Package: gglasso (via r-universe)

September 8, 2024

Title Group Lasso Penalized Learning Using a Unified BMD Algorithm

Version 1.4

Date 2017-9-12

Author Yi Yang <yi.yang6@mcgill.ca>, Hui Zou <hzou@stat.umn.edu>

Description A unified algorithm, blockwise-majorization-descent (BMD), for efficiently computing the solution paths of the group-lasso penalized least squares, logistic regression, Huberized SVM and squared SVM. The package is an implementation of Yang, Y. and Zou, H. (2015) DOI: <doi:10.1007/s11222-014-9498-5>.

License GPL-2

Imports methods

URL https://github.com/emeryyi/gglasso

BugReports https://github.com/emeryyi/gglasso/issues

Date/Publication 2017-9-12 11:11:11

RoxygenNote 7.0.2

Suggests testthat, knitr, rmarkdown

Encoding UTF-8

LazyData TRUE

VignetteBuilder knitr

Repository https://archer-yang-lab.r-universe.dev

RemoteUrl https://github.com/archer-yang-lab/gglasso

RemoteRef HEAD

RemoteSha 7d98ba6ffb1dd4aa6a72c0eb9bd002f79530609a

Contents

`bardet` *Simplified gene expression data from Scheetz et al. (2006)*

Description

Gene expression data (20 genes for 120 samples) from the microarray experiments of mammalian eye tissue samples of Scheetz et al. (2006).

Usage

```
bardet
```

Format

An object of class `list` of length 2.

Details

This data set contains 120 samples with 100 predictors (expanded from 20 genes using 5 basis B-splines, as described in Yang, Y. and Zou, H. (2015)).

Value

A list with the following elements:

`x`	a [120 x 100] matrix (expanded from a [120 x 20] matrix) giving the expression levels of 20 filtered genes for the 120 samples. Each row corresponds to a subject, each 5 consecutive columns to a grouped gene.
`y`	a numeric vector of length 120 giving expression level of gene TRIM32, which causes Bardet-Biedl syndrome.

References

Scheetz, T., Kim, K., Swiderski, R., Philp, A., Braun, T., Knudtson, K., Dorrance, A., DiBona, G., Huang, J., Casavant, T. et al. (2006), "Regulation of gene expression in the mammalian eye and its relevance to eye disease", *Proceedings of the National Academy of Sciences* **103**(39), 14429-14434.

Huang, J., S. Ma, and C.-H. Zhang (2008). "Adaptive Lasso for sparse high-dimensional regression models". *Statistica Sinica* 18, 1603-1618.

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(bardet)

# how many samples and how many predictors ?
dim(bardet$x)

# repsonse y
bardet$y
```

`coef.cv.gglasso` *get coefficients or make coefficient predictions from a "cv.gglasso" object.*

Description

This function gets coefficients or makes coefficient predictions from a cross-validated `gglasso` model, using the stored `"gglasso.fit"` object, and the optimal value chosen for `lambda`.

Usage

```
## S3 method for class 'cv.gglasso'
coef(object, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

`object`	fitted `cv.gglasso` object.

s	value(s) of the penalty parameter lambda at which predictions are required. Default is the value s="lambda.1se" stored on the CV object, it is the largest value of lambda such that error is within 1 standard error of the minimum. Alternatively s="lambda.min" can be used, it is the optimal value of lambda that gives minimum cross validation error cvm. If s is numeric, it is taken as the value(s) of lambda to be used.
...	not used. Other arguments to predict.

Details

This function makes it easier to use the results of cross-validation to get coefficients or make coefficient predictions.

Value

The coefficients at the requested values for lambda.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

Friedman, J., Hastie, T., and Tibshirani, R. (2010), "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software, 33, 1.*
http://www.jstatsoft.org/v33/i01/

See Also

cv.gglasso, and predict.cv.gglasso methods.

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# define group index
group <- rep(1:20,each=5)

# 5-fold cross validation using group lasso
# penalized logisitic regression
cv <- cv.gglasso(x=colon$x, y=colon$y, group=group, loss="logit",
```

```
pred.loss="misclass", lambda.factor=0.05, nfolds=5)

# the coefficients at lambda = lambda.1se
pre = coef(cv$gglasso.fit, s = cv$lambda.1se)
```

`coef.gglasso` *get coefficients or make coefficient predictions from an "gglasso" object.*

Description

Computes the coefficients at the requested values for `lambda` from a fitted `gglasso` object.

Usage

```
## S3 method for class 'gglasso'
coef(object, s = NULL, ...)
```

Arguments

`object`	fitted `gglasso` model object.
`s`	value(s) of the penalty parameter `lambda` at which predictions are required. Default is the entire sequence used to create the model.
`...`	not used. Other arguments to predict.

Details

`s` is the new vector at which predictions are requested. If `s` is not in the lambda sequence used for fitting the model, the `coef` function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right `lambda` indices.

Value

The coefficients at the requested values for `lambda`.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

See Also

predict.gglasso method

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# define group index
group <- rep(1:20,each=5)

# fit group lasso
m1 <- gglasso(x=colon$x,y=colon$y,group=group,loss="logit")

# the coefficients at lambda = 0.01 and 0.02
coef(m1,s=c(0.01,0.02))
```

---

colon *Simplified gene expression data from Alon et al. (1999)*

Description

Gene expression data (20 genes for 62 samples) from the microarray experiments of colon tissue samples of Alon et al. (1999).

Usage

```
colon
```

Format

An object of class `list` of length 2.

Details

This data set contains 62 samples with 100 predictors (expanded from 20 genes using 5 basis B-splines, as described in Yang, Y. and Zou, H. (2015)): 40 tumor tissues, coded 1 and 22 normal tissues, coded -1.

Value

A list with the following elements:

x	a [62 x 100] matrix (expanded from a [62 x 20] matrix) giving the expression levels of 20 genes for the 62 colon tissue samples. Each row corresponds to a patient, each 5 consecutive columns to a grouped gene.
y	a numeric vector of length 62 giving the type of tissue sample (tumor or normal).

Source

The data are described in Alon et al. (1999) and can be freely downloaded from http://microarray.princeton.edu/oncology/affydata/index.html.

References

Alon, U. and Barkai, N. and Notterman, D.A. and Gish, K. and Ybarra, S. and Mack, D. and Levine, A.J. (1999). "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays", *Proc. Natl. Acad. Sci. USA*, **96**(12), 6745–6750.

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# how many samples and how many predictors ?
dim(colon$x)

# how many samples of class -1 and 1 respectively ?
sum(colon$y==-1)
sum(colon$y==1)
```

---

cv.gglasso *Cross-validation for gglasso*

Description

Does k-fold cross-validation for gglasso, produces a plot, and returns a value for lambda. This function is modified based on the cv function from the glmnet package.

Usage

```
cv.gglasso(
  x,
  y,
  group,
  lambda = NULL,
  pred.loss = c("misclass", "loss", "L1", "L2"),
  nfolds = 5,
  foldid,
  delta,
  ...
)
```

Arguments

x	matrix of predictors, of dimension $n \times p$; each row is an observation vector.
y	response variable. This argument should be quantitative for regression (least squares), and a two-level factor for classification (logistic model, huberized SVM, squared SVM).
group	a vector of consecutive integers describing the grouping of the coefficients (see example below).
lambda	optional user-supplied lambda sequence; default is NULL, and gglasso chooses its own sequence.
pred.loss	loss to use for cross-validation error. Valid options are: • "loss" for classification, margin based loss function. • "misclass" for classification, it gives misclassification error. • "L1" for regression, mean square error used by least squares regression loss="ls", it measure the deviation from the fitted mean to the response. • "L2" for regression, mean absolute error used by least squares regression loss="ls", it measure the deviation from the fitted mean to the response. Default is "loss".
nfolds	number of folds - default is 5. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds=3.
foldid	an optional vector of values between 1 and nfold identifying what fold each observation is in. If supplied, nfold can be missing.
delta	parameter δ only used in huberized SVM for computing log-likelihood on validation set, only available with pred.loss = "loss", loss = "hsvm".
...	other arguments that can be passed to gglasso.

Details

The function runs gglasso nfolds+1 times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The average error and standard deviation over the folds are computed.

Value

an object of class cv.gglasso is returned, which is a list with the ingredients of the cross-validation fit.

lambda	the values of lambda used in the fits.
cvm	the mean cross-validated error - a vector of length length(lambda).
cvsd	estimate of standard error of cvm.
cvupper	upper curve = cvm+cvsd.
cvlower	lower curve = cvm-cvsd.
name	a text string indicating type of measure (for plotting purposes).
gglasso.fit	a fitted gglasso object for the full data.
lambda.min	The optimal value of lambda that gives minimum cross validation error cvm.
lambda.1se	The largest value of lambda such that error is within 1 standard error of the minimum.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

See Also

gglasso, plot.cv.gglasso, predict.cv.gglasso, and coef.cv.gglasso methods.

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(bardet)

# define group index
group <- rep(1:20,each=5)

# 5-fold cross validation using group lasso
# penalized logisitic regression
cv <- cv.gglasso(x=bardet$x, y=bardet$y, group=group, loss="ls",
pred.loss="L2", lambda.factor=0.05, nfolds=5)
```

`gglasso` *Fits the regularization paths for group-lasso penalized learning problems*

Description

Fits regularization paths for group-lasso penalized learning problems at a sequence of regularization parameters lambda.

Usage

```
gglasso(
  x,
  y,
  group = NULL,
  loss = c("ls", "logit", "sqsvm", "hsvm", "wls"),
  nlambda = 100,
  lambda.factor = ifelse(nobs < nvars, 0.05, 0.001),
  lambda = NULL,
  pf = sqrt(bs),
  weight = NULL,
  dfmax = as.integer(max(group)) + 1,
  pmax = min(dfmax * 1.2, as.integer(max(group))),
  eps = 1e-08,
  maxit = 3e+08,
  delta,
  intercept = TRUE
)
```

Arguments

`x`	matrix of predictors, of dimension $n \times p$; each row is an observation vector.
`y`	response variable. This argument should be quantitative for regression (least squares), and a two-level factor for classification (logistic model, huberized SVM, squared SVM).
`group`	a vector of consecutive integers describing the grouping of the coefficients (see example below).
`loss`	a character string specifying the loss function to use, valid options are: • `"ls"` least squares loss (regression), • `"logit"` logistic loss (classification). • `"hsvm"` Huberized squared hinge loss (classification), • `"sqsvm"` Squared hinge loss (classification), Default is `"ls"`.
`nlambda`	the number of `lambda` values - default is 100.

`lambda.factor` the factor for getting the minimal lambda in `lambda` sequence, where `min(lambda) = lambda.factor * max(lambda)`. `max(lambda)` is the smallest value of `lambda` for which all coefficients are zero. The default depends on the relationship between n (the number of rows in the matrix of predictors) and p (the number of predictors). If $n >= p$, the default is `0.001`, close to zero. If $n < p$, the default is `0.05`. A very small value of `lambda.factor` will lead to a saturated fit. It takes no effect if there is user-defined `lambda` sequence.

`lambda` a user supplied `lambda` sequence. Typically, by leaving this option unspecified users can have the program compute its own `lambda` sequence based on `nlambda` and `lambda.factor`. Supplying a value of `lambda` overrides this. It is better to supply a decreasing sequence of `lambda` values than a single (small) value, if not, the program will sort user-defined `lambda` sequence in decreasing order automatically.

`pf` penalty factor, a vector in length of bn (bn is the total number of groups). Separate penalty weights can be applied to each group of βs to allow differential shrinkage. Can be 0 for some groups, which implies no shrinkage, and results in that group always being included in the model. Default value for each entry is the square-root of the corresponding size of each group.

`weight` a nxn observation weight matrix in the where n is the number of observations. Only used if `loss='wls'` is specified. Note that cross-validation is NOT IMPLEMENTED for `loss='wls'`.

`dfmax` limit the maximum number of groups in the model. Useful for very large `bs` (group size), if a partial path is desired. Default is `bs+1`.

`pmax` limit the maximum number of groups ever to be nonzero. For example once a group enters the model, no matter how many times it exits or re-enters model through the path, it will be counted only once. Default is `min(dfmax*1.2,bs)`.

`eps` convergence termination tolerance. Defaults value is `1e-8`.

`maxit` maximum number of outer-loop iterations allowed at fixed lambda value. Default is 3e8. If models do not converge, consider increasing `maxit`.

`delta` the parameter δ in `"hsvm"` (Huberized squared hinge loss). Default is 1.

`intercept` Whether to include intercept in the model. Default is TRUE.

Details

Note that the objective function for `"ls"` least squares is

$$RSS/(2 * n) + lambda * penalty;$$

for `"hsvm"` Huberized squared hinge loss, `"sqsvm"` Squared hinge loss and `"logit"` logistic regression, the objective function is

$$-loglik/n + lambda * penalty.$$

Users can also tweak the penalty by choosing different penalty factor.

For computing speed reason, if models are not converging or running slow, consider increasing `eps`, decreasing `nlambda`, or increasing `lambda.factor` before increasing `maxit`.

Value

An object with S3 class gglasso.

call	the call that produced this object
b0	intercept sequence of length length(lambda)
beta	a p*length(lambda) matrix of coefficients.
df	the number of nonzero groups for each value of lambda.
dim	dimension of coefficient matrix (ices)
lambda	the actual sequence of lambda values used
npasses	total number of iterations (the most inner loop) summed over all lambda values
jerr	error flag, for warnings and errors, 0 if no error.
group	a vector of consecutive integers describing the grouping of the coefficients.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

See Also

plot.gglasso

Examples

```
# load gglasso library
library(gglasso)

# load bardet data set
data(bardet)

# define group index
group1 <- rep(1:20,each=5)

# fit group lasso penalized least squares
m1 <- gglasso(x=bardet$x,y=bardet$y,group=group1,loss="ls")

# load colon data set
data(colon)

# define group index
group2 <- rep(1:20,each=5)
```

```
# fit group lasso penalized logistic regression
m2 <- gglasso(x=colon$x,y=colon$y,group=group2,loss="logit")
```

`plot.cv.gglasso` *plot the cross-validation curve produced by cv.gglasso*

Description

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the `lambda` values used. This function is modified based on the `plot.cv` function from the `glmnet` package.

Usage

```
## S3 method for class 'cv.gglasso'
plot(x, sign.lambda = 1, ...)
```

Arguments

`x`	fitted `cv.gglasso` object
`sign.lambda`	either plot against `log(lambda)` (default) or its negative if `sign.lambda=-1`.
`...`	other graphical parameters to plot

Details

A plot is produced.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

Friedman, J., Hastie, T., and Tibshirani, R. (2010), "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, 33, 1.
http://www.jstatsoft.org/v33/i01/

See Also

`cv.gglasso`.

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# define group index
group <- rep(1:20,each=5)

# 5-fold cross validation using group lasso
# penalized logisitic regression
cv <- cv.gglasso(x=colon$x, y=colon$y, group=group, loss="logit",
pred.loss="misclass", lambda.factor=0.05, nfolds=5)

# make a CV plot
plot(cv)
```

`plot.gglasso` *Plot solution paths from a "gglasso" object*

Description

Produces a coefficient profile plot of the coefficient paths for a fitted `gglasso` object.

Usage

```
## S3 method for class 'gglasso'
plot(x, group = FALSE, log.l = TRUE, ...)
```

Arguments

`x`	fitted `gglasso` model
`group`	what is on the Y-axis. Plot the norm of each group if `TRUE`. Plot each coefficient if `FALSE`.
`log.l`	what is on the X-axis. Plot against the log-lambda sequence if `TRUE`. Plot against the lambda sequence if `FALSE`.
`...`	other graphical parameters to plot

Details

A coefficient profile plot is produced.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(bardet)

# define group index
group <- rep(1:20,each=5)

# fit group lasso
m1 <- gglasso(x=bardet$x,y=bardet$y,group=group,loss="ls")

# make plots
par(mfrow=c(1,3))
plot(m1) # plots the coefficients against the log-lambda sequence
plot(m1,group=TRUE) # plots group norm against the log-lambda sequence
plot(m1,log.l=FALSE) # plots against the lambda sequence
```

---

predict.cv.gglasso *make predictions from a "cv.gglasso" object.*

Description

This function makes predictions from a gglasso model, using the stored "gglasso.fit" object, and the optimal value chosen for lambda.

Usage

```
## S3 method for class 'cv.gglasso'
predict(object, newx, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

object	fitted cv.gglasso object.
newx	matrix of new values for x at which predictions are to be made. Must be a matrix. See documentation for predict.gglasso.
s	value(s) of the penalty parameter lambda at which predictions are required. Default is the value s="lambda.1se" stored on the CV object. Alternatively s="lambda.min" can be used. If s is numeric, it is taken as the value(s) of lambda to be used.
...	not used. Other arguments to predict.

Details

This function makes it easier to use the results of cross-validation to make a prediction.

Value

The returned object depends on the ... argument which is passed on to the `predict` method for `gglasso` objects.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

See Also

`cv.gglasso`, and `coef.cv.gglasso` methods.

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# define group index
group <- rep(1:20,each=5)

# 5-fold cross validation using group lasso
# penalized logisitic regression
cv <- cv.gglasso(x=colon$x, y=colon$y, group=group, loss="logit",
pred.loss="misclass", lambda.factor=0.05, nfolds=5)

# the coefficients at lambda = lambda.min, newx = x[1,]
pre = predict(cv$gglasso.fit, newx = colon$x[1:10,],
s = cv$lambda.min, type = "class")
```

`predict.gglasso` *make predictions from a "gglasso" object.*

Description

Similar to other predict methods, this functions predicts fitted values and class labels from a fitted `gglasso` object.

Usage

```
## S3 method for class 'gglasso'
predict(object, newx, s = NULL, type = c("class", "link"), ...)
```

Arguments

`object`	fitted `gglasso` model object.
`newx`	matrix of new values for `x` at which predictions are to be made. Must be a matrix.
`s`	value(s) of the penalty parameter `lambda` at which predictions are required. Default is the entire sequence used to create the model.
`type`	type of prediction required: • Type `"link"`, for regression it returns the fitted response; for classification it gives the linear predictors. • Type `"class"`, only valid for classification, it produces the predicted class label corresponding to the maximum probability.
`...`	Not used. Other arguments to predict.

Details

`s` is the new vector at which predictions are requested. If `s` is not in the lambda sequence used for fitting the model, the `predict` function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right `lambda` indices.

Value

The object returned depends on type.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

See Also

coef method

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# define group index
group <- rep(1:20,each=5)

# fit group lasso
m1 <- gglasso(x=colon$x,y=colon$y,group=group,loss="logit")

# predicted class label at x[10,]
print(predict(m1,type="class",newx=colon$x[10,]))

# predicted linear predictors at x[1:5,]
print(predict(m1,type="link",newx=colon$x[1:5,]))
```

---

print.gglasso *print a gglasso object*

Description

Print the nonzero group counts at each lambda along the gglasso path.

Usage

```
## S3 method for class 'gglasso'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

x	fitted gglasso object
digits	significant digits in printout
...	additional print arguments

Details

Print the information about the nonzero group counts at each lambda step in the gglasso object. The result is a two-column matrix with columns Df and Lambda. The Df column is the number of the groups that have nonzero within-group coefficients, the Lambda column is the the corresponding lambda.

Value

a two-column matrix, the first columns is the number of nonzero group counts and the second column is Lambda.

Author(s)

Yi Yang and Hui Zou
Maintainer: Yi Yang <yi.yang6@mcgill.ca>

References

Yang, Y. and Zou, H. (2015), "A Fast Unified Algorithm for Computing Group-Lasso Penalized Learning Problems," *Statistics and Computing*. 25(6), 1129-1141.
BugReport: https://github.com/emeryyi/gglasso

Examples

```
# load gglasso library
library(gglasso)

# load data set
data(colon)

# define group index
group <- rep(1:20,each=5)

# fit group lasso
m1 <- gglasso(x=colon$x,y=colon$y,group=group,loss="logit")

# print out results
print(m1)
```

Index