ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230333

Structured Sparse Multi-Task Learning with Generalized Group Lasso

Luhuan Fei[a;*], Lu Sun[a;**], Mineichi Kudo[b;***] and Keigo Kimura[b;****]

[a]ShanghaiTech University
[b]Hokkaido University

Abstract. Multi-task learning (MTL) improves generalization by sharing information among related tasks. Structured sparsity-inducing regularization has been widely used in MTL to learn interpretable and compact models, especially in high-dimensional settings. These methods have achieved much success in practice, however, there are still some key limitations, such as limited generalization ability due to specific sparse constraints on parameters, usually restricted in matrix form that ignores high-order feature interactions among tasks, and formulated in various forms with different optimization algorithms. Inspired by Generalized Lasso, we propose the Generalized Group Lasso (GenGL) to overcome these limitations. In GenGL, a linear operator is introduced to make it adaptable to diverse sparsity settings, and helps it to handle hierarchical sparsity and multi-component decomposition in general tensor form, leading to enhanced flexibility and expressivity. Based on GenGL, we propose a novel framework for Structured Sparse MTL (SSMTL), that unifies a number of existing MTL methods, and implement its two new variants in shallow and deep architectures, respectively. An efficient optimization algorithm is developed to solve the unified problem, and its effectiveness is validated by synthetic and real-world experiments.

1 Introduction

Multi-task learning (MTL) aims to improve generalization by learning multiple tasks jointly, so that useful knowledge can be transferred among tasks. Nowadays, how to save task correlations in a compact and interpretable model by promoting structured sparsity becomes the key challenge of MTL, especially for the real-world applications with various task specificities and high dimensionality. To tackle this issue, a variety of MTL methods with different sparsity-inducing regularizations are proposed under different scenarios and have achieved great success so far [41].

Existing structured sparse MTL works can be roughly divided into three categories: linear MTL, multilinear MTL and deep MTL. As the most widely used MTL method, linear MTL aims to transfer useful task correlations between linear models based on the individual effects of features. For instance, feature learning approaches [10, 13] seek to learn a sparse feature representation shared by tasks; task clustering approaches [12, 23] learn task group structures by promoting structured sparsity on the weight matrix; decomposition approaches [19, 11] decompose the weight into multiple components to model hierarchical sparse structures. The additive nature of linear MTL makes it fail to handle the case where the task response is correlated with interactions between features. Such high-order feature interactions are common in practice. For example, Parkinson's disease is a result of complicated interactions between environmental factors and genetic factors [17]. In contrast, multilinear MTL learns high-order feature interactions shared by related tasks [16, 13] and represents model weights by a tensor structure. Different from the aforementioned shallow MTL methods, deep MTL uses neural networks to model the complex nonlinear structure of real data. A number of methods with sparsity-inducing regularizations [25, 26, 39] are proposed to capture complex nonlinear relations among tasks in a compact way. For current structured sparse MTL methods, there are three main problems: 1) Each model works under a specific assumption on the structured sparsity of parameters, that limits its generalization ability to tackle various real applications. 2) Existing sparsity-inducing regularizations usually restrict to the matrix form, and thus ignore high-order feature interactions among tasks that are typically represented in a tensor structure. 3) Different models are formulated in different forms, and thus have to use different algorithms to solve the problems.

In order to deal with the above three problems, inspired by Generalized Lasso (GenLA) [29], we propose the **Gen**eralized **G**roup **L**asso (**GenGL**). Specifically, for Problem 1, a linear operator is introduced for GenGL to flexibly define group structures of model parameters, and sparsity is promoted at the inter-group level. It makes GenGL adaptable to diverse sparsity settings, and helps it to handle hierarchical sparsity and multi-component decomposition. For Problem 2, GenGL is further extended to cope with the tensor form, in order to capture high-order relations for both multilinear MTL and deep MTL. Fro Problem 3, a novel MTL method, namely **S**tructured **S**parse **MTL** (**SSMTL**), is proposed based on GenGL, which unifies a number of current MTL methods with structured sparsity in a general framework. We implement two novel variants of SSMTL, and develop an efficient algorithm to optimize the unified problem. Experiments on both synthetic and real-world datasets show the superior performance of SSMTL, compared with state-of-the-art MTL methods. The contributions can be summarized as follows:

- We propose a novel regularization, namely GenGL, that simultaneously handles hierarchical sparsity and multi-component decomposition in general tensor form.
- Based on GenGL, we propose a general framework, namely SSMTL, that unifies several MTL methods, and solve the optimization problem by an efficient algorithm.

* Email: feilh@shanghaitech.edu.cn
** Email: sunlu1@shanghaitech.edu.cn
*** Email: mine@ist.hokudai.ac.jp
**** Email: kimura5@ist.hokudai.ac.jp

- We implement two novel variants of SSMTL in shallow and deep architectures, respectively, and evaluate their effectiveness by experiments on both synthetic and real-world datasets.

2 Related Works

The existing MTL methods can be generally categorized into linear MTL, multilinear MTL and deep MTL. In linear MTL, one way is to learn a more powerful feature representation based on the original feature space. [22] is a naive feature selection method to enforce row-sparsity, while [10] selects features and captures outliers together. As low-rank approaches, KMSV [5] uses a new tight approximations for rank constraints, and MTPL [31] regularizes low-rank matrix factorization via sparse network lasso. Another way is to learn groups among tasks [4]. CCMTL [12] uses a convex clustering algorithm based on the kNN graph. GBDSP [36] learns a generalized block-diagonal structure for the weight matrix. [23] uses sparse network lasso to extract latent task clusters for compositional data. Besides, decomposition approaches such as, rMTFL[10] and MeTaG [11], learn a hierarchical structure to explore task relations. Different from linear MTL, multilinear MTL methods [16, 13, 40] are recently proposed to learn high-order feature interactions based on a consensus latent representation represented in a tensor structure.

In deep MTL, there are two ways to share knowledge across tasks. Hard-sharing approaches [7, 26] use a single network, which forces all tasks to own the same hidden space and allows for modeling task-specificity only in the top layer. To remove redundant parameters across tasks, [25] combines l_1-norm and $l_{2,1}$-norm, while [8] combines Group Lasso and Adaptive Group Lasso; [39] introduces a novel topic-task-element penalty to promote topic-level sparsity; [26] learns a sparse sharing structure by extracting sub-nets from the base network. On the other hand, soft-sharing approaches [18] use a separate network for each task and task relationship is captured by jointly regularizing the weights of these networks, which is usually represented as a tensor by concatenating layer-wise weight matrices from multiple tasks. [35] applies the tensor trace norm to learn cross-task subspace structure, and [40] generalizes the tensor trace norm to capture all the low-rank structures stored in the weight tensor.

Different from the widely used lasso [27], Generalized Lasso (GenLA) [29] imposes the l_1-norm on a linear transformation of the weight vector to obtain a more general sparse structure. A number of well-studied problems can be regarded as special cases of GenLA by using different linear transformations, such as the lasso [27], fused lasso [28], trend filtering [14] and the graph fused lasso [3]. Based on GenLA, [20] proposes an approach for penalized tensor decomposition. [2] extends the GenLA by substituting the l_1-norm by the $l_{2,1}$-norm, which can capture more flexible sparse structure at group level, and unifies the group lasso [37] and the group fused lasso[1]. In addition, some algorithms are proposed to solve the GenLA problem. [29] focuses on solving the dual of GenLA, [9] uses the QR decomposition, while [21] presents a one-layer projection neural network to find the global optimal solutions of GenLA.

Previous structured sparse MTL methods are specifically designed in different scenarios with various formulations, and have to rely on different optimization algorithms, which probably limits their effectiveness and efficiency in real applications. In this paper, SSMTL is proposed based on GenGL, which enables to induce various sparse structures for weights stored in both matrix and tensor forms, leading to improved generalization in various applications.

3 Preliminary

3.1 Notations

For an arbitrary vector $\mathbf{y} \in \mathbb{R}^p$, the ith entry is represented by y_i, and its l_1-norm and l_2-norm are denoted by $\|\mathbf{y}\|_1$ and $\|\mathbf{y}\|_2$, respectively. Similarly, for a matrix $\mathbf{Y} \in \mathbb{R}^{n\times p}$, y_{ij}, $\mathbf{y}_{i:}$ and $\mathbf{y}_{:j}$ denote the (i,j)th entry, the ith row and the jth column, respectively. Let the index set of n elements be $[n] = \{1, 2, ..., n\}$. We denote the Frobenius norm by $\|\mathbf{Y}\|_F = (\sum_{i=1}^n \sum_{j=1}^p y_{ij}^2)^{\frac{1}{2}}$ and the $l_{2,1}$-norm by $\|\mathbf{Y}\|_{2,1} = \sum_{i=1}^p \|\mathbf{y}_{i:}\|_2$. Let $\mathbf{I}_p$ denotes the identity matrix in size of $p \times p$, $\otimes$ is the kronecker product, and $\circ$ denotes the outer product.

The notations for tensor in this paper are adopted by [15]. For a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_N}$, its tensor order is N. The (i,j,k)th element of a third-order tensor $\boldsymbol{\mathcal{A}}$ is denoted by a_{ijk}. The column, row and tube fibers are denoted by $\mathbf{a}_{:jk}$, $\mathbf{a}_{i:k}$ and $\mathbf{a}_{ij:}$, respectively. The horizontal, lateral, and frontal slices are denoted by $\mathbf{A}_{i::}$, $\mathbf{A}_{:i:}$ and $\mathbf{A}_{::k}$, respectively. For the mode-k matricization $\mathbf{A}_{(k)} \in \mathbb{R}^{p_k\times\prod_{j\neq k} p_j}$ of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_N}$, the mode-$k$ fibers are selected to form its columns. The vectorization operation is defined as $vec(\boldsymbol{\mathcal{A}}) = vec(\mathbf{A}_{(1)}) \in \mathbb{R}^{\prod_j p_j}$. The inner product of two same-sized tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_N}$ is denoted by:

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1=1}^{p_1} \sum_{i_2=1}^{p_2} \cdots \sum_{i_N=1}^{p_N} a_{i_1\ldots i_N} b_{i_1\ldots i_N}.$$

The k-mode product of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_N}$ with a matrix $\mathbf{B} \in \mathbb{R}^{m\times p_k}$, denoted by $\boldsymbol{\mathcal{A}} \times_k \mathbf{B}$, is the tensor of size $p_1 \times \ldots \times p_{k-1} \times m \times p_{k+1} \times \ldots \times p_N$, whose element is

$$(\boldsymbol{\mathcal{A}} \times_k \mathbf{B})_{i_1\ldots i_{k-1} i_j i_{k+1}\ldots i_N} = \sum_{i_k=1}^{p_k} a_{i_1\ldots i_N} b_{i_j i_k}.$$

3.2 General MTL Formulations

Given m tasks, the training data is denoted by $\mathcal{T} = \{\mathbf{X}_t, \mathbf{y}_t\}_{t=1}^m$, where $\mathbf{X}_t \in \mathbb{R}^{n_t\times p}$ is associated with n_t samples and $\mathbf{y}_t \in \mathbb{R}^{n_t}$ is the response of the tth task. For linear MTL, let $\mathbf{W} \in \mathbb{R}^{p\times m}$ be the weight matrix, and the response $y_{t,i}$ of the ith sample $\mathbf{x}_{t,i}$ in the tth task is obtained by $y_{t,i} = \langle \mathbf{x}_{t,i}, \mathbf{w}_{:t} \rangle$, where $\mathbf{w}_{:t}$ is the tth column of $\mathbf{W} \in \mathbb{R}^{p\times m}$. For multilinear MTL, let $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_N}$ $(p_N = m)$ be the weight tensor. Take $N = 3$ [16, 13] for instance, $y_{t,i} = \langle \mathbf{x}_{t,i} \circ \mathbf{x}_{t,j}, \mathbf{W}_{::t} \rangle$, where $\mathbf{W}_{::t}$ [1] is the tth slice of $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{p_1\times p_2\times m}$. Hence, let $\mathbf{w} = vec(\mathbf{W})$ or $vec(\boldsymbol{\mathcal{W}})$, the goal of MTL is formulated as:

$$\min_{\mathbf{w}} \; L(\mathbf{w}|\mathcal{T}) + \Omega(\mathbf{w}), \tag{1}$$

where $L(\mathbf{w}|\mathcal{T})$ is a loss function and $\Omega(\mathbf{w})$ is a regularization term.

3.3 Generalized Lasso

As an extension of Lasso [27], Generalized Lasso (GenLA) [29] uses the l_1-norm to induce structured sparsity on a linear transformation of the weight vector $\mathbf{w}$. The regularization of GenLA is

$$\Omega(\mathbf{w}) = \gamma\|\mathbf{D}\mathbf{w}\|_1, \tag{2}$$

[1] In $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{p\times p\times m}$, an arbitrary entry w_{jkt} $(\forall i,j \in [p], \forall t \in [m])$ saves the second-order feature interaction between the jth feature $\mathbf{x}_{t,ij}$ and the kth feature $\mathbf{x}_{t,ik}$ of the ith sample $\mathbf{x}_{t,i}$ in the tth task. Higher-order interactions can be modeled in a similar way.

Table 1: A summary of detailed settings of GenGL for selected MTL methods. For decomposition methods, suppose there are h components and the ith dimension of the lth component is associated with an operator $\boldsymbol{\pi}_{l,i}$. For clarity, when $h \in \mathrm{N}_+$, we show $\Omega(\mathbf{w}_l)$ instead.

Architecture	Method	$\Omega(\mathbf{w})$	$\boldsymbol{\pi}$	h
Shallow	Group Lasso (GL) [37]	$\gamma\|\mathbf{W}\|_{2,1}$	$\boldsymbol{\pi}_1 \in \mathcal{P}_F, \boldsymbol{\pi}_2 \in \mathcal{P}_C$	$h = 1$
	MeTaG [11]	$\frac{\gamma}{\phi^l}\sum_{j<k}\|\mathbf{w}_{l,:j} - \mathbf{w}_{l,:k}\|_2$	$\boldsymbol{\pi}_{l,1} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2} \in \mathcal{P}_{F^2}$	$h \in \mathrm{N}_+$
	SSMTL$_m$ (The proposed model)	$\frac{\gamma_1}{\phi^l}\sum_{j<k}\|\mathbf{w}_{l,:j} - \mathbf{w}_{l,:k}\|_2 + \frac{\gamma_2}{\phi^{-l}}\|\mathbf{W}_l\|_{2,1}$ $+\frac{\gamma_1}{\phi^{-l}}\sum_i\sum_{j<k}\|w_{l,ij} - w_{l,ik}\|^2 + \frac{\gamma_2}{\phi^l}\|\mathbf{w}_l\|_1$	$\boldsymbol{\pi}_{l,1}^{(1)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2}^{(1)} \in \mathcal{P}_{F^2}$ $\boldsymbol{\pi}_{l,1}^{(2)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,2}^{(2)} \in \mathcal{P}_C$ $\boldsymbol{\pi}_{l,1}^{(3)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,2}^{(3)} \in \mathcal{P}_{F^2}$ $\boldsymbol{\pi}_{l,1}^{(4)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,2}^{(4)} \in \mathcal{P}_F$	$h \in \mathrm{N}_+$
Deep	Sparse GL (SGL) [25]	$\frac{\gamma_1}{\phi^l}\sum_k^m\sum_j^{p_2}\|\mathbf{w}_{l,:jk}\|_2 + \frac{\gamma_2}{\phi^l}\sum_{i,j,k}\|\mathbf{w}_{l,ijk}\|$	$\boldsymbol{\pi}_{l,1}^{(1)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2}^{(1)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,3}^{(1)} \in \mathcal{P}_F$ $\boldsymbol{\pi}_{l,1}^{(2)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,2}^{(2)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,3}^{(2)} \in \mathcal{P}_F$	$h \in \mathrm{N}_+$
	GL + Adaptive GL (GLAGL) [8]	$\frac{\gamma}{\phi^l}\sum_k^m\sum_j^{p_2}\|\mathbf{w}_{l,:jk}\|_2 + \sum_k^m\sum_j^{p_2}\frac{\gamma_j}{\phi^l}\|\mathbf{w}_{l,:jk}\|_2$	$\boldsymbol{\pi}_{l,1}^{(1)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2}^{(1)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,3}^{(1)} \in \mathcal{P}_F$ $\boldsymbol{\pi}_{l,1}^{(2)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,2}^{(2)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,3}^{(2)} \in \mathcal{P}_F$	$h \in \mathrm{N}_+$
	SSMTL$_t$ (The proposed model)	$\frac{\gamma_1}{\phi^l}\sum_{j<k}\|\mathbf{W}_{l,::j} - \mathbf{W}_{l,::k}\|_F + \frac{\gamma_2}{\phi^{-l}}\sum_k^m\sum_j^{p_2}\|\mathbf{w}_{l,:jk}\|_2$ $+\frac{\gamma_1}{\phi^{-l}}\sum_i^{p_2}\sum_{j<k}\|\mathbf{w}_{l,:ij} - \mathbf{w}_{l,:ik}\|_2 + \frac{\gamma_2}{\phi^l}\sum_{i,j,k}\|\mathbf{w}_{l,ijk}\|$	$\boldsymbol{\pi}_{l,1}^{(1)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2}^{(1)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,3}^{(1)} \in \mathcal{P}_{F^2}$ $\boldsymbol{\pi}_{l,1}^{(2)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2}^{(2)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,3}^{(2)} \in \mathcal{P}_F$ $\boldsymbol{\pi}_{l,1}^{(3)} \in \mathcal{P}_C, \boldsymbol{\pi}_{l,2}^{(3)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,3}^{(3)} \in \mathcal{P}_{F^2}$ $\boldsymbol{\pi}_{l,1}^{(4)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,2}^{(4)} \in \mathcal{P}_F, \boldsymbol{\pi}_{l,3}^{(4)} \in \mathcal{P}_F$	$h \in \mathrm{N}_+$

where $\mathbf{D}$ is a generalized penalty matrix that transforms $\mathbf{w}$ by its specific setting and $\gamma > 0$ is a hyperparameter. When $\mathbf{D} = \mathbf{I}$, GenLA becomes Lasso. In contrast to GenLA, [2] substitutes the l_1-norm in (2) by the $l_{2,1}$-norm to learn structured sparsity at group level:

$$\Omega(\mathbf{W}) = \gamma\|\mathbf{D}\mathbf{W}\|_{2,1}. \tag{3}$$

When $\mathbf{D} = \mathbf{I}$, it becomes the Group Lasso (GL) [37]; when $\mathbf{D}$ is designed to penalize group differences, it becomes the Group Fused Lasso (GFL) [1].

Existing GenLA-related methods are restricted in indecomposable vector or matrix forms and fail to find hierarchical sparsity. In this paper, we generalize both GenLA [29] and [2], and propose Generalized Group Lasso (GenGL) to promote hierarchical structured sparsity for decomposable weight tensors.

4 Methodology

In this section, we first discuss how to define the GenGL by introducing a linear operator, then represent it in both matrix and tensor forms, and finally propose SSMTL and implement two novel formulations.

4.1 *Formulating Sparse Structures for GenGL*

To flexibly define a group sparse structure, we introduce two extreme equivalence relations as two partitions $\mathcal{P}$ over $[p_i]$:

$$\mathcal{P} = \begin{cases} \mathcal{P}_C = \{[p_i]\} & \text{(coarsest)}, \\ \mathcal{P}_F = \{\{j\}|j \in [p_i]\} & \text{(finest)}. \end{cases} \tag{4}$$

In addition, we consider one more partition over $[p_i] \times [p_i]$:

$$\mathcal{P}_{F^2} = \mathcal{P}_F \times \mathcal{P}_F = \{\{j,k\}|j,k \in [p_i]\} \ \text{(finest in pair)}. \tag{5}$$

The projections to each member of those partitions are realized as sets of *linear operators*[2]:

$$\boldsymbol{\pi}_i \in \begin{cases} \{\mathbf{I}_{p_i}\} & \text{(for } \mathcal{P}_C\text{)}, \\ \{\mathbf{e}_j|j = 1,2,...,p_i\} & \text{(for } \mathcal{P}_F\text{)}, \\ \{\mathbf{e}_j - \mathbf{e}_k|j,k = 1,2,...,p_i\} & \text{(for } \mathcal{P}_{F^2}\text{)}. \end{cases}, \tag{6}$$

[2] Note that more flexibility can be achieved by other choices of $\boldsymbol{\pi}$. For instance, linear trend filtering [14] is implemented by $\boldsymbol{\pi} \in \mathcal{P}_{F^3} = \mathcal{P}_F \times \mathcal{P}_F \times \mathcal{P}_F$.

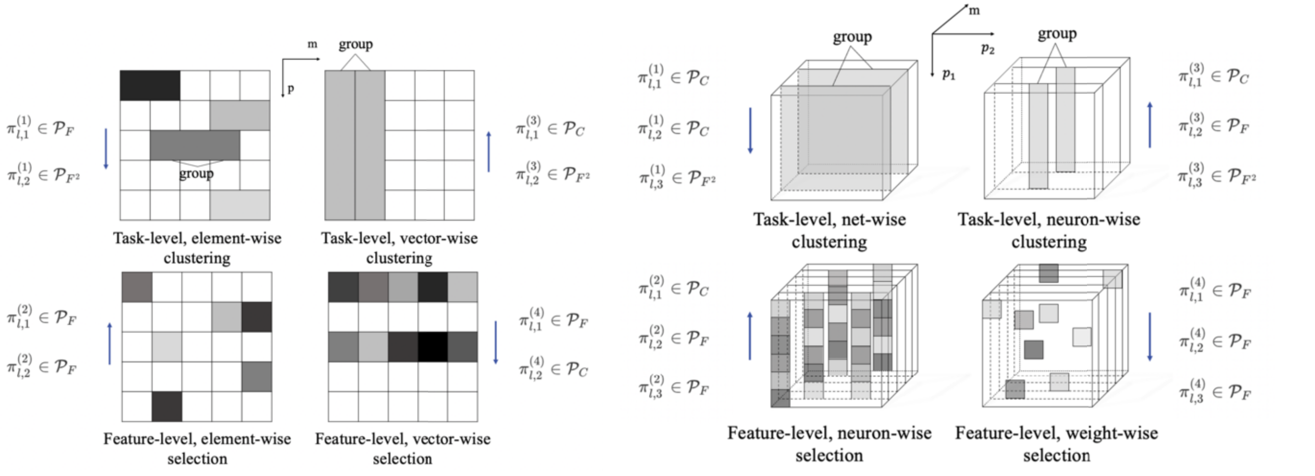
where $\mathbf{e}_j$ is a unit vector in size of p_i with the jth entry being 1. Note that there is a single projection in $\mathcal{P}_C$, p_i projections in $\mathcal{P}_F$, and p_i^2 projections in $\mathcal{P}_{F^2}$. For simplicity, we identify the set of linear operators with the partitions by denoting $\boldsymbol{\pi} \in \mathcal{P}$. Specifically, $\boldsymbol{\pi}_i = \mathbf{e}_j$ leads to a *selection* operation by penalizing $\|\boldsymbol{\mathcal{W}} \times_i \mathbf{e}_j\|_F = \|\boldsymbol{\mathcal{W}}_{:...j...:}\|_F$, that promotes group sparsity along the ith dimension of $\boldsymbol{\mathcal{W}}$. Similarly, $\boldsymbol{\pi}_i = \mathbf{e}_j - \mathbf{e}_k$ leads to a *clustering* operation by penalizing $\|\boldsymbol{\mathcal{W}} \times_i (\mathbf{e}_j - \mathbf{e}_k)\|_F = \|\boldsymbol{\mathcal{W}}_{:...j...:} - \boldsymbol{\mathcal{W}}_{:...k...:}\|_F$, that makes groups along the ith dimension of $\boldsymbol{\mathcal{W}}$ as similar as possible. Table 1 summarizes selected MTL methods, and more details are provided in the supplement.

Based on the settings of linear operators $\boldsymbol{\pi}$, different types of operations can be conducted on $\mathbf{W}$ or $\boldsymbol{\mathcal{W}}$ to encourage various patterns of structured sparsity. We obtain *task-level* or *feature-level* operations once $\boldsymbol{\pi}_i \notin \mathcal{P}_C$ is applied for tasks ($i = N$) or features ($1 \leqslant i \leqslant N-1$). Besides, operations can be distinguished by the dimensionality of groups. Take the matrix for example, $\boldsymbol{\pi}_i \notin \mathcal{P}_C$ on single dimension leads to a *vector-wise* operation, while $\boldsymbol{\pi}_i \notin \mathcal{P}_C$ on both dimensions gives rise to a *element-wise* operation. Similar definitions for *net-wise*, *neuron-wise* and *weight-wise* can be simply derived for the layer-wise weight tensors in soft-sharing networks. Take SGL [25] in Table 1 for instance, a *neuron-wise* selection and a *weight-wise* selection are applied simultaneously at *feature-level* to select both task-common features and task-specific features. We summarize different kinds of operations in the supplement.

4.2 *Representing GenGL in Matrix Form*

Based on the definition of linear operators in Sec. 4.1, we propose the GenGL regularization to capture structured sparsity in matrix-based models. For the index set $[p]$ of features and the index set $[m]$ of tasks, according to the linear operators $\boldsymbol{\pi}$ with combinations of $(\mathcal{P}_1, \mathcal{P}_2) \in \{\mathcal{P}_C, \mathcal{P}_F, \mathcal{P}_{F^2}\}^2$, the regularization term of $\mathbf{W}$ in (1) is reformulated as:

$$\begin{aligned}\Omega(\mathbf{W}) =& \gamma \sum_{\boldsymbol{\pi}_1 \in \mathcal{P}_1} \sum_{\boldsymbol{\pi}_2 \in \mathcal{P}_2} \|\boldsymbol{\pi}_1^T \mathbf{W} \boldsymbol{\pi}_2\|_2 \\ =& \gamma \sum_{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2} \|(\boldsymbol{\pi}_2^T \otimes \boldsymbol{\pi}_1^T)\mathbf{w}\|_2 \\ =& \gamma \sum_{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2} \|\mathbf{D}_{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2}\mathbf{w}\|_2, \end{aligned} \tag{7}$$



(a) $\mathbf{W}_l$ in SSMTL$_m$ (b) $\boldsymbol{\mathcal{W}}_l$ in SSMTL$_t$

Figure 1: Illustration of our specific implementations of SSMTL in shallow architecture (SSMTL$_m$ in (a)) and deep architecture (SSMTL$_t$ in (b)). For each component $\mathbf{W}_l$ ($\boldsymbol{\mathcal{W}}_l$), four types of operations (defined in Sec.4.1 and Table A2 in the supplement) with different $\boldsymbol{\pi}_i$s are performed to achieve hierarchical sparsity. An upward (downward) arrow means the penalty strength of this operation increases (decreases) component by component. White indicates zero values and grey otherwise.

.

where $\mathbf{w} = vec(\mathbf{W})$, and $\mathbf{D}$ is a column concatenation of $\mathbf{D}_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}$, $\forall \boldsymbol{\pi}_1 \in \mathcal{P}_1, \boldsymbol{\pi}_2 \in \mathcal{P}_2$. The second equation in (7) holds due to $vec(\mathbf{AXB}) = (\mathbf{B}^T \otimes \mathbf{A})vec(\mathbf{X})$.

For model decomposition, we decompose $\mathbf{W}$ into h components, either by summation $\mathbf{W} = \sum_{l=1}^h \mathbf{W}_l$ for decomposition approaches [42, 11], or by product $\mathbf{W} = \prod_{l=1}^h \mathbf{W}_l$ for multi-level lasso [19, 32] and deep models[3]. Then we introduce a series of operators $\{\boldsymbol{\pi}_{l,i}\}_{l=1}^h$, and each is associated with a specific component, that helps to obtain:

$$\begin{aligned}\Omega(\mathbf{W}) &= \sum_{l=1}^h \gamma_l \sum_{\boldsymbol{\pi}_{l,1}\in\mathcal{P}_{l,1}} \sum_{\boldsymbol{\pi}_{l,2}\in\mathcal{P}_{l,2}} \|(\boldsymbol{\pi}_{l,2}^T \otimes \boldsymbol{\pi}_{l,1}^T)\mathbf{w}_l\|_2 \\ &= \sum_{l=1}^h \gamma_l \sum_{\boldsymbol{\pi}_{l,1},\boldsymbol{\pi}_{l,2}} \|\mathbf{D}_{\boldsymbol{\pi}_{l,1},\boldsymbol{\pi}_{l,2}}\mathbf{w}_l\|_2 \\ &= \sum_{l=1}^h \gamma_l \|\mathbf{D}_l\mathbf{w}_l\|_{ggl}, \end{aligned} \tag{8}$$

where $\mathbf{w}_l = vec(\mathbf{W}_l)$, $\mathbf{D}_l$ is a column concatenation of $\mathbf{D}_{\boldsymbol{\pi}_{l,1},\boldsymbol{\pi}_{l,2}}$s, $\|\cdot\|_{ggl}$ is a self-defined norm, and $\gamma_l \in \mathbb{R}^+$ controls the component-wise strength of the penalty. In experiments, we set $\gamma_l = \frac{\gamma}{\phi^l}$ to make γ_l adaptable to different components. For example, when $\phi > 1$, stronger sparsity is imposed on the bottom (small l) components than the top (large l) ones. Thanks to the variety of $\mathbf{D}_l$, GenGL enables to promote hierarchical structured sparsity with multi-component decomposition.

4.3 Extending GenGL into Tensor Form

Here we extend GenGL into tensor form, to detect sparse patterns of feature interactions in multilinear MTL models [24, 16, 13] and adapt it to soft-sharing networks [40] and CNN [33]. Given $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_N}$ ($p_N = m$), which is decomposed by $\boldsymbol{\mathcal{W}} = \sum_{l=1}^h \boldsymbol{\mathcal{W}}_l$. For the index sets $[p_i](1 \leqslant i \leqslant N-1)$ of features and the index set $[p_N]$ of tasks, according to the linear operators $\boldsymbol{\pi}$ with combinations of $(\mathcal{P}_{l,i}, \ldots, \mathcal{P}_{l,N}) \in \{\mathcal{P}_C, \mathcal{P}_F, \mathcal{P}_{F^2}\}^N$, GenGL in tensor form is formulated as below:

$$\begin{aligned}\Omega(\mathbf{w}) &= \sum_{l=1}^h \gamma_l \sum_{\boldsymbol{\pi}_{l,1}\in\mathcal{P}_{l,1}} \ldots \sum_{\boldsymbol{\pi}_{l,N}\in\mathcal{P}_{l,N}} \|(\boldsymbol{\pi}_{l,N}^T \otimes \ldots \otimes \boldsymbol{\pi}_{l,1}^T)\mathbf{w}_l\|_2 \\ &= \sum_{l=1}^h \gamma_l \sum_{\boldsymbol{\pi}_{l,1},\ldots,\boldsymbol{\pi}_{l,N}} \|\mathbf{D}_{\boldsymbol{\pi}_{l,1},\ldots,\boldsymbol{\pi}_{l,N}}\mathbf{w}_l\|_2 \\ &= \sum_{l=1}^h \gamma_l \|\mathbf{D}_l\mathbf{w}_l\|_{ggl}, \end{aligned} \tag{9}$$

where $\mathbf{w}_l = vec(\boldsymbol{\mathcal{W}}_l)$ $\mathbf{D}_l$ is a column concatenation of $\mathbf{D}_{\boldsymbol{\pi}_{l,1},\ldots,\boldsymbol{\pi}_{l,N}}$s, $\forall \boldsymbol{\pi}_{l,i} \in \mathcal{P}_{l,i}, i \in [N], l \in [h]$.

The proposed GenGL unifies several sparsity-inducing regularizations for linear, multilinear and deep models, and it can capture structured sparsity at element-wise, vector-wise, etc, or any combinations of them by properly designing $\mathbf{D}_l$s, achieving hierarchical sparsity of high-order interactions stored in multiple components.

4.4 Implementing MTL with GenGL

Based on GenGL in (9), we propose a general formulation for Structured Sparse MTL (SSMTL) [4]:

$$\min_{\mathbf{w}} \; L(\mathbf{w}|\mathcal{T}) + \sum_{l=1}^h \gamma_l \|\mathbf{D}_l\mathbf{w}_l\|_{ggl}. \tag{10}$$

As shown in Table 1, a number of existing structured sparse MTL methods can be unified in (10). To evaluate its effectiveness, we implement two new models of SSMTL: **SSMTL$_m$** in shallow linear architecture and **SSMTL$_t$** in deep nonlinear architecture (soft-sharing networks). For SSMTL$_m$, the weight matrix is decomposed in the sum of h components. For SSMTL$_t$, it is decomposed non-linearly in

[3] Deep models with linear activations. Once non-linear ones are used, product decomposition is nonlinearly mapped layer by layer.

[4] The code and the supplement are provided at: https://github.com/Selina-FEI/ECAI2023_SSMTL.

the product of h layers, and the layer-wise weights are organized as a third-order tensor. For each component (layer), we perform four types of operations to implement *task-level clustering* and *feature-level selection* on hierarchical groups. Specific settings of $\boldsymbol{\pi}$ and the learned structured sparse patterns are shown in Fig. 1. For example, SSMTL$_t$ in Fig. 1(b) promotes feature sparsity in both neuron-wise and weight-wise, and as l increases, $\mathcal{W}_l$ is likely to be more (less) neuron-wise (weight-wise) sparse. Besides, SSMTL$_t$ makes a trade-off between hard-sharing and soft-sharing by adaptively adjusting the strength of task-level clustering. As l increases, task commonality gradually decreases because the strength of task-level net-wise (neuron-wise) clustering decreases (increases).

Notice that SSMTL$_m$ and SSMTL$_t$ are two special implementations, the proposed SSMTL framework can be implemented for multilinear MTL, multi-view MTL and other deep architectures, such as convolutional network[33], recurrent network[38] and transformers[30]. Besides, GenGL can be extended to a more flexible form instead of penalizing l_2-norm of an arbitrary group. For example, non-convex l_p-norm ($0 < p < 1$) can be used to provide a tighter approximation to the ideal l_0-norm than the l_2-norm and lead to stronger sparsity.

5 Optimization

The optimization problem in (10) involves the models with weight decomposition by either product or summation. For the model with product decomposition, like SSMTL$_t$, gradient backpropagation is applied to optimize it. For the model with summation decomposition, like SSMTL$_m$, an iteratively cascade algorithm [43] is utilized. Thus, we focus on developing the algorithm to solve (10) with $h = 1$, and this problem is reformulated by:

$$\min_{\mathbf{w}} \; L(\mathbf{w}|\mathcal{T}) + \gamma \sum_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_N} \|\mathbf{D}_{\boldsymbol{\pi}_1,\ldots\boldsymbol{\pi}_N}\mathbf{w}\|_2. \tag{11}$$

Since the regularization term $\Omega(\mathbf{w})$ is non-smooth and convex, and common loss function $L(\mathbf{w}|\mathcal{T})$ is Lipschitz continuous and smooth, we employ the smoothing proximal gradient (SPG) method [6] to solve (11). According to the definition of dual norm, $\Omega(\mathbf{w})$ can be formulated as:

$$\Omega(\mathbf{w}) = \max_{\boldsymbol{\beta}\in\mathcal{B}} \gamma\boldsymbol{\beta}^T\mathbf{D}\mathbf{w}, \tag{12}$$

where $\boldsymbol{\beta}$ is a column concatenation of $\boldsymbol{\beta}_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_N}$s, and $\boldsymbol{\beta}_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_N}$ is a vector of auxiliary variables corresponding to $\mathbf{D}_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_N}\mathbf{w}$ with the constraint $\mathcal{B} = \{\boldsymbol{\beta}| \|\boldsymbol{\beta}_{\boldsymbol{\pi}_1,\ldots,\boldsymbol{\pi}_N}\|_2 \leqslant 1, \boldsymbol{\pi}_i \in \mathcal{P}_i, i \in [N]\}$. Then we construct a smooth approximation to $\Omega(\mathbf{w})$:

$$f_\mu(\mathbf{w}) = \max_{\boldsymbol{\beta}\in\mathcal{B}}(\gamma\boldsymbol{\beta}^T\mathbf{D}\mathbf{w} - \mu q(\boldsymbol{\beta})), \tag{13}$$

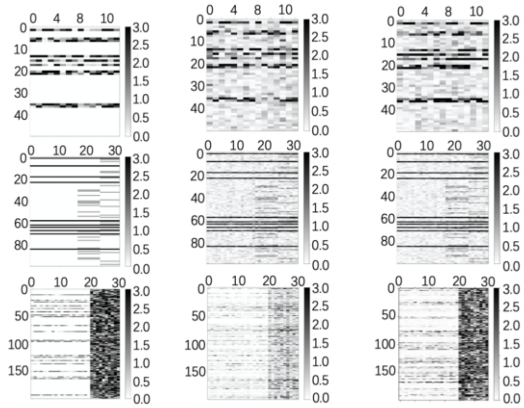
where μ is a positive smoothness parameter and $q(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|_2^2$ is a smoothing function. Instead of solving (11) directly, we focus on solving the following problem:

$$\min_{\mathbf{w}} \; F(\mathbf{w}) = L(\mathbf{w}|\mathcal{T}) + f_\mu(\mathbf{w}). \tag{14}$$

The gradient $\nabla_{\mathbf{w}} F(\mathbf{w})$ of $F(\mathbf{w})$ can be calculated as below, and detailed derivations are provided in the supplement.

$$\nabla_{\mathbf{w}} F(\mathbf{w}) = \nabla_{\mathbf{w}} L(\mathbf{w}|\mathcal{T}) + \gamma\mathbf{D}^T\boldsymbol{\beta}^*, \tag{15}$$

where $\boldsymbol{\beta}^*$ is got by a projection operator $S(\cdot)$. The update rule in each iteration is $\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla_{\mathbf{w}} F(\mathbf{w})$, where η is the learning rate limited by the Lipschitz constant. Note that the per-iteration complexity is linear w.r.t. h, m, $\sum_t n_t$ and $\sum_i |\mathcal{G}_i|$. The pseudocode and the code are given in the supplement.



(a) Ground Truth (b) Standard Method (c) SSMTL$_m$

Figure 2: Visualization for the weight matrices recovered by standard methods (GL, MeTaG and rMTFL) and SSMTL$_m$ on the synthetic dataset. The top row, middle row and bottom row show the results of GL, MeTaG and rMTFL settings, respectively.

6 Experiment

6.1 *Synthetic Experiments*

6.1.1 *Data Generation*

To verify the ability of GenGL to adapt to various structured sparse settings, we generate three synthetic datasets. To generate the first dataset in the GL [37] scenario (row-wise sparsity), we generate the ground truth weight matrix $\mathbf{W}^* \in \mathbb{R}^{p\times m}$ with $p = 80$ and $m = 10$, and randomly select 10 non-zero rows of $\mathbf{W}^*$ to represent the relevant features, whose entries are sampled from the normal distribution $\mathcal{N}(0, 3)$. To generate the second dataset in the MeTaG [11] scenario (hierarchical sparsity of differences of column pairs), the weight matrix is decomposed by $\mathbf{W}^* = \mathbf{W}_1^* + \mathbf{W}_2^*$ with $p = 100$ and $m = 32$. Here $\mathbf{W}_1^*$ assumes that all tasks are in the same group, and thus we randomly select 10 non-zero rows with value 3, and $\mathbf{W}_2^*$ assumes that tasks 17-24 are in a group while tasks 25-32 are in another group. We randomly choose 20 non-zero rows for two column groups of $\mathbf{W}_2^*$, and assign an identical value of 1. To generate the third dataset in the rMTFL [10] scenario (row-wise sparsity + column-wise sparsity), the weight matrix is decomposed by $\mathbf{W}^* = \mathbf{W}_1^* + \mathbf{W}_2^*$ with $p = 200$ and $m = 30$. For $\mathbf{W}_1^*$, we assign the entries of 20 non-zero rows from $\mathcal{N}(0, 3)$ to indicate the relevant features; for $\mathbf{W}_2^*$, we select the last 10 columns as non-zero columns to represent the task outliers, and their entries are sampled from $\mathcal{N}(0, 1)$. In these three scenarios, we generate 50 samples for each task. The prediction rule of the t-th task is made by $\mathbf{y}_t = \mathbf{X}_t\mathbf{w}_{:t} + \boldsymbol{\epsilon}_t$, where each entry of $\mathbf{X}_t$ is sampled from $\mathcal{N}(0, 1)$ and the noise $\boldsymbol{\epsilon}_t$ is sampled from $\mathcal{N}(0, 0.01)$. Fig. 2(a) shows the designed sparse patterns of the scenarios.

6.1.2 *Evaluation of Structured Sparsity Recovery*

Figs. 2(b) and (c) show the restored weight matrices on the synthetic datasets by standard methods (GL, MeTaG and rMTFL) and SSMTL$_m$, respectively. According to Fig. 2, SSMTL$_m$ successfully restores the structures of those weight matrices in all scenarios. GenGL helps to learn sparse patterns consistent with the hypothesis, that makes SSMTL$_m$ adapt to the assumptions in different scenarios and flexibly deal with complex sparsity as well as multi-component

Table 2: Results (mean with std) for shallow MTL on four real-world datasets. The best results are highlighted in boldface.

		Shallow MTL						
Datasets	Metric	Lasso	GL	rMTFL	MeTaG	GBDSP	KMSV	SSMTL$_m$
RF1	nMSE ↓	0.3260(0.0454)	0.3214(0.0465)	0.3205(0.0467)	0.3132(0.0486)	0.3291(0.0393)	0.3248(0.0496)	**0.3115(0.0493)**
	MAE ↓	0.3800(0.0114)	0.3786(0.0110)	0.3779(0.0106)	0.3684(0.0127)	0.3765(0.0105)	0.3803(0.0108)	**0.3681(0.0127)**
	EV ↑	0.6734(0.0454)	0.6785(0.0465)	0.6791(0.0465)	0.6878(0.0492)	0.6709(0.0393)	0.6725(0.0496)	**0.6885(0.0493)**
SARCOS	nMSE ↓	0.1294(0.0108)	0.1291(0.0110)	0.1291(0.0110)	0.1285(0.0113)	0.1302(0.0127)	0.1291(0.0112)	**0.1284(0.0113)**
	MAE ↓	0.2538(0.0070)	0.2527(0.0063)	0.2527(0.0064)	0.2521(0.0066)	0.2541(0.0083)	0.2526(0.0061)	**0.2519(0.0066)**
	EV ↑	0.8676(0.0126)	0.8709(0.0110)	0.8709(0.0110)	0.8712(0.0116)	0.8699(0.0127)	0.8710(0.0112)	**0.8715(0.0113)**
Parkinsons	nMSE ↓	0.9203(0.0309)	0.9012(0.0102)	0.9014(0.0127)	0.8964(0.0096)	0.8962(0.0092)	0.9285(0.0071)	**0.8958(0.0081)**
	MAE ↓	0.7871(0.0158)	0.7832(0.0083)	0.7811(0.0092)	0.7799(0.0135)	0.7797(0.0083)	0.7846(0.0098)	**0.7795(0.0103)**
	EV ↑	0.0587(0.0189	0.0647(0.0073)	0.0655(0.0093)	0.0934(0.0063)	**0.1012(0.0043)**	0.0757(0.0032)	0.0981(0.0052)
Isolet	nMSE ↓	0.4644(0.0153)	0.4642(0.0154)	0.4639(0.0159)	0.4528(0.0139)	0.4550(0.0183)	0.4590(0.0205)	**0.4507(0.0147)**
	MAE ↓	0.5450(0.0105)	0.5446(0.0111)	0.5447(0.0109)	**0.5346(0.0088)**	0.5408(0.0084)	0.5363(0.0132)	0.5347(0.0103)
	EV ↑	0.5357(0.0154)	0.5358(0.0155)	0.5357(0.0154)	**0.5472(0.0140)**	0.5504(0.0182)	0.5410(0.0205)	0.5493(0.0146)

Table 3: The statistics of used real-world datasets.

	Shallow MTL				Deep MTL		
Datasets	RF1	SARCOS	Parkinsons	Isolet	MNIST	COVER	SSD
# of features	64	21	16	671	784	54	48
# of tasks	8	7	42	5	10	7	11
# of samples	9125	48933	5875	7797	70000	581012	58505
Category	Regression				Classification		

decomposition. In contrast, standard methods are limited to handle specific settings, due to their restricted sparse constraints.

6.2 Real-World Experiments

6.2.1 Datasets

We use four regression datasets to conduct experiments for shallow MTL: RF1[5], Isolet[6], Parkinsons[7], and SARCOS[8]. In experiments for deep MTL, three classification datasets are used: SSD[7], MNIST[7] and COVER[7]. The datasets are divided into training set, validation set and testing set in a ratio of 6:2:2. This procedure is repeated ten times, and the mean results and standard deviation are reported. The details of the used real-world datasets are provided in Table 3.

6.2.2 Comparison Methods and Configuration

In experiments for shallow MTL, we compare SSMTL$_m$ with Lasso [27], GL [37], rMTFL [10], MeTaG [11], GBDSP [36] and KMSV [5]. The numbers k and K of latent bases in KMSV and GBDSP are selected from {1, 3, 5, 7, 9}. The factor ϕ in SSMTL$_m$ is selected from {2, 5, 10, 50} and the number h of components is fixed by 2. Other hyperparameters are selected from $\{10^{-3}, 10^{-2}, ..., 10^3\}$. We terminate the algorithm once the relative change of the objective value is below 10^{-4}, and set the maximum number of iterations as 2000. We adopt normalized Mean Squared Error (nMSE), Mean Absolute Error (MAE) and Explained Variance (EV) as metrics.

In experiments for deep MTL, we adopt the same soft-sharing network with independent sub-nets (MLPs) for multiple tasks, where the number h of layers is set as 4. We compare SSMTL$_t$ with Deep Lasso, SGL [25], GL+AGL [8], DMTRL [35] and STG [34]. The factor ϕ in SSMTL$_t$ is selected from {2, 5} and other hyperparameters are selected from $\{10^{-7}, 10^{-6}, ..., 1\}$. The maximum number of iterations is set as 200. For evaluation metrics, we adopt Accuracy and Area Under the Curve (AUC). Detailed information of comparing methods, network settings and metrics are given in the supplement.

[5] https://mulan.sourceforge.net/datasets-mtr.html.
[6] http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html.
[7] https://archive.ics.uci.edu/ml/datasets.php.
[8] http://www.gaussianprocess.org/gpml/data.

6.2.3 Evaluation of Comparing Methods

Table 2 shows the performance results of shallow MTL methods on four regression real-world datasets. From Table 2, SSMTL$_m$ outperforms the other comparing methods in most cases. We summarize two possible reasons to explain its performance advantage: 1) SSMTL$_m$ decomposes the weight into h components, which enables to capture latent multi-level structures that regularized in different strengths by adaptively adjusting γ_l. 2) SSMTL$_m$ extracts *feature-level* and *task-level* information as well as *vector-wise* and *element-wise* information simultaneously, leading to the improved ability to detect complex sparse structures. As a cutting-edge grouped MTL method, MeTaG works the second best in total cases, and achieve the best performance on the Isolet dataset, probably because tasks in the dataset are similar with each other, and its task grouping assumption is well satisfied. Results of statistical test between SSMTL$_m$ and two competitive methods, MeTaG and GBDSP, are reported in the supplement.

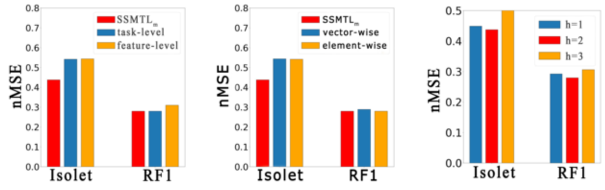
Table 4 shows the performance results of deep MTL methods on three classification real-world datasets. From Table 4, we can see that SSMTL$_t$ achieves the performance advantage in accuracy and AUC on the MNIST and COVER datasets, but fails to outperform STG on the SSD dataset. However, SSMTL$_t$ is able to explore hierarchical and high-order information among tasks and features, that helps to model multi-level task relevance and feature sparsity across different layers, and we clarify this in Sec. 6.2.4. In addition, the performance superiority of SSMTL$_t$ also shows that even though four types of operations are applied simultaneously, over-regularization can be avoided once hyperparameters are properly set.

6.2.4 Ablation Study

To demonstrate effectiveness of different types of operations of GenGL used in SSMTL$_m$, we implement four degenerated variants of SSMTL$_m$ by only considering *feature-level*, *task-level*, *vector-wise* and *element-wise* operations, respectively. Fig. 3(a) and 3(b) show the results in nMSE on two datasets. We also provide the results in MAE and EV in the supplement. The results on other datasets are omitted as similar results are observed. We can see that SSMTL$_m$ outperforms the four variants on both datasets, while the task-level method outperforms the other three variants. It verifies the importance of using appropriate operations in GenGL on improving generalization, and SSMTL$_m$ is flexible enough to extract complex information stored in real-world datasets. In addition, we conduct an experiment on the same datasets by varying $h \in 1, 2, 3$ and report the results in Fig. 3(c). We can see that the model performs the best when $h = 2$. The results in Fig. 3 demonstrate the importance of promoting hierarchical

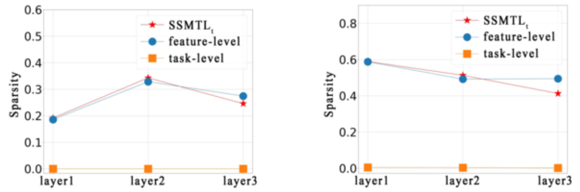
Table 4: Results (mean with std) for deep MTL on three real-world datasets. The best results are highlighted in boldface.

		Deep MTL					
Datasets	Metric	Deep Lasso	SGL	GL+AGL	DMTRL	STG	SSMTL$_t$
MNIST	Accuracy ↑	0.9646(0.0048)	0.9648(0.0043)	0.9621(0.0025)	0.9639(0.0046)	0.9589(0.0020)	**0.9670(0.0041)**
	AUC ↑	0.9928(0.0013)	0.9904(0.0018)	0.9926(0.0010)	0.9926(0.0018)	0.9910(0.0022)	**0.9951(0.0032)**
COVER	Accuracy ↑	0.8867(0.0073)	0.8888(0.0049)	0.8884(0.0065)	0.8849(0.0060)	0.8766(0.0045)	**0.8904(0.0050)**
	AUC ↑	0.9557(0.0027)	0.9542(0.0119)	0.9530(0.0031)	0.9550(0.0040)	0.9433(0.0082)	**0.9566(0.0021)**
SSD	Accuracy ↑	0.9278(0.0059)	0.9255(0.0033)	0.9268(0.0031)	0.9311(0.0022)	**0.9534(0.0018)**	0.9450(0.0020)
	AUC ↑	0.9717(0.0012)	0.9700(0.0032)	0.9725(0.0028)	0.9745(0.0024)	**0.9824(0.0017)**	0.9802(0.0015)



(a) Task/featrue-level (b) Vector/element-wise (c) Decomposition

Figure 3: Effect of different operations and model decomposition of SSMTL$_m$ on the Isolet and RF1 datasets.



(a) Neuron sparsity (b) Weight sparsity

Figure 4: Sparsity of the three hidden layers of SSMTL$_t$ and its variants on the COVER dataset. For each successive layer-pair, neuron (weight) sparsity is the percentage of removed neurons (weights).

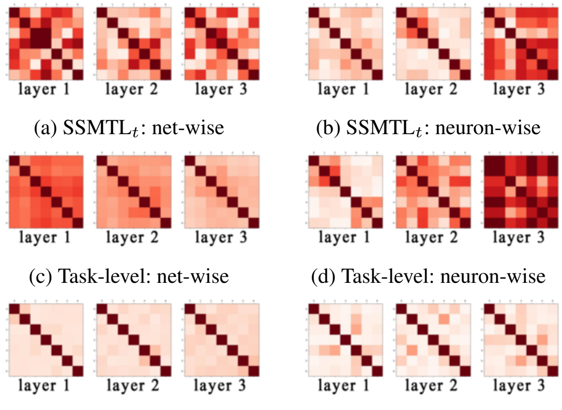
sparsity in model decomposition for SSMTL$_m$.

To investigate the structural characteristics of weights in different layers of SSMTL$_t$, we compare SSMTL$_t$ with its two variants that only consider *feature-level* and *task-level* operations in GenGL, respectively. In the experiments, we set $\gamma_1 = 10^{-6}, \gamma_2 = 10^{-5}$ and $\phi = 2$. Figs. 4 and 5 show the results on the sparsity and task correlations[9] of hidden layers, respectively. For SSMTL$_t$ and the feature-level method, as shown in Fig. 4, neuron sparsity roughly increases while weight sparsity decreases, layer by layer. For SSMTL$_t$ and the task-level method, as shown in Fig. 5, net-wise task correlation increases while neuron-wise task correlation decreases, layer by layer, implying that task correlations are progressively modeled from a strict net-wise to a flexible neuron-wise. Unlike SSMTL$_t$, neither task-level method nor feature-level method can model both neuron/weight-wise sparsity and net/neuron-wise task correlation correctly. Therefore, SSMTL$_t$ can improve its generalization by maintaining a balance between feature sparsity and task correlations.

6.2.5 *Hyperparameter Sensitivity Analysis*

The sensitivity on γ_1, γ_2 and ϕ of SSMTL$_t$ (in Table 1) is investigated on the COVER dataset. In SSMTL$_t$, γ_1 and γ_2 control the regularization strengths of *task-level* and *feature-level* operations, respectively, both of which are selected from $\{10^{-9}, 10^{-8}, ..., 1\}$, and ϕ adjusts the regularization strengths across layers by $\gamma_l = \frac{\gamma}{\phi^l}$,

[9] Pearson correlation coefficient is calculated by $\rho_{\mathbf{X},\mathbf{Y}} = \frac{cov(\mathbf{X},\mathbf{Y})}{\sigma_{\mathbf{X}}\sigma_{\mathbf{Y}}}$.

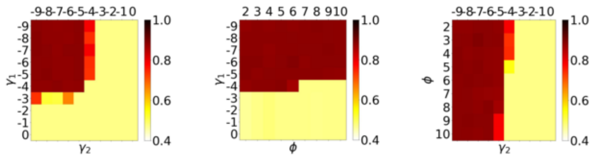


(a) SSMTL$_t$: net-wise (b) SSMTL$_t$: neuron-wise

(c) Task-level: net-wise (d) Task-level: neuron-wise

(e) Feature-level: net-wise (f) Feature-level: neuron-wise

Figure 5: Task correlations of the three hidden layers of SSMTL$_t$ and its variants on the COVER dataset. Each subfigure shows the correlation matrices of layers 1, 2 and 3 from left to right. For each layer, net-wise and neuron-wise correlations are calculated by the whole weights and the weights of a selected neuron, respectively. The warmer the color, the higher the value.



(a) $\phi = 2$ (b) $\gamma_2 = 10^{-6}$ (c) $\gamma_1 = 10^{-6}$

Figure 6: Hyperparameter Sensitivity Analysis on γ_1, γ_2 and ϕ of SSMTL$_t$ on the COVER dataset. The values of γ_1 and γ_2 are shown in the logarithmic scale.

which is selected from $\{2, 3, ..., 10\}$. Fig. 6 shows the result in nMSE. Specifically, Figs. 4(a), 4(b) and 4(c) are shown by fixing $\phi = 2$, $\gamma_2 = 10^{-6}$ and $\gamma_1 = 10^{-6}$, respectively. The result shows that: 1) it is recommended to set $\gamma_1 \leqslant 10^{-4}$ and $\gamma_2 \leqslant 10^{-5}$ on the COVER dataset; 2) ϕ is not as sensitive as other parameters. The hyperparameter sensitivity analysis of SSMTL$_m$ is provided in the supplement.

7 Conclusion

In this paper, we propose a novel SSMTL method based on GenGL. GenGL helps to induce complex hierarchical structured sparsity in multiple components of model parameters, and capture high-order information among features and tasks, leading to enhanced robustness and expressivity. Thanks to the flexibility of GenGL, SSMTL unifies a wide range of structured sparse MTL methods, and its problem is solved by an efficient algorithm. Experiments on both synthetic and real-world datasets demonstrate the effectiveness of SSMTL.

References

[1] Carlos M. Alaíz, Álvaro Barbero, and José R. Dorronsoro, 'Enforcing group structure through the group fused lasso', in *Artificial Neural Networks*, eds., Petia Koprinkova-Hristova, Valeri Mladenov, and Nikola K. Kasabov, pp. 349–371, Cham, (2015).

[2] Carlos M. Alaíz and José R. Dorronsoro, 'The generalized group lasso', in *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, (2015).

[3] S Derin Babacan, Martin Luessi, Rafael Molina, and Aggelos K Katsaggelos, 'Sparse bayesian methods for low-rank matrix estimation', *IEEE Transactions on Signal Processing*, **60**(8), 3964–3977, (2012).

[4] Aviad Barzilai and Koby Crammer, 'Convex Multi-Task Learning by Clustering', in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, eds., Guy Lebanon and S. V. N. Vishwanathan, volume 38 of *Proceedings of Machine Learning Research*, pp. 65–73, San Diego, California, USA, (09–12 May 2015). PMLR.

[5] 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 & Knowledge Management*, pp. 2910–2914, (2021).

[6] Xi Chen, Qihang Lin, Seyoung Kim, Jaime G. Carbonell, and Eric P. Xing, 'Smoothing proximal gradient method for general structured sparse regression', *The Annals of Applied Statistics*, **6**(2), 719 – 752, (2012).

[7] Ronan Collobert and Jason Weston, 'A unified architecture for natural language processing: Deep neural networks with multitask learning', in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, p. 160–167, New York, NY, USA, (2008).

[8] Vu C Dinh and Lam S Ho, 'Consistent feature selection for analytic deep neural networks', in *Advances in Neural Information Processing Systems*, eds., H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, volume 33, pp. 2420–2431, (2020).

[9] Brian R. Gaines, Juhyun Kim, and Hua Zhou, 'Algorithms for fitting the constrained lasso', *Journal of Computational and Graphical Statistics*, **27**(4), 861–871, (2018). PMID: 30618485.

[10] Pinghua Gong, Jieping Ye, and Changshui Zhang, 'Robust multi-task feature learning', in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, p. 895–903, New York, NY, USA, (2012).

[11] Lei Han and Yu Zhang, 'Learning multi-level task groups in multi-task learning', in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, p. 2638–2644. AAAI Press, (2015).

[12] 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**, 3763–3770, (07 2019).

[13] Jun-Yong Jeong and Chi-Hyuck Jun, 'Sparse tensor decomposition for multi-task interaction selection', in *2019 IEEE International Conference on Big Knowledge (ICBK)*, pp. 105–114, (2019).

[14] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky, 'ℓ_1 trend filtering', *SIAM Review*, **51**(2), 339–360, (2009).

[15] Tamara G. Kolda and Brett W. Bader, 'Tensor decompositions and applications', *SIAM Review*, **51**(3), 455–500, (2009).

[16] Kaixiang Lin, Jianpeng Xu, Inci M. Baytas, Shuiwang Ji, and Jiayu Zhou, 'Multi-task feature interaction learning', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, p. 1735–1744, New York, NY, USA, (2016).

[17] Max Little, Patrick Mcsharry, Stephen Roberts, Declan Costello, and Irene Moroz, 'Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection', *Nature Precedings*, 1–1, (2007).

[18] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang, 'Recurrent neural network for text classification with multi-task learning', in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, p. 2873–2879, (2016).

[19] Aurélie C. Lozano and Grzegorz Swirszcz, 'Multi-level lasso for sparse multi-task regression', in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, p. 595–602, Madison, WI, USA, (2012).

[20] Oscar Hernan Madrid-Padilla and James Scott, 'Tensor decomposition with generalized lasso penalties', *Journal of Computational and Graphical Statistics*, **26**(3), 537–546, (2017).

[21] Majid Mohammadi, 'A projection neural network for the generalized lasso', *IEEE Transactions on Neural Networks and Learning Systems*, **31**(6), 2217–2221, (2020).

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

[23] Akira Okazaki and Shuichi Kawano, 'Multi-task learning for compositional data via sparse network lasso', *Entropy*, **24**(12), (2022).

[24] Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil, 'Multilinear multitask learning', in *Proceedings of the 30th International Conference on Machine Learning*, eds., Sanjoy Dasgupta and David McAllester, volume 28 of *Proceedings of Machine Learning Research*, pp. 1444–1452, Atlanta, Georgia, USA, (17–19 Jun 2013). PMLR.

[25] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini, 'Group sparse regularization for deep neural networks', *Neurocomput.*, **241**(C), 81–89, (jun 2017).

[26] Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang, 'Learning sparse sharing architectures for multiple tasks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2020).

[27] Robert Tibshirani, 'Regression shrinkage and selection via the lasso', *Journal of the royal statistical society series b-methodological*, **58**, 267–288, (1996).

[28] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight, 'Sparsity and smoothness via the fused lasso', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **67**(1), 91–108, (2005).

[29] Ryan J. Tibshirani and Jonathan Taylor, 'The solution path of the generalized lasso', *The Annals of Statistics*, **39**(3), (jun 2011).

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, 'Attention is all you need', *Advances in neural information processing systems*, **30**, (2017).

[31] Jiankun Wang and Lu Sun, 'Multi-task personalized learning with sparse network lasso', in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, ed., Lud De Raedt, pp. 3516–3522, (7 2022).

[32] Xin Wang, Jinbo Bi, Shipeng Yu, Jiangwen Sun, and Minghu Song, 'Multiplicative multitask feature learning', *Journal of Machine Learning Research*, **17**(80), 1–33, (2016).

[33] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, 'Learning structured sparsity in deep neural networks', in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, p. 2082–2090, Red Hook, NY, USA, (2016).

[34] Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger, 'Feature selection using stochastic gates', in *International Conference on Machine Learning*, pp. 10648–10659. PMLR, (2020).

[35] Yongxin Yang and Timothy M. Hospedales, 'Deep multi-task representation learning: A tensor factorisation approach', in *International Conference on Learning Representations*, (2017).

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

[37] Ming Yuan and Yi Lin, 'Model selection and estimation in regression with grouped variables', *Journal of the Royal Statistical Society Series B*, **68**, 49–67, (02 2006).

[38] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, 'Recurrent neural network regularization', *arXiv preprint arXiv:1409.2329*, (2014).

[39] Jason (Jiasheng) Zhang and Dongwon Lee, 'Tomato: A topic-wise multi-task sparsity model', in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, CIKM '20, p. 1793–1802, New York, NY, USA, (2020).

[40] Yi Zhang, Yu Zhang, and Wei Wang, 'Multi-task learning via generalized tensor trace norm', in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD21, p. 2254–2262, New York, NY, USA, (2021).

[41] Yu Zhang and Qiang Yang, 'A survey on multi-task learning', *IEEE Transactions on Knowledge and Data Engineering*, **34**(12), 5586–5609, (2022).

[42] Leon Wenliang Zhong and James T. Kwok, 'Convex multitask learning with flexible task clusters', in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, p. 483–490, Madison, WI, USA, (2012). Omnipress.

[43] Alon Zweig and Daphna Weinshall, 'Hierarchical regularization cascade for joint learning', in *International Conference on Machine Learning*, (2013).