regularized multi-task support vector machine (SVM) [Evgeniou and Pontil 2004] enforces the SVM parameters for all tasks to be close to each other. Another widely studied approach for multi-task learning is the task clustering approach [Thrun and O'Sullivan 1996; Bakker and Heskes 2003; Xue et al. 2007; Kumar and III 2012]. Its main idea is to group all the tasks into several clusters and then learn similar data features or model parameters for the tasks within each cluster. An advantage of this approach is its robustness against outlier tasks because they reside in separate clusters that do not affect other tasks. As different tasks are related in multi-task learning, model parameters of different tasks are assumed to share a common subspace in [Ando and Zhang 2005; Chen et al. 2009] and to deal with outlier tasks which are not related with other remaining tasks, the methods in [Chen et al. 2010; Jalali et al. 2010; Chen et al. 2011] assumed the model parameter matrix consists of a low-rank part to capture the correlated tasks and a structurally sparse part to model the outlier tasks. Moreover, some Bayesian models have been proposed for multi-task learning by using Gaussian process [Yu et al. 2005; Bonilla et al. 2007], t process [Yu et al. 2007; Zhang and Yeung 2010b], Dirichlet process [Xue et al. 2007], Indian buffet process [Rai and III 2010; Zhu et al. 2011; Passos et al. 2012], and sparse Bayesian models [Archambeau et al. 2011; Titsias and Lázaro-Gredilla 2011]. Different from the above global learning methods, some multi-task local learning algorithms are proposed in [Zhang 2013] to extend the k-Nearest-Neighbor algorithm and the kernel regression method. Moreover, to improve the interpretability, the multi-task feature selection methods [Obozinski et al. 2006; Obozinski1 et al. 2010; Zhang et al. 2010] are to select one subset of the original features by utilizing some sparsity-inducing priors, e.g., l_1/l_p norm ($p > 1$). Most of the above methods focus on *symmetric* multi-task learning, but there also exist some previous works that study *asymmetric* multi-task learning [Xue et al. 2007] or transfer learning [Raina et al. 2006; Kienzle and Chellapilla 2006; Eaton et al. 2008; Zhang and Yeung 2010c; 2012].

Since multi-task learning seeks to improve the performance of a task with the help of other related tasks, a central issue is to characterize the relationships between tasks accurately. Given the training data in multiple tasks, there are two important aspects that distinguish between different methods for characterizing the task relationships. The first aspect is on *what* task relationships can be represented by a method. Generally speaking there are three types of pairwise task relationships: positive task correlation, negative task correlation, and task unrelatedness (corresponding to outlier tasks). Positive task correlation is very useful for characterizing task relationships since similar tasks are likely to have similar model parameters. For negative task correlation, since the model parameters of two tasks with negative correlation are more likely to be dissimilar, knowing that two tasks are negatively correlated can help to reduce the search space of the model parameters. As for task unrelatedness, identifying outlier tasks can prevent them from impairing the performance of other tasks since outlier tasks are unrelated to other tasks. The second aspect is on *how* to obtain the relationships, either from the model assumption or automatically learned from data. Obviously, learning the task relationships from data automatically is the more favorable option because the model assumption adopted may be incorrect and, even worse, it is not easy to verify the correctness of the assumption from data.

Multi-layered feedforward neural networks and multi-task feature learning assume that all tasks share the same representation without actually learning the task relationships from data automatically. Moreover, they do not consider negative task correlation and are not robust against outlier tasks. The regularization methods in [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Kato et al. 2008] assume that the task relationships are given and then utilize this prior knowledge to learn the model parameters. The task clustering methods in [Thrun and O'Sullivan 1996; Bakker and Heskes 2003; Xue et al. 2007; Jacob et al. 2008] may be viewed as a way to learn the task relationships from data. Similar tasks will be grouped into the same task cluster and outlier tasks will be grouped separately, making these methods more robust against outlier tasks. However, they are *local* methods in the sense that only similar tasks within the same task cluster can interact to help each other, thus ignoring negative task correlation which may exist between tasks residing in different clusters. On the other hand, a multi-task learning method based on Gaussian process (GP) [Bonilla et al. 2007] provides a *global* approach to model and learn task relationships in the form of a task covariance matrix. A task covariance matrix can model all three types of task relationships: positive task correlation, negative task correlation, and task unrelatedness. However, although this method provides a powerful way to model task relationships, learning of the task covariance matrix gives rise to a non-convex optimization problem which is sensitive to parameter initialization. When the number of tasks is large, the authors proposed to use low-rank approximation [Bonilla et al. 2007] which will then weaken the expressive power of the task covariance matrix. Moreover, since the method is based on GP, scaling it to large data sets poses a serious computational challenge.

Our goal in this paper is to inherit the advantages of [Bonilla et al. 2007] while overcoming its disadvantages. Specifically, we propose a regularization approach, called *multi-task relationship learning* (MTRL), which also models the relationships between tasks in a nonparametric manner as a task covariance matrix. By utilizing a matrix-variate normal distribution [Gupta and Nagar 2000] as a prior on the model parameters of all tasks, we obtain a convex objective function which allows us to learn the model parameters and the task relationships simultaneously under the regularization framework. This model can be viewed as a generalization of the regularization framework for single-task learning to the multi-task setting. For efficiency, we use an alternating optimization method in which each subproblem is a convex problem. We study MTRL in the symmetric multi-task learning setting and then generalize it to the asymmetric setting as well. We discuss some variants of the regularization approach to demonstrate the use of other priors for learning task relationships. Moreover, to gain more insight into our model, we also study the relationships between MTRL and some existing multi-task learning methods, such as [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Kato et al. 2008; Jacob et al. 2008; Bonilla et al. 2007], showing that the regularized methods in [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Kato et al. 2008; Jacob et al. 2008] can be viewed as special cases of MTRL and the multi-task GP model in [Bonilla et al. 2007] and multi-task feature learning [Argyriou et al. 2008] are related to our model.

The rest of this paper is organized as follows. We present MTRL in Section 2. The relationships between MTRL and some existing multi-task learning methods are analyzed in Section 3. Section 4 reports experimental results based on some benchmark data sets. Concluding remarks are given in the final section.[1]

2. MULTI-TASK RELATIONSHIP LEARNING

Suppose we are given m learning tasks $\{T_i\}_{i=1}^m$. For the ith task T_i, the training set $\mathcal{D}_i$ consists of n_i data points $(\mathbf{x}_j^i, y_j^i)$, $j = 1, \ldots, n_i$, with $\mathbf{x}_j^i \in \mathbb{R}^d$ and its corresponding output $y_j^i \in \mathbb{R}$ if it is a regression problem and $y_j^i \in \{-1, 1\}$ if it is a binary classification problem. The linear function for T_i is defined as $f_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + b_i$.

2.1 Objective Function

The likelihood for y_j^i given $\mathbf{x}_j^i$, $\mathbf{w}_i$, b_i and ε_i is:

$$y_j^i \mid \mathbf{x}_j^i, \mathbf{w}_i, b_i, \varepsilon_i \sim \mathcal{N}(\mathbf{w}_i^T\mathbf{x}_j^i + b_i, \varepsilon_i^2), \tag{1}$$

where $\mathcal{N}(\mathbf{m}, \mathbf{\Sigma})$ denotes the multivariate (or univariate) normal distribution with mean $\mathbf{m}$ and covariance matrix (or variance) $\mathbf{\Sigma}$.

The prior on $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$ is defined as

$$\mathbf{W} \mid \epsilon_i \sim \left(\prod_{i=1}^m \mathcal{N}(\mathbf{w}_i \mid \mathbf{0}_d, \epsilon_i^2\mathbf{I}_d)\right) q(\mathbf{W}), \tag{2}$$

where $\mathbf{I}_d$ is the $d \times d$ identity matrix and $\mathbf{0}_d$ is the $d \times 1$ zero vector. The first term of the prior on $\mathbf{W}$ is to penalize the complexity of each column of $\mathbf{W}$ separately and the second term is to model the structure of $\mathbf{W}$. Since $\mathbf{W}$ is a matrix variable, it is natural to use a matrix-variate distribution [Gupta and Nagar 2000] to model it. Here we use the matrix-variate normal distribution for $q(\mathbf{W})$. More specifically,

$$q(\mathbf{W}) = \mathcal{MN}_{d\times m}(\mathbf{W} \mid \mathbf{0}_{d\times m}, \mathbf{I}_d \otimes \mathbf{\Omega}) \tag{3}$$

where $\mathbf{0}_{d\times m}$ is the $d\times m$ zero matrix and $\mathcal{MN}_{d\times m}(\mathbf{M}, \mathbf{A}\otimes\mathbf{B})$ denotes the matrix-variate normal distribution whose probability density function is defined as $p(\mathbf{X} \mid \mathbf{M}, \mathbf{A}, \mathbf{B}) = \frac{\exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{A}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{B}^{-1}(\mathbf{X}-\mathbf{M})^T\right)\right)}{(2\pi)^{md/2}|\mathbf{A}|^{m/2}|\mathbf{B}|^{d/2}}$ with mean $\mathbf{M} \in \mathbb{R}^{d\times m}$, row covariance matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ and column covariance matrix $\mathbf{B} \in \mathbb{R}^{m\times m}$. For the prior in Eq. (3), the row covariance matrix $\mathbf{I}_d$ models the relationships between features and the column covariance matrix $\mathbf{\Omega}$ models the relationships between different $\mathbf{w}_i$'s. In other words, $\mathbf{\Omega}$ models the relationships between tasks.

[1]An abridged version [Zhang and Yeung 2010a] of this paper was published in UAI 2010.

When there is only one task and $\mathbf{\Omega}$ is given as a positive scalar value, our model will degenerate to the probabilistic model for regularized least-squares regression and least-squares SVM [Gestel et al. 2004]. So our model can be viewed as a generalization of the model for single-task learning. However, unlike single-task learning, $\mathbf{\Omega}$ cannot be given *a priori* for most multi-task learning applications and so we seek to estimate it from data automatically.

It follows that the posterior distribution for $\mathbf{W}$, which is proportional to the product of the prior and the likelihood function [Bishop 2006], is given by:

$$p(\mathbf{W} \mid \mathbf{X}, \mathbf{y}, \mathbf{b}, \boldsymbol{\varepsilon}, \boldsymbol{\epsilon}, \mathbf{\Omega}) \propto p(\mathbf{y} \mid \mathbf{X}, \mathbf{W}, \mathbf{b}, \boldsymbol{\varepsilon})\, p(\mathbf{W} \mid \boldsymbol{\epsilon}, \mathbf{\Omega}), \tag{4}$$

where $\mathbf{y} = (y_1^1, \ldots, y_{n_1}^1, \ldots, y_1^m, \ldots, y_{n_m}^m)^T$, $\mathbf{X}$ denotes the data matrix of all data points in all tasks, and $\mathbf{b} = (b_1, \ldots, b_m)^T$. Taking the negative logarithm of Eq. (4) and combining with Eqs. (1)–(3), we obtain the *maximum a posteriori* (MAP) estimation of $\mathbf{W}$ and the *maximum likelihood estimation* (MLE) of $\mathbf{b}$ and $\mathbf{\Omega}$ by solving the following problem:

$$\min_{\mathbf{W}, \mathbf{b}, \mathbf{\Omega} \succeq \mathbf{0}} \sum_{i=1}^{m} \frac{1}{\varepsilon_i^2} \sum_{j=1}^{n_i} (y_j^i - \mathbf{w}_i^T \mathbf{x}_j^i - b_i)^2 + \sum_{i=1}^{m} \frac{1}{\epsilon_i^2} \mathbf{w}_i^T \mathbf{w}_i + \mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) + d \ln |\mathbf{\Omega}|, \tag{5}$$

where $\mathrm{tr}(\cdot)$ denotes the trace of a square matrix, $|\cdot|$ denotes the determinant of a square matrix, and $\mathbf{A} \succeq \mathbf{0}$ means that the matrix $\mathbf{A}$ is positive semidefinite (PSD). Here the PSD constraint on $\mathbf{\Omega}$ holds due to the fact that $\mathbf{\Omega}$ is defined as a task covariance matrix. For simplicity of discussion, we assume that $\varepsilon = \varepsilon_i$ and $\epsilon = \epsilon_i, \forall i = 1, \ldots, m$. The effect of the last term in problem (5) is to penalize the complexity of $\mathbf{\Omega}$. However, as we will see later, the first three terms in problem (5) are jointly convex with respect to all variables but the last term is concave since $-\ln |\mathbf{\Omega}|$ is a convex function with respect to $\mathbf{\Omega}$, according to [Boyd and Vandenberghe 2004]. Moreover, for kernel extension, we have no idea about d which may even be infinite after feature mapping, making problem (5) difficult to optimize. In the following, we first present a useful lemma which will be used later and present a proof for this well-known result to make this article self-contained.

LEMMA 1. *For any $m \times m$ PSD matrix $\mathbf{\Omega}$,* $\ln |\mathbf{\Omega}| \le \mathrm{tr}(\mathbf{\Omega}) - m$.

Proof:
We denote the eigenvalues of $\mathbf{\Omega}$ by $e_1, \ldots, e_m$. Then $\ln |\mathbf{\Omega}| = \sum_{i=1}^m \ln e_i$ and $\mathrm{tr}(\mathbf{\Omega}) = \sum_{i=1}^m e_i$. Due to the concavity of the logarithm function, we can obtain

$$\ln x \le \ln 1 + \frac{1}{1}(x - 1) = x - 1$$

by applying the first-order condition. Then

$$\ln |\mathbf{\Omega}| = \sum_{i=1}^m \ln e_i \le \sum_{i=1}^m e_i - m = \mathrm{tr}(\mathbf{\Omega}) - m.$$

This proves the lemma. ■

Based on Lemma 1, we can relax the optimization problem (5) as

$$\min_{\mathbf{W},\mathbf{b},\mathbf{\Omega}\succeq\mathbf{0}} \sum_{i=1}^{m}\frac{1}{\varepsilon^2}\sum_{j=1}^{n_i}(y_j^i-\mathbf{w}_i^T\mathbf{x}_j^i-b_i)^2+\sum_{i=1}^{m}\frac{1}{\epsilon^2}\mathbf{w}_i^T\mathbf{w}_i+\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)+d\,\mathrm{tr}(\mathbf{\Omega}). \tag{6}$$

However, the last term in problem (6) is still related to the data dimensionality d which usually cannot be estimated accurately in kernel methods. So we incorporate the last term into the constraint, leading to the following problem

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b},\mathbf{\Omega}} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_j^i-\mathbf{w}_i^T\mathbf{x}_j^i-b_i)^2+\frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T)+\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)\\ \text{s.t.} \quad & \mathbf{\Omega}\succeq\mathbf{0}\\ & \mathrm{tr}(\mathbf{\Omega})\le c,\end{aligned} \tag{7}$$

where $\lambda_1=\frac{2\varepsilon^2}{\epsilon^2}$ and $\lambda_2=2\varepsilon^2$ are regularization parameters. By using the method of Lagrange multipliers, it is easy to show that problems (6) and (7) are equivalent. Here we simply set $c=1$.

The first term in (7) measures the empirical loss on the training data, the second term penalizes the complexity of $\mathbf{W}$, and the third term measures the relationships between all tasks based on $\mathbf{W}$ and $\mathbf{\Omega}$.

To avoid the task imbalance problem in which one task has so many data points that it dominates the empirical loss, we modify problem (7) as

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b},\mathbf{\Omega}} \quad & \sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_j^i-\mathbf{w}_i^T\mathbf{x}_j^i-b_i)^2+\frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T)+\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)\\ \text{s.t.} \quad & \mathbf{\Omega}\succeq\mathbf{0}\\ & \mathrm{tr}(\mathbf{\Omega})\le 1.\end{aligned} \tag{8}$$

Note that (8) is a semi-definite programming (SDP) problem which is computationally demanding. In what follows, we will present an efficient algorithm for solving it.

2.2 Optimization Procedure

We first prove the joint convexity of problem (8) with respect to all variables.

THEOREM 1. *Problem (8) is jointly convex with respect to* $\mathbf{W}$, $\mathbf{b}$ *and* $\mathbf{\Omega}$.

Proof:
It is easy to see that the first two terms in the objective function of problem (8) are jointly convex with respect to all variables and the constraints in (8) are also convex with respect to all variables. We rewrite the third term in the objective function as

$$\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)=\sum_t \mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T,$$

where $\mathbf{W}(t,:)$ denotes the tth row of $\mathbf{W}$. $\mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T$ is called a matrix fractional function in Example 3.4 (page 76) of [Boyd and Vandenberghe 2004] and it is proved to be a jointly convex function with respect to $\mathbf{W}(t,:)$ and $\mathbf{\Omega}$ there when $\mathbf{\Omega}$ is a PSD matrix (which is satisfied by the first constraint of (8)). Even though $\mathbf{b}$ and $\mathbf{W}(\tilde{t},:)$, where $\mathbf{W}(\tilde{t},:)$

is a submatrix of $\mathbf{W}$ by eliminating the tth row, do not appear in $\mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T$, it is easy to show that $\mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T$ is jointly convex with respect to $\mathbf{W}$, $\mathbf{\Omega}$ and $\mathbf{b}$. This is because the Hessian matrix of $\mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T$ with respect to $\mathbf{W}$, $\mathbf{\Omega}$ and $\mathbf{b}$, taking the form of $\begin{pmatrix} \mathbf{H} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ after some permutation where $\mathbf{0}$ denotes a zero matrix of appropriate size and $\mathbf{H}$ is the PSD Hessian matrix of $\mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T$ with respect to $\mathbf{W}(t,:)$ and $\mathbf{\Omega}$, is also a PSD matrix. Because the summation operation can preserve convexity according to the analysis on page 79 of [Boyd and Vandenberghe 2004], $\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) = \sum_t \mathbf{W}(t,:)\mathbf{\Omega}^{-1}\mathbf{W}(t,:)^T$ is jointly convex with respect to $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{\Omega}$. So the objective function and the constraints in problem (8) are jointly convex with respect to all variables and hence problem (8) is jointly convex. ■

Even though the optimization problem (8) is jointly convex with respect to $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{\Omega}$, it is not easy to optimize the objective function with respect to all the variables simultaneously. Here we propose an alternating method to solve the problem more efficiently. Specifically, we first optimize the objective function with respect to $\mathbf{W}$ and $\mathbf{b}$ when $\mathbf{\Omega}$ is fixed, and then optimize it with respect to $\mathbf{\Omega}$ when $\mathbf{W}$ and $\mathbf{b}$ are fixed. This procedure is repeated until convergence. In what follows, we will present the two subproblems separately.

Optimizing w.r.t. $\mathbf{W}$ and $\mathbf{b}$ when $\mathbf{\Omega}$ is fixed

When $\mathbf{\Omega}$ is given and fixed, the optimization problem for finding $\mathbf{W}$ and $\mathbf{b}$ is an unconstrained convex optimization problem. The optimization problem can be stated as:

$$\min_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} (y_j^i - \mathbf{w}_i^T \mathbf{x}_j^i - b_i)^2 + \frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T). \tag{9}$$

To facilitate a kernel extension to be given later for the general nonlinear case, we reformulate the optimization problem into a dual form by first expressing problem (9) as a constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{b},\{\varepsilon_j^i\}} \quad & \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) \\ \text{s.t.} \quad & y_j^i - (\mathbf{w}_i^T \mathbf{x}_j^i + b_i) = \varepsilon_j^i \quad \forall i,j. \end{aligned} \tag{10}$$

The Lagrangian of problem (10) is given by

$$\begin{aligned} G = & \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) \\ & + \sum_{i=1}^{m} \sum_{j=1}^{n_i} \alpha_j^i \left[y_j^i - (\mathbf{w}_i^T \mathbf{x}_j^i + b_i) - \varepsilon_j^i \right]. \end{aligned} \tag{11}$$

We calculate the gradients of G with respect to $\mathbf{W}$, b_i and ε_j^i and set them to 0 to obtain

$$\begin{aligned}
\frac{\partial G}{\partial \mathbf{W}} &= \mathbf{W}(\lambda_1 \mathbf{I}_m + \lambda_2 \mathbf{\Omega}^{-1}) - \sum_{i=1}^{m}\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i \mathbf{e}_i^T = 0 \\
\Rightarrow \mathbf{W} &= \sum_{i=1}^{m}\sum_{j=1}^{n_i} \alpha_j^i \mathbf{x}_j^i \mathbf{e}_i^T \mathbf{\Omega}(\lambda_1 \mathbf{\Omega} + \lambda_2 \mathbf{I}_m)^{-1} \\
\frac{\partial G}{\partial b_i} &= -\sum_{j=1}^{n_i} \alpha_j^i = 0 \\
\frac{\partial G}{\partial \varepsilon_j^i} &= \frac{2}{n_i}\varepsilon_j^i - \alpha_j^i = 0,
\end{aligned}$$

where $\mathbf{e}_i$ is the ith column vector of $\mathbf{I}_m$. Combining the above equations, we obtain the following linear system:

$$\begin{pmatrix} \mathbf{K} + \frac{1}{2}\mathbf{\Lambda} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{0}_{m\times m} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_{m\times 1} \end{pmatrix}, \tag{12}$$

where $k_{MT}(\mathbf{x}_{j_1}^{i_1}, \mathbf{x}_{j_2}^{i_2}) = \mathbf{e}_{i_1}^T \mathbf{\Omega}(\lambda_1 \mathbf{\Omega} + \lambda_2 \mathbf{I}_m)^{-1} \mathbf{e}_{i_2} (\mathbf{x}_{j_1}^{i_1})^T \mathbf{x}_{j_2}^{i_2}$ is the linear multi-task kernel, $\mathbf{K}$ is the kernel matrix defined on all data points for all tasks using the linear multi-task kernel, $\boldsymbol{\alpha} = (\alpha_1^1, \ldots, \alpha_{n_m}^m)^T$, $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal element is equal to n_i if the corresponding data point belongs to the ith task, $N_i = \sum_{j=1}^{i} n_j$, and $\mathbf{M}_{12} = \mathbf{M}_{21}^T = (\mathbf{e}_{N_0+1}^{N_1}, \mathbf{e}_{N_1+1}^{N_2}, \ldots, \mathbf{e}_{N_{m-1}+1}^{N_m})$ where $\mathbf{e}_q^p$ is a zero vector with only the elements whose indices are in $[q, p]$ being equal to 1.

When the total number of data points for all tasks is very large, the computational cost required to solve the linear system (12) directly will be very high. In this situation, we can use another optimization method to solve it. It is easy to show that the dual form of problem (10) can be formulated as:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & h(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T \tilde{\mathbf{K}} \boldsymbol{\alpha} - \sum_{i,j} \alpha_j^i y_j^i \\
\text{s.t.} \quad & \sum_j \alpha_j^i = 0 \quad \forall i,
\end{aligned} \tag{13}$$

where $\tilde{\mathbf{K}} = \mathbf{K} + \frac{1}{2}\mathbf{\Lambda}$. Note that it is similar to the dual form of least-squares SVM [Gestel et al. 2004] except that there is only one constraint in least-squares SVM but here there are m constraints with each constraint corresponding to one task. Here we use an SMO algorithm similar to that for least-squares SVM [Keerthi and Shevade 2003]. The detailed SMO algorithm is given in Appendix A.

Optimizing w.r.t. $\mathbf{\Omega}$ when $\mathbf{W}$ and $\mathbf{b}$ are fixed

When $\mathbf{W}$ and $\mathbf{b}$ are fixed, the optimization problem for finding $\mathbf{\Omega}$ becomes

$$\begin{aligned}
\min_{\mathbf{\Omega}} \quad & \text{tr}(\mathbf{\Omega}^{-1}\mathbf{W}^T\mathbf{W}) \\
\text{s.t.} \quad & \mathbf{\Omega} \succeq \mathbf{0} \\
& \text{tr}(\mathbf{\Omega}) \leq 1.
\end{aligned} \tag{14}$$

Then we have

$$\begin{aligned}\mathrm{tr}(\mathbf{\Omega}^{-1}\mathbf{A}) &\geq \mathrm{tr}(\mathbf{\Omega}^{-1}\mathbf{A})\mathrm{tr}(\mathbf{\Omega})\\ &= \mathrm{tr}((\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{A}^{\frac{1}{2}})(\mathbf{A}^{\frac{1}{2}}\mathbf{\Omega}^{-\frac{1}{2}}))\mathrm{tr}(\mathbf{\Omega}^{\frac{1}{2}}\mathbf{\Omega}^{\frac{1}{2}})\\ &\geq (\mathrm{tr}(\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{A}^{\frac{1}{2}}\mathbf{\Omega}^{\frac{1}{2}}))^2 = (\mathrm{tr}(\mathbf{A}^{\frac{1}{2}}))^2,\end{aligned}$$

where $\mathbf{A} = \mathbf{W}^T\mathbf{W}$. The first inequality holds because of the last constraint in problem (14), and the last inequality holds because of the Cauchy-Schwarz inequality for the Frobenius norm. Moreover, $\mathrm{tr}(\mathbf{\Omega}^{-1}\mathbf{A})$ attains its minimum value $(\mathrm{tr}(\mathbf{A}^{\frac{1}{2}}))^2$ if and only if $\mathbf{\Omega}^{-\frac{1}{2}}\mathbf{A}^{\frac{1}{2}} = a\mathbf{\Omega}^{\frac{1}{2}}$ for some constant a and $\mathrm{tr}(\mathbf{\Omega}) = 1$. So we can get the analytical solution $\mathbf{\Omega} = \frac{(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}}{\mathrm{tr}((\mathbf{W}^T\mathbf{W})^{\frac{1}{2}})}$. By plugging the analytical solution of $\mathbf{\Omega}$ into the original problem (8), we can see the last term in the objective function is related to the trace norm.

We set the initial value of $\mathbf{\Omega}$ to $\frac{1}{m}\mathbf{I}_m$ which corresponds to the assumption that all tasks are unrelated initially.

After learning the optimal values of $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{\Omega}$, we can make prediction for a new data point. Given a test data point $\mathbf{x}_\star^i$ for task T_i, the predictive output $y_\star^i$ is given by

$$y_\star^i = \sum_{p=1}^{m}\sum_{q=1}^{n_p}\alpha_q^p k_{MT}(\mathbf{x}_q^p, \mathbf{x}_\star^i) + b_i.$$

2.3 Incorporation of New Tasks

The method described above can only learn from multiple tasks simultaneously which is the setting for symmetric multi-task learning. In asymmetric multi-task learning, when a new task arrives, we could add the data for this new task to the training set and then train a new model from scratch for the $m+1$ tasks using the above method. However, it is undesirable to incorporate new tasks in this way due to the high computational cost incurred. Here we introduce an algorithm for asymmetric multi-task learning which is more efficient.

For notational simplicity, let $\tilde{m}$ denote $m+1$. We denote the new task by $T_{\tilde{m}}$ and its training set by $\mathcal{D}_{\tilde{m}} = \{(\mathbf{x}_j^{\tilde{m}}, y_j^{\tilde{m}})\}_{j=1}^{n_{\tilde{m}}}$. The task covariances between $T_{\tilde{m}}$ and the m existing tasks are represented by the vector $\boldsymbol{\omega}_{\tilde{m}} = (\omega_{\tilde{m},1}, \ldots, \omega_{\tilde{m},m})^T$ and the task variance for $T_{\tilde{m}}$ is defined as σ. Thus the augmented task covariance matrix for the $m+1$ tasks is:

$$\tilde{\mathbf{\Omega}} = \begin{pmatrix} (1-\sigma)\mathbf{\Omega} & \boldsymbol{\omega}_{\tilde{m}} \\ \boldsymbol{\omega}_{\tilde{m}}^T & \sigma \end{pmatrix},$$

where $\mathbf{\Omega}$ is scaled by $(1-\sigma)$ to make $\tilde{\mathbf{\Omega}}$ satisfy the constraint $\mathrm{tr}(\tilde{\mathbf{\Omega}}) = 1$.[2] The linear function for task T_{m+1} is defined as $f_{m+1}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$.

With $\mathbf{W}_m = (\mathbf{w}_1, \ldots, \mathbf{w}_m)$ and $\mathbf{\Omega}$ at hand, the optimization problem can be formulated

[2]Due to the analysis in the above section, we find that the optimal solution of $\mathbf{\Omega}$ satisfies $\mathrm{tr}(\mathbf{\Omega}) = 1$. So here we directly apply this optimality condition.

as follows:

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\omega}_{\tilde{m}},\sigma} \quad & \frac{1}{n_{\tilde{m}}}\sum_{j=1}^{n_{\tilde{m}}}(y_j^{\tilde{m}} - \mathbf{w}^T\mathbf{x}_j^{\tilde{m}} - b)^2 + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}_{\tilde{m}}\tilde{\boldsymbol{\Omega}}^{-1}\mathbf{W}_{\tilde{m}}^T) \\ \text{s.t.} \quad & \tilde{\boldsymbol{\Omega}} \succeq \mathbf{0}, \end{aligned} \tag{15}$$

where $\|\cdot\|_2$ denotes the 2-norm of a vector and $\mathbf{W}_{\tilde{m}} = (\mathbf{W}_m, \mathbf{w})$. Problem (15) is an SDP problem. Here we assume $\boldsymbol{\Omega}$ is positive definite.[3] So if the constraint in (15) holds, then according to the Schur complement [Boyd and Vandenberghe 2004], this constraint is equivalent to $\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} \le \sigma - \sigma^2$. Thus problem (15) becomes

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\omega}_{\tilde{m}},\sigma} \quad & \frac{1}{n_{\tilde{m}}}\sum_{j=1}^{n_{\tilde{m}}}(y_j^{\tilde{m}} - \mathbf{w}^T\mathbf{x}_j^{\tilde{m}} - b)^2 + \frac{\lambda_1}{2}\|\mathbf{w}\|_2^2 + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}_{\tilde{m}}\tilde{\boldsymbol{\Omega}}^{-1}\mathbf{W}_{\tilde{m}}^T) \\ \text{s.t.} \quad & \boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} \le \sigma - \sigma^2. \end{aligned} \tag{16}$$

Similar to Theorem 1, it is easy to show that this is a jointly convex problem with respect to all variables and thus we can also use an alternating method to solve it.

When using the alternating method to optimize with respect to $\mathbf{w}$ and b, from the block matrix inversion formula, we can get

$$\tilde{\boldsymbol{\Omega}}^{-1} = \begin{pmatrix} \left((1-\sigma)\boldsymbol{\Omega} - \frac{1}{\sigma}\boldsymbol{\omega}_{\tilde{m}}\boldsymbol{\omega}_{\tilde{m}}^T\right)^{-1} & -\frac{1}{(1-\sigma)\sigma'}\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} \\ -\frac{1}{(1-\sigma)\sigma'}\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1} & \frac{1}{\sigma'} \end{pmatrix},$$

where $\sigma' = \sigma - \frac{1}{1-\sigma}\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}}$. Then the optimization problem is formulated as

$$\min_{\mathbf{w},b} \frac{1}{n_{\tilde{m}}}\sum_{j=1}^{n_{\tilde{m}}}(y_j^{\tilde{m}} - \mathbf{w}^T\mathbf{x}_j^{\tilde{m}} - b)^2 + \frac{\lambda_1'}{2}\|\mathbf{w}\|_2^2 - \lambda_2'\mathbf{u}^T\mathbf{w}, \tag{17}$$

where $\lambda_1' = \lambda_1 + \frac{\lambda_2}{\sigma'}$, $\lambda_2' = \frac{\lambda_2}{(1-\sigma)\sigma'}$ and $\mathbf{u} = \mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}}$. We first investigate the physical meaning of problem (17) before giving the detailed optimization procedure. We rewrite problem (17) as

$$\min_{\mathbf{w},b} \frac{1}{n_{\tilde{m}}}\sum_{j=1}^{n_{\tilde{m}}}(y_j^{\tilde{m}} - \mathbf{w}^T\mathbf{x}_j^{\tilde{m}} - b)^2 + \frac{\lambda_1'}{2}\|\mathbf{w} - \frac{\lambda_2'}{\lambda_1'}\mathbf{u}\|_2^2,$$

which enforces $\mathbf{w}$ to approach the scaled $\mathbf{u}$ as *a priori* information. This problem is similar to that of [Wu and Dietterich 2004], but there exist crucial differences between them. For example, the model in [Wu and Dietterich 2004] can only handle the situation that $m = 1$ but our method can handle the situations $m = 1$ and $m > 1$ in a unified framework. Moreover, $\mathbf{u}$ also has explicit physical meaning. Considering a special case when $\boldsymbol{\Omega} \propto \mathbf{I}_m$ which means the existing tasks are uncorrelated. We can show that $\mathbf{u}$ is proportional to a weighted combination of the model parameters learned from the existing tasks where each combination weight is the task covariance between an existing task and the new task. This is in line with our intuition that a positively correlated existing task has a large weight on the prior of $\mathbf{w}$, an outlier task has negligible contribution and a negatively correlated task

[3]When $\boldsymbol{\Omega}$ is positive semi-definite, the optimization procedure is similar.

even has opposite effect. We reformulate the problem (17) as a constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{w},\mathbf{b},\{\varepsilon_j\}} \quad & \frac{1}{n_{\tilde{m}}}\sum_{j=1}^{n_{\tilde{m}}}\varepsilon_j^2 + \frac{\lambda_1'}{2}\|\mathbf{w}\|_2^2 - \lambda_2'\mathbf{u}^T\mathbf{w} \\ \text{s.t.} \quad & y_j^{\tilde{m}} - \mathbf{w}^T\mathbf{x}_j^{\tilde{m}} - b = \varepsilon_j \quad \forall j. \end{aligned}$$

The Lagrangian is given by

$$G' = \frac{1}{n_{\tilde{m}}}\sum_{j=1}^{n_{\tilde{m}}}\varepsilon_j^2 + \frac{\lambda_1'}{2}\|\mathbf{w}\|_2^2 - \lambda_2'\mathbf{u}^T\mathbf{w} + \sum_{j=1}^{n_{\tilde{m}}}\beta_j\left[y_j^{\tilde{m}} - \mathbf{w}^T\mathbf{x}_j^{\tilde{m}} - b - \varepsilon_j\right].$$

We calculate the gradients of G' with respect to $\mathbf{w}$, b and ε_j and set them to 0 to obtain

$$\frac{\partial G'}{\partial \mathbf{w}} = \lambda_1'\mathbf{w} - \lambda_2'\mathbf{u} - \sum_{j=1}^{n_{\tilde{m}}}\beta_j\mathbf{x}_j^{\tilde{m}} = 0 \tag{18}$$

$$\frac{\partial G'}{\partial b} = -\sum_{j=1}^{n_{\tilde{m}}}\beta_j = 0$$

$$\frac{\partial G'}{\partial \varepsilon_j} = \frac{2}{n_{\tilde{m}}}\varepsilon_j - \beta_j = 0.$$

Combining the above equations, we obtain the following linear system:

$$\begin{pmatrix} \frac{1}{\lambda_1'}\mathbf{K}' + \frac{n_{\tilde{m}}}{2}\mathbf{I}_{n_{\tilde{m}}} & \mathbf{1}_{n_{\tilde{m}}} \\ \mathbf{1}_{n_{\tilde{m}}}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\beta} \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{y}' - \frac{\lambda_2'}{\lambda_1'}(\mathbf{X}')^T\mathbf{u} \\ 0 \end{pmatrix}, \tag{19}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{n_{\tilde{m}}})^T$, $\mathbf{1}_p$ is the $p \times 1$ vector of all ones, $\mathbf{X}' = (\mathbf{x}_1^{\tilde{m}}, \ldots, \mathbf{x}_{n_{\tilde{m}}}^{\tilde{m}})$, $\mathbf{y}' = (y_1^{\tilde{m}}, \ldots, y_{n_{\tilde{m}}}^{\tilde{m}})^T$ and $\mathbf{K}' = (\mathbf{X}')^T\mathbf{X}'$ is the linear kernel matrix on the training set of the new task.

When optimizing with respect to $\boldsymbol{\omega}_{m+1}$ and σ, the optimization problem is formulated as

$$\begin{aligned} \min_{\boldsymbol{\omega}_{\tilde{m}},\sigma,\tilde{\boldsymbol{\Omega}}} \quad & \mathrm{tr}(\mathbf{W}_{\tilde{m}}\tilde{\boldsymbol{\Omega}}^{-1}\mathbf{W}_{\tilde{m}}^T) \\ \text{s.t.} \quad & \boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} \le \sigma - \sigma^2 \\ & \tilde{\boldsymbol{\Omega}} = \begin{pmatrix} (1-\sigma)\boldsymbol{\Omega} & \boldsymbol{\omega}_{\tilde{m}} \\ \boldsymbol{\omega}_{\tilde{m}}^T & \sigma \end{pmatrix}. \end{aligned} \tag{20}$$

We impose a constraint as $\mathbf{W}_{\tilde{m}}\tilde{\boldsymbol{\Omega}}^{-1}\mathbf{W}_{\tilde{m}}^T \preceq \frac{1}{t}\mathbf{I}_d$ and the objective function becomes $\min \frac{1}{t}$ which is equivalent to $\min -t$ since $t > 0$. Using the Schur complement, we can get

$$\mathbf{W}_{\tilde{m}}\tilde{\boldsymbol{\Omega}}^{-1}\mathbf{W}_{\tilde{m}}^T \preceq \frac{1}{t}\mathbf{I}_d \Longleftrightarrow \begin{pmatrix} \tilde{\boldsymbol{\Omega}} & \mathbf{W}_{\tilde{m}}^T \\ \mathbf{W}_{\tilde{m}} & \frac{1}{t}\mathbf{I}_d \end{pmatrix} \succeq \mathbf{0}.$$

By using the Schur complement again, we get

$$\begin{pmatrix} \tilde{\boldsymbol{\Omega}} & \mathbf{W}_{\tilde{m}}^T \\ \mathbf{W}_{\tilde{m}} & \frac{1}{t}\mathbf{I}_d \end{pmatrix} \succeq \mathbf{0} \Longleftrightarrow \tilde{\boldsymbol{\Omega}} - t\mathbf{W}_{\tilde{m}}^T\mathbf{W}_{\tilde{m}} \succeq \mathbf{0}.$$

So problem (20) can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\omega}_{\tilde{m}},\sigma,\tilde{\boldsymbol{\Omega}},t} \quad & -t \\ \text{s.t.} \quad & \boldsymbol{\omega}_{\tilde{m}}^T \boldsymbol{\Omega}^{-1} \boldsymbol{\omega}_{\tilde{m}} \leq \sigma - \sigma^2 \\ & \tilde{\boldsymbol{\Omega}} = \begin{pmatrix} (1-\sigma)\boldsymbol{\Omega} & \boldsymbol{\omega}_{\tilde{m}} \\ \boldsymbol{\omega}_{\tilde{m}}^T & \sigma \end{pmatrix}. \\ & \tilde{\boldsymbol{\Omega}} - t\mathbf{W}_{\tilde{m}}^T \mathbf{W}_{\tilde{m}} \succeq \mathbf{0}, \end{aligned} \tag{21}$$

which is an SDP problem. In real applications, the number of tasks m is usually not very large and we can use a standard SDP solver to solve problem (21). Moreover, we may also reformulate problem (21) as a second-order cone programming (SOCP) problem [Lobo et al. 1998] which is more efficient than SDP when m is large. We will present the procedure in Appendix B.

In case two or more new tasks arrive together, the above formulation only needs to be modified slightly to accommodate all the new tasks simultaneously.

2.4 Kernel Extension

So far we have only considered the linear case for MTRL. In this section, we will apply the kernel trick to provide a nonlinear extension of the algorithm presented above.

The optimization problem for the kernel extension is essentially the same as that for the linear case, with the only difference being that the data point $\mathbf{x}_j^i$ is mapped to $\Phi(\mathbf{x}_j^i)$ in some reproducing kernel Hilbert space where $\Phi(\cdot)$ denotes the feature map. Then the corresponding kernel function $k(\cdot,\cdot)$ satisfies $k(\mathbf{x}_1,\mathbf{x}_2) = \Phi(\mathbf{x}_1)^T\Phi(\mathbf{x}_2)$.

For symmetric multi-task learning, we can also use an alternating method to solve the optimization problem. In the first step of the alternating method, we use the nonlinear multi-task kernel

$$k_{MT}(\mathbf{x}_{j_1}^{i_1}, \mathbf{x}_{j_2}^{i_2}) = \mathbf{e}_{i_1}^T \boldsymbol{\Omega}(\lambda_1\boldsymbol{\Omega} + \lambda_2\mathbf{I}_m)^{-1}\mathbf{e}_{i_2} k(\mathbf{x}_{j_1}^{i_1}, \mathbf{x}_{j_2}^{i_2}).$$

The rest is the same as the linear case. For the second step, the change needed is in the calculation of $\mathbf{W}^T\mathbf{W}$. Since

$$\mathbf{W} = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \alpha_j^i \Phi(\mathbf{x}_j^i)\mathbf{e}_i^T\boldsymbol{\Omega}(\lambda_1\boldsymbol{\Omega} + \lambda_2\mathbf{I}_m)^{-1}$$

which is similar to the representer theorem in single-task learning, we have

$$\mathbf{W}^T\mathbf{W} = \sum_{i,j}\sum_{p,q} \alpha_j^i \alpha_q^p k(\mathbf{x}_j^i, \mathbf{x}_q^p)(\lambda_1\boldsymbol{\Omega} + \lambda_2\mathbf{I}_m)^{-1}\boldsymbol{\Omega}\mathbf{e}_i\mathbf{e}_p^T\boldsymbol{\Omega}(\lambda_1\boldsymbol{\Omega} + \lambda_2\mathbf{I}_m)^{-1}. \tag{22}$$

In the asymmetric setting, when a new task arrives, we still use the alternating method to solve the problem. In the first step of the alternating method, the analytical solution (19) needs to calculate $(\Phi(\mathbf{X}'))^T\mathbf{u}$ where $\Phi(\mathbf{X}') = (\Phi(\mathbf{x}_1^{\tilde{m}}), \ldots, \Phi(\mathbf{x}_{n_{\tilde{m}}}^{\tilde{m}}))$ denotes the data matrix of the new task after feature mapping and $\mathbf{u} = \mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}}$. Since $\mathbf{W}_m$ is derived from symmetric multi-task learning, we get

$$\mathbf{W}_m = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \alpha_j^i \Phi(\mathbf{x}_j^i)\mathbf{e}_i^T\boldsymbol{\Omega}(\lambda_1\boldsymbol{\Omega} + \lambda_2\mathbf{I}_m)^{-1}$$

Then

$$(\Phi(\mathbf{X}'))^T\mathbf{u} = (\Phi(\mathbf{X}'))^T\mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} = \mathbf{M}\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}}$$

where

$$\begin{aligned}\mathbf{M} &= (\Phi(\mathbf{X}'))^T\mathbf{W}_m \\ &= (\Phi(\mathbf{X}'))^T\sum_{i=1}^{m}\sum_{j=1}^{n_i}\alpha_j^i\Phi(\mathbf{x}_j^i)\mathbf{e}_i^T\boldsymbol{\Omega}(\lambda_1\boldsymbol{\Omega}+\lambda_2\mathbf{I}_m)^{-1} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n_i}\alpha_j^i\tilde{\mathbf{k}}_j^i\mathbf{e}_i^T\boldsymbol{\Omega}(\lambda_1\boldsymbol{\Omega}+\lambda_2\mathbf{I}_m)^{-1}\end{aligned}$$

and $\tilde{\mathbf{k}}_j^i = \big(k(\mathbf{x}_j^i,\mathbf{x}_1^{\tilde{m}}),\ldots,k(\mathbf{x}_j^i,\mathbf{x}_{n_{\tilde{m}}}^{\tilde{m}})\big)^T$. In the second step of the alternating method, we need to calculate $\mathbf{W}_{\tilde{m}}^T\mathbf{W}_{\tilde{m}}$ where $\mathbf{W}_{\tilde{m}} = (\mathbf{W}_m,\mathbf{w})$. Following the notations in Appendix B, we denote $\mathbf{W}_{\tilde{m}}^T\mathbf{W}_{\tilde{m}}$ as $\mathbf{W}_{\tilde{m}}^T\mathbf{W}_{\tilde{m}} = \begin{pmatrix}\boldsymbol{\Psi}_{11} & \boldsymbol{\Psi}_{12}\\ \boldsymbol{\Psi}_{12}^T & \Psi_{22}\end{pmatrix}$ where $\boldsymbol{\Psi}_{11}\in\mathbb{R}^{m\times m}$, $\boldsymbol{\Psi}_{12}\in\mathbb{R}^{m\times 1}$ and $\Psi_{22}\in\mathbb{R}$. Then $\boldsymbol{\Psi}_{11} = \mathbf{W}_m^T\mathbf{W}_m$, $\boldsymbol{\Psi}_{12} = \mathbf{W}_m^T\mathbf{w}$ and $\Psi_{22} = \mathbf{w}^T\mathbf{w}$. It is easy to show that $\boldsymbol{\Psi}_{11}$ can be calculated as in Eq. (22) which only need to be computed once. Recall that

$$\mathbf{w} = \frac{\lambda_2'}{\lambda_1'}\mathbf{u} + \frac{1}{\lambda_1'}\Phi(\mathbf{X}')\boldsymbol{\beta} = \frac{\lambda_2'}{\lambda_1'}\mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{\lambda_1'}\Phi(\mathbf{X}')\boldsymbol{\beta}$$

from Eq. (18). So we can get

$$\begin{aligned}&\boldsymbol{\Psi}_{12}\\ &= \mathbf{W}_m^T\Big[\frac{\lambda_2'}{\lambda_1'}\mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{\lambda_1'}\Phi(\mathbf{X}')\boldsymbol{\beta}\Big]\\ &= \frac{\lambda_2'}{\lambda_1'}\boldsymbol{\Psi}_{11}\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{\lambda_1'}\mathbf{M}^T\boldsymbol{\beta}\end{aligned}$$

and

$$\begin{aligned}&\Psi_{22}\\ &= \Big\|\frac{\lambda_2'}{\lambda_1'}\mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{\lambda_1'}\Phi(\mathbf{X}')\boldsymbol{\beta}\Big\|_2^2\\ &= \frac{(\lambda_2')^2}{(\lambda_1')^2}\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\mathbf{W}_m^T\mathbf{W}_m\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{(\lambda_1')^2}\boldsymbol{\beta}^T\Phi(\mathbf{X}')^T\Phi(\mathbf{X}')\boldsymbol{\beta} + \frac{2\lambda_2'}{(\lambda_1')^2}\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\mathbf{W}_m^T\Phi(\mathbf{X}')\boldsymbol{\beta}\\ &= \frac{(\lambda_2')^2}{(\lambda_1')^2}\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\Psi}_{11}\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{(\lambda_1')^2}\boldsymbol{\beta}^T\mathbf{K}'\boldsymbol{\beta} + \frac{2\lambda_2'}{(\lambda_1')^2}\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\mathbf{M}^T\boldsymbol{\beta}\end{aligned}$$

where $\mathbf{K}'$ is the kernel matrix. In the testing phrase, when given a test data point $\mathbf{x}_\star$, the output can be calculated as

$$
\begin{aligned}
y_\star &= \mathbf{w}^T\Phi(\mathbf{x}_\star) + b \\
&= \Big(\frac{\lambda_2'}{\lambda_1'}\mathbf{W}_m\mathbf{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} + \frac{1}{\lambda_1'}\Phi(\mathbf{X}')\boldsymbol{\beta}\Big)^T\Phi(\mathbf{x}_\star) + b \\
&= \frac{\lambda_2'}{\lambda_1'}\boldsymbol{\omega}_{\tilde{m}}^T\mathbf{\Omega}^{-1}\mathbf{W}_m^T\Phi(\mathbf{x}_\star) + \frac{1}{\lambda_1'}\boldsymbol{\beta}^T\mathbf{k}_\star + b \\
&= \frac{\lambda_2'}{\lambda_1'}\boldsymbol{\omega}_{\tilde{m}}^T\mathbf{\Omega}^{-1}(\lambda_1\mathbf{\Omega}+\lambda_2\mathbf{I}_m)^{-1}\mathbf{\Omega}\sum_{i=1}^{m}\sum_{j=1}^{n_i}\alpha_j^i k(\mathbf{x}_j^i,\mathbf{x}_\star)\mathbf{e}_i + \frac{1}{\lambda_1'}\boldsymbol{\beta}^T\mathbf{k}_\star + b \\
&= \frac{\lambda_2'}{\lambda_1'}\boldsymbol{\omega}_{\tilde{m}}^T(\lambda_1\mathbf{\Omega}+\lambda_2\mathbf{I}_m)^{-1}\sum_{i=1}^{m}\sum_{j=1}^{n_i}\alpha_j^i k(\mathbf{x}_j^i,\mathbf{x}_\star)\mathbf{e}_i + \frac{1}{\lambda_1'}\boldsymbol{\beta}^T\mathbf{k}_\star + b,
\end{aligned}
$$

where $\mathbf{k}_\star = \big(k(\mathbf{x}_\star,\mathbf{x}_1^{\tilde{m}}),\ldots,k(\mathbf{x}_\star,\mathbf{x}_{n_{\tilde{m}}}^{\tilde{m}})\big)^T$.

2.5 Discussions

By replacing $\ln|\mathbf{\Omega}|$ with $\mathrm{tr}(\mathbf{\Omega})$, problem (6) is a convex relaxation of problem (5). It is easy to show that the optimal solution of $\mathbf{\Omega}$ in problem (5) is proportional to $\mathbf{W}^T\mathbf{W}$ and that in problem (6) is proportional to $(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}$ when $\mathbf{W}$ is given. We denote the singular value decomposition (SVD) of $\mathbf{W}$ as $\mathbf{W} = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$. So by reparameterization, the optimal solution of $\mathbf{\Omega}$ in problem (5) is proportional to $\mathbf{V}\mathbf{\Delta}^2\mathbf{V}^T$ and that in problem (6) is proportional to $\mathbf{V}\mathbf{\Delta}\mathbf{V}^T$. So the optimal solution of $\mathbf{\Omega}$ in problem (5) overemphasizes the right singular vectors in $\mathbf{V}$ with large singular values and neglects those with small singular values. Different from problem (5), the solution of $\mathbf{\Omega}$ in problem (6) depends on both large and small singular vectors in $\mathbf{V}$ and thus can utilize the full spectrum of $\mathbf{W}$ more effectively.

In some applications, there may exist prior knowledge about the relationships between some tasks, e.g., two tasks are more similar than other two tasks, some tasks are from the same task cluster, etc. It is easy to incorporate the prior knowledge by introducing additional constraints into problem (8). For example, if tasks T_i and T_j are more similar than tasks T_p and T_q, then the corresponding constraint can be represented as $\Omega_{ij} > \Omega_{pq}$; if we know that some tasks are from the same cluster, then we can enforce the covariances between these tasks very large while their covariances with other tasks very close to 0.

2.6 Some Variants

In our regularized model, the prior on $\mathbf{W}$ given in Eq. (2) is very general. Here we discuss some different choices for $q(\mathbf{W})$.

2.6.1 *Utilizing Other Matrix-Variate Normal Distributions.* When we choose another matrix-variate normal distribution for $q(\mathbf{W})$, such as $q(\mathbf{W}) = \mathcal{MN}_{d\times m}(\mathbf{W} \mid \mathbf{0}_{d\times m}, \mathbf{\Sigma}\otimes\mathbf{I}_m)$, it leads to a formulation similar to multi-task feature learning [Argyriou et al. 2008;

Argyriou et al. 2008]:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b},\mathbf{\Sigma}} \quad & \sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_j^i - \mathbf{w}_i^T\mathbf{x}_j^i - b_i)^2 + \frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}^T\mathbf{\Sigma}^{-1}\mathbf{W}) \\ \text{s.t.} \quad & \mathbf{\Sigma} \succeq \mathbf{0} \\ & \mathrm{tr}(\mathbf{\Sigma}) \le 1.\end{aligned}$$

From this aspect, we can understand the difference between our method and multi-task feature learning even though those two methods are both related to trace norm. Multi-task feature learning is to learn the covariance structure on the model parameters and the parameters of different tasks are independent given the covariance structure. However, the task relationship is not very clear in this method in that we do not know which task is helpful. In our formulation (8), the relationships between tasks are described explicitly in the task covariance matrix $\mathbf{\Omega}$. Another advantage of formulation (8) is that kernel extension is very natural as that in single-task learning. For multi-task feature learning, however, Gram-Schmidt orthogonalization on the kernel matrix is needed [Argyriou et al. 2008] and hence it will incur additional computational cost.

The above choices for $q(\mathbf{W})$ either assume the tasks are correlated but the data features are independent, or the data features are correlated but the tasks are independent. Here we can generalize them to the case which assumes that the tasks and the data features are both correlated by defining $q(\mathbf{W})$ as $q(\mathbf{W}) = \mathcal{MN}_{d\times m}(\mathbf{W} \mid \mathbf{0}_{d\times m}, \mathbf{\Sigma}\otimes\mathbf{\Omega})$ where $\mathbf{\Sigma}$ describes the correlations between data features and $\mathbf{\Omega}$ models the correlations between tasks. Then the corresponding optimization problem becomes

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b},\mathbf{\Sigma},\mathbf{\Omega}} \quad & \sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_j^i - \mathbf{w}_i^T\mathbf{x}_j^i - b_i)^2 + \frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}^T\mathbf{\Sigma}^{-1}\mathbf{W}\mathbf{\Omega}^{-1}) \\ \text{s.t.} \quad & \mathbf{\Sigma} \succeq \mathbf{0},\ \mathrm{tr}(\mathbf{\Sigma}) \le 1 \\ & \mathbf{\Omega} \succeq \mathbf{0},\ \mathrm{tr}(\mathbf{\Omega}) \le 1. \end{aligned} \tag{23}$$

Unfortunately this optimization problem is non-convex due to the third term in the objective function which makes the performance of this model sensitive to the initial values of the model parameters. But we can also use an alternating method to obtain a locally optimal solution. Moreover, the kernel extension of this method is not very easy to derive since we cannot estimate the covariance matrix $\mathbf{\Sigma}$ for feature correlation in an infinite-dimensional kernel space. But we can also get an approximation by assuming that the primal space of $\mathbf{\Sigma}$ is spanned by the training data points in the kernel space, which is similar to the representer theorem in [Argyriou et al. 2008]. Compared with this problem, problem (8) is jointly convex and its kernel extension is very natural. Moreover, for problem (8), the feature correlations can be considered in the construction of the kernel function by using the following linear and RBF kernels:

$$\begin{aligned}k_{linear}(\mathbf{x}_1,\mathbf{x}_2) &= \mathbf{x}_1^T\mathbf{\Sigma}^{-1}\mathbf{x}_2 \\ k_{rbf}(\mathbf{x}_1,\mathbf{x}_2) &= \exp\Big(-(\mathbf{x}_1-\mathbf{x}_2)^T\mathbf{\Sigma}^{-1}(\mathbf{x}_1-\mathbf{x}_2)/2\Big).\end{aligned}$$

Moreover, by placing sparse priors, such as Laplace distribution, on the inverse of $\mathbf{\Sigma}$ and $\mathbf{\Omega}$, we can recover the method proposed in [Zhang and Schneider 2010], which is also non-convex.

2.6.2 *Utilizing Matrix-Variate t Distribution.* It is well known that the t distribution has heavy-tail behavior which makes it more robust against outliers than the corresponding normal distribution. This also holds for the matrix-variate normal distribution and the matrix-variate t distribution [Gupta and Nagar 2000]. So we can use the matrix-variate t distribution for $q(\mathbf{W})$ to make the model more robust.

We assign the matrix-variate t distribution to $q(\mathbf{W})$:

$$q(\mathbf{W}) = \mathcal{MT}_{d\times m}(\nu, \mathbf{0}_{d\times m}, \mathbf{I}_d \otimes \mathbf{\Omega}),$$

where $\mathcal{MT}_{d\times m}(\nu, \mathbf{M}, \mathbf{A}\otimes\mathbf{B})$ denotes the matrix-variate t distribution [Gupta and Nagar 2000] with the degree of freedom ν, mean $\mathbf{M}\in\mathbb{R}^{d\times m}$, row covariance matrix $\mathbf{A}\in\mathbb{R}^{d\times d}$ and column covariance matrix $\mathbf{B}\in\mathbb{R}^{m\times m}$. Its probability density function is

$$\frac{\Gamma_d(\nu'/2)\,|\mathbf{I}_d + \mathbf{A}^{-1}(\mathbf{W}-\mathbf{M})\mathbf{B}^{-1}(\mathbf{W}-\mathbf{M})^T|^{-\nu'/2}}{\pi^{dm/2}\Gamma_d((\nu'-m)/2)|\mathbf{A}|^{m/2}|\mathbf{B}|^{d/2}},$$

where $\nu' = \nu + d + m - 1$ and $\Gamma_d(\cdot)$ is the multivariate gamma function. Then the corresponding optimization problem can be formulated as

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b},\mathbf{\Omega}} \quad & \sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_j^i - \mathbf{w}_i^T\mathbf{x}_j^i - b_i)^2 + \frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2}\ln|\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T| \\ \text{s.t.} \quad & \mathbf{\Omega}\succeq\mathbf{0} \\ & \mathrm{tr}(\mathbf{\Omega})\le 1.\end{aligned}$$

This is a non-convex optimization problem due to the non-convexity of the last term in the objective function. By using Lemma 1, we can obtain

$$\ln|\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T| \le \mathrm{tr}(\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T) - d = \mathrm{tr}(\mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T). \tag{24}$$

So the objective function of problem (8) is the upper bound of that in this problem and hence this problem can be relaxed to the convex problem (8). Moreover, we may also use the MM algorithm [Lange et al. 2000] to solve this problem. The MM algorithm is an iterative algorithm which seeks an upper bound of the objective function based on the solution from the previous iteration as a surrogate function for the minimization problem and then optimizes with respect to the surrogate function. The MM algorithm is guaranteed to find a local optimum and is widely used in many optimization problems. For our problem, we denote the solution of $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{\Omega}$ in the t-th iteration as $\mathbf{W}^{(t)}$, $\mathbf{b}^{(t)}$ and $\mathbf{\Omega}^{(t)}$. Then by using Lemma 1, we can obtain

$$\begin{aligned} & \ln|\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T| - \ln|\mathbf{M}| \\ = \; & \ln|\mathbf{M}^{-1}(\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)| \\ \le \; & \mathrm{tr}\big(\mathbf{M}^{-1}(\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)\big) - d,\end{aligned}$$

where $\mathbf{M} = \mathbf{I}_d + \mathbf{W}^{(t)}(\mathbf{\Omega}^{(t)})^{-1}(\mathbf{W}^{(t)})^T$. So we can get

$$\ln|\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T| \le \mathrm{tr}\big(\mathbf{M}^{-1}(\mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T)\big) + \ln|\mathbf{M}| - d. \tag{25}$$

We can prove that this bound is tighter than the previous one in Eq. (24) and the proof is given in Appendix C. So in the $(t+1)$-th iteration, the MM algorithm is to solve the following optimization problem:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{b},\mathbf{\Omega}} \quad & \sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}(y_j^i-\mathbf{w}_i^T\mathbf{x}_j^i-b_i)^2+\frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T)+\frac{\lambda_2}{2}\mathrm{tr}(\mathbf{W}^T\mathbf{M}^{-1}\mathbf{W}\mathbf{\Omega}^{-1})\\ \text{s.t.} \quad & \mathbf{\Omega}\succeq\mathbf{0}\\ & \mathrm{tr}(\mathbf{\Omega})\le 1.\end{aligned}$$

This problem is similar to problem (23) with the difference that $\mathbf{\Sigma}$ in problem (23) is a variable but here $\mathbf{M}$ is a constant matrix. Similar formulations lead to similar limitations though. For example, the kernel extension is not very natural.

3. RELATIONSHIPS WITH EXISTING METHODS

In this section, we discuss some connection between our method and other existing multi-task learning methods.

3.1 Relationships with Existing Regularized Multi-Task Learning Methods

Some existing multi-task learning methods [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Kato et al. 2008; Jacob et al. 2008] also model the relationships between tasks under the regularization framework. The methods in [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Kato et al. 2008] assume that the task relationships are given *a priori* and then utilize this prior knowledge to learn the model parameters. On the other hand, the method in [Jacob et al. 2008] learns the task cluster structure from data. In this section, we discuss the relationships between MTRL and these methods.

The objective functions of the methods in [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Kato et al. 2008; Jacob et al. 2008] are all of the following form which is similar to that of problem (8):

$$J=\sum_{i=1}^{m}\sum_{j=1}^{n_i}l(y_j^i,\mathbf{w}_i^T\mathbf{x}_j^i+b_i)+\frac{\lambda_1}{2}\mathrm{tr}(\mathbf{W}\mathbf{W}^T)+\frac{\lambda_2}{2}f(\mathbf{W}),$$

with different choices for the formulation of $f(\cdot)$.

The method in [Evgeniou and Pontil 2004] assumes that all tasks are similar and so the parameter vector of each task is similar to the average parameter vector. The corresponding formulation for $f(\cdot)$ is given by

$$f(\mathbf{W})=\sum_{i=1}^{m}\Big\|\mathbf{w}_i-\frac{1}{m}\sum_{j=1}^{m}\mathbf{w}_j\Big\|_2^2.$$

After some algebraic operations, we can rewrite $f(\mathbf{W})$ as

$$f(\mathbf{W})=\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{1}{2m}\|\mathbf{w}_i-\mathbf{w}_j\|_2^2=\mathrm{tr}(\mathbf{W}\mathbf{L}\mathbf{W}^T),$$

where $\mathbf{L}$ is the Laplacian matrix defined on a fully connected graph with edge weights equal to $\frac{1}{2m}$. This corresponds to a special case of MTRL with $\mathbf{\Omega}^{-1}=\mathbf{L}$. Obviously, a limitation of this method is that only positive task correlation can be modeled.

The methods in [Evgeniou et al. 2005] assume that the task cluster structure or the task similarity between tasks is given. $f(\cdot)$ is formulated as

$$f(\mathbf{W}) = \sum_{i,j} s_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2 = \mathrm{tr}(\mathbf{W}\mathbf{L}\mathbf{W}^T),$$

where $s_{ij} \geq 0$ denotes the similarity between tasks T_i and T_j and $\mathbf{L}$ is the Laplacian matrix defined on the graph based on $\{s_{ij}\}$. Again, it corresponds to a special case of MTRL with $\mathbf{\Omega}^{-1} = \mathbf{L}$. Note that this method requires that $s_{ij} \geq 0$ and so it also can only model positive task correlation and task unrelatedness. If negative task correlation is modeled as well, the problem will become non-convex making it more difficult to solve. Moreover, in many real-world applications, prior knowledge about s_{ij} is not available.

In [Kato et al. 2008] the authors assume the existence of a task network and that the neighbors in the task network, encoded as index pairs (p_k, q_k), are very similar. $f(\cdot)$ can be formulated as

$$f(\mathbf{W}) = \sum_k \|\mathbf{w}_{p_k} - \mathbf{w}_{q_k}\|_2^2.$$

We can define a similarity matrix G whose the (p_k, q_k)th elements are equal to 1 for all k and 0 otherwise. Then $f(\mathbf{W})$ can be simplified as $f(\mathbf{W}) = \mathrm{tr}(\mathbf{W}\mathbf{L}\mathbf{W}^T)$ where $\mathbf{L}$ is the Laplacian matrix of G, which is similar to [Evgeniou et al. 2005]. Thus it also corresponds to a special case of MTRL with $\mathbf{\Omega}^{-1} = \mathbf{L}$. Similar to [Evgeniou et al. 2005], a difficulty of this method is that prior knowledge in the form of a task network is not available in many applications.

The method in [Jacob et al. 2008] is more general in that it learns the task cluster structure from data, making it more suitable for real-world applications. The formulation for $f(\cdot)$ is described as

$$f(\mathbf{W}) = \mathrm{tr}\left(\mathbf{W}\big[\alpha\mathbf{H}_m + \beta(\mathbf{M} - \mathbf{H}_m) + \gamma(\mathbf{I}_m - \mathbf{M})\big]\mathbf{W}^T\right),$$

where $\mathbf{H}_m$ is the centering matrix and $\mathbf{M} = \mathbf{E}(\mathbf{E}^T\mathbf{E})\mathbf{E}^T$ with the cluster assignment matrix $\mathbf{E}$. If we let $\mathbf{\Omega}^{-1} = \alpha\mathbf{H}_m + \beta(\mathbf{M} - \mathbf{H}_m) + \gamma(\mathbf{I}_m - \mathbf{M})$ or $\mathbf{\Omega} = \frac{1}{\alpha}\mathbf{H}_m + \frac{1}{\beta}(\mathbf{M} - \mathbf{H}_m) + \frac{1}{\gamma}(\mathbf{I}_m - \mathbf{M})$, MTRL will reduce to this method. However, [Jacob et al. 2008] is a local method which can only model positive task correlations within each cluster but cannot model negative task correlations among different task clusters. Another difficulty of this method lies in determining the number of task clusters.

Compared with existing methods, MTRL is very appealing in that it can learn all three types of task relationships in a nonparametric way. This makes it easy to identify the tasks that are useful for multi-task learning and those that should not be exploited.

3.2 Relationships with Multi-Task Gaussian Process

The multi-task GP model in [Bonilla et al. 2007] directly models the task covariance matrix $\mathbf{\Sigma}$ by incorporating it into the GP prior as follows:

$$\langle f_j^i, f_s^r \rangle = \Sigma_{ir} k(\mathbf{x}_j^i, \mathbf{x}_s^r), \tag{26}$$

where $\langle \cdot, \cdot \rangle$ denotes the covariance of two random variables, f_j^i is the latent function value for $\mathbf{x}_j^i$, and Σ_{ir} is the (i, r)th element of $\mathbf{\Sigma}$. The output y_j^i given f_j^i is distributed as

$$y_j^i \mid f_j^i \sim \mathcal{N}(f_j^i, \sigma_i^2),$$

which defines the likelihood for $\mathbf{x}_j^i$. Here σ_i^2 is the noise level of the ith task.

Recall that GP has an interpretation from the weight-space view [Rasmussen and Williams 2006]. In our previous work [Zhang and Yeung 2010b], we also give a weight-space view of this multi-task GP model:

$$\begin{aligned} y_j^i &= \mathbf{w}_i^T \phi(\mathbf{x}_j^i) + \varepsilon_j^i \\ \mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_m] &\sim \mathcal{MN}_{d' \times m}(\mathbf{0}_{d' \times m}, \mathbf{I}_{d'} \otimes \mathbf{\Sigma}) \\ \varepsilon_j^i &\sim \mathcal{N}(0, \sigma_i^2), \end{aligned} \tag{27}$$

where $\phi(\cdot)$, which maps $\mathbf{x} \in \mathbb{R}^d$ to $\phi(\mathbf{x}) \in \mathbb{R}^{d'}$ and may have no explicit form, denotes a feature mapping corresponding to the kernel function $k(\cdot, \cdot)$. The equivalence between the model formulations in (26) and (27) is due to the following which is a consequence of the property of the matrix-variate normal distribution:[4]

$$f_j^i \overset{\text{def}}{=} \phi(\mathbf{x}_j^i)^T \mathbf{w}_i = \phi(\mathbf{x}_j^i)^T \mathbf{W} \mathbf{e}_{m,i} \sim \mathcal{N}(0, \Sigma_{ii} k(\mathbf{x}_j^i, \mathbf{x}_j^i)) \tag{28}$$

$$\langle f_j^i, f_s^r \rangle = \int \phi(\mathbf{x}_j^i)^T \mathbf{W} \mathbf{e}_{m,i} \mathbf{e}_{m,r}^T \mathbf{W}^T \phi(\mathbf{x}_s^r) p(\mathbf{W}) d\mathbf{W} = \Sigma_{ir} k(\mathbf{x}_j^i, \mathbf{x}_s^r), \tag{29}$$

where $\mathbf{e}_{m,i}$ is the ith column of $\mathbf{I}_m$. The weight-space view of the conventional GP can be seen as a special case of that of the multi-task GP with $m = 1$, under which the prior for $\mathbf{W}$ in (27) will become the ordinary normal distribution with zero mean and identity covariance matrix by setting $\mathbf{\Sigma} = 1$.

It is easy to see that the weight-space view model (27) is similar to our model which shows the relationship of our method with multi-task GP. However, the optimization problem in [Bonilla et al. 2007] is non-convex which makes the multi-task GP more sensitive to the initial values of model parameters. To reduce the number of model parameters, multi-task GP seeks a low-rank approximation of the task covariance matrix which may weaken the expressive power of the task covariance matrix and limit the performance of the model. Moreover, since multi-task GP is based on the GP model, the complexity of multi-task GP is cubic with respect to the number of data points in all tasks. This high complexity requirement may limit the use of multi-task GP for large-scale applications.

Recently Dinuzzo et al. [Dinuzzo et al. 2011; Dinuzzo and Fukumizu 2011] proposed methods to learn an output kernel which has similar objective as our method. One advantage of our method over [Dinuzzo et al. 2011; Dinuzzo and Fukumizu 2011] is that the objective function of our method is jointly convex with respect to all model parameters which may bring some additional computational benefits.

4. EXPERIMENTS

In this section, we study MTRL empirically on some data sets and compare it with a single-task learning (STL) method, multi-task feature learning (MTFL) [Argyriou et al. 2008] method[5] and a multi-task GP (MTGP) method [Bonilla et al. 2007] which can also learn the global task relationships.

[4]The proofs for the following two equations can be found in Appendix D.

[5]`http://www0.cs.ucl.ac.uk/staff/A.Argyriou/code/`.

4.1 Toy Problem

We first generate a toy data set to conduct a "proof of concept" experiment before we do experiments on real data sets. The toy data set is generated as follows. The regression functions corresponding to three regression tasks are defined as $y = 3x + 10$, $y = -3x - 5$ and $y = 1$. For each task, we randomly sample five points uniformly from $[0, 10]$. Each function output is corrupted by a Gaussian noise process with zero mean and variance equal to 0.1. One example of the data set is plotted in Figure 1, with each color (and point type) corresponding to one task. We repeat the experiment 10 times. From the coefficients of the regression functions, we expect the correlation between the first two tasks to approach -1 and those for the other two pairs of tasks to approach 0. To apply MTRL, we use the linear kernel and set λ_1 to 0.01 and λ_2 to 0.005. After the learning procedure converges, we find that the mean estimated regression functions for the three tasks are $y = 2.9964x + 10.0381$, $y = -3.0022x - 4.9421$ and $y = 0.0073x + 0.9848$. Based on the task covariance matrix learned, we obtain the following the mean task correlation matrix:

$$\mathbf{C} = \begin{pmatrix} 1.0000 & -0.9985 & 0.0632 \\ -0.9985 & 1.0000 & -0.0623 \\ 0.0632 & -0.0623 & 1.0000 \end{pmatrix},$$

where the calculation of the task correlation matrix $\mathbf{C}$ follows the relation between covariance matrices and correlation matrices that the (i, j)th element in $\mathbf{C}$ equals $\frac{\omega_{ij}}{\sqrt{\omega_{ii}\omega_{jj}}}$ with ω_{ij} being the (i, j)th element in the task covariance matrix $\mathbf{\Omega}$. We can see that the task correlations learned confirm our expectation, showing that MTRL can indeed learn the relationships between tasks for this toy problem.

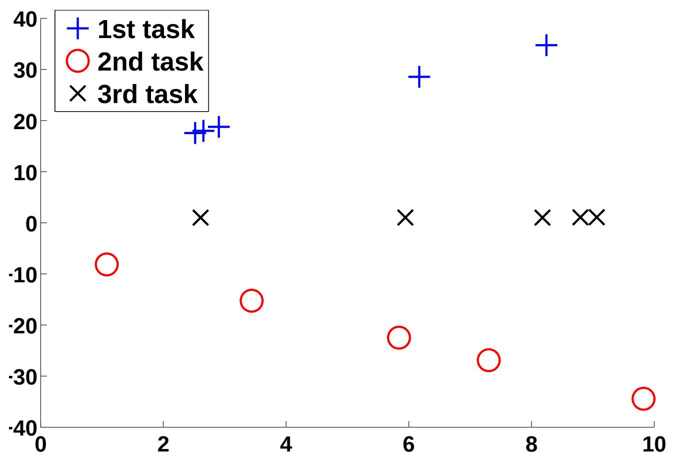

Fig. 1. One example of the toy problem. The data points with each color (and point type) correspond to one task.

4.2 Robot Inverse Dynamics

We now study the problem of learning the inverse dynamics of a 7-DOF SARCOS anthropomorphic robot arm[6]. Each observation in the SARCOS data set consists of 21 input features, corresponding to seven joint positions, seven joint velocities and seven joint accelerations, as well as seven joint torques for the seven degrees of freedom (DOF). Thus the input has 21 dimensions and there are seven tasks. We randomly select 600 data points for each task to form the training set and 1400 data points for each task for the test set. The performance measure used is the normalized mean squared error (nMSE), which is the mean squared error divided by the variance of the ground truth. The single-task learning method is kernel ridge regression. The kernel used is the RBF kernel. Five-fold cross validation is used to determine the values of the kernel parameter and the regularization parameters λ_1 and λ_2. We perform 10 random splits of the data and report the mean and standard derivation over the 10 trials. The results are summarized in Table I and the mean task correlation matrix over 10 trials is recorded in Table II. From the results, we can see that the performance of MTRL is better than that of STL, MTFL and MTGP. From Table II, we can see that some tasks are positively correlated (e.g., third and sixth tasks), some are negatively correlated (e.g., second and third tasks), and some are uncorrelated (e.g., first and seventh tasks).

Table I. Comparison of different methods on SARCOS data. Each column represents one task. The first row of each method records the mean of nMSE over 10 trials and the second row records the standard derivation.

Method	1st DOF	2nd DOF	3rd DOF	4th DOF	5th DOF	6th DOF	7th DOF
STL	0.2874	0.2356	0.2310	0.2366	0.0500	0.5208	0.6748
	0.0067	0.0043	0.0068	0.0042	0.0034	0.0205	0.0048
MTFL	0.2876	0.1611	0.2125	0.2215	0.0858	0.5224	0.7135
	0.0178	0.0105	0.0225	0.0151	0.0225	0.0269	0.0196
MTGP	0.3430	0.7890	0.5560	0.3147	0.0100	**0.0690**	0.6455
	0.1038	0.0480	0.0511	0.1235	0.0067	0.0171	0.4722
MTRL	**0.0968**	**0.0229**	**0.0625**	**0.0422**	**0.0045**	0.0851	**0.3450**
	0.0047	0.0023	0.0044	0.0027	0.0002	0.0095	0.0127

Moreover, we plot in Figure 2 the change in value of the objective function in problem (8). We find that the objective function value decreases rapidly and then levels off, showing the fast convergence of the algorithm which takes no more than 15 iterations.

4.3 Multi-Domain Sentiment Application

We next study a multi-domain sentiment classification application[7] which is a multi-task classification problem. Its goal is to classify the reviews of some products into two classes: positive and negative reviews. In the data set, there are four different products (tasks) from

[6]`http://www.gaussianprocess.org/gpml/data/`
[7]http://www.cs.jhu.edu/∼mdredze/datasets/sentiment/

Table II. Mean task correlation matrix learned from SARCOS data on different tasks.

	1st	2nd	3rd	4th	5th	6th	7th
1st	1.0000	0.7435	-0.7799	0.4819	-0.5325	-0.4981	0.0493
2nd	0.7435	1.0000	-0.9771	0.1148	-0.0941	-0.7772	-0.4419
3rd	-0.7799	-0.9771	1.0000	-0.1872	0.1364	0.8145	0.3987
4th	0.4819	0.1148	-0.1872	1.0000	-0.1889	-0.3768	0.7662
5th	-0.5325	-0.0941	0.1364	-0.1889	1.0000	-0.3243	-0.2834
6th	-0.4981	-0.7772	0.8145	-0.3768	-0.3243	1.0000	0.2282
7th	0.0493	-0.4419	0.3987	0.7662	-0.2834	0.2282	1.0000

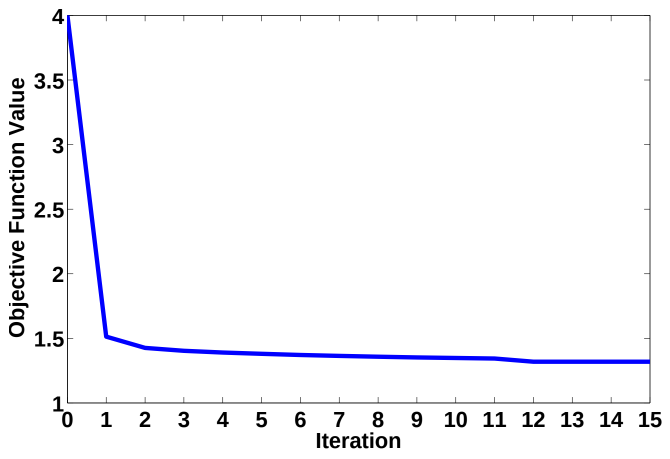

Fig. 2. Convergence of objective function value for SARCOS data

Amazon.com: books, DVDs, electronics, and kitchen appliances. For each task, there are 1000 positive and 1000 negative data points corresponding to positive and negative reviews, respectively. Each data point has 473856 feature dimensions. To see the effect of varying the training set size, we randomly select 10%, 30% and 50% of the data for each task to form the training set and the rest for the test set. The performance measure used is the classification error. We use SVM as the single-task learning method. The kernel used is the linear kernel which is widely used for text applications with high feature dimensionality. Five-fold cross validation is used to determine the values of the regularization parameters λ_1 and λ_2. We perform 10 random splits of the data and report the mean and standard derivation over the 10 trials. The results are summarized in the left column of Table III. From the table, we can see that the performance of MTRL is better than that of STL, MTFL and MTGP on every task under different training set sizes. Moreover, the mean task correlation matrices over 10 trials for different training set sizes are recorded in the right column of Table III. From Table III, we can see that the first task 'books' is more correlated with the second task 'DVDs' than with the other tasks; the third and fourth tasks achieve the largest correlation among all pairs of tasks. The findings from Table III can be easily interpreted as follows: 'books' and 'DVDs' are mainly for entertainment; almost all the elements in ‘kitchen appliances’ belong to ‘electronics’. So the knowledge found by our method about the relationships between tasks matches our intuition. Moreover, some interesting patterns exist in the mean task correlation matrices for different training set sizes. For example, the correlation between the third and fourth tasks is always the largest when training size varies; the correlation between the first and second tasks is larger than that between the first and third tasks, and also between the first and fourth tasks.

Table III. Comparison of different methods on multi-domain sentiment data for different training set sizes. The three tables in the left column record the classification errors of different methods when 10%, 30% and 50%, respectively, of the data are used for training. For each method, the first row records the mean classification error over 10 trials and the second row records the standard derivation. The three tables in the right column record the mean task correlations when 10%, 30% and 50%, respectively, of the data are used for training.

Method	1st Task	2nd Task	3rd Task	4th Task
STL	0.2680	0.3142	0.2891	0.2401
	0.0112	0.0110	0.0113	0.0154
MTFL	0.2667	0.3071	0.2880	0.2407
	0.0160	0.0136	0.0193	0.0160
MTGP	0.2332	0.2739	0.2624	0.2061
	0.0159	0.0231	0.0150	0.0152
MTRL	**0.2233**	**0.2564**	**0.2472**	**0.2027**
	0.0055	0.0050	0.0082	0.0044

	1st	2nd	3rd	4th
1st	1.0000	0.7675	0.6878	0.6993
2nd	0.7675	1.0000	0.6937	0.6805
3rd	0.6878	0.6937	1.0000	0.8793
4th	0.6993	0.6805	0.8793	1.0000

Method	1st Task	2nd Task	3rd Task	4th Task
STL	0.1946	0.2333	0.2143	0.1795
	0.0102	0.0119	0.0110	0.0076
MTFL	0.1932	0.2321	0.2089	0.1821
	0.0094	0.0115	0.0054	0.0078
MTGP	0.1852	0.2155	0.2088	0.1695
	0.0109	0.0101	0.0120	0.0074
MTRL	**0.1688**	**0.1987**	**0.1975**	**0.1482**
	0.0103	0.0120	0.0094	0.0087

	1st	2nd	3rd	4th
1st	1.0000	0.6275	0.5098	0.5936
2nd	0.6275	1.0000	0.4900	0.5345
3rd	0.5098	0.4900	1.0000	0.7286
4th	0.5936	0.5345	0.7286	1.0000

Method	1st Task	2nd Task	3rd Task	4th Task
STL	0.1854	0.2162	0.2072	0.1706
	0.0102	0.0147	0.0133	0.0024
MTFL	0.1821	0.2096	0.2128	0.1681
	0.0095	0.0095	0.0106	0.0085
MTGP	0.1722	0.2040	0.1992	0.1496
	0.0101	0.0152	0.0083	0.0051
MTRL	**0.1538**	**0.1874**	**0.1796**	**0.1334**
	0.0096	0.0149	0.0084	0.0036

	1st	2nd	3rd	4th
1st	1.0000	0.6252	0.5075	0.5901
2nd	0.6252	1.0000	0.4891	0.5328
3rd	0.5075	0.4891	1.0000	0.7256
4th	0.5901	0.5328	0.7256	1.0000

4.4 Examination Score Prediction

The school data set[8] has been widely used for studying multi-task regression. It consists of the examination scores of 15362 students from 139 secondary schools in London during the years 1985, 1986 and 1987. Thus, there are totally 139 tasks. The input consists of the year of the examination, four school-specific and three student-specific attributes. We replace each categorical attribute with one binary variable for each possible attribute value, as in [Evgeniou et al. 2005]. As a result of this preprocessing, we have a total of 27 input attributes. The experimental settings are the same as those in [Argyriou et al. 2008], i.e.,

[8] `http://www0.cs.ucl.ac.uk/staff/A.Argyriou/code/`

we use the same 10 random splits of the data to generate the training and test sets, so that 75% of the examples from each school belong to the training set and 25% to the test set. For our performance measure, we use the measure of percentage explained variance from [Argyriou et al. 2008], which is defined as the percentage of one minus nMSE. We use five-fold cross validation to determine the values of the kernel parameters in the RBF kernel and the regularization parameters λ_1 and λ_2. Since the experimental setting is the same, we compare our result with the results reported in [Argyriou et al. 2008; Bonilla et al. 2007]. The results are summarized in Table IV. We can see that the performance of MTRL is better than both STL and MTFL and is slightly better than MTGP.

Table IV. Comparison of different methods on school data (in mean±std-dev).

Method	Explained Variance
STL	23.5±1.9%
MTFL	26.7±2.0%
MTGP	29.2±1.6%
MTRL	**29.9±1.8%**

4.5 Experiments on One Variant based on Matrix-Variate t Distribution

In this section, we investigate the performance of one variant of our method which is based on matrix-variate t distribution and discussed in section 2.6.2. Here we denote this variant by the MTRL$_t$ method.

We first conduct experiments on the toy problem where the synthetic data are generated in a way similar to the generation process described in Section 4.1. The initial values for $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{\Omega}$ are set to be $\mathbf{0}_{d\times m}$, $\mathbf{0}_m$ and $\frac{1}{m}\mathbf{I}_m$ respectively. We set the values for the regularization parameters as $\lambda_1 = \lambda_2 = 0.01$. The estimated task correlation matrix $\mathbf{C}$ which is calculated from the task covariance matrix $\mathbf{\Omega}$ is as follows:

$$\mathbf{C} = \begin{pmatrix} 1.0000 & -0.9979 & 0.0535 \\ -0.9979 & 1.0000 & -0.0535 \\ 0.0535 & -0.0535 & 1.0000 \end{pmatrix},$$

where the correlation between the first two tasks (i.e., -0.9979) approaches -1 and those for the other two pairs of tasks (i.e., 0.0535 and -0.0535) are close to 0. So the task correlations learned match our expectation, which demonstrates the effectiveness of the MTRL$_t$ method to learn the task relationships on this toy problem.

Since the MTRL$_t$ is not very easy to be extended to use the kernel trick, we just consider the linear model for MTRL$_t$ and so does MTRL. We compare MTRL$_t$ with MTRL on the SARCOS and school datasets with the same settings as in the previous experiments. The experimental results are record in Table V and VI. From the results, we can see that the performance of MTRL$_t$ is comparable to and even better than that of MTRL, which confirms that the robustness of matrix-variate t distribution is helpful to the performance improvement.

Moreover, to show the convergence of the proposed MM algorithm in Section 2.6.2, we plot the change in value of the objective function in Figure 3. We can see that the proposed algorithm converges very fast, i.e., in no more than 15 iterations.

Table V. Comparison between MTRL and MTRL$_t$ on SARCOS data. Each column represents one task. The first row of each method records the mean of nMSE over 10 trials and the second row records the standard derivation.

Method	1st DOF	2nd DOF	3rd DOF	4th DOF	5th DOF	6th DOF	7th DOF
MTRL	0.1523	0.0625	0.1243	0.1117	**0.0151**	0.1679	0.5528
	0.0033	0.0031	0.0029	0.0041	0.0006	0.0044	0.0060
MTRL$_t$	**0.1346**	**0.0593**	**0.1140**	**0.1062**	**0.0149**	**0.1068**	**0.5156**
	0.0039	0.0030	0.0054	0.0028	0.0008	0.0032	0.0106

Table VI. Comparison between MTRL and MTRL$_t$ on school data (in mean±std-dev).

Method	Explained Variance
MTRL	12.75±0.43%
MTRL$_t$	**18.68±2.07%**

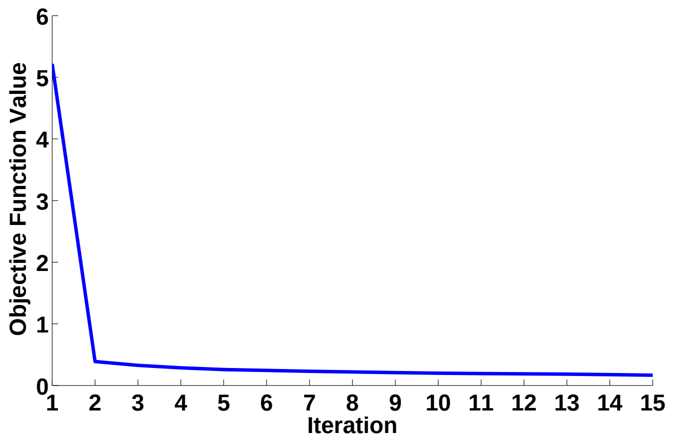

Fig. 3. Convergence of objective function value of MTRL$_t$ on SARCOS data

4.6 Experiments on Asymmetric Multi-Task Learning

The above sections mainly focus on symmetric multi-task learning. Here in this section we report some experimental results on asymmetric multi-task learning [Xue et al. 2007; Argyriou et al. 2008; Romera-Paredes et al. 2012; Maurer et al. 2012] and choose the DP-MTL method in [Xue et al. 2007] as a baseline method. Since the DP-MTL method in [Xue et al. 2007] focuses on classification applications, we compare it with our method on the multi-domain sentiment application. Moreover, we also make comparison with conventional SVM which serves as a baseline single-task learning method.

All compared methods are tested under the leave-one-task-out (LOTO) setting. That is, in each fold, one task is treated as the new task while all other tasks are treated as existing tasks. Moreover, to see the effect of varying the training set size, we randomly sample

10%, 30% or 50% of the data in the new task to form the training set and the rest is used as the test set. Each configuration is repeated 10 times and we record the mean and standard deviation of the classification error in the experimental results. The results are recorded in Table VII. We can see that our method outperforms both single-task learning and DP-MTL. In fact the performance of DP-MTL is even worse than that of single-task learning. One reason is that the relationships between tasks do not exhibit strong cluster structure, as can be revealed from the task correlation matrix in Table III. Since the tasks have no cluster structure, merging several tasks into one and learning common model parameters for the merged tasks will likely deteriorate the performance.

Table VII. Classification errors (in mean±std-dev) of different methods on the multi-domain sentiment data for different training set sizes under the asymmetric multi-task setting. The three tables record the classification errors of different methods when 10%, 30% and 50%, respectively, of the data are used for training.

New Task	STL	DP-MTL	MTRL
1st Task	0.3013±0.0265	0.3483±0.0297	**0.2781±0.0170**
2nd Task	0.3073±0.0117	0.3349±0.0121	**0.2801±0.0293**
3rd Task	0.2672±0.0267	0.2936±0.0274	**0.2451±0.0078**
4th Task	0.2340±0.0144	0.2537±0.0128	**0.2114±0.0208**

New Task	STL	DP-MTL	MTRL
1st Task	0.2434±0.0097	0.2719±0.0212	**0.2164±0.0098**
2nd Task	0.2479±0.0101	0.2810±0.0253	**0.2120±0.0160**
3rd Task	0.2050±0.0172	0.2306±0.0131	**0.1883±0.0106**
4th Task	0.1799±0.0057	0.2141±0.0362	**0.1561±0.0123**

New Task	STL	DP-MTL	MTRL
1st Task	0.2122±0.0083	0.2576±0.0152	**0.1826±0.0156**
2nd Task	0.2002±0.0112	0.2582±0.0275	**0.1870±0.0151**
3rd Task	0.1944±0.0069	0.2252±0.0208	**0.1692±0.0107**
4th Task	0.1678±0.0109	0.1910±0.0227	**0.1398±0.0131**

5. CONCLUSION

In this paper, we have presented a regularization approach to learning the relationships between tasks in multi-task learning. Our method can model global task relationships and the learning problem can be formulated directly as a convex optimization problem by utilizing the matrix-variate normal distribution as a prior. We study the proposed method under both symmetric and asymmetric multi-task learning settings.

In some multi-task learning applications, there exist additional sources of data such as unlabeled data. In our future research, we will consider incorporating additional data sources into our regularization formulation in a way similar to manifold regularization [Belkin et al. 2006] to further boost the learning performance under both symmetric and asymmetric multi-task learning settings.

Acknowledgments

We thank Xuejun Liao for providing the source code of the DP-MTL method. Yu Zhang is supported by HKBU 'Start Up Grant for New Academics' and Dit-Yan Yeung is supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

REFERENCES

ANDO, R. K. AND ZHANG, T. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research 6*, 1817–1853.

ARCHAMBEAU, C., GUO, S., AND ZOETER, O. 2011. Sparse Bayesian multi-task learning. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds. Granada, Spain, 1755–1763.

ARGYRIOU, A., EVGENIOU, T., AND PONTIL, M. 2008. Convex multi-task feature learning. *Machine Learning 73,* 3, 243–272.

ARGYRIOU, A., MAURER, A., AND PONTIL, M. 2008. An algorithm for transfer learning in a heterogeneous environment. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*. Antwerp, Belgium, 71–85.

ARGYRIOU, A., MICCHELLI, C. A., PONTIL, M., AND YING, Y. 2008. A spectral regularization framework for multi-task structure learning. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Vancouver, British Columbia, Canada, 25–32.

BAKKER, B. AND HESKES, T. 2003. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research 4*, 83–99.

BAXTER, J. 1997. A Bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning 28,* 1, 7–39.

BELKIN, M., NIYOGI, P., AND SINDHWANI, V. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research 7*, 2399–2434.

BISHOP, C. M. 2006. *Pattern Recognition and Machine Learning*. Springer, New York.

BONILLA, E., CHAI, K. M. A., AND WILLIAMS, C. 2007. Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Vancouver, British Columbia, Canada, 153–160.

BOYD, S. AND VANDENBERGHE, L. 2004. *Convex Optimization*. Cambridge University Press, New York, NY.

CARUANA, R. 1997. Multitask learning. *Machine Learning 28,* 1, 41–75.

CHEN, J., LIU, J., AND YE, J. 2010. Learning incoherent sparse and low-rank patterns from multiple tasks. In *Proceedings of the Sixteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, DC, USA, 1179–1188.

CHEN, J., TANG, L., LIU, J., AND YE, J. 2009. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the Twenty-sixth International Conference on Machine Learning*. Montreal, Quebec, Canada, 137–144.

CHEN, J., ZHOU, J., AND YE, J. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Diego, CA, USA, 42–50.

DINUZZO, F. AND FUKUMIZU, K. 2011. Learning low-rank output kernels. In *Proceedings of the 3rd Asian Conference on Machine Learning*. Taoyuan, Taiwan, 181–196.

DINUZZO, F., ONG, C. S., GEHLER, P. V., AND PILLONETTO, G. 2011. Learning output kernels with block coordinate descent. In *Proceedings of the 28th International Conference on Machine Learning*. Bellevue, Washington, USA, 49–56.

EATON, E., DESJARDINS, M., AND LANE, T. 2008. Modeling transfer relationships between learning tasks for improved inductive transfer. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*. Antwerp, Belgium, 317–332.

EVGENIOU, T., MICCHELLI, C. A., AND PONTIL, M. 2005. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research 6*, 615–637.

EVGENIOU, T. AND PONTIL, M. 2004. Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Seattle, Washington, USA, 109–117.

GESTEL, T. V., SUYKENS, J. A. K., BAESENS, B., VIAENE, S., VANTHIENEN, J., DEDENE, G., MOOR, B. D., AND VANDEWALLE, J. 2004. Benchmarking least squares support vector machine classifiers. *Machine Learning 54,* 1, 5–32.

GUPTA, A. K. AND NAGAR, D. K. 2000. *Matrix variate distributions.* Chapman & Hall.

JACOB, L., BACH, F., AND VERT, J.-P. 2008. Clustered multi-task learning: a convex formulation. In *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Vancouver, British Columbia, Canada, 745–752.

JALALI, A., RAVIKUMAR, P., SANGHAVI, S., AND RUAN, C. 2010. A dirty model for multi-task learning. In *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds. 964–972.

KATO, T., KASHIMA, H., SUGIYAMA, M., AND ASAI, K. 2008. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Vancouver, British Columbia, Canada, 737–744.

KEERTHI, S. S. AND SHEVADE, S. K. 2003. SMO algorithm for least-squares SVM formulation. *Neural Computation 15,* 2, 487–507.

KIENZLE, W. AND CHELLAPILLA, K. 2006. Personalized handwriting recognition via biased regularization. In *Proceedings of the Twenty-Third International Conference on Machine Learning*. Pittsburgh, Pennsylvania, USA, 457–464.

KUMAR, A. AND III, H. D. 2012. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29 th International Conference on Machine Learning*. Edinburgh, Scotland, UK.

LANGE, K., HUNTER, D. R., AND YANG, I. 2000. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics 9,* 1, 1–59.

LOBO, M. S., VANDENBERGHE, L., BOYD, S., AND LEBRET, H. 1998. Applications of second-order cone programming. *Linear Algebra and its Applications 284*, 193–228.

MAURER, A., PONTIL, M., AND ROMERA-PAREDES, B. 2012. Sparse coding for multitask and transfer learning. *CoRR abs/1209.0738.*

OBOZINSKI, G., TASKAR, B., AND JORDAN, M. 2006. Multi-task feature selection. Tech. rep., Department of Statistics, University of California, Berkeley. June.

OBOZINSKI1, G., TASKAR, B., AND JORDAN, M. I. 2010. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing 20,* 2, 231–252.

PAN, S. AND YANG, Q. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering 22,* 10, 1345–1359.

PASSOS, A., RAI, P., WAINER, J., AND III, H. D. 2012. Flexible modeling of latent task structures in multitask learning. In *Proceedings of the 29 th International Conference on Machine Learning*. Edinburgh, Scotland, UK.

RAI, P. AND III, H. D. 2010. Infinite predictor subspace models for multitask learning. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 613–620.

RAINA, R., NG, A. Y., AND KOLLER, D. 2006. Constructing informative priors using transfer learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning*. Pittsburgh, Pennsylvania, USA, 713–720.

RASMUSSEN, C. E. AND WILLIAMS, C. K. I. 2006. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA.

ROMERA-PAREDES, B., ARGYRIOU, A., BERTHOUZE, N., AND PONTIL, M. 2012. Exploiting unrelated tasks in multi-task learning. *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, 951–959.

THRUN, S. 1996. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. Mozer, and M. E. Hasselmo, Eds. Denver, CO, 640–646.

THRUN, S. AND O'SULLIVAN, J. 1996. Discovering structure in multiple learning tasks: The TC algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*. Bari, Italy, 489–497.

TITSIAS, M. K. AND LÁZARO-GREDILLA, M. 2011. Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds. Granada, Spain, 2339–2347.

WU, P. AND DIETTERICH, T. G. 2004. Improving SVM accuracy by training on auxiliary data sources. In *Proceedings of the Twenty-first International Conference on Machine Learning*. Banff, Alberta, Canada.

XUE, Y., LIAO, X., CARIN, L., AND KRISHNAPURAM, B. 2007. Multi-task learning for classification with Dirichlet process priors. *Journal of Machine Learning Research 8*, 35–63.

YU, K., TRESP, V., AND SCHWAIGHOFER, A. 2005. Learning Gaussian processes from multiple tasks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*. Bonn, Germany, 1012–1019.

YU, S., TRESP, V., AND YU, K. 2007. Robust multi-task learning with t-processes. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*. Corvalis, Oregon, USA, 1103–1110.

ZHANG, Y. 2013. Heterogeneous-neighborhood-based multi-task local learning algorithms. In *Advances in Neural Information Processing Systems 26*. Lake Tahoe, Nevada, USA.

ZHANG, Y. AND SCHNEIDER, J. G. 2010. Learning multiple tasks with a sparse matrix-normal penalty. In *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Vancouver, British Columbia, Canada, 2550–2558.

ZHANG, Y. AND YEUNG, D.-Y. 2010a. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*. Catalina Island, California, 733–742.

ZHANG, Y. AND YEUNG, D.-Y. 2010b. Multi-task learning using generalized t process. In *Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy, 964–971.

ZHANG, Y. AND YEUNG, D.-Y. 2010c. Transfer metric learning by learning task relationships. In *Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Washington, DC, USA, 1199–1208.

ZHANG, Y. AND YEUNG, D.-Y. 2012. Transfer metric learning with semi-supervised extension. *ACM Transactions on Intelligent Systems and Technology 3,* 3, article 54.

ZHANG, Y., YEUNG, D.-Y., AND XU, Q. 2010. Probabilistic multi-task feature selection. In *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Vancouver, British Columbia, Canada, 2559–2567.

ZHU, J., CHEN, N., AND XING, E. P. 2011. Infinite latent SVM for classification and multi-task learning. In *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds. Granada, Spain, 1620–1628.

Appendix A

In this section, we present an SMO algorithm to solve problem (13).

Recall that the dual form is formulated as follows:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & h(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T\tilde{\mathbf{K}}\boldsymbol{\alpha} + \sum_{i,j}\alpha_j^i y_j^i \\ \text{s.t.} \quad & \sum_j \alpha_j^i = 0 \quad \forall i, \end{aligned} \tag{30}$$

where $\mathbf{K}$ is the kernel matrix of all data points from all tasks using the multi-task kernel, and $\tilde{\mathbf{K}} = \mathbf{K} + \frac{1}{2}\boldsymbol{\Lambda}$ where $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal element is equal to n_i if the corresponding data point belongs to the ith task. So the kernel function for calculating $\tilde{\mathbf{K}}$ is $\tilde{k}_{MT}(\mathbf{x}_{j_1}^{i_1}, \mathbf{x}_{j_2}^{i_2}) = k_{MT}(\mathbf{x}_{j_1}^{i_1}, \mathbf{x}_{j_2}^{i_2}) + \frac{n_{i_1}}{2}\delta(i_1, i_2)\delta(j_1, j_2)$ where $\delta(\cdot, \cdot)$ is the Kronecker delta.

Note that for multiple tasks, there are m constraints in problem (30) with one for each task. For the single-task setting, however, there is only one constraint in the dual form.

We define

$$F_j^i = -\frac{\partial h}{\partial \alpha_j^i} = \boldsymbol{\alpha}^T\tilde{\mathbf{k}}_j^i - y_j^i,$$

where $\tilde{\mathbf{k}}_j^i$ is a column of $\tilde{\mathbf{K}}$ corresponding to $\mathbf{x}_j^i$. The Lagrangian of the dual form is

$$\tilde{L} = \frac{1}{2}\boldsymbol{\alpha}^T\tilde{\mathbf{K}}\boldsymbol{\alpha} - \sum_{i,j}\alpha_j^i y_j^i - \sum_i \beta_i \sum_j \alpha_j^i. \tag{31}$$

The KKT conditions for the dual problem are

$$\frac{\tilde{L}}{\partial \alpha_j^i} = \beta_i - F_j^i = 0 \quad \forall i, j.$$

So the optimality conditions will hold at a given $\boldsymbol{\alpha}$ if and only if for all j we have $F_j^i = \beta_i$, that is, all $\{F_j^i\}_{j=1}^{n_i}$ are identical for $i = 1, \ldots, m$. We introduce an index triple (i, j, k) to define a violation at $\boldsymbol{\alpha}$ if $F_j^i \neq F_k^i$. Thus the optimality conditions will hold at $\boldsymbol{\alpha}$ if and only if there does not exist any index triple that defines a violation.

Suppose (i, j, k) defines a violation at some $\boldsymbol{\alpha}$. So we can adjust α_j^i and α_k^i to achieve an increase in h while maintaining the equality constraints $\sum_j \alpha_j^i = 0$ for $i = 1, \ldots, m$. We define the following update:

$$\begin{aligned}
&\tilde{\alpha}_j^i(t) = \alpha_j^i - t;\\
&\tilde{\alpha}_k^i(t) = \alpha_k^i + t;\\
&\text{other elements in } \boldsymbol{\alpha} \text{ remain fixed.}
\end{aligned}$$

The updated $\boldsymbol{\alpha}$ is denoted by $\tilde{\boldsymbol{\alpha}}(t)$. We define $\phi(t) = h(\tilde{\boldsymbol{\alpha}}(t))$ and maximize $\phi(t)$ to find the optimal $t^\star$. Since $\phi(t)$ is a quadratic function of t, $\phi(t) = \phi(0) + t\phi'(0) + \frac{t^2}{2}\phi''(0)$. So the optimal $t^\star$ can be calculated as

$$t^\star = -\frac{\phi'(0)}{\phi''(0)}. \tag{32}$$

It is easy to show that

$$\begin{aligned}
\phi'(t) &= \frac{\partial \phi(t)}{\partial t}\\
&= \frac{\partial \phi(t)}{\partial \tilde{\alpha}_j^i(t)}\frac{\partial \tilde{\alpha}_j^i(t)}{\partial t} + \frac{\partial \phi(t)}{\partial \tilde{\alpha}_k^i(t)}\frac{\partial \tilde{\alpha}_k^i(t)}{\partial t}\\
&= \tilde{F}_j^i(t) - \tilde{F}_k^i(t)\\
\phi''(t) &= \frac{\partial \phi'(t)}{\partial t}\\
&= \frac{\partial \phi'(t)}{\partial \tilde{\alpha}_j^i(t)}\frac{\partial \tilde{\alpha}_j^i(t)}{\partial t} + \frac{\partial \phi'(t)}{\partial \tilde{\alpha}_k^i(t)}\frac{\partial \tilde{\alpha}_k^i(t)}{\partial t}\\
&= 2k_{MT}(\mathbf{x}_j^i, \mathbf{x}_k^i) - k_{MT}(\mathbf{x}_j^i, \mathbf{x}_j^i) - k_{MT}(\mathbf{x}_k^i, \mathbf{x}_k^i) - n_i,
\end{aligned}$$

where $\tilde{F}_j^i(t)$ is the value of F_j^i at $\tilde{\boldsymbol{\alpha}}(t)$. So

$$t^\star = -\frac{F_j^i - F_k^i}{\eta}, \tag{33}$$

where $\eta = \phi''(0)$ is a constant. After updating $\boldsymbol{\alpha}$, we can update F_q^p for all p, q as well as

h as:

$$(F_q^p)^{new} = F_q^p + \tilde{k}_{MT}(\mathbf{x}_j^i, \mathbf{x}_q^p)[\tilde{\alpha}_j^i(t^\star) - \alpha_j^i] + \tilde{k}_{MT}(\mathbf{x}_k^i, \mathbf{x}_q^p)[\tilde{\alpha}_k^i(t^\star) - \alpha_k^i] \quad (34)$$

$$h^{new} = h^{old} + \phi(t^\star) - \phi(0) = h^{old} - \frac{\eta(t^\star)^2}{2}. \quad (35)$$

The SMO algorithm is an iterative method and we need to define the stopping criterion. Similar to the SMO algorithm for SVM which uses the duality gap to define the stopping criterion, we also use a similar criterion here. When given an $\boldsymbol{\alpha}$, let E denote the current primal objective function value, h the dual objective function value, $E^\star$ the optimal primal objective function value, and $h^\star$ the optimal dual objective function value. By the Wolfe duality, we have

$$E \geq E^\star = h^\star \geq h.$$

Since $E^\star$ and $h^\star$ are unknown, we define the duality gap as $D_{gap} = E - h$. So the stopping criterion is defined as $D_{gap} \leq \epsilon h$ where ϵ is a small constant. From this stopping criterion, we can get

$$E - E^\star \leq D_{gap} \leq \epsilon h \leq \epsilon E.$$

Next we show how to calculate D_{gap} in term of $\{F_j^i\}$ and $\boldsymbol{\alpha}$. From the constraints in the primal form, we can get

$$\begin{aligned} \varepsilon_j^i &= y_j^i - (\mathbf{w}_i^T \Phi(\mathbf{x}_j^i) + b_i) \\ &= \frac{n_i \alpha_j^i}{2} - b_i - F_j^i. \end{aligned}$$

Finally, D_{gap} can be calculated as

$$\begin{aligned} & D_{gap} \\ =& E - h \\ =& \sum_{i=1}^m \frac{1}{n_i} \sum_{j=1}^{n_i} (\varepsilon_j^i)^2 + \frac{\lambda_1}{2} \mathrm{tr}(\mathbf{W}\mathbf{W}^T) + \frac{\lambda_2}{2} \mathrm{tr}(\mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^T) - \left(-\frac{1}{2} \boldsymbol{\alpha}^T \tilde{\mathbf{K}} \boldsymbol{\alpha} + \sum_{i,j} \alpha_j^i y_j^i \right) \\ =& \sum_{i,j} \left[\alpha_j^i (F_j^i - \frac{n_i \alpha_j^i}{4}) + \frac{1}{n_i} (\varepsilon_j^i)^2 \right]. \end{aligned}$$

In the above calculation, we need to determine $\{b_i\}$. Here we choose $\{b_i\}$ to minimize D_{gap} at the given $\boldsymbol{\alpha}$, which is equivalent to minimizing $\sum_{i,j} (\varepsilon_j^i)^2$. So b_i can be calculated as

$$b_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \left(\frac{n_i \alpha_j^i}{2} - F_j^i \right). \quad (36)$$

The whole procedure for the SMO algorithm is summarized in Table VIII.

Appendix B

In this section, we show how to formulate problem (21) as a second-order cone programming (SOCP) problem.

Table VIII. SMO algorithm for problem (30)

```
Input: training data $\{\mathbf{x}_j^i, y_j^i\}_{j=1}^{n_i}$, $\epsilon$
Initialize $\boldsymbol{\alpha}$ as a zero vector;
Initialize $\{F_j^i\}$ and $h$ according to $\boldsymbol{\alpha}$;
Repeat
    Find a triple $(i, j, k)$ that defines a violation for each task;
    Calculate the optimal adjusted value $t^\star$ using Eq. (33);
    Update $\{F_j^i\}$ and $h$ according to Eqs. (34) and (35);
    Calculate $\{b_i\}$ according to Eq. (36)
Until $D_{gap} \leq \epsilon f$
Output: $\boldsymbol{\alpha}$ and $\mathbf{b}$.
```

We write $\mathbf{W}_{\tilde{m}}^T \mathbf{W}_{\tilde{m}} = \begin{pmatrix} \boldsymbol{\Psi}_{11} & \boldsymbol{\Psi}_{12} \\ \boldsymbol{\Psi}_{12}^T & \Psi_{22} \end{pmatrix}$ where $\boldsymbol{\Psi}_{11} \in \mathbb{R}^{m\times m}$, $\boldsymbol{\Psi}_{12} \in \mathbb{R}^{m\times 1}$ and $\Psi_{22} \in \mathbb{R}$. Then $\tilde{\boldsymbol{\Omega}} - t\mathbf{W}_{\tilde{m}}^T \mathbf{W}_{\tilde{m}} \succeq \mathbf{0}$ is equivalent to

$$\begin{aligned} &(1-\sigma)\boldsymbol{\Omega} - t\boldsymbol{\Psi}_{11} \succeq \mathbf{0} \\ &\sigma - t\Psi_{22} \geq (\boldsymbol{\omega}_{\tilde{m}} - t\boldsymbol{\Psi}_{12})^T \Big((1-\sigma)\boldsymbol{\Omega} - t\boldsymbol{\Psi}_{11}\Big)^{-1} (\boldsymbol{\omega}_{\tilde{m}} - t\boldsymbol{\Psi}_{12}), \end{aligned}$$

which can be reformulated as

$$\begin{aligned} &(1-\sigma)\mathbf{I}_m - t\boldsymbol{\Omega}^{-\frac{1}{2}}\boldsymbol{\Psi}_{11}\boldsymbol{\Omega}^{-\frac{1}{2}} \succeq \mathbf{0} \\ &\sigma - t\Psi_{22} \geq (\boldsymbol{\omega}_{\tilde{m}} - t\boldsymbol{\Psi}_{12})^T \boldsymbol{\Omega}^{-\frac{1}{2}} \Big((1-\sigma)\mathbf{I}_m - t\boldsymbol{\Omega}^{-\frac{1}{2}}\boldsymbol{\Psi}_{11}\boldsymbol{\Omega}^{-\frac{1}{2}}\Big)^{-1} \boldsymbol{\Omega}^{-\frac{1}{2}} (\boldsymbol{\omega}_{\tilde{m}} - t\boldsymbol{\Psi}_{12}), \end{aligned}$$

where $\boldsymbol{\Omega}^{-\frac{1}{2}}$ can be computed in advance. Let $\tilde{\boldsymbol{\Psi}}_{11} = \boldsymbol{\Omega}^{-\frac{1}{2}}\boldsymbol{\Psi}_{11}\boldsymbol{\Omega}^{-\frac{1}{2}}$, $\mathbf{U}$ and $\lambda_1, \ldots, \lambda_m$ denote the eigenvector matrix and eigenvalues of $\tilde{\boldsymbol{\Psi}}_{11}$ with $\lambda_1 \geq \ldots \geq \lambda_m \geq 0$. Then

$$(1-\sigma)\mathbf{I}_m - t\tilde{\boldsymbol{\Psi}}_{11} \succeq \mathbf{0} \Longleftrightarrow 1 - \sigma \geq \lambda_1 t$$

and

$$\Big((1-\sigma)\mathbf{I}_m - t\tilde{\boldsymbol{\Psi}}_{11}\Big)^{-1} = \mathbf{U}\,\mathrm{diag}\Big(\frac{1}{1-\sigma-t\lambda_1}, \ldots, \frac{1}{1-\sigma-t\lambda_m}\Big)\,\mathbf{U}^T,$$

where the operator diag$(\cdot)$ converts a vector to a diagonal matrix. Combining the above results, problem (20) is formulated as

$$\begin{aligned} \min_{\boldsymbol{\omega}_{\tilde{m}}, \sigma, \mathbf{f}, t} \quad & -t \\ \text{s.t.} \quad & 1 - \sigma \geq t\lambda_1 \\ & \mathbf{f} = \mathbf{U}^T \boldsymbol{\Omega}^{-\frac{1}{2}} (\boldsymbol{\omega}_{\tilde{m}} - t\boldsymbol{\Psi}_{12}) \\ & \sum_{j=1}^m \frac{f_j^2}{1-\sigma-t\lambda_j} \leq \sigma - t\Psi_{22} \\ & \boldsymbol{\omega}_{\tilde{m}}^T \boldsymbol{\Omega}^{-1} \boldsymbol{\omega}_{\tilde{m}} \leq \sigma - \sigma^2, \end{aligned} \tag{37}$$

where f_j is the jth element of $\mathbf{f}$. By introducing new variables h_j and r_j $(j = 1, \ldots, m)$,

(37) is reformulated as

$$\begin{aligned}
\min_{\boldsymbol{\omega}_{\tilde{m}},\sigma,\mathbf{f},t,\mathbf{h},\mathbf{r}} \quad & -t \\
\text{s.t.} \quad & 1-\sigma \geq t\lambda_1 \\
& \mathbf{f} = \mathbf{U}^T\boldsymbol{\Omega}^{-\frac{1}{2}}(\boldsymbol{\omega}_{\tilde{m}} - t\boldsymbol{\Psi}_{12}) \\
& \sum_{j=1}^{m} h_j \leq \sigma - t\Psi_{22} \\
& r_j = 1-\sigma-t\lambda_j \ \forall j \\
& \frac{f_j^2}{r_j} \leq h_j \ \forall j \\
& \boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} \leq \sigma - \sigma^2.
\end{aligned} \tag{38}$$

Since

$$\frac{f_j^2}{r_j} \leq h_j \ (r_j, h_j > 0) \iff \left\| \begin{pmatrix} f_j \\ \frac{r_j-h_j}{2} \end{pmatrix} \right\|_2 \leq \frac{r_j+h_j}{2}$$

and

$$\boldsymbol{\omega}_{\tilde{m}}^T\boldsymbol{\Omega}^{-1}\boldsymbol{\omega}_{\tilde{m}} \leq \sigma - \sigma^2 \iff \left\| \begin{pmatrix} \boldsymbol{\Omega}^{-\frac{1}{2}}\boldsymbol{\omega}_{\tilde{m}} \\ \frac{\sigma-1}{2} \\ \sigma \end{pmatrix} \right\|_2 \leq \frac{\sigma+1}{2},$$

problem (38) is an SOCP problem [Lobo et al. 1998] with $O(m)$ variables and $O(m)$ constraints. Then we can use a standard solver to solve problem (38) efficiently.

Appendix C

In this section, we will prove that the upper bound in Eq. (25) is tighter than that in Eq. (24), which means that the following inequality holds:

$$\operatorname{tr}\big(\mathbf{M}^{-1}(\mathbf{I}_d + \mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^T)\big) + \ln|\mathbf{M}| - d \leq \operatorname{tr}(\mathbf{W}\boldsymbol{\Omega}^{-1}\mathbf{W}^T), \tag{39}$$

where $\mathbf{M} = \mathbf{I}_d + \mathbf{W}^{(t)}(\boldsymbol{\Omega}^{(t)})^{-1}(\mathbf{W}^{(t)})^T$ or, more generally, any positive definite matrix.

To prove (39), we first prove the following Lemma.

LEMMA 2. *For two $d \times d$ positive definite matrices $\mathbf{A}$ and $\mathbf{B}$, the following equality holds:*

$$\operatorname{tr}(\mathbf{A}^{-1}\mathbf{B}) + \ln|\mathbf{A}| \leq \operatorname{tr}(\mathbf{B}).$$

Proof:
Consider the function $F(\mathbf{X}) = \operatorname{tr}(\mathbf{X}^{-1}\mathbf{B}) + \ln|\mathbf{X}|$. We set its derivative to zero to get

$$\frac{\partial F(\mathbf{X})}{\partial \mathbf{X}} = \mathbf{X}^{-1} - \mathbf{X}^{-1}\mathbf{B}\mathbf{X}^{-1} = 0 \ \Rightarrow \ \mathbf{X} = \mathbf{B}.$$

It is easy to prove that the maximum of $F(\mathbf{X})$ holds at $\mathbf{X} = \mathbf{B}$, which implies

$$\operatorname{tr}(\mathbf{A}^{-1}\mathbf{B}) + \ln|\mathbf{A}| = F(\mathbf{A}) \leq F(\mathbf{B}) = \ln|\mathbf{B}| + d.$$

By using Lemma 1, we can get

$$\ln|\mathbf{B}| + d \leq \operatorname{tr}(\mathbf{B}).$$

Finally, we can get

$$\mathrm{tr}(\mathbf{A}^{-1}\mathbf{B}) + \ln|\mathbf{A}| \leq \ln|\mathbf{B}| + d \leq \mathrm{tr}(\mathbf{B}),$$

which is the conclusion. ■

By using Lemma 2 where we let $\mathbf{A} = \mathbf{M}$ and $\mathbf{B} = \mathbf{I}_d + \mathbf{W}\mathbf{\Omega}^{-1}\mathbf{W}^T$, we can prove (39).

Appendix D

In this section, we provide the proofs for Eqs. (28) and (29).

Before we present our proofs, we first review some relevant properties of the matrix-variate normal distribution as given in [Gupta and Nagar 2000].

LEMMA 3. ([Gupta and Nagar 2000], Corollary 2.3.10.1) *If* $\mathbf{X} \sim \mathcal{MN}_{q\times s}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$*,* $\mathbf{d}\in\mathbb{R}^q$ *and* $\mathbf{c}\in\mathbb{R}^s$*, then*

$$\mathbf{d}^T\mathbf{X}\mathbf{c} \sim \mathcal{N}(\mathbf{d}^T\mathbf{M}\mathbf{c}, (\mathbf{d}^T\mathbf{\Sigma}\mathbf{d})(\mathbf{c}^T\mathbf{\Psi}\mathbf{c})).$$

LEMMA 4. ([Gupta and Nagar 2000], Theorem 2.3.5) *If* $\mathbf{X} \sim \mathcal{MN}_{q\times s}(\mathbf{M}, \mathbf{\Sigma}\otimes\mathbf{\Psi})$ *and* $\mathbf{A}\in\mathbb{R}^{s\times s}$*, then*

$$\mathbb{E}(\mathbf{X}\mathbf{A}\mathbf{X}^T) = \mathrm{tr}(\mathbf{A}^T\mathbf{\Psi})\mathbf{\Sigma} + \mathbf{M}\mathbf{A}\mathbf{M}^T.$$

For Eq. (28), using Lemma 3 and the fact that $\mathbf{W} \sim \mathcal{MN}_{d'\times m}(\mathbf{0}_{d'\times m}, \mathbf{I}_{d'}\otimes\mathbf{\Sigma})$, we can get

$$f_j^i \overset{\text{def}}{=} \phi(\mathbf{x}_j^i)^T\mathbf{w}_i = \phi(\mathbf{x}_j^i)^T\mathbf{W}\mathbf{e}_{m,i} \sim \mathcal{N}(0, (\phi(\mathbf{x}_j^i)^T\mathbf{I}_{d'}\phi(\mathbf{x}_j^i))(\mathbf{e}_{m,i}^T\mathbf{\Sigma}\mathbf{e}_{m,i})).$$

Since $\phi(\mathbf{x}_j^i)^T\mathbf{I}_{d'}\phi(\mathbf{x}_j^i) = k(\mathbf{x}_j^i, \mathbf{x}_j^i)$ and $\mathbf{e}_{m,i}^T\mathbf{\Sigma}\mathbf{e}_{m,i} = \Sigma_{ii}$, we can get

$$f_j^i \sim \mathcal{N}(0, \Sigma_{ii}k(\mathbf{x}_j^i, \mathbf{x}_j^i)).$$

For Eq. (29), we have

$$\begin{aligned}\langle f_j^i, f_s^r\rangle &= \int \phi(\mathbf{x}_j^i)^T\mathbf{W}\mathbf{e}_{m,i}\mathbf{e}_{m,r}^T\mathbf{W}^T\phi(\mathbf{x}_s^r)p(\mathbf{W})d\mathbf{W}\\ &= \phi(\mathbf{x}_j^i)^T\mathbb{E}(\mathbf{W}\mathbf{e}_{m,i}\mathbf{e}_{m,r}^T\mathbf{W}^T)\phi(\mathbf{x}_s^r),\end{aligned}$$

then using Lemma 4 and the fact that $\mathbf{W} \sim \mathcal{MN}_{d'\times m}(\mathbf{0}_{d'\times m}, \mathbf{I}_{d'}\otimes\mathbf{\Sigma})$, we can get

$$\begin{aligned}\langle f_j^i, f_s^r\rangle &= \phi(\mathbf{x}_j^i)^T\mathrm{tr}(\mathbf{e}_{m,r}\mathbf{e}_{m,i}^T\mathbf{\Sigma})\mathbf{I}_{d'}\phi(\mathbf{x}_s^r)\\ &= \mathrm{tr}(\mathbf{e}_{m,r}\mathbf{e}_{m,i}^T\mathbf{\Sigma})k(\mathbf{x}_j^i, \mathbf{x}_s^r)\\ &= \mathbf{e}_{m,i}^T\mathbf{\Sigma}\mathbf{e}_{m,r}k(\mathbf{x}_j^i, \mathbf{x}_s^r)\\ &= \Sigma_{ir}k(\mathbf{x}_j^i, \mathbf{x}_s^r).\end{aligned}$$

The second last equation holds because $\mathbf{e}_{m,i}$ and $\mathbf{e}_{m,r}$ are two vectors.