Package ‘glmtrans’

February 8, 2022

Type Package
Title Transfer Learning under Regularized Generalized Linear Models
Version 2.0.0
Description We provide an efficient implementation for two-step multi-source transfer learning algorithms in high-dimensional generalized linear models (GLMs). The elastic-net penalized GLM with three popular families, including linear, logistic and Poisson regression models, can be fitted. To avoid negative transfer, a transferable source detection algorithm is proposed. We also provides visualization for the transferable source detection results. The relevant paper is available on arXiv: <arXiv:2105.14328>.
Imports glmnet, ggplot2, foreach, doParallel, caret, assertthat, formatR, stats
License GPL-2
Depends R (>= 3.5.0)
Encoding UTF-8
RoxygenNote 7.1.2
Suggests knitr, rmarkdown
VignetteBuilder knitr
NeedsCompilation no
Author Ye Tian [aut, cre], Yang Feng [aut]
Maintainer Ye Tian <ye.t@columbia.edu>
Repository CRAN
Date/Publication 2022-02-08 07:30:02 UTC

R topics documented:

- glmtrans ........................................... 2
- glmtrans_inf ..................................... 6
- models ............................................. 8
- plot glmtrans .................................... 10
glmtrans

Fit a transfer learning generalized linear model (GLM) with elasticnet regularization.

Description

Fit a transfer learning generalized linear model through elastic net regularization with target data set and multiple source data sets. It also implements a transferable source detection algorithm, which helps avoid negative transfer in practice. Currently can deal with Gaussian, logistic and Poisson models.

Usage

glmtrans(
  target,
  source = NULL,
  family = c("gaussian", "binomial", "poisson"),
  transfer.source.id = "auto",
  alpha = 1,
  standardize = TRUE,
  intercept = TRUE,
  nfolds = 10,
  cores = 1,
  valid.proportion = NULL,
  valid.nfolds = 3,
  lambda = c(transfer = "lambda.1se", debias = "lambda.min", detection = "lambda.1se"),
  detection.info = TRUE,
  target.weights = NULL,
  source.weights = NULL,
  C0 = 2,
  ...
)

Arguments

target target data. Should be a list with elements x and y, where x indicates a predictor matrix with each row/column as a(n) observation/variable, and y indicates the response vector.

source source data. Should be a list with some sublists, where each of the sublist is a source data set, having elements x and y with the same meaning as in target data.
family
response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian".
  • "gaussian": Gaussian distribution.
  • "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.
  • "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.

transfer.source.id
transferable source indices. Can be either a subset of \{1,...,length(source)\}, "all" or "auto". Default = "auto".
  • a subset of \{1,...,length(source)\}: only transfer sources with the specific indices.
  • "all": transfer all sources.
  • "auto": run transferable source detection algorithm to automatically detect which sources to transfer. For the algorithm, refer to the documentation of function source_detection.

alpha
the elasticnet mixing parameter, with \(0 \leq \alpha \leq 1\). The penalty is defined as
\[
(1 - \alpha)/2\|\beta\|^2 + \alpha\|\beta\|_1
\]
. \(\alpha = 1\) encodes the lasso penalty while \(\alpha = 0\) encodes the ridge penalty. Default = 1.

standardize
the logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is TRUE.

intercept
the logical indicator of whether the intercept should be fitted or not. Default = TRUE.

nfolds
the number of folds. Used in the cross-validation for GLM elastic net fitting procedure. Default = 10. Smallest value allowable is nfolds = 3.

cores
the number of cores used for parallel computing. Default = 1.

valid.proportion
the proportion of target data to be used as validation data when detecting transferable sources. Useful only when transfer.source.id = "auto". Default = NULL, meaning that the cross-validation will be applied.

valid.nfolds
the number of folds used in cross-validation procedure when detecting transferable sources. Useful only when transfer.source.id = "auto" and valid.proportion = NULL. Default = 3.

lambda
a vector indicating the choice of lambdas in transferring, debiasing and detection steps. Should be a vector with names "transfer", "debias", and "detection", each component of which can be either "lambda.min" or "lambda.1se". Component transfer is the lambda (the penalty parameter) used in transferring step. Component debias is the lambda used in debiasing step. Component detection is the lambda used in the transferable source detection algorithm. Default choice of lambda.transfer and lambda.detection are "lambda.1se", while default lambda.debias = "lambda.min". If the user wants to change the default setting, input a vector with corresponding lambda.transfer/lambda.debias/lambda.detection
names and corresponding values. Examples: `lambda = list(transfer = "lambda.min", debias = "lambda.1se")`; `lambda = list(transfer = "lambda.min", detection = "lambda.min")`.

- "lambda.min": value of lambda that gives minimum mean cross-validated error in the sequence of lambda.
- "lambda.1se": largest value of lambda such that error is within 1 standard error of the minimum.

`detection.info` the logistic flag indicating whether to print detection information or not. Useful only when `transfer.source.id = "auto"`. Default = TRUE.

`target.weights` weight vector for each target instance. Should be a vector with the same length of target response. Default = NULL, which makes all instances equal-weighted.

`source.weights` a list of weight vectors for the instances from each source. Should be a list with the same length of the number of sources. Default = NULL, which makes all instances equal-weighted.

`C0` the constant used in the transferable source detection algorithm. See Algorithm 2 in Tian, Y. and Feng, Y., 2021. Default = 2.

... additional arguments.

**Value**

An object with S3 class "glmtrans".

- `beta` the estimated coefficient vector.
- `family` the response type.
- `transfer.source.id` the transferable source 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` a list of other parameters of the fitted model.

- `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"`.
- `threshold` the threshold to determine transferability. Only available when `transfer.source.id = "auto"`.

**References**


See Also

`predict.glmtrans`, `source_detection`, `models`, `plot.glmtrans`, `cv.glmnet`, `glmnet`.

Examples

```r
set.seed(0, kind = "L'Ecuyer-CMRG")

# fit a linear regression model
D.training <- models("gaussian", type = "all", n.target = 100, K = 2, p = 500)
D.test <- models("gaussian", type = "target", n.target = 100, p = 500)
fit.gaussian <- glmtrans(D.training$target, D.training$source)
y.pred.glmtrans <- predict(fit.gaussian, D.test$target$x)

# compare the test MSE with classical Lasso fitted on target data
library(glmnet)
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y)
y.pred.lasso <- predict(fit.lasso, D.test$target$x)
mean((y.pred.glmtrans - D.test$target$y)^2)
mean((y.pred.lasso - D.test$target$y)^2)

# fit a logistic regression model
D.training <- models("binomial", type = "all", n.target = 100, K = 2, p = 500)
D.test <- models("binomial", type = "target", n.target = 100, p = 500)
fit.binomial <- glmtrans(D.training$target, D.training$source, family = "binomial")
y.pred.glmtrans <- predict(fit.binomial, D.test$target$x)

# compare the test error with classical Lasso fitted on target data
library(glmnet)
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y, family = "binomial")
y.pred.lasso <- as.numeric(predict(fit.lasso, D.test$target$x, type = "class"))
mean(y.pred.glmtrans != D.test$target$y)
mean(y.pred.lasso != D.test$target$y)

# fit a Poisson regression model
D.training <- models("poisson", type = "all", n.target = 100, K = 2, p = 500)
D.test <- models("poisson", type = "target", n.target = 100, p = 500)
fit.poisson <- glmtrans(D.training$target, D.training$source, family = "poisson")
y.pred.glmtrans <- predict(fit.poisson, D.test$target$x)

# compare the test MSE with classical Lasso fitted on target data
library(glmnet)
fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y, family = "poisson")
y.pred.lasso <- as.numeric(predict(fit.lasso, D.test$target$x, type = "response"))
```

fit.lasso <- cv.glmnet(x = D.training$target$x, y = D.training$target$y, family = "poisson")
y.pred.lasso <- as.numeric(predict(fit.lasso, D.test$target$x, type = "response"))

mean((y.pred.glmtrans - D.test$target$y)^2)
mean((y.pred.lasso - D.test$target$y)^2)

---

**glmtrans_inf**

Calculate asymptotic confidence intervals based on desparsified Lasso and two-step transfer learning method.

**Description**

Given the point estimate of the coefficient vector from glmtrans, calculate the asymptotic confidence interval of each component. The detailed inference algorithm can be found as Algorithm 3 in the latest version of Tian, Y. and Feng, Y., 2021. The algorithm is constructed based on a modified version of desparsified Lasso (Van de Geer, S. et al, 2014; Dezeure, R. et al, 2015).

**Usage**

```r
glmtrans_inf(
  target,
  source = NULL,
  family = c("gaussian", "binomial", "poisson"),
  beta.hat = NULL,
  nodewise.transfer.source.id = "all",
  cores = 1,
  level = 0.95,
  intercept = TRUE,
  ...
)
```

**Arguments**

- **target**: target data. Should be a list with elements x and y, where x indicates a predictor matrix with each row/column as a(n) observation/variable, and y indicates the response vector.
- **source**: source data. Should be a list with some sublists, where each of the sublist is a source data set, having elements x and y with the same meaning as in target data.
- **family**: response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian".
  - "gaussian": Gaussian distribution.
  - "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.
  - "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.
glmtrans_inf

- **beta.hat**: initial estimate of the coefficient vector (the intercept should be the first component). Can be from the output of function `glmtrans`.

- **nodewise.transfer.source.id**: transferable source indices in the inference (the set A in Algorithm 3 of Tian, Y. and Feng, Y., 2021). Can be either a subset of \( \{1, \ldots, \text{length(source)}\} \), "all" or `NULL`. Default = "all".
  - a subset of \( \{1, \ldots, \text{length(source)}\} \): only transfer sources with the specific indices.
  - "all": transfer all sources.
  - `NULL`: don’t transfer any sources and only use target data.

- **cores**: the number of cores used for parallel computing. Default = 1.

- **level**: the level of confidence interval. Default = 0.95. Note that the level here refers to the asymptotic level of confidence interval of a single component rather than the multiple intervals.

- **intercept**: whether the model includes the intercept or not. Default = TRUE. Should be set as TRUE if the intercept of `beta.hat` is not zero.

- **...**: additional arguments.

**Value**

- a list of output. `b.hat = b.hat`, `beta.hat = beta.hat`, `CI = CI`, `var.est = var.est`

- **b.hat**: the center of confidence intervals. A \( p \)-dimensional vector, where \( p \) is the number of predictors.

- **beta.hat**: the initial estimate of the coefficient vector (the same as input).

- **CI**: confidence intervals (CIs) with the specific level. A \( p \) by 3 matrix, where three columns indicate the center, lower limit and upper limit of CIs, respectively. Each row represents a coefficient component.

- **var.est**: the estimate of variances in the CLT (Theta transpose times Sigma times Theta, in section 2.5 of Tian, Y. and Feng, Y., 2021). A \( p \)-dimensional vector, where \( p \) is the number of predictors.

**References**


**See Also**

- `glmtrans`
## Not run:

```r
set.seed(0, kind = "L'Ecuyer-CMRG")
```

# generate binomial data
`D.training <- models("binomial", type = "all", K = 2, p = 200)`

# fit a logistic regression model via two-step transfer learning method
`fit.binomial <- glmtrans(D.training$target, D.training$source, family = "binomial")`

# 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 = "binomial", beta.hat = fit.binomial$beta, cores = 2)`

## End(Not run)

---

### Description

Generate data from Gaussian, logistic and Poisson models used in the simulation part of Tian, Y. and Feng, Y., 2021.

### Usage

```r
models(
  family = c("gaussian", "binomial", "poisson"),
  type = c("all", "source", "target"),
  cov.type = 1,
  h = 5,
  K = 5,
  n.target = 200,
  n.source = rep(100, K),
  s = 5,
  p = 500,
  Ka = K
)
```

### Arguments

- **family**: response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian".
  - "gaussian": Gaussian distribution.
  - "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.
• "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.

**type**
the type of generated data. Can be "all", "source" or "target".

**cov.type**
the type of covariates. Can be 1 or 2 (numerical). If it equals to 1, the predictors will be generated from the distribution used in Section 4.1.1 (Ah-Trans-GLM) in the latest version of Tian, Y. and Feng, Y., 2021. If it equals to 2, the predictors will be generated from the distribution used in Section 4.1.2 (When transferable sources are unknown).

• "all": generate a list with a target data set of size \(n.\) target and \(K\) source data set of size \(n.\) source.
• "source": generate a list with \(K\) source data set of size \(n.\) source.
• "target": generate a list with a target data set of size \(n.\) target.

**h**
measures the deviation (\(l_1\)-norm) of transferable source coefficient from the target coefficient.

**K**
the number of source data sets. Default = 5.

**n.target**
the sample size of target data. Should be a positive integer. Default = 100.

**n.source**
the sample size of each source data. Should be a vector of length \(K\). Default is a \(K\)-vector with all elements 150.

**s**
how many components in the target coefficient are non-zero, which controls the sparsity of target problem. Default = 15.

**p**
the dimension of data. Default = 1000.

**Ka**
the number of transferable sources. Should be an integer between 0 and \(K\). Default = \(K\).

**Value**
a list of data sets which depend on the value of type.

- **type = "all"**: a list of two components named "target" and "source" storing the target and source data, respectively. Component source is a list containing \(K\) components with the first \(Ka\) ones \(h\)-transferable and the remaining ones \(h\)-nontransferable. The target data set and each source data set have components "x" and "y", as the predictors and responses, respectively.
- **type = "source"**: a list with a signle component "source". This component contains a list of \(K\) components with the first \(Ka\) ones \(h\)-transferable and the remaining ones \(h\)-nontransferable. Each source data set has components "x" and "y", as the predictors and responses, respectively.
- **type = "target"**: a list with a signle component "target". This component contains another list with components "x" and "y", as the predictors and responses of target data, respectively.

**References**

**See Also**
glmtrans.
Examples

```r
set.seed(0, kind = "L'Ecuyer-CMRG")

D.all <- models("binomial", type = "all")
D.target <- models("binomial", type = "target")
D.source <- models("binomial", type = "source")
```

plot.glmtrans

Visualize the losses of different sources and the threshold to determine transferability.

Description

Plot the losses of different sources and the threshold to determine transferability for object with class "glmtrans" or "glmtrans_source_detection".

Usage

```r
## S3 method for class 'glmtrans'
plot(x, ...)
```

Arguments

- `x`: an object from class "glmtrans" or "glmtrans_source_detection", which are the output of functions glmtrans and source_detection, respectively.
- `...`: additional arguments that can be passed to ggplot function.

Value

A "ggplot" visualization with the transferable threshold and losses of different sources.

References


See Also

`glmtrans`, `source_detection`, `ggplot`.
Examples

set.seed(1, kind = "L’Ecuyer-CMRG")

D.training <- models("gaussian", K = 2, p = 500, Ka = 1)

# plot for class "glmtrans"
fit.gaussian <- glmtrans(D.training$target, D.training$source)
plot(fit.gaussian)

# plot for class "glmtrans_source_detection"
detection.gaussian <- source_detection(D.training$target, D.training$source)
plot(detection.gaussian)

predict.glmtrans

Predict for new data from a "glmtrans" object.

Description

Predict from a "glmtrans" object based on new observation data. There are various types of output available.

Usage

## S3 method for class 'glmtrans'
predict(
  object,
  newx,
  type = c("link", "response", "class", "integral response"),
  ...
)

Arguments

object an object from class "glmtrans", which comes from the output of function glmtrans.
newx the matrix of new values for predictors at which predictions are to be made. Should be in accordance with the data for training object.
type the type of prediction. Default = "link".
... additional arguments.

- "link" the linear predictors. When family = "gaussian", it is the same as the predicted responses.
- "response" gives the predicted probabilities when family = "binomial", the predicted mean when family = "poisson", and the predicted responses when family = "gaussian".
• "class" the predicted 0/1 responses for logistic distribution. Applies only when family = "binomial".
• "integral response" the predicted integral response for Poisson distribution. Applies only when family = "poisson".

Value

the predicted result on new data, which depends on type.

References


See Also

glmtrans.

Examples

set.seed(1, kind = "L'EcuY-CMRG")

# fit a logistic model
D.training <- models("binomial", type = "all", K = 1, p = 500)
D.test <- models("binomial", type = "target", n.target = 10, p = 500)
fit.binomial <- glmtrans(D.training$target, D.training$source, family = "binomial")
predict(fit.binomial, D.test$target$x, type = "link")
predict(fit.binomial, D.test$target$x, type = "response")
predict(fit.binomial, D.test$target$x, type = "class")

# fit a Poisson model
D.training <- models("poisson", type = "all", K = 1, p = 500)
D.test <- models("poisson", type = "target", n.target = 10, p = 500)
fit.poisson <- glmtrans(D.training$target, D.training$source, family = "poisson")
predict(fit.poisson, D.test$target$x, type = "response")
predict(fit.poisson, D.test$target$x, type = "integral response")
source_detection

Usage

## S3 method for class 'glmtrans'
print(x, ...)

Arguments

x fitted "glmtrans" model object.
...
additional arguments.

Value

No value is returned.

See Also

glmtrans.

Examples

set.seed(1, kind = "L'Ecuyer-CMRG")

# fit a linear model
D.training <- models("gaussian", K = 2, p = 500)
fit.gaussian <- glmtrans(D.training$target, D.training$source)

fit.gaussian

source_detection

Transferable source detection for GLM transfer learning algorithm.

Description

Detect transferable sources from multiple source data sets. Currently can deal with Gaussian, logistic and Poisson models.

Usage

source_detection(
    target,
    source = NULL,
    family = c("gaussian", "binomial", "poisson"),
    alpha = 1,
    standardize = TRUE,
    intercept = TRUE,
    nfolds = 10,
    cores = 1,
    valid.nfolds = 3,
source_detection

lambda = "lambda.1se",
detection.info = TRUE,
target.weights = NULL,
source.weights = NULL,
C0 = 2,
...
)

Arguments

target target data. Should be a list with elements x and y, where x indicates a predictor matrix with each row/column as a(n) observation/variable, and y indicates the response vector.

source source data. Should be a list with some sublists, where each of the sublist is a source data set, having elements x and y with the same meaning as in target data.

family response type. Can be "gaussian", "binomial" or "poisson". Default = "gaussian".
  • "gaussian": Gaussian distribution.
  • "binomial": logistic distribution. When family = "binomial", the input response in both target and source should be 0/1.
  • "poisson": poisson distribution. When family = "poisson", the input response in both target and source should be non-negative.

alpha the elasticnet mixing parameter, with $0 \leq \alpha \leq 1$. The penalty is defined as

$$(1 - \alpha)/2||\beta||^2_2 + \alpha||\beta||_1$$

. alpha = 1 encodes the lasso penalty while alpha = 0 encodes the ridge penalty. Default = 1.

standardize the logical flag for x variable standardization, prior to fitting the model sequence. The coefficients are always returned on the original scale. Default is TRUE.

intercept the logical indicator of whether the intercept should be fitted or not. Default = TRUE.

nfolds the number of folds. Used in the cross-validation for GLM elastic net fitting procedure. Default = 10. Smallest value allowable is nfolds = 3.

cores the number of cores used for parallel computing. Default = 1.

valid.nfolds the number of folds used in cross-validation procedure when detecting transferable sources. Useful only when transfer.source.id = "auto". Default = 3.

lambda lambda (the penalty parameter) used in the transferable source detection algorithm. Can be either "lambda.min" or "lambda.1se". Default = "lambda.1se".

detection.info the logistic flag indicating whether to print detection information or not. Useful only when transfer.source.id = "auto". Default = TRUE.

target.weights weight vector for each target instance. Should be a vector with the same length of target response. Default = NULL, which makes all instances equal-weighted.
source.weights

A list of weight vectors for the instances from each source. Should be a list with the same length of the number of sources. Default = NULL, which makes all instances equal-weighted.

C0


- "lambda.min": value of lambda that gives minimum mean cross-validated error in the sequence of lambda.
- "lambda.1se": largest value of lambda such that error is within 1 standard error of the minimum.

... additional arguments.

Value

An object with S3 class "glmtrans_source_detection".

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".

threshold

The threshold to determine transferability. Only available when transfer.source.id = "auto".

Note

source.loss and threshold outputed by source_detection can be visualized by function plot.glmtrans.

References


See Also

glmtrans, predict.glmtrans, models, plot.glmtrans, cv.glmnet, glmnet.
Examples

```r
set.seed(0, kind = "L'Ecuyer-CMRG")

# study the linear model
D.training <- models("gaussian", type = "all", K = 2, p = 500, Ka = 1, n.target = 100, cov.type = 2)
detection.gaussian <- source_detection(D.training$target, D.training$source)
detection.gaussian$transferable.source.id

# study the logistic model
D.training <- models("binomial", type = "all", K = 2, p = 500, Ka = 1, n.target = 100, cov.type = 2)
detection.binomial <- source_detection(D.training$target, D.training$source,
    family = "binomial", cores = 2)
detection.binomial$transferable.source.id

# study Poisson model
D.training <- models("poisson", type = "all", K = 2, p = 500, Ka = 1, n.target = 100, cov.type = 2)
detection.poisson <- source_detection(D.training$target, D.training$source,
    family = "poisson", cores = 2)
detection.poisson$transferable.source.id
```
Index

cv.glmnet, 5, 15

ggplot, 10

glmnet, 5, 15

glmtrans, 2, 7, 9, 10, 12, 13, 15

glmtrans_inf, 6

models, 5, 8, 15

plot.glmtrans, 5, 10, 15

plot.glmtrans_source_detection

(predict.glmtrans), 10

predict.glmtrans, 5, 11, 15

print.glmtrans, 12

source_detection, 5, 10, 13