A demonstration of the glmtrans package

Ye Tian and Yang Feng

2025-02-28

We provide an introductory demo of the usage for the `glmtrans` package. This package implements the transfer learning algorithms for high-dimensional generalized linear models (Tian and Feng (2021)).

Introduction

Generalized linear models (GLMs)

Given the predictor $\boldsymbol{x}$, if response y follows the generalized linear models (GLMs), then its distribution satisfies

$$y|\boldsymbol{x} \sim \mathbb{P}(y|\boldsymbol{x}) = \rho(y)\exp\{y\boldsymbol{x}^T\boldsymbol{w} - \psi(\boldsymbol{x}^T\boldsymbol{w})\},$$

where $\psi'(\boldsymbol{x}^T\boldsymbol{w}) = \mathbb{E}(y|\boldsymbol{x})$ is called the *inverse link function* (McCullagh and Nelder (1989)). Another important property is that $\text{Var}(y|\boldsymbol{x}) = \psi''(\boldsymbol{x}^T\boldsymbol{w})$, which is derived from the exponential family property. It is ψ which characterizes different GLMs. For example, in Gaussian model, we have the continuous response y and $\psi(u) = \frac{1}{2}u^2$; in the logistic model, y is binary and $\psi(u) = \log(1 + e^u)$; and in Poisson model, we have the integral response y and $\psi(u) = e^u$.

Two-step transfer learning algorithms

Consider the multi-source transfer learning problem. Suppose we have the *target* data $(X^{(0)}, Y^{(0)}) = \{\boldsymbol{x}_i^{(0)}, y_i^{(0)}\}_{i=1}^{n_0}$ and *source* data $\{(X^{(k)}, Y^{(k)})\}_{k=1}^K = \{\{(\boldsymbol{x}_i^{(k)}, y_i^{(k)})\}_{i=1}^{n_k}\}_{k=1}^K$ for $k = 1, \ldots, K$. Denote the target coefficient $\boldsymbol{\beta} = \boldsymbol{w}^{(0)}$. Suppose target and source data follow the GLM as

$$y^{(k)}|\boldsymbol{x}^{(k)} \sim \mathbb{P}(y^{(k)}|\boldsymbol{x}^{(k)}) = \rho(y^{(k)})\exp\{y^{(k)}(\boldsymbol{x}^{(k)})^T\boldsymbol{w}^{(k)} - \psi((\boldsymbol{x}^{(k)})^T\boldsymbol{w}^{(k)})\}.$$

In order to borrow information from transferable sources, Bastani (2020) and Li, Cai, and Li (2020) developed *two-step transfer learning algorithms* for high-dimensional linear models. In the first step, an approximate estimator is achieved via the information from the target data and useful source data. In the second step, the target data is used to debias the estimator obtained from the first step, leading to the final estimator.

Tian and Feng (2021) extends the idea into GLM and proposes the corresponding *oracle* algorithm, which can be easily applied when transferable sources are known. It is proved to enjoy a sharper bound of ℓ_2-estimation error when the transferable source and target data are sufficiently similar.

Transferable source detection

In the multi-source transfer learning problem, some adversarial sources may share little similarity with the target, which can mislead the fitted model. We call this phenomenon as *negative transfer* (Pan and Yang (2009), Torrey and Shavlik (2010), Weiss, Khoshgoftaar, and Wang (2016)).

To detect which sources are transferable, Tian and Feng (2021) develops an *algorithm-free* detection approach. Simply speaking, it tries to compute the gain for transferring each single source and compare it with the baseline where only the target data is used. The sources enjoying significant performance gain compared with the baseline are regarded as transferable ones. Tian and Feng (2021) also proves the detection consistency property for this method under the high-dimensional GLM setting.

Implementation

The implementation of this package leverages on package `glmnet`, which applies the cyclic coordinate gradient descent and is very efficient (Friedman, Hastie, and Tibshirani (2010)). We use the argument `offset` provided by the function `glmnet` and `cv.glmnet` to implement our two-step algorithms. Besides Lasso (Tibshirani (1996)), this package can adapt the elastic net type penalty (Zou and Hastie (2005)).

Installation

`glmtrans` is now available on CRAN and can be easily intalled by one-line code.

```r
install.packages("glmtrans", repos = "http://cran.us.r-project.org")
```

Then we can load the package:

```r
library(glmtrans)
```

Example Codes on Simulated Data

In this section, we show the user how to use the provided functions to fit the model, make predictions and visualize the results. We take logistic data as an example.

Model fitting and prediction

We first generate some logistic data through function `models`. For target data, we set the coefficient vector $\boldsymbol{\beta} = (0.5 \cdot \mathbf{1}_s, \mathbf{0}_{p-s})$, where $p = 500$ and $s = 10$. For k in transferable source index set $\mathcal{A}$, let $\boldsymbol{w}^{(k)} = \boldsymbol{\beta} + h/p \cdot \mathcal{R}_p^{(k)}$, where $h = 5$ and $\mathcal{R}_p^{(k)}$ are 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 $k \neq k'$. The coefficient of non-transferable sources is set to be $\boldsymbol{\xi} + h/p \cdot \mathcal{R}_p^{(k)}$. And $\boldsymbol{\xi}_{S'} = 0.5 \cdot \mathbf{1}_{2s}$, $\boldsymbol{\xi}_{(S')^c} = \mathbf{0}_{p-2s}$, where $S' = S_1' \cup S_2'$ and $|S_1'| = |S_2'| = s = 10$. $S_1' = \{s+1, \ldots, 2s\}$, and S_2' is randomly sampled from $\{2s+1, \ldots, p\}$. We also add an intercept 0.5. The generating procedure of each non-transferable source data is independent. The target predictor $\boldsymbol{x}_i^{(k)} \overset{i.i.d.}{\sim} N(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = (0.9^{|i-j|})_{p \times p}$. The source predictor $\boldsymbol{x}_i^{(k)} \overset{i.i.d.}{\sim} t_4$. The target sample size $n_0 = 100$ and each source sample size $n_k = 100$ for any $k = 1, \ldots, K$. Let $K = 5$, $\mathcal{A} = \{1, 2, 3\}$.

We generate the training data as follows.

```r
set.seed(1, kind = "L'Ecuyer-CMRG")
D.training <- models(family = "binomial", type = "all", cov.type = 2, Ka = 3,
    K = 5, s = 10, n.target = 100, n.source = rep(100, 5))
```

Then suppose we know $\mathcal{A}$, let's fit an "oracle" GLM transfer learning model on the target data and source data in $\mathcal{A}$ by the oracle algorithm. We denote this procedure as Oralce-Trans-GLM.

```r
fit.oracle <- glmtrans(target = D.training$target, source = D.training$source,
    family = "binomial", transfer.source.id = 1:3, cores = 2)
```

Notice that we set the argument `transfer.source.id` equal to $\mathcal{A} = \{1, 2, 3\}$ to transfer only the first three sources.

And the output of `glmtrans` function is an object belonging to S3 class "glmtrans". It contains:

- beta: the estimated coefficient vector.
- family: the response type.
- transfer.source.id: the transferable souce index. If in the input, `transfer.source.id = 1:length(source)` or `transfer.source.id = "all"`, then the outputed `transfer.source.id = 1:length(source)`. If the inputed `transfer.source.id = "auto"`, only transferable source detected by the algorithm will be outputed.
- fitting.list:
  - w_a: the estimator obtained from the transferring step.
  - delta_a: the estimator obtained from the debiasing step.
  - target.valid.loss: the validation (or cross-validation) loss on target data. Only available when `transfer.source.id = "auto"`.
  - source.loss: the loss on each source data. Only available when `transfer.source.id = "auto"`.
  - epsilon0: the threshold to determine transferability will be set as (1+`epsilon0`)·loss of validation (cv) target data. Only available when `transfer.source.id = "auto"`.
  - threshold: the threshold to determine transferability. Only available when `transfer.source.id = "auto"`.

Then suppose we do not know $\mathcal{A}$, let's set `transfer.source.id = "auto"` to apply the transferable source detection algorithm to get estimate $\hat{\mathcal{A}}$. After that, `glmtrans` will automatically run the oracle algorithm on $\hat{\mathcal{A}}$ to fit the model. We denote the approach as Trans-GLM.

```r
fit.detection <- glmtrans(target = D.training$target, source = D.training$source,
    family = "binomial", transfer.source.id = "auto", cores = 2)
```

```
## Loss difference between source data and the threshold: (negative to be transferable)
## Source 1: -0.046214
## Source 2: -0.064628
## Source 3: -0.105844
## Source 4: 0.113911
## Source 5: 0.211745
##
## Source data set(s) 1, 2, 3 are transferable!
```

From the results, we could see that $\mathcal{A} = \{1, 2, 3\}$ is successfully detected via the detection algorithm. Next, to demonstrate the effectiveness of GLM transfer learning algorithm and the transferable source detection algorithm, we also fit the naive Lasso on target data (Lasso) and transfer learning model using all source data (Pooled-Trans-GLM) as baselines.

```r
library(glmnet)
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y,
    family = "binomial")
fit.pooled <- glmtrans(target = D.training$target, source = D.training$source,
    family = "binomial", transfer.source.id = "all", cores = 2)
```

Finally, we compare the ℓ_2-estimation errors of the target coefficient $\boldsymbol{\beta}$ by different methods.

```r
beta <- c(0, rep(0.5, 10), rep(0, 500 - 10))
er <- numeric(4)
names(er) <- c("Lasso", "Pooled-Trans-GLM", "Trans-GLM", "Oracle-Trans-GLM")
er["Lasso"] <- sqrt(sum((coef(fit.lasso) - beta)^2))
er["Pooled-Trans-GLM"] <- sqrt(sum((fit.pooled$beta - beta)^2))
er["Trans-GLM"] <- sqrt(sum((fit.detection$beta - beta)^2))
er["Oracle-Trans-GLM"] <- sqrt(sum((fit.oracle$beta - beta)^2))
er
```

```
##            Lasso Pooled-Trans-GLM        Trans-GLM Oracle-Trans-GLM
##         1.322086         1.298818         1.100324         1.185506
```

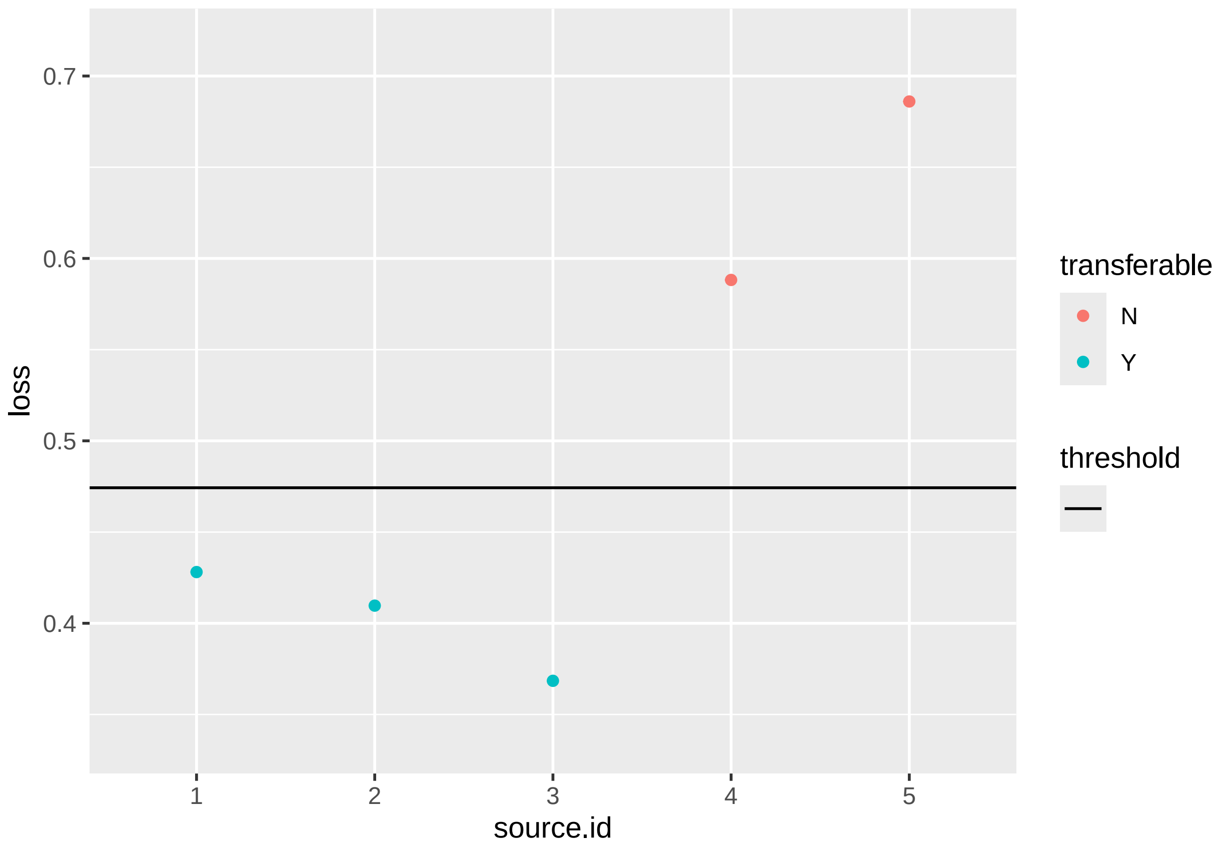
Note that the transfer learning models outperform the classical Lasso fitted on target data. And due to negative transfer, Pooled-Trans-GLM performs worse than Oracle-Trans-GLM. By correctly detecting $\mathcal{A}$, the behavior of Trans-GLM mimics the oracle.

Plotting the source detection result

We could visualize the transferable source detection results by applying `plot` function on objects in class "glmtrans" or "glmtrans_source_detection". Loss of each source and the transferability threshold will be drawed. Function `glmtrans` outputs objects in class "glmtrans", while function `source_detection` outputs objects in class "glmtrans_source_detection". The function `source_detection` detects $\mathcal{A}$ without the post-detecting model fitting step.

Call `plot` function to visualize the results as follows.

```r
plot(fit.detection)
```



Reference

Bastani, Hamsa. 2020. "Predicting with Proxies: Transfer Learning in High Dimension." *Management Science*.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. 2010. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33 (1): 1.

Li, Sai, T Tony Cai, and Hongzhe Li. 2020. "Transfer Learning for High-Dimensional Linear Regression: Prediction, Estimation, and Minimax Optimality." *arXiv Preprint arXiv:2006.10593*.

McCullagh, P, and John A Nelder. 1989. *Generalized Linear Models*. Vol. 37. CRC Press.

Pan, Sinno Jialin, and Qiang Yang. 2009. "A Survey on Transfer Learning." *IEEE Transactions on Knowledge and Data Engineering* 22 (10): 1345–59.

Tian, Ye, and Yang Feng. 2021. "Transfer Learning with High-Dimensional Generalized Linear Models." *Submitted*.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society: Series B (Methodological)* 58 (1): 267–88.

Torrey, Lisa, and Jude Shavlik. 2010. "Transfer Learning." In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 242–64. IGI global.

Weiss, Karl, Taghi M Khoshgoftaar, and DingDing Wang. 2016. "A Survey of Transfer Learning." *Journal of Big Data* 3 (1): 1–40.

Zou, Hui, and Trevor Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2): 301–20.