
A APPENDIX

In Appendix A.1, we further expand on the related works described in Section 2.

In Appendix A.2, we provide the detailed DGP used in Section 4.2.

In Appendix A.3, we conduct ablation experiments to demonstrate the individual roles of the two steps in our proposed method.

In Appendix A.4, we provide the pseudo-code for the proposed MTL-HMB.

In Appendix A.5, we include detailed experimental information, including the real data description and implementation details.

In Appendix A.6, we discuss the limitations of our work and potential future research directions.

A.1 EXPANDED RELATED WORKS

Multi-group data integration. Multi-group data integration and MTL share the common goal of learning from multiple datasets or tasks simultaneously. The input features and response of a single task can be viewed as a separate group. There are several existing methods in the statistical literature for multi-group data analysis, which can be broadly classified into three categories. The first category designs specialized regression models (Meinshausen & Bühlmann, 2015; Zhao et al., 2016; Wang et al., 2018; Huang et al., 2023a;b) or factor regression models (Wang et al., 2023a;b) to handle large-scale heterogeneous data and identify group-specific structures. The second category employs specified parameter space constraints, such as fused penalties, to estimate regression coefficients that capture subgroup structures (Tang & Song, 2016; Ma & Huang, 2017; Chen et al., 2021; Li & Sang, 2019; Tang et al., 2021; Lam et al., 2022; Duan & Wang, 2023; Zhang et al., 2024b). The third category involves transfer learning, which borrows information from source data to target data (Li et al., 2022; Tian et al., 2022; Zhang & Zhu, 2022; Tian & Feng, 2023; Cai & Pu, 2024; Cai et al., 2024; He et al., 2024; Zhang et al., 2024a). The aforementioned multi-group data integration approaches address distribution and posterior heterogeneity but overlook block-wise missing issues. Additionally, most methods rely on structured model assumptions, such as linearity, limiting their capacity to capture complex relationships.

Heterogeneous feature spaces. Existing transfer learning methods mainly addressed either distribution shift or posterior shift separately, with fewer studies considering both types of shifts simultaneously. For instance, Moon & Carbonell (2017) investigated scenarios with both heterogeneous feature and label spaces in the context of natural language processing. They proposed a method that learned a common embedding for the features and labels and then established a mapping between them. Similarly, Bica & van der Schaar (2022) focused on a shared label space but assumed that all tasks had a common source, utilizing the same encoder to extract shared representations. However, this assumption was often unrealistic in practice. Even when sources were identical, different tasks could exhibit significant heterogeneity due to variations in subjects, locations, and experimental settings. For example, in our ADNI real data (Section 4.3), tasks sharing MRI features might still differ due to varying experimental conditions. A key distinction in our method is that we treat this problem as a block-wise missing data issue rather than simply considering each task to have only the observed features. This perspective aligns more closely with the reality of medical data, where missing problem is common, and these missing features can also influence the response. Additionally, we focus on MTL, which is designed for numerous small-sample and challenging tasks. In contrast, transfer learning often assumed the existence of a large-scale dataset to support a smaller-sample task. For example, in the experiments conducted by Bica & van der Schaar (2022), the source domain’s sample size was typically more than ten times that of the target domain. However, in real-world scenarios, it is more common for all tasks to have relatively small and limited sample sizes. Our method aims to provide a more comprehensive and robust learning framework by integrating heterogeneous information across these small-sample tasks.

A.2 DATA GENERATION PROCESS IN SECTION 4.2

We consider MTL for multiple tasks. The DGP is similar to that in Section 4.1 but is extended to accommodate more tasks. For three-task learning, the features for the t -th task are denoted as $\mathbf{x}^t =$

$[\mathbf{x}_0^t | \mathbf{x}_1^t | \mathbf{x}_2^t | \mathbf{x}_3^t]$ and follow a Gaussian distribution with mean $\mathbf{0}$ and an exchangeable covariance matrix. The variance is fixed at 1, and the covariance structure is determined by $(\rho_t)^{0.01|i-j|}$. We randomly generate n_t samples, with only \mathbf{x}_0^t and \mathbf{x}_t^t being observed. The response y^t is given by:

$$y^t = \alpha \sum_{d=1}^p v_{c,d} (x_d^t)^2 / p + (1 - \alpha) \sum_{d=1}^p v_{t,d} x_d^t / p + \varepsilon_r, \quad \forall r \in [3].$$

where $p = \sum_{s=0}^3 p_s$. For three-task learning, we choose the following parameters: $n_1 = n_2 = n_3 = 300$, $p_0 = 125$, $p_1 = p_2 = p_3 = 25$, $\rho_1 = 0.95$, $\rho_2 = \rho_3 = 0.9$, $\alpha = 0.3$, $v_c, v_t \sim N(-10, 10^2)$, and $\varepsilon_t \sim N(0, 0.01)$ for $t \in [3]$.

For four-task learning, the features for the t -th task are denoted as $\mathbf{x}^t = [\mathbf{x}_0^t | \mathbf{x}_1^t | \mathbf{x}_2^t | \mathbf{x}_3^t | \mathbf{x}_4^t]$ and follow a Gaussian distribution with mean $\mathbf{0}$ and an exchangeable covariance matrix. The variance is fixed at 1, and the covariance structure is determined by $(\rho_t)^{0.01|i-j|}$. We randomly generate n_t samples, with only \mathbf{x}_0^t and \mathbf{x}_t^t being observed. The response y^t is given by:

$$y^t = \alpha \sum_{d=1}^p v_{c,d} (x_d^t)^2 / p + (1 - \alpha) \sum_{d=1}^p v_{t,d} x_d^t / p + \varepsilon_r, \quad \forall r \in [4].$$

where $p = \sum_{s=0}^4 p_s$. For four-task learning, we choose the following parameters: $n_1 = n_2 = n_3 = n_4 = 300$, $p_0 = 125$, $p_1 = p_2 = p_3 = p_4 = 25$, $\rho_1 = 0.95$, $\rho_2 = \rho_3 = \rho_4 = 0.9$, $\alpha = 0.3$, $v_c, v_t \sim N(-10, 10^2)$, and $\varepsilon_t \sim N(0, 0.01)$ for $t \in [4]$.

For evaluation, we focus on the average RMSE across all tasks in the testing data, defined as follows:

$$\text{RMSE} = \frac{1}{T} \sum_{t=1}^T \sqrt{\frac{1}{n_{t,\text{test}}} \sum_{i=1}^{n_{t,\text{test}}} (\hat{y}_i^t - y_i^t)^2}.$$

A.3 ABLATION EXPERIMENTS

We propose an MTL framework that involves two steps: Step 1 for HBI (see Section 3.1) and Step 2 for heterogeneous MTL (see Section 3.2). To assess the independent effect of each step, we design ablation experiments. In addition to comparing with STL and HTL, we consider two new ablation experiments. The first is Step 1 + STL, which applies HBI followed by STL to evaluate the effect of Step 2 and is denoted as Ablation 1. The second is homogeneous imputation + Step 2, where we ignore distribution heterogeneity to analyze the impact of disregarding heterogeneity in imputation, denoted as Ablation 2. The data generation process is consistent with Section 4.1, but we adopt a more challenging setting. Specifically, we set $p_1 = 100$, $p_2 = 25$, $p_3 = 25$, $\rho_1 = 0.8$, $\rho_2 = 0.6$, $\alpha = 0.3$, and $\sigma_1 = \sigma_2 = 0.1$. We analyze the impact of sample sizes n_1 and n_2 on three methods by fixing $n_1 = 300$ and varying n_2 as $n_2 = k \times 100$ for $k = 1, \dots, 4$. The experiments are repeated 30 times, and the mean RMSE per task is computed, with the results summarized in Figure 1. It is important to note that, due to the presence of distribution heterogeneity in this setting, HTL performs the worst. We analyze the ablation results from three perspectives. First, it is evident that both Ablation 2 and our proposed MTL-HMB outperform STL and Ablation 1, indicating that Step 2 plays a crucial role in enhancing STL performance. Second, by comparing Ablation 1 with STL, we observe that Ablation 1 consistently achieves lower loss across different sample sizes, demonstrating that Step 1 improves predictions for a single dataset. Third, when comparing Ablation 2 with our proposed method, Ablation 2 shows higher loss, suggesting that ignoring distribution heterogeneity negatively impacts performance. Overall, our ablation experiments demonstrate that when both distribution and posterior heterogeneity are present, both steps of our proposed framework are crucial.

A.4 PSEUDO-CODE FOR OUR PROPOSED MTL-HMB

Algorithm 1 provides the pseudo-code for training our proposed MTL method. For simplicity, we set the mini-batch size to be the same across all T datasets: $B^t = B$ for $t \in [T]$. For HBI, we divide the data into training and testing sets and train the parameters on the training set. Early stopping is applied to \mathcal{L}_{pre} on the t -th dataset's testing data to check for convergence and perform model selection. For heterogeneous MTL, the data is split into training, validation, and testing sets. Parameters are

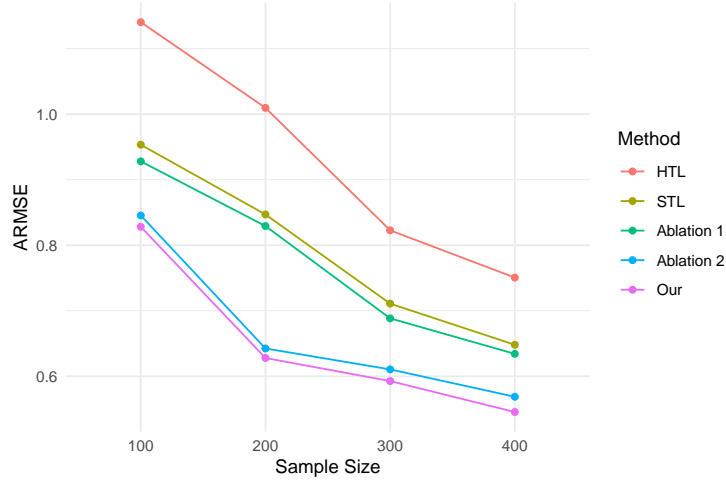


Figure 1: The average RMSEs of all methods across different n_2 sample sizes.

trained on the training set, and the best hyperparameter combination is selected using the validation set. Early stopping is applied to $\mathcal{L}_{\text{integ}}$ on the validation set to check for convergence, and the final prediction metrics are calculated on the test set. In practice, we choose the regularization parameters γ , δ , and κ from the set $[0.01, 0.1, 1]$ for $\mathcal{R}_{\text{orth}}$, \mathcal{R}_{imp} , and \mathcal{R}_{dr} . In our experiments, we found that the selection of γ , δ , and κ is robust, having minimal impact on the final prediction performance.

A.5 EXPERIMENTAL DETAILS

A.5.1 DATASET DESCRIPTION

In this subsection, we provide a detailed description of the ADNI database used in Section 4.3. The ADNI study (Mueller et al., 2005) aims to identify biomarkers that track the progression of Alzheimer’s disease (AD). The MMSE score, which measures cognitive impairment, is treated as the response variable, and we aim to select biomarkers from three complementary data sources: MRI, PET, and gene expression. Given the sparsity assumption, we use region of interest (ROI) level data rather than raw imaging data, as the latter might not be suitable for our method. MRI variables include volumes, cortical thickness, and surface areas, while PET features represent standard uptake value ratios (SUVr) of different ROIs. Gene expression variables are derived from blood samples and represent expression levels at different gene probes. To reduce the number of gene expression variables, we apply sure independence screening (SIS), narrowing it down to 300 variables. This results in a total of 680 features, including 267 MRI features and 113 PET features. The data is sourced from ADNI-2 at month 48, where block-wise missingness occurs due to factors such as low-quality images or patient dropout. Using visit codes, we align MMSE with the imaging data to ensure they are measured within the same month. Ultimately, we obtained two datasets: dataset 1 contains only MRI and PET sources, while dataset 2 includes MRI and gene expression sources. Both datasets have relatively small sample sizes, underscoring the importance of effectively using incomplete observations in the analysis.

A.5.2 IMPLEMENTATION DETAILS AND HYPERPARAMETER SETTING

In Section 4, we compare our proposed method (MTL-HMB) with Single Task Learning (STL) and Heterogeneous Transfer Learning (HTL). Here, we provide the implementation details of these three methods.

STL. For STL, we use standard deep neural networks to train each dataset individually. Each dataset is split into 60% for training, 20% for validation, and 20% for testing. On the training set, we perform hyperparameter tuning, including network width from $\{32, 64, 128\}$, depth from $\{2, 3, 4, 5\}$, and batch size from $\{8, 16, 32\}$ (with 8 included due to the smaller sample size in the ADNI database). Additionally, we set the learning rate to 0.001 and the early-stopping patience to 30. To

stabilize the optimization during iterations, we use the exponential scheduler (Patterson & Gibson, 2017), which decays the learning rate by a constant per epoch. In all numerical tasks, we set the decay constant to 0.95, applied every 200 iterations. We tune the hyperparameters and select the best model on the validation set. Finally, the tuned hyperparameters are used to compute the prediction loss on the testing set.

HTL. For HTL, we adapt the network architecture from Bica & van der Schaar (2022) and modify it for our setting. Following their approach, the framework for handling heterogeneous feature spaces consists of a common encoder for shared source and task-specific encoders for task-specific sources, implemented using deep neural networks. The network widths are selected from $\{32, 64, 128\}$ and depths from $\{2, 3, 4\}$. The remaining components are incorporated into an MTL network architecture, similar to the structure described in Section 3.2, where shared and task-specific pathways have depths chosen from $\{2, 3, 4\}$. The output dimensions of the first $L - 1$ layers are selected from $\{32, 64, 128\}$, with the final layer predicting the corresponding response. The batch size is chosen from $\{8, 16, 32\}$, and the learning rate is set to 0.001. To remain consistent with Bica & van der Schaar (2022), we train the prediction loss on the training set, along with regularization terms $\mathcal{R}_{\text{orth}}$ and \mathcal{R}_{dr} . Early stopping and hyperparameter tuning are performed based on the sum of the prediction losses across all datasets on the validation set, with an early-stopping patience of 30. Finally, the tuned hyperparameters are used to compute the prediction loss on the testing set.

MTL-HMB. For the proposed MTL-HMB, we describe the method in two steps: Step 1 and Step 2. **Step 1:** In HBI, the common encoder, task-specific encoders, decoder, and predictor use network architectures with widths selected from $\{8, 16, 32\}$ and depths from $\{1, 2, 3\}$. The batch size is chosen from $\{8, 16, 32\}$, and the learning rate is set to 0.001. Notably, since the features in our simulated data exhibit relatively simple linear relationships, we include smaller network widths and depths in our tuning. **Step 2:** To construct task-shared and task-specific mappings, the network architecture for the shared encoder ϕ_c and the task-specific encoders ϕ_p^t have widths selected from $\{32, 64, 128\}$ and depths from $\{2, 3, 4\}$. The output dimensions are also chosen from $\{32, 64, 128\}$. For the prediction function, both shared and task-specific pathways have depths chosen from $\{2, 3, 4\}$, with the output dimensions of the first $L - 1$ layers selected from $\{32, 64, 128\}$. The final layer predicts the corresponding response. The batch size is chosen from $\{8, 16, 32\}$, and the learning rate is set to 0.001. Early stopping and hyperparameter tuning are conducted based on the sum of the prediction losses across all datasets on the validation set, using an early-stopping patience of 30. Finally, the tuned hyperparameters are applied to compute the prediction loss on the testing set.

A.6 DISCUSSION ABOUT LIMITATIONS AND FUTURE WORK

First, our proposed method essentially assumes that there is common information across all tasks that can be fused, which implies a relatively strong shared structure. For example, in Section 3.2, we assume the existence of a common mapping, ψ_c , between input features and responses for all $r \in [R]$. However, in reality, when there is strong heterogeneity across multiple datasets, the shared structure is often only partial. For instance, in three datasets, only two may share the common ψ_c , while the third task may be too heterogeneous to fuse with the first two. In such cases, an adaptive approach for MTL is needed, one that explores partially shared information among tasks while preserving the uniqueness of the highly heterogeneous task. Currently, some studies have considered adaptive MTL in relatively simple settings, such as linear cases (Duan & Wang, 2023; Tian et al., 2023). However, adaptive MTL in the presence of block-wise, distribution, and posterior heterogeneity remains unexplored, making it a meaningful direction for future research.

Moreover, it is worth noting that in both Section 3.1 (HBI) and Section 3.2 (MTL), the hidden representations in each dataset are learned in two steps: the first step for imputation and the second step for learning the response. This process introduces some computational redundancy. A possible improvement would be to combine these two steps into one, unifying multiple tasks to learn the hidden representations for each task, which can then be used for both imputation and response learning. However, this approach poses computational challenges, such as how to balance different loss functions to achieve both accurate imputation and prediction. Thus, this remains a future research direction worth exploring.

Algorithm 1 Pseudo-code for Our Proposed MTL-HMB.

```

1: Input:  $T$  datasets denoted by  $\{\mathbf{x}_i^t, \mathbf{y}_i^t\}_{i=1}^{n_t}$ , where  $\mathbf{x}_i^t$  includes two blocks  $\mathbf{x}_{0,i}^t$  and  $\mathbf{x}_{t,i}^t$ , learning
   rate  $\eta$ , mini-batch size for the  $t$ -th dataset is denoted by  $B^t$ .
2: Step 1: HBI
3: for  $t = 1, \dots, T$  do ▷ Imputation for task  $t$ -specific source
4:   Initialize:  $\theta^t$  (all parameters in this step)
5:   while not converged do
6:     Sample mini-batch of  $B^t$  demonstrations from the  $t$ -th dataset  $\{\mathbf{x}_i^t, \mathbf{y}_i^t\}_{i=1}^{n_t}$  and mini-
       batch combination of  $B^{-t} = \sum_{s \neq t} B^s$  demonstrations from the rest  $T - 1$  datasets.
7:     for  $i = 1, \dots, B^t$  do ▷ Process batch from the  $t$ -th dataset.
8:        $\mathbf{f}_i^t = E_c(\mathbf{x}_{0,i}^t), \mathbf{g}_i^t = E_p(\mathbf{x}_{0,i}^t)$ 
9:     end for
10:    Compute prediction loss  $\mathcal{L}_{\text{pre}}^t = \sum_{i=1}^{B^t} l(\mathbf{x}_{t,i}^t, G(\mathbf{f}_i^t))$ 
11:    Compute reconstruction loss  $\mathcal{L}_{\text{recon}}^t = \sum_{i=1}^{B^t} l(\mathbf{x}_{0,i}^t, D(\mathbf{f}_i^t, \mathbf{g}_i^t))$ 
12:    for  $i = 1, \dots, B^{-t}$  do ▷ Process batch from the rest  $T - 1$  datasets.
13:       $\mathbf{f}_i^{-t} = E_c(\mathbf{x}_{0,i}^{-t}), \mathbf{g}_i^{-t} = E_p(\mathbf{x}_{0,i}^{-t})$ 
14:    end for
15:    Compute reconstruction loss:  $\mathcal{L}_{\text{recon}}^{-t} = \sum_{i=1}^{B^{-t}} l(\mathbf{x}_{0,i}^{-t}, D(\mathbf{f}_i^{-t}, \mathbf{g}_i^{-t}))$ 
16:    Parameter update  $\theta^t \leftarrow \theta^t - \eta \nabla_{\theta^t} (\mathcal{L}_{\text{pre}}^t + \mathcal{L}_{\text{recon}}^t + \mathcal{L}_{\text{recon}}^{-t})$ 
17:  end while
18:  for  $i = 1, \dots, B^{-t}$  do
19:    Imputation for task  $t$ -specific source:  $\hat{\mathbf{x}}_{t,i}^{-t} = \hat{G}(\hat{E}_c(\mathbf{x}_{0,i}^{-t}))$ 
20:  end for
21: end for
22: Obtain samples with reconstructed features  $\{(\mathbf{x}_{0,i}^t, \dots, \hat{\mathbf{x}}_{t-1,i}^t, \mathbf{x}_{t,i}^t, \hat{\mathbf{x}}_{t+1,i}^t, \dots, \hat{\mathbf{x}}_{T,i}^t), \mathbf{y}_i^t\}_{i=1}^{n_t}$ 
23: Step 2: Heterogeneous MTL
24: Initialize:  $\Theta$  (all parameters in this step)
25: while not converged do
26:   for  $t = 1, \dots, T$  do
27:     for  $i = 1 \dots B^t$  do ▷ Process batch from the  $r$ -th dataset.
28:        $\mathbf{h}_i^t = \phi_c(\mathbf{x}_{0,i}^t), \mathbf{k}_i^t = \phi_p([\mathbf{x}_{0,i}^t | \dots | \hat{\mathbf{x}}_{t-1,i}^t | \mathbf{x}_{t,i}^t | \hat{\mathbf{x}}_{t+1,i}^t | \dots | \hat{\mathbf{x}}_{T,i}^t])$ 
29:       Set  $\mathbf{H}^t = [\mathbf{h}_1^t \dots \mathbf{h}_{B^t}^t]^\top, \mathbf{K}^t = [\mathbf{k}_1^t \dots \mathbf{k}_{B^t}^t]^\top$ 
30:       for  $l = 1 \dots L$  do
31:         if  $l == 1$  then
32:            $\bar{\mathbf{h}}_{l,i}^t = \mathbf{h}_{l,i}^t, \bar{\mathbf{k}}_{l,i}^t = [\mathbf{h}_{l,i}^t | \mathbf{k}_{l,i}^t]$ 
33:         else
34:            $\bar{\mathbf{h}}_{l,i}^t = \mathbf{h}_{l-1,i}^t, \bar{\mathbf{k}}_{l,i}^t = [\mathbf{h}_{l-1,i}^t | \mathbf{k}_{l-1,i}^t]$ 
35:            $\mathbf{h}_{l,i}^t = \text{Shared\_Path}(\bar{\mathbf{h}}_{l,i}^t), \mathbf{k}_{l,i}^t = \text{Task\_Specific\_Path}^t(\bar{\mathbf{k}}_{l,i}^t)$ 
36:         end if
37:       end for
38:        $\hat{\mathbf{y}}_i^t = g^t([\mathbf{h}_{L,i}^t | \mathbf{k}_{L,i}^t])$ 
39:     end for
40:   end for
41:   Compute integration loss:  $\mathcal{L}_{\text{integ}} = \sum_{t=1}^T \sum_{i=1}^{B^t} l(\mathbf{y}_i^t, \hat{\mathbf{y}}_i^t)$ 
42:   Compute orthogonal regularizer for features:  $\mathcal{R}_{\text{orth}} = \sum_{t=1}^T \|(\mathbf{H}^t)^\top \mathbf{K}^t\|_F^2$ 
43:   Compute robust regularizer for imputation:  $\mathcal{R}_{\text{imp}} = \sum_{t=1}^T \sum_{s \neq 0, t} \|\Theta_{s,p,1}^t\|_F^2$ 
44:   Compute regularizer for redundancy:  $\mathcal{R}_{\text{dr}} = \sum_{t=1}^T \sum_{l=1}^L \|(\Theta_{c,l}^t)^\top \Theta_{p,l,1:d_{c,l-1}}^t\|_F^2$ 
45:   Parameters update:
46:    $\Theta \leftarrow \Theta - \eta \nabla_{\Theta} (\mathcal{L}_{\text{integ}} + \mathcal{R}_{\text{orth}} + \mathcal{R}_{\text{imp}} + \mathcal{R}_{\text{dr}})$ 
47: end while
48: Output: Learnt parameters  $\Theta$ 

```

REFERENCES

- Ioana Bica and Mihaela van der Schaar. Transfer learning on heterogeneous feature spaces for treatment effects estimation. *Advances in Neural Information Processing Systems*, 35:37184–37198, 2022.
- Changxiao Cai, T Tony Cai, and Hongzhe Li. Transfer learning for contextual multi-armed bandits. *The Annals of Statistics*, 52(1):207–232, 2024.
- T Tony Cai and Hongming Pu. Transfer learning for nonparametric regression: Non-asymptotic minimax analysis and adaptive procedure. *arXiv preprint arXiv:2401.12272*, 2024.
- Jingxiang Chen, Quoc Tran-Dinh, Michael R Kosorok, and Yufeng Liu. Identifying heterogeneous effect using latent supervised clustering with adaptive fusion. *Journal of Computational and Graphical Statistics*, 30(1):43–54, 2021.
- Yaqi Duan and Kaizheng Wang. Adaptive and robust multi-task learning. *The Annals of Statistics*, 51(5):2015–2039, 2023.
- Zelin He, Ying Sun, and Runze Li. Transfusion: Covariate-shift robust transfer learning for high-dimensional regression. In *International Conference on Artificial Intelligence and Statistics*, pp. 703–711. PMLR, 2024.
- Jian Huang, Yuling Jiao, Wei Wang, Xiaodong Yan, and Liping Zhu. Integrative analysis for high-dimensional stratified models. *Statistica Sinica*, 33, 2023a.
- Xinmeng Huang, Kan Xu, Donghwan Lee, Hamed Hassani, Hamsa Bastani, and Edgar Dobriban. Optimal heterogeneous collaborative linear regression and contextual bandits. *arXiv preprint arXiv:2306.06291*, 2023b.
- Henry Lam, Kaizheng Wang, Yuhang Wu, and Yichen Zhang. Adaptive data fusion for multi-task non-smooth optimization. *arXiv preprint arXiv:2210.12334*, 2022.
- Furong Li and Huiyan Sang. Spatial homogeneity pursuit of regression coefficients for large datasets. *Journal of the American Statistical Association*, 2019.
- Sai Li, T Tony Cai, and Hongzhe Li. Transfer learning for high-dimensional linear regression: Prediction, estimation and minimax optimality. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):149–173, 2022.
- Shujie Ma and Jian Huang. A concave pairwise fusion approach to subgroup analysis. *Journal of the American Statistical Association*, 112(517):410–423, 2017.
- Nicolai Meinshausen and Peter Bühlmann. Maximin effects in inhomogeneous large-scale data. *The Annals of Statistics*, 43(4):1801–1830, 2015.
- Seungwhan Moon and Jaime G Carbonell. Completely heterogeneous transfer learning with attention-what and what not to transfer. In *IJCAI*, volume 1, pp. 1–2, 2017.
- Susanne G Mueller, Michael W Weiner, Leon J Thal, Ronald C Petersen, Clifford Jack, William Jagust, John Q Trojanowski, Arthur W Toga, and Laurel Beckett. The alzheimer’s disease neuroimaging initiative. *Neuroimaging Clinics of North America*, 15(4):869, 2005.
- Josh Patterson and Adam Gibson. *Deep learning: A practitioner’s approach*. ” O’Reilly Media, Inc.”, 2017.
- Lu Tang and Peter XK Song. Fused lasso approach in regression coefficients clustering–learning parameter heterogeneity in data integration. *Journal of Machine Learning Research*, 17(113):1–23, 2016.
- Xiwei Tang, Fei Xue, and Annie Qu. Individualized multidirectional variable selection. *Journal of the American Statistical Association*, 116(535):1280–1296, 2021.
- Ye Tian and Yang Feng. Transfer learning under high-dimensional generalized linear models. *Journal of the American Statistical Association*, 118(544):2684–2697, 2023.

-
- Ye Tian, Haolei Weng, and Yang Feng. Unsupervised multi-task and transfer learning on gaussian mixture models. *arXiv preprint arXiv:2209.15224*, 2022.
- Ye Tian, Yuqi Gu, and Yang Feng. Learning from similar linear representations: Adaptivity, minimaxity, and robustness. *arXiv preprint arXiv:2303.17765*, 2023.
- Peiyao Wang, Yufeng Liu, and Dinggang Shen. Flexible locally weighted penalized regression with applications on prediction of alzheimer’s disease neuroimaging initiative’s clinical scores. *IEEE transactions on medical imaging*, 38(6):1398–1408, 2018.
- Peiyao Wang, Quefeng Li, Dinggang Shen, and Yufeng Liu. High-dimensional factor regression for heterogeneous subpopulations. *Statistica Sinica*, 33(1):27, 2023a.
- Peiyao Wang, Haodong Wang, Quefeng Li, Dinggang Shen, and Yufeng Liu. Joint and individual component regression. *Journal of Computational and Graphical Statistics*, pp. 1–16, 2023b.
- Ruqian Zhang, Yijiao Zhang, Annie Qu, Zhongyi Zhu, and Juan Shen. Concert: Covariate-elaborated robust local information transfer with conditional spike-and-slab prior. *arXiv preprint arXiv:2404.03764*, 2024a.
- Xin Zhang, Jia Liu, and Zhengyuan Zhu. Learning coefficient heterogeneity over networks: A distributed spanning-tree-based fused-lasso regression. *Journal of the American Statistical Association*, 119(545):485–497, 2024b.
- Yijiao Zhang and Zhongyi Zhu. Transfer learning for high-dimensional quantile regression via convolution smoothing. *arXiv preprint arXiv:2212.00428*, 2022.
- Tianqi Zhao, Guang Cheng, and Han Liu. A partially linear framework for massive heterogeneous data. *Annals of statistics*, 44(4):1400, 2016.