Transfer Learning under High-dimensional Generalized Linear Models Notes

19331053 纪传宇

2022 年 7 月 22 日

1 INTRODUCTION

Transfer learning can be promising in the high-dimensional data analysis, where the sample size is much less than the dimension with some sparsity structure in the data

(1) Propose multi-source transfer learning algorithms on generalized linear models (GLMs) and assume both target and source data to be high-dimensional. We assume the contrast between target and source coefficients to be '1-sparse'. The theoretical analysis shows that when the target and source are sufficiently close to each other, the estimation error bound of target coefficients could be improved over that of the classical penalized estimator using only target data under mild conditions. Moreover, the error rate is shown to be minimax optimal under certain conditions.

(2)Negative transfer is that some sources might be far away from the

target, and transferring them can be harmful. To avoid this issue, we develop an algorithm-free transferable source detection algorithm, which can help identify informative sources. And with certain conditions satisfied, the algorithm is shown to be able to distinguish useful sources from useless ones

(3) propose an algorithm on the basis of our two-step transfer learning procedure and nodewise regression to construct the confidence interval for each coefficient component. The corresponding asymptotic theories are established

2 Notation

2.1 title

表 1: Notation for Trans-Lasso algorithm

Indices	
$X^{(0)} \in R^{n_0 * p}$	design matirx for the primary data
$Y^{(0)} \in R^{n_0}$	response vextor for the primary data
$X^{(k)} \in R^{n_k * p}$	design matrix for the kth auxiliary data
$Y^{(k)} \in R^{n_k}$	the response vector for the kth auxiliary data
$\ell_q - norm$	$\|\boldsymbol{x}\| = (\sum_{i=1}^{p} \|x_i\|^q)^{1/q} \, (q \in (0, 2])$
$\ell_0 - norm$	$\|\boldsymbol{x}\|_0 = \#\{j : x_j \neq 0\}$
For a matrix $\boldsymbol{A}_{p \times q} = [a_{ij}]_{p \times q}$	
$\|\boldsymbol{A}\|_1$	$\sup_j \sum_{i=1}^{p} \|a_{ij}\|$
$\|\boldsymbol{A}\|_2$	$\max_{\boldsymbol{x}:\|\boldsymbol{x}\|_2=1} \|\boldsymbol{A}\boldsymbol{x}\|_2$
$\|\boldsymbol{A}\|_\infty$	$\sup_i \sum_{j=1}^{q} \|a_{ij}\|$
$\|\boldsymbol{A}\|_{\max}$	$\sup_{i,j} \|a_{ij}\|$
$n_{\mathcal{A}_q}$	$\sum_{k \in \mathcal{A}_q} n_k$
$\Lambda_{\max}(\Sigma)$	largest eigenvalues of Σ respectively
$\Lambda_{\min}(\Sigma)$	smallest eigenvalues of Σ respectively
$a \vee b$	$\max\{a, b\}$
$a \wedge b$	$\min\{a, b\}$
$a_n = O(b_n)$ and $a_n \lesssim b_n$	$\|a_n / b_n\| \leq c$for some constant c when n is large enough
$a_n \asymp b_n$	$\|a_n / b_n\| \to c$ for some constant c as $n \to \infty$
$a_n = O_P(b_n)$ and $a_n \lesssim_{\mathbb{P}} b_n$	$\mathbb{P}(\|a_n / b_n\| \leq c) \to 1$ for some constant $c < \infty$
$a_n = o_P(b_n)$	$(\|anbn\| > c) \to 0 for any constant c > \ 0$

3 Generalized linear models (GLMs)

Basic knowledge about GLMs:

3.1 Generalized Linear Models

A generalized linear model is made up of a linear predictor:

$$\eta_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi} \tag{1}$$

and two functions

A link function that describes how the mean , $E(Y_i) = \mu_i$, depends on the linear predictor

$$g(\mu_i) = \eta_i$$

a variance function that describes how the variance, $\text{var}(Y_i)$ depends on the mean

$$\text{var}(Y_i) = \phi V(\mu)$$

where the dispersion parameter ϕ is a constant

3.2 Exponential Family

Exponential family distributions' densities can be written in the form

$$f(\boldsymbol{y}; \boldsymbol{\theta}, \phi) = \exp\left\{ \frac{\boldsymbol{y'\theta} - b(\boldsymbol{\theta})}{a(\phi)} + c(\phi, \boldsymbol{y}) \right\} \tag{2}$$

where ϕ is the dispersion parameter and θ is the canonical parameter. It can be shown that

$$\begin{aligned} \text{E(Y)} &= b'(\theta) = \mu \\ \text{and} \quad \text{var}(Y) &= \phi b''(\theta) = \phi V(\mu) \end{aligned}$$

3.3 Canonical Links

For a glm where the response follows an exponential distribution we have

$$g\left(\mu_i\right) = g\left(b'\left(\theta_i\right)\right) = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi}$$

The canonical link is defined as

$$\begin{aligned} \mathrm{g} &= \left(b'\right)^{-1} \\ \Rightarrow g\left(\mu_i\right) &= \theta_i = \beta_0 + \beta_1 x_{1i} + \ldots + \beta_p x_{pi} \end{aligned}$$

Canonical links lead to desirable statistical properties of the glm hence tend to be used by default

Given the predictors $\boldsymbol{x} \in \mathbb{R}^p$, if the response y follows the generalized linear models (GLMs),we set $\theta = x^\top w$,$\phi = 1$ and $b(\theta) = \psi\left(x^\top \omega\right)$,$\exp(c(\phi, y)) = \rho(y)$ then according to (2) its conditional distribution takes the form

$$y \mid \boldsymbol{x} \sim \mathbb{P}(y \mid \boldsymbol{x}) = \rho(y) \exp\left\{y \boldsymbol{x}^T \boldsymbol{w} - \psi\left(\boldsymbol{x}^T \boldsymbol{w}\right)\right\} \tag{3}$$

where $\boldsymbol{w} \in \mathbb{R}^p$ is the coefficient, ρ and ψ are some known univariate functions. $\psi'\left(\boldsymbol{x}^T \boldsymbol{w}\right) = \mathbb{E}(y \mid \boldsymbol{x})$ is called the inverse link function (McCullagh and Nelder, 1989). Another important property is that $\mathrm{Var}(y \mid \boldsymbol{x}) = \psi''\left(\boldsymbol{x}^T \boldsymbol{w}\right)$, which follows from the fact that the distribution belongs to the exponential family that I mentioned before. It is ψ that characterizes different GLMs. For example, in linear model with Gaussian noise, we have a continuous response y and $\psi(u) = \frac{1}{2}u^2$; in the logistic regression model, y is binary and $\psi(u) = \log\left(1 + e^u\right)$; and in Poisson regression model, y is a nonnegative integer and $\psi(u) = e^u$. For most GLMs, ψ is strictly convex and infinitely differentiable.

4 Methodology

4.1 Target data, source data, and transferring level

In this paper, we consider the following multi-source transfer learning problem.The goal is to transfer useful information from source data to obtain a better model for the target data.We assume the responses in the target and source data all follow the generalized linear model:

$$y^{(k)} \mid \boldsymbol{x} \sim \mathbb{P}(y \mid \boldsymbol{x}) = \rho(y) \exp\left\{ y \boldsymbol{x}^T \boldsymbol{w}^{(k)} - \psi\left(\boldsymbol{x}^T \boldsymbol{w}^{(k)} \right) \right\} \tag{4}$$

for k=0, ..., K , with possibly different coefficient $\boldsymbol{w}^{(k)} \in \mathbb{R}^p$, the predictor $\boldsymbol{x} \in \mathbb{R}^p$, and some known univariate functions ρ and ψ. Denote the target parameter as $\boldsymbol{\beta} = \boldsymbol{w}^{(0)}$. Suppose the target model is ℓ_0-sparse, which satisfies $\|\boldsymbol{\beta}\|_0 = s \ll p$. This means that only s of the p variables contribute to the target response. Intuitively, if $\boldsymbol{w}^{(k)}$ is close to $\boldsymbol{\beta}$, the k -th source could be useful for transfer learning.

Define the k -th contrast $\boldsymbol{\delta}^{(k)} = \boldsymbol{\beta} - \boldsymbol{w}^{(k)}$ and we say $\left\| \boldsymbol{\delta}^{(k)} \right\|_1$ is the transferring level of source k . And we define the level-h transferring set $\mathcal{A}_h = \left\{ k : \left\| \boldsymbol{\delta}^{(k)} \right\|_1 \leq h \right\}$ as the set of sources which has transferring level lower than h . Note that in general, h can be any positive values and different h values define different $\mathcal{A}_h$ set. However, in our regime of interest, h shall be reasonably small to guarantee that transferring sources in$\mathcal{A}_h$ is beneficial. Denote $n_{\mathcal{A}_h} = \sum_{k \in \mathcal{A}_h} n_k$, $\alpha_k = \frac{n_k}{n_{\mathcal{A}_h} + n_0}$ for k $\in \{0\} \cup \mathcal{A}_h$ and $K_{\mathcal{A}_h} = |\mathcal{A}_h|$.

Note that in (1), we assume GLMs of the target and all sources share the same inverse link function ψ. For simplicity, in the following discussion, we assume all these GLMs belong to the same family and hence have the same function ψ .

4.2 Two-step GLM transfer learning

The main strategy is to first transfer the information from those sources by pooling all the data to obtain a rough estimator, then correct the bias in

the second step using the target data. More specifically, we fit a GLM with $\ell_1 - penalty$ by pooled samples first, then fit the contrast in the second step using only the target by another ℓ_1-regularization. The detailed algorithm (A-Trans-GLM) is presented in Algorithm 1.

A single algorithm can be used to estimate the parameters of an exponential family glm using maximum likelihood. The log-likelihood for the sample $y_1^1, ..., y_{n_1}^1, y_1^2, ..., y_{n_2}^2, ..., y_1^k, ..., y_{n_k}^k$ is

$$L(w) = \prod_{k\in\{0\}\cup A} \prod_{i=1}^{n_k} \exp\left\{ y_i^{k)} x_i^{(k)^\top} w - \psi\left(x_i^{(k)^\top} w \right) \right\} \tag{5}$$

where the target data sets as $\left(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)}\right)$ and K source data sets with the k -th source denoted as $\left(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)}\right)$, where $\boldsymbol{X}^{(k)} \in \mathbb{R}^{n_k \times p}, \boldsymbol{y}^{(k)} \in \mathbb{R}^{n_k}$ for k=0, . . ., K . The i -th row of $\boldsymbol{X}^{(k)}$ and the i -th element of $\boldsymbol{y}^{(k)}$ are denoted as $\boldsymbol{x}_i^{(k)}$ and $y_i^{(k)}$, respectively.

In order to avoid underflow caused by multiple operation, logarithmic likelihood is used and simplified

$$l(w) = \log L(w) = \sum_{k\in\{0]\cup A} \left\{ \sum_{i=1}^{n_k} y_i^{(k)} x_i^{(k)^\top} w - \sum_{i=1}^{n_k} \psi\left(x_i^{(k)^\top} w \right) \right\} \tag{6}$$

Then the maximum likelihood estimates are obtained by solving the

$$\sum_{k\in\{0\}\cup\mathcal{A}} \left[\left(\boldsymbol{X}^{(k)}\right)^T \boldsymbol{y}^{(k)} - \sum_{i=1}^{n_k} \psi'\left((\boldsymbol{w})^T \boldsymbol{x}_i^{(k)} \right) \boldsymbol{x}_i^{(k)} \right] = \mathbf{0}_p \tag{7}$$

which converges to the solution of its population version under certain conditions

$$\sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k \mathbb{E}\left\{ \left[\psi'\left(\left(\boldsymbol{w}^{\mathcal{A}}\right)^T \boldsymbol{x}^{(k)} \right) - \psi'\left(\left(\boldsymbol{w}^{(k)}\right)^T \boldsymbol{x}^{(k)} \right) \right] \boldsymbol{x}^{(k)} \right\} = \mathbf{0}_p \tag{8}$$

where $\alpha_k = \frac{n_k}{n_{\mathcal{A}}+n_0}$. Notice that in the linear case, $\boldsymbol{w}^{\mathcal{A}}$ can be explicitly expressed as a linear transformation of the true parameter $\boldsymbol{w}^{(k)}$.

According to equation (8), we have

$$\sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k \mathbb{E}\left[\psi'\left(\left(\boldsymbol{w}^{\mathcal{A}}\right)^T \boldsymbol{x}^{(k)}\right)\boldsymbol{x}^{(k)}\right] = \sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k \mathbb{E}\left[\psi'\left(\left(\boldsymbol{w}^{(k)}\right)^T \boldsymbol{x}^{(k)}\right)\boldsymbol{x}^{(k)}\right] \tag{9}$$

Notice that $\boldsymbol{x}^{(k)}$ are random variables, and $\boldsymbol{w}^{(k)}$are fixed, and equation (10)'s both sides are nested by function ψ' , so we have

$$\sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k E\left[x^{(k)} x^{(k)\top}\right] w^{A} = \sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k E\left[x^{(k)} x^{(k)\top}\right] w^{(k)} \tag{10}$$

then $\boldsymbol{w}^{\mathcal{A}} = \boldsymbol{\Sigma}^{-1}\sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k \boldsymbol{\Sigma}^{(k)}\boldsymbol{w}^{(k)}$, where $\boldsymbol{\Sigma}^{(k)} = \mathbb{E}\left[\boldsymbol{x}^{(k)}\left(\boldsymbol{x}^{(k)}\right)^T\right]$ and $\boldsymbol{\Sigma} = \sum_{k\in\{0\}\cup\mathcal{A}} \alpha_k \mathbb{E}\left[\boldsymbol{x}^{(k)}\left(\boldsymbol{x}^{(k)}\right)^T\right]$ (Li et al., 2021).

5 Trans-GLM

If we have $\mathcal{A}$ first,then we can use Algorithm 1

Algorithm 1: $\mathcal{A}$-Trans-GLM

Input: target data $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)})$, source data $\{(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})\}_{k=1}^{K}$, penalty parameters $\lambda_{\boldsymbol{w}}$ and $\lambda_{\boldsymbol{\delta}}$, transferring set $\mathcal{A}$

Output: the estimated coefficient vector $\hat{\boldsymbol{\beta}}$

1 **Transferring step**: Compute

$$\hat{\boldsymbol{w}}^{\mathcal{A}} \leftarrow \arg\min_{\boldsymbol{w}} \left\{ \frac{1}{n_{\mathcal{A}}+n_0} \sum_{k\in\{0\}\cup\mathcal{A}} \left[-(\boldsymbol{y}^{(k)})^T \boldsymbol{X}^{(k)}\boldsymbol{w} + \sum_{i=1}^{n_k} \psi(\boldsymbol{w}^T \boldsymbol{x}_i^{(k)}) \right] + \lambda_{\boldsymbol{w}} \|\boldsymbol{w}\|_1 \right\}$$

2 **Debiasing step**: Compute

$$\hat{\boldsymbol{\delta}}^{\mathcal{A}} \leftarrow \arg\min_{\boldsymbol{\delta}} \left\{ -\frac{1}{n_0}(\boldsymbol{y}^{(0)})^T \boldsymbol{X}^{(0)}(\hat{\boldsymbol{w}}^{\mathcal{A}} + \boldsymbol{\delta}) + \frac{1}{n_0}\sum_{i=1}^{n_0} \psi((\hat{\boldsymbol{w}}^{\mathcal{A}} + \boldsymbol{\delta})^T \boldsymbol{x}_i^{(0)}) + \lambda_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_1 \right\}$$

3 Let $\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{w}}^{\mathcal{A}} + \hat{\boldsymbol{\delta}}^{\mathcal{A}}$

4 Output $\hat{\boldsymbol{\beta}}$

Fig. 1. algorithm of Oracle Trans-Lasso

However, transferring certain sources may not improve the performance of the fitted model based on only target, and can even lead to worse per-

formance. In transfer learning, we say negative transfer happens when the source data leads to an inferior performance on the target task.

we propose a simple, algorithm-free, and data-driven method to determine an informative transferring set $\widehat{\mathcal{A}}$. We call this approach a transferable source detection algorithm and refer to it as Trans-GLM.

6 Transferable source detection

The detection algorithm.

(1)Divide the target data into three folds, that is, $\left\{\left(\boldsymbol{X}^{(0)[r]}, \boldsymbol{y}^{(0)[r]}\right)\right\}_{r=1}^{3}$. Note that we choose three folds only for convenience. (2)Run the transferring step on each source data and every two folds of target data. For a given loss function, we calculate its value on the leftout fold of target data and compute the average cross-validation loss $\hat{L}_0^{(k)}$ for each source. As a benchmark, we also fit Lasso on every choice of two folds of target data and calculate the loss on the remaining fold. The average cross-validation loss $\hat{L}_0^{(0)}$ is viewed as the loss of target. Finally, the difference between $\hat{L}_0^{(k)}$ and $\hat{L}_0^{(0)}$ is calculated and compared with some threshold, and sources with a difference less than the threshold will be recruited into $\widehat{\mathcal{A}}$. Under the GLM setting, a natural loss function is the negative log-likelihood. For convenience, suppose n_0 is divisible by 3 .

We denote that log-likelihood for the sample $y_1^{(0)[r]}, ..., y_{n_0/3}^{(0)[r]}$ as

$$l_0^{[r]}(w) = \prod_{i=1}^{n_0/3} \rho\left(y_i^{(0)[r]}\right) \exp\left[y_i^{(0)} x_i^{(0)\top} w - \psi\left(w^\top x_i^{(0)[r]}\right)\right] \tag{11}$$

then according to equation (6) and (7), we have

$$\sum_{i=1}^{n_0/3} y_i^{(0)} x_i^{(0)\top} w = \left(\boldsymbol{y}^{(0)[r]}\right)^T \boldsymbol{X}^{(0)} \boldsymbol{w} \tag{12}$$

and under the GLM setting, a natural loss function is the negative log-likelihood. So

$$\hat{L}_0^{[r]}(w) = -\frac{1}{n_0/3} \log l_0^{ir]}(w) \tag{13}$$

According to (1), for any coefficient estimate $\boldsymbol{w}$, the average of negative log-likelihood on the r -th fold of target data $\left(\boldsymbol{X}^{(0)[r]}, \boldsymbol{y}^{(0)[r]}\right)$ is

$$\hat{L}_0^{[r]}(\boldsymbol{w}) = -\frac{1}{n_0/3}\sum_{i=1}^{n_0/3}\log\rho\left(y_i^{(0)[r]}\right) - \frac{1}{n_0/3}\left(\boldsymbol{y}^{(0)[r]}\right)^T \boldsymbol{X}^{(0)}\boldsymbol{w} + \frac{1}{n_0/3}\sum_{i=1}^{n_0/3}\psi\left(\boldsymbol{w}^T\boldsymbol{x}_i^{(0)[r]}\right) \tag{14}$$

The detailed algorithm is presented as Algorithm 2.

Algorithm 2: Trans-GLM

Input: target data $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)})$, all source data $\{(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})\}_{k=1}^K$, a constant $C_0 > 0$, penalty parameters $\{\{\lambda^{(k)[r]}\}_{k=0}^K\}_{r=1}^3$

Output: the estimated coefficient vector $\hat{\boldsymbol{\beta}}$, and the determined transferring set $\widehat{\mathcal{A}}$

Transferable source detection: Randomly divide $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)})$ into three sets of equal size as $\{(\boldsymbol{X}^{(0)[i]}, \boldsymbol{y}^{(0)[i]})\}_{i=1}^3$

for $r = 1$ **to** 3 **do**

  $\hat{\boldsymbol{\beta}}^{(0)[r]} \leftarrow$ fit the Lasso on $\{(\boldsymbol{X}^{(0)[i]}, \boldsymbol{y}^{(0)[i]})\}_{i=1}^3 \backslash (\boldsymbol{X}^{(0)[r]}, \boldsymbol{y}^{(0)[r]})$ with penalty parameter $\lambda^{(0)[r]}$

  $\hat{\boldsymbol{\beta}}^{(k)[r]} \leftarrow$ run step 1 in Algorithm 1 with $(\{(\boldsymbol{X}^{(0)[i]}, \boldsymbol{y}^{(0)[i]})\}_{i=1}^3 \backslash (\boldsymbol{X}^{(0)[r]}, \boldsymbol{y}^{(0)[r]})) \cup (\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})$ and penalty parameter $\lambda^{(k)[r]}$ for all $k \neq 0$

  Calculate the loss function $\hat{L}_0^{[r]}(\hat{\boldsymbol{\beta}}^{(k)[r]})$ on $(\boldsymbol{X}^{(0)[r]}, \boldsymbol{y}^{(0)[r]})$ for $k = 1, \ldots, K$

end

$\hat{L}_0^{(k)} \leftarrow \sum_{r=1}^3 \hat{L}_0^{[r]}(\hat{\boldsymbol{\beta}}^{(k)[r]})/3$, $\hat{L}_0^{(0)} \leftarrow \sum_{r=1}^3 \hat{L}_0^{[r]}(\hat{\boldsymbol{\beta}}^{(0)[r]})/3$, $\hat{\sigma} = \sqrt{\sum_{r=1}^3 (\hat{L}_0^{[r]}(\hat{\boldsymbol{\beta}}^{(0)[r]}) - \hat{L}_0^{(0)})^2/2}$

$\widehat{\mathcal{A}} \leftarrow \{k \neq 0 : \hat{L}_0^{(k)} - \hat{L}_0^{(0)} \leq C_0(\hat{\sigma} \vee 0.01)\}$

$\widehat{\mathcal{A}}$-Trans-GLM: $\hat{\boldsymbol{\beta}} \leftarrow$ run Algorithm 1 using $\{(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})\}_{k \in \{0\} \cup \widehat{\mathcal{A}}}$

Output $\hat{\boldsymbol{\beta}}$ and $\widehat{\mathcal{A}}$

Fig. 2. algorithm of Trans-Lasso

It' s important to point out that Algorithm 2 does not require the input of h. We will show that $\widehat{\mathcal{A}} = \mathcal{A}_h$ for some specific h if certain conditions hold.This is the reason that this algorithm is called the transferable source detection algorithm.

7 Confidence intervals

In this section, we would like to construct the asymptotic confidence interval (CI) for each component of β based on that point estimate.

In the following, we will propose a transfer learning procedure to construct CI based on the desparsified Lasso (Van de Geer et al., 2014). Recall that desparsified Lasso contains two main steps. (1) Learn the inverse Fisher information matrix of GLM by nodewise regression

(2)The second step is to "debias" the initial point estimator and then construct the asymptotic CI.Intuitively, if the predictors from target and source data are similar and satisfy some sparsity conditions, it might be possible to use Algorithm 1 for learning the inverse Fisher information matrix of target data, which effectively combines the information from target and source data.

Before formalizing the procedure to construct the CI, let's first define several additional notations. For any $\boldsymbol{w} \in \mathbb{R}^n$, denote

$$\boldsymbol{W}_{\boldsymbol{w}}^{(k)} = \operatorname{diag}\left(\sqrt{\psi''\left(\left(\boldsymbol{x}_1^{(k)}\right)^T \boldsymbol{w}\right)}, \ldots, \sqrt{\psi''\left(\left(\boldsymbol{x}_{n_k}^{(k)}\right)^T \boldsymbol{w}\right)}\right) \tag{15}$$

$\boldsymbol{X}_{\boldsymbol{w}}^{(k)} = \boldsymbol{W}_{\boldsymbol{w}}^{(k)} \boldsymbol{X}^{(k)}$, $\boldsymbol{\Sigma}_{\boldsymbol{w}}^{(k)} = \mathbb{E}\left[\boldsymbol{x}^{(k)} \left(\boldsymbol{x}^{(k)}\right)^T \psi''\left(\left(\boldsymbol{x}^{(k)}\right)^T \boldsymbol{w}\right)\right]$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{w}}^{(k)} = n_k^{-1}\left(\boldsymbol{X}_{\boldsymbol{w}}^{(k)}\right)^T \boldsymbol{X}_{\boldsymbol{w}}^{(k)}$.$\boldsymbol{X}_{\boldsymbol{w},j}^{(k)}$represents the j -th column of $\boldsymbol{X}_{\boldsymbol{w}}^{(k)}$ and $\boldsymbol{X}_{\boldsymbol{w},-j}^{(k)}$ represents the matrix $\boldsymbol{X}_{\boldsymbol{w}}^{(k)}$ without the j -th column. $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{w},j,-j}^{(k)}$ represents the j -th row of $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{w}}^{(k)}$ without the diagonal (j, j) element, and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{w},j,j}^{(k)}$ is the diagonal (j, j) element of $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{w}}^{(k)}$.

Algorithm 3: Confidence interval construction via nodewise regression

Input: target data $(\boldsymbol{X}^{(0)}, \boldsymbol{y}^{(0)})$, source data $\{(\boldsymbol{X}^{(k)}, \boldsymbol{y}^{(k)})\}_{k=1}^{K}$, penalty parameters $\{\lambda_j\}_{j=1}^{p}$ and $\{\tilde{\lambda}_j\}_{j=1}^{p}$, transferring set $\mathcal{A}$, confidence level $(1-\alpha)$

Output: Level-$(1-\alpha)$ confidence interval $\mathcal{I}_j$ for β_j with $j = 1, \ldots, p$

1 Compute $\hat{\boldsymbol{\beta}}$ via Algorithm 1

2 Compute $\hat{\boldsymbol{\gamma}}_j^{\mathcal{A}} \leftarrow \arg\min_{\boldsymbol{\gamma}} \left\{ -\frac{1}{2(n_{\mathcal{A}}+n_0)} \sum_{k\in\{0\}\cup\mathcal{A}} \|\boldsymbol{X}^{(k)}_{\hat{\boldsymbol{\beta}},j} - \boldsymbol{X}^{(k)}_{\hat{\boldsymbol{\beta}},-j}\boldsymbol{\gamma}\|_2^2 + \lambda_j\|\boldsymbol{\gamma}\|_1 \right\}$ for $j = 1, \ldots, p$

3 Compute $\hat{\boldsymbol{\varrho}}_j \leftarrow \arg\min_{\boldsymbol{\varrho}} \left\{ -\frac{1}{2n_0}\|\boldsymbol{X}^{(0)}_{\hat{\boldsymbol{\beta}},j} - \boldsymbol{X}^{(0)}_{\hat{\boldsymbol{\beta}},-j}(\hat{\boldsymbol{\gamma}}_j^{\mathcal{A}} + \boldsymbol{\varrho})\|_2 + \tilde{\lambda}_j\|\boldsymbol{\varrho}\|_1 \right\}$

4 Compute $\hat{\boldsymbol{\gamma}}_j^{(0)} \leftarrow \hat{\boldsymbol{\gamma}}_j^{\mathcal{A}} + \hat{\boldsymbol{\varrho}}_j$, $\widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}} \leftarrow \sum_{k\in\{0\}\cup\mathcal{A}} \frac{n_k}{n_{\mathcal{A}}+n_0} \widehat{\boldsymbol{\Sigma}}^{(k)}_{\hat{\boldsymbol{\beta}}}$, $\hat{\tau}_j^2 = \widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}},j,j} - \widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}},j,-j}\hat{\boldsymbol{\gamma}}_j$ and calculate $\widehat{\boldsymbol{\Theta}}$ via (4), where $\hat{\boldsymbol{\gamma}}_j^{(0)} = (\hat{\gamma}_{j,1}^{(0)}, \ldots, \hat{\gamma}_{j,j-1}^{(0)}, \hat{\gamma}_{j,j+1}^{(0)}, \ldots, \hat{\gamma}_{j,p}^{(0)})^T$.

5 Compute $\mathcal{I}_j \leftarrow [\hat{b}_j - \widehat{\boldsymbol{\Theta}}_j^T\widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}}\widehat{\boldsymbol{\Theta}}_j q_{\alpha/2}/\sqrt{n_0}, \hat{b}_j + \widehat{\boldsymbol{\Theta}}_j^T\widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}}\widehat{\boldsymbol{\Theta}}_j q_{\alpha/2}/\sqrt{n_0}]$ for $j = 1, \ldots, p$, where $\hat{b}_j$ is the j-th component of $\hat{\boldsymbol{b}}$ in (5), and $q_{\alpha/2}$ is the $\alpha/2$-left tail quantile of $\mathcal{N}(0,1)$

6 Output $\{\mathcal{I}_j\}_{j=1}^{p}$

Fig. 3. Confidence interval construction via nodewise regression

Next, we explain the details of the CI construction procedure in Algorithm 3 . In step 1, we obtain a point estimator $\hat{\boldsymbol{\beta}}$ from $\mathcal{A}$ -Trans-GLM (Algorithm 1), given a specific transferring set $\mathcal{A}$. Then in steps 2-4 , we estimate the target inverse Fisher information matrix $\left(\Sigma_\beta^{(0)}\right)^{-1}$ as

$$\operatorname{diag}\left(\hat{\tau}_1^{-2}, \ldots, \hat{\tau}_p^{-2}\right) \begin{pmatrix} 1 & -\hat{\gamma}_{1,2}^{(0)} & \ldots & -\hat{\gamma}_{1,p}^{(0)} \\ -\hat{\gamma}_{2,1}^{(0)} & 1 & \ldots & -\hat{\gamma}_{2,p}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ -\hat{\gamma}_{p,1}^{(0)} & -\hat{\gamma}_{p,2}^{(0)} & \ldots & 1 \end{pmatrix} \tag{16}$$

Finally in step 5, we "debias" $\hat{\boldsymbol{\beta}}$ using the target data to get a new point estimator $\hat{\boldsymbol{b}}$ which is asymptotically unbiased as

$$\hat{\boldsymbol{b}} = \hat{\boldsymbol{\beta}} + \frac{1}{n_0}\widehat{\boldsymbol{\Theta}}\left(\boldsymbol{X}^{(0)}\right)^T\left[\boldsymbol{Y}^{(0)} - \boldsymbol{\psi}'\left(\boldsymbol{X}^{(0)}\hat{\boldsymbol{\beta}}\right)\right] \tag{17}$$

where $\boldsymbol{\psi}'\left(\boldsymbol{X}^{(0)}\hat{\boldsymbol{\beta}}\right) := \left(\psi'\left(\left(\boldsymbol{x}_1^{(0)}\right)^T\hat{\boldsymbol{\beta}}\right), \ldots, \psi'\left(\left(\boldsymbol{x}_{n_0}^{(0)}\right)^T\hat{\boldsymbol{\beta}}\right)\right)^T \in \mathbb{R}^{n_0}$

I learnt from "Confidence Intervals and Hypothesis Testing for High-Dimensional Regression" that $\hat{\boldsymbol{b}}$ is approximately Gaussian, with mean $\hat{\boldsymbol{\beta}}$ and covariance $\widehat{\boldsymbol{\Theta}}^T \widehat{\boldsymbol{\Sigma}}_{\hat{\boldsymbol{\beta}}} \widehat{\boldsymbol{\Theta}}/n_0$ It' s necessary to emphasize that the confidence level $(1- \alpha)$ is for every single CI rather than for all p CIs simultaneously. As discussed in Sections 2.2 and 2.3 of Van de Geer et al. (2014), it is possible to get simultaneous CIs for different coefficient components and do multiple hypothesis tests when the design is fixed. In other cases, e.g., random design in different replications (which we focus on in this paper), multiple hypothesis testing might be more challenging.

8 Numerical Experiment

In the simulation part, we study the performance of different methods under various settings of h . The methods include TransGLM (Algorithm 2), näve-Lasso (Lasso on target data), $\mathcal{A}_h$ -Trans-GLM (Algorithm 1 with $\mathcal{A} = \mathcal{A}_h$) and Pooled-Trans-GLM (Algorithm 1 with all sources).

8.1 Transfer learning on $\mathcal{A}_h$

In this section, we study the performance of $\mathcal{A}_h$ -Trans-GLM and compare it with that of naïve-Lasso. The purpose of the simulation is to verify that $\mathcal{A}_h$ -Trans-GLM can outperform naïve-Lasso in terms of the target coefficient estimation error, when h is not too large.

Consider the simulation setting as follows. We set the target sample size $n_0 = 200$ and source sample sample size $n_k = 100$ for each $k \neq 0$. The dimension p=500 for both target and source data. For the target, the coefficient is set to be $\boldsymbol{\beta} = (0.5 \cdot \mathbf{1}_s, \mathbf{0}_{p-s})^T$, where $\mathbf{1}_s$ has all s elements 1 and $\mathbf{0}_{p-s}$ has all (p-s) elements 0 , where s is set to be 5 . Denote$\mathcal{R}_p^{(k)}$ as p independent Rademacher variables (being -1 or 1 with equal probability) for any k .$\mathcal{R}_p^{(k)}$ is independent with $\mathcal{R}_p^{(k')}$ for any k$\neq k'$. For any source data k in $\mathcal{A}_h$, we set $\boldsymbol{w}^{(k)} = \boldsymbol{\beta} + (h/p)\mathcal{R}_p^{(k)}$. For linear and logistic regression

models, predictors from target $\boldsymbol{x}_i^{(0)} \overset{i.i.d}{\sim} \mathcal{N}(\mathbf{0}_p, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} = \left[\Sigma_{jj'}\right]_{p\times p}$ where $\Sigma_{jj'} = 0.5^{|j-j'|}$, for all $i = 1, \cdots, n$. And for $k \in \mathcal{A}_h$, we generate p - dimensional predictors from $\mathcal{N}\left(\mathbf{0}_p, \boldsymbol{\Sigma} + \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right), where \boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}_p, 0.3^2\boldsymbol{I}_p\right)$ and is independently generated. For Poisson regression model, predictors are from the same Gaussian distributions as those in linear and binomial cases with coordinate-wise truncation at $\pm$ 0.5 .

Note that naïve-Lasso is fitted on only target data, and $\mathcal{A}_h$ -Trans-GLM denotes Algorithm 1 on source data in $\mathcal{A}_h$ as well as target data. We train naïve-Lasso and $\mathcal{A}_h$ -TransGLM models under different settings of h and $K_{\mathcal{A}_h}$, then calculate the ℓ_2 -estimation error of $\boldsymbol{\beta}$. All the experiments are replicated 200 times and the average ℓ_2 -estimation errors of $\mathcal{A}_h$-Trans-GLM and näve-Lasso under linear, logistic, and Poisson regression models are shown in Fig.4. Fig.5. Fig.6.

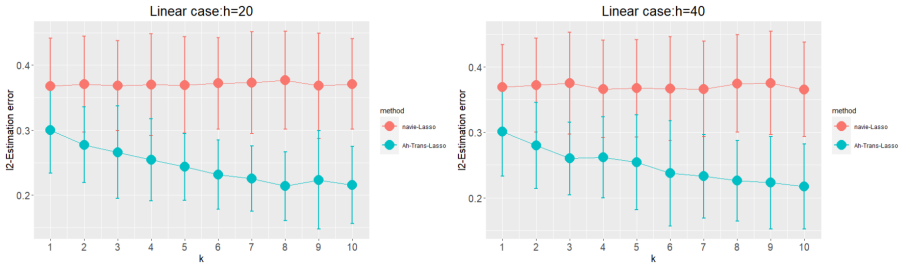

Fig. 4. Linear Model

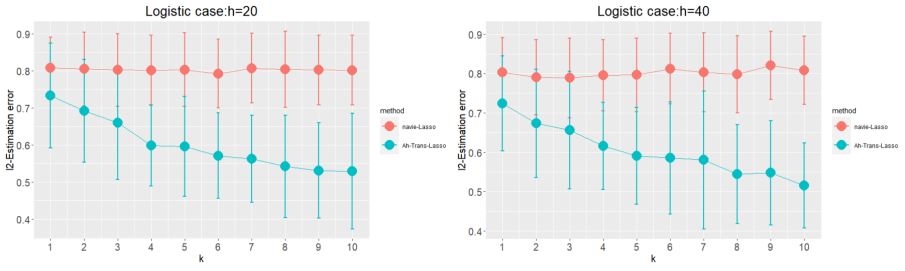

Fig. 5. Logistic Model

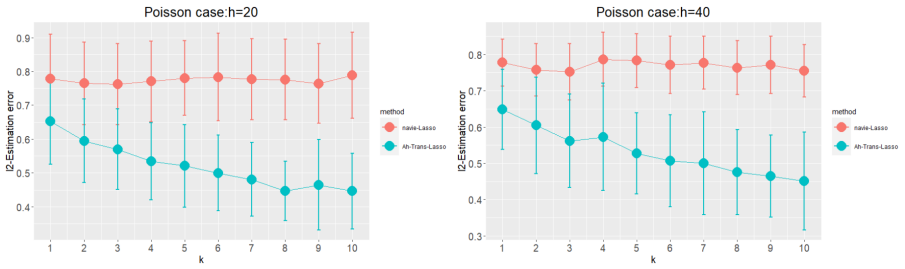

Fig. 6. Poisson Model

We can see that $\mathcal{A}_h$-Trans-GLM outperforms naive-Lasso for all combinations of h and K in the three figure. As more and more source data become available, the performance of $\mathcal{A}_h$-Trans-GLM improves significantly.But the improvement is slowing when the k increases.The reason maybe the negative transfer. When h increases, the performance of Ah-Trans-GLM becomes worse.

8.2 Transfer learning when $\mathcal{A}_h$ is unknown

Different from the previous subsection, now we fix the total number of sources as K=10 .

There are two types of sources, which belong to either $\mathcal{A}_h$ or $\mathcal{A}_h^c$. Sources from $\mathcal{A}_h$ have similar coefficients to the target one, while the coefficients of sources from $\mathcal{A}_h$ can be quite different. Intuitively, using more sources from $\mathcal{A}_h$ benefits the estimation of the target coefficient. But in practice, $\mathcal{A}_h$ may not be known as a priori. As we argued before, Trans-GLM can detect useful sources automatically, therefore it is expected to be helpful in such a scenario. Simulations in this section aim to justify the effectiveness of Trans-GLM.

Here is the detailed setting. We set the target sample size $n_0 = 200$ and source sample sample size $n_k = 200$ for all $k \neq 0$. The dimension p=2000 . Target coefficient is the same as the one used in Section 4.1.1 and we fix the signal number s=20 . Recall $\mathcal{R}_p^{(k)}$ denotes p independent

Rademacher variables and $\mathcal{R}_p^{(k')}$ are independent for any $k \neq k'$. Consider h=20 and 40. For any source data k in $\mathcal{A}_h$, we set $\boldsymbol{w}^{(k)} = \boldsymbol{\beta} + (h/p)\mathcal{R}_p^{(k)}$. For linear and logistic regression models, predictors from target $\boldsymbol{x}_i^{(0)} \overset{i.i.d.}{\sim} N(\mathbf{0}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} = \left[\Sigma_{jj'}\right]_{p\times p}$ where $\Sigma_{jj'} = 0.9^{|j-j'|}$, for all $i = 1, \cdots, n_0$. For the source, we generate p -dimensional predictors from independent t -distribution with degrees of freedom 4. For the target and sources of Poisson regression model, we generate predictors from the same Gaussian distribution and t -distribution respectively, and truncate each predictor at $\pm$ 0.5 .

To generate the coefficient $\boldsymbol{w}^{(k)}$ for $k \notin \mathcal{A}_h$, we randomly generate $S^{(k)}$ of size s from $\{2s+1, \ldots, p\}$. Then, the j -th component of coefficient $\boldsymbol{w}^{(k)}$ is set to be

$$w_j^{(k)} = \begin{cases} 0.5 + 2hr_j^{(k)}/p, & j \in \{s+1, \ldots, 2s\} \cup S^{(k)} \\ 2hr_j^{(k)}/p, & \text{otherwise} \end{cases} \tag{18}$$

where $r_j^{(k)}$ is a Rademacher variable. We also add an intercept 0.5 . The generating process of each source data is independent. Compared to the setting in Section 4.1.1, the current setting is more challenging because source predictors come from t -distribution with heavier tails than sub-Gaussian tails. However, although Assumption 2 is violated, in the following analysis, we will see that Trans-GLM can still succeed in detecting informative sources.

As before, we fit naïve-Lasso on only target data. $\mathcal{A}_h$ -Trans-GLM and Pooled-TransGLM represent Algorithm 1 on source data in $\mathcal{A}_h$ and target data or all sources and target data, respectively. Trans-GLM runs Algorithm 2 by first identifying the informative source set $\widehat{\mathcal{A}}$, then applying Algorithm 1 to fit the model on sources in $\widehat{\mathcal{A}}$. We vary the values of $K_{\mathcal{A}_h}$and h , and repeat simulations in each setting 200 times.

Because there is too much simulated data, I only simulated one situation that is Linear case when h equal to 20.

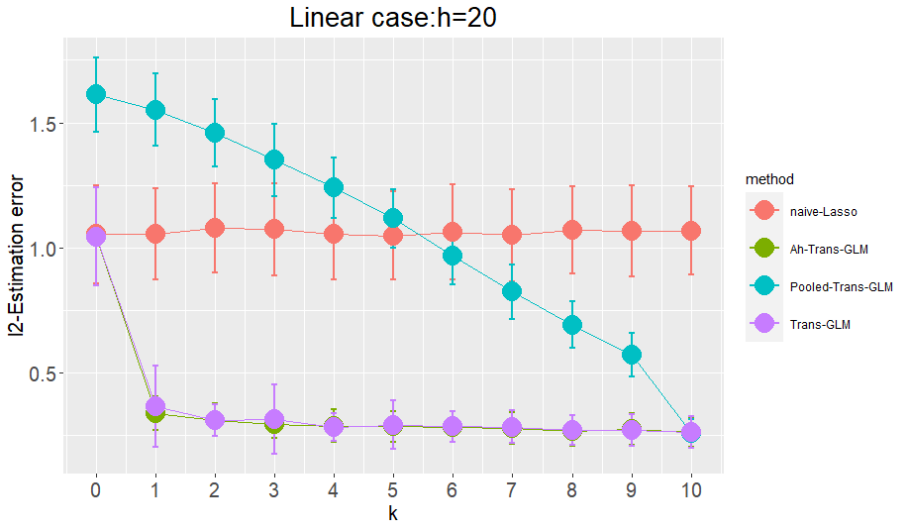

Fig. 7. Linear Model

We can be observe that in this Fig, $\mathcal{A}_h$-Trans-GLM's performance is the best. Trans-GLM mimics the behavior of $\mathcal{A}_h$ -Trans-GLM very well, implying that the transferable source detection algorithm can successfully recover $\mathcal{A}_h$. When $K_{\mathcal{A}_h}$ is small, PooledTrans-GLM performs worse than naïve-Lasso because of the negative transfer. As $K_{\mathcal{A}_h}$increases, the performance of Pooled-Trans-GLM improves and finally matches those of Ah-Trans-GLM and Trans-GLM when KAh = K = 10.

8.3 R Code

```r
library(glmtrans)
library(glmnet)
library(ggplot2)
# A is known

erg1 <- matrix(NA, ncol = 10, nrow = 200)
erg2 <- matrix(NA, ncol = 10, nrow = 200)
erg3 <- matrix(NA, ncol = 10, nrow = 200)
erg4 <- matrix(NA, ncol = 10, nrow = 200)
```

```
    erb3 <-  matrix(NA, ncol = 10, nrow = 200)
    erb4 <-  matrix(NA, ncol = 10, nrow = 200)
    erp1 <-  matrix(NA, ncol = 10, nrow = 200)
    erp2 <-  matrix(NA, ncol = 10, nrow = 200)
    erp3 <-  matrix(NA, ncol = 10, nrow = 200)
    erp4 <-  matrix(NA, ncol = 10, nrow = 200)
    for(i in 1:10)
    {
      for(j in 1:200)
      {
        D.training <- models(family = "gaussian", type = "all",
cov.type = 1, Ka = i,
        K = 20, s = 5, n.target = 200, n.source = rep(100, 20),p
=500)


        fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
training$target$y,
        family = "gaussian")
        fit.oracle <- glmtrans(target = D.training$target,
source = D.training$source,
        family = "gaussian",transfer.source.id=1:i, cores = 2)


        beta <- c(0, rep(0.5, 5), rep(0, 500 - 5))
        erg1[j,i] <- sqrt(sum((coef(fit.lasso) - beta)^2))
        erg2[j,i] <- sqrt(sum((fit.oracle$beta - beta)^2))
      }
    }
    for(i in 1:10)
    {
      for(j in 1:200)
      {
        D.training <- models(family = "poisson", type = "all",
cov.type = 1, Ka = i,
        K = 20, s = 5, n.target = 200, n.source = rep(100, 20),p
=500)

```

```
42
43          fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
    training$target$y,
44          family = "poisson")
45          fit.oracle <- glmtrans(target = D.training$target,
    source = D.training$source,
46          family = "poisson",transfer.source.id=1:i, cores = 2)
47
48
49          beta <- c(0, rep(0.5, 5), rep(0, 500 - 5))
50          erp1[j,i] <- sqrt(sum((coef(fit.lasso) - beta)^2))
51          erp2[j,i] <- sqrt(sum((fit.oracle$beta - beta)^2))
52        }
53      }
54
55      for(i in 1:10)
56      {
57        for(j in 1:200)
58        {
59          D.training <- models(family = "poisson", type = "all",
    cov.type = 1, Ka = i,
60          K = 40, s = 5, n.target = 200, n.source = rep(100, 40),p
    =500)
61
62
63          fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
    training$target$y,
64          family = "poisson")
65          fit.oracle <- glmtrans(target = D.training$target,
    source = D.training$source,
66          family = "poisson",transfer.source.id=1:i, cores = 2)
67
68
69          beta <- c(0, rep(0.5, 5), rep(0, 500 - 5))
70          erp3[j,i] <- sqrt(sum((coef(fit.lasso) - beta)^2))
71          erp4[j,i] <- sqrt(sum((fit.oracle$beta - beta)^2))
72          D.training <- models(family = "binomial", type = "all",
    cov.type = 1, Ka = i,
```

```
        K = 40, s = 5, n.target = 200, n.source = rep(100, 40),p
    =500)


        fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
    training$target$y,
        family = "binomial")
        fit.oracle <- glmtrans(target = D.training$target,
    source = D.training$source,
        family = "binomial",transfer.source.id=1:i, cores = 2)


        beta <- c(0, rep(0.5, 5), rep(0, 500 - 5))
        erb3[j,i] <- sqrt(sum((coef(fit.lasso) - beta)^2))
        erb4[j,i] <- sqrt(sum((fit.oracle$beta - beta)^2))
       D.training <- models(family = "gaussian", type = "all",
    cov.type = 1, Ka = i,
        K = 40, s = 5, n.target = 200, n.source = rep(100, 40),p
    =500)


        fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
    training$target$y,
        family = "gaussian")
        fit.oracle <- glmtrans(target = D.training$target,
    source = D.training$source,
        family = "gaussian",transfer.source.id=1:i, cores = 2)


        beta <- c(0, rep(0.5, 5), rep(0, 500 - 5))
        erg3[j,i] <- sqrt(sum((coef(fit.lasso) - beta)^2))
        erg4[j,i] <- sqrt(sum((fit.oracle$beta - beta)^2))
      }
    }

    times<-vector()
    name<-vector()
    errg1m<-vector()
```

```
104       errg3m<-vector()
105       errb1m<-vector()
106       errb3m<-vector()
107       errp1m<-vector()
108       errp3m<-vector()
109       errg1s<-vector()
110       errg3s<-vector()
111       errb1s<-vector()
112       errb3s<-vector()
113       errp1s<-vector()
114       errp3s<-vector()
115       for (i in 1:10)
116       {
117         times=c(times,i)
118         times=c(times,i)
119         name=c(name,"A")
120         name=c(name,"B")
121         errg1m = c(errg1m,mean(erg1[,i]))
122         errg1s = c(errg1s,sd(erg1[,i]))
123         errg1m = c(errg1m,mean(erg2[,i]))
124         errg1s = c(errg1s,sd(erg2[,i]))
125         errg3m = c(errg3m,mean(erg3[,i]))
126         errg3s = c(errg3s,sd(erg3[,i]))
127         errg3m = c(errg3m,mean(erg4[,i]))
128         errg3s = c(errg3s,sd(erg4[,i]))
129         errb3m = c(errb3m,mean(erb3[,i]))
130         errb3s = c(errb3s,sd(erb3[,i]))
131         errb3m = c(errb3m,mean(erb4[,i]))
132         errb3s = c(errb3s,sd(erb4[,i]))
133         errp1m = c(errp1m,mean(erp1[,i]))
134         errp1s = c(errp1s,sd(erp1[,i]))
135         errp1m = c(errp1m,mean(erp2[,i]))
136         errp1s = c(errp1s,sd(erp2[,i]))
137         errp3m = c(errp3m,mean(erp3[,i]))
138         errp3s = c(errp3s,sd(erg3[,i]))
139         errp3m = c(errp3m,mean(erp4[,i]))
140         errp3s = c(errp3s,sd(erp4[,i]))
141
```

```
142       }
143
144       fig1<-data.frame(name,times,errg1m,errg1s)
145       fig3<-data.frame(name,times,errg3m,errg3s)
146       fib3<-data.frame(name,times,errb3m,errb3s)
147       fip1<-data.frame(name,times,errp1m,errp1s)
148       fip3<-data.frame(name,times,errp3m,errp3s)
149       library(ggplot2)
150       ggplot(fig1, aes(x=times, y=errg1m, colour=name)) +
151       geom_errorbar(aes(ymin=errg1m-errg1s, ymax=errg1m+errg1s),
      width=.1,size=0.75) +
152       geom_line() +
153       geom_point(size=10,shape=20)+
154       ylab('l2-Estimation error')+
155       xlab('k')+
156       scale_colour_hue(name="method",
157       breaks=c("A", "B"),
158       labels=c("navie-Lasso", "Ah-Trans-Lasso"))+
159       labs(title="Linear case:h=20")+
160       theme(plot.title = element_text(size=20,hjust=0.5),
161       axis.title = element_text(size = 15),
162       axis.text = element_text(size=15))+
163       scale_x_continuous(breaks=1:10)
164
165       ggplot(fig3, aes(x=times, y=errg3m, colour=name)) +
166       geom_errorbar(aes(ymin=errg3m-errg3s, ymax=errg3m+errg3s),
      width=.1,size=0.75) +
167       geom_line() +
168       geom_point(size=10,shape=20)+
169       ylab('l2-Estimation error')+
170       xlab('k')+
171       scale_colour_hue(name="method",
172       breaks=c("A", "B"),
173       labels=c("navie-Lasso", "Ah-Trans-Lasso"))+
174       labs(title="Linear case:h=40")+
175       theme(plot.title = element_text(size=20,hjust=0.5),
176       axis.title = element_text(size = 15),
177       axis.text = element_text(size=15))+
```

```
178       scale_x_continuous(breaks=1:10)
179
180       ggplot(fib1, aes(x=times, y=errb1m, colour=name)) +
181       geom_errorbar(aes(ymin=errb1m-errb1s, ymax=errb1m+errb1s),
      width=.1,size=0.75) +
182       geom_line() +
183       geom_point(size=10,shape=20)+
184       ylab('l2-Estimation error')+
185       xlab('k')+
186       scale_colour_hue(name="method",
187       breaks=c("A", "B"),
188       labels=c("navie-Lasso", "Ah-Trans-Lasso"))+
189       labs(title="Logistic case:h=20")+
190       theme(plot.title = element_text(size=20,hjust=0.5),
191       axis.title = element_text(size = 15),
192       axis.text = element_text(size=15))+
193       scale_x_continuous(breaks=1:10)
194
195       ggplot(fib3, aes(x=times, y=errb3m, colour=name)) +
196       geom_errorbar(aes(ymin=errb3m-errb3s, ymax=errb3m+errb3s),
      width=.1,size=0.75) +
197       geom_line() +
198       geom_point(size=10,shape=20)+
199       ylab('l2-Estimation error')+
200       xlab('k')+
201       scale_colour_hue(name="method",
202       breaks=c("A", "B"),
203       labels=c("navie-Lasso", "Ah-Trans-Lasso"))+
204       labs(title="Logistic case:h=40")+
205       theme(plot.title = element_text(size=20,hjust=0.5),
206       axis.title = element_text(size = 15),
207       axis.text = element_text(size=15))+
208       scale_x_continuous(breaks=1:10)
209
210       ggplot(fip1, aes(x=times, y=errp1m, colour=name)) +
211       geom_errorbar(aes(ymin=errp1m-errp1s, ymax=errp1m+errp1s),
      width=.1,size=0.75) +
212       geom_line() +
```

```
      geom_point(size=10,shape=20)+
      ylab('l2-Estimation error')+
      xlab('k')+
      scale_colour_hue(name="method",
      breaks=c("A", "B"),
      labels=c("navie-Lasso", "Ah-Trans-Lasso"))+
      labs(title="Poisson case:h=20")+
      theme(plot.title = element_text(size=20,hjust=0.5),
      axis.title = element_text(size = 15),
      axis.text = element_text(size=15))+
      scale_x_continuous(breaks=1:10)

      ggplot(fip3, aes(x=times, y=errp3m, colour=name)) +
      geom_errorbar(aes(ymin=errp3m-errp3s, ymax=errp3m+errp3s),
    width=.1,size=0.75) +
      geom_line() +
      geom_point(size=10,shape=20)+
      ylab('l2-Estimation error')+
      xlab('k')+
      scale_colour_hue(name="method",
      breaks=c("A", "B"),
      labels=c("navie-Lasso", "Ah-Trans-Lasso"))+
      labs(title="Poisson case:h=40")+
      theme(plot.title = element_text(size=20,hjust=0.5),
      axis.title = element_text(size = 15),
      axis.text = element_text(size=15))+
      scale_x_continuous(breaks=1:10)

      #A is unkonwn
      erg12 <- matrix(NA, ncol = 10, nrow = 200)
      erg22 <- matrix(NA, ncol = 10, nrow = 200)
      erg32 <- matrix(NA, ncol = 10, nrow = 200)
      erg42 <- matrix(NA, ncol = 10, nrow = 200)

      for(i in 1:10)
      {
        for(j in 1:200)
        {
```

```
250         D.training <- models(family = "gaussian", type = "all",
    cov.type = 2, Ka = i,
251         K = 10, s = 20, n.target = 200, n.source = rep(200, 10),
    p=2000)
252         fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
    training$target$y,
253         family = "gaussian")
254         fit.pooled <- glmtrans(target = D.training$target,
    source = D.training$source,
255         family = "gaussian", transfer.source.id = "all", cores =
     2)
256         fit.detection <- glmtrans(target = D.training$target,
    source = D.training$source,
257         family = "gaussian", transfer.source.id = "auto", cores
    = 2)
258         fit.oracle <- glmtrans(target = D.training$target,
    source = D.training$source,
259         family = "gaussian", transfer.source.id = 1:i, cores =
    2)
260
261         beta <- c(0, rep(0.5, 20), rep(0, 2000 - 20))
262         erg12[j,i] <- sqrt(sum((coef(fit.lasso) - beta)^2))
263         erg22[j,i] <- sqrt(sum((fit.oracle$beta - beta)^2))
264         erg32[j,i] <- sqrt(sum((fit.pooled$beta - beta)^2))
265         erg42[j,i] <- sqrt(sum((fit.detection$beta - beta)^2))
266       }
267     }
268     erge1<-vector()
269     erge2<-vector()
270     erge3<-vector()
271     erge4<-vector()
272     for(j in 1:200)
273     {
274       D.training <- models(family = "gaussian", type = "all",
    cov.type = 2, Ka = i,
275       K = 10, s = 20, n.target = 200, n.source = rep(200, 10),p
    =2000)
276       fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.
```

```
    training$target$y,
277         family = "gaussian")
278         fit.pooled <- glmtrans(target = D.training$target, source
    = D.training$source,
279         family = "gaussian", transfer.source.id = "all", cores =
    2)
280         fit.detection <- glmtrans(target = D.training$target,
    source = D.training$source,
281         family = "gaussian", transfer.source.id = "auto", cores =
    2)
282         fit.oracle <- glmtrans(target = D.training$target, source
    = D.training$source,
283         family = "gaussian", transfer.source.id = NULL, cores = 2)
284
285         beta <- c(0, rep(0.5, 20), rep(0, 2000 - 20))
286         erge1[j] <- sqrt(sum((coef(fit.lasso) - beta)^2))
287         erge2[j] <- sqrt(sum((fit.oracle$beta - beta)^2))
288         erge3[j] <- sqrt(sum((fit.pooled$beta - beta)^2))
289         erge4[j] <- sqrt(sum((fit.detection$beta - beta)^2))
290       }
291       times<-vector()
292       name<-vector()
293       errgm1<-vector()
294       errgm2<-vector()
295       errgm3<-vector()
296       errgm4<-vector()
297       errgs1<-vector()
298       errgs2<-vector()
299       errgs3<-vector()
300       errgs4<-vector()
301       for (i in 1:10)
302       {
303         times=c(times,i)
304         times=c(times,i)
305         times=c(times,i)
306         times=c(times,i)
307         name=c(name,"A")
308         name=c(name,"B")
```

```
309         name=c(name,"C")
310         name=c(name,"D")
311         errgm1 = c(errgm1,mean(erge1))
312         errgs1 = c(errgs1,sd(erge1))
313         errgm1 = c(errgm1,mean(erge2))
314         errgs1 = c(errgs1,sd(erge2))
315         errgm1 = c(errgm1,mean(erge3))
316         errgs1 = c(errgs1,sd(erge3))
317         errgm1 = c(errgm1,mean(erge4))
318         errgs1 = c(errgs1,sd(erge4))
319
320       }
321
322
323       figg1<-data.frame(name,times,errgm1,errgs1)
324       ggplot(figg1, aes(x=times, y=errgm1, colour=name)) +
325       geom_errorbar(aes(ymin=errgm1-errgs1, ymax=errgm1+errgs1),
    width=.1,size=0.75) +
326       geom_line() +
327       geom_point(size=10,shape=20)+
328       ylab('l2-Estimation error')+
329       xlab('k')+
330       scale_colour_hue(name="method",
331       breaks=c("A", "B","C","D"),
332       labels=c("naive-Lasso", "Ah-Trans-GLM","Pooled-Trans-GLM","
    Trans-GLM"))+
333       labs(title="Linear case:h=20")+
334       theme(plot.title = element_text(size=20,hjust=0.5),
335       axis.title = element_text(size = 15),
336       axis.text = element_text(size=15))+
337       scale_x_continuous(breaks=0:10)
338
339      #three evaluation metrics in Fig2
340       test1 <-  matrix(NA, ncol = 10, nrow = 20)
341       test2 <-  matrix(NA, ncol = 10, nrow = 20)
342       test3 <-  matrix(NA, ncol = 10, nrow = 20)
343       test4 <-  matrix(NA, ncol = 10, nrow = 20)
344       testt1 <-  matrix(NA, ncol = 10, nrow = 20)
```

```
345       testt2 <-  matrix(NA, ncol = 10, nrow = 20)
346       testt3 <-  matrix(NA, ncol = 10, nrow = 20)
347       testt4 <-  matrix(NA, ncol = 10, nrow = 20)
348       testp1 <-  matrix(NA, ncol = 10, nrow = 20)
349       testp2 <-  matrix(NA, ncol = 10, nrow = 20)
350       testp3 <-  matrix(NA, ncol = 10, nrow = 20)
351       testp4 <-  matrix(NA, ncol = 10, nrow = 20)
352       beta <- c(0, rep(0.5, 5), rep(0, 500 - 5))
353       for(t1 in 1:10)
354       {
355         for(t2 in 1:200)
356         {
357           D.training <- models(family = "gaussian", type = "all",
    cov.type = 1, Ka = t1,
358           K = 20, s = 5, n.target = 200, n.source = rep(100, 20),p
    =500)
359           # fit a logistic regression model via two-step transfer
    learning method
360           fit.binomial <-   glmtrans(target = D.training$target,
    source = D.training$source,
361           family = "gaussian",transfer.source.id=1:t1, cores = 2,
    lambda = list(transfer = "lambda.min", detection =
362           "lambda.min"))
363           # calculate the CI based on the point estimate from two-
    step transfer learning method
364           fit.inf <- glmtrans_inf(target = D.training$target,
    source = D.training$source,
365           family = "gaussian", nodewise.transfer.source.id=1:t1,
    beta.hat = fit.binomial$beta, cores = 2)
366
367           rl1=0
368           rl2=0
369           rl3=0
370           rl4=0
371           for(i in 1:501)
372           {
373             if(i==1|i>5)
374             {
```

```
            if(fit.inf$CI$lb[i]<=0 & fit.inf$CI$ub[i]>=0)
            {
              rl1=rl1+1
            }
            rl3=rl3+fit.inf$CI$ub[i]-fit.inf$CI$lb[i]
          }
          else
          {
            if(fit.inf$CI$lb[i]<=0.5 & fit.inf$CI$ub[i]>=0.5)
            {
              rl2=rl2+1
            }
            rl4=rl4+fit.inf$CI$ub[i]-fit.inf$CI$lb[i]
          }
        }
        rl1=rl1/496
        rl2=rl2/5
        rl3=rl3/496
        rl4=rl4/5
        test1[t2,t1]=rl1
        test2[t2,t1]=rl2
        test3[t2,t1]=rl3
        test4[t2,t1]=rl4
        testp1[t2,t1]=sum(abs(fit.inf$beta.hat-beta))-sum((abs(
    fit.inf$beta.hat-beta))[2:6])
        testp2[t2,t1]=sum((abs(fit.inf$beta.hat-beta))[2:6])
        fit.binomial <-   glmtrans(target = D.training$target,
    source = D.training$source,
        family = "gaussian",transfer.source.id=NULL, cores = 2,
    lambda = list(transfer = "lambda.min", detection =
        "lambda.min"))
        # calculate the CI based on the point estimate from two-
    step transfer learning method
        fit.inf <- glmtrans_inf(target = D.training$target,
    source = D.training$source,
        family = "gaussian", nodewise.transfer.source.id=NULL,
    beta.hat = fit.binomial$beta, cores = 2)

```

```
407             rl1=0
408             rl2=0
409             rl3=0
410             rl4=0
411             for(i in 1:501)
412             {
413               if(i==1|i>5)
414               {
415                 if(fit.inf$CI$lb[i]<=0 & fit.inf$CI$ub[i]>=0)
416                 {
417                   rl1=rl1+1
418                 }
419                 rl3=rl3+fit.inf$CI$ub[i]-fit.inf$CI$lb[i]
420               }
421               else
422               {
423                 if(fit.inf$CI$lb[i]<=0.5 & fit.inf$CI$ub[i]>=0.5)
424                 {
425                   rl2=rl2+1
426                 }
427                 rl4=rl4+fit.inf$CI$ub[i]-fit.inf$CI$lb[i]
428               }
429             }
430             rl1=rl1/496
431             rl2=rl2/5
432             rl3=rl3/496
433             rl4=rl4/5
434             testt1[t2,t1]=rl1
435             testt2[t2,t1]=rl2
436             testt3[t2,t1]=rl3
437             testt4[t2,t1]=rl4
438             testp3[t2,t1]=sum(abs(fit.inf$beta.hat-beta))-sum((abs(
    fit.inf$beta.hat-beta))[2:6])
439             testp4[t2,t1]=sum((abs(fit.inf$beta.hat-beta))[2:6])
440           }
441         }
442
443
```