# Semisoft Task Clustering for Multi-Task Learning

Zhang Yuzhao and Sun Yifan

Center for Applied Statistics,
Renmin University, Beijing, P. R. China
19980617zyz@gmail.com

**Abstract**

Multi-task learning (MTL) aims to improve the performance of multiple related prediction tasks by leveraging useful information from them. Due to their flexibility and ability to reduce unknown coefficients substantially, the task-clustering-based MTL approaches have attracted considerable attention. Motivated by the idea of semisoft clustering of data, we propose a semisoft task clustering approach, which can simultaneously reveal the task cluster structure for both pure and mixed tasks as well as select the relevant features. The main assumption behind our approach is that each cluster has some pure tasks, and each mixed task can be represented by a linear combination of pure tasks in different clusters. To solve the resulting non-convex constrained optimization problem, we design an efficient three-step algorithm. The experimental results based on synthetic and real-world datasets validate the effectiveness and efficiency of the proposed approach. Finally, we extend the proposed approach to a robust task clustering problem.

## 1 Introduction

The learning of multiple related tasks is often observed in real-world applications, such as computer vision [1], web research [4], bioinformatics [30], and others. Multi-task learning (MTL) is a common machine learning method that is used to solve this problem. To be more specific, MTL learns multiple related tasks simultaneously while exploiting commonalities and differences across the tasks. This can lead to better generalization performance compared to that from learning each task separately. The key challenge in MTL is how to screen the common information across related tasks while preventing information from being shared among unrelated tasks. To tackle this challenge, a multitude of MTL approaches have been proposed in the last decade. Among these approaches, feature selection approaches [23, 36] and task clustering approaches [34, 24] have been the most commonly investigated.

The feature selection approaches aim to select a subset of the original features to serve as the shared features for different tasks, which can be achieved via the regularization [23] or sparse prior [38] of the coefficient matrix. Compared with other methods, the feature selection methods provide better interpretability. However, the assumption that all tasks share identical or similar relevant features is too strong to be valid in many applications. Moreover, information can only be shared in the original feature space, which limits the ability to share information across tasks.

Inspired by the idea of data clustering, the task clustering approaches group tasks into several clusters with similar coefficients. Earlier studies focused on identifying the disjoint task clusters, which can be realized via a two-stage strategy [8], a Gaussian mixture model [34], or regularizations [39, 40]. Such hard clustering may not always reflect the true clustering structure and may result in the inaccurate extraction of information shared among tasks. To this end, recent studies have focused on improvements through allowing partial overlaps between different clusters, such that each task can belong to multiple clusters [16, 19, 15]. This is achieved by decomposing the original coefficient matrix into a product of a matrix consisting of coefficient vectors in different clusters and a matrix containing combination weights.

Due to their flexibility and ability to reduce unknown coefficients substantially, task-clustering approaches have attracted widespread attention in the field of MTL. Although great progress has been made in the task-clustering approaches, the existing approaches still have several limitations. First, few task clustering approaches conduct feature selection, which may affect interpretability and generalizability to some degree. Second, most task-clustering approaches assume that task parameters within a cluster lie in a low dimensional subspace, which we have little information about. As a result, two major issues would arise: the cluster structure could become non-identifiable in a latent space, and the characteristics of individual clusters would be blurred. This would limit their applications in areas such as biomedicine and economics, where the reliability and interpretability of results are especially valued.

Motivated by the idea of the semisoft clustering of data [41, 7], we propose a novel semisoft task clustering MTL (STCMTL) approach, which can simultaneously reveal the task cluster structure for both pure tasks and mixed tasks and select relevant features. A pure task is a task that belongs to a single cluster, while a mixed task is one that belongs to two or more clusters. The key insights of the STCMTL are as follows: (a) pure and mixed tasks co-exist (which is why the term "semisoft" is used); and (b) each mixed task can be represented by a linear combination of pure tasks in different clusters, where the linear combination weights are nonnegative numbers that sum to one (called memberships) and represent the proportions of the mixed task in the corresponding clusters. The idea of representing a task as a linear combination of a set of pure (or representative) tasks has been widely adopted in the literature [40, 33]. However, these approaches cannot remove the irrelevant features. VSTG-MTL [15] is the most relevant method to ours, which can simultaneously perform feature selection and learn an overlapping task cluster structure by encouraging sparsity within the column of the weight matrix. However, this method fails to classify the pure and mixed tasks.

We formulate the semisoft task clustering problem as a constrained optimization problem. A novel and efficient algorithm is developed to solve this complex problem. Specifically, we decompose the original non-convex optimization problem into three sub-problems. We then employ the semisoft clustering with pure cells (SOUP) algorithm [41] that was recently proposed for data semisoft clustering to identify the set of pure tasks and estimate the soft memberships for mixed tasks. Extensive simulation studies have demonstrated the superiority of STCMTL to natural competitors in identifying cluster structure and relevant features, estimating coefficients, and especially, time cost. The effectiveness and efficiency of STCMTL are again validated in the four real-world data sets. In addition, STCMTL can be easily extended to other task clustering problems (e.g., robust task clustering) by replacing the SOUP algorithm with other corresponding algorithms.

In summary, the proposed STCMTL approach has the following advantages:

- It can simultaneously reveal the task cluster structure for both pure and mixed tasks as well as perform feature selection.

- It provides an identifiable and interpretable task cluster structure.

- Compared with its direct competitors, it requires much less calculation time and results in more precise feature selection, and it can be easily extended to other task clustering problems.

The rest of the paper is organized as follows. We first introduce the notations and problem formulation in Section 2. The optimization procedure is presented in Section 3. In Section 4, we campare our method's experimental results with 5 other baselines on synthetic and real-world dataset. We rewrite our algotithm to a general framework and provide an alternatives under the framework in Section 5.

## 2    Formulation

Suppose we have $T$ tasks and $D$ features. In task $i(= 1, 2 \ldots, T)$, there are $n_i$ observations. Denote $\mathbf{X}_i = [(\mathbf{x}_i^1)^\top, \ldots, (\mathbf{x}_i^{n_i})^\top]^\top \in \mathcal{R}^{n_i \times D}$ as the input matrix with $\mathbf{x}_i^j \in \mathcal{R}^{1 \times D}$ and $\mathbf{y}_i = [y_i^1, \ldots, y_i^{n_i}]^\top \in \mathcal{R}^{n_i \times 1}$ as the output vector, where $y_i^j \in \mathcal{R}$ for regression problems and $y_i^j \in \{-1, 1\}$ for binary classification problems. For each task, consider the linear relationship between the input matrix and output vector:

$$\mathbf{y}_i = f(\mathbf{X}_i \mathbf{w}_i), \quad i = 1, 2, \ldots, T,$$

where $f$ is an identify function for regression problems and a logit function for classification problems, and $\mathbf{w}_i \in \mathcal{R}^{D \times 1}$ is the coefficient vector for the $i$th task. Denote $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_T] \in \mathcal{R}^{D \times T}$ as the coefficient matrix to be estimated.

We consider a setting in which $T$ tasks form $K(K \ll \min\{D, T\})$ clusters, and each cluster $k$ has the unique coefficient vector $\mathbf{u}_k \in \mathcal{R}^{D \times 1}$. Different from many task clustering methods that partition tasks into disjoint clusters, we allow for an overlapping

cluster structure in the sense that it allows a task to be a member of multiple clusters. We call the tasks that belong to a single cluster *pure tasks* and the tasks that belong to two or more clusters *mixed tasks*. We assume that the coefficient vector of each task $\mathbf{w}_i$ can be represented by the linear combination of the coefficient vector of the cluster to which this task belongs; that is, $\mathbf{w}_i = \sum_{k=1}^{K} v_{ki}\mathbf{u}_k$, where $v_{ki}$ is the membership weight that represents the proportions of the $i$th task belonging to cluster $k$ with $0 \leq v_{ki} \leq 1$ and $\sum_{k=1}^{K} v_{ki} = 1$. To obtain a more intuitive, and more importantly, identifiable, cluster structure, we assume that each cluster has some pure tasks. Obviously, a pure task in cluster $k$ has $v_{ki} = 1$ and zeros elsewhere. Denote a full rank matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \in \mathcal{R}^{D \times K}$ as the cluster coefficients and $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_T] \in \mathcal{R}^{K \times T}$ as the membership matrix with $i$th column $\mathbf{v}_i = [v_{1i}, \ldots, v_{Ki}]^{\top} \in \mathcal{R}^{K \times 1}$ containing the membership weight of task $i$. The above assumption enables us to write the original coefficient matrix as $\mathbf{W} = \mathbf{U}\mathbf{V}$. Formally, we formulate our approach as a constrained optimization problem:

$$
\begin{aligned}
\min_{\mathbf{U},\mathbf{V}} &\sum_{i=1}^{T} \frac{1}{n_i} L\left(\mathbf{y}_i, \mathbf{X}_i \mathbf{U}\mathbf{v}_i\right) \\
\text{s.t } &\prod_{i=1}^{T}(v_{ki} - 1) = 0, \text{for } k = 1, \ldots, K, \\
&\mathbf{V} \geq 0, \mathbf{V}^{\top}\mathbb{I}_K = \mathbb{I}_T, \\
&\parallel \mathbf{U} \parallel_1 \leq \alpha.
\end{aligned}
\tag{2.1}
$$

where $L(\cdot, \cdot)$ is the empirical loss function, which is a squared loss for a regression problem and a logistic loss for a binary classification problem; the first constraint ensures the existence of pure tasks in each cluster $k$; $\mathbf{V} \geq 0$ means that each element in matrix $\mathbf{V}$ is nonnegative; $\mathbb{I}_m$ represents a $m \times 1$ vector with all elements being 1; and $\parallel \mathbf{U} \parallel_1 = \sum_{k=1}^{K} \parallel \mathbf{u}_k \parallel_1$ is the $l_1$ norm, which encourages the sparsity in the cluster coefficients, and $\alpha$ is the constraint parameter.

## 3    Optimization and Algorithm

Traditional constrained optimization algorithms, such as the interior point method and alternating direction method of multipliers (ADMM) method, are time consuming, and will inevitably involve tedious tuning parameters. Recently, [41] proposed SOUP, a novel semisoft clustering algorithm, which can simultaneously identify the pure and mixed samples. Extensive simulation and real-data analysis demonstrate its advantages over direct competitors. Motivated by the idea of SOUP, we develop a new optimization algorithm to solve the problem in Eq. (2.1). As the standard alternating optimization algorithm, the new algorithm updates the membership matrix $\mathbf{V}$ and cluster coefficient matrix $\mathbf{U}$ alternately. The biggest difference is that the coefficient matrix $\mathbf{W}$ is also updated after updating $\mathbf{V}$ and $\mathbf{U}$. The key ingredient of the algorithm is to adopt SOUP to extract the overlapping cluster structure from the coefficient matrix $\mathbf{W}$ by identifying the set of pure tasks and then estimating the membership matrix $\mathbf{V}$. Hence, $\mathbf{W}$ is required. In addition, to increase the numerical stability and accelerate convergence, we modify each column of $\mathbf{W}$ separately before inputting it into SOUP (see more details in

the follows). In sum, the new algorithm involves a three-step process: updating $\mathbf{V}$, $\mathbf{U}$, and $\mathbf{W}$.

## 3.1   Updating V

We directly apply the SOUP algorithm to the coefficient matrix $\mathbf{W}$ to reveal the cluster structure among tasks. The SOUP algorithm involves two steps: identifying the set of pure tasks and estimating the membership matrix $\mathbf{V}$. In the following, we introduce the main ideas of the two steps.

**Identify the set of pure tasks**. Pure tasks provide valuable information from which to recover their memberships, further guiding the estimation of the membership weights for the mixed tasks. Define a task similarity matrix: $\mathbf{S} = \mathbf{W}^{\top}\mathbf{W} \in \mathcal{R}^{T \times T}$. To find the set of pure tasks, SOUP exploits the special block structure formed by the pure tasks in the similarity matrix $\mathbf{S}$ to calculate a purity score for each task. After sorting the purity scores in descending order, the top $\theta$ percent of tasks are declared as pure tasks, denoted as $\mathcal{P}$, and then partitioned into $K$ clusters by $K$-means algorithm.

**Estimate the membership matrix V**. It is noticed that there is a matrix $\mathbf{A} \in \mathcal{R}^{K \times K}$, such that $\mathbf{V} = \mathbf{A}\boldsymbol{\Theta}$, where $\boldsymbol{\Theta} \in \mathcal{R}^{K \times T}$ is the matrix consisting of the top $K$ eigenvectors of matrix $\mathbf{S}$. Because we have identified the set of pure tasks $\mathcal{P}$ and their memberships $\mathbf{V}_{\mathcal{P}}$, the desired matrix $\mathbf{A}$ can be automatically determined from $\mathbf{V}_{\mathcal{P}} = \mathbf{A}\boldsymbol{\Theta}_{\mathcal{P}}$, where $\boldsymbol{\Theta}_{\mathcal{P}}$ refers to the sub-matrix of matrix $\boldsymbol{\Theta}$ formed by the columns in set $\mathcal{P}$. As a result, the full membership matrix can be recovered: $\mathbf{V} = \mathbf{A}\boldsymbol{\Theta}$.

## 3.2   Updating U

For a fixed membership matrix $\mathbf{V}$, the optimization problem in Eq. (2.1) with respect to the cluster coefficient matrix $\mathbf{U}$, becomes as follows:

$$\min_{\mathbf{U}} \sum_{i=1}^{T} \frac{1}{n_i} L\left(\mathbf{y}_i, \mathbf{X}_i \mathbf{U} \mathbf{v}_i\right)$$
$$\text{s.t } \parallel \mathbf{U} \parallel_1 \leq \alpha.$$

We transform the above constraint problem to the following regularized objective function

$$\sum_{i=1}^{T} \frac{1}{n_i} L\left(\mathbf{y}_i, \mathbf{X}_i \mathbf{U} \mathbf{v}_i\right) + \gamma \parallel \mathbf{U} \parallel_1, \tag{3.1}$$

where $\gamma$ is the regularization parameter. We optimize the objective function with respect to one column vector $\mathbf{u}_k$ at a time and iteratively cycle through all the columns until the Euclidian distances between $\mathbf{u}_k$'s in two adjacent steps converge. Specifically, for $k \in \{1, \ldots, K\}$, we minimize $M(\mathbf{u}_k)$ with respect to $\mathbf{u}_k$, where

$$M(\mathbf{u}_k) = \sum_{i=1}^{T} \frac{1}{n_i} \parallel \mathbf{y}_i - \mathbf{X}_i \sum_{l \neq k} v_{li}\mathbf{u}_l - v_{ki}\mathbf{X}_i\mathbf{u}_k \parallel_2^2 + \gamma \parallel \mathbf{u}_k \parallel_1, \tag{3.2}$$

for linear regression problems with a squared loss, and

$$
\begin{aligned}
M(\mathbf{u}_k) = \sum_{i=1}^{T} \frac{1}{n_i} \sum_{j=1}^{n_i} & \left\{ \log \left[ \exp \left( y_i^j (\mathbf{x}_i^j)^\top \sum_{l \neq k} v_{li} \mathbf{u}_l \right) \right. \right. \\
& \left. + \exp \left( -y_i^j v_{ki} (\mathbf{x}_i^j)^\top \mathbf{u}_k \right) \right] + \log \left[ \exp \left( -y_i^j (\mathbf{x}_i^j)^\top \sum_{l \neq k} v_{li} \mathbf{u}_l \right) \right] \right\} \\
& + \gamma \parallel \mathbf{u}_k \parallel_1,
\end{aligned}
\tag{3.3}
$$

for binary classification problems with a logistic loss. The above problems are solved by using the coordinate descent [9, 35], a well-developed method for tackling a regularized regression model.

## 3.3 Updating **W**

After updating $\mathbf{V}$ and $\mathbf{U}$, we obtain a new coefficient matrix $\tilde{\mathbf{W}} = \mathbf{UV}$. To increase the numerical stability and achieve faster convergence, we update each column of the coefficient matrix $\mathbf{W}$ separately with $\tilde{\mathbf{W}}$ as initial values. Specifically, the $i$th column vector $\mathbf{w}_i$ can be updated by solving the following optimization problem, which optimizes the following objective function

$$
\min_{\mathbf{w}_i} \frac{1}{n_i} L(\mathbf{y}_i, \mathbf{X}_i \mathbf{w}_i) + \lambda_i \parallel \mathbf{w}_i \parallel_1,
\tag{3.4}
$$

where $\lambda_i$ is the regularization parameter, which is fixed during iteration (see more details in the following). We again leverage the coordinate descent algorithm to solve the problem efficiently. Note that, in practice, we apply early stopping to the coordinate descent to update each $\mathbf{w}_i$ with a given number $(2 \sim 5)$ of iterations until completion.

## 3.4 Algorithm

Algorithm 1 summarizes the whole procedure to solve the optimization problem in Eq. (2.1). To start the algorithm, it is important to find a reasonable initial value of $\mathbf{W}$. For this purpose, in each task, we learn a regularized regression or logistic regression coefficient:

$$
\mathbf{w}_i^0 = \min_{\mathbf{w}_i} \frac{1}{n_i} L(\mathbf{y}_i, \mathbf{X}_i \mathbf{w}_i) + \lambda_i \parallel \mathbf{w}_i \parallel_1,
\tag{3.5}
$$

which can be solved with the coordinate descent method. The initial value of $\mathbf{W}$ is given by $\mathbf{W}^0 = [\mathbf{w}_1^0, \ldots, \mathbf{w}_T^0]$.

**Tuning parameters**. SOUP has two tuning parameters: $\epsilon$, the fraction of most neighbors to be examined for each task; and $\theta$, the fraction of tasks that are declared as pure tasks. In practice, SOUP is robust with respect to these two parameters. Following the setup in [41], we set $\theta = 0.5$ and $\epsilon = 0.1$ throughout this study. As for the tuning parameters $\lambda_i$ $(i = 1, \ldots, T)$ in Eqs. (3.4-3.5) and $\gamma$ in Eq. (3.1), cross-validation

---

**Algorithm 1** STCMTL

---

**Input:** Training datasets $\{\mathbf{X}_i, \mathbf{y}_i\}_{i=1}^T$ and number of clusters $K$
**Output:** $\mathbf{U}$ and $\mathbf{V}$

---

1: **Initialization**: $\mathbf{W}^0$;
2: **Repeat**
3:    Update the membership matrix $\mathbf{V}$ by using SOUP;
4:    Update the cluster coefficient matrix $\mathbf{U}$ by minimizing Eq. (3.1);
5:    Update the coefficient matrix $\mathbf{W}$ by solving Eq. (3.4);
6: **Until** the objective function of Eq. (2.1) converges.

---

over a grid search is the commonly adopted method to select the optimal combination of parameters, but this becomes increasingly prohibitive with increases in $T$. Therefore, instead of first performing regularized regression or logistic regression for the given parameters and then searching for the optimal combination of parameters, we propose conducting parameter tuning with cross-validation inside the algorithm. Specifically, we perform the cross-validation within each task at the initialization step. Hence, at the initialization step, we not only provide an initial estimator $\mathbf{W}^0$ but also find the best tuning parameter $\lambda_i$ for each task. The selected $\lambda_i$ are also used at the step of updating $\mathbf{W}$. Then, we set $\gamma$ as the average of $\lambda_i$'s: $\gamma = \frac{1}{T}\sum_{i=1}^T \lambda_i$, where $\lambda_i$ are the parameters determined at the initialization step. As a result, the tuning parameters have been selected inside the algorithm. Since we no longer need to run the algorithm multiple times over a grid space of penalty parameters, we can largely reduce the computational time. The effectiveness and efficiency of the proposed approach to selecting parameters have both been validated in both synthetic and real-world datasets.

## 4   Experiments

This section aims to assess the performance of the proposed STCMTL approach on both synthetic and real-world datasets. We compare STCMTL with the following baseline methods:

**LASSO** [27]: The single-task learning method learns a sparse prediction model by applying LASSO for each task separately.

**CMTL** [39]: This MTL method learns a disjoint cluster structure among the tasks and does not conduct feature selection.

**Trace** [24]: This MTL method learns a low-rank structure among the tasks by penalizing both the Frobenius norm and trace norm of the coefficient matrix.

**FLARCC** [25]: This MTL method achieves feature selection and coefficient clustering across tasks by using LASSO and fused LASSO penalties, respectively.

**VSTG-MTL** [15]: This MTL method is the first method that simultaneously performs feature selection and learns an overlapping cluster structure among tasks using a low-rank approach.

Table 1: STCMTL and baseline methods

|          | Feature selection | Task clustering | Prediction |
|----------|:-----------------:|:---------------:|:----------:|
| STCMTL   | √                 | √               | √          |
| LASSO    | √                 |                 | √          |
| CMTL     |                   | √               | √          |
| Trace    |                   |                 | √          |
| FLARCC   | √                 | √               | √          |
| VSTG -MTL| √                 | √               | √          |

These methods differ in their ability to perform feature selection, task clustering, and prediction, as shown in Table 1. LASSO is implemented using the R package "glmnet", and CMTL and Trace using the R package "RMTL". The implementation of VSTG-MTL was released at https://github.com/ JunYongJeong/VSTG-MTL. STCMTL is implemented in a R package and is available at https://github.com/ RUCyuzhao/STCMTL. Suggested by [15], for VSTG-MTL, the penalty parameter $\mu$ of $k$-support norm is set equal to the penalty parameter $\gamma_1$ of $l_1$ norm, which reduces the computational burden. The hyper-parameters of all methods are selected via five-fold cross validation over a range of $\{2^{-15}, \cdots, 2^3\}$. The number of clusters for STCMTL, CMTL, and VSTG-MTLare selected from the search grid $\{2, 3, ..., 9\}$.

We evaluate the prediction performance by using the root mean squared error (**RMSE**) for the regression problem and the error rate (**ER**) for the classification problem. For the synthetic dataset, we also evaluate the estimation and feature selection performance. Specifically, the estimation is quantified by using the root mean estimation error (**REE**), which is defined as $||\mathbf{W} - \hat{\mathbf{W}}||_F / \sqrt{T}$, whereas feature selection is quantified by using Matthew's Correlation Coefficient ( **MCC**), which is defined as:

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FN})(\text{TP} + \text{FP})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}.$$

## 4.1  Synthetic Datasets

We generate 60 tasks ($T = 60$). For the $i$th task, 100 training observations ($n_i = 100$) and 100 testing observations are generated from $\mathbf{X}_i \sim \mathcal{N}(0, \mathbf{I}_d)$ and $\mathbf{y}_i = \mathbf{X}_i\mathbf{w}_i + \mathcal{N}(0, 0.5\mathbf{I}_{n_i})$. To simulate the high-dimensional scenario ($D > n_i$), which is more challenging than the low-dimensional one, we set $D = 200$ and 600. The 60 tasks are divided into five overlapping clusters ($K = 5$), that is, the true coefficient matrix $\mathbf{W} = \mathbf{UV}$, where $\mathbf{U} \in \mathcal{R}^{D \times K}$ and $\mathbf{V} \in \mathcal{R}^{K \times T}$. For each $\mathbf{W}$ we repeat the data generation process for 10 times and the average performances on the test sets are reported

In the $k$th column of the cluster coefficient matrix $\mathbf{U}$, only the $[5(k-1)+1]$th to the $[5(k+1)]$th components are non-zero, which are generated from two uniform distributions, $\mathcal{U}[-0.5, -0.1]$ and $\mathcal{U}[0.1, 0.5]$, randomly and independently; the other components are zeros. The 60 tasks consist of 50 pure tasks, equally assigned to $K$ clusters, and 10 mixed tasks. Without a loss of generality, we set the first 50 tasks as pure tasks and the last 10 as mixed tasks. As a result, for $i = 1, \ldots, 50$, the membership vectors

Table 2: Results for the synthetic datasets: RMSE, REE, and MCC based on 10 repetitions. Each cell shows the mean $\pm$ s.d.

| | | | LASSO | CMTL | Trace | FLARCC | VSTG-MTL | STCMTL |
|---|---|---|---|---|---|---|---|---|
| $D$=200 | Sparse | RMSE | 0.659±0.015 | 0.651±0.007 | 0.911±0.001 | 0.591±0.011 | 0.543±0.005 | **0.510±0.005** |
| | | REE | 0.030±0.001 | 0.029±0.006 | 0.054±0.001 | 0.019±0.001 | 0.011±0.001 | **0.007±0.001** |
| | | MCC | 0.481±0.021 | 0.109±0.002 | 0.071±0.003 | 0.583±0.023 | 0.705±0.044 | **0.779±0.018** |
| | Dense | RMSE | 0.665±0.016 | 0.650±0.008 | 0.897±0.010 | 0.590±0.011 | **0.515±0.007** | **0.515±0.008** |
| | | REE | 0.031±0.001 | 0.029±0.006 | 0.053±0.004 | 0.019±0.001 | **0.009±0.001** | **0.009±0.001** |
| | | MCC | 0.470±0.018 | 0.111±0.003 | 0.071±0.002 | 0.613±0.080 | 0.722±0.025 | **0.783±0.024** |
| $D$=600 | Sparse | RMSE | 0.743±0.015 | 0.829±0.011 | 1.038±0.013 | 0.635±0.019 | 0.525±0.010 | **0.516±0.005** |
| | | REE | 0.022±0.000 | 0.027±0.000 | 0.036±0.000 | 0.016±0.000 | **0.006±0.001** | 0.010±0.002 |
| | | MCC | 0.400±0.010 | 0.083±0.002 | 0.070±0.004 | 0.660±0.011 | 0.477±0.070 | **0.715±0.157** |
| | Dense | RMSE | 0.734±0.015 | 0.824±0.011 | 1.017±0.015 | 0.624±0.036 | **0.525±0.010** | 0.526±0.006 |
| | | REE | 0.022±0.000 | 0.026±0.000 | 0.036±0.000 | 0.018±0.000 | 0.010±0.001 | **0.005±0.001** |
| | | MCC | 0.386±0.013 | 0.088±0.002 | 0.073±0.006 | 0.643±0.059 | 0.542±0.084 | **0.717±0.159** |

$\mathbf{v}_i$ only have one component equal to 1, while the others are all equal to zero. For the last 10 mixed tasks, we consider two types of mixing patterns: (a) sparse, where the membership vectors $\mathbf{v}_i$ ($i = 51, \ldots, 60$) have positive values for only two components, which are taken randomly from $K$ components; and (b) dense, where the membership vectors $\mathbf{v}_i$ ($i = 51, \ldots, 60$) have positive values on all $K$ components. All membership vectors are normalized such that they sum to 1.
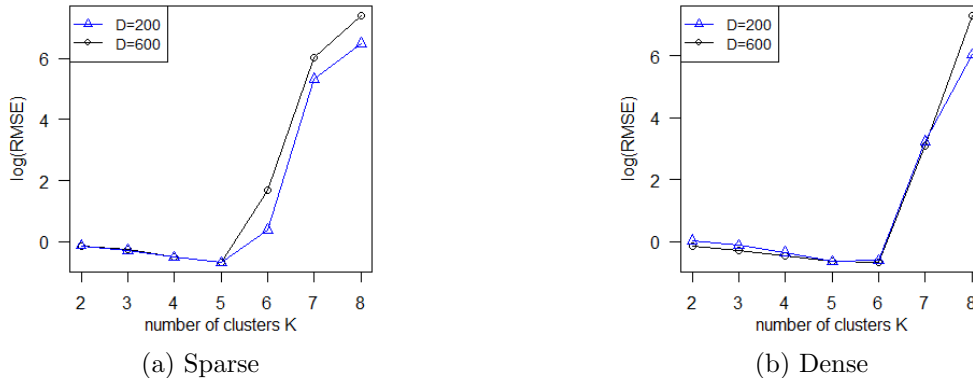


(a) Sparse　　　　　　　　　　　　　　(b) Dense

Figure 1: Results for the synthetic datasets: logarithm of RMSE as a function of the number of clusters $K$.

With the proposed STCMTL approach, we first examine the effect of the number of clusters $K$ on RMSE. Fig. 1 presents the logarithm of RMSE as a function of $K$ for a random replicate, where other hyper-parameters are selected by cross-validation inside the algorithm. RMSE reaches the minimum at the true number of clusters, which is five for the synthetic datasets. We also examine a few other replicates and observe similar patterns.

We then compute the summary statistics based on 10 repetitions. Table 2 presents the simulation results for the synthetic datasets. The values after $\pm$ are the standard
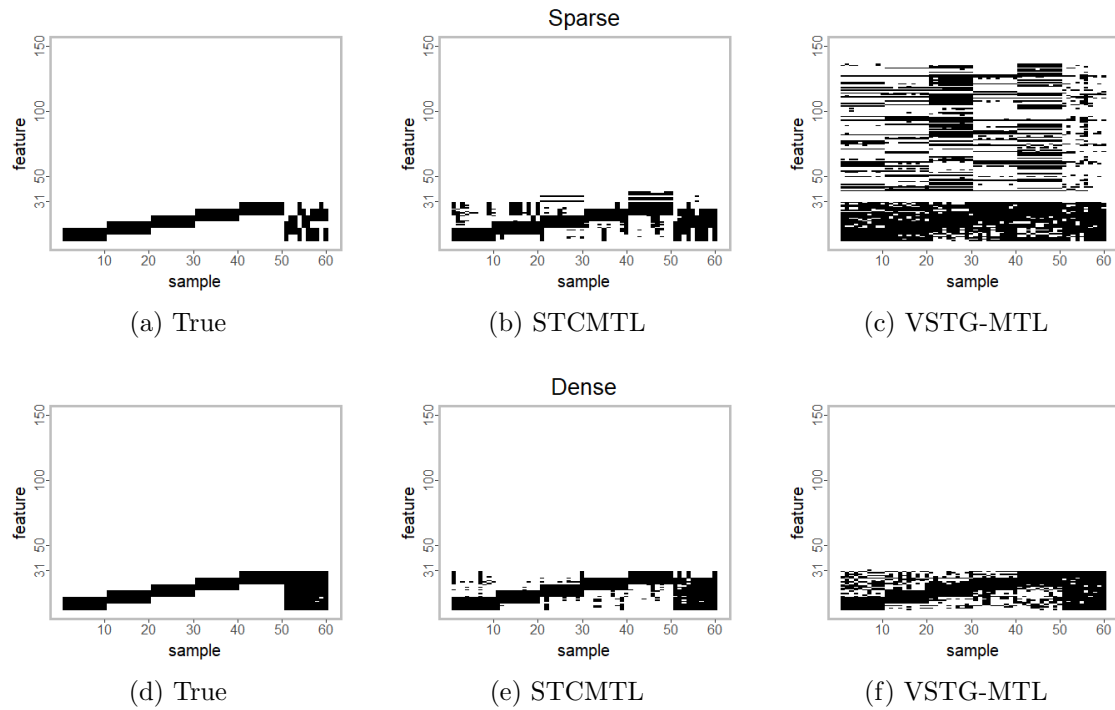
Figure 2: Results for the synthetic datasets: true and estimated coefficient matrices by STCMTL and VSTG-MTL. The dark- and white-colored entries indicate the nonzero and zero values, respectively.

deviations of the corresponding metrics values. In general, STCMTL outperforms the five baseline methods in terms of the three evaluation metrics in the majority of scenarios. STCMTL and VSTG-MTL perform significantly better than other methods. This is expected, as LASSO does not consider the task heterogeneity, CMTL and Trace do not conduct feature selection, and FLARCC clusters tasks based on each coefficient separately. For prediction, STCMTL always have the favorable performance, especially in Sparse condition. In terms of estimation, STCMTL has the best or nearly the best performance. Regarding the accuracy of feature selection, STCMTL performs significantly better than the baseline methods. VSTG-MTL provide a passable accuracy when $D = 200$ but it clearly suffers when $D$ becomes large. The selection of correct features is crucial in some areas, such as biomedicine, which values the interpretability of results. VSTG-MTL, while it provides comparable prediction and estimation results, performs poorly in feature selection. Fig. 2 shows the true coefficient matrix and estimated coefficient matrix by using STCMTL and VSTG-MTL with $D = 600$, where the dark- and white-colored entries indicate non-zero and zero values, respectively [1]. It can be seen that STCMTL can almost identify the true cluster structure among the tasks and remove most of the irrelevant features. However, VSTG-MTL suffers from selecting more false features, which inevitably affects the identification accuracy of the cluster structure.

As VSTG-MTL is the method that is most relevant to the proposed STCMTL approach, we also compare the computational efficiencies of the two methods under all simulation settings. Table 3 shows the time spent in updating $\mathbf{U}$ and $\mathbf{V}$ with the fixed hyper-parameters, and completing the entire training process (including performing five-fold cross-validation), respectively, and their standard deviations for two approaches over 10 repetitions. Clearly, the computational efficiency of STCMTL is significantly higher than that of VSTG-MTL. The reasons for this are summarized as follows:

- STCMTL only contains a $\ell_1$ norm penalty term on $\mathbf{U}$, and thus, uses the coordinate descent algorithm, a highly efficient algorithm, to update $\mathbf{U}$. Meanwhile VSTG-MTL adopts the ADMM algorithm, a more complex algorithm, to address two penalty terms on $\mathbf{U}$. In particular, the time that VSTG-MTL spends on the updating process $\mathbf{U}$ grows significantly when the number of features $D$ increases from 200 to 600.

- As described in Section 3.4, STCMTL conducts the tuning of penalty parameters with cross-validation only at the initialization step, avoiding running the entire algorithm multiple times on the grid space of the penalty parameters. Hence, considerable time is saved.

In summary, from the synthetic datasets, we can see that STCMTL achieves the most desirable performance in terms of prediction, estimation, and feature selection compared to the five baseline methods. While VSTG-MTL is competitive in some scenarios, it suffers severely from poor feature selection and a prohibitive computational cost compared with STCMTL.

---

[1]For demonstration purpose, we reorder the features and put the features selected by STCMTL or VSTG-MTL on the top of list.

Table 3: Results for the synthetic datasets: time consumption (in seconds) of STCMTL and VSTG-MTL based on 10 repetitions. Each cell shows the mean ± s.d.

|  |  |  | Updating **U** | Updating **V** | Total |
|---|---|---|---|---|---|
| $D$=200 | Sparse | VSTG-MTL | 13.0±2.3 | **0.8±0.1** | 4448.7 ± 308.2 |
|  |  | STCMTL | **5.1±0.8** | 0.9±0.2 | **56.9±7.2** |
|  | Dense | VSTG-MTL | 12.1±1.9 | **0.8±0.1** | 4532.9±191.6 |
|  |  | STCMTL | **5.2± 0.6** | 0.9 ± 0.1 | **44.1±3.6** |
| $D$=600 | Sparse | VSTG-MTL | 141.2 ± 16.8 | **1.1 ± 0.2** | 39020.2 ± 1522.6 |
|  |  | STCMTL | **48.3±3.8** | 2.1 ±0.1 | **190.2 ± 20.8** |
|  | Dense | VSTG-MTL | 128.6 ± 11.5 | **1.0 ± 0.1** | 39169.5 ± 1921.5 |
|  |  | STCMTL | **45.3± 3.9** | 2.9 ±0.2 | **191.4±20.2** |

## 4.2    Real-World Datasets

We evaluate the performance of STCMTL on the following four real-world benchmark datasets. **Isolet data**[2] and **School data**[3] are two regression tasks datasets, which are widely used in previous works [[21, 15, 33]]. Training and test sets are obtained by splitting the dataset 75%-25%. **MNIST data**[4] and **USPS data**[5] are both handwritten 10 digits datasets. We treat these 10-class classification as a multi-task learning problem where each task is the classification of one digit from all the other digits, thus yielding 10 tasks. Following the procedure in [[16, 39, 15]], each image is preprocessed with PCA, and the dimensionality is reduced to 64 and 87, respectively. For each task, the training (test) set is generated by randomly selecting 100 train (test) observations of the corresponding digit and 100 train (test) observations of other digits.

We apply STCMTL to the four real datasets. Fig. 3 shows the objective function value by iteration, and we can see that the objective function converges to the optimal value within 20 iterations in most repetitions.



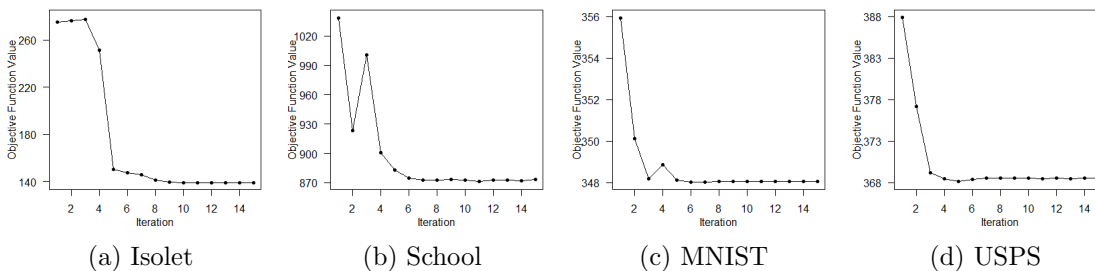|  |  |  |  |
|---|---|---|---|
| (a) Isolet | (b) School | (c) MNIST | (d) USPS |

Figure 3: Objective function value versus iteration on four real-world datasets.

Table 4 summarizes the prediction performance on the test sets of the five methods in the four real-world datasets over 10 repetitions. In the linear regression problem,

---

[2]www.zjucadcg.cn/dengcai/Data/data.html
[3]http://ttic.uchicago.edu/ argyriou/code/index.html
[4]http://yann.lecun.com/exdb/mnist/
[5]http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

Table 4: Results for the real-world datasets: RMSE and ER based on 10 repetitions. Each cell shows the mean ± s.d.

| Dataset | | LASSO | CMTL | Trace | FLARCC | VSTG-MTL | STCMTL |
|---|---|---|---|---|---|---|---|
| Isolet | RMSE | 0.241±0.005 | 0.273±0.003 | 0.237±0.003 | 0.262±0.006 | 0.193±0.018 | **0.191±0.003** |
| School | | 10.638±0.031 | 10.564±0.056 | 12.210±0.053 | 10.771±0.059 | **10.020±0.189** | 10.075±0.107 |
| MNIST | ER | 0.202±0.013 | 0.187±0.007 | 0.165±0.007 | 0.190±0.009 | 0.108±0.012 | **0.088±0.008** |
| USPS | | 0.196±0.011 | 0.199±0.010 | 0.170±0.009 | 0.184±0.012 | 0.101±0.009 | **0.091±0.008** |

Table 5: Results for the real-world datasets: time consumption (in seconds) of STCMTL and VSTG-MTL based on 10 repetitions. Each cell shows the mean ± s.d.

| | Isolet | | School | | MNIST | | USPS | |
|---|---|---|---|---|---|---|---|---|
| | VSTG-MTL | STCMTL | VSTG-MTL | STCMTL | VSTG-MTL | STCMTL | VSTG-MTL | STCMTL |
| Total | 50029.8±3657.7 | **525.3±35.9** | 934.1±95.6 | **20.8±2.6** | 2514.4±99.4 | **209.3±30.2** | 3077.8±154.3 | **121.1±11.1** |

STCMTL and VSTG-MTL stand out. The two methods' results are comparable, with no statistically significant differences. However, in the classification problem, the STCMTL outperforms all the other baseline methods. In addition, once again, the computational cost of VSTG-MTL is prohibitive compared to STCMTL (Table 5).

## 5   Extension to Robust Task Clustering

The proposed approach decouples task clustering and feature selection into estimating three tractable sub-problems: the membership matrix $\mathbf{V}$, cluster coefficients matrix $\mathbf{U}$, and coefficient matrix $\mathbf{W}$. The proposed approach employs the SOUP algorithm to identify the membership matrix but it is not the only available option, as numerous other matrix-factorization-based clustering methods have also been proposed. Hence, in practice, we could replace or combine SOUP with other matrix-factorization-based clustering methods to tackle more complex scenarios in task clustering. In this section, we consider task clustering with several outlier tasks and modify the proposed approach to a robust task clustering one.

We assume that there are some outlier tasks that cannot be represented by a linear combination of cluster coefficient vectors $\mathbf{v}_k$'s. To improve the robustness against these outlier tasks, in the step of updating $\mathbf{V}$, we first apply a $l_{12}$-norm-based robust non-negative matrix factorization (NMF) algorithm to identify the outlier tasks. Then, we estimate $\mathbf{V}$ by using the SOUP algorithm. Specifically, we apply the robust NMF algorithm [18] on $\mathbf{W}$, yielding two rank-$K$ matrices, $\mathbf{C}$, and $\mathbf{\Theta}$, such that $\mathbf{W} \approx \mathbf{C}\mathbf{\Theta}$. Then, we define a distance $D_i = \|\mathbf{W}_i - \mathbf{C}\mathbf{\Theta}_i\|_2$ for each task, which measures the discrepancy between the true and estimated coefficients. If $\mathbf{W}$ is precise enough, the distances of the outlier tasks should be much larger than those of the normal tasks; thus, the outlier tasks can be identified. For simplicity, we employ the definition of outliers in bar plots. In words, a task is declared an outlier if it is above 4.5 times the interquartile range (IQR) over the upper quartile of the distance. Practically, we can use other advanced

---

**Algorithm 2** Robust STCMTL

---

**Input:** Training datasets $\{\mathbf{X}_i, \mathbf{y}_i\}_{i=1}^T$, number of clusters $K$ and index of $T$ tasks $\mathcal{T}$;
**Output:** matrix $\mathbf{U}$, $\mathbf{V}$ and the set of outlier tasks $\Gamma$

---

1: **Initialization**: $\mathbf{W}^0$, $\Gamma = \emptyset$;
2: **Repeat**
3:   Apply the robust NMF algorithm on $\mathbf{W}_{\mathcal{T}\backslash\Gamma}$, yielding two rank-$K$ matrices $\mathbf{C}$ and $\mathbf{\Theta}$, such that $\mathbf{W}_{\mathcal{T}\backslash\Gamma} \approx \mathbf{C\Theta}$;
4:   For each task $i \in \mathcal{T} \backslash \Gamma$, compute its distance: $D_i = \|\mathbf{W}_i - \mathbf{C\Theta}_i\|_2$;
5:   Update $\Gamma$ by: $\Gamma \leftarrow \Gamma \cup \{i : D_i \geq$ the upper quartile of $\{D_i\} + 4.5 \times \text{IQR}\}$;
6:   Update the cluster coefficient matrix $\mathbf{V}_{\mathcal{T}\backslash\Gamma}$ by using SOUP;
7:   Update the cluster coefficient matrix $\mathbf{U}$ by minimizing Eq. (3.1);
8:   Update the coefficient matrix $\mathbf{W}_{\mathcal{T}\backslash\Gamma}$ by solving Eq. (3.4);
9: **Until** the objective function of Eq. (2.1) with tasks in $\Gamma$ removed converges.

---

Table 6: Results for the synthetic datasets with outlier tasks: RMSE, REE, and MCC based on 10 repetitions. Each cell shows the mean $\pm$ s.d.

|         |      | LASSO             | VSTG-MTL          | STCMTL (Robust)       |
|---------|------|-------------------|-------------------|-----------------------|
|         | RMSE | 0.836±0.014       | 0.884±0.046       | **0.577±0.029**       |
| $D$=100 | REE  | 0.067±0.002       | 0.073±0.003       | **0.029±0.006**       |
|         | MCC  | 0.413±0.012       | 0.647±0.064       | **0.660±0.035**       |
|         | RMSE | 1.011±0.019       | 0.878±0.046       | **0.685±0.038**       |
| $D$=200 | REE  | 0.125±0.001       | 0.073±0.004       | **0.033±0.004**       |
|         | MCC  | 0.331±0.078       | 0.637±0.066       | **0.731±0.017**       |

anomaly detection methods. Once the outlier tasks are identified, they are removed from the set of tasks. Then, we apply the SOUP algorithm on the remaining normal tasks to recover their membership weights. The overall procedure is summarized in Algorithm 2, where $\mathbf{B}_{\mathcal{S}}$ refers to the sub-matrix of matrix $\mathbf{B}$ formed by columns in set $\mathcal{S}$.

We use synthetic datasets to validate the effectiveness of the modified approach in terms of task clustering, feature selection, and robustness against outlier tasks. The data generation procedure is exactly the same as the sparse condition in section 4.1, except we add five outliers tasks, each of which has 10 nonzero coefficients generated from $\mathcal{U}[0.5, 1]$. As a result, we have total $T = 65$ tasks. We set $D = 100$ and $200$.

We compare the modified approach with the two baseline methods, LASSO and VSTG-MTL. Table 6 reports the simulation results for the three methods. It can be observed that the modified approach outperforms the two baseline methods in terms of prediction, estimation, and feature selection.

# 6   Conclusion

This study proposes a novel task clustering approach STCMTL, which can simultaneously learn an overlapping structure among tasks and perform feature selection. Although it is similar in spirit to the existing task clustering methods, it is also significantly more advanced. In the classification of pure and mixed tasks, it produces an identifiable and interpretable task cluster structure and selects relevant features.

STCMTL factorizes the coefficient matrix into a product of the cluster coefficient matrix and membership matrix, and it imposes some constraints on the membership matrix values and sparsities on the cluster coefficient matrix. T4edhe resulting constrained optimization problem is challenging to solve. To circumvent this, a new alternating algorithm is developed, which repeats a three-step process: identifying the set of pure tasks and estimating the membership matrix; updating the cluster coefficient matrix; and updating coefficient matrix until convergence. The tuning of hyper-parameters with validation is conducted inside the algorithm, leading to substantial improvements in computational efficiency. The experimental results show that the proposed STCMTL approach outperforms the baseline methods using synthetic and real-world datasets. Moreover, STCMTL can be extended to other, more complex task clustering problems, such as robust task clustering.

An important limitation of this study is its lack of theoretical investigation. This will be deferred to future research. We also defer additional possible extensions of the proposed approach to future research.

# References

[1] Qi An, Chunping Wang, Ivo D Shterev, Eric Wang, Lawrence Carin, and David B Dunson. 2008. Hierarchical kernel stick-breaking process for multi-task image analysis. In *Proceedings of the 25th international conference on Machine learning*, ACM, 17–24.

[2] Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research* , 6:1817–1853.

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

[4] Yi Chang, Jing Bi, Ke Zhou, Guirong Xue, Hongyuan Zha, and Zhaohui Zheng. 2012. Multi-task learning for learning to rank in web search. In *Proceedings of the 18th ACM conference on Information and knowledge management*, ACM, 1549-1552.

[5] Bart Bakker and Tom Heskes. 2003. Task clustering and gating for bayesian multi-task learning. *Journal of Machine Learning Research*, 3:83–99.

[6] Aviad Barzilai and Koby Crammer. Convex multi-task learning by clustering. 2015. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, PMLR, 65–73.

[7] Xin Bing, Florentina Bunea, Yang Ning, and Marten Wegkamp. 2019. Adaptive estimation in structured factor models with applications to overlapping clustering. *arXiv:1704.06977.*

[8] Koby Crammer and Yishay Mansour. Learning multiple tasks using shared hypotheses. 2012. In *Advances in Neural Information Processing Systems 25*, Curran Associates, 312–323.

[9] Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1–22.

[10] Pinhua Gong, Jieping Ye, and Changshui Zhang. Multi-stage multi-task feature learning. 2012. In *Advances in Neural Information Processing Systems 25*, Curran Associates, 1988–1996.

[11] Lei Han and Yu Zhang. Multi-stage multi-task learning with reduced rank. 2016. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, AAAI Press, 1638–1644.

[12] Lei Han, Yu Zhang, Guojie Song, and Kunqing Xie. Encoding tree sparsity in multi-task learning: A probabilistic framework. 2014. In *Proceedings of the National 24th Conference on Artificial Intelligence*, AAAI Press, 1854–1860.

[13] Daniel Hernandez-Lobato and Jose Miguel Hernandez Lobato. 2013. Learning feature selection dependencies in multi-task learning. In *Advances in Neural Information Processing Systems 26*, Curran Associates, 212–221.

[14] Jonathan J Hull. A database for handwritten text recognition research. 1994. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:550–554.

[15] Jeong Jun-Yong and Jun Chi-Hyuck. Variable selection and task grouping for multi-task learning. 2018. In *The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, 1589-1598.

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

[17] Seyoung Kim and Eric Po Xing. Tree-guided group lasso for multi-task regression with structured sparsity. 2010. In *Proceedings of the 27th International Conference on Machine Learning*, PMLR, 543–550.

[18] Deguang Kong, Chris Ding, and Heng Huang. Robust nonnegative matrix factorization using l21-norm. 2011. In *Proceedings of the 20th ACM international conference on Information and knowledge managemen*, ACM, October, 673–682.

[19] Abhishek Kumar and Hal Daumé III. Learning task grouping and overlap in multi-task learning. 2012. In *Proceedings of the 29th International Conference on Machine Learning*, Omnipress, 1383–1390.

[20] Yann Lecun, Leon Bottou, Yoshua Bengio and Patrick Haffner. Gradient-based learning applied to document recognition. 1998. *Proceedings of the IEEE*, 86:2278–2324.

[21] Ya Li, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Multi-task model and feature joint learning. 2015. In *Proceedings of the National 25th Conference on Artificial Intelligence* AAAI Press, 3643–3649 .

[22] Aurelie C. Lozano and Grzegorz Swirszcz. Multi-level lasso for sparse multi-task regression. 2012. In *Proceedings of the 29th International Conference on Machine Learning*. PMLR, 207–215.

[23] Guillaume Obozinski, Ben Taskar, and Michael Jordan. 2006. Multi-task feature selection. *UC Berkeley Technical Report* ,2: 1–15 .

[24] Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. 2010. *SIAM Journal on Optimization*, 20:3465–3489.

[25] Lu Tang and Peter X.K. Song. Fused lasso approach in regression coefficients clustering – learning parameter heterogeneity in data integration. 2016. *Journal of Machine Learning Research*, 17:1–23.

[26] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. 1996. In *Proceedings of the 13th International Conference on Machine Learning*, Omnipress, 108–117.

[27] Robert Tibshirani. Regression shrinkage and selection via the lasso. 1996. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 58:267–288.

[28] Xin Wang, Jinbo Bi, Shipeng Yu, and Jiangwen Sun. On multiplicative multitask feature learning. 2014. In *Advances in Neural Information Processing Systems 27*, Curran Associates, 2411–2419.

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

[30] Jiepeng Xu, Jiayu Zhou, and Pang-Ning Tan. Formula: factorized multi-task learning for task discovery in personalized medical models. 2015. In *Proceedings of the 15th SIAM International Conference on Data Mining*, SIAM, 200–209 .

[31] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 88:35–63.

[32] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *CoRR*, abs/1605.06391, 2016.

[33] Yaqiang Yao, Jie Cao, and Heng Chen. Robust task grouping with representative tasks for clustere multi-task learning. 2019. In *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, 1408–1417.

[34] Kai Yu, Valker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. 2005. In *Proceedings of the 22nd International Conference on Machine Learning*, Omnipress, 1012-1019.

[35] Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. 2010. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183-3234.

[36] Jiashuai Zhang, Jianyu Miao, Kun Zhao, and Yingjie Tian. 2019. Multi-task feature selection with sparse regularization to extract common and task-specific features. *Neurocomputing*, 340:76–89.

[37] Yu Zhang and Qiang Yang. 2018. A survey on multi-task learning. *CoRR*, abs/1707.08114.

[38] Yu Zhang, Dit-Yan Yeung, and Qian Xu. 2010. Probabilistic multi-task feature selection. In *Advances in Neural Information Processing Systems 23*. Curran Associates, 2559–2567.

[39] Jiayu Zhou, Jianhui Chen, and Jieping Ye. 2011. Clustered multi-task learning via alternating structure optimization. In *Advances in Neural Information Processing Systems 24*, Curran Associates, 702–710.

[40] Qiang Zhou and Qi Zhao. 2016. Flexible clustered multi-task learning by learning representative tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 28:266–278.

[41] Lingxue Zhu, Jing Lei, Lambertus Kleib, Bernie Devlinb, and Kathryn Roeder. 2019. Semisoft clustering of single-cell data. *Proceedings of the National Academy of Sciences of the United States of America*, 116:466–471.