Multi-Task Learning with Group-Specific Feature Space Sharing

Niloofar Yousefi[1(✉)], Michael Georgiopoulos[1], and Georgios C. Anagnostopoulos[2]

[1] Department of Electrical Engineering and Computer Science, University of Central Florida, 4000 Central Florida Blvd., Orlando, FL 32816, USA
{niloofar.yousefi,michaelg}@ucf.edu
[2] Department of Electrical and Computer Engineering, Florida Institute of Technology, 150 W. University Blvd., Melbourne, FL 32901, USA
georgio@fit.edu

Abstract. When faced with learning a set of inter-related tasks from a limited amount of usable data, learning each task independently may lead to poor generalization performance. (MTL) exploits the latent relations between tasks and overcomes data scarcity limitations by co-learning all these tasks simultaneously to offer improved performance. We propose a novel Multi-Task Multiple Kernel Learning framework based on Support Vector Machines for binary classification tasks. By considering pair-wise task affinity in terms of similarity between a pair's respective feature spaces, the new framework, compared to other similar MTL approaches, offers a high degree of flexibility in determining how similar feature spaces should be, as well as which pairs of tasks should share a common feature space in order to benefit overall performance. The associated optimization problem is solved via a block coordinate descent, which employs a consensus-form Alternating Direction Method of Multipliers algorithm to optimize the Multiple Kernel Learning weights and, hence, to determine task affinities. Empirical evaluation on seven data sets exhibits a statistically significant improvement of our framework's results compared to the ones of several other Clustered Multi-Task Learning methods.

1 Introduction

Multi-Task Learning (MTL) is a machine learning paradigm, where several related task are learnt simultaneously with the hope that, by sharing information among tasks, the generalization performance of each task will be improved. The underlying assumption behind this paradigm is that the tasks are related to each other. Thus, it is crucial how to capture task relatedness and incorporate it into an MTL framework. Although, many different MTL methods [1,7,12,15,18,27] have been proposed, which differ in how the relatedness across multiple tasks is modeled, they all utilize the parameter or structure sharing strategy to capture the task relatedness.

However, the previous methods are restricted in the sense that they assume all tasks are similarly related to each other and can equally contribute to the joint

A. Appice et al. (Eds.): ECML PKDD 2015, Part II, LNAI 9285, pp. 120–136, 2015.
DOI: 10.1007/978-3-319-23525-7_8

learning process. This assumption can be violated in many practical applications as "outlier" tasks often exist. In this case, the effect of "negative transfer", *i.e.*, sharing information between irrelevant tasks, can lead to a degraded generalization performance.

To address this issue, several methods, along different directions, have been proposed to discover the inherent relationship among tasks. For example, some methods [3,26–28], use a regularized probabilistic setting, where sharing among tasks is done based on a common prior. These approaches are usually computationally expensive. Another family of approaches, known as the Clustered Multi-Task Learning (CMTL), assumes that tasks can be clustered into groups such that the tasks within each group are close to each other according to a notion of similarity. Based on the current literature, clustering strategies can be broadly classified into two categories: task-level CMTL and feature-level CMTL.

The first one, task-level CMTL, assumes that the model parameters used by all tasks within a group are close to each other. For example, in [2,13,17], the weight vectors of the tasks belonging to the same group are assumed to be similar to each other. However, the major limitations for these methods are: (i) that such an assumption might be too risky, as similarity among models does not imply that meaningful sharing of information can occur between tasks, and (ii) for these methods, the group structure (number of groups or basis tasks) is required to be known a priori.

The other strategy for task clustering, referred to as feature-level CMTL, is based on the assumption that task relatedness can be modeled as learning shared features among the tasks within each group. For example, in [19] the tasks are clustered into different groups and it is assumed that tasks within the same group can jointly learn a shared feature representation. The resulting formulation leads to a non-convex objective, which is optimized using an alternating optimization algorithm converging to local optima, and suffers potentially from slow convergence. Another similar approach has been proposed in [25], which assumes that tasks should be related in terms of feature subsets. This study also leads to a non-convex co-clustering structure that captures task-feature relationship. These methods are restricted in the sense that they assume that tasks from different groups have nothing in common with each other. However, this assumption is not always realistic, as tasks in disjoint groups might still be inter-related, albeit weekly. Hence, assigning tasks into different groups may not take full advantage of MTL. Another feature-level clustering model has been proposed in [29], in which the cluster structure can vary from feature to feature. While, this model is more flexible compared to other CMTL methods, it is, however, more complicated and also less general compared to our framework, as it tries to find a shared feature representation for tasks by decomposing each task parameter into two parts: one to capture the shared structure between tasks and another to capture the variations specific to each task. This model is further extended in [16], where a multi-level structure has been introduced to learn task groups in the context of MTL. Interestingly, it has been shown that there is an equivalent relationship between CMTL and alternating structure optimization [30], wherein the basic idea is to identify a shared low-dimensional predictive structure for all tasks.

In this paper, we develop a new MTL model capable of modeling a more general type of task relationship, where the tasks are implicitly grouped according to a notion of feature similarity. In our framework, the tasks are not forced to have a common feature space; instead, the data automatically suggests a flexible group structure, in which a common, similar or even distinct feature spaces can be determined between different pairs of tasks. Additionally, our MTL framework is kernel-based and, thus, may take advantage of the non-linearity introduced by the feature mapping of the associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Also, to avoid a degradation in generalization performance due to choosing an inappropriate kernel function, our framework employs a Multiple Kernel Learning (MKL) strategy [21], hence, rendering it a Multi-Task Multiple Kernel Learning (MT-MKL) approach.

It is worth mentioning that a widely adopted practice for combining kernels is to place an L_p-norm constraint on the combination coefficients $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_M]$, which are learned during training. For example, a conically combination of task objectives with an L_p-norm feasible region is introduced in [23] and further extended in [22]. Also, another method introduced in [24] proposes a partially shared kernel function $k_t \triangleq \sum_{m=1}^{M} (\mu^m + \lambda_t^m) k_m$, along with L_1-norm constraints on $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$. The main advantage of such a method over the traditional MT-MKL methods, which consider a common kernel function for all tasks (by letting $\lambda_t^m = 0, \forall t, m$), is that it allows tasks to have their own task-specific feature spaces and, potentially, alleviate the effect of negative transfer. However, popular MKL formulations in the context of MTL, such as this one, are capable of modeling two types of tasks: those that share a global, common feature space and those that employ their own, task-specific feature space. In this work we propose a more flexible framework, which, in addition to allowing some tasks to use their own specific feature spaces (to avoid negative transfer learning), it permits forming arbitrary groups of tasks sharing the same, group-specific (instead of a single, global), common feature space, whenever warranted by the data. This is accomplished by considering a group lasso regularizer applied to the set of all pair-wise differences of task-specific MKL weights. For no regularization penalty, each task is learned independently of each other and will utilize its own feature space. As the regularization penalty increases, pairs of MKL weights are forced to equal each other leading the corresponding pairs of tasks to share a common feature space. We demonstrate that the resulting optimization problem can be solved by employing a 2-block coordinate descent approach, whose first block consists of the Support Vector Machine (SVM) weights for each task and which can be optimized efficiently using existing solvers, while its second block comprises the MKL weights from all tasks and is optimized via a consensus-form, Alternating Direction Method of Multipliers (ADMM)-based step.

The rest of the paper is organized as follows: In Sect. 2 we describe our formulation for jointly learning the optimal feature spaces and the parameters of all the tasks. Sect. 3 provides an optimization technique to solve our non-smooth convex optimization problem derived in Sect. 2. Sect. 4 presents a Rademacher complexity-based generalization bound for the hypothesis space corresponding to our model. Experiments are provided in Sect. 5, which demonstrate the

effectiveness of our proposed model compared to several MTL methods. Finally, in Sect. 6 we conclude our work and briefly summarize our findings.

Notation: In what follows, we use the following notational conventions: vectors and matrices are depicted in bold face. A prime $'$ denotes vector/matrix transposition. The ordering symbols $\succeq$ and $\preceq$ when applied to vectors stand for the corresponding component-wise relations. If $\mathbb{Z}_+$ is the set of postivie integers, for a given $S \in \mathbb{Z}_+$, we define $\mathbb{N}_S \triangleq \{1, \ldots, S\}$. Additional notation is defined in the text as needed.

2 Formulation

Assume T supervised learning tasks, each with a training set $\{(x_t^n, y_t^n)\}_{n=1}^{n_t}, t \in \mathbb{N}_T$, which is sampled from an unknown distribution $P_t(x, y)$ on $\mathcal{X} \times \{-1, 1\}$. Here, $\mathcal{X}$ denotes the native space of samples for all tasks and ± 1 are the associated labels. Without loss of generality, we will assume an equal number n of training samples per task. The objective is to learn T binary classification tasks using discriminative functions $f_t(\boldsymbol{x}) \triangleq \langle \boldsymbol{w}_t, \boldsymbol{\phi}_t(\boldsymbol{x}) \rangle_{\mathcal{H}_{t,\boldsymbol{\theta}}} + b_t$ for $t \in \mathbb{N}_T$, where $\boldsymbol{w}_t$ is the weight vector associated to task t. Moreover, the feature space of task t is served by $\mathcal{H}_{t,\boldsymbol{\theta}} = \bigoplus_{m=1}^{M} \sqrt{\theta_t^m} \mathcal{H}_m$ with induced feature mapping $\phi_t \triangleq [\sqrt{\theta_t^1}\phi_1{}' \cdots \sqrt{\theta_t^M}\phi_M{}']'$ and endowed with the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}_{t,\boldsymbol{\theta}}} = \sum_{m=1}^{M} \theta_t^m \langle \cdot, \cdot \rangle_{\mathcal{H}_m}$. The reproducing kernel function for this feature space is given as $k_t(x_t^i, x_t^j) = \sum_{m=1}^{M} \theta_t^m k_m(x_t^i, x_t^j)$ for all $x_t^i, x_t^j \in \mathcal{X}$. In our framework, we attempt to learn the $\boldsymbol{w}_t$'s and b_t's jointly with the $\boldsymbol{\theta}_t$'s via the following regularized risk minimization problem:

$$\min_{\boldsymbol{w} \in \Omega(\boldsymbol{w}), \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta}), \boldsymbol{b}} \sum_{t=1}^{T} \frac{\|\boldsymbol{w}_t\|^2}{2} + C \sum_{t=1}^{T} \sum_{i=1}^{n} \left[1 - \boldsymbol{y}_t^i f_t(x_t^i)\right]_+ + \lambda \sum_{t=1}^{T-1} \sum_{s>t}^{T} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2$$
$$\Omega(\boldsymbol{w}) \triangleq \{\boldsymbol{w} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T) : \boldsymbol{w}_t \in \mathcal{H}_{t,\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Omega(\boldsymbol{\theta})\}$$
$$\Omega(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} = (\boldsymbol{\theta}_t, \cdots, \boldsymbol{\theta}_T) : \boldsymbol{\theta}_t \succeq \mathbf{0}, \|\boldsymbol{\theta}_t\|_1 \leq 1, \forall t \in \mathbb{N}_T\} \quad (1)$$

where $\boldsymbol{w} \triangleq (\boldsymbol{w}_t, \cdots, \boldsymbol{w}_T)$ and $\boldsymbol{\theta} \triangleq (\boldsymbol{\theta}_t, \cdots, \boldsymbol{\theta}_T)$, $\Omega(\boldsymbol{w})$ and $\Omega(\boldsymbol{\theta})$ are the corresponding feasible sets for $\boldsymbol{w}$ and $\boldsymbol{\theta}$ respectively, and $[u]_+ = \max\{u, 0\}$, $u \in \mathbb{R}$ denotes the hinge function. Finally, C and λ are non-negative regularization parameters.

The last term in Problem 1 is the sum of pairwise differences between the tasks' feature weight vectors. For each pair of $(\boldsymbol{\theta}_t, \boldsymbol{\theta}_s)$, the pairwise penalty $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2$ may favor a small number of non-identical $\boldsymbol{\theta}_t$. Therefore, it ensures that a flexible (common, similar or distinct) feature space, will be selected between tasks t and s. In this manner, a flexible group structure of shared features across multiple tasks can be achieved by this framework. It is also worth mentioning that two special cases are covered by the proposed model: (i) if $\lambda \to \infty$ (λ is only required to be sufficiently large), for all task pairs $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 \to 0$ and, thus, all tasks share a single common feature space. (ii) As $\lambda \to 0$, the proposed model reduces to T independent classification tasks.

It is easy to verify that Problem 1 is a convex minimization problem, which can be solved using a block coordinate descent method alternating between the minimization with respect to $\boldsymbol{\theta}$ and the $(\boldsymbol{w}, \boldsymbol{b})$ pair. Motivated by the non-smooth nature of the last regularization term, in Sect. 3 we develop a consensus version of the ADMM to solve the minimization problem with respect to $\boldsymbol{\theta}$.

3 The Proposed Consensus Optimization Algorithm

Problem 1 can be formulated as the following equivalent problem, which entails T inter-related SVM training problems:

$$\begin{aligned}
\min_{\boldsymbol{\theta},\boldsymbol{w},\boldsymbol{b},\boldsymbol{\xi}} & \sum_{t=1}^{T}\sum_{m=1}^{M}\frac{\|\boldsymbol{w}_t^m\|_{\mathcal{H}_m}^2}{2\theta_t^m} + C\sum_{t=1}^{T}\sum_{i=1}^{n}\xi_t^i + \lambda\sum_{t=1}^{T-1}\sum_{s>t}^{T}\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 \\
& s.t.\ y_t^i\left(\langle w_t, \phi(x_t^i)\rangle_{\mathcal{H}_t} + b_t\right) \geq 1 - \xi_t^i,\ \xi_t^i \geq 0,\ \forall\, t \in \mathbb{N}_T, i \in \mathbb{N}_n \\
& \boldsymbol{\theta}_t \succeq \mathbf{0}, \|\boldsymbol{\theta}_t\|_1 \leq 1, \forall\, t \in \mathbb{N}_T
\end{aligned} \tag{2}$$

It can be shown that the primal-dual form of Problem 2 with respect to $\boldsymbol{\theta}$ and $\{\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\xi}\}$ is given by

$$\begin{aligned}
\min_{\boldsymbol{\theta}_t \in \Omega(\boldsymbol{\theta})} \max_{\boldsymbol{\alpha}_t \in \Omega(\boldsymbol{\alpha})} & \sum_{t=1}^{T}\boldsymbol{\alpha}_t^{'}\mathbf{1}_n - \frac{1}{2}\sum_{t=1}^{T}\sum_{m=1}^{M}\theta_t^m(\boldsymbol{\alpha}_t^{'}Y_tK_t^mY_t\boldsymbol{\alpha}_t) + \lambda\sum_{t=1}^{T-1}\sum_{s>t}^{T}\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 \\
& \Omega(\boldsymbol{\alpha}) \triangleq \{\boldsymbol{\alpha} = (\boldsymbol{\alpha}_t, \cdots, \boldsymbol{\alpha}_T) : 0 \preceq \boldsymbol{\alpha}_t \preceq C\mathbf{1}_n,\ \boldsymbol{\alpha}_t^{'}\boldsymbol{y}_t = 0,\ \forall\, t \in \mathbb{N}_T\} \\
& \Omega(\boldsymbol{\theta}) \triangleq \{\boldsymbol{\theta} = (\boldsymbol{\theta}_t, \cdots, \boldsymbol{\theta}_T) : \boldsymbol{\theta}_t \succeq \mathbf{0}, \|\boldsymbol{\theta}_t\|_1 \leq 1, \forall\, t \in \mathbb{N}_T\}
\end{aligned} \tag{3}$$

where $\mathbf{1}_n$ is a vector containing n 1's, $\boldsymbol{Y}_t \triangleq diag(\boldsymbol{y}_t)$, $\boldsymbol{K}_t^m \in \mathbb{R}^{n\times n}$ is the kernel matrix, whose (i,j) entry is given as $k_m(x_t^i, x_t^j)$, $\boldsymbol{\theta}_t \triangleq [\theta_t^1, \ldots, \theta_t^M]'$, and $\boldsymbol{\alpha}_t$ is the Lagrangian dual variable for the minimization problem w.r.t.$\{\boldsymbol{w}_t, b_t, \boldsymbol{\xi}_t\}$.

It is not hard to verify that the optimal objective value of the dual problem is equal to the optimal objective value of the primal one, as the strong duality holds for the primal-dual optimization problems w.r.t.$\{\boldsymbol{w}, \boldsymbol{b}, \boldsymbol{\xi}\}$ and $\boldsymbol{\alpha}$ respectively. Therefore, a block coordinate descent framework[1] can be applied to decompose Problem 3 into two subproblems. The first subproblem, which is the maximization problem with respect to $\boldsymbol{\alpha}$, can be efficiently solved via `LIBSVM` [8], and the second subproblem, which is the minimization problem with respect to $\boldsymbol{\theta}$, takes the form

$$\begin{aligned}
\min_{\boldsymbol{\theta}_t} & \lambda\sum_{t=1}^{T-1}\sum_{s>t}^{T}\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 + \sum_{t=1}^{T}\boldsymbol{\theta}_t^{'}\boldsymbol{q}_t \\
& s.t.\ \boldsymbol{\theta}_t \succeq \mathbf{0}, \|\boldsymbol{\theta}_t\|_1 \leq 1,\ \forall\, t \in \mathbb{N}_T
\end{aligned} \tag{4}$$

[1] A `MATLAB`® implementation of our framework is available at https://github.com/niloofaryousefi/ECML2015

where we defined $q_t^m \triangleq -\frac{1}{2}\boldsymbol{\alpha}_t^{'} Y_t K_t^m Y_t \boldsymbol{\alpha}_t$ and $\boldsymbol{q}_t \triangleq [q_t^1, \ldots, q_t^M]'$. Due to the non-smooth nature of Problem 4, we derive a consensus ADMM-based optimization algorithm to solve it efficiently. Based on the exposition provided in Sections 5 and 7 of [6], it is straightforward to verify that Problem 4 can be written in ADMM form as

$$\begin{aligned} \min_{\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{z}} \lambda \sum_{i=1}^{N} h_i(\boldsymbol{s}_i) + g(\boldsymbol{\theta}) + I_{\Omega(\boldsymbol{\theta})}(\boldsymbol{z}) \\ s.t.\ \boldsymbol{s}_i - \tilde{\boldsymbol{\theta}}_i = \boldsymbol{0},\ i \in \mathbb{N}_N \\ \boldsymbol{z} - \boldsymbol{\theta} = \boldsymbol{0} \end{aligned} \tag{5}$$

where $N \triangleq \frac{T(T-1)}{2}$, and the local variable $\boldsymbol{s}_i \in \mathbb{R}^{2M}$ consists of two vector variables $(\boldsymbol{s}_i)_j$ and $(\boldsymbol{s}_i)_{j'}$, where $(\boldsymbol{s}_i)_j = \boldsymbol{\theta}_{\mathcal{M}(i,j)}$. Note that the index mapping $t = \mathcal{M}(i,j)$ maps the j^{th} component of the local variable $\boldsymbol{s}_i$ to the t^{th} component of the global variable $\boldsymbol{\theta}$. Also, $\tilde{\boldsymbol{\theta}}_i$ can be considered as the global variable's idea of what the local variable $\boldsymbol{s}_i$ should be. Moreover, for each i, the function $h_i(\boldsymbol{s}_i)$ is defined as $\|(\boldsymbol{s}_i)_j - (\boldsymbol{s}_i)_{j'}\|_2$, and the objective term $g(\boldsymbol{\theta})$ is given as $\sum_{t=1}^{T} \boldsymbol{\theta}_t^{'} \boldsymbol{q}_t$. Finally, $I_{\Omega(\boldsymbol{\theta})}(\boldsymbol{z})$ is the indicator function for the constraint set $\boldsymbol{\theta}$ (*i.e.*, $I_{\Omega(\boldsymbol{\theta})}(\boldsymbol{z}) = 0$ for $\boldsymbol{z} \in \Omega(\boldsymbol{\theta})$, and $I_{\Omega(\boldsymbol{\theta})}(\boldsymbol{z}) = \infty$ for $\boldsymbol{z} \notin \Omega(\boldsymbol{\theta})$).

The augmented Lagrangian (using scaled dual variables) for Problem 5 is

$$\begin{aligned} L_\rho(\boldsymbol{s},\boldsymbol{\theta},\boldsymbol{z},\boldsymbol{u},\boldsymbol{v}) = &\lambda \sum_{i=1}^{N} h_i(\boldsymbol{s}_i) + g(\boldsymbol{\theta}) + I_{\Omega(\boldsymbol{\theta})}(\boldsymbol{z}) + (\rho/2) \sum_{i=1}^{N} \|\boldsymbol{s}_i - \tilde{\boldsymbol{\theta}}_i + \boldsymbol{u}_i\|_2^2 \\ &+ (\rho/2)\|\boldsymbol{z} - \boldsymbol{\theta} + \boldsymbol{v}\|_2^2, \end{aligned} \tag{6}$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}$ are the dual variables for the constraints $\boldsymbol{s}_i = \tilde{\boldsymbol{\theta}}_i$ and $\boldsymbol{z} = \boldsymbol{\theta}$ respectively. Applying ADMM on the Lagrangian function given in (6), the following steps are carried out in the k^{th} iteration

$$\boldsymbol{s}_i^{k+1} = \arg\min_{\boldsymbol{s}_i} \{\lambda h_i(\boldsymbol{s}_i) + (\rho/2)\|\boldsymbol{s}_i - \tilde{\boldsymbol{\theta}}_i^k + \boldsymbol{u}_i^k\|_2^2\} \tag{7}$$

$$\boldsymbol{\theta}^{k+1} = \arg\min_{\boldsymbol{\theta}} \{g(\boldsymbol{\theta}) + (\rho/2) \sum_{i=1}^{N} \|\boldsymbol{s}_i^{k+1} - \tilde{\boldsymbol{\theta}}_i + \boldsymbol{u}_i^k\|_2^2 + (\rho/2)\|\boldsymbol{z}^k - \boldsymbol{\theta} + \boldsymbol{v}^k\|_2^2\} \tag{8}$$

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} \{I_{\Omega(\boldsymbol{\theta})}(\boldsymbol{z}) + (\rho/2)\|\boldsymbol{z} - \boldsymbol{\theta}^{k+1} + \boldsymbol{v}^k\|_2^2\} \tag{9}$$

$$\boldsymbol{u}_i^{k+1} = \boldsymbol{u}_i^k + \boldsymbol{s}_i^{k+1} - \tilde{\boldsymbol{\theta}}_i^{k+1} \tag{10}$$

$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \boldsymbol{z}^{k+1} - \boldsymbol{\theta}^{k+1} \tag{11}$$

where, for each $i \in \mathbb{N}_N$, the $\boldsymbol{s}$- and $\boldsymbol{u}$-updates can be carried out independently and in parallel. It is also worth mentioning that the $\boldsymbol{s}$-update is a proximal operator evaluation for $\|.\|_2$ which can be simplified to

$$\boldsymbol{s}_i^{k+1} = \mathcal{S}_{\lambda/\rho}(\tilde{\boldsymbol{\theta}}_i^k + \boldsymbol{u}_i^k),\ \forall\ i \in \mathbb{N}_N \tag{12}$$

where $\mathcal{S}_\kappa$ is the vector-valued soft thresholding (or shrinkage) operator and which is defined as

$$\mathcal{S}_\kappa(\boldsymbol{a}) \triangleq (1 - \kappa/\|\boldsymbol{a}\|_2)_+ \boldsymbol{a}, \quad \mathcal{S}_\kappa(0) \triangleq 0. \tag{13}$$

Furthermore, as the objective term g is separable in $\boldsymbol{\theta}_t$, the $\boldsymbol{\theta}$-update can be decomposed into T independent minimization problems, for which a closed from solution exists

$$\boldsymbol{\theta}_t^{k+1} = \frac{1}{T-1}\left[\sum_{\mathcal{M}(i,j)=t} \left((\boldsymbol{s}_i)_j^{k+1} + (\boldsymbol{u}_i)_j^k\right) + \left(\boldsymbol{z}_t^k + \boldsymbol{v}_t^k\right) - (1/\rho)\boldsymbol{q}_t\right], \ \forall\, t \in \mathbb{N}_T \tag{14}$$

Algorithm 1. Algorithm for solving Problem 3.

Require: $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_T, \boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_T, C, \lambda$
Ensure: $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_T$
1: **Initialize:** $\boldsymbol{\theta}_1^{(0)}, \ldots, \boldsymbol{\theta}_T^{(0)}$, $r = 1$
2: **Calculate:** Base kernel matrices K_t^m using $\boldsymbol{X}_t$'s for the T tasks and the M kernels.
3: **while** not converged **do**
4: $\quad \boldsymbol{\alpha}^{(r)} \leftarrow \arg\max_{\boldsymbol{\alpha}\in\Omega(\boldsymbol{\alpha})} \sum_{t=1}^T \boldsymbol{\alpha}_t^{'}\boldsymbol{e} - \frac{1}{2}\sum_{t=1}^T \sum_{m=1}^M (\theta_t^m)^{(r-1)} (\boldsymbol{\alpha}_t^{'} Y_t K_t^m Y_t \boldsymbol{\alpha}_t)$
5: $\quad (q_t^m)^{(r)} \leftarrow -\frac{1}{2}(\boldsymbol{\alpha}_t^{'})^{(r)} Y_t K_t^m Y_t (\boldsymbol{\alpha}_t)^{(r)}, \ \forall t, m$
6: $\quad \boldsymbol{\theta}^{(r)} \leftarrow \arg\min_{\boldsymbol{\theta}\in\Omega(\boldsymbol{\theta})} \lambda \sum_{t=1}^{T-1}\sum_{s>t}^T \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 + \sum_{t=1}^T \boldsymbol{\theta}_t^{'} \boldsymbol{q}_t^{(r)}$ using Algorithm 2
7: **end while**
8: $\boldsymbol{\alpha}^* = \boldsymbol{\alpha}^{(r)}$
9: $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(r)}$

In the third step of the ADMM, we project $(\boldsymbol{\theta}^{k+1} - \boldsymbol{v}^k)$ onto the constraint set $\Omega(\boldsymbol{\theta})$. Note that, this set is separable in $\boldsymbol{\theta}$, so the projection step can also be performed independently and in parallel for each variable $\boldsymbol{z}_t$, *i.e.*,

$$\boldsymbol{z}_t^{k+1} = \boldsymbol{\Pi}_{\Omega(\boldsymbol{\theta})}(\boldsymbol{\theta}_t^{k+1} + \boldsymbol{v}_t^k), \ \forall\, t \in \mathbb{N}_T. \tag{15}$$

The $\boldsymbol{z}_t$-update can also be seen as the problem of finding the intersection between two closed convex sets $\Omega_1(\boldsymbol{\theta}) = \{\boldsymbol{\theta}_t \succeq \mathbf{0}, \ \forall\ t \in \mathbb{N}_T\}$ and $\Omega_2(\boldsymbol{\theta}) = \{\|\boldsymbol{\theta}_t\|_1 \leq 1, \ \forall\ t \in \mathbb{N}_T\}$, which can be handled using Dykstra's alternating projections method [5,11] as follows

$$\boldsymbol{y}_t^{k+1} = \boldsymbol{\Pi}_{\Omega_1(\boldsymbol{\theta})}(\boldsymbol{\theta}_t^{k+1} + \boldsymbol{v}_t^k - \boldsymbol{\beta}_t^k) = \frac{1}{2}\left[\boldsymbol{\theta}_t^{k+1} + \boldsymbol{v}_t^k - \boldsymbol{\beta}_t^k\right]_+, \ \forall\, t \in \mathbb{N}_T \tag{16}$$

$$\boldsymbol{z}_t^{k+1} = \boldsymbol{\Pi}_{\Omega_2(\boldsymbol{\theta})}(\boldsymbol{y}_t^{k+1} + \boldsymbol{\beta}_t^k) = \mathbf{P}_M(\boldsymbol{y}_t^{k+1} + \boldsymbol{\beta}_t^k) + \frac{1}{M}\mathbf{1}_M, \ \forall\, t \in \mathbb{N}_T \tag{17}$$

$$\boldsymbol{\beta}_t^{k+1} = \boldsymbol{\beta}_t^k + \boldsymbol{y}_t^{k+1} - \boldsymbol{z}_t^{k+1}, \ \forall\, t \in \mathbb{N}_T \tag{18}$$

where $\mathbf{P}_M \triangleq \left(\mathbf{I}_M - \frac{\mathbf{1}_M \mathbf{1}_M^{'}}{M}\right)$ is the centering matrix. Furthermore, the $\boldsymbol{y}_t$- and $\boldsymbol{z}_t$ updates are the Euclidean projections onto $\Omega_1(\boldsymbol{\theta})$ and $\Omega_2(\boldsymbol{\theta})$ respectively with dual variables $\boldsymbol{\beta}_t \in \mathbb{R}^{M\times 1}$, $t = 1, \ldots, T$. Finally, we update the dual variables $\boldsymbol{u}_i$ and $\boldsymbol{v}$ using the equations given in (10) and (11).

Algorithm 2. Consensus ADMM algorithm to solve optimization Problem 4

Require: $\boldsymbol{q}_1^{(r)}, \ldots, \boldsymbol{q}_T^{(r)}, \rho$
Ensure: $\boldsymbol{\theta}_1^{(r)}, \ldots, \boldsymbol{\theta}_T^{(r)}$
Initialize: $\hat{\boldsymbol{\theta}}_1^{(0)}, \ldots, \hat{\boldsymbol{\theta}}_T^{(0)}, k = 0$
while not converged **do**
  for $i \in \mathbb{N}_N, t \in \mathbb{N}_T$ **do**
    $\boldsymbol{s}_i^{k+1} \leftarrow \mathcal{S}_{\lambda/\rho}(\tilde{\boldsymbol{\theta}}_i^k + \boldsymbol{u}_i^k)$
    $\hat{\boldsymbol{\theta}}_t^{k+1} \leftarrow \frac{1}{T-1}\left[\sum_{\mathcal{M}(i,j)=t}\left((\boldsymbol{s}_i)_j^{k+1} + (\boldsymbol{u}_i)_j^k\right) + \left(\boldsymbol{z}_t^k + \boldsymbol{v}_t^k\right) - (1/\rho)\boldsymbol{q}_t\right]$
    $\boldsymbol{y}_t^{k+1} \leftarrow \frac{1}{2}\left[\hat{\boldsymbol{\theta}}_t^{k+1} + \boldsymbol{v}_t^k - \boldsymbol{\beta}_t^k\right]_+$
    $\boldsymbol{z}_t^{k+1} \leftarrow \mathbf{P}_M(\boldsymbol{y}_t^{k+1} + \boldsymbol{\beta}_t^k) + \frac{1}{M}\mathbf{1}_M$
    $\boldsymbol{\beta}_t^{k+1} \leftarrow \boldsymbol{\beta}_t^k + \boldsymbol{y}_t^{k+1} - \boldsymbol{z}_t^{k+1}$
    $\boldsymbol{u}_i^{k+1} \leftarrow \boldsymbol{u}_i^k + \boldsymbol{s}_i^{k+1} - \tilde{\boldsymbol{\theta}}_i^{k+1}$
    $\boldsymbol{v}_t^{k+1} \leftarrow \boldsymbol{v}_t^k + \boldsymbol{z}_t^{k+1} - \hat{\boldsymbol{\theta}}_t^{k+1}$
  end for
end while
$\boldsymbol{\theta}^{(r)} \leftarrow \hat{\boldsymbol{\theta}}^{(k+1)}$

3.1 Convergence Analysis and Stopping Criteria

Convergence of Algorithm 2 can be derived based on two mild assumptions similar to the standard convergence theory of the ADMM method discussed in [6]; (i) the objective functions $h(\boldsymbol{s}) = \sum_{i=1}^{N} \|(\boldsymbol{s}_i)_j - (\boldsymbol{s}_i)_{j'}\|_2$ and $g(\boldsymbol{\theta}) = \sum_{t=1}^{T} \boldsymbol{\theta}_t^{'}\boldsymbol{q}_t$ are closed, proper and convex, which implies that the subproblems arising in the $\boldsymbol{s}$-update (7) and $\boldsymbol{\theta}$-update (8) are solvable, and (ii) the augmented Lagrangian (6) for $\rho = 0$ has a saddle point. Under these two assumptions, it can be shown that our ADMM-based algorithm satisfies the following

- Convergence of residuals : $\boldsymbol{s}_i{}^k - \tilde{\boldsymbol{\theta}}_i^k \rightarrow \mathbf{0}$, $\forall\ i \in \mathbb{N}_N$, and $\boldsymbol{z}^k - \boldsymbol{\theta}^k \rightarrow \mathbf{0}$ as $k \rightarrow \infty$.
- Convergence of dual variables: $\boldsymbol{u}_i^k \rightarrow \boldsymbol{u}_i^*, \forall i \in \mathbb{N}_N$, and $\boldsymbol{v}^k \rightarrow \boldsymbol{v}^*$ as $k \rightarrow \infty$, where $\boldsymbol{u}^*$ and $\boldsymbol{v}^*$ are the dual optimal points.
- Convergence of the objective : $h(\boldsymbol{s}^k) + g(\boldsymbol{z}^k) \rightarrow p^*$ as $k \rightarrow \infty$, which means the objective function (4) converges to its optimal value as the algorithm proceeds.

Also, the algorithm is terminated, when the primal and dual residuals satisfy the following stopping criteria

$$\|e_{p_1}^k\|_2 \leq \epsilon_1^{pri}, \qquad \|e_{p_2}^k\|_2 \leq \epsilon_2^{pri}, \qquad \|e_{p_3}^k\|_2 \leq \epsilon_3^{pri}$$

$$\|e_{d_1}^k\|_2 \leq \epsilon_1^{dual}, \quad \|e_{d_2}^k\|_2 \leq \epsilon_2^{dual}, \quad \|e_{d_3}^k\|_2 \leq \epsilon_3^{dual} \tag{19}$$

where the primal residuals of the k^{th} iteration are given as $e_{p_1}^k = \boldsymbol{s}^k - \boldsymbol{\theta}^k$, $e_{p_2}^k = \boldsymbol{z}^k - \boldsymbol{\theta}^k$ and $e_{p_3}^k = \boldsymbol{y}^k - \boldsymbol{z}^k$. Similarly $e_{d_1}^k = \rho(\boldsymbol{\theta}^{k+1} - \boldsymbol{\theta}^k)$, $e_{d_2}^k = \rho(\boldsymbol{z}^k - \boldsymbol{z}^{k+1})$ and $e_{d_3}^k = \rho(\boldsymbol{y}^k - \boldsymbol{y}^{k+1})$are dual residuals at iteration k. Also, the tolerances $\epsilon^{pri} > 0$, and $\epsilon^{dual} > 0$ can be chosen appropriately using the method described in Chapter 3 of [6].

3.2 Computational Complexity

Algorithm 1 needs to compute and cache TM kernel matrices; however, they are computed only once in $\mathcal{O}(TMn^2)$ time. Also, as long as the number of tasks T is not excessive, all the matrices can be computed and stored on a single machine, since (i) the number M of kernels, is typically chosen small (*e.g.*, we chose $M = 10$), and (ii) the number n of training samples per task is not usually large; if it were large, MTL would probably not be able to offer any advantages over training each task independently. For each iteration of Algorithm 1, T independent SVM problems are solved at a time cost of $\mathcal{O}(n^3)$ per task. Therefore, if Algorithm 2 converges in K iterations, the runtime complexity of Algorithm 1 becomes $\mathcal{O}(Tn^3 + KMT^2)$ per iteration. Note, though, that K is not usually more than a few tens of iterations [6].

On the other hand, if the number of tasks T is large, the nature of our problem allows our algorithm to be implemented in parallel. The $\boldsymbol{\alpha}$-update can be handled as T independent optimization problems, which can be easily distributed to T subsystems. Each subsystem N needs to compute once and cache M kernel matrices for each task. Then, for each iteration, one SVM problem is required to be solved by each subsystem, which takes $\mathcal{O}(n^3)$ time. Moreover, our ADMM-based algorithm updating the $\boldsymbol{\theta}$ parameters can also be implemented in parallel over $i \in \mathbb{N}_N$. Assuming that exchanging data and updates between subsystems consumes negligible time, the ADMM only requires $\mathcal{O}(KM)$ time. Therefore, taking advantage of a distributed implementation, the complexity of Algorithm 1 is only $\mathcal{O}(n^3 + KM)$ per iteration.

4 Generalization Bound Based on Rademacher Complexity

In this section, we provide a Rademacher complexity-based generalization bound for the Hypothesis Space (HS) considered in Problem 1, which can be identified with the help of the following Proposition [1].

[1] Note that Proposition 1 here utilizes the first part of Proposition 12 in [20] and does not require the strong duality assumption, which is necessary for the second part of Proposition 12 in [20].

Proposition 1. *(Proposition 12 in [20], part (a)) Let $\mathcal{C} \subseteq \mathcal{X}$ and let $f, g : \mathcal{C} \mapsto \mathbb{R}$ be two functions. For any $\nu > 0$, there must exist a $\eta > 0$, such that the optimal solution of (20) is also optimal in (21)*

$$\min_{x \in \mathcal{C}} f(x) + \nu g(x) \tag{20}$$

$$\min_{x \in \mathcal{C}, g(x) \leq \eta} f(x) \tag{21}$$

Using Proposition 1, one can show that Problem 1 is equivalent to the following problem

$$\begin{aligned} &\min_{\boldsymbol{w} \in \Omega^{'}(\boldsymbol{w})} \; C \sum_{t=1}^{T} \sum_{i=1}^{n} l\left(\boldsymbol{w}_t, \phi_t\left(x_t^i\right), y_t^i\right) \\ &\Omega^{'}(\boldsymbol{w}) \triangleq \{\boldsymbol{w} = (\boldsymbol{w}_1, \cdots, \boldsymbol{w}_T) : \boldsymbol{w}_t \in \mathcal{H}_{t,\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Omega^{'}(\boldsymbol{\theta}), \; \|\boldsymbol{w}_t\|^2 \leq R_t, \; t \in \mathbb{N}_T\} \end{aligned} \tag{22}$$

where

$$\Omega^{'}(\boldsymbol{\theta}) \triangleq \Omega(\boldsymbol{\theta}) \cap \left\{ \boldsymbol{\theta} = (\boldsymbol{\theta}_t, \cdots, \boldsymbol{\theta}_T) : \sum_{t=1}^{T-1} \sum_{s>t}^{T} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 \leq \gamma \right\}$$

The goal here is to choose the $\boldsymbol{w}$ and $\boldsymbol{\theta}$ from their relevant feasible sets, such that the objective function of (22) is minimized. Therefore, the relevant hypothesis space for Problem 22 becomes

$$\mathcal{F} \triangleq \left\{ x \mapsto [\langle \boldsymbol{w}_1, \boldsymbol{\phi}_1 \rangle, \ldots, \langle \boldsymbol{w}_T, \boldsymbol{\phi}_T \rangle]^{'} : \forall t \boldsymbol{w}_t \in \mathcal{H}_{t,\boldsymbol{\theta}}, \|\boldsymbol{w}_t\|^2 \leq R_t, \boldsymbol{\theta} \in \Omega^{'}(\boldsymbol{\theta}) \right\} \tag{23}$$

Note that finding the Empirical Rademacher Complexity (ERC) of $\mathcal{F}$ is complicated due to the non-smooth nature of the constraint $\sum_{t=1}^{T-1} \sum_{s>t}^{T} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2 \leq \gamma$. Instead, we will find the ERC of the HS $\mathcal{H}$ defined in (24); notice that $\mathcal{F} \subseteq \mathcal{H}$.

$$\mathcal{H} \triangleq \left\{ x \mapsto [\langle \boldsymbol{w}_1, \boldsymbol{\phi}_1 \rangle, \ldots, \langle \boldsymbol{w}_T, \boldsymbol{\phi}_T \rangle]^{'} : \forall t \boldsymbol{w}_t \in \mathcal{H}_{t,\boldsymbol{\theta}}, \|\boldsymbol{w}_t\|^2 \leq R_t, \boldsymbol{\theta} \in \Omega^{''}(\boldsymbol{\theta}) \right\} \tag{24}$$

where

$$\Omega^{''}(\boldsymbol{\theta}) \triangleq \Omega(\boldsymbol{\theta}) \cap \left\{ \boldsymbol{\theta} = (\boldsymbol{\theta}_t, \cdots, \boldsymbol{\theta}_T) : \sum_{t=1}^{T-1} \sum_{s>t}^{T} \|\boldsymbol{\theta}_t - \boldsymbol{\theta}_s\|_2^2 \leq \gamma^2 \right\} \tag{25}$$

Using the first part of Theorem (12) in [4], it can be shown that the ERC of $\mathcal{H}$ upper bounds the ERC of function class $\mathcal{F}$. Thus, the bound derived for $\mathcal{H}$ is also valid for $\mathcal{F}$. The following theorem provides the generalization bound for $\mathcal{H}$.

Theorem 1. *Let $\mathcal{H}$ defined in (24) be the multi-task HS for a class of functions $\boldsymbol{f} = (f_1, \ldots, f_T) : \mathcal{X} \mapsto \mathbb{R}^T$. Then for all $f \in \mathcal{H}$, for $\delta > 0$ and for fixed $\rho > 0$, with probability at least $1 - \delta$ it holds that*

$$R(\boldsymbol{f}) \leq \hat{R}_\rho(\boldsymbol{f}) + \frac{2}{\rho}\hat{\mathfrak{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log\frac{1}{\delta}}{2Tn}} \tag{26}$$

where

$$\hat{\mathfrak{R}}_S\left(\mathcal{H}\right) \leq \hat{\mathfrak{R}}_{ub}\left(\mathcal{H}\right) = \sqrt{\frac{\sqrt{3}\gamma RM}{nT}} \tag{27}$$

where $\hat{\mathfrak{R}}_S(\mathcal{H})$, the ERC of $\mathcal{H}$, is given as

$$\hat{\mathfrak{R}}_S(\mathcal{H}) = \frac{1}{nT}\mathrm{E}_\sigma\left\{\sup_{\boldsymbol{f}=(f_1,\ldots,f_T)\in\mathcal{F}}\sum_{t=1}^{T}\sum_{i=1}^{n}\sigma_t^i f_t(x_t^i)\,\middle|\,\left\{x_t^i\right\}_{t\in\mathbb{N}_T, i\in\mathbb{N}_n}\right\} \tag{28}$$

the ρ-empirical large margin error $\hat{R}_\rho(\boldsymbol{f})$, for the training sample $S = \left\{\left(x_t^i, y_t^i\right)\right\}_{i,t=1}^{n,T}$ is defined as

$$\hat{R}_\rho(\boldsymbol{f}) = \frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\min\left(1, [1 - y_t^i f_t(x_t^i)/\rho]_+\right)$$

Also, $R(f) = \Pr[yf(x) < 0]$ is the expected risk w.r.t. 0-1 loss, n is the number of training samples for each task, T is the number of tasks to be trained, and M is the number of kernel functions utilized for MKL.

The proof of this theorem is omitted due to space constraints. Based on Theorem 1, the second term in (26), the upper bound for ERC of $\mathcal{H}$, decreases as the number of tasks increases. Therefore, it is reasonable to expect that the generalization performance to improve, when the number T of tasks or the number n of training samples increase. Also, due to the formulation's group lasso (L_1/L_2-norm) regularizer on the pair-wise MKL weight differences, the ERC in (27) depends on M as $\mathcal{O}\sqrt{M}$. It is worth mentioning, that, while this could be improved to $\mathcal{O}\sqrt{\log M}$ as in [9], if one considers instead a L_p/L_q-norm regularizer, we won't pursue this avenue here. Let us finally note, that (26) allows one to construct data-dependent confidence intervals for the true, pooled (averaged over tasks) misclassification rate of the MTL problem under consideration.

5 Experiments

In this section, we demonstrate the merit of the proposed model via a series of comparative experiments. For reference, we consider two baseline methods

referred to as **STL** and **MTL**, which present the two extreme cases discussed in Sect. 2. We also compare our method with five state-of-the-art methods which, like ours, fall under the CMTL family of approaches. These methods are briefly described below.

- **STL:** single-task learning approach used as a baseline, according to which each task is individually trained via a traditional single-task MKL strategy.
- **MTL:** a typical MTL approach, for which all tasks share a common feature space. An SVM-based formulation with multiple kernel functions was utilized and the common MKL parameters for all tasks were learned during training.
- **CMTL** [17]**:** in this work, the tasks are grouped into disjoint clusters, such that the model parameters of the tasks belonging to the same group are close to each other.
- **Whom** [19]**:** clusters the task, into disjoint groups and assumes that tasks of the same group can jointly learn a shared feature representation.
- **FlexClus** [29]**:** a flexible clustering structure of tasks is assumed, which can vary from feature to feature.
- **CoClus** [25]**:** a co-clustering structure is assumed aiming to capture both the feature and task relationship between tasks.
- **MeTaG** [16]**:** a multi-level grouping structure is constructed by decomposing the matrix of tasks' parameters into a sum of components, each of which corresponds to one level and is regularized with a L_2-norm on the pairwise difference between parameters of all the tasks.

5.1 Experimental Settings

For all experiments, all kernel-based methods (including **STL**, **MTL** and our method) utilized 1 Linear, 1 Polynomial with degree 2, and 8 Gaussian kernels with spread parameters $\{2^0, \ldots, 2^7\}$ for MKL. All kernel functions were normalized as $k(\boldsymbol{x}, \boldsymbol{y}) \leftarrow k(\boldsymbol{x}, \boldsymbol{y}) / \sqrt{k(\boldsymbol{x}, \boldsymbol{x}) k(\boldsymbol{y}, \boldsymbol{y})}$. Moreover, for **CMTL**, **Whom** and **CoClus** methods, which require the number of task clusters to be pre-specified, cross-validation over the set $\{1, \ldots, T/2\}$ was used to select the optimal number of clusters. Also, the regularization parameters of all methods were chosen via cross-validation over the set $\{2^{-10}, \ldots, 2^{10}\}$.

5.2 Experimental Results

We assess the performance of our proposed method compared to the other methods on 7 widely-used data sets including 3 real-world data sets: Wall-Following Robot Navigation (*Robot*), Statlog Vehicle Silhouettes (*Vehicle*) and Statlog Image Segmentation (*Image*) from the UCI repository [14], 2 handwritten digit data sets, namely MNIST Handwritten Digit (*MNIST*) and Pen-Based Recognition of Handwritten Digits (*Pen*), as well as *Letter* and *Landmine*.

The data sets from the UCI repository correspond to three multi-class problems. In the *Robot* data set, each sample is labeled as: "Move-Forward, "SlightRight-Turn", "Sharp-Right-Turn" and "Slight-Left-Turn". These classes

are designed to navigate a robot through a room following the wall in a clockwise direction. The *Vehicle* data set describes four different types of vehicles as "4 Opel", "SAAB", "Bus" and "Van". On the other hand, the instances of the *Image* data set were drawn randomly from a database of 7 outdoor images which are labeled as "Sky", "Foliage", "Cement", "Window", "Path" and "Grass".

Also, two multi-class handwritten digit data sets, namely *MNIST* and *Pen*, consist of samples of handwritten digits from 0 to 9. Each example is labeled as one of ten classes. A one-versus-one strategy was adopted to cast all multi-class learning problems into MTL problems, and the average classification accuracy across tasks was calculated for each data set. Moreover, an equal number of samples from each class was chosen for training for all five multi-class problems.

We also compare our method on two widely-used multi-task data sets, namely the *Letter* and *Landmine* data sets. The former one is a collection of handwritten words collected by Rob Kassel of MIT's spoken Language System Group, and involves eight tasks: 'C' vs. 'E', 'G' vs. 'Y', 'M' vs. 'N', 'A' vs. 'G', 'I' vs. 'J', 'A' vs. 'O', 'F' vs. 'T' and 'H' vs. 'N'. Each letter is represented by a 8 by 16 pixel image, which forms a 128 dimensional feature vector per sample. We randomly chose 200 samples for each letter. An exception is letter J, for which only 189 samples were available. The *Landmine* data set consists of 29 binary classification tasks collected from various landmine fields. The objective is to recognize whether there is a landmine or not based on a region's characteristics, which are described by four moment-based features, three correlation-based features, one energy ratio feature, and one spatial variance feature.

In all our experiments, for all methods, we considered training set sizes of 10%, 20% and 50% of the original data set to investigate the influence of the data set size on generalization performance. An exception was the *Landmine* data set, for which we used 20% and 50% of the data set for training purposes due to its small size. The rest of data were split into equal sizes for validation and testing.

In Table 1, we report the average classification accuracy over 20 runs of randomly sampled training sets for each experiment. Note that we utilized the method proposed in [10] for our statistical analysis. More specifically, Friedman's and Holm's post-hoc tests at significance level $\alpha = 0.05$ were employed to compare our proposed method with the other methods.

As shown in Table 1, for each data set, Friedman's test ranks the best performing model as first, the second best as second and so on. The superscript next to each value in Table 1 indicates the rank of the corresponding model on the relevant data set, while the superscript next to each model reflects its average rank over all data sets for the corresponding training set size. Note that methods depicted in boldface are deemed statistically similar to our model, since their corresponding p-values are not smaller than the adjusted α values obtained by Holm's post-hoc test. Overall, it can be observed that our method dominates three, six and five out of seven methods, when trained with 10%, 20% and 50% training set sizes respectively.

Table 1. Experimental comparison between our method and seven benchmark methods

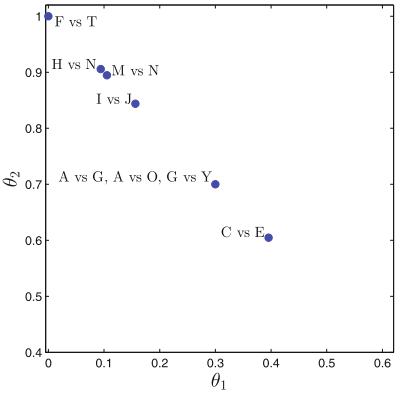
10%	STL$^{(7)}$	MTL$^{(5.42)}$	CMTL$^{(6.33)}$	Whom$^{(3.25)}$	FlexClus$^{(4.33)}$	Coclus$^{(4)}$	MetaG$^{(5)}$	Our Method$^{\mathbf{(1.67)}}$
Robot	84.51$^{(7)}$	84.82$^{(6)}$	84.15$^{(8)}$	**88.90**$^{(1)}$	88.34$^{(4)}$	87.83$^{(5)}$	88.77$^{(2)}$	88.67$^{(3)}$
Vehicle	79.73$^{(8)}$	80.38$^{(6)}$	80.23$^{(7)}$	83.14$^{(4)}$	82.45$^{(5)}$	**86.79**$^{(1)}$	83.53$^{(3)}$	84.51$^{(2)}$
Image	97.08$^{(7)}$	97.43$^{(3)}$	97.09$^{(6)}$	97.27$^{(4)}$	98.05$^{(2)}$	97.24$^{(5)}$	97.05$^{(8)}$	**98.19**$^{(1)}$
Pen	98.16$^{(7)}$	98.28$^{(5.5)}$	95.78$^{(8)}$	98.28$^{(5.5)}$	98.67$^{(3)}$	**99.26**$^{(1)}$	98.57$^{(4)}$	99.12$^{(2)}$
MNIST	94.09$^{(7)}$	94.87$^{(4)}$	94.49$^{(6)}$	95.56$^{(3)}$	94.59$^{(5)}$	93.09$^{(8)}$	96.13$^{(2)}$	**96.70**$^{(1)}$
Letter	84.12$^{(6)}$	83.12$^{(8)}$	85.62$^{(3)}$	86.82$^{(2)}$	83.72$^{(7)}$	85.46$^{(4)}$	85.41$^{(5)}$	**87.41**$^{(1)}$
20%	STL$^{(6)}$	MTL$^{(4.43)}$	CMTL$^{(6.14)}$	Whom$^{(3.29)}$	FlexClus$^{(5.57)}$	Coclus$^{(4.57)}$	MetaG$^{(4.71)}$	Our Method$^{\mathbf{(1.14)}}$
Robot	87.67$^{(7)}$	88.23$^{(6)}$	85.08$^{(8)}$	**90.76**$^{(1)}$	90.15$^{(3)}$	88.43$^{(5)}$	89.12$^{(4)}$	90.34$^{(2)}$
Vehicle	85.88$^{(4)}$	86.16$^{(3)}$	82.29$^{(8)}$	85.67$^{(6)}$	85.29$^{(7)}$	87.15$^{(2)}$	85.78$^{(5)}$	**87.76**$^{(1)}$
Image	97.41$^{(6)}$	98.02$^{(3)}$	97.32$^{(7)}$	98.46$^{(2)}$	97.44$^{(5)}$	97.50$^{(4)}$	97.29$^{(8)}$	**98.54**$^{(1)}$
Pen	98.57$^{(7)}$	99.01$^{(6)}$	96.06$^{(8)}$	99.14$^{(3)}$	99.13$^{(4)}$	99.30$^{(2)}$	99.02$^{(4)}$	**99.63**$^{(1)}$
MNIST	96.13$^{(6)}$	96.71$^{(4)}$	96.56$^{(5)}$	96.76$^{(3)}$	95.04$^{(7)}$	94.09$^{(8)}$	96.84$^{(2)}$	**97.86**$^{(1)}$
Landmine	58.76$^{(8)}$	61.89$^{(7)}$	65.28$^{(2)}$	62.53$^{(5)}$	62.46$^{(6)}$	63.52$^{(3)}$	62.59$^{(4)}$	**65.82**$^{(1)}$
Letter	88.75$^{(4)}$	89.98$^{(2)}$	88.24$^{(5)}$	88.88$^{(3)}$	83.79$^{(7)}$	82.26$^{(8)}$	87.99$^{(6)}$	**90.72**$^{(1)}$
50%	STL$^{(5.64)}$	MTL$^{(3.85)}$	CMTL$^{(6.29)}$	Whom$^{(3.29)}$	FlexClus$^{(6.21)}$	Coclus$^{(5.29)}$	MetaG$^{(4.42)}$	Our Method$^{\mathbf{(1)}}$
Robot	91.26$^{(5.5)}$	91.49$^{(3)}$	86.26$^{(8)}$	91.70$^{(2)}$	91.26$^{(5.5)}$	89.04$^{(7)}$	91.27$^{(4)}$	**92.41**$^{(1)}$
Vehicle	88.33$^{(3)}$	88.71$^{(2)}$	83.91$^{(8)}$	87.3$^{(5)}$	86.72$^{(7)}$	87.55$^{(4)}$	86.81$^{(6)}$	**89.83**$^{(1)}$
Image	98.40$^{(6)}$	98.43$^{(5)}$	97.56$^{(8)}$	98.58$^{(2)}$	98.04$^{(7)}$	98.52$^{(3)}$	98.49$^{(4)}$	**99.07**$^{(1)}$
Pen	98.77$^{(7)}$	99.23$^{(5)}$	96.17$^{(8)}$	99.32$^{(4)}$	99.33$^{(3)}$	99.34$^{(2)}$	99.21$^{(6)}$	**99.77**$^{(1)}$
MNIST	97.20$^{(6)}$	97.37$^{(4)}$	97.31$^{(5)}$	97.78$^{(3)}$	96.60$^{(7)}$	95.87$^{(8)}$	98.46$^{(2)}$	**98.64**$^{(1)}$
Landmine	63.76$^{(8)}$	64.98$^{(6)}$	66.76$^{(2)}$	65.57$^{(4)}$	64.87$^{(7)}$	65.15$^{(5)}$	66.24$^{(3)}$	**67.15**$^{(1)}$
Letter	91.18$^{(4)}$	91.62$^{(2)}$	90.97$^{(5)}$	91.25$^{(3)}$	86.47$^{(7)}$	86.27$^{(8)}$	90.66$^{(6)}$	**92.49**$^{(1)}$

Table 2. Comparison of our method against the other methods with the Holm test

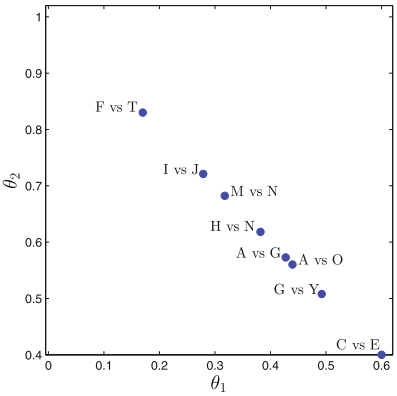
10%	STL	MTL	CMTL	**Whom**	**FlexClus**	**Coclus**	**MeTaG**
Test statistic	3.93	2.13	3.49	1.25	2.40	2.62	2.29
p value	0.0005	0.0138	0.0022	**0.2869**	**0.0777**	**0.1214**	**0.1214**
Adjusted α	0.0071	0.0083	0.0100	0.0125	0.01667	0.0250	0.0500
20%	STL	MTL	CMTL	**Whom**	FlexClus	Coclus	MeTaG
Test statistic	3.71	2.51	3.82	1.64	3.38	2.62	2.73
p value	0.00021	0.0121	0.0001	**0.1017**	0.0007	0.0088	0.0064
Adjusted α	0.0083	0.0250	0.0071	0.0500	0.0100	0.01667	0.0125
50%	STL	**MTL**	CMTL	**Whom**	FlexClus	Coclus	MeTaG
Test statistic	3.55	2.18	4.04	1.75	3.98	3.27	2.61
p value	0.0004	**0.0291**	0.0001	**0.0809**	0.0001	0.0011	0.0089
Adjusted α	0.0100	0.0250	0.0071	0.0500	0.0083	0.0125	0.01667

Also, in Figure 1, we provide better insight of how the grouping of task feature spaces might be determined in our framework. For the purpose of visualization, we applied two Gaussian kernel functions with spread parameters 2 and 2^8 and used the *Letter* multi-task data set.

In this figure, the x and y axes represent the weights of these two kernel functions for each task. From Figure 1(a), when a small training size (10%) is chosen, it can be seen that our framework yields a cluster of 3 tasks, namely {"A" vs "G", "A" vs "O", "G" vs "Y"} that share a common feature space to benefit



(a) Traning set size 10%



(b) Traning set size 50%

Fig. 1. Feature space parameters for *Letter* multi-task data set

from each other's data. However, as the number n of training samples per task increases, every task is allowed to employ its own feature space to guarantee good performance. This is shown in Figure 1 (b), which displays the results obtained for a 50% training set size. Note, that the displayed MKL weights lie on the $\theta_1 + \theta_2 = 1$ line due to the framework's L_1 MKL weight constraint.

6 Conclusions

In this work, we proposed a novel MT-MKL framework for SVM-based binary classification, where a flexible group structure is determined between each pair of tasks. In this framework, tasks are allowed to have a common, similar, or distinct feature spaces. Recently, some MTL frameworks have been proposed, which also consider clustering strategies to capture task relatedness. However, our method

is capable of modeling a more general type of task relationship, where tasks may be implicitly grouped according to a notion of feature space similarity. Also, our proposed optimization algorithm allows for a distributed implementation, which can be significantly advantageous for MTL settings involving large number of tasks. The performance advantages reported on 7 multi-task SVM-based classification problems largely seem to justify our arguments in favor of our framework.

Acknowledgments. N. Yousefi acknowledges support from National Science Foundation (NSF) grants No. 0806931 and No. 1161228. Moreover, M. Georgiopoulos acknowledges partial support from NSF grants No. 0806931, No. 0963146, No. 1200566, No. 1161228, and No. 1356233. Finally, G. C. Anagnostopoulos acknowledges partial support from NSF grant No. 1263011. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Argyriou, A., Clémençon, S., Zhang, R.: Learning the graph of relations among multiple tasks. In: ICML 2014 workshop on New Learning Frameworks and Models for Big Data (2013)
2. Argyriou, A., Evgeniou, T., Pontil, M.: Convex multi-task feature learning. Machine Learning **73**(3), 243–272 (2008)
3. Bakker, B., Heskes, T.: Task clustering and gating for bayesian multitask learning. The Journal of Machine Learning Research **4**, 83–99 (2003)
4. Bartlett, P.L., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. The Journal of Machine Learning Research **3**, 463–482 (2003)
5. Bauschke, H., Borwein, J.M.: Dykstra's alternating projection algorithm for two sets. Journal of Approximation Theory **79**(3), 418–443 (1994)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends in Machine Learning **3**(1), 1–122 (2011)
7. Caruana, R.: Multitask learning. Machine Learning **28**(1), 41–75 (1997)
8. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011). http://www.csie.ntu.edu.tw/~cjlin/libsvm
9. Cortes, C., Mohri, M., Rostamizadeh, A.: Generalization bounds for learning kernels. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 247–254 (2010)
10. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
11. Dykstra, R.L.: An algorithm for restricted least squares regression. Journal of the American Statistical Association **78**(384), 837–842 (1983)
12. Evgeniou, A., Pontil, M.: Multi-task feature learning. Advances in Neural Information Processing Systems **19**, 41 (2007)
13. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning multiple tasks with kernel methods. Journal of Machine Learning Research, 615–637 (2005)
14. Frank, A., Asuncion, A.: UCI machine learning repository (2010). http://archive.ics.uci.edu/ml

15. Gu, Q., Li, Z., Han, J.: Joint feature selection and subspace learning. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, p. 1294 (2011)
16. Han, L., Zhang, Y.: Learning multi-level task groups in multi-task learning. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI) (2015)
17. Jacob, L., Vert, J.p., Bach, F.R.: Clustered multi-task learning: a convex formulation. In: Advances in Neural Information Processing Systems, pp. 745–752 (2009)
18. Jalali, A., Sanghavi, S., Ruan, C., Ravikumar, P.K.: A dirty model for multi-task learning. In: Advances in Neural Information Processing Systems, pp. 964–972 (2010)
19. Kang, Z., Grauman, K., Sha, F.: Learning with whom to share in multi-task feature learning. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 521–528 (2011)
20. Kloft, M., Brefeld, U., Sonnenburg, S., Zien, A.: Lp-norm multiple kernel learning. The Journal of Machine Learning Research **12**, 953–997 (2011)
21. Lanckriet, G.R., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. The Journal of Machine Learning Research **5**, 27–72 (2004)
22. Li, C., Georgiopoulos, M., Anagnostopoulos, G.C.: Conic multi-task classification. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (eds.) ECML PKDD 2014, Part II. LNCS, vol. 8725, pp. 193–208. Springer, Heidelberg (2014)
23. Li, C., Georgiopoulos, M., Anagnostopoulos, G.C.: Pareto-path multitask multiple kernel learning. IEEE Transactions on Neural Networks and Learning Systems **26**(1), 51–61 (2015)
24. Tang, L., Chen, J., Ye, J.: On multiple kernel learning with multiple labels. In: IJCAI, pp. 1255–1260 (2009)
25. Xu, L., Huang, A., Chen, J., Chen, E.: Exploiting task-feature co-clusters in multi-task learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015) (2015)
26. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task learning for classification with dirichlet process priors. The Journal of Machine Learning Research **8**, 35–63 (2007)
27. Zhang, Y., Yeung, D.Y.: A convex formulation for learning task relationships in multi-task learning. arXiv preprint arXiv:1203.3536 (2012)
28. Zhang, Y., Yeung, D.Y.: A regularization approach to learning task relationships in multitask learning. ACM Transactions on Knowledge Discovery from Data (TKDD) **8**(3), 12 (2014)
29. Zhong, W., Kwok, J.: Convex multitask learning with flexible task clusters. arXiv preprint arXiv:1206.4601 (2012)
30. Zhou, J., Chen, J., Ye, J.: Clustered multi-task learning via alternating structure optimization. In: Advances in Neural Information Processing Systems, pp. 702–710 (2011)