Package ‘glmtrans’

April 28, 2021

Type Package

Title Transfer Learning with Regularized Generalized Linear Models

Version 1.0.0

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 models, can be fitted. To avoid negative transfer, a transferable source detection algorithm is available. We also provide visualization for the transferable source detection results. A relevant paper by Ye Tian and Yang Feng (2021) will be available soon on arXiv.

Imports glmnet, ggplot2, foreach, doParallel, caret, assertthat, formatR, stats

License GPL-2

Depends R (>= 3.5.0)

Encoding UTF-8

LazyData TRUE

RoxygenNote 7.1.0

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 2021-04-28 07:50:02 UTC

R topics documented:

glmtrans ................................................................. 2
micromass .................................................................. 6
models ..................................................................... 7
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,
  epsilon0 = 0.01,
  cores = 1,
  valid.proportion = NULL,
  valid.nfolds = 3,
  lambda.transfer = "lambda.1se",
  lambda.debias = "lambda.min",
  lambda.detection = "lambda.min",
  detection.info = 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.

transfer.source.id: transferable source index. Can be either a subset of \{1,\ldots,length(source)\}, "all" or "auto". Default = "auto".

- a subset of \{1,\ldots,length(source)\}: only transfer sources with the specific index.
- "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 ≤ α ≤ 1. The penalty is defined as

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

. α = 1 encodes the lasso penalty while α = 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.

epsilon0: a positive number. Useful only when transfer.source.id = "auto". The threshold to determine transferability will be set as \((1+\epsilon_0)\times(\text{validation orcross-validation loss of target data})\). Default = 0.01. For details, refer to Algorithm 3 in Tian, Y. and Feng, Y., 2021.

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.transfer: lambda (the penalty parameter) used in transferrring step. Can be either "lambda.min" or "lambda.1se". Default = "lambda.1se". The sequence of lambda will be generated automatically by cv.glmnet. For more details about lambda choice, see the documentation of cv.glmnet in package glmnet.
• "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.

lambda.debias lambda (the penalty parameter) used in debiasing step. Can be either "lambda.min" or "lambda.1se". Default = "lambda.min".

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

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

... 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_ath estimator obtained from the transferring step.
• delta_ath 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 \(1+\epsilon_0\) * loss of validation(cv)targetdata. 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

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

# fit a linear model
D.training <- models("gaussian", type = "all", 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 model
D.training <- models("binomial", type = "all", 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, type = "class")

# 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 model
D.training <- models("poisson", type = "all", 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, type = "response")

# compare the test MSE with classical Lasso fitted on target data
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"))
Description

A data set about the identification of microorganisms (Mahe, P. et al., 2014). The original data set includes positive and negative gram from 9 genera, 20 species. There are 541 observations and 1300 variables in total. To verify the power of GLM transfer learning algorithms, Tian, Y. and Feng, Y., 2021 divides the whole data into 10 groups, each of which contains two species of gram data.

Usage

micromass

Format

A list with 10 groups of gram, each of which includes data from two species. There are 1300 variables in each group, characterizing the features of the gram. The pair of species contained in each group:

- "QBG.CRP-JNH.ZIJ"
- "AUG.AEX-RTO.JFR"
- "QWP.LRO-RTO.TQH"
- "AUG.HSS-QWP.DRH"
- "QBG.KGI-JNH.FLH"
- "VVJ.KWJ-BUT.DNW"
- "VVJ.KSF-BUT.TRH"
- "NYV.VCE-EMD.FZO"
- "NYV.XSY-EMD.WXC"
- "BUT.BIK-BUT.YZE"

Source

Original data link: https://archive.ics.uci.edu/ml/datasets/MicroMass#

References


Generate data from Gaussian, logistic and Poisson models.

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"),
  h = 5,
  K = 5,
  n.target = 100,
  n.source = rep(150, K),
  s = 15,
  p = 1000,
  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".
  - "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.
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 single 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 single 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
set.seed(1, 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
## S3 method for class 'glmtrans'
plot(x, ...)
predict.glmtrans

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)
Usage

```r
## 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

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

---

**print.glmtrans**

Print a fitted "glmtrans" object.

### Description

Similar to the usual print methods, this function summarizes results from a fitted "glmtrans" object.

### Usage

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

### Arguments

- `x`: fitted "glmtrans" model object.
- `...`: additional arguments.

### Value

No value is returned.

### See Also

- `glmtrans`

### Examples

```r
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, 
    epsilon0 = 0.01, 
    cores = 1, 
    valid.proportion = NULL, 
    valid.nfolds = 3, 
    lambda.detection = "lambda.min", 
    detection.info = 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.

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

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

\[ \alpha = 1 \] 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.

**epsilon0**

a positive number. Useful only when transfer.source.id = "auto". The threshold to determine transferability will be set as \((1 + \epsilon_0) \times (\text{validation or cross-validation loss of target data})\). Default = 0.01. For details, refer to Algorithm 3 in Tian, Y. and Feng, Y., 2021.

**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.detection**

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

**...**

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".

**epsilon0**

the threshold to determine transferability will be set as \((1 + \epsilon_0) \times (\text{validation or cross-validation loss of target data})\). Only available when transfer.source.id = "auto".

**threshold**

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

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

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

# study the linear model
D.training <- models("gaussian", type = "all", K = 2, p = 500, Ka = 1)
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", p = 500)
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", p = 200)
detection.poisson <- source_detection(D.training$target, D.training$source,
family = "poisson", cores = 2)
detection.poisson$transferable.source.id
Index

* datasets
  micromass, 6

cv.glmnet, 5, 14

ggplot, 9
glmnet, 5, 14
glmtrans, 2, 8–11, 14

micromass, 6
models, 5, 7, 14

plot.glmtrans, 5, 8, 14
plot.glmtrans_source_detection
  (plot.glmtrans), 8
predict.glmtrans, 5, 9, 14
print.glmtrans, 11

source_detection, 5, 9, 12