A Difference Standardization Method for Mutual Transfer Learning

Haoqing Xu [1] **Meng Wang** [1 2] **Beilun Wang** [1 2 *]

Abstract

In many real-world applications, mutual transfer learning is the paradigm that each data domain can potentially be a source or target domain. This is quite different from transfer learning tasks where the source and target are known a priori. However, previous studies about mutual transfer learning either suffer from high computational complexity or oversimplified hypothesis. To overcome these challenges, in this paper, we propose the Difference Standardization method (**DiffS**) for mutual transfer learning. Specifically, we put forward a novel distance metric between domains, the standardized domain difference, to obtain fast structure recovery and accurate parameter estimation simultaneously. We validate the method's performance using both synthetic and real-world data. Compared to previous methods, DiffS demonstrates a speed-up of approximately 3000 times that of similar methods and achieves the same accurate learnability structure estimation.

1. Introduction

In various real-world applications, it is a common practice to take advantage of domain-specific knowledge in a source domain to improve the performance in a target domain, which is called transfer learning (Zhuang et al., 2020). For example, human beings could transfer the knowledge about how to play the violin into playing the piano, or learn to drive a car using the experience when riding a bicycle. In transfer learning, the source domains and target domain are usually specified as a priori.

However, nowadays there are a growing amount of big data applications. Such transferring scenario can not be easily applied in those data collections. As an example, climate analysis (Vose et al., 2014) utilizes a collection of data sets recorded in different time periods, different places or different devices. Due to the variety of the distributions in these data sets, they are split into different data domains. In this case, every data domain could possibly be the target domain depending on their inherent characteristics. The source and target domains are hence not known a priori. Transfer learning strategy may apply to such a task but it will suffer from exponential explosion problem since it needs to try every pair of source and target domains. As a result, mutual transfer learning was proposed by (Cheng et al., 2020) in order to efficiently resolve the problem. In mutual transfer learning, every data domain is considered to be a candidate source domain or target domain. Since some of the domains are more related with each other, there exists several domain subgroups. Mutual transfer learning aims to reveal such subgroup structure, or called the learnability structure, within the data collection. Although mutual transfer learning precisely define the features in such Big Data tasks, it also encounters several challenges:

(1) Large-scale setting: In a real-world scenario, data scale grows exponentially, e.g., the number of domains, sample size of each domain and feature dimension. In the NOAA's nClimDiv Database, there have been more than 500,000 samples recorded monthly ever since 1895 coming from hundreds of data domains. If we need daily or even hourly predictions, the scale would be further enlarged by hundreds of times. The large-scale setting leads to extremely long time cost or even infeasibility for some methods. **(2) Mixed-effects heterogeneity:** In mutual transfer learning tasks, some features may have the same effect on the response among different domains, which are called the global features. Yet, there may exist some features that behave differently in various domains. These features can be called the heterogeneous features. Global and heterogeneous features may simultaneously exist in real-world data. To illustrate, in user preference modeling (Aaker et al., 2013; Lenk et al., 1996), the low price may attract every consumer while a nice appearance of the product may be only appreciated by part of people.

One approach to addressing these challenges is to integrate learnability structure recovery into optimization problems.

[1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China [2]Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China. *Correspondence to: Beilun Wang <beilun@seu.edu.cn>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

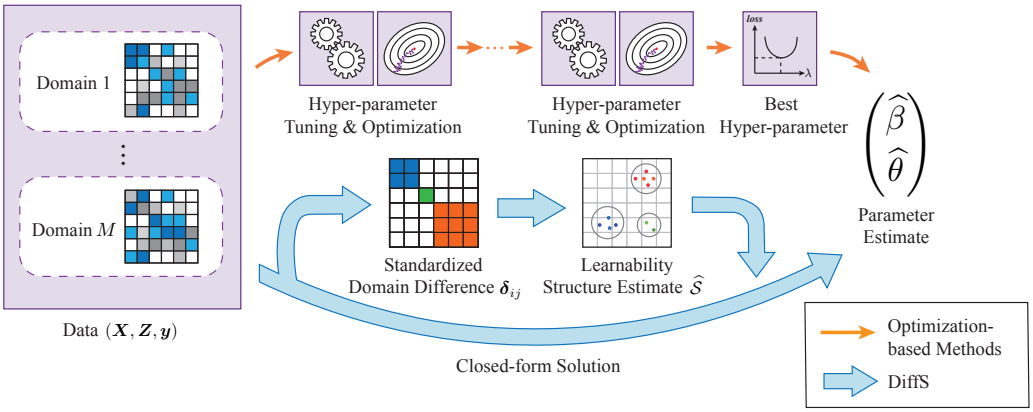


Figure 1. Comparison between DiffS and previous optimization-based methods. Optimization-based methods (orange arrow path) typically estimate parameters multiple times when tuning hyper-parameters. They could be infeasible in large-scale settings. Instead, DiffS (blue arrow path) uses the proposed standardized domain difference to accurately obtain the learnability structure. Meanwhile, we derive a closed-form solution to further accelerate the procedure.

The generalized least squares method (Anh & Chelliah, 1999) is a scalable method with a closed-form solution that solves the mixed-effects parameter estimation problem. However, it lacks the ability to recover learnability structure since it treats heterogeneous parameters of each domain individually. As an improvement, (Cheng et al., 2020) minimizes generalized least square loss with a penalty on heterogeneous parameters to fuse domains into subgroups. Nevertheless, it suffers from extremely high computational cost and memory usage when dealing with large-scale data. To our best knowledge, most of the optimization methods that address mixed-effects problem have large time complexities, either because of hyper-parameter tuning or algorithm complexity. In order to efficiently estimate the learnability structure, another category of methods turn to statistical approaches. For instance, (Xue et al., 2007) utilizes the Dirichlet process to estimate heterogeneous parameters. (Bakker & Heskes, 2003) uses an EM algorithm for its Gaussian mixture model assumption. Despite the fast computation, they cannot deal with mixed-effects heterogeity situation since they simply equally treat all the features.

In this paper, we propose the Difference Standardization method (**DiffS**) to overcome the two challenges in mutual transfer learning. Our method constructs a novel domain distance metric called the *standardized domain difference* and estimates the learnability structure by evaluating such difference between data domains. We also further research the properties of parameters in different domains and theoretically guarantee the learnability structure recovery using our proposed algorithm. DiffS is able to estimate the parameters via a closed-form solution, which much accelerates the estimation as illustrated in Figure 1. The main contributions of this work can be summarized as follows:

- **Novel method:** We propose DiffS using the designed *standardized domain difference* to address learnability structure recovery problem. With standardized domain difference, the learnability structure can be estimated from raw data without any need of prior knowledge.
- **Fast and tuning-free estimation:** DiffS is feasible with large-scale data and avoids hyper-parameter tuning problems compared to common regularization methods. In the synthetic experiments, result shows DiffS is about 3000 times faster than previous methods in large-scale settings and spends only about 1/10 of the time cost by k-means.
- **Theoretical guarantees:** We theoretically guarantee the perfect learnability recovery ability of DiffS and analyze the computational complexity. Also, we prove the property of the standardized domain difference.
- **Synthetic and real-world experiments:** We compare DiffS with state-of-the-art baseline methods. Results show that DiffS outperforms the baseline methods in almost all cases of experiment settings. We also apply DiffS to the monthly temperature prediction problem on NOAA's nClimDiv Database and Microarray Data to prove the feasibility in real-world applications.

Notations We denote bold lowercase characters like $\boldsymbol{\alpha}$ as column vectors, bold uppercase characters like $\boldsymbol{A}$ as matrices and calligraphic characters like $\mathcal{A}$, except $\mathcal{P}$ and $\mathcal{I}$, as sets. $|\mathcal{A}|$ denotes the cardinality of the set $\mathcal{A}$. $\mathcal{P}(\mathcal{A})$ denotes the set consisting of all partitions for set $\mathcal{A}$. $\|\cdot\|$ refers to the ℓ_2 norm of a vector. The indicator function $\mathcal{I}(P)$ equals 1 if P is true and 0 otherwise. The squared root $\boldsymbol{A}^{1/2}$ of a positive semi-definite matrix $\boldsymbol{A}$ is defined as $\boldsymbol{A}^{1/2}\boldsymbol{A}^{1/2} = \boldsymbol{A}$. $\mathbf{1}_{p\times q}, \mathbf{0}_{p\times q} \in \mathbb{R}^{p\times q}$ denote $p \times q$ matrices

with all entries filled with 1 or 0, respectively. $\boldsymbol{I}_q \in \mathbb{R}^{q\times q}$ denotes the $q \times q$ identity matrix.

2. Background

2.1. Formalization of Mixed-effects Heterogeneity with Learnability Structure

We formalize the learnability structure following the two-layer mixed-effects regression model in (Cheng et al., 2020). Suppose that M data domains are divided into S subgroups, which can be expressed as a partition $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_S\}$ of $\{1, 2, \ldots, M\}$. In data domain i, there are n_i data samples with p-dimensional global features and q-dimensional heterogeneous features. The global parameters are shared among all the domains and the heterogeneous parameters are shared in one subgroup. Since we do not known how many subgroups there are and what member domains they have, the heterogenous parameters are only shared in one data domain at the very beginning. Combine these samples of data domain i in a vector form, we use the linear mixed-effects model to describe the mutual transfer learning task as

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i(\boldsymbol{\alpha}_s + \boldsymbol{u}_i) + \boldsymbol{\varepsilon}_i,\ i \in \mathcal{S}_s, \tag{1}$$

where $\boldsymbol{y}_i \in \mathbb{R}^{n_i}$ is the response vector, $\boldsymbol{X}_i \in \mathbb{R}^{n_i\times p}$ is the global feature design matrix with parameters $\boldsymbol{\beta}$, $\boldsymbol{Z}_i \in \mathbb{R}^{n_i\times q}$ is the heterogeneous feature design matrix with parameters $\boldsymbol{\alpha}_s$, $\boldsymbol{u}_i \sim \mathcal{N}(\boldsymbol{0}, \sigma_u^2\boldsymbol{I}_q)$ denotes the random effect and $\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \sigma_\varepsilon^2\boldsymbol{I}_{n_i})$ denotes the observation noise. To be explicit, random effect $\boldsymbol{u}_i$ is domain-specific and stays fixed in a single domain sampling procedure while observation noise varies among samples. The global parameters $\boldsymbol{\beta}$ are the same among all the domains, i.e., a hard sharing. The heterogeneous features show a two-layer heterogeneity: the shared subgroup-specific heterogeneous parameters $\boldsymbol{\alpha}$ results in the 1st-layer heterogeneity and the domain-specific random effect $\boldsymbol{u}_i$ of each data domain (lacking in usefulness for knowledge transfering) contributes to the 2nd-layer heterogeneity. Our goal is to obtain an estimate of the true learnability structure $\mathcal{S}^*$ and parameters $\boldsymbol{\beta}^*$, $\boldsymbol{\alpha}_s^*$, $1 \leqslant s \leqslant S$.

2.2. Generalized Least Squares

Under the situations that the learnability structure is unknown (i.e., $\mathcal{S}$ is unknown), the model (1) can be revised as

$$\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{\beta} + \boldsymbol{Z}_i(\boldsymbol{\theta}_i + \boldsymbol{u}_i) + \boldsymbol{\varepsilon}_i,\ 1 \leqslant i \leqslant M, \tag{2}$$

where $\boldsymbol{\theta}_i \in \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_S\}$ are the heterogenous parameters for each data domain since they are simply known shared in one domain at present. Then our main goal becomes to figure out $\boldsymbol{\beta}^*$, $\boldsymbol{\alpha}_s^*$, $1 \leqslant s \leqslant S$ and all the $\boldsymbol{\theta}_i$. In fact, if we ignore the learnability structure embedded in the data, which means we simply put one data domain in each subgroup, the heterogeneity parameters can be analyzed independently among domains. Therefore, this problem can be solved by minimizing

$$L_{\text{GLS}}(\boldsymbol{\beta}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M) = \sum_{i=1}^{M}(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{Z}_i\boldsymbol{\theta}_i)^\top\boldsymbol{W}_i(\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{Z}_i\boldsymbol{\theta}_i), \tag{3}$$

where $\boldsymbol{W}_i = \text{Cov}(\boldsymbol{y}_i|\boldsymbol{X}_i, \boldsymbol{Z}_i)^{-1} = (\sigma_\varepsilon^2\boldsymbol{I} + \sigma_u^2\boldsymbol{Z}_i\boldsymbol{Z}_i^\top)^{-1}$ is the inverse of covariance of $\boldsymbol{y}_i$ given $\boldsymbol{X}_i, \boldsymbol{Z}_i$. The parameters σ_ε^2, σ_u^2 can be consistently estimated via restricted maximum likelihood method (Cheng et al., 2020; Richardson & Welsh, 1994). We here assume they are known. The GLS estimator (3) has a closed-form solution, characterized by

$$\begin{aligned}\widehat{\boldsymbol{\beta}} &= (\sum_{i=1}^{M}\boldsymbol{X}_i^\top\boldsymbol{W}_i\boldsymbol{X}_i)^{-1}\sum_{i=1}^{M}\boldsymbol{X}_i^\top\boldsymbol{W}_i\boldsymbol{X}_i\boldsymbol{\beta}_i^{\mathcal{D}},\\ \widehat{\boldsymbol{\theta}}_i &= \boldsymbol{\theta}_i^{\mathcal{D}},\end{aligned} \tag{4}$$

where

$$\left(\boldsymbol{\beta}_i^{\mathcal{D}}, \boldsymbol{\theta}_i^{\mathcal{D}}\right)^\top = [\boldsymbol{G}_i^\top\boldsymbol{W}_i\boldsymbol{G}_i]^{-1}\boldsymbol{G}_i^\top\boldsymbol{W}_i\boldsymbol{y}_i, \tag{5}$$

is called the domain GLS estimator, just for symbol simplification. Here, $\boldsymbol{G}_i = (\boldsymbol{X}_i, \boldsymbol{Z}_i)$ is the augmented design matrix. Although closed-form solutions are available, we cannot extract learnability structure directly from the estimates since in the model we treat $\boldsymbol{\theta}$ independently. Two of the estimated $\boldsymbol{\theta}$ from the same subgroup can rarely turn out to be exactly the same due to sample noise and other disturbance.

3. Methodology

3.1. DiffS

Given design matrix $\boldsymbol{X}_i \in \mathbb{R}^{n_i\times p}$, $\boldsymbol{Z}_i \in \mathbb{R}^{n_i\times q}$, $1 \leqslant i \leqslant M$, we assume the response $\boldsymbol{y}_i \in \mathbb{R}^{n_i}$ follows the mixed-effects heterogeneity model (1) with true parameters $\boldsymbol{\beta}^*$, $\boldsymbol{\alpha}^*$ but with learnability structure $\mathcal{S}^*$ unknown. Combining optimization and statistical method premises, we propose the Difference Standardization method (**DiffS**) in order to directly obtain the learnability structure and further simultaneously estimate the global and heterogeneous parameters.

Basically, we involve constraints to ensure that the heterogeneous parameters are the same in one subgroup and try to extract a learnability structure out of raw data. Derived from the generalized least square loss, DiffS considers the

problem of

$$\begin{aligned}\min\ & L_{\text{DiffS}}(\boldsymbol{\beta}, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_M, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{\widehat{S}}) = \\ & \sum_{i=1}^{M} (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{Z}_i\boldsymbol{\theta}_i)^\top \boldsymbol{W}_i (\boldsymbol{y}_i - \boldsymbol{X}_i\boldsymbol{\beta} - \boldsymbol{Z}_i\boldsymbol{\theta}_i), \\ \text{s.t.}\ & \widehat{\mathcal{S}} = \Psi(\boldsymbol{X}, \boldsymbol{Z}), \\ & \boldsymbol{\theta}_i = \boldsymbol{\alpha}_s,\ \forall i \in \widehat{\mathcal{S}}_s, 1 \leqslant s \leqslant |\widehat{S}|. \end{aligned} \tag{6}$$

Here, $\boldsymbol{X}, \boldsymbol{Z}$ stand for the design matrices of all the data domains, $N = \sum_{i=1}^{M} n_i$ is the total number of samples, the learnability recovering function $\Psi : \mathbb{R}^{N\times(p+q)} \mapsto \mathcal{P}(\{i \in \mathbb{N}_+, i \leqslant M\})$, derived by us, generates a learnability structure estimated directly from data, and $\widehat{S} = |\widehat{\mathcal{S}}|$ is the number of estimated subgroups.

The problem (6) can be divided into two parts, consistent with the two steps in Section 3.3: the learnability structure recovery part, and the parameter estimating part. In the learning structure recovery part, the recovering function Ψ extracts the relationships of data domains from the raw data. It is designed to utilize the distances between domains in order to divide them into subgroups. To be specific, the recovering function firstly converts the data $\boldsymbol{X}_i, \boldsymbol{Z}_i$ into a distance matrix $\boldsymbol{\Delta}$ as its input:

$$\Psi(\boldsymbol{X}, \boldsymbol{Z}) = \Psi\left(\boldsymbol{\Delta}(\boldsymbol{X}, \boldsymbol{Z})\right), \tag{7}$$

where $\boldsymbol{\Delta} = (\Delta_{ij})_{M\times M}$ and Δ_{ij} denotes the distance between domain i and j. It further merges similar domains depending on the distance matrix as described in the Step I in Section 3.3. The distance measurement can be various, but in order to recover the learnability structure perfectly, we further propose the *standardized domain difference* in Section 3.2. Once we obtain the estimated subgroups $\widehat{\mathcal{S}}$, in the parameter estimating part, we can solve (6) with a simple closed-form solution described in Section 3.3, the Step II.

3.2. Standardized Domain Difference

In order to accurately recover the learnability structure, the remaining problem is to design a effective distance metric Δ_{ij} and the recovering function Ψ. Firstly, we look into the probability property of the differences between the domain GLS estimates for heterogenous parameters. It can be proved that

$$\boldsymbol{\theta}_i^{\mathcal{D}} - \boldsymbol{\theta}_j^{\mathcal{D}} \sim \mathcal{N}(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_j^*, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j), \tag{8}$$

where $\boldsymbol{\theta}_i^*$, $\boldsymbol{\theta}_i^*$ are the true parameters for domain i, j, $\boldsymbol{\Sigma}_i = (\boldsymbol{0}_{q\times p}, \boldsymbol{I}_q)[\boldsymbol{G}_i^\top \boldsymbol{W}_i \boldsymbol{G}_i]^{-1}(\boldsymbol{0}_{q\times p}, \boldsymbol{I}_q)^\top$ is the covariance matrix with regard to domain GLS estimator. Since the covariance matrices vary among difference domain pair i, j, directly applying ℓ_2-norm on the original domain difference is likely to have a wrong learnability structure estimate. In order to address such problem, we standardize the original domain difference and propose the *standardized domain difference*, which defined as

$$\boldsymbol{\delta}_{ij} = (\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)^{-1/2}(\boldsymbol{\theta}_i^{\mathcal{D}} - \boldsymbol{\theta}_j^{\mathcal{D}}). \tag{9}$$

In DiffS, we use the ℓ_2 norm of the standardized domain difference as the distance between domains:

$$\Delta_{ij} = \|\boldsymbol{\delta}_{ij}\|^2. \tag{10}$$

It can be proven that

$$\boldsymbol{\delta}_{ij} \sim \begin{cases} \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), & i, j \text{ in the same subgroup,} \\ \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{I}), & \text{otherwise,} \end{cases} \tag{11}$$

where $\boldsymbol{\mu}_{ij} = (\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)^{-1/2}(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_j^*)$. See proof in Appendix A.1. If domain i, j are in the same subgroup, $\boldsymbol{\delta}_{ij}$ follows a standard normal distribution. Otherwise, it follows another normal distribution with identity covariance but non-zero mean. As a result, standardized domain difference extracts the hidden relationship between two domains from heterogeous covariances and we can further utilize it in the following estimation. Additionally, standardized domain difference helps guarantee the perfect recovery of learnability structure (Theorem 4.3) with the learnability recovering function Ψ. The details of Ψ is described in the following section.

Algorithm 1 DiffS

1: **Input:** Design matrices $\boldsymbol{X}_i \in \mathbb{R}^{n_i\times p}$, $\boldsymbol{Z}_i \in \mathbb{R}^{n_i\times q}$, response vector $\boldsymbol{y}_i \in \mathbb{R}^{n_i}$, $1 \leqslant i \leqslant M$;
2: **Prepare:** Calculate the domain GLS estimates $\boldsymbol{\theta}_i^{\mathcal{D}}$ with (5) and the standardized domain differences $\boldsymbol{\delta}_{ij}$ with (9);
3: **Step 1:** Calculate the distance matrix $\boldsymbol{\Delta} = (\|\boldsymbol{\delta}_{ij}\|^2)_{M\times M}$ and then apply learnability recovering function Ψ (Algorithm 2) with $\boldsymbol{\Delta}$ to obtain $\widehat{\mathcal{S}}$.
4: **Step 2:** Calculate $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\alpha}}_s$ with (12) and set $\widehat{\boldsymbol{\theta}}_i = \widehat{\boldsymbol{\alpha}}_s$ for $i \in \widehat{\mathcal{S}}_s, 1 \leqslant s \leqslant \widehat{S}$;
5: **return** $\widehat{S}$, $\widehat{\boldsymbol{\beta}}$, $\widehat{\boldsymbol{\alpha}}_s$, $1 \leqslant s \leqslant S$;

3.3. Two-step Solution

DiffS solves the problem (6) via two main steps as proposed in Algorithm 1.

3.3.1. Step I: Learnability Structure Recovering with Standardized Domain Difference

The recovering function Ψ is implemented in this step. With the help of standardized domain difference, a basic intuition to decide whether the two domains are in the same

Algorithm 2 Learnability Structure Recovering Ψ

Input: Distance matrix $\boldsymbol{\Delta} \in \mathbb{R}^{M\times M}$;
Initialize subgroups $\widehat{\mathcal{S}} = \{\{1\}, \{2\}, \ldots, \{M\}\}$;
Set $\nu = 0.001$ (Recommended);
Obtain threshold $\lambda = F_q^{-1}(1-\nu)$;
for $m = 1$ to $M-1$ **do**
  if $\min_{i\neq j}\ \Delta_{ij} > \lambda$ **then**
    break;
  end if
  Find $u = \arg\min_i \sum_{j\neq i} \mathcal{I}(\Delta_{ij} \leqslant \lambda)$ subject to $\exists j \neq i, \Delta_{ij} \leqslant \lambda$;
  Find $v = \arg\min_j\ \Delta_{uj}$;
  Add $\widehat{\mathcal{S}}_t = \widehat{\mathcal{S}}_u \cup \widehat{\mathcal{S}}_v$ to $\widehat{\mathcal{S}}$ and remove $\widehat{\mathcal{S}}_u$, $\widehat{\mathcal{S}}_v$ from $\widehat{\mathcal{S}}$;
  Insert a new row and column into $\boldsymbol{\Delta}$ indexed with t, where $\Delta_{t,j} = \max(\Delta_{u,j}, \Delta_{v,j})$, $\forall j \neq u, v, t$;
  Delete the rows and columns in $\boldsymbol{\Delta}$ w.r.t. subgroup u, v;
end for
return $\widehat{\mathcal{S}}$;

subgroup or not is to decide a threshold. If Δ_{ij} is less than the threshold, domain i, j are potentially in the same subgroup whereas if Δ_{ij} is larger than the threshold, the two domains are considered as not being from the same subgroup. Inspired by (9), DiffS adopts a fixed threshold $\lambda = F_q^{-1}(1-\nu)$ where F_q^{-1} denotes the inverse CDF of the χ^2-distribution where q denotes the degrees of freedom and ν is the significance parameter that controls the probability of false partitioning with recommended value of 0.001. The choice of ν is discussed in detail in the Appendix A.3.

Therefore, our proposed learnability structure recovering function Ψ is materialized as a clustering-based algorithm summarized in Algorithm 2 with standardized domain distance matrix input. The algorithm starts from the basic learnability structure that there is one domain in each subgroup. In each of iterations (up to $M-1$ iterations in total), we find the subgroup u that has the **least** number of between-subgroup distances below the threshold λ. Then we merge $\widehat{\mathcal{S}}_u$ and its 'nearest neighbor' $\widehat{\mathcal{S}}_v$ that has the smallest distance between the two. To guarantee that the estimated subgroups have complete inner-group linkages (here linkage refers to the potential to be in the same subgroup), the subgroup distance is set to the **largest** distance between domains from the two groups. When the algorithm stops, we can guarantee that the true learnability structure can be recovered by $\widehat{\mathcal{S}}$ via Theorem 4.3.

3.3.2. Step II: Synchronized Parameter Estimating

As long as the $\widehat{\mathcal{S}}$ is obtained, we can synchronously calculate the estimate for both global and heterogeneous parameters via a closed-form solution. To prevent lengthy expressions, we consider an $Mq \times \widehat{S}q$ labeling matrix $\boldsymbol{B}(\widehat{\mathcal{S}})$ with $\boldsymbol{B}_{ij}(\widehat{\mathcal{S}}) = \mathcal{I}(i \in \widehat{\mathcal{S}}_j)\boldsymbol{I}_q$ as its (i,j)-th $q\times q$ block. The labeling matrix satisfies $(\boldsymbol{\theta}_1^\top, \ldots, \boldsymbol{\theta}_M^\top)^\top = \boldsymbol{B}(\widehat{\mathcal{S}})(\boldsymbol{\alpha}_1^\top, \ldots, \boldsymbol{\alpha}_{\widehat{S}}^\top)^\top$, which maps each $\boldsymbol{\alpha}$ to the corresponding $\boldsymbol{\theta}$ of the domains in the subgroup. With the help of the labeling matrix, we denote $\boldsymbol{G} = (\boldsymbol{X}, \boldsymbol{Z}\boldsymbol{B}(\widehat{\mathcal{S}})) \in \mathbb{R}^{N\times(p+\widehat{S}q)}$ as the augmented design matrix. Then the optimal solution for (6), the DiffS estimate, can be concisely given as

$$\left(\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\alpha}}_1, \ldots, \widehat{\boldsymbol{\alpha}}_{\widehat{S}}\right)^\top = [\boldsymbol{G}^\top\boldsymbol{W}\boldsymbol{G}]^{-1}\boldsymbol{G}^\top\boldsymbol{W}\boldsymbol{y}, \qquad (12)$$

where $\boldsymbol{X} = (\boldsymbol{X}_i^\top)_{1\leqslant i\leqslant M}^\top$, $\boldsymbol{Z} = \mathrm{diag}(\boldsymbol{Z}_i)_{1\leqslant i\leqslant M}$, $\boldsymbol{W} = \mathrm{diag}(\boldsymbol{W}_i)_{1\leqslant i\leqslant M}$, $\boldsymbol{y} = (\boldsymbol{y}_i^\top)_{1\leqslant i\leqslant M}^\top$. The estimated heterogeneous parameters for each domain is obtained by $\boldsymbol{\theta}_i^* = \boldsymbol{\alpha}_s^*, i \in \widehat{\mathcal{S}}_s, 1 \leqslant s \leqslant \widehat{S}$. So far, we have estimated all the required variables and we can output $\widehat{\mathcal{S}}$, $\widehat{\boldsymbol{\beta}}$ and $\widehat{\boldsymbol{\alpha}}$ as the results.

3.3.3. Time Complexity

DiffS contains four major computation steps. To simplify the expressions, here we assume all the domains have the same number of samples, which is $n_i = n, \forall i$. We only concern about multiplication operations. The calculation of domain GLS estimates has $O(Mn^2(p+q))$ complexity. Each of the standardized domain differences costs $O(q^3)$. Thus, the whole distance matrix calculation has $O(M^2q^3)$ complexity. Notice that both the domain GLS estimator and the DiffS estimator only consist of matrix multiplication and inversion. Their calculation can be accelerated by GPU or other multi-thread methods in order to further reduce the time cost.

4. Theoretical Analysis

To theoretically evaluate the effectiveness of our proposed method, the following assumption is needed:

Assumption 4.1 (Subgroup differentiation)**.** Denoting $\mathcal{S}$ as the true learnability structure and $S = |\mathcal{S}|$, there exists $0 < \lambda_-^* < \lambda_+^*$ that satisfies $\forall s, 1 \leqslant s \leqslant S$, $\forall i, j \in \mathcal{S}_s, \max_{i,j\in\mathcal{S}_s}\Delta_{ij} < \lambda_-^*$, and $\exists i \in \mathcal{S}_s, \min_{j\notin\mathcal{S}_s}\Delta_{ij} > \lambda_+^*$.

Remark 4.2. The above assumption stands for the distinct differentiation of each pair of subgroups. It can be easily satisfied if the true parameters are far enough from each other. Under such assumption, we can guarantee the learnability structure recovery ability of DiffS.

Theorem 4.3 (Learnability structure recovery guarantee)**.** *Denoting $\mathcal{S}^*$ as the true learnability structure, supposing that the Assumption 4.1 is satisfied and learnability structure recovering is applied via Algorithm 2, thus $\widehat{\mathcal{S}} = \mathcal{S}^*$.*

Remark 4.4. See proof in Appendix B. This theorem shows that DiffS has the capability to perfectly recover the true learnability structure, which further guarantees the performance.

5. Related Works

5.1. Mutual Transfer Learning

There are some researches related with mutual transfer learning task that use different but similar hypotheses. They all propose their own solutions to the specific problem discussed in their papers. For example, (Kumar & Daume III, 2012) assumes that parameters are the linear combination of some vectors from a low dimensional subspace. (Thrun & O'Sullivan, 1996; Barzilai & Crammer, 2015; Jacob et al., 2008) model the parameters from different domains into clusters. (Agarwal et al., 2010) uses a manifold to describe the behaviour of parameters. Also, some researches thinks the parameters follow a prior distribution like Gaussian mixture model (Bakker & Heskes, 2003) or Dirichlet process (Xue et al., 2007). For those whose assumptions are consistent with our model (1), the methods can be mainly categorized into two types: optimization or statistics.

Optimization methods aim to minimize loss functions with regularization on parameters in order to estimate learnability structure. (Kumar & Daume III, 2012; Barzilai & Crammer, 2015) treat parameters as a linear combination of several low dimensional subgroup centers. (Han & Zhang, 2015) assumes the domains are constructed with multiple levels of subgroups and optimizes least square loss with a pairwise penalty to fuse parameters. (Cheng et al., 2020) further utilizes the confidence distribution of parameters to accelerate computation and proposes the CD Fusion method. Hyperparameter tuning and high computation complexity are the two main common problems within optimization methods.

Statistical methods begin from a set of prior distribution assumptions for parameters. (Yu et al., 2005; Bakker & Heskes, 2003) adopts an EM algorithm to obtain the hidden parameters of a Gaussian distribution or Gaussian mixture model. (Xue et al., 2007) puts effort into analyzing the Dirichlet process. Additionally, (Thrun & O'Sullivan, 1996) provides a simple framework for clustering based on the nearest neighbor algorithm. Although statistical methods have the advantage of simple implementation and fast computation, drawbacks also exist, including local minima troubles and improper distance metric problems. In addition, only a few of the mentioned methods can be adapted to the mixed-effects heterogeneity model discussed in this paper.

5.2. Multi-Task Learning and Clustering

The idea of multi-task learning (Caruana, 1997; Ben-David & Schuller, 2003; Evgeniou et al., 2005) is similar with mutual transfer learning. MALSAR (Zhou et al., 2011b) summarizes the common multi-task learning method with structure regularizations. Most of the methods solve the regularized optimization problem of the form $\min \mathcal{L}(\boldsymbol{W})+\Omega(\boldsymbol{W})$, which is identical to CD Fusion. The loss $\mathcal{L}$ is least squares or logistic. The regularization Ω could be Lasso (Tibshirani, 1996), Frobenius norm (Evgeniou & Pontil, 2004), $\ell_{1,2}$ norm (Argyriou et al., 2006), and their combinations (Gong et al., 2012; Zhou et al., 2011a; Jalali et al., 2010). However, DiffS does not contain the regularizer $\Omega(\boldsymbol{W})$ because it is originated from the probabilistic analysis for learnability sturcture recovery instead of loss regularization. This makes DiffS a tuning-free method.

Among the multi-task learning methods, there are a category of methods, called task clustering (Zhang & Yang, 2018; Ruder, 2018), that consider the hidden learnabiliy structure within the data and use common clustering approaches (Xu & Tian, 2015; Dafir et al., 2021) to figure out these subgroups. For example, k-means we use in the experiments is a typical clustering baseline. DD (Rodriguez & Laio, 2014) proposes a novel idea for the clustering center from the perspective of data point density. CURE (Guha et al., 1998) focuses on large-scale data settings and clusters the data by mini-batches. The learnability structure recovery function in DiffS is also a complete linkage based clustering algorithm, but it is deeply combined with the proposed standardization domain difference and our theoretical guarantees.

6. Experiments

We implement DiffS by Matlab, consistent with other baselines, and conduct experiments on a Linux server with an Intel Xeon Bronze 3204 CPU with up to 12 threads and 32GiB memory. These following baseline methods are involved in the experiments: **(1) CD Fusion (Cheng et al., 2020):** An optimization method that adds pairwise penalty on heterogeneous parameters, tuned by modified BIC. **(2) MeTaG (Han & Zhang, 2015):** A method that aims to extract multi-level learnability structure among domains. We set the level as 2 in experiments since the model (1) has a two-level structure. **(3) k-means with BIC:** Applying the k-means algorithm directly on domain GLS estimates $\boldsymbol{\theta}_i$ to obtain the learnability structure. The k is tuned via modified BIC as in (Cheng et al., 2020). The estimate of parameters is similar to the Step II of DiffS. **(4) S-means:** Directly providing the true number S of subgroups to k-means. We name this open-book approach as "S-means". It is only included in error comparisons.

6.1. Experiments on Synthetic Data

Data Preparation We generate 18 cases of synthetic datasets with different settings. In each case, we divide the M domains into S subgroups with at least one do-

main in each subgroup. Thus, the number of domains in these subgroups follows a multinomial distribution: $(M_1, M_2, \ldots, M_S)^\top \sim \mathbf{1}_{S\times 1} + \text{Multi}(\mathbf{1}_{S\times 1}/S, M-S)$. All M domains are randomly splited into training set, validation set and test set, with the proportion of $7:1:2$. For the true global parameters, each dimension is drawn from a uniform distribution $U(-2,2)$ independently. To ensure the true heterogeneous parameters are different enough between subgroups, we generate base values $\widetilde{\boldsymbol{\alpha}} = (\widetilde{\alpha}_1, \widetilde{\alpha}_2, \ldots, \widetilde{\alpha}_S)^\top$ from evenly spaced points in $[-S^{1.4}/2, S^{1.4}/2]$. Then the parameters are generated by $\boldsymbol{\alpha}_1^* = (\widetilde{\alpha}_1, \widetilde{\alpha}_2, \cdots, \widetilde{\alpha}_S)^\top$, $\boldsymbol{\alpha}_2^* = (\widetilde{\alpha}_S, \widetilde{\alpha}_1, \widetilde{\alpha}_2, \cdots, \widetilde{\alpha}_{S-1})^\top$, and so on. It means the next $\boldsymbol{\alpha}^*$ is the values of the previous one shifted by 1 dimension to the right. Meanwhile, if S is odd, we add 1 to all the element to avoid some of the parameters to be 0. For the design matrix $(\boldsymbol{X}_i, \boldsymbol{Z}_i)$, we let all domains have the same sample size n and generate each sample row from a normal distribution $\mathcal{N}(0, \boldsymbol{\Sigma}_D)$ where $\boldsymbol{\Sigma}_D = 0.3\cdot\mathbf{1}_{(p+q)\times(p+q)} + 0.7\boldsymbol{I}_{p+q}$. Random effects $\boldsymbol{u}$, ε are drawn from two normal distributions $\mathcal{N}(0, \boldsymbol{I}_q)$, $\mathcal{N}(0,2)$, respectively. The response is calculated via (1). In addition, we generate 20 replications for each setting case.

Metrics We use the following metrics to evaluate each methods: **(1) Wall-clock time:** The median real time cost while estimating is recorded for each case. We omit the time cost of the calculating domain GLS estimates since it is an initialization step for CD Fusion, k-means and DiffS and only need to calculate once. **(2) Parameter error:** We calculate the estimation error between the true parameters and the estimated parameters, represented as $\text{RMSE}(\boldsymbol{\theta}^*, \widehat{\boldsymbol{\theta}}) = \sum_{i=1}^{M} \|\boldsymbol{\theta}_i^* - \widehat{\boldsymbol{\theta}}_i\| / \sqrt{q}$ where $\widehat{\boldsymbol{\theta}}$ is the estimated parameters from each method. **(3) Subgroup number:** The mean number and standard deviation of the estimated subgroups is reported for each setting. **(4) Normalized mutual information:** We use the normalized mutual information (NMI) (Ana & Jain, 2003) to quantify how well the learnability structure estimates are. The normalizer is the mean of entropies of the two clusters. The higher value of NMI indicates a closer estimation of learnability structure. **(5) Prediction error:** We calculate the mean squared prediction error on test set to compare the performances of the methods for linear mixed-effects model regression tasks.

Settings In the base case settings, we set the number of domains $M = 50$, the number of subgroups $S = 5$, the sample size $n = 100$, the dimension $p = q = 10$. We vary domain size M, dimension $p+q$, sample size n, subgroup size S, one at a time. To show the impact of data scale, we vary M from 50 to 300 and n from 100 to 600. We set $\nu = 0.001$ for DiffS in all cases. For CD Fusion, the BIC tuning procedure optimizes the parameters for 5 times to find a better hyper-parameter. Due to its high time cost, we only run it on half of the replications. For k-means, we vary k from 1 to M and choose the best k with the least BIC value. For MeTaG, we tune on λ for 5 times and merge the same parameters from different domains (if possible) and generate subgroups based on that.

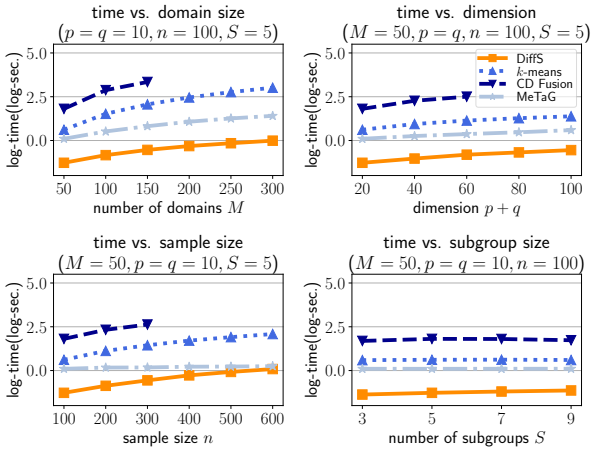


Figure 2. Time cost tendency of DiffS, k-means, CD Fusion and MeTaG. Wall-clock estimation time includes hyper-parameter tuning for k-means, CD Fusion and MeTaG. CD Fusion fails to estimate in some cases due to high time cost (above 3000 seconds).

Results: (a) Time cost comparison Figure 2 shows the time cost tendency of the methods when M, $p+q$, n or S grows. We use log-time to the base 10 as y axis. The CD Fusion method takes more than 3000 seconds per replication, whereas DiffS only takes about 0.2 second to estimate when the scale is 3 times larger than base case. This indicates that the CD Fusion method is infeasible in large-scale settings, so we discard it in some cases. Also, DiffS is about 50 to 1000 times faster than k-means and about 2 to 20 times faster than MeTaG among all the cases. The dimensionality of data influences the methods less. However, DiffS is still about 1200 times faster than CD Fusion, 100 times faster than k-means and 20 times faster than MeTaG. It proves that DiffS is feasible in large-scale settings and has the fastest estimation speed among the 5 methods.

Results: (b) Parameter error comparison Figure 3 illustrates the parameter estimation error of the 5 methods. We take the average result among all the replicates in each experiment setting. DiffS performs better with respect to mean RMSE than the baselines in most of the cases. The k-means based methods estimate parameters poorly since they generate bad learnability structure estimates. We notice that the baselines has their "comfort zone" that in some settings they behave well while in other settings they generate meaningless estimates. This may explain the zig-zag lines in the figure. For example, MeTaG performs well in most

Table 1. Learnability structure estimation performance of DiffS, k-means, S-means, CD Fusion and MeTaG. DiffS (w/o) refers to DiffS without standardization. The discussion about it is in the Ablation Study paragraph. We calculate the mean of $\widehat{S}$ and NMI in all the replicates of one setting. CD Fusion method are only run on half of the replications due to high time cost and it fails to apply in some cases with "N/A". The bold-type number indicates the best performance out of the 5 methods.

Data settings					Methods									
M	n	p	q	S	DiffS		k-means + BIC		S-means	CD Fusion (10 Reps)		MeTaG		DiffS (w/o)	
					$\widehat{S}\pm$std	NMI	$\widehat{S}\pm$std	NMI	NMI	$\widehat{S}\pm$std	NMI	$\widehat{S}\pm$std	NMI	$\widehat{S}\pm$std	NMI
50	100	10	10	3	3.20 ± 0.41	0.993	3.15 ± 0.49	0.970	0.950	$\mathbf{3.00 \pm 0.00}$	**1.000**	35.00 ± 0.00	0.476	8.25 ± 1.12	0.720
50	100	10	10	5	5.10 ± 0.31	0.998	5.15 ± 1.09	0.941	0.912	$\mathbf{5.00 \pm 0.00}$	**1.000**	35.00 ± 0.00	0.628	10.10 ± 1.33	0.851
50	100	10	10	7	$\mathbf{7.00 \pm 0.32}$	**0.998**	7.90 ± 1.29	0.945	0.902	4.40 ± 2.84	0.621	35.00 ± 0.00	0.708	12.50 ± 1.64	0.891
50	100	10	10	9	$\mathbf{9.00 \pm 0.00}$	**1.000**	8.00 ± 1.81	0.925	0.902	1.00 ± 0.00	0.000	35.00 ± 0.00	0.764	13.60 ± 1.27	0.930
50	100	30	30	5	$\mathbf{5.15 \pm 0.37}$	**0.997**	1.15 ± 0.67	0.047	0.890	1.20 ± 0.63	0.036	35.00 ± 0.00	0.628	18.50 ± 1.85	0.729
50	100	50	50	5	$\mathbf{5.10 \pm 0.31}$	**0.954**	1.00 ± 0.00	0.000	0.603	N/A	N/A	35.00 ± 0.00	0.614	35.00 ± 0.00	0.611
50	300	10	10	5	$\mathbf{5.15 \pm 0.49}$	**0.992**	5.35 ± 1.09	0.948	0.893	4.80 ± 0.63	0.977	35.00 ± 0.00	0.626	10.75 ± 1.21	0.834
50	600	10	10	5	$\mathbf{5.05 \pm 0.22}$	**0.998**	6.75 ± 1.68	0.938	0.937	N/A	N/A	35.00 ± 0.00	0.627	10.25 ± 1.21	0.845
150	100	10	10	5	6.05 ± 0.89	0.987	5.65 ± 0.75	0.978	0.902	$\mathbf{5.00 \pm 0.00}$	**1.000**	105.00 ± 0.00	0.529	18.90 ± 1.97	0.735
300	100	10	10	5	6.85 ± 1.23	**0.980**	$\mathbf{5.95 \pm 0.89}$	0.967	0.942	N/A	N/A	210.00 ± 0.00	0.475	28.30 ± 1.72	0.677

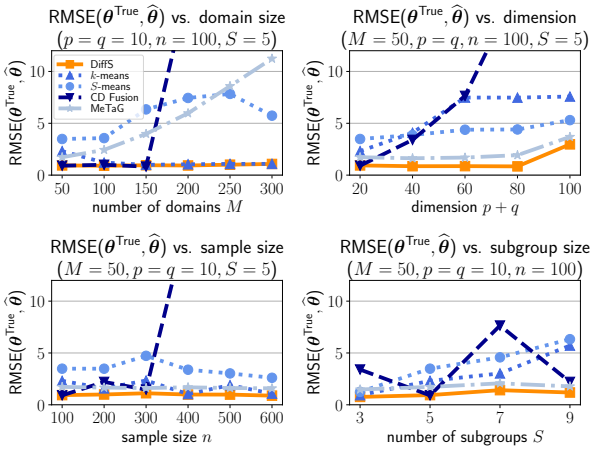


Figure 3. The parameter estimation error comparisons of the 5 methods. The experiment settings are the same as 2. We draw lines of CD Fusion method outside the figure to mean that it is not applicable in the following cases due to high time cost.

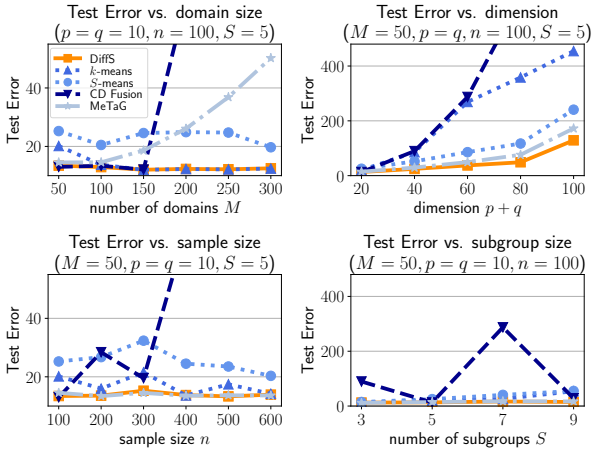


Figure 4. The test set prediction error comparisons of the 5 methods. We draw lines outside the figure for infeasible cases of the method.

of the cases but behaves badly with high domain size. DiffS mantains a smooth curve, showing a stable performance.

Results: (c) Learnability structure estimation Table 1 compares the performance of learnability structure estimation. DiffS shows sound ability in learnability structure recovery with almost all correct estimates. k-means has much practicability in some proper feature dimension settings, but it still behaves worse than DiffS. CD Fusion cannot generate valid estimates when S increases. MeTaG can hardly provide useful learnability structure since the parameters it estimates are always different among domains. We speculate that both k-means and CD Fusion behave poorly because the hyper-parameter tuning is difficult in complicated cases. Additionally, S-means generate nice NMI results while parameter estimations are terrible.

Results: (d) Prediction error Figure 4 shows the comparison of prediction error on test set. DiffS gives the smallest or second smallest test errors among most of the cases. Generally, with good enough learnability structure estimate, methods can provide nice predictions on test set, while poor subgroup estimates can lead to a disaster in test error.

Ablation Study: Raw domain difference and GLS estimates In the ablation study, we analyze two more baselines to show the effectiveness of DiffS: (1) a variant of DiffS that do not standardize the domain differences (i.e., $\boldsymbol{\delta}_{ij} = \boldsymbol{\theta}_i - \boldsymbol{\theta}_j$, DiffS (w/o) for short). (2) original GLS estimates (i.e., $\boldsymbol{\beta}^{\mathcal{D}}$, $\boldsymbol{\theta}_i^{\mathcal{D}}$). The learnability structure estimate performance of DiffS (w/o) is listed in the last two columns of Table 1. Figure 5 shows the parameter estimation performance comparisons of DiffS and the two baselines. Comparing DiffS (w/o) and GLS, learnability structure does help reduce the noise within heterogeneous parameter estimates since knowledge can be transferred in the subgroup. From

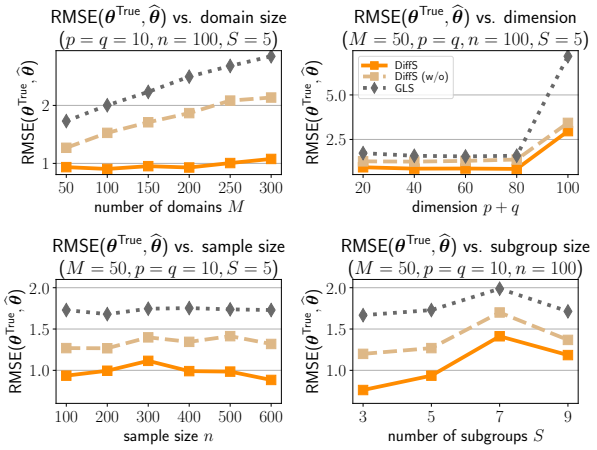


Figure 5. The parameter estimation error comparisons of DiffS, DiffS without standardization (referred to as DiffS (w/o) in figure) and raw GLS estimates. The results of DiffS is the same as in Figure 3 and is used here just for ablation study.

Table 2. Comparison of DiffS, k-means and MeTaG in real-world datasets. Time costs are recorded in seconds.

Method	**NOAA nClimDiv Database**		
	Timecost	S	MSE
DiffS	33.79	7.10 ± 0.32	82.44 ± 3.44
k-means	119.73	4.40 ± 0.52	88.72 ± 8.91
MeTaG	32.49	3.10 ± 1.60	$> 10^{200}$
	Microarray Data		
DiffS	0.0169	2.10 ± 0.32	0.93 ± 0.14
k-means	0.2677	1.00 ± 0.00	0.96 ± 0.13
MeTaG	0.2988	14.00 ± 0.00	0.90 ± 0.14

the comparison of DiffS and DiffS (w/o) in the Table 1, there is a clear performance drop with regards to the learnability structure estimates. This indicates that the standardized domain difference can help distinguish the domains from or not from the same subgroup. The parameter estimation error drop shown in Figure 5 also illustrates the benefit of a better learnability structure estimate.

6.2. Experiment on Real Datasets

We apply DiffS on the NOAA nClimDiv Database[1] (Vose et al., 2014) and Microarray Data (Wille et al., 2004) together with baseline k-means and MeTaG.

In the NOAA nClimDiv Database, we estimate the monthly average temperature. Palmer Drought Severity Index (PDSI), Palmer Hydrological Drought Index (PHDI), precipitation (PCPN) and Palmer Z Index (ZNDX) are chosen as features. Three dummy variables for Summer (June, July, August), Fall (September, October, November) and Winter (December, January, February) are added to the features. Spring (March, April, May) feature is learned within the intercept term. Following (Cheng et al., 2020), intercept, PCPN and ZNDX is chosen to be the heterogenous features ($q = 3$) and the rests are global features ($p = 5$). In the database, data have been collected monthly from 344 divisions for 126 years. The monthly average temperature prediction problem contains $M = 344$ domains and $n = 1512$ samples each domain.

The Microarray Data is a gene expression dataset with microarray data related to isoprenoid biosynthesis in plant organism. 18 genes in the plastidial pathway (18 domains) are expected to be expressed by the 21 genes in the mevalonate pathway (21 dimensions). In each data domain, there are 118 samples ($n = 118$). Heterogeneous features are selected via the same way as described by (Cheng et al., 2020) in their NOAA experiment. We choose 8 features for global and 12 for heterogeneous ($p = 8, q = 12$).

We randomly split the domains into $7 : 1 : 2$ for training, validation and testing. We generate 10 splits to reduce randomness. In k-means, the k is limited in $[1, 5]$. We try some value of ν in DiffS for each dataset and use a fixed value in experiment. The result is summarized in Table 2. A case study on NOAA is in the Appendix C. CD Fusion is not included because it requires too much memory (about 1000GiB) in NOAA database, while DiffS only requires about 10GiB in this experiment. Notice that the time cost of k-means is limited and if we tune k from 1 to 344, the timecost could reach 32000 seconds. In NOAA Database, MeTaG generates bad estimates which make MSE explode. DiffS is able to control the estimation timecost in an acceptable range and meanwhile provides reasonable estimates. These experiments show that DiffS is scalable and applicable in large scale settings and real-world applications.

Acknowledgements

This work was supported by the National Key R&D Program of China [Grant No. 2018AAA0100500]; National Natural Science Foundation of China [Grant No. 61906040]; the Natural Science Foundation of Jiangsu Province [Grant Numbers BK20190335, BK20190345]; National Natural Science Foundation of China [Grant Numbers 61906037, 61972085]; the Fundamental Research Funds for the Central Universities [2242021R41084]; Jiangsu Provincial Key Laboratory of Network and Information Security [Grant No. BM2003201], Key Laboratory of Computer Network and Information Integration of Ministry of Education of China [Grant No. 93K-9].

[1] https://www.ncei.noaa.gov/access/metadata/landing-page/bin/iso?id=gov.noaa.ncdc:C00005

References

Aaker, D. A., Kumar, V., Leone, R. P., and Day, G. S. *Marketing research: International student version*. John Wiley & Sons New York, NY, 2013.

Agarwal, A., Gerber, S., and Daume, H. Learning multiple tasks using manifold regularization. In *Advances in neural information processing systems*, pp. 46–54, 2010.

Ana, L. and Jain, A. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pp. II–II. IEEE, 2003.

Anh, V. and Chelliah, T. Estimated generalized least squares for random coefficient regression models. *Scandinavian journal of statistics*, 26(1):31–46, 1999.

Argyriou, A., Evgeniou, T., and Pontil, M. Multi-task feature learning. In *Advances in neural information processing systems*, pp. 41–48, 2006.

Bakker, B. and Heskes, T. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research*, 4(May):83–99, 2003.

Barzilai, A. and Crammer, K. Convex multi-task learning by clustering. In *Artificial Intelligence and Statistics*, pp. 65–73. PMLR, 2015.

Ben-David, S. and Schuller, R. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pp. 567–580. Springer, 2003.

Caruana, R. Multitask learning. *Machine learning*, 28(1): 41–75, 1997.

Cheng, C.-W., Qiao, X., and Cheng, G. Mutual transfer learning for massive data. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1800–1809, 2020.

Dafir, Z., Lamari, Y., and Slaoui, S. C. A survey on parallel clustering algorithms for big data. *Artificial Intelligence Review*, 54(4):2411–2443, 2021.

Evgeniou, T. and Pontil, M. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 109–117, 2004.

Evgeniou, T., Micchelli, C. A., and Pontil, M. Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(Apr):615–637, 2005.

Gong, P., Ye, J., and Zhang, C. Robust multi-task feature learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 895–903, 2012.

Guha, S., Rastogi, R., and Shim, K. Cure: An efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84, 1998.

Han, L. and Zhang, Y. Learning multi-level task groups in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, pp. 2638–2644, 2015.

He, X. and Shao, Q.-M. On parameters of increasing dimensions. *Journal of multivariate analysis*, 73(1):120–135, 2000.

Jacob, L., Vert, J.-p., and Bach, F. Clustered multi-task learning: A convex formulation. *Advances in neural information processing systems*, 21:745–752, 2008.

Jalali, A., Sanghavi, S., Ruan, C., and Ravikumar, P. A dirty model for multi-task learning. *Advances in neural information processing systems*, 23, 2010.

Kumar, A. and Daume III, H. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1723–1730, 2012.

Lenk, P. J., DeSarbo, W. S., Green, P. E., and Young, M. R. Hierarchical bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.

Richardson, A. and Welsh, A. H. Asymptotic properties of restricted maximum likelihood (reml) estimates for hierarchical mixed linear models. *Australian Journal of statistics*, 36(1):31–43, 1994.

Rodriguez, A. and Laio, A. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.

Ruder, S. An overview of multi-task learning in deep neural networks. *National Science Review*, 5(1):30–43, 2018.

Thrun, S. and O'Sullivan, J. Discovering structure in multiple learning tasks: The tc algorithm. In *International Conference on Machine Learning*, volume 96, pp. 489–497, 1996.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Vose, R. S., Applequist, S., Squires, M., Durre, I., Menne, M. J., Williams, C., and Arndt, D. Noaa's gridded climate divisional dataset (climdiv). *NOAA National Climatic Data Center*, 2014. doi: 10.7289/V5M32STR.

Wille, A., Zimmermann, P., Vranová, E., Fürholz, A., Laule, O., Bleuler, S., Hennig, L., Prelić, A., von Rohr, P., Thiele, L., et al. Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome biology*, 5(11):1–13, 2004.

Xu, D. and Tian, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

Yu, K., Tresp, V., and Schwaighofer, A. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, pp. 1012–1019, 2005.

Zhang, Y. and Yang, Q. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018.

Zhou, J., Chen, J., and Ye, J. Clustered multi-task learning via alternating structure optimization. *Advances in neural information processing systems*, 24, 2011a.

Zhou, J., Chen, J., and Ye, J. Malsar: Multi-task learning via structural regularization. *Arizona State University*, 2011b.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

A. Properties of Standardized Domain Difference

A.1. Proof for Equation (9)

Recall that eqution (9) reads

$$\boldsymbol{\delta}_{ij} = (\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)^{-1/2}(\boldsymbol{\theta}_i^{\mathcal{D}} - \boldsymbol{\theta}_j^{\mathcal{D}}) \sim \begin{cases} \mathcal{N}(\mathbf{0}, \boldsymbol{I}), & i, j \text{ in the same subgroup,} \\ \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{I}), & \text{otherwise.} \end{cases} \tag{9}$$

Proof. The domain GLS estimators follow (He & Shao, 2000; Cheng et al., 2020)

$$(\boldsymbol{\beta}_i^{\mathcal{D}}, \boldsymbol{\theta}_i^{\mathcal{D}})^\top \sim \mathcal{N}((\boldsymbol{\beta}_i^*, \boldsymbol{\theta}_i^*)^\top, [\boldsymbol{G}_i^\top \boldsymbol{W}_i \boldsymbol{G}_i]^{-1}). \tag{12}$$

So that

$$\boldsymbol{\theta}_i^{\mathcal{D}} \sim \mathcal{N}(\boldsymbol{\theta}_i^*, \boldsymbol{\Sigma}_i), \tag{13}$$

where $\boldsymbol{\Sigma}_i = (\mathbf{0}_{q\times p}, \boldsymbol{I}_q)[\boldsymbol{G}_i^\top \boldsymbol{W}_i \boldsymbol{G}_i]^{-1}(\mathbf{0}_{q\times p}, \boldsymbol{I}_q)^\top$. Thus, for any $i \neq j$, since $\boldsymbol{\theta}_i^{\mathcal{D}}$ is independent from $\boldsymbol{\theta}_j^{\mathcal{D}}$,

$$\boldsymbol{\theta}_i^{\mathcal{D}} - \boldsymbol{\theta}_j^{\mathcal{D}} \sim \mathcal{N}(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_j^*, \boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j), \tag{14}$$

and

$$\boldsymbol{\delta}_{ij} \sim \begin{cases} \mathcal{N}(\mathbf{0}, \boldsymbol{I}), & i, j \text{ in the same subgroup,} \\ \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{I}), & \text{otherwise,} \end{cases} \tag{15}$$

where $\boldsymbol{\mu}_{ij} = (\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)^{-1/2}(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_j^*)$. □

A.2. The Distribution of the Distance Between Domains

The distribution of Δ_{ij} can be derived from equation (9).

Corollary A.1. *The elements in the similarity matrix $\boldsymbol{\Delta}$ satisfies*

$$\begin{aligned} \Delta_{ij} &= \|\boldsymbol{\delta}_{ij}\|^2 \\ &\sim \begin{cases} \chi_q^2, & i, j \textit{ in the same subgroup,} \\ \chi_q'^2(\|\boldsymbol{\mu}_{ij}\|^2), & \textit{otherwise,} \end{cases} \end{aligned} \tag{16}$$

where χ_q^2 denotes the χ^2-distribution with q degrees of freedom, $\chi_q'^2(\|\boldsymbol{\mu}_{ij}\|^2)$ denotes the noncentral χ^2-distribution with q degrees of freedom and non-centrality parameter $\|\boldsymbol{\mu}_{ij}\|^2$ and $\boldsymbol{\mu}_{ij} = (\boldsymbol{\Sigma}_i + \boldsymbol{\Sigma}_j)^{-1/2}(\boldsymbol{\theta}_i^* - \boldsymbol{\theta}_j^*)$.

Proof. Note that the standardized domain difference

$$\boldsymbol{\delta}_{ij} \sim \begin{cases} \mathcal{N}(\mathbf{0}, \boldsymbol{I}), & i, j \text{ in the same subgroup,} \\ \mathcal{N}(\boldsymbol{\mu}_{ij}, \boldsymbol{I}), & \text{otherwise.} \end{cases} \tag{17}$$

Provided domain i, j are in the same subgroup,

$$\Delta_{ij} = \sum_{k=1}^{q} Z_k^2,$$

where $Z_k \sim \mathcal{N}(0, 1)$ i.i.d.. Thus according to the definition of χ^2-distribution,

$$\Delta_{ij} \sim \chi_q^2. \tag{18}$$

Similarily, provided domain i, j are from two different subgroups,

$$\Delta_{ij} = \sum_{k=1}^{q} Z_k'^2,$$

where $Z_k'^2 \sim \mathcal{N}(\mu_k, 1)$. Here μ_k is the k-th element of $\boldsymbol{\mu}_{ij}$. Follow the definition of noncentral χ^2-distribution,

$$\Delta_{ij} \sim \chi_q'^2(\|\boldsymbol{\mu}_{ij}\|^2). \tag{19}$$

□

A.3. Choice of ν

There is only one tunable hyper-parameter ν in DiffS. No parameter tuning work will exist as long as users leave it a fix number. Synthetic experiments also prove that when we set it to 0.001, DiffS behaves stably well. However, the change in ν is also acceptable and we will discuss the impact it has on the performance of DiffS.

From Appendix A.2 we know

$$\Delta_{ij} = \|\boldsymbol{\delta}_{ij}\|^2 \sim \begin{cases} \chi_q^2, & i,j \text{ in the same subgroup,} \\ \chi_q'^2(\|\boldsymbol{\mu}_{ij}\|^2), & \text{otherwise.} \end{cases} \tag{20}$$

The optimal threshold value can be calculated via Bayesian classifiers where the true parameters are known. When dealing with real-world problems, the true parameters are unknown, which makes the optimal threshold unavailable. In DiffS we propose to set the threshold to a fixed value in any setting of data, namely $\lambda = F_q^{-1}(\nu)$. Thus provided $\nu = 0.001$, there is a probability of 99.9% that $\Delta_{ij} \leqslant \lambda$ if i, j come from the same subgroup when $\boldsymbol{G}_i$ vary. ν controls the probability that the two domains in the same subgroup would have zero distance. If ν is set to be larger, there is less probability that the data domains in the same subgroup will have zero distances, which may result in the disassembly of some big subgroups. If ν is tuned to be much smaller, there is a chance that more data domains from different subgroups will have zero distances, leading to misclustering.

It can be inferred that as the domain size M grows, the outliers of Δ_{ij} within the same subgroup appear more frequently. In order to include these outliers as much as possible, it is better to decrease ν (i.e., increase threshold λ) when dealing with large M data.

B. Proof for Theorem 4.3

First we restate Assumption 4.1 into a proposition.

Proposition B.1. *The estimated subgroup in each iteration of Algorithm 2 satisfies*

- $\forall s, t$, *if* $\exists s', \widehat{\mathcal{S}}_s \cup \widehat{\mathcal{S}}_t \subseteq \mathcal{S}_{s'}$, *then* $\Delta_{st} \leqslant \lambda$.
- $\forall u'$,$\exists u$ *such that,* $\widehat{\mathcal{S}}_u \subseteq \mathcal{S}_{u'}$ *and* $\forall t$ *if* $\widehat{\mathcal{S}}_t \cap \mathcal{S}_{u'} = \varnothing$, *then* $\Delta_{ut} > \lambda$.

Then, we consider the lemma below:

Lemma B.2. *In each iteration of Algorithm 2,*

$$\forall s, \exists s' \text{ such that } \widehat{\mathcal{S}}_s \subseteq \mathcal{S}_{s'}, \tag{21}$$

and Proposition B.1 holds, as long as the Assumption 4.1 is satisfied and we take $\lambda \in (\lambda_-^*, \lambda_+^*)$ *as the threshold in Algorithm 2.*

Proof. We use Mathematical Induction (MI) to prove the lemma.

Base case: For the initial subgroup estimate $\widehat{\mathcal{S}} = \{\{1\}, \{2\}, \ldots, \{M\}\}$, since $\mathcal{S}$ is a partition of $\{1, 2, \ldots, M\}$, it is clearly that $\forall s, \exists s', \widehat{\mathcal{S}}_s \subseteq \mathcal{S}_{s'}$. Also, since we have not modify the similarity matrix $\boldsymbol{\Delta}$, the Assumption 4.1 naturally meets Proposition B.1.

Inductive step: We are going to show that for any iteration number m that $0 \leqslant m < M - 1$ ($m = 0$ refers to the base case), if in iteration #m (21) holds and Proposition B.1 is met, in iteration #$(m+1)$ it also holds and Proposition B.1 is also met.

Firstly, we are going to prove that the u found by Algorithm 2 Line #9 is exactly the u described by the second item of Proposition B.1. According to (21), $\forall s, \exists s', \widehat{\mathcal{S}}_s \subseteq \mathcal{S}_{s'}$. Due to the first item of Propsition B.1, $\forall s$, if $\exists j \neq s$ such that $\Delta_{sj} \leqslant \lambda$, there is

$$\mathcal{M}_s' = \{j \mid \Delta_{sj} \leqslant \lambda\} \supseteq \mathcal{M}_{s'} = \{j \mid \widehat{\mathcal{S}}_j \subseteq \mathcal{S}_{s'}\}, \tag{22}$$

where $\mathcal{M}_s'$ stands for the set of j such that $\Delta_{sj} \leqslant \lambda$, $\mathcal{M}_{s'}$ stands for the set of j such that $\widehat{\mathcal{S}}_j \subseteq \mathcal{S}_{s'}$. According to the second item of Proposition B.1, there exists u such that $\forall t, \widehat{\mathcal{S}}_t \cap \mathcal{S}_{u'} = \varnothing \rightarrow \Delta_{ut} > \lambda$. It is easy to prove by contradiction

that (22) acquires the equal sign if and only if $s = u$, which means $\mathcal{M}'_u = \mathcal{M}_{u'}$. Let $\bar{u}$ be the u found by Algorithm 2 Line #9. If $\mathcal{M}'_{\bar{u}} \neq \mathcal{M}_{\bar{u}'}$ (or equivalently $\mathcal{M}'_{\bar{u}} \supsetneqq \mathcal{M}_{\bar{u}'}$), according to the second item of Proposition B.1, for the $\bar{u}'$ corresponding to $\bar{u}$, there exists u such that $\mathcal{M}'_u = \mathcal{M}_{\bar{u}'}$. Therefore, $\sum_{j\neq\bar{u}} \mathcal{I}(\Delta_{\bar{u}j} \leqslant \lambda) > \sum_{j\neq u} \mathcal{I}(\Delta_{uj} \leqslant \lambda)$, which means $\bar{u}$ is not the minimal point for $\sum_{j\neq i} \mathcal{I}(\Delta_{ij} \leqslant \lambda)$, leading to contradiction. Thus, we proved that $\mathcal{M}'_{\bar{u}} = \mathcal{M}_{\bar{u}'}$. As a result, the u found by Algorithm 2 Line #9 is exactly the u described by the second item of Proposition B.1. In fact, the u found by algorithm is the u of the smallest subgroup described by the second item of Proposition B.1.

Secondly, we are going to prove (21) holds in iteration #$(m+1)$. Apply (21) on the $\widehat{\mathcal{S}}_u$ we find in the iteration #$(m+1)$, and we denote $\mathcal{S}_{s'}$ such that $\widehat{\mathcal{S}}_u \subseteq \mathcal{S}_{s'}$. Since Proposition B.1 is met, the $\widehat{\mathcal{S}}_u$ should satisfy that $\forall j$,

$$\Delta_{uj} \leqslant \lambda \rightarrow \widehat{\mathcal{S}}_u \cup \widehat{\mathcal{S}}_j \subseteq \mathcal{S}_{s'}. \tag{23}$$

Here "$\rightarrow$" means logical implication. Thus, the v found in the iteration #$(m+1)$ satisfies

$$\widehat{\mathcal{S}}_u \cup \widehat{\mathcal{S}}_v \subseteq \mathcal{S}_{s'}, \tag{24}$$

which means $\widehat{\mathcal{S}}_t = \widehat{\mathcal{S}}_u \cup \widehat{\mathcal{S}}_v \subseteq \mathcal{S}_{s'}$. Since we only merge $\widehat{\mathcal{S}}_u$ and $\widehat{\mathcal{S}}_u$ in this iteration, we have proven that (21) holds in iteration #$(m+1)$.

Lastly, we are going to prove that Proposition B.1 is true in iteration #$(m+1)$. For all $r \neq u, v$, without loss of generality, there are only possible two situations for $\widehat{\mathcal{S}}_r$ before merging the two subgroups:

- $\widehat{\mathcal{S}}_r \cup \widehat{\mathcal{S}}_u \subseteq \mathcal{S}_{u'}$. Thus $\widehat{\mathcal{S}}_v \cup \widehat{\mathcal{S}}_r \subseteq \mathcal{S}_{s'}$ and $\widehat{\mathcal{S}}_t \cup \widehat{\mathcal{S}}_r \subseteq \mathcal{S}_{s'}$. As a result, $\Delta_{t,r} = \max(\Delta_{u,r}, \Delta_{v,r}) \leqslant \lambda$. Therefore the first item of Proposition B.1 is also true after merging.
- $\widehat{\mathcal{S}}_r \cap \mathcal{S}_{u'} = \varnothing$, and thus $\Delta_{u,r} > \lambda$. Since $\widehat{\mathcal{S}}_t \subseteq \mathcal{S}_{u'}$, and $\Delta_{t,r} \geqslant \Delta_{u,r} > \lambda$, the second item of Proposition B.1 is also true after merging. In fact, $\widehat{\mathcal{S}}_t$ takes the place of $\widehat{\mathcal{S}}_u$.

As a result, Proposition B.1 holds in iteration #$(m+1)$.

Conclusion: Since both the base case and the inductive step are proven as true, Lemma B.2 holds.

□

When Algorithm 2 stops, the $\mathbf{\Delta}$ is either a 1×1 matrix (actually it is 0) or with all off-diagonal elements larger than λ. If $\mathbf{\Delta}$ turns out to be 0, it means that all the domains are clustered into a single subgroup, which reflects the ground truth since that $\widehat{\mathcal{S}}_1 = \{1, 2, \ldots, M\} \subseteq \mathcal{S}_1 \subseteq \{1, 2, \ldots, M\} \Rightarrow \widehat{\mathcal{S}}_1 = \mathcal{S}_1 = \{1, 2, \ldots, M\}$. Otherwise, $\forall s, t$ we assume that $\widehat{\mathcal{S}}_s \subseteq \mathcal{S}_{s'}$ and $\widehat{\mathcal{S}}_t \subseteq \mathcal{S}_{t'}$. Since $\Delta_{st} > \lambda$, one knows that

$$\forall u', \widehat{\mathcal{S}}_s \cup \widehat{\mathcal{S}}_t \nsubseteq \mathcal{S}_{u'}. \tag{25}$$

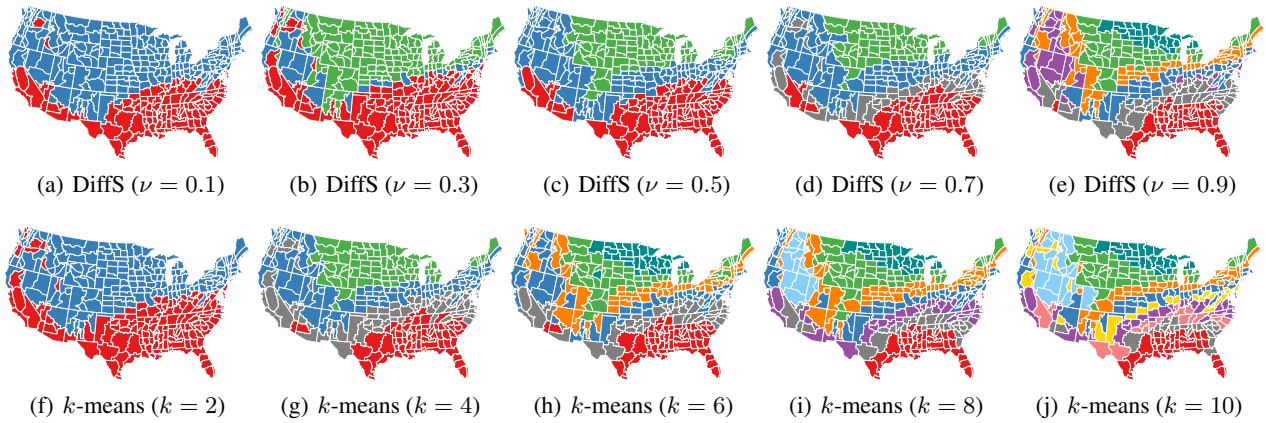
Because that $\mathcal{S}$ is a partition of $\{1, 2, \ldots, M\}$, we can infer that $\mathcal{S}_{s'} \cap \mathcal{S}_{t'} = \varnothing$. Thus, for all s', there exists only one s such that $\widehat{\mathcal{S}}_s \subseteq \mathcal{S}_{s'}$. Since $\widehat{\mathcal{S}}$ is also a partition of $\{1, 2, \ldots, M\}$, then $\widehat{\mathcal{S}}_s = \mathcal{S}_{s'}$, which means $\widehat{\mathcal{S}} = \mathcal{S}$. Thus, Theorem 4.3 is proved.

C. Case Study: NOAA nClimDiv Database

The subgroup estimation maps are shown in Figure 6. We vary ν in DiffS from 0.1 to 0.9 and k in k-means from 2 to 10. As an example, DiffS estimates that there are 3 main subgroups under the setting of $\nu = 0.5$. Notice that in the subgroup estimation we do not use any geographical information. Compared with climate zones defined by IECC[2], the pattern of the blue subgroup matches zone 5 perfectly, the red subgroup consists of zone 2, 3, 4, and the green subgroup contains zone 6 and 7. Also, the shape of the 3 subgroups is approximately in accord with the US temperature outlook[3]: the blue and red subgroups have the shape of areas that tend to be warmer and the green region corresponds to the areas that have equal chances to be hotter or cooler. This shows that the learnability structure estimate from DiffS has pratical value to a certain extent.

[2]https://up.codes/viewer/nevada/iecc-2012/chapter/CE_3/ce-general-requirements#C301.1

[3]The first figure in https://www.climate.gov/news-features/videos/noaas-2019-20-winter-outlook-temperature-precipitation-and-drought.



(a) DiffS ($\nu = 0.1$) (b) DiffS ($\nu = 0.3$) (c) DiffS ($\nu = 0.5$) (d) DiffS ($\nu = 0.7$) (e) DiffS ($\nu = 0.9$)

(f) k-means ($k = 2$) (g) k-means ($k = 4$) (h) k-means ($k = 6$) (i) k-means ($k = 8$) (j) k-means ($k = 10$)

Figure 6. Subgroup estimation result maps from DiffS and k-means for the average temperature estimation task in NOAA nClimDiv Database.

k-means generates similar learnability structure estimations with DiffS. It is consistent with the behaviour of k-means in synthetic experiments that it generates moderate estimations. However, the prediction error of k-means is larger than DiffS because the unstable clustering results. CD Fusion method mentioned in the paper cannot handle such large-scale data using the algorithm described in their original paper due to extremely high memory cost. In addition, even if the memory requirement is met, CD Fusion still would take hours to estimate. DiffS, however, could complete grid search on ν in 70s. The selection of k in k-means is quite random and subjective, and full tuning of k takes very long time. These disadvantages of baselines make DiffS a better approach in such real-world applications.