Package 'glinternet'

May 8, 2026

Type Package

Title Learning Interactions via Hierarchical Group-Lasso Regularization

Version 1.0.12

Date 2021-09-01

Author Michael Lim, Trevor Hastie

Maintainer Michael Lim <michael626@gmail.com>

Description Group-Lasso INTERaction-NET. Fits linear pairwise-interaction models that satisfy strong hierarchy: if an interaction coefficient is estimated to be nonzero, then its two associated main effects also have nonzero estimated coefficients. Accommodates categorical variables (factors) with arbitrary numbers of levels, continuous variables, and combinations thereof. Implements the machinery described in the paper ``Learning interactions via hierarchical group-lasso regularization" (JCGS 2015, Volume 24, Issue 3). Michael Lim & Trevor Hastie (2015) <DOI:10.1080/10618600.2014.938812>.

License GPL-2

URL http://web.stanford.edu/~hastie/Papers/glinternet_jcgs.pdf

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-09-03 04:50:42 UTC

Contents

`coef.glinternet` *Return main effect and interaction coefficients.*

Description

Returns the actual main effect and interaction coefficients that satisfy the sum constraints in a linear interaction model. See the paper below for details.

Usage

```
## S3 method for class 'glinternet'
coef(object, lambdaIndex = NULL, ...)
```

Arguments

`object`	Fitted `"glinternet"` model object.
`lambdaIndex`	Index of `lambda` value at which to extract coefficients. If NULL, return the coefficients for all values of `lambda` in the path.
`...`	Not used.

Details

Returns the actual main effect and interaction coefficients. These satisfy the sum constraints in the original linear interaction model.

Value

A list of length `lambda` if `lambdaIndex` is not provided. Otherwise, of length `lambdaIndex`. Each component (for each lambda) is itself a list, with components

`mainEffects`	A list with components `cat` and `cont`, each an index vector of the categorical and continuous (resp) main-effect variables. Just as in `activeSet`, the indexing is separate for each type of variable. See ?glinternet for details.
`mainEffectsCoef`	List of coefficients for the main effects in `mainEffects`, also with names `cat` and `cont`
`interactions`	List of interactions, with components `contcont`, `catcont` and `catcat`, each 2-column matrices of variable indices.
`interactionsCoef`	List of interaction coefficients for `interactions`, also with names `contcont`, `catcont` and `catcat`. For categorical-categorical interactions, each is provided as a L1 x L2 matrix.

Author(s)

Michael Lim and Trevor Hastie
Maintainer: Michael Lim <michael626@gmail.com>

References

"Learning interactions via hierarchical group-lasso regularization"

See Also

`glinternet.cv`, `predict.glinternet`, `predict.glinternet.cv`, `plot.glinternet.cv`, `glinternet`

Examples

```
Y = rnorm(100)
X = matrix(rnorm(100*10), nrow=100)
numLevels = rep(1, 10)
fit = glinternet(X, Y, numLevels)
coeffs = coef(fit)
```

`glinternet` *Fit a linear interaction model with group-lasso regularization that enforces strong hierarchy in the estimated coefficients*

Description

The regularization path is computed along a grid of values for the regularization parameter lambda. Can deal with categorical variables with arbitrary numbers of levels, continuous variables, and combinations of the two. Accommodates squared error loss and logistic loss.

The multicore option requires that the package be compiled with OpenMP support. Examples of compilers that qualify include gcc (>= 4.2) and icc. We also recommend a higher level of optimization, such as -O3 in gcc.

Usage

```
glinternet(X, Y, numLevels, lambda = NULL, nLambda = 50, lambdaMinRatio = 0.01,
 interactionCandidates=NULL, interactionPairs=NULL, screenLimit = NULL, numToFind = NULL,
family = c("gaussian","binomial"), tol = 1e-05, maxIter=5000, verbose=FALSE,
numCores = 1)
```

Arguments

`X`	Matrix of features or predictors with dimension nobs x nvars; each row is an observation vector. Categorical variables must be coded as 0, 1, 2, ...
`Y`	Target variable of length nobs. Continuous for squared error loss, 0-1 for logistic loss.
`numLevels`	Number of levels for each variable, of length nvars. Set to 1 for continuous variables.
`lambda`	A user supplied `lambda` sequence. Typical usage is to have the program compute its own `lambda` sequence based on `nLambda` and `lambdaMinRatio`. Supplying a value of `lambda` overrides this.

`nLambda` The number of `lambda` values. Default is 50.

`lambdaMinRatio` Smallest value for `lambda`, as a fraction of `lambdaMax`, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default is `0.01`.

`interactionCandidates`
An optional vector of variable indices. This will force the algorithm to only consider interactions between `interactionCandidates` and all other variables.

`interactionPairs`
An optional nx2 matrix of variable indices. This will force the algorithm to only consider the interaction pairs defined by this matrix. For example, matrix(c(1,2,1,5), ncol=2, byrow=TRUE) restricts the model to two interaction pairs: one between variables 1 and 2, and another between 1 and 5.

`screenLimit` If not null (the default), limits the size of the interaction search space to `screenLimit` x nvars by only considering interactions with the best `screenLimit` candidate main effects at each point along the regularization path. Set this accordingly for large problems or if there are memory limitations.

`numToFind` Stops the program after `numToFind` interaction pairs are found. Default is null - fit all values of `lambda`.

`family` A character string describing the target variable: "gaussian" for continuous (the default), "binomial" for logistic.

`tol` Convergence tolerance in the adaptive FISTA algorithm.

`maxIter` Maximum number of iterations in adaptive FISTA. Default 5000.

`verbose` Prints progress. False by default.

`numCores` Number of threads to run. For this to work, the package must be installed with OpenMP enabled. Default is 1 thread.

Details

The sequence of models implied by `lambda` is fit by FISTA (fast iterative soft thresholding) with adaptive step size and adaptive momentum restart. The continuous features are standardized to have unit norm and mean zero before computing the `lambda` sequence and fitting. The returned coefficients are unstandardized. Categorical variables are not standardized.

Value

An object of class `glinternet` with the components

`call` The user function call.

`fitted` The fitted values, with dimension nobs x `nLambda`. If `numToFind` is specified, the program is likely to stop before all `nLambda` models have been fit.

`lambda` The actual `lambda` sequence used.

`objValue` Objective values for each lambda.

`activeSet` A list (of length `nLambda`) of the variables found. Internally, the categorical and continuous variables are separated into two groups, and each has their own indexing system (1-based). For example, the categorical-continuous interaction `c(i, j)` refers to the interaction between the `i`-th categorical variable with the `j`-th continuous variable.

betahat	List (of length lambda) of coefficients for the variables in activeSet. The first component is the intercept. Subsequent entries correspond to the variables in activeSet. For example, if the first variable in activeSet is a 3-level categorical variable, then components 2-4 of betahat are the coefficients for this variable.
numLevels	The number of levels for each variable.
family	The target variable type.

Author(s)

Michael Lim and Trevor Hastie
Maintainer: Michael Lim <michael626@gmail.com>

References

Michael Lim and Trevor Hastie (2013)*Learning interactions via hierarchical group-lasso regularization*, https://arxiv.org/abs/1308.2719

See Also

glinternet.cv, predict.glinternet, predict.glinternet.cv, plot.glinternet.cv, coef.glinternet

Examples

```
# gaussian response, continuous features
Y = rnorm(100)
X = matrix(rnorm(100*10), nrow=100)
numLevels = rep(1, 10)
fit = glinternet(X, Y, numLevels)

#binary response, continuous features
Y = rbinom(100, 1, 0.5)
fit = glinternet(X, Y, numLevels, family="binomial")

#binary response, categorical variables
X = matrix(sample(0:2, 100*10, replace=TRUE), nrow=100)
numLevels = rep(3, 10)
fit = glinternet(X, Y, numLevels, family="binomial")
```

---

glinternet.cv *Cross-validation for glinternet*

Description

Does k-fold cross validation for glinternet and returns a value of lambda.

Usage

```
glinternet.cv(X, Y, numLevels, nFolds = 10, lambda=NULL, nLambda=50,
lambdaMinRatio=0.01, interactionCandidates=NULL, interactionPairs=NULL,
screenLimit=NULL, family=c("gaussian", "binomial"), tol=1e-5, maxIter=5000,
verbose=FALSE, numCores=1)
```

Arguments

X	X matrix as in glinternet.
Y	Target Y as in glinternet.
numLevels	Number of levels numLevels as in glinternet.
nFolds	Number of folds - default is 10.
lambda	lambda as in glinternet.
nLambda	nLambda as in glinternet.
lambdaMinRatio	lambdaMinRatio as in glinternet.
interactionCandidates	interactionCandidates as in glinternet.
interactionPairs	interactionPairs as in glinternet.
screenLimit	screenLimit as in glinternet.
family	family as in glinternet.
tol	tol as in glinternet.
maxIter	maxIter as in glinternet.
verbose	verbose as in glinternet.
numCores	numCores as in glinternet.

Details

The lambda sequence is computed using all the data. nFolds models are fit, each time with one of the folds omitted. The error is accumulated, and the average error and standard deviation over the folds is computed. The lambda value that minimizes the average error is returned, and a model with this lambda is fit to the full data set.

Value

An object of class glinternet.cv with the components

call	The user function call.
glinternetFit	Glinternet object fitted on the full data using a lambda sequence that terminates at lambdaHat.
fitted	Vector for fitted values (same length as Y). This is from the model fitted at lambdaHat.
activeSet	activeSet is a list variables found for the model fitted with lambdaHat.
betahat	Unstandardized coefficients for the variables in activeSet.

lambda	The actual sequence of lambda values used for the cross validation.
lambdaHat	The value of lambda that minimizes the cv error curve.
lambdaHat1Std	The largest value of lambda that produces a cv error that is within 1 standard deviation of the minimum cv error. This will always be at least as large as lambdaHat.
cvErr	The vector of cross validation errors. Same length as lambda.
cvErrStd	Standard deviation for cv errors across the nFolds folds.
family	The response type.
numLevels	Input number of levels for each variable.
nFolds	The number of folds used.

Author(s)

Michael Lim and Trevor Hastie
Maintainer: Michael Lim <michael626@gmail.com>

See Also

glinternet, predict.glinternet, predict.glinternet.cv, plot.glinternet.cv

Examples

```
Y = rnorm(100)
numLevels = sample(1:5, 10, replace=TRUE)
X = sapply(numLevels, function(x) if (x==1)
rnorm(100) else sample(0:(x-1), 100, replace=TRUE))
fit = glinternet.cv(X, Y, numLevels, nFolds=3)
```

---

plot.glinternet.cv *Plot CV error from* glinternetCV *object.*

Description

Plots the cross validation error against the lambda index. Uses ggplot2 if found on the user's system.

Usage

```
## S3 method for class 'glinternet.cv'
plot(x, ...)
```

Arguments

x	"glinternetCV" object.
...	Not used.

Value

A plot of CV error.

Author(s)

Michael Lim and Trevor Hastie
Maintainer: Michael Lim <michael626@gmail.com>

See Also

glinternet, glinternet.cv, predict.glinternet.cv, predict.glinternet

---

predict.glinternet *Make predictions from a "glinternet" object.*

Description

Similar to other predict methods, this function returns fitted values on the response scale. Also gives the option to return the link function.

Usage

```
## S3 method for class 'glinternet'
predict(object, X, type = c("response", "link"), lambda=NULL, ...)
```

Arguments

object	Fitted "glinternet" model object.
X	Matrix of new values for which to make predictions. Must have the same number of variables as during training the model, and all the variables must have the same number of levels.
type	Return fitted Y values or the link function.
lambda	User input lambda sequence. Must be subset of the lambdas used in fitting. If NULL (the default), predict at all the lambdas used during fitting.
...	Not used. Other arguments to predict.

Details

If lambda is not specified, makes predictions at all the fitted lambda values. Users may provide their own lambda sequence, but this must be a subset of the values used to fit the models.

Value

A matrix of predicted values, with columns corresponding to each fitted model.

Author(s)

Michael Lim and Trevor Hastie
Maintainer: Michael Lim <michael626@gmail.com>

See Also

glinternet, glinternet.cv, predict.glinternet.cv, plot.glinternet.cv

Examples

```
Y = rnorm(100)
numLevels = sample(1:5, 10, replace=TRUE)
X = sapply(numLevels, function(x) if (x==1)
rnorm(100) else sample(0:(x-1), 100, replace=TRUE))
fit = glinternet(X, Y, numLevels)
max(abs(fit$fitted - predict(fit, X)))
```

---

predict.glinternet.cv *Make predictions from a "glinternetCV" object.*

Description

Similar to other predict methods, this function returns fitted values on the response scale. Also gives the option to return the link function.

Usage

```
## S3 method for class 'glinternet.cv'
predict(object, X, type = c("response", "link"),
lambdaType=c("lambdaHat", "lambdaHat1Std"), ...)
```

Arguments

object	"glinternetCV" object.
X	Matrix of new values for which to make predictions. Must have the same number of variables as during training the model, and all the variables must have the same number of levels.
type	Return fitted Y values or the link function.
lambdaType	lambdaHat corresponds to the lambda value that gives the minimum CV error. lambdaHat1Std picks the largest value of lambda for which the CV error is within 1 standard error of the minimum.
...	Not used.

Details

Makes predictions using the model fitted at the appropriate lambda value.

Value

A vector of predicted values.

Author(s)

Michael Lim and Trevor Hastie
Maintainer: Michael Lim <michael626@gmail.com>

See Also

glinternet, glinternet.cv, predict.glinternet

Examples

```
Y = rnorm(100)
numLevels = sample(1:5, 10, replace=TRUE)
X = sapply(numLevels, function(x) if (x==1)
rnorm(100) else sample(0:(x-1), 100, replace=TRUE))
fit = glinternet.cv(X, Y, numLevels, nFolds=3)
max(abs(fit$fitted - predict(fit, X)))
```

Index