Grouped Multi-Task Learning with Hidden Tasks Enhancement

Jiachun Jin[1], **Jiankun Wang**[1], **Lu Sun**[1], **Jie Zheng**[1] **and Mineichi Kudo**[2]

[1]ShanghaiTech University, {jinjch, wangjk, sunlu1, zhengjie}@shanghaitech.edu.cn
[2]Hokkaido University, mine@ist.hokudai.ac.jp

Abstract. In multi-task learning (MTL), multiple prediction tasks are learned jointly, such that generalization performance is improved by transferring information across the tasks. However, not all tasks are related, and training unrelated tasks together can worsen the prediction performance because of the phenomenon of negative transfer. To overcome this problem, we propose a novel MTL method that can robustly group correlated tasks into clusters and allow useful information to be transferred only within clusters. The proposed method is based on the assumption that the task clusters lie in the low-rank subspaces of the parameter space, and the number of them and their dimensions are both unknown. By applying subspace clustering to task parameters, parameter learning and task grouping can be done in a unified framework. To relieve the error induced by the basic linear learner and robustify the model, the effect of hidden tasks is exploited. Moreover, the framework is extended to a multi-layer architecture so as to progressively extract hierarchical subspace structures of tasks, which helps to further improve generalization. The optimization algorithm is proposed, and its effectiveness is validated by experimental results on both synthetic and real-world datasets.

1 Introduction

Multi-task learning (MTL) aims to learn multiple related tasks together to achieve better generalization performance than independent learning in each task [7, 38, 4]. Different assumptions about task relatedness result in different ways of information sharing across tasks. Some assume that all task parameter vectors share a common set of active features or a common prior [28, 39], while some assume that they lie in a shared low-rank subspace [1, 8]. However, due to the phenomenon of negative transfer [31, 37], the generalization performance will degrade if information is transferred among unrelated tasks. Thus the above assumptions all face the problem of how to decide which tasks should transfer information to each other.

To resolve this issue, grouped MTL methods assume that different tasks form several clusters, each of which consists of similar tasks, and information is restricted to transfer only within each cluster. Some methods assume that similar tasks should have parameters that lie close to each other in terms of Euclidean distance [15, 41, 14]. However, these methods all neglect the possible negative correlation among tasks. To address this problem, another line of methods assumes that each task cluster exists in a low-rank subspace of the parameter space. Under this assumption, some methods do task grouping by solving a mixed integer programming problem [18]. Some methods further assume that task parameters share the same set of latent basis and tasks in the same cluster should have similar sparse representation with that basis [20, 16]. Nevertheless, in these methods, we need to know the number of task clusters or the dimension of the latent basis in advance, which is usually unavailable in reality.

Recently, a more appealing approach was proposed to directly learn the number of task clusters, their dimensions and the cluster structure from data. Instead of learning a set of latent basis, all the task parameters are treated as the basis and can be reconstructed by a linear combination of other tasks' parameters [21]. It is worth noting that, if the task parameters are treated as fixed data points, then finding out their cluster structure is actually the problem of subspace clustering (SC) [32]. However, the difference is the clustering objects in grouped MTL are task parameters, which are learned from data, probably resulting in a large deviation from the ground-truth values. So directly using the learned task parameters as a dictionary to represent themselves as SC does can potentially amplify such an error and hurt the MTL model's generalization performance.

Inspired by Latent Low-Rank Representation (LatLRR) [23, 40], which achieves robust subspace clustering by utilizing the effect of hidden data, we propose the Hidden Tasks Enhanced Multi-Task Learning (**HTEMTL**) method [1]. The key idea is to perform subspace clustering on task parameters by exploiting the effect of hidden tasks, so as to robustify the procedure of task grouping and consequently improve the MTL model's generalization performance. In addition, to explore deep latent group structures, we propose the **p**rogressive **HTEMTL** (**pHTEMTL**) by a multi-layer architecture that cascades multiple hidden task enhanced self-expressive layers. For empirical studies, we compare the proposed methods with state-of-the-art MTL methods on both synthetic and real-world datasets. The experimental results validate the effectiveness of the proposed method. The contributions of this paper can be summarized as follows:

1. We propose a novel grouped MTL method, named **HTEMTL**, that incorporates subspace clustering with MTL, and meanwhile utilizes the effect of hidden tasks, leading to improved generalization ability.
2. We extend **HTEMTL** to a multi-layer model to progressively recover hierarchical group structure, which further robustifies the model by deep hidden tasks enhancement.
3. We develop optimization algorithms for both **HTEMTL** and **pHTEMTL**, and empirically validate the effectiveness on synthetic and real-world datasets.

[1] The code is provided at: https://github.com/jiachunjin/HTEMTL

2 Related Work

The existing MTL methods can be separated into two main categories: the ungrouped ones and grouped ones. The ungrouped MTL methods directly assume that all the tasks are related [1, 2]. To deploy these methods on real-world applications, domain knowledge is required to make sure that all the input tasks are related, such that negative transfer [31, 25, 37] can be avoided. For the grouped MTL methods, one line of methods uses the Euclidean distance between task parameters to measure the task similarity. In [15], task parameters are assumed to stay close to each other within a task cluster, and parameters in different task clusters are far away from each other. In [43], one representative task is identified for each task cluster, and all the other parameters in that cluster are assumed to stay close to it. However, they fail to capture the negative correlation between tasks.

To remedy this problem, another line of the grouped MTL methods takes the assumption that the task parameters in the same cluster lie in a shared low-rank subspace. There are three different ways to achieve this. The first way is to do task grouping by solving a mixed integer programming problem and impose different regularizations on each cluster of task parameters. For instance, the nuclear norm is used in [18] to promote low-rank structure, and the $l_{2,1}$-norm is used in [19] to induce row sparsity of the task parameters in each cluster.

The second way is to decompose the task parameter matrix into the product of latent basis and latent assignment. In [20], the subspaces of task clusters are assumed to be correlated, and sparsity pattern of the latent assignment matrix is promoted to get a flexible task grouping structure. In [16], the row-sparsity of the latent basis is promoted, and the k-support norm [3] is regularized on the latent assignment, in order to simultaneously perform variable selection and learn the group structure. In [35, 34], the generalized k-block-diagonal structure of the assignment matrix is pursued to reduce inter-group information transfer. A common issue of the methods is how to manually determine the dimension of the latent basis.

The third way utilizes the self-expressiveness property of the task parameters. Instead of learning a new latent basis, all the observed tasks are used as the dictionary to represent themselves. The task parameter matrix is thus decomposed into the product of itself and a task correlation matrix. In this way, the number of clusters and their dimensions can be directly learned from data. The learned task correlation matrix is always block-diagonal up to some column and row permutations, and reveals the task cluster structure. In [21, 30], all task parameters are reconstructed as non-negative sparse linear combinations of other task parameters to achieve asymmetric transfer. In [26], a very similar assumption is taken and the trace Lasso [12] regularizer is used to adaptively group tasks. In [36], a unified framework is proposed to identify outlier tasks, and construct the rest task parameters as linear combinations of some representative task parameters. However, in these methods, the learned task parameters may have a large deviation from the ground-truth values and directly using the learned parameters as a dictionary can amplify such a deviation, and consequently hurt the MTL model's generalization performance. To relieve this issue, our proposed method can robustly group correlated tasks into clusters by exploiting the effect of hidden tasks, and allow useful information to be transferred only within clusters.

3 Preliminary

3.1 Notations

Given T tasks, all the tasks share the same feature space with dimension d. Each task consists of a set of training data $\mathcal{D}_t = \{\mathbf{X}_t, \mathbf{y}_t\}$, where $\mathbf{X}_t \in \mathbb{R}^{d\times N_t}$ and $\mathbf{y}_t \in \mathbb{R}^{N_t}$, with N_t being the number of training samples of the tth task. Following the common multi-task learning paradigm, the linear model $\mathbf{y}_t = \mathbf{X}_t^\top \mathbf{w}_{:,t}$ is adopted, where $\mathbf{w}_{:,t}$ is the tth column of $\mathbf{W} \in \mathbb{R}^{d\times T}$, which denotes the task parameter vector for the tth task, $\forall t \in \{1, 2, ..., T\}$. We use $\mathcal{L}(\cdot,\cdot)$ to denote the total training loss, i.e., $\mathcal{L}(\mathcal{D}, \mathbf{W}) = \sum_t l(\mathcal{D}_t, \mathbf{w}_{:,t})$, with l being the loss function. Specifically, we use the squared loss for regression tasks and the logistic loss for classification tasks. Given two arbitrary matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{p\times q}$, $[\mathbf{A}, \mathbf{B}] \in \mathbb{R}^{p\times 2q}$ denotes column concatenation and $[\mathbf{A}; \mathbf{B}] \in \mathbb{R}^{2p\times q}$ denotes row concatenation.

3.2 Subspace Clustering

Given a set of data points sampled from a union of subspaces, subspace clustering (SC) aims to partition the data points into their underlying subspaces [32]. Recent SC algorithms are developed mainly based on the self-expressiveness model. In this model, a point is expressed as a linear combination of the other data points from the same subspace, i.e., $\mathbf{x}_i = \mathbf{X}\mathbf{c}_i$, where $\mathbf{X}$ is the data matrix and the ith column $\mathbf{c}_i$ of $\mathbf{C}$ corresponds to the representation of the ith data point $\mathbf{x}_i$ [11]. When some proper regularizer is imposed on $\mathbf{C}$, it holds a block-diagonal structure, in which each block indicates a data subspace. For instance, Low-Rank Representation (LRR) [22] solves the following optimization problem to do subspace clustering:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_*, \quad \text{s.t.} \quad \mathbf{X} = \mathbf{X}\mathbf{C}, \tag{1}$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix. We write the skinny SVD of the data matrix $\mathbf{X}$ as $\mathbf{U}_0\mathbf{\Sigma}_0\mathbf{V}_0^\top$. If the data matrix $\mathbf{X}$ is noise-free, i.e., all the data points definitely come from the underlying union of subspaces, the unique optimal solution to (1) will be $\mathbf{C}^* = \mathbf{V}_0\mathbf{V}_0^\top$. It is known as the *shape interaction matrix* [17], which is quite sensitive to the noise in $\mathbf{X}$. When the subspaces are independent, $\mathbf{C}^*$ forms a block-diagonal matrix and each block indicates a cluster [9].

4 The Proposed Method

4.1 Task-Level Subspace Clustering

The fundamental assumption of our model is that all the task parameters come from a union of low-rank subspaces, and the parameters of related tasks lie in the same subspace. In the grouped MTL scenarios, if the task parameters are taken as fixed points, we can simply do task-level subspace clustering by replacing the data matrix $\mathbf{X}$ with the task parameter matrix $\mathbf{W}$ in (1). In order to jointly carry out data fitting and task-level subspace clustering, we propose the naive version of **HTEMTL**:

$$\min_{\mathbf{W},\mathbf{C}} \mathcal{L}(\mathcal{D}, \mathbf{W}) + \frac{\lambda}{2}\|\mathbf{W} - \mathbf{W}\mathbf{C}\|_F^2 + \gamma\|\mathbf{C}\|_* + \frac{\beta}{2}\|\mathbf{W}\|_F^2. \tag{2}$$

Here λ, γ and β are non-negative hyper-parameters, that need to be determined via validation. The last regularization term in (2) controls the model complexity. For simplicity, we omit the intercept in the linear model by adding an extra dimension with value one to the end of all the data vectors.

4.2 Hidden Tasks Enhanced Multi-Task Learning

In this subsection, we exploit the effect of hidden tasks in the MTL scenarios, so as to robustify the performance of the MTL model.

Since the task parameters learned from data are not reliable, some learned parameters may be far away from the ground-truth values. Thus only using the learned parameters as the dictionary to represent themselves, as the model does in (2), might potentially amplify the error. To relieve this problem, we can extend the original dictionary by concatenating $\mathbf{W}$ with the hidden task parameters $\mathbf{H}$ from some unobserved tasks, and assume that each column of $\mathbf{H}$ is a hidden task sampled from the same union of subspaces as $\mathbf{W}$, which is unobserved in the dataset but crucial for representing the underlying task subspaces. In this way, the model in (2) becomes:

$$\min_{\mathbf{W},\mathbf{C}} \mathcal{L}(\mathcal{D},\mathbf{W}) + \frac{\lambda}{2}\|\mathbf{W}-[\mathbf{W},\mathbf{H}]\mathbf{C}\|_F^2 + \gamma\|\mathbf{C}\|_* + \frac{\beta}{2}\|\mathbf{W}\|_F^2. \quad (3)$$

Using $\mathbf{H}$ in (3) enables the ability of representing the underlying subspaces during data fitting, thereby enhancing task clustering. However, it is not solvable since we never know what $\mathbf{H}$ exactly is. Inspired by LatLRR [23], we transform the formulation and make it solvable. Suppose that both $\mathbf{W}$ and $\mathbf{H}$ are known and fixed, the task-level subspace clustering problem can be formulated as:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_*, \quad \text{s.t.} \quad \mathbf{W} = [\mathbf{W},\mathbf{H}]\mathbf{C}. \quad (4)$$

Let the SVD of $[\mathbf{W},\mathbf{H}]$ be $\mathbf{U\Sigma V}^\top = \mathbf{U\Sigma}[\mathbf{V}_W;\mathbf{V}_H]^\top$, where $\mathbf{V}$ is partitioned into a row concatenation of two submatrices, such that $\mathbf{W} = \mathbf{U\Sigma V}_W^\top$ and $\mathbf{H} = \mathbf{U\Sigma V}_H^\top$. By left multiplying $\mathbf{\Sigma}^{-1}\mathbf{U}^\top$ to both sides of the constraint of (4), the problem is equivalent to:

$$\min_{\mathbf{C}} \|\mathbf{C}\|_*, \quad \text{s.t.} \quad \mathbf{V}_W^\top = \mathbf{V}^\top\mathbf{C}. \quad (5)$$

Since both $\mathbf{V}_W$ and $\mathbf{V}$ are known, it has a unique optimal solution: $\mathbf{C}^* = \mathbf{VV}_W^\top = [\mathbf{V}_W;\mathbf{V}_H]\mathbf{V}_W^\top$ [22]. By re-plugging $\mathbf{C}^*$ into the constraint of (4), it becomes:

$$\begin{aligned}\mathbf{W} &= [\mathbf{W},\mathbf{H}][\mathbf{V}_W\mathbf{V}_W^\top;\mathbf{V}_H\mathbf{V}_W^\top]\\ &= \mathbf{WV}_W\mathbf{V}_W^\top + \mathbf{HV}_H\mathbf{V}_W^\top\\ &= \mathbf{WV}_W\mathbf{V}_W^\top + \mathbf{U\Sigma V}_H^\top\mathbf{V}_H\mathbf{V}_W^\top\\ &= \mathbf{WV}_W\mathbf{V}_W^\top + \mathbf{U\Sigma V}_H^\top\mathbf{V}_H\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{W}. \quad (6)\end{aligned}$$

Given $\mathbf{Z} = \mathbf{V}_W\mathbf{V}_W^\top$ and $\mathbf{L} = \mathbf{U\Sigma V}_H^\top\mathbf{V}_H\mathbf{\Sigma}^{-1}\mathbf{U}^\top$, task parameters $\mathbf{W}$ can always be represented as the following

$$\mathbf{W} = \mathbf{WZ} + \mathbf{LW}. \quad (7)$$

Here $\mathbf{Z}$ works as the task correlation matrix, which should hold a block-diagonal structure and reveal the task clusters like $\mathbf{C}$ in (2). And $\mathbf{L}$ is the feature correlation matrix, that enables $\mathbf{W}$ to be reconstructed from its row space.

In reality, the hidden task matrix $\mathbf{H}$ is unreachable. Therefore, instead of exactly recovering $\mathbf{Z}$ and $\mathbf{L}$ from data, we approximately recover the effect of hidden tasks by solving

$$\min_{\mathbf{Z},\mathbf{L}} \text{rank}(\mathbf{Z}) + \text{rank}(\mathbf{L}), \quad \text{s.t.} \quad \mathbf{W} = \mathbf{WZ} + \mathbf{LW}, \quad (8)$$

where $\text{rank}(\cdot)$ denotes the rank of a matrix. From our basic assumption that all the task parameters come from a union of low-rank subspaces, the task parameter matrix $\mathbf{W}$ should also hold a low-rank structure. Since $\text{rank}(\mathbf{W}) \leq \min\{\text{rank}(\mathbf{W}), \text{rank}(\mathbf{Z})\} + \min\{\text{rank}(\mathbf{W}), \text{rank}(\mathbf{L})\}$, minimizing both $\text{rank}(\mathbf{Z})$ and $\text{rank}(\mathbf{L})$ can meet the requirement.

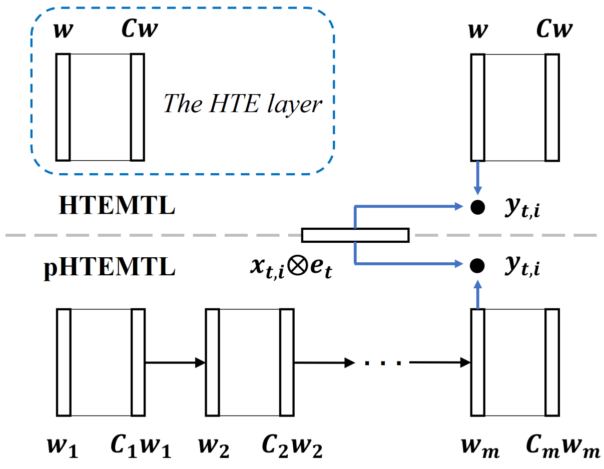


Figure 1: The frameworks of **HTEMTL** and **pHTEMTL**. **HTEMTL** builds a shallow model for task parameters $\mathbf{w} = \text{vec}(\mathbf{W})$ and regularizes them with the HTE layer, which is a linear fully-connected self-expressive layer that reconstructs $\mathbf{w}$ by $\mathbf{Cw}$ with weights $\mathbf{C} = \mathbf{Z}\otimes\mathbf{I}_d + \mathbf{I}_T\otimes\mathbf{L}$. In contrast, **pHTEMTL** extends **HTEMTL** to a deep model for $\mathbf{w}$, in order to progressively extract hierarchical structures of tasks. Once $\mathbf{w}$ is obtained, the prediction $y_{t,i}$ for regression is made by $y_{t,i} = \mathbf{w}^\top(\mathbf{x}_{t,i}\otimes\mathbf{e}_t)$, where $\mathbf{e}_t$ is the task indicator vector with the tth entry being 1 and 0 otherwise.

Since regularizing the rank operator usually results in a NP-hard problem, as a common practice, we seek the convex surrogate of the non-convex rank operator. Here we use the nuclear norm to replace the rank operator, and obtain Hidden Task Enhanced Multi-Task Learning (**HTEMTL**):

$$\begin{aligned}\min_{\mathbf{W},\mathbf{Z},\mathbf{L}} &\mathcal{L}(\mathcal{D},\mathbf{W}) + \frac{\lambda}{2}\|\mathbf{W}-\mathbf{WZ}-\mathbf{LW}\|_F^2\\ &+ \gamma(\|\mathbf{Z}\|_* + \|\mathbf{L}\|_*) + \frac{\beta}{2}\|\mathbf{W}\|_F^2. \quad (9)\end{aligned}$$

It is worth noting that, if we ignore the matrix $\mathbf{L}$ in (9), the model will degrade to (2). Intuitively, introducing $\mathbf{L}$ allows $\mathbf{W}$ to be reconstructed from both the column direction and the row direction, that enables **HTEMTL** to utilize more useful information. In addition, replacing the original constraint $\mathbf{W} = \mathbf{WC}$ to $\mathbf{W} = \mathbf{WZ} + \mathbf{LW}$ drives the final task correlation away from the shape interaction matrix [17], which makes the model less sensitive to the deviation of $\mathbf{W}$ from its ground-truth values, and thus improves generalization.

4.3 *Progressive HTEMTL (pHTEMTL)*

HTEMTL essentially performs parameter learning in MTL by recovering the subspace structures of tasks via a single-layer model. For clarity, we can reformulate (7) by

$$\mathbf{w} = (\mathbf{Z}\otimes\mathbf{I}_d + \mathbf{I}_T\otimes\mathbf{L})\,\mathbf{w} = \mathbf{Cw}, \quad (10)$$

where $\mathbf{w} = \text{vec}(\mathbf{W})$ is the vectorization of $\mathbf{W}$ and $\otimes$ denotes the Kronecker product. By treating each entry w_j of $\mathbf{w}$ as a node in a network, (10) is represented by a fully-connected linear layer, namely *hidden tasks enhanced self-expressive layer* (HTE layer), as w_j is a linear combination of $\mathbf{w}$ with the jth row of $\mathbf{C}$ being its coefficients. Fig. 1 shows the framework of **HTEMTL** with the HTE layer.

However, **HTEMTL** cannot extract deep hierarchical information embedded in the parameter space, due to its single-layer nature as shown in (10). Here we present an effective approach to extend it to a multi-layer one, in order to detect deep task group structures, that helps to further improve generalization. To this end, a hierarchical architecture of model parameters is developed by cascading multiple HTE layers, so that subspace structures can be refined layer by layer. Specifically, in the kth layer of total m layers, the parameters $\mathbf{w}_k = \text{vec}(\mathbf{W}_k)$ is recursively represented by

$$\mathbf{w}_k = \mathbf{C}_{k-1}\mathbf{w}_{k-1} = \prod_{\ell=1}^{k-1} C_\ell \mathbf{w}_1, \tag{11}$$

where $\mathbf{C}_\ell = \mathbf{Z}_\ell \otimes \mathbf{I}_d + \mathbf{I}_T \otimes \mathbf{L}_\ell$ denotes the layer weights, and $\mathbf{W}_m$ in the last layer is used to calculate the loss $\mathcal{L}(\mathcal{D}, \mathbf{W}_m)$. In this way, $\mathbf{W}_m$ is learned progressively across multiple layers, extracting deep task subspace structures by $\mathbf{Z}_m$, which is enhanced by deep hidden tasks through $\mathbf{L}_m$.

Therefore, based on Eqs. (9) and (11), we propose the **p**rogressive **HTEMTL** (**pHTEMTL**) method:

$$\begin{aligned}\min_{\substack{\mathbf{W}_1,\\ \{\mathbf{Z}_k\},\{\mathbf{L}_k\}}} \quad & \mathcal{L}(\mathcal{D}, \mathbf{W}_m) + \sum_{k=1}^{m} \Big(\frac{\lambda_k}{2} \|\mathbf{W}_k - \mathbf{W}_k\mathbf{Z}_k - \mathbf{L}_k\mathbf{W}_k\|_F^2 \\ & + \gamma_k(\|\mathbf{Z}_k\|_* + \|\mathbf{L}_k\|_*)\Big) + \frac{\beta}{2}\|\mathbf{W}_1\|_F^2,\end{aligned} \tag{12}$$

where $\mathbf{w}_k = \prod_{\ell=1}^{k-1} \mathbf{C}_\ell \mathbf{w}_1$ ($k = 2, 3, ..., m$). Instead of directly setting appropriate values of λ_ks and γ_ks, we use $\lambda_k = \frac{\lambda}{\phi^{k-1}}$ and $\gamma_k = \frac{\gamma}{\phi^{k-1}}$ to control the strength of the HTE regularizer in the kth layer, and treat ϕ as a positive hyperparameter. Obviously, **pHTEMTL** is equivalent to **HTEMTL** when $m = 1$, but it has a chance to learn more accurate and robust task subspace structures based on deep information extracted in previous layers when $m > 1$.

4.4 Probabilistic Interpretation

In this subsection, we show that **HTEMTL** is actually related to the Maximum a Posterior (MAP) solution of a probabilistic model, from which perspective it is also related to several popular MTL methods. We first rewrite the second term of (9) with (10):

$$\begin{aligned}&\|\mathbf{W} - \mathbf{WZ} - \mathbf{LW}\|_F^2 \\ =&\|\mathbf{w} - (\mathbf{Z} \otimes \mathbf{I}_d)\mathbf{w} - (\mathbf{I}_T \otimes \mathbf{L})\mathbf{w}\|_2^2 \\ =&\| (\mathbf{I}_{dT} - (\mathbf{Z} \otimes \mathbf{I}_d) - (\mathbf{I}_T \otimes \mathbf{L}))\, \mathbf{w}\|_2^2 \\ =&\mathbf{w}^\top \mathbf{M}\mathbf{w},\end{aligned} \tag{13}$$

where $\mathbf{M}$ is defined by[2]

$$\mathbf{M} = (\mathbf{I} - \mathbf{Z} \otimes \mathbf{I} - \mathbf{I} \otimes \mathbf{L})^2. \tag{14}$$

Based on (13), we can reformulate (9) by a compact form:

$$\min_{\mathbf{w},\mathbf{Z},\mathbf{L}} \mathcal{L}(\mathcal{D}, \mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^\top(\mathbf{M} + \frac{\beta}{\lambda}\mathbf{I})\mathbf{w} + \gamma\Omega(\mathbf{Z}, \mathbf{L}). \tag{15}$$

Given the compact form in (15), to learn multiple tasks, the MAP estimation of parameters $\mathbf{w}$ given $\mathcal{D}$ is

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{w}) \cdot \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{X}_t, \mathbf{w}_t), \tag{16}$$

where $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_T]$ and $\mathbf{y} = [\mathbf{y}_1; \mathbf{y}_2; ...; \mathbf{y}_T]$. The Maximum Likelihood Estimation (MLE) part $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$ can be modeled by the Gaussian distribution and the Bernoulli distribution, leading to the squared loss and the logistic loss, respectively. The prior part $p(\mathbf{w})$ in (16) is crucial to **HTEMTL**, as it should hold the fundamental assumption that task parameters $\mathbf{w}$ come from a union of low-rank subspaces. According to (15), **HTEMTL** actually defines the prior by matrix-variate normal distribution [10] as

$$p(\mathbf{w}) = \mathcal{N}\left(\mathbf{0}, (\mathbf{M} + \frac{\beta}{\lambda}\mathbf{I})^{-1}\right), \tag{17}$$

where $(\mathbf{M} + \frac{\beta}{\lambda}\mathbf{I})^{-1}$ is a $dT \times dT$ positive definite matrix indicating the covariance between elements of $\mathbf{w}$. Different from many popular MTL methods focusing on modeling either task covariance [15, 41, 21] or feature covariance [28, 29, 24], according to (17), **HTEMTL** enables to capture both task correlation and feature correlation by $\mathbf{Z}$ and $\mathbf{L}$, respectively.

5 Optimization

In this section, we present the optimization algorithm of **pHTEMTL**, and the algorithm of **HTEMTL** can be simply obtained with $m = 1$. The objective function in (12) is bi-convex w.r.t. $\mathbf{W}_1$, $\{\mathbf{Z}_k\}$ and $\{\mathbf{L}_k\}$, respectively, so the alternating optimization is applied to solve for the variables, which is guaranteed to converge to the local minimum[3].

5.1 Initialization

Since (12) is a non-convex problem, initialization can affect the result. Therefore, for **pHTEMTL**, we execute **HTEMTL** to initialize $\mathbf{W}_1$, $\mathbf{Z}_1$ and $\mathbf{L}_1$ for the first layer, and set $\mathbf{Z}_k = \mathbf{Z}_1$ and $\mathbf{L}_k = \mathbf{L}_1$ ($k = 2, 3, ..., m$) for the other layers. For **HTEMTL**, we initialize $\mathbf{W}$ by single task learning with the l_2-norm regularization, and simply set $\mathbf{Z}, \mathbf{L} = \mathbf{0}$.

5.2 Updating $\mathbf{W}_1$

With fixed $\{\mathbf{Z}_k\}$ and $\{\mathbf{L}_k\}$, the subproblem becomes:

$$\min_{\mathbf{W}_1} \mathcal{L}(\mathcal{D}, \mathbf{w}_m) + \sum_{k=1}^{m} \left(\frac{\lambda_k}{2} \|\mathbf{w}_k - \mathbf{C}_k\mathbf{w}_k\|_2^2 \right) + \frac{\beta}{2}\|\mathbf{w}_1\|_2^2, \tag{18}$$

where $\mathbf{w}_k = \mathbf{C}_{k-1}\mathbf{w}_{k-1}$ ($k = 2, 3, ..., m$). We propose to use gradient descent to solve this subproblem. Let $\nabla_{\mathbf{W}_1}\mathcal{L}$ be the gradient of $\mathcal{L}(\mathcal{D}, \mathbf{W}_m)$ w.r.t. $\mathbf{W}_1$, it is recursively calculated based on the chain rule of derivatives, i.e.,

$$\nabla_{\mathbf{W}_{i-1}}\mathcal{L} = \nabla_{\mathbf{W}_i}\mathcal{L}\, \mathbf{Z}_{i-1}^\top + \mathbf{L}_{i-1}^\top \nabla_{\mathbf{W}_i}\mathcal{L}, \tag{19}$$

[2] We omit the subscript of the identity matrix $\mathbf{I}$ for clarity.

[3] Although large values of m may complicate the optimization landscape, $m = 2$ or 3 is typically recommended in experiments (Fig. 5), and the algorithm usually converges within a small number of iterations.

where $i = 2, 3, ..., m$. Similarly, the gradient $\nabla_{\mathbf{W}_1}\mathcal{Q}$ of the HTE regularizer $\mathcal{Q}$ w.r.t. $\mathbf{W}_1$ is obtained by

$$\nabla_{\mathbf{W}_{i-1}}\mathcal{Q} = \nabla_{\mathbf{W}_{i-1}}\mathcal{Q}_{i-1} + \sum_{k=i}^{m}\left(\nabla_{\mathbf{W}_i}\mathcal{Q}_k\,\mathbf{Z}_i^\top + \mathbf{L}_i^\top\nabla_{\mathbf{W}_i}\mathcal{Q}_k\right), \tag{20}$$

where $i = 2, 3, ..., m$, $\nabla_{\mathbf{W}_k}\mathcal{Q}_k = \mathbf{H}_k$ ($\forall k$) is the kth component of $\nabla_{\mathbf{W}_k}\mathcal{Q}$, and $\mathbf{H}_k$ is defined by

$$\mathbf{H}_k = \Delta\mathbf{W}_k - \Delta\mathbf{W}_k\mathbf{Z}_k^\top - \mathbf{L}_k^\top\Delta\mathbf{W}_k, \tag{21}$$

with $\Delta\mathbf{W}_k = \mathbf{W}_k - \mathbf{W}_k\mathbf{Z}_k - \mathbf{L}_k\mathbf{W}_k$. Thus, with certain stepsize η, the update of $\mathbf{W}_1$ in each step is:

$$\mathbf{W}_1 = \mathbf{W}_1 - \eta\left(\nabla_{\mathbf{W}_1}\mathcal{L} + \nabla_{\mathbf{W}_1}\mathcal{Q} + \beta\mathbf{W}_1\right). \tag{22}$$

5.3 *Updating* $\{\mathbf{Z}_k\}$

With fixed $\mathbf{W}_1$ and $\{\mathbf{L}_k\}$, we propose to update $\{\mathbf{Z}_k\}$ for (12) from the mth HTE layer to the 1st layer. We fix the rest of weights for the ith layer, and obtain the problem w.r.t. $\mathbf{Z}_i$:

$$\min_{\mathbf{Z}_i}\mathcal{L}(\mathcal{D}, \mathbf{w}_m) + \sum_{k=i}^{m}\left(\frac{\lambda_k}{2}\|\mathbf{w}_k - \mathbf{C}_k\mathbf{w}_k\|_2^2\right) + \gamma_i\|\mathbf{Z}_i\|_*, \tag{23}$$

where $\mathbf{w}_k = \prod_{\ell=i}^{k-1}\mathbf{C}_\ell\mathbf{w}_i$, $\forall k > i$, with $\mathbf{w}_i = \prod_{\ell=1}^{i-1}\mathbf{C}_\ell\mathbf{w}_1$. It is a convex but non-smooth problem, and thus we propose to solve it by proximal method [27], which decomposes its objective into two components, smooth $f(\mathbf{Z}_i)$ and non-smooth $g(\mathbf{Z}_i)$:

$$\begin{aligned} f(\mathbf{Z}_i) =& \mathcal{L}(\mathcal{D}, \mathbf{w}_m) + \sum_{k=i}^{m}\left(\frac{\lambda_k}{2}\|\mathbf{w}_k - \mathbf{C}_k\mathbf{w}_k\|_2^2\right), \\ g(\mathbf{Z}_i) =& \gamma_i\|\mathbf{Z}_i\|_*. \end{aligned} \tag{24}$$

In the jth iteration, the optimal solution is calculated by the following proximal operator:

$$\min_{\mathbf{Z}_i}\frac{1}{2\eta}\|\mathbf{Z}_i - (\mathbf{Z}_i^j - \eta\nabla_{\mathbf{Z}_i}f)\|_F^2 + \gamma_i\|\mathbf{Z}_i\|_*, \tag{25}$$

where $\nabla_{\mathbf{Z}_i}f$ is the gradient of $f(\mathbf{Z}_i^j)$ w.r.t. $\mathbf{Z}_i$ and η is a stepsize. In each iteration, the update in (30) is done by the singular value thresholding (SVT) algorithm [6]. Let $\nabla_{\mathbf{Z}_i}\mathcal{L}$ and $\nabla_{\mathbf{Z}_i}\mathcal{Q}$ denote the derivatives of the loss function and the HTE regularizer w.r.t. $\mathbf{Z}_i$, respectively, we have $\nabla_{\mathbf{Z}_i}f = \nabla_{\mathbf{Z}_i}\mathcal{L} + \nabla_{\mathbf{Z}_i}\mathcal{Q}$. Based on the chain rule of derivatives, $\nabla_{\mathbf{Z}_i}\mathcal{L}$ is recursively calculated as

$$\nabla_{\mathbf{Z}_i}\mathcal{L} = \mathbf{W}_i^\top\nabla_{\mathbf{W}_{i+1}}\mathcal{L}, \tag{26}$$

where $i = 1, 2, ..., m$ and $\nabla_{\mathbf{W}}\mathcal{L}$ is calculated by (19). Similarly, $\nabla_{\mathbf{Z}_i}\mathcal{Q}$ is obtained by

$$\nabla_{\mathbf{Z}_i}\mathcal{Q} = -\lambda_i\mathbf{W}_i^\top\Delta\mathbf{W}_i + \lambda_{i+1}\mathbf{W}_i^\top\mathbf{H}_{i+1} + \sum_{k=i+2}^{m}\mathbf{W}_i^\top\nabla_{\mathbf{W}_{i+1}}\mathcal{Q}_k, \tag{27}$$

where $\nabla_{\mathbf{W}}\mathcal{Q}$ and $\mathbf{H}$ are given in (20) and (21), respectively.

5.4 *Updating* $\{\mathbf{L}_k\}$

As it is similar to the step on updating $\{\mathbf{Z}_k\}$, we omit the procedure for simplicity and present it in the supplement.

5.5 *Computational Analysis*

The procedure for the above algorithm is provided in the supplement. In terms of time complexity, when updating $\mathbf{W}_1$, each iteration of the gradient descent algorithm costs $\mathcal{O}(mdT(d+T) + dN)$, where $N = \sum_{t=1}^{T}N_t$. To update $\{\mathbf{Z}_k\}$ and $\{\mathbf{L}_k\}$, in each step of proximal gradient algorithm, calculating $\nabla_{\mathbf{Z}}f$ and $\nabla_{\mathbf{Z}}f$ takes $\mathcal{O}(mdT(d+T) + dN)$, and conducting SVT needs $\mathcal{O}(T^3)$ and $\mathcal{O}(d^3)$ for $\mathbf{Z}$ and $\mathbf{L}$, respectively. Thus, the complexity is linear w.r.t. the number N of samples and the number m of layers.

6 Experiments

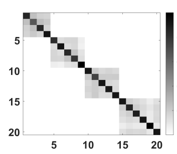
6.1 *Experimental Setting*

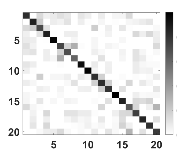
We compare the proposed methods with six benchmark methods. Single-task learning (**STL**) learns each task independently with the l_2-norm regularization. **GOMTL** [20] factorizes the parameter matrix into a product of a shared latent basis and a sparse latent assignment matrix for task grouping and overlapping. **AMTL** [21] assumes that each task is represented by a sparse non-negative linear combination of other tasks, and thus information is transferred between tasks in an asymmetric manner. **GAMTL** [26] assumes that each task is a linear combination of other tasks, and the trace Lasso [12] is imposed on the correlation matrix to maintain a group structure. **GBDSP** [35] uses a k-block diagonal regularizer to encourage the latent basis to have exactly k blocks, and thus inter-group information transferring is punished. However, it needs k as prior knowledge. **KMSV** [8] jointly learns multiple tasks based on tight approximations of the rank operator.

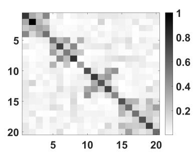
The hyperparameter for STL with the l_2-norm regularization is selected from $\{10^{-5}, 10^{-4}, \cdots, 1\}$. For GOMTL, AMTL, GAMTL and GBDSP, the search grids of their hyperparameters are set to the recommendation in the original papers, and the codes are obtained from the authors. For KMSV, the number of smallest singular values is selected from $\{3, 5, 7, 9, 11\}$. For **pHTEMTL**, we set the number m of layers by $m = 2$ or 3. The search grids for λ and γ are both set to $\{10^{-4}, 10^{-3}, \cdots, 10^2\}$, and β's and ϕ's search grids are set to $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$. For regression tasks, 25% of the dataset is randomly drawn as the testing set and the rest as the training set, and 10-fold cross validation is used to select the best hyperparameter. For classification tasks, the data is split into the training set, the validation set and the testing set. All experiments are repeated for 10 times by using different subsets of the data. For regression tasks, the two evaluation metrics we adopt are root Mean Squared Error (rMSE) and Mean Absolute Error (MAE). For classification tasks, we use Error Rate (ER) and Area Under ROC Curve (AUC) to measure the performance. All metrics are formally defined in the supplement.

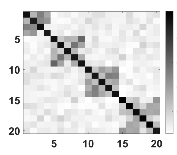
6.2 *Synthetic Data*

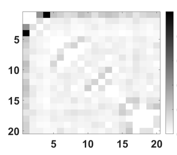
We generate a synthetic dataset with $d = 150$. There are 4 clusters of task parameters with dimensions $3, 4, 4$ and 5, that contain $4, 5, 5$ and 6 binary classification tasks, respectively. The procedure of generating the task parameters is summarized as the following: (1) We generate a set of orthogonal basis of the d-dimensional space by applying QR decomposition to a full rank $d \times d$ matrix. (2) All subspaces' bases are chosen as disjoint subsets of the orthogonal basis produced by the last step. (3) The task parameters are generated by multiplying the subspace's basis with a random vector, in which each element of the vector is sampled from a uniform distribution in the

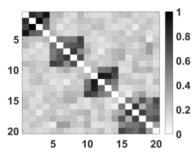

(a) Ground-truth

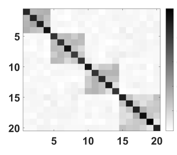

(b) STL

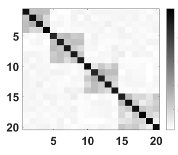

(c) GOMTL


(d) GBDSP


(e) AMTL


(f) GAMTL


(g) **HTEMTL** (**Z**)


(h) **pHTEMTL** ($\mathbf{Z}_1$)

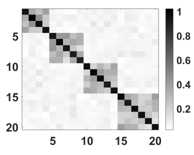

(i) **pHTEMTL** ($\mathbf{Z}_2$)

Figure 2: The task correlation matrices learned by different algorithms on the synthetic dataset. Each matrix is an average of 10 trials. The generated dataset has 4 task clusters, that contain 4, 5, 5 and 6 binary classification tasks, respectively.

interval of $[-0.5, 0.5]$. With the simulated task parameters, we generated 200 training samples, 100 validation samples and 100 testing samples for each task. For the i-th observation of the t-th task, we have $\mathbf{x}_t^i \sim \mathcal{N}(0, \mathbf{I}_d)$, and $\mathbf{y}_t^i = \text{sign}(\mathbf{w}_t^\top \mathbf{x}_t^i + \mathcal{N}(0, 0.01))$, where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

Fig. 2 shows the average task correlation matrices from 10 trials. The task correlation matrix shown in (a) is produced by directly applying LRR to the ground-truth task parameters. Similarly, the result in (b) is produced by applying LRR to the task parameters $\mathbf{W}$ learned by STL. However, the task correlation structure is poorly recovered, mainly because of the deviation of the learned task parameters from the ground-truth values. For GOMTL and GBDSP, both methods factorize the task parameter matrix with $\mathbf{W} = \mathbf{LS}$, and we represent their task correlation matrices $\mathbf{Z}$ with $\mathbf{S}^\top\mathbf{S}$ in (c) and (d). For AMTL, GAMTL, and **HTEMTL**, the task correlation matrices $\mathbf{Z}$ are directly produced, which are shown in (e), (f) and (g), respectively. For **pHTEMTL**, $\mathbf{Z}_1$ and $\mathbf{Z}_2$ are obtained by its two layers, which are shown in (h) and (i), respectively. To make the structure clearer, we plot $(|\mathbf{Z}| + |\mathbf{Z}|^\top)/2$ and normalize all its elements to the scale of $[0, 1]$. We can see that the task correlation learned by AMTL is much worser than the others. This is probably because it holds a strong assumption that all the task parameters are reconstructed by a positive linear combination of the others, i.e., all task parameters stay in the same quadrant of the feature space. We see that **HTEMTL** and **pHTEMTL** can recover the task cluster structure better than the other methods, and the deeper the layer, the clearer the structure. The testing results in terms of ER and AUC are reported in the first two rows of Table 1, where the best two results are highlighted in boldface. As shown in Table 1, we find out that the better a method can recover the task correlation matrix, the better performance it can achieve. Thus, **HTEMTL** and **pHTEMTL** perform the best among all the comparing methods.

6.3 Real-world Data

In experiments, we use four real-world multi-task datasets:

- **Fashion-MNIST**[4]: This is a dataset comprised of 10 types of clothing, such as shoes, t-shirts and more. We reduce the dimensionality to 128 by Principal Components Analysis (PCA) to retain 95% variance. There are $\binom{10}{2} = 45$ one vs. one tasks in total. For each task, we construct a training set by randomly choosing 10 positive samples and 10 negative samples, and there are 100 positive samples and 100 negative samples in the validation set and the testing set.
- **CIFAR-10**[5]: Like Fashion-MNIST, this is also an image classification dataset with 10 classes. Again, we generate $\binom{10}{2} = 45$ one vs. one binary classification tasks. The dimensionality is reduced to 210 after applying PCA to retain 95% variance. In each task, we randomly drawn 100 training samples from each class as the training set. For the validation set and the testing set, there are 400 samples from each class.
- **AWA2-Attribute**[6]: This dataset is constructed based on the AWA-2 dataset [33]. There are 85 tasks. Each task is a binary classification on recognizing whether a given attribute is presented in a given instance. We adopt the ILSVRC-pretrained ResNet101 feature [33] and use PCA to reduce the dimensionality to 467 to retain 90% variance. The training set for each task is generated by randomly choosing 50 positive samples and 50 negative samples. Both the validation set and the testing set have 100 positive samples and 100 negative samples.
- **School**[7]: This is a regression dataset which is consisted of exam scores of students from 139 schools. Each school corresponds to a task. We use the features provided in the MALSAR package [42], and the dimension of the feature space is 28. We use 75% of the dataset as the training set and the left 25% as the testing set.

The last twelve rows of Table 1 show the prediction performance of the eight comparing methods on the six real-world datasets. In most cases, MTL methods can achieve better generalization performance than STL. This indicates the effectiveness of learning tasks together. Both **HTEMTL** and **pHTEMTL** can outperform the other comparing algorithms on all the classification datasets in terms of ER, and on most classification datasets in terms of AUC. This confirms our hypothesis that exploiting the effect of hidden tasks can achieve better task grouping effect and boost the generalization performance. GBDSP's performance is close to ours, but it requires the number of task clusters and the dimension of the latent basis as extra prior knowledge, which are usually not available in practice and need to be determined by validation, making it very time-consuming. For the School dataset, both **HTEMTL** and **pHTEMTL** fail to outperform AMTL, GBDSP and KMSV, probably because the tasks in this dataset are quite similar with each other, and the subspace structure of the task clusters is not obvious [26].

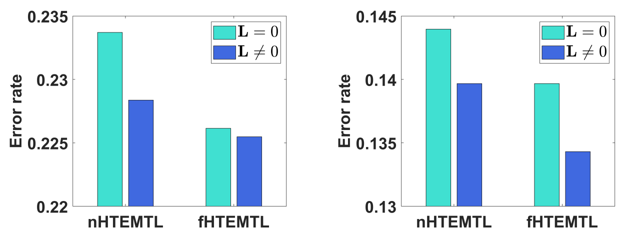
6.4 Effectiveness of Hidden Tasks Enhancement

In this subsection, we study the effect of hidden tasks. Deactivating the effect of hidden tasks is equivalent to set the matrix $\mathbf{L}$ in (9) and (12) to be zero. In this way, **HTEMTL** degrades to the naive model in (2), and **pHTEMTL** degrades to a naive one that ignores deep hidden

[4] https://github.com/zalandoresearch/fashion-mnist
[5] https://www.cs.toronto.edu/ kriz/cifar.html
[6] https://cvml.ist.ac.at/AwA2/
[7] https://github.com/jiayuzhou/MALSAR

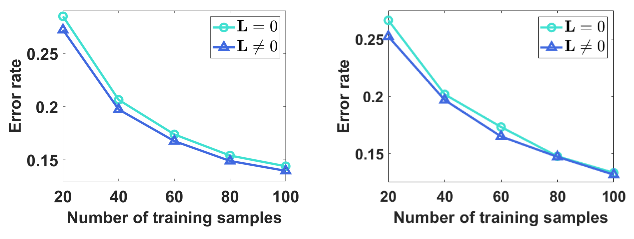
Table 1: Experimental results (mean ± std) with different evaluation metrics. The best two results are highlighted in boldface.

Dataset	Measure	STL	GOMTL	AMTL	GAMTL	GBDSP	KMSV	**HTEMTL**	**pHTEMTL**
Synthetic	ER↓	0.2392±0.0137	0.2271 ±0.0122	0.2377 ±0.0168	0.2233 ±0.0203	0.2179±0.0119	0.2340±0.0115	**0.2076±0.0128**	**0.2042±0.0044**
	AUC↑	0.8535±0.0415	0.8617±0.0464	0.8651±0.0371	0.8795±0.0348	0.8795±0.0313	0.8235±0.0281	**0.8906±0.0362**	**0.8868±0.0099**
Fashion-MNIST	ER↓	0.1097±0.0042	0.1017±0.0045	0.0751±0.0035	0.0717±0.0061	0.0847±0.0066	0.1012±0.0084	**0.0699±0.0130**	**0.0710±0.0051**
	AUC↑	0.9812±0.0107	**0.9937±0.0032**	0.9824±0.0086	0.9893±0.0078	0.9888±0.0052	0.9785±0.0072	0.9865±0.0163	**0.9905±0.0019**
CIFAR-10	ER↓	0.2880±0.0029	0.2393±0.0034	0.2387±0.0032	0.2361±0.0036	0.2359±0.0032	0.2525±0.0039	**0.2284±0.0041**	**0.2234±0.0011**
	AUC↑	0.8183±0.0089	0.8836±0.0094	0.8511±0.0139	0.8730±0.0099	0.8809±0.0118	0.8347±0.0168	**0.8878±0.0083**	**0.8880±0.0019**
AWA2-Attribute	ER↓	0.1784±0.0019	0.1753±0.0055	0.1493±0.0030	0.1550±0.0036	0.1789±0.0047	0.1794±0.0054	**0.1397±0.0031**	**0.1316±0.0009**
	AUC↑	0.7300±0.0421	0.7276±0.0595	0.7270±0.0595	0.7286±0.0725	0.7608±0.0622	**0.7917±0.0749**	0.7436±0.0584	**0.7619±0.0194**
School	rMSE↓	10.3127±0.0602	10.1606±0.0712	10.1604±0.0712	10.2398±0.0557	**10.1218±0.1035**	**10.1320±0.0711**	10.1806±0.0878	10.1769±0.0038
	MAE↓	8.1472±0.0379	8.1502±0.1764	**8.0321±0.0463**	8.0949±0.0305	**7.9732±0.0739**	8.0427±0.0668	8.0443±0.0480	8.0370±0.0168



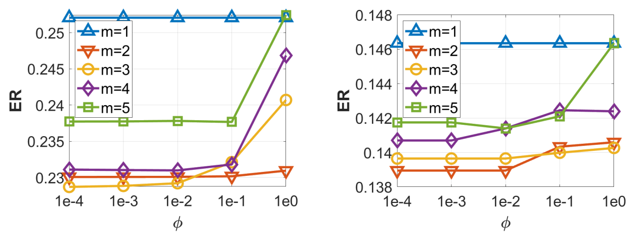
(a) CIFAR-10 (b) AWA2-Attribute

Figure 3: Comparison of the prediction performance in ER between the naive models ($\mathbf{L} = \mathbf{0}$) and the original models ($\mathbf{L} \neq \mathbf{0}$).



(a) **HTEMTL** (b) **pHTEMTL**

Figure 4: Comparison of the prediction performance in ER between the naive models ($\mathbf{L} = \mathbf{0}$) and the original models ($\mathbf{L} \neq \mathbf{0}$) on the AWA2-Attribute dataset by varying the number of training samples from 20 to 100 by step 20.



(a) CIFAR-10 (b) AWA2-Attribute

Figure 5: Analysis on the effectiveness of cascading HTE layers in **pHTEMTL** on the CIFAR-10 and AWA2-Attribute datasets.

tasks. We conduct the same experiments to compare the performance between the original models and their naive variants. The results on CIFAR-10 and AWA2-Attribute are shown in Fig. 8. We can see that on the two datasets, for both **HTEMTL** and **pHTEMTL**, the introduction of hidden tasks effectively improves the MTL model's generalization performance. This validates our hypothesis that exploiting the effect of hidden tasks can prevent the task grouping procedure from amplifying the deviation of the learned task parameters from the ground-truth values.

To further show the effectiveness of hidden tasks enhancement, we vary the number of training samples for the experiments carried on the AWA2-Attribute dataset. In each turn, we add 10 more positive samples and 10 more negative samples to the training set. By cumulatively adding training samples, we can indirectly decrease the amount of the deviation between the learned task parameters and the ground-truth values. From Fig. 4, we can see that at different levels of the deviation, the introduction of hidden tasks can consistently improve the MTL model's generalization performance and effectively robustify the model.

6.5 *Effectiveness of Cascading HTE Layers*

To show the effectiveness of cascading HTE layers in **pHTEMTL**, we evaluate its performance by varying the number m of HTE layers, $m \in \{1, 2, 3, 4, 5\}$. In (12), ϕ controls the layer-wise strength of the HTE regularizer. When $\phi < 1$, stronger regularization is imposed on deeper layers and $\phi > 1$ otherwise. In this experiment, we select the value of ϕ from $\{10^{-4}, 10^{-3}, ..., 1\}$ and fix $\gamma = 10^{-1}$, $\lambda = 10^{-2}$ and $\beta = 10^{-2}$. Fig. 5 reports the results in terms of ER on CIFAR-10 and AWA2-Attribute. From Fig. 5, we can see that **pHTEMTL** consistently outperforms **HTEMTL** ($m = 1$), probably because its multi-layer nature indeed helps to learn accurate and robust task subspace, which leads to improved generalization. In most cases, the best performance is achieved when $m = 2$ or 3, indicating using more layers does not necessarily improve the performance. Regardless of m, the best performance is always achieved when ϕ is around 10^{-3}, Thus, it is recommended to set $m \in \{2, 3\}$ and $10^{-4} < \phi < 10^{-2}$ on the two datasets.

7 Conclusion

In this work, we propose a novel MTL method, named **HTEMTL**, that incorporates the grouped MTL problem with subspace clustering, and exploits the effect of hidden tasks to improve generalization. Moreover, **pHTEMTL** is proposed by extending **HTEMTL** to a multi-layer model, so as to progressively extract deep latent information of parameters. Experimental results on both synthetic and real-world datasets show that our method not only recovers the task cluster structure clearly, but also achieves better prediction performance than state-of-the-art grouped MTL methods.

References

[1] Rie Kubota Ando, Tong Zhang, and Peter Bartlett, 'A framework for learning predictive structures from multiple tasks and unlabeled data.', *Journal of Machine Learning Research*, **6**(11), (2005).

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil, 'Convex multi-task feature learning', *Machine learning*, **73**(3), 243–272, (2008).

[3] Andreas Argyriou, Rina Foygel, and Nathan Srebro, 'Sparse prediction with the k-support norm', in *Advances in Neural Information Processing Systems*, volume 25, (2012).

[4] Guangji Bai, Johnny Torres, Junxiang Wang, Liang Zhao, Cristina Abad, and Carmen Vaca, 'Sign-regularized multi-task learning', in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*, pp. 793–801. SIAM, (2023).

[5] Stephen Boyd, Neal Parikh, and Eric Chu, *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Now Publishers Inc, 2011.

[6] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen, 'A singular value thresholding algorithm for matrix completion', *SIAM Journal on optimization*, **20**(4), 1956–1982, (2010).

[7] Rich Caruana, 'Multitask learning', *Machine learning*, **28**(1), 41–75, (1997).

[8] Wei Chang, Feiping Nie, Rong Wang, and Xuelong Li, 'New tight relaxations of rank minimization for multi-task learning', in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, (2021).

[9] Joao Paulo Costeira and Takeo Kanade, 'A multibody factorization method for independently moving objects', *International Journal of Computer Vision*, **29**(3), 159–179, (1998).

[10] A. P. Dawid, 'Some matrix-variate distribution theory: Notational considerations and a bayesian application', *Biometrika*, **68**(1), 265–274, (1981).

[11] Ehsan Elhamifar and René Vidal, 'Sparse subspace clustering: Algorithm, theory, and applications', *IEEE transactions on pattern analysis and machine intelligence*, **35**(11), 2765–2781, (2013).

[12] Edouard Grave, Guillaume R Obozinski, and Francis Bach, 'Trace lasso: a trace norm regularization for correlated designs', *Advances in Neural Information Processing Systems*, **24**, 2187–2195, (2011).

[13] David J Hand and Robert J Till, 'A simple generalisation of the area under the roc curve for multiple class classification problems', *Machine learning*, **45**(2), 171–186, (2001).

[14] Xiao He, Francesco Alesiani, and Ammar Shaker, 'Efficient and scalable multi-task regression on massive number of tasks', *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**(01), 3763–3770, (Jul. 2019).

[15] Laurent Jacob, Francis Bach, and Jean-Philippe Vert, 'Clustered multi-task learning: a convex formulation', in *Advances in Neural Information Processing Systems*, volume 21, (2009).

[16] Jun-Yong Jeong and Chi-Hyuck Jun, 'Variable selection and task grouping for multi-task learning', in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1589–1598, (2018).

[17] Pan Ji, Mathieu Salzmann, and Hongdong Li, 'Shape interaction matrix revisited and robustified: Efficient subspace clustering with corrupted and incomplete data', in *Proceedings of the IEEE International Conference on computer Vision*, pp. 4687–4695, (2015).

[18] Zhuoliang Kang, Kristen Grauman, and Fei Sha, 'Learning with whom to share in multi-task feature learning', in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, p. 521–528, (2011).

[19] Meghana Kshirsagar, Eunho Yang, and Aurélie C Lozano, 'Learning task clusters via sparsity grouped multitask learning', in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 673–689. Springer, (2017).

[20] Abhishek Kumar and Hal Daumé, 'Learning task grouping and overlap in multi-task learning', in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, p. 1723–1730, (2012).

[21] Giwoong Lee, Eunho Yang, and Sung Hwang, 'Asymmetric multi-task learning based on task relatedness and loss', in *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pp. 230–238, (2016).

[22] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma, 'Robust recovery of subspace structures by low-rank representation', *IEEE transactions on pattern analysis and machine intelligence*, **35**(1), 171–184, (2012).

[23] Guangcan Liu and Shuicheng Yan, 'Latent low-rank representation for subspace segmentation and feature extraction', in *2011 international conference on computer vision*, pp. 1615–1622. IEEE, (2011).

[24] Jun Liu, Shuiwang Ji, and Jieping Ye, 'Multi-task feature learning via efficient l2, 1-norm minimization', in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, p. 339–348, Arlington, Virginia, USA, (2009). AUAI Press.

[25] Shengchao Liu, Yingyu Liang, and Anthony Gitter, 'Loss-balanced task weighting to reduce negative transfer in multi-task learning', *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**(01), 9977–9978, (Jul. 2019).

[26] Sulin Liu and Sinno Jialin Pan, 'Adaptive group sparse multi-task learning via trace lasso.', in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp. 2358–2364, (2017).

[27] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, volume 87, Springer Science & Business Media, 2003.

[28] Guillaume Obozinski, Ben Taskar, and Michael Jordan, 'Multi-task feature selection', *Statistics Department, UC Berkeley, Tech. Rep*, **2**(2.2), 2, (2006).

[29] Guillaume Obozinski, Ben Taskar, and Michael I Jordan, 'Joint covariate selection and joint subspace selection for multiple classification problems', *Statistics and Computing*, **20**(2), 231–252, (2010).

[30] Saullo H. G. Oliveira, André R. Gonçalves, and Fernando J. Von Zuben, 'Group lasso with asymmetric structure estimation for multi-task learning', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3202–3208, (7 2019).

[31] Sinno Jialin Pan and Qiang Yang, 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering*, **22**(10), 1345–1359, (2009).

[32] René Vidal, 'Subspace clustering', *IEEE Signal Processing Magazine*, **28**(2), 52–68, (2011).

[33] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata, 'Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly', *IEEE transactions on pattern analysis and machine intelligence*, **41**(9), 2251–2265, (2018).

[34] Zhiyong Yang, Qianqian Xu, Xiaochun Cao, and Qingming Huang, 'Task-feature collaborative learning with application to personalized attribute prediction', *IEEE transactions on pattern analysis and machine intelligence*, (2020).

[35] Zhiyong Yang, Qianqian Xu, Yangbangyan Jiang, Xiaochun Cao, and Qingming Huang, 'Generalized block-diagonal structure pursuit: Learning soft latent task assignment against negative transfer', in *Advances in Neural Information Processing Systems*, volume 32, (2019).

[36] Yaqiang Yao, Jie Cao, and Huanhuan Chen, 'Robust task grouping with representative tasks for clustered multi-task learning', in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1408–1417, (2019).

[37] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn, 'Gradient surgery for multi-task learning', in *Advances in Neural Information Processing Systems*, eds., H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, volume 33, pp. 5824–5836. Curran Associates, Inc., (2020).

[38] Yu Zhang and Qiang Yang, 'A survey on multi-task learning', *IEEE Transactions on Knowledge and Data Engineering*, (2021).

[39] Yu Zhang, Dit-Yan Yeung, and Qian Xu, 'Probabilistic multi-task feature selection', *Advances in neural information processing systems*, **23**, 2559–2567, (2010).

[40] Zhao Zhang, Jiahuan Ren, Zheng Zhang, and Guangcan Liu, 'Deep latent low-rank fusion network for progressive subspace discovery', in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, ed., Christian Bessiere, pp. 2762–2768. International Joint Conferences on Artificial Intelligence Organization, (7 2020). Main track.

[41] Jiayu Zhou, Jianhui Chen, and Jieping Ye, 'Clustered multi-task learning via alternating structure optimization', *Advances in neural information processing systems*, **24**, (2011).

[42] Jiayu Zhou, Jianhui Chen, and Jieping Ye, 'Malsar: Multi-task learning via structural regularization', *Arizona State University*, **21**, (2011).

[43] Qiang Zhou and Qi Zhao, 'Flexible clustered multi-task learning by learning representative tasks', *IEEE transactions on pattern analysis and machine intelligence*, **38**(2), 266–278, (2015).

A More details on Optimization

A.1 Optimization of **pHTEMTL**

A.1.1 Updating $\{\mathbf{L}_k\}$

With fixed $\mathbf{W}_1$ and $\{\mathbf{Z}_k\}$, we propose to update $\{\mathbf{L}_k\}$ for (**??**) from the mth HTE layer to the 1st layer. We fix the rest of weights for the ith layer, and obtain the problem w.r.t. $\mathbf{L}_i$:

$$\min_{\mathbf{L}_i} \mathcal{L}(\mathcal{D}, \mathbf{w}_m) + \sum_{k=i}^{m} \left(\frac{\lambda_k}{2} \|\mathbf{w}_k - \mathbf{C}_k \mathbf{w}_k\|_2^2 \right) + \gamma_i \|\mathbf{L}_i\|_*, \quad (28)$$

where $\mathbf{w}_k = \prod_{\ell=i}^{k-1} \mathbf{C}_\ell \mathbf{w}_i, \forall k > i$, and $\mathbf{w}_i$ can be reconstructed by $\mathbf{w}_i = \prod_{\ell=1}^{i-1} \mathbf{C}_\ell \mathbf{w}_1$. It is a convex but non-smooth problem, and thus we propose to solve it by proximal method [27], which decomposes its objective into two components, smooth $f(\mathbf{L}_i)$ and non-smooth $g(\mathbf{L}_i)$:

$$\begin{aligned} f(\mathbf{L}_i) =& \mathcal{L}(\mathcal{D}, \mathbf{w}_m) + \sum_{k=i}^{m} \left(\frac{\lambda_k}{2} \|\mathbf{w}_k - \mathbf{C}_k \mathbf{w}_k\|_2^2 \right), \\ g(\mathbf{L}_i) =& \gamma_i \|\mathbf{L}_i\|_*. \end{aligned} \quad (29)$$

In the jth iteration, the optimal solution can be calculated by the following proximal operator:

$$\min_{\mathbf{L}_i} \frac{1}{2\eta} \|\mathbf{L}_i - (\mathbf{L}_i^j - \eta \nabla_{\mathbf{L}_i} f)\|_F^2 + \gamma_i \|\mathbf{L}_i\|_*. \quad (30)$$

where $\nabla_{\mathbf{L}_i} f$ is the gradient of $f(\mathbf{L}_i^j)$ w.r.t. $\mathbf{L}_i$ and η is a stepsize that satisfies

$$f(\mathbf{L}_i^{j+1}) \le f(\mathbf{L}_i^j) + \langle \nabla_{\mathbf{L}_i} f, \mathbf{L}_i^{j+1} - \mathbf{L}_i^j \rangle + \frac{1}{2\eta} \|\mathbf{L}_i^{j+1} - \mathbf{L}_i^j\|_F^2. \quad (31)$$

In each iteration, the update in (30) is done by the singular value thresholding (SVT) algorithm [6]. Let $\nabla_{\mathbf{L}_i} \mathcal{L}$ and $\nabla_{\mathbf{L}_i} \mathcal{Q}$ denote the derivatives of the loss function and the HTE regularizer w.r.t. $\mathbf{L}_i$, respectively, we have $\nabla_{\mathbf{L}_i} f = \nabla_{\mathbf{L}_i} \mathcal{L} + \nabla_{\mathbf{L}_i} \mathcal{Q}$. Based on the chain rule of derivatives, $\nabla_{\mathbf{L}_i} \mathcal{L}$ is recursively calculated by

$$\nabla_{\mathbf{L}_i} \mathcal{L} = \nabla_{\mathbf{W}_{i+1}} \mathcal{L} \, \mathbf{W}_i^\top, \quad (32)$$

where $i = 1, 2, ..., m$ and $\nabla_{\mathbf{W}} \mathcal{L}$ is calculated by (14). Let $\nabla_{\mathbf{L}_i} \mathcal{Q}_k$ be the kth component of $\nabla_{\mathbf{L}_i} \mathcal{Q}$, $\nabla_{\mathbf{L}_i} \mathcal{Q}$ is calculated in a similar way,

$$\begin{aligned} \nabla_{\mathbf{L}_i} \mathcal{Q} &= \sum_{k=i}^{m} \nabla_{\mathbf{L}_i} \mathcal{Q}_k \\ &= \nabla_{\mathbf{L}_i} \mathcal{Q}_i + \nabla_{\mathbf{L}_i} \mathcal{Q}_{i+1} + \sum_{k=i+2}^{m} \nabla_{\mathbf{L}_i} \mathcal{Q}_k, \\ &= -\lambda_i \Delta \mathbf{W}_i \mathbf{W}_i^\top + \lambda_{i+1} \mathbf{H}_{i+1} \mathbf{W}_i^\top \\ &+ \sum_{k=i+2}^{m} \nabla_{\mathbf{W}_{i+1}} \mathcal{Q}_k \mathbf{W}_i^\top \end{aligned} \quad (33)$$

where $i = 1, 2, ..., m$, and $\mathbf{H}$ and $\nabla_{\mathbf{W}} \mathcal{Q}_k$ are defined in (21) and (20), respectively.

The optimization algorithm of **pHTEMTL** is provided in Alg. 1.

Algorithm 1: Pseudocode for **pHTEMTL**

```
Input       : $\{\mathcal{D}_t\}_{t=1}^T$, $\lambda$, $\gamma$, $\beta$, $\phi$ and $m$
Initialize  : Initialize $\mathbf{W}_1$ $\{\mathbf{Z}_k\}$ and $\{\mathbf{L}_k\}$ by HTEMTL,
              that is initialized by STL.
while the objective is not converged do
    1. Update $\mathbf{W}_1$ : Gradient descent by (17).
    2. Update $\mathbf{Z}_k$ : Proximal method by (20) for
                   pHTEMTL, $\forall k$.
    3. Update $\mathbf{L}_k$ : Proximal method by (30) for
                   pHTEMTL, $\forall k$.
end
Output      : $\mathbf{W}_1$, $\{\mathbf{Z}_k\}$ and $\{\mathbf{L}_k\}$
```

A.2 Optimization of **HTEMTL**

Different from Alg. 1, we present an alternative algorithm for **HTEMTL**. The objective functions in (9) are both bi-convex with respect to $\mathbf{W}$, $\mathbf{Z}$ and $\mathbf{L}$, so the alternating optimization can be applied to solve the variables, which is guaranteed to converge to the local minimum. Specifically, in each iteration, we optimize one variable and keep the other two fixed, until the objective function satisfies certain stopping criteria.

A.2.1 Initialization

Since (9) is non-convex, the initialization can affect the final result. Therefore, we initialize $\mathbf{W}$ by the parameters learned by single task learning with the l_2-norm regularization, and simply set $\mathbf{Z}$ and $\mathbf{L}$ as zero matrices.

A.2.2 Updating $\mathbf{Z}$

With fixed $\mathbf{W}$ and $\mathbf{L}$, we propose to use ADMM [5] to update $\mathbf{Z}$ for (9). We introduce an auxiliary variable $\mathbf{J}$, and solve the following equivalent problem:

$$\min_{\mathbf{Z}, \mathbf{J}} \quad \frac{\lambda}{2} \|\mathbf{W} - \mathbf{W}\mathbf{Z} - \mathbf{L}\mathbf{W}\|_F^2 + \gamma \|\mathbf{J}\|_* \quad \text{s.t.} \quad \mathbf{Z} - \mathbf{J} = 0. \quad (34)$$

The augmented Lagrangian with $\mathbf{U}$ as the scaled dual variable and ρ as the penalty parameter is then defined by:

$$\mathcal{L}_\rho(\mathbf{Z}, \mathbf{J}, \mathbf{U}) = \frac{\lambda}{2} \|\mathbf{W} - \mathbf{W}\mathbf{Z} - \mathbf{L}\mathbf{W}\|_F^2 + \gamma \|\mathbf{J}\|_* + \frac{\rho}{2} \|\mathbf{J} - (\mathbf{Z} + \mathbf{U})\|_F^2. \quad (35)$$

The ADMM algorithm runs the following iterations until the stopping criteria is satisfied:

$$\mathbf{J} \leftarrow \arg\min_{\mathbf{J}} \frac{1}{2} \|\mathbf{J} - (\mathbf{Z} + \mathbf{U})\|_F^2 + \frac{\gamma}{\rho} \|\mathbf{J}\|_*, \quad (36)$$

$$\mathbf{Z} \leftarrow \arg\min_{\mathbf{Z}} \frac{\lambda}{2} \|\mathbf{W} - \mathbf{W}\mathbf{Z} - \mathbf{L}\mathbf{W}\|_F^2 + \frac{\rho}{2} \|\mathbf{J} - (\mathbf{Z} + \mathbf{U})\|_F^2, \quad (37)$$

$$\mathbf{U} \leftarrow \mathbf{U} + \mathbf{Z} - \mathbf{J}. \quad (38)$$

In each iteration, the update in (36) is done by the singular value thresholding algorithm [6]. To solve (37), we directly set $\mathbf{Z}$ to its closed-form solution:

$$\mathbf{Z} = \left(\frac{\lambda}{\rho} \mathbf{W}^\top \mathbf{W} + \mathbf{I}_T \right)^{-1} \left[\mathbf{J} - \mathbf{U} + \frac{\lambda}{\rho} \mathbf{W}^\top (\mathbf{W} - \mathbf{L}\mathbf{W}) \right]. \quad (39)$$

A.2.3 *Updating* $\mathbf{L}$

This is similar to the step on updating $\mathbf{Z}$, we introduce an auxiliary variable $\mathbf{S}$ with the same shape as $\mathbf{L}$, and the augmented Lagrangian with $\mathbf{V}$ as the scaled dual variable and ρ as the penalty parameter is given as the following:

$$\mathcal{L}_\rho(\mathbf{L},\mathbf{S},\mathbf{V})=\frac{\lambda}{2}\|\mathbf{W}-\mathbf{WZ}-\mathbf{LW}\|_F^2+\gamma\|\mathbf{S}\|_*+\frac{\rho}{2}\|\mathbf{S}-(\mathbf{L}+\mathbf{V})\|_F^2. \tag{40}$$

The ADMM algorithm does the following iterations until the stopping criteria is satisfied:

$$\mathbf{S} \leftarrow \arg\min_{\mathbf{S}} \frac{1}{2}\|\mathbf{S}-(\mathbf{L}+\mathbf{V})\|_F^2+\frac{\gamma}{\rho}\|\mathbf{S}\|_*, \tag{41}$$

$$\mathbf{L} \leftarrow \arg\min_{\mathbf{L}} \frac{\lambda}{2}\|\mathbf{W}-\mathbf{WZ}-\mathbf{LW}\|_F^2+\frac{\rho}{2}\|\mathbf{S}-(\mathbf{L}+\mathbf{V})\|_F^2, \tag{42}$$

$$\mathbf{V} \leftarrow \mathbf{V}+\mathbf{L}-\mathbf{S}. \tag{43}$$

Similarly, the problem in (41) is solved by singular value thresholding, while the problem in (42) has the closed-form solution:

$$\mathbf{L}=\left[\frac{\lambda}{\rho}\left(\mathbf{W}-\mathbf{WZ}\right)\mathbf{W}^\top+\mathbf{S}-\mathbf{V}\right]\left(\frac{\lambda}{\rho}\mathbf{W}\mathbf{W}^\top+\mathbf{I}_d\right)^{-1}. \tag{44}$$

A.2.4 *Updating* $\mathbf{W}$

The subproblem w.r.t. $\mathbf{W}$ is:

$$\min_{\mathbf{W}}\sum_{t=1}^{T}\mathcal{L}(\mathcal{D}_t,\mathbf{w}_t)+\frac{\lambda}{2}\|\mathbf{W}-\mathbf{WZ}-\mathbf{LW}\|_F^2+\frac{\beta}{2}\|\mathbf{W}\|_F^2. \tag{45}$$

We propose to use gradient descent to solve this subproblem. Let $\nabla_{\mathbf{W}}\mathcal{L}$ be the gradient of $\sum_{t=1}^{T}\mathcal{L}(\mathcal{D}_t,\mathbf{w}_t)$ w.r.t. $\mathbf{W}$ and $\mathbf{M}=\mathbf{W}-\mathbf{WZ}-\mathbf{LW}$, then the gradient of the whole objective $\mathcal{J}$ w.r.t. $\mathbf{W}$ is given by:

$$\nabla_{\mathbf{W}}\mathcal{J}=\nabla_{\mathbf{W}}\mathcal{L}+\lambda\left[(\mathbf{I}_d-\mathbf{L})^\top\mathbf{M}-\mathbf{MZ}^\top\right]+\beta\mathbf{W}, \tag{46}$$

With certain stepsize η, the update of $\mathbf{W}$ in each step is:

$$\mathbf{W}=\mathbf{W}-\eta\nabla_{\mathbf{W}}\mathcal{J}. \tag{47}$$

The optimization algorithm of **HTEMTL** is provided in Alg. 2. In experiments, we initialize the ADMM penalty ρ to 1, and vary it according to the scheme given by Eq. (3.13) in [5].

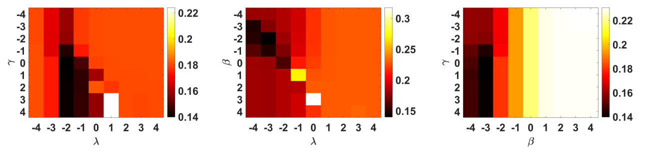
B More on Experiments

B.1 *Statements*

For the comparison methods, it was reported in [35] that GBDSP can always outperform VSTG-MTL [16], so we directly compare our model with GBDSP. The models proposed in [36] and [34] have no public codes, and we actually reproduced [36] by ourselves, however, it cannot provide competitive results. The model proposed in [30] needs extra group information of features, thus we do not treat it as one of our baselines.

Algorithm 2: Pseudocode for **HTEMTL**

```
Input      : $\{\mathcal{D}_t\}_{t=1}^T, \lambda, \gamma, \beta$
Initialize : Initialize $\mathbf{W}$ with STL: $\mathbf{W} \leftarrow \mathbf{W}_{STL}$,
             $\mathbf{Z} \leftarrow 0, \mathbf{L} \leftarrow 0$
while the objective is not converged do
    1. Update Z  : Iteratively solve (36), (37) and (38) until
                   the convergence of the ADMM algorithm.
    2. Update L  : Iteratively solve (41), (42) and (43) until
                   the convergence of the ADMM algorithm.
    3. Update W  : Use gradient descent to solve problem
                   (45).
end
Output     : $\mathbf{W}, \mathbf{Z}, \mathbf{L}$
```



(a) λ and γ (b) λ and β (c) β and γ

Figure 6: Sensitivity analysis on λ, γ and β of **HTEMTL** in terms of ER on the AWA2-Attribute dataset. Values of hyperparameters are shown in the logarithmic scale.

B.2 *Hyperparameter Sensitivity Analysis*

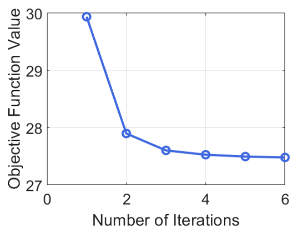
The sensitivity of three regularization parameters, λ, γ and β, of **HTEMTL** is investigated. Values of λ and γ are selected from $\{10^a \mid |a| \in [4]\}$ and the value of β is selected from $\{2\times 10^a \mid |a| \in [4]\}$. In **HTEMTL**, γ controls the low-rankness of the task cluster structure matrix, λ controls the deviation between the learned task parameter $\mathbf{W}$ and the model's assumption, and β controls the model's complexity. Fig. 6 shows the experimental results of **HTEMTL** in terms of ER on the AWA2-Attribute dataset. The subfigure in (a) is shown by fixing $\beta = 10^{-2}$ and the last two subfigures are shown by fixing $\gamma = 1$ and $\lambda = 10^{-2}$, respectively. As shown in Fig. 6, **HTEMTL** achieves its best performance with $\gamma \geq 10^{-1}, \beta \leq 10^{-1}$ and $\lambda \leq 10^{-2}$. Generally, it is recommended to set λ one or two orders of magnitude smaller than γ and set $\beta \leq 10^{-1}$. For **pHTEMTL**, the similar rule applies.

B.3 *Convergence Analysis*

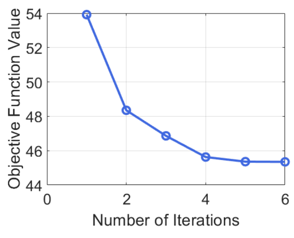
The objective functions of **HTEMTL** and **pHTEMTL** are both nonconvex, and the proposed optimization algorithms in Sec. 5 converge to the local minimum only if in each alternating step the global minimum is reached. Theoretically analyzing the convergence is difficult. As an alternative, we demonstrate the convergence property of **pHTEMTL** empirically by setting $m = 3$ in Fig. 7, and the convergence property of **HTEMTL** is very similar, which is presented in the supplement. When the relative change of its objective value is below to 10^{-3}, we think it is converged. From Fig. 7, we can observe that the objective function of **pHTEMTL** can converge to its local minimum within 10 iterations in most cases.

B.4 *More details on experimental setting*

All the 10 repeats are using different subsets of the data except for the School dataset. For the School dataset, instead of using a hold-



(a) CIFAR-10



(b) AWA2-Attribute

Figure 7: Convergence analysis of **pHTEMTL** by setting $m = 3$ on the CIFAR-10 and AWA2-Attribute datasets.

out validation set, we do 10-fold cross-validation on the training set. In each repeat, 25% of the dataset is randomly drawn as the testing set and the rest as the training set.

For STL and the initialization, the ℓ_2 penalty applied to each task is the same. Actually we find out that even if we allow each task to choose its own penalty parameter via validation, all the tasks tend to choose the same value of the hyper-parameter.

The environment for running all the experiments is: MATLAB R2020b, Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, 32 GB RAM.

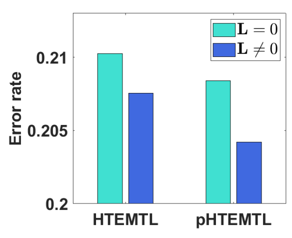
B.5 *Comparing Methods*

We compare our method with six benchmark methods:

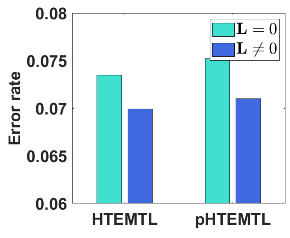
- **STL**: Single-task learning method, in which each task is learned independently with the l_2-norm regularization. The ℓ_2 penalty applied to each task is the same[8].
- **GOMTL** [20]: For task grouping and overlapping, the task parameter matrix is factorized into a product of a shared latent basis and a sparse latent basis assignment matrix.
- **AMTL** [21]: It assumes that each task is represented by a sparse non-negative linear combination of other tasks, and thus information is transferred between tasks in an asymmetric manner.
- **GAMTL** [26]: It assumes that each task is a linear combination of other tasks, and the trace Lasso [12] regularizer is imposed on the task correlation matrix to maintain a task grouping structure.
- **GBDSP** [35]: It adopts the same assumption as GOMTL, but uses a k-block diagonal regularizer to encourage the latent basis matrix to have exactly k blocks, and thus inter-group information transferring is punished. However, it needs the number k of task clusters as prior knowledge.
- **KMSV** [8]: It learns multiple tasks jointly by sharing a low-rank common subspace. Based on tight approximations of rank minimization, a re-weighted iterative algorithm is proposed to minimize k minimal singular values.

For the comparison methods, it was reported in [35] that GBDSP can always outperform VSTG-MTL [16], so we directly compare our model with GBDSP. KMSV is selected as it can outperform several popular MTL methods with low-rankness assumptions [8]. The models proposed in [36] and [34] have no public codes, and we actually reproduced [36] by ourselves, however, it cannot provide competitive results. The model proposed in [30] needs extra group information of features, thus we do not treat it as one of our baselines.

[8] Actually we find out that even if we allow each task to choose its own penalty parameter via validation, all the tasks tend to choose the same value of the hyper-parameter.



(a) Synthetic



(b) Fashion-MNIST

Figure 8: Comparison of the prediction performance in ER between the naive models ($\mathbf{L} = \mathbf{0}$) and the original models ($\mathbf{L} \neq \mathbf{0}$) on the Synthetic and Fashion-MNIST datasets.

B.6 *Evaluation Metrics*

For regression tasks, the two evaluation metrics we adopt are root Mean Squared Error (rMSE) and Mean Absolute Error (MAE), which take the following forms:

$$\text{rMSE} = \sqrt{\frac{\sum_{t=1}^{T} \|\mathbf{y}_t - \mathbf{X}_t^\top \mathbf{w}_t\|_2^2}{\sum_{t=1}^{T} N_t}}, \tag{48}$$

$$\text{MAE} = \frac{\sum_{t=1}^{T} \|\mathbf{y}_t - \mathbf{X}_t^\top \mathbf{w}_t\|_1}{\sum_{t=1}^{T} N_t}. \tag{49}$$

For classification tasks, we use Error Rate (ER) and Area Under ROC-Curve (AUC) to measure the performance. They are defined as the following:

$$\text{ER} = \frac{\sum_{t=1}^{T} \sum_{i=1}^{N_t} \mathbb{I}\left[\text{sign}(\left(\mathbf{X}_t^\top \mathbf{w}_t\right)_i) \neq \mathbf{y}_{ti}\right]}{\sum_{t=1}^{T} N_t}, \tag{50}$$

$$\text{AUC} = \frac{1}{T} \sum_{t=1}^{T} \text{AUC}_t, \tag{51}$$

where AUC_t denotes the AUC score [13] for the t-th task.

B.7 *Effectiveness of Hidden Tasks Enhancement*

In this subsection, we study the effect of hidden tasks. Deactivating the effect of hidden tasks is equivalent to set the matrix $\mathbf{L}$ in (9) and (12) to be zero. In this way, **HTEMTL** degrades to the naive model in (2), and **pHTEMTL** degrades to a naive one that ignores deep hidden tasks. We conduct the same experiments to compare the performance between the original models and their naive variants. The results on the Synthetic and Fashion-MNIST datasets are shown in Fig. 8. We can see that on the two datasets, for both **HTEMTL** and **pHTEMTL**, the introduction of hidden tasks effectively improves the MTL model's generalization performance. This validates our hypothesis that exploiting the effect of hidden tasks can prevent the task grouping procedure from amplifying the deviation of the learned task parameters from the ground-truth values.

B.8 *Case Study on Time Efficiency*

To study the time efficiency of all the comparing methods, we record their training time on the AWA2-Attribute dataset. The result is reported in Table 2. We can see GOMTL can outperform other methods in term of training time efficiency, mainly because of its simplicity. Our proposed method **HTEMTL** performs the best among all the self-epressiveness based methods (AMTL and GAMTL).

Table 2: Training time on the AWA2-Attribute dataset (mean ± std). The best result is highlighted in boldface.

	GOMTL	AMTL	GAMTL	GBDSP	HTEMTL
Training time (s)	**0.27±0.05**	331.02±42.11	26.95±1.37	1.55±0.08	9.50±1.11

B.9 Comparison with neural network

Table 3: Experimental results (mean ± std) with different evaluation metrics. The best two results are highlighted in boldface.

Dataset	Measure	Neural Network	pHTEMTL	HTEMTL
Synthetic	ER↓	0.2940±0.0149	**0.2042±0.0044**	0.2076±0.0128
	AUC↑	0.7874±0.0149	0.8868±0.0099	**0.8906±0.0362**
Fashion-MNIST	ER↓	0.1625±0.0173	0.0710±0.0051	**0.0699±0.0130**
	AUC↑	0.9140±0.0082	**0.9905±0.0019**	0.9865±0.0163
CIFAR-10	ER↓	**0.1777±0.0063**	0.2234±0.0011	0.2284±0.0041
	AUC↑	0.8843±0.0068	**0.8880±0.0083**	0.8878±0.0064
AWA attribute	ER↓	0.2361±0.0098	**0.1316±0.0009**	0.1397±0.0031
	AUC↑	**0.8337±0.0105**	0.7619±0.0194	0.7436±0.0584

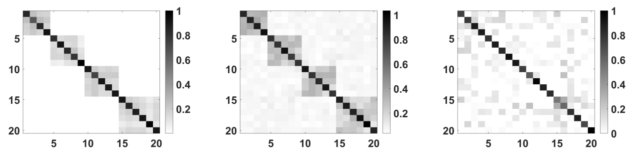
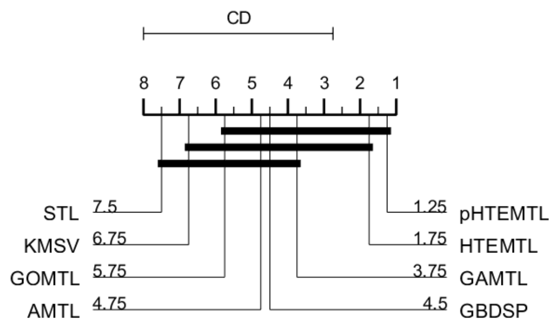

(a) Ground-truth (b) HTEMTL (c) Neural Network

Figure 9: Comparison on the task group structures learned from the synthetic dataset.

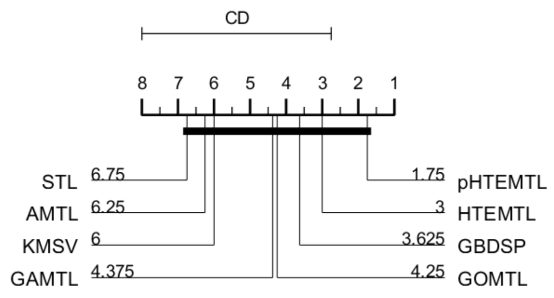
We train a fully connected neural network with three hidden layers and conduct all the classification experiments on it. The hidden layers are shared by all tasks as the common feature representation. The number of neurons in each hidden layer is selected from {64, 128, 256, 512} via validation. The results are reported in Table 3. We can see that in most cases, the three layer neural network can not outperform **HTEMTL** and **pHTEMTL**. We think this is because the assumption that all tasks share the same feature representation is too strong for most real-world data. Another benefit of our proposed method is its interpretability, as shown in Figure 9. The neural network fails to find out the task group structure of the synthetic dataset, while our proposed method can achieve that.

B.10 Statistical test

To perform statistical test on experimental results reported in Table 1 of the main paper, in this subsection we employ Nemenyi test [**?**], that enables to statistically evaluate the performance difference between each pair of comparing methods. In Nemenyi test, the performance of two methods is treated as significantly different if their average ranks differ by at least the critical difference (CD). Fig. 10 shows the CD diagrams for two evaluation metrics (ER and AUC) at 0.05 significance level, based on the results on four classification datasets, including one synthetic dataset and three real-world datasets. In each subfigure, the CD is shown above the axis, where the averaged rank is marked. In Fig. 10, comparing methods which are not significantly different are connected by a thick line. As shown in Fig. 10, **pHTEMTL** and **HTEMTL** ranked 1st and 2nd, respectively. Besides, **pHTEMTL** statistically outperformed STL and KMSV in terms of ER. The other comparing methods achieved statistically comparable performance with each other. The observation shows that hidden tasks enhancement indeed helps to achieve better generalization ability with the learning model, leading to superior prediction accuracy than the baselines. In addition, in order to demonstrate statistical difference between the proposed methods and other comparing methods, more datasets are needed in experiments.



(a) ER



(b) AUC

Figure 10: CD diagrams (at 0.05 significance level) of eight comparing methods on four classification datasets in terms of ER and AUC. The performance of two methods is considered as significantly different if the average ranks differ by at least the Critical Difference (CD). The rank is shown next to the corresponding method.