Package ‘gomp’

July 22, 2025

Type Package

Title The gamma-OMP Feature Selection Algorithm

Version 1.0

Date 2025-01-11

Author Michail Tsagris [aut, cre]

Maintainer Michail Tsagris <mtsagris@uoc.gr>

Depends R (>= 4.0)

Imports doParallel, foreach, Hmisc, MASS, nnet, ordinal, parallel, quantreg, Rfast, Rfast2, stats, survival

Suggests dcorVS

Description The gamma-Orthogonal Matching Pursuit (gamma-OMP) is a recently suggested modification of the OMP feature selection algorithm for a wide range of response variables. The package offers many alternative regression models, such linear, robust, survival, multivariate etc., including k-fold cross-validation. References: Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2018). ``Efficient feature selection on gene expression data: Which algorithm to use?'' BioRxiv. <doi:10.1101/431734>. Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2022). ``The gamma-OMP algorithm for feature selection with application to gene expression data''. IEEE/ACM Transactions on Computational Biology and Bioinformatics 19(2): 1214--1224. <doi:10.1109/TCBB.2020.3029952>.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-20 16:02:00 UTC

Contents

gomp-package *The gamma-OMP Feature Selection Algorithm*

Description

The gamma-Orthogonal Matching Pursuit (gamma-OMP) is a recently suggested modification of the OMP feature selection algorithm for a wide range of response variables. The package offers many alternative regression models, such linear, robust, survival, multivariate etc., including k-fold cross-validation. References: Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2018). "Efficient feature selection on gene expression data: Which algorithm to use?" BioRxiv. <doi:10.1101/431734>. Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2022). "The gamma-OMP algorithm for feature selection with application to gene expression data". IEEE/ACM Transactions on Computational Biology and Bioinformatics 19(2): 1214–1224.

Details

Package:	gomp
Type:	Package
Version:	1.0
Date:	2025-01-11
License:	GPL-2

Maintainers

Michail Tsagris <`mtsagris@uoc.gr`>.

Author(s)

Michail Tsagris <`mtsagris@uoc.gr`>.

References

Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2018). Efficient feature selection on gene expression data: Which algorithm to use? BioRxiv.

Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2022). The γ-OMP algorithm for feature selection with application to gene expression data". IEEE/ACM Transactions on Computational Biology and Bioinformatics 19(2): 1214–1224.

Alharbi N. (2024). Variable selection with time-to-event data: Cox or Weibull regression? Communications in Statistics: Case Studies, Data Analysis and Applications (accepted for publication).

Bootstrap bias correction for the performance of the cross-validation procedure

Bootstrap bias correction for the performance of the cross-validation procedure

Description

Bootstrap bias correction for the performance of the cross-validation procedure.

Usage

```
bbc(predictions, y, metric = "auc.gomp", conf = 0.95, B = 1000)
```

Arguments

`predictions` A matrix with the predictived values.

`y` A vector with the response variable, survival object, factor (ordered or unordered) or a numerical vector.

`metric` The possible values are:

a) Binary response: "auc.gomp" (area under the curve), "fscore.gomp" (F-score), "prec.gomp" (precision), "euclid_sens.spec.gomp" (Euclidean distance of sensitivity and specificity), "spec.gomp" (specificity), "sens.gomp" (sensitivity), "acc.gomp" (accuracy, proportion of correct classification).

b) Multinomial response: "acc_multinom.gomp" (accuracy, proportion of correct classification).

c) Ordinal response: "ord_mae.gomp" (mean absolute error).

d) Continuous response: "mae.gomp" (MAE with continuous response), "mse.gomp" (mean squared error), "pve.gomp" (percentage of variance explained).

e) Survival response "ci.gomp" (concordance index for Cox regression), "ciwr.gomp" (concordance index for Weibull regression).

g) Count response "poisdev.gomp".

h) Binomial response "binomdev.gomp" (deviance of binomial regression).

The "nbdev.gomp" (negative binomial deviance) is missing. For more information on these see `cv.gomp`. **Note** that they come with "".

`conf` A number between 0 and 1, the confidence level.

`B` The number of bootstrap replicates. The default number is 1000.

Details

Upon completion of the cross-validation, the predicted values produced by all predictive models across all folds is collected in a matrix P of dimensions $n \times M$, where n is the number of samples and M the number of trained models or configurations. Sampled with replacement a fraction of rows (predictions) from P are denoted as the in-sample values. On average, the newly created set will be comprised by 63.2% of the original individuals (The probability of sampling, with replacement, a sample of n numbers from a set of n numbers is $1 - \left(1 - \frac{1}{n}\right)^n \simeq 1 - \frac{1}{e} = 0.632$), whereas the rest 36.8% will be random copies of them. The non re-sampled rows are denoted as out-of-sample values. The performance of each model in the in-sample rows is calculated and the model (or configuration) with the optimal performance is selected, followed by the calculation of performance in the out-of-sample values. This process is repeated B times and the average performance is returned.

Note, that the only computational overhead is with the repetitive re-sampling and calculation of the predictive performance, i.e. no model is fitted nor trained. The final estimated performance usually underestimates the true performance, but this negative bias is smaller than the optimistic uncorrected performance.

Note, that all metrics are for maximization. For this reason "mse.gomp", "mae.gomp", "ord_mae.gomp", "poisdev.gomp", "binomdev.gomp" are multiplied by -1.

Value

A list including:

`out.perf` The B out sampled performances. Their mean is the "bbc.perf" given above.

`bbc.perf` The bootstrap bias corrected performance of the chosen algorithm, model or configuration.

`ci` The (1- conf)% confidence interval of the BBC performance. It is based on the empirical or percentile method for bootstrap samples. The lower and upper 2.5% of the "out.perf".

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris <`mtsagris@uoc.gr`>.

References

Ioannis Tsamardinos, Elissavet Greasidou and Giorgos Borboudakis (2018). Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation. Machine Learning, 107: 1895–1922.

See Also

`cv.gomp`, `gomp`

Examples

```
predictions <- matrix( rbinom(200 * 50, 1, 0.7), ncol = 50)
y <- rbinom(200, 1, 0.5)
gomp::bbc(predictions, y, metric = ”auc.gomp”)
```

`Cross-validation for gamma-Orthogonal Matching Pursuit (gamma-OMP) algorithm`

Cross-validation for the gamma-Orthogonal Matching Pursuit (gamma-OMP) algorithm

Description

The function performs a k-fold cross-validation for identifying the best tolerance values for the γ-Orthogonal Matching Pursuit (γ-OMP) algorithm.

Usage

```
cv.gomp(y, x, kfolds = 10, folds = NULL, tol = seq(4, 9, by = 1),
task = "C", metric = NULL, metricbbc = NULL, modeler = NULL, test = NULL,
method = "ar2", B = 1)
```

Arguments

`y` The response variable.

`x` A matrix with the predictor variables.

`kfolds` The number of the folds in the k-fold Cross Validation (integer).

`folds` The folds of the data to use. If NULL the folds are created internally with the same function.

`tol` A vector of tolerance values.

`task` A character ("C", "R" or "S"). It can be "C" for classification (logistic, multinomial or ordinal regression), "R" for regression (robust and non robust linear regression, median regression, (zero inflated) poisson and negative binomial regression, beta regression), "S" for survival regresion (Cox, Weibull or exponential regression).

`metric` A metric function provided by the user. If NULL the following functions will be used: auc.gomp, mse.gomp, ci.gomp for classification, regression and survival analysis tasks, respectively. See details for more. If you know what you have put it here to avoid the function choosing somehting else. **Note** that you put these words as they are, without "".

`metricbbc` This is the same argument as "metric" with the difference that " " must be placed. If for example, metric = auc.mxm, here metricbbc = "auc.mxm". The same value must be given here. This argument is to be used with the function `bbc` which does bootstrap bias correction of the estimated performance (Tsamardinos, Greasidou and Borboudakis, 2018). This argument is valid if the last argument (B) is more than 1.

`modeler` A modeling function provided by the user. If NULL the following functions will be used: glm.gomp, lm.gomp, coxph.gomp for classification, regression and survival analysis tasks, respectively. See details for more. If you know what you have put it here to avoid the function choosing somehting else. **Note** that you put these words as they are, without "".

`test` A function object that defines the conditional independence test used in the SES function (see also SES help page). If NULL, "cor", "logistic" and "cox" are used for classification, regression and survival analysis tasks, respectively. If you know what you have put it here to avoid the function choosing sometting else. Not all tests can be included here. "mv", "gamma", and "tobit" are anot available.

`method` This is only for the "cor". You can either specify, "ar2" for the adjusted R-square or "sse" for the sum of squares of errors. The tolerance value in both cases must a number between 0 and 1. That will denote a percentage. If the percentage increase or decrease is less than the nubmer the algorithm stops. An alternative is "BIC" for BIC and the tolerance values are like in all other regression models.

`B` How many bootstrap re-samples to draw. This argument is to be used with the function `bbc` which does bootstrap bias correction of the estimated performance (Tsamardinos, Greasidou and Borboudakis, 2018). If you have thousands of observations then this might not be necessary, as there is no optimistic bias to be corrected. What is the lower limit cannot be told beforehand however. SES and MMPC however were designed for the low sample cases, hence, bootstrap bias correction is perhaps a must thing to do.

Details

For more details see also `gomp`.

Value

A list including:

`cv_results_all` A list with predictions, performances and selected variables for each fold and each tolerance value. The elements are called "preds", "performances" and "selectedVars".

`best_performance`
A numeric value that represents the best average performance.

`best_configuration`
A numeric value that represents the best tolerance value.

`bbc_best_performance`
The bootstrap bias corrected best performance if B was more than 1, othwerwise this is NULL.

`runtime` The runtime of the cross-validation procedure.

Bear in mind that the values can be extracted with the $ symbol, i.e. this is an S3 class output.

Author(s)

R implementation and documentation: Michail Tsagris <`mtsagris@uoc.gr`>.

References

Tsamardinos I., Greasidou E. and Borboudakis G. (2018). Bootstrapping the out-of-sample predictions for efficient and accurate cross-validation. Machine Learning, 107: 1895–1922. https://link.springer.com/article/10.1007/s10994-018-5714-4

Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos, I. (2022). The γ-OMP algorithm for feature selection with application to gene expression data. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 19(2): 1214–1224.

See Also

gomp, gomp.path, bbc

Examples

```
# simulate a dataset with continuous data
x <- matrix( rnorm(200 * 50), ncol = 50 )
# the target feature is the last column of the dataset as a vector
y <- x[, 50]
x <- x[, -50]
# run a 10 fold CV for the regression task
best_model <- cv.gomp(y, x, kfolds = 5, task = "R",
tol = seq(0.001, 0.01,by = 0.001), method = "ar2" )
```

`Generate random folds for cross-validation`

Generate random folds for cross-validation

Description

Random folds for use in a cross validation are generated. There is the option for stratified splitting as well.

Usage

```
makefolds(ina, nfolds = 10, stratified = TRUE, seed = NULL)
```

Arguments

`ina` A variable indicating the groupings.

`nfolds` The number of folds to produce.

`stratified` A boolean variable specifying whether stratified random (TRUE) or simple random (FALSE) sampling is to be used when producing the folds.

`seed` You can specify your own seed number here or leave it NULL.

Details

I was inspired by the command in the package **TunePareto** in order to do the stratified version.

Value

A list with nfolds elements where each elements is a fold containing the indices of the data.

Author(s)

Michail Tsagris.

R implementation and documentation: Michail Tsagris <mtsagris@uoc.gr>.

See Also

cv.gomp

Examples

```
a <- gomp::makefolds(iris[, 5], nfolds = 5, stratified = TRUE)
table(iris[a[[1]], 5])  ## 10 values from each group
```

The gamma-Orthogonal Matching Pursuit (gamma-OMP) algorithm

The gama-Orthogonal Matching Pursuit (gamma-OMP) algorithm

Description

The γ-Orthogonal Matching Pursuit (γ-OMP) algorithm.

Usage

```
gomp(y, x, xstand = TRUE, tol = qchisq(0.95, 1), test = "logistic", method = "ar2" )

gomp.path(y, x, xstand = TRUE, tol = c(4, 5, 6), test = "logistic", method = "ar2" )

boot.gomp(y, x, tol = qchisq(0.95, 1), test = "logistic", method = "ar2",
B = 500, ncores = 1)
```

Arguments

`y` The response variable, a numeric vector, a matrix or a Surv object.

`x` A matrix with continuous data, where the rows denote the observations and the columns are the variables.

`xstand` If this is TRUE the independent variables are standardised.

`tol` The tolerance value to terminate the algorithm. This is the change in the criterion value between two successive steps. The default value is the 95% quantile of the χ^2 distribution with 1 degree of freedom. For test = "normal" the BIC is already calculated.

In the case of "gomp.path" this is a vector of values. For each tolerance value the result of the gOMP is returned. It returns the whole path of solutions.

`test`	This denotes the parametric model to be used each time. It depends upon the nature of the target variable. The possible values are "normal" (or "cor" for the same purpose), "logistic", "poisson", "qpoisson", "qlogistic", "normlog", "mvreg", "negbin", "beta", "gamma", "mm", "quantreg", "ordinal", "tobit", "cox", "weibull", "log-logistic" and "multinom".
`method`	This is only for the "testIndFisher". You can either specify, "ar2" for the adjusted R-square or "sse" for the sum of squares of errors. The tolerance value in both cases must a number between 0 and 1. That will denote a percentage. If the percentage increase or decrease is less than the nubmer the algorithm stops. An alternative is "BIC" for BIC and the tolerance values are like in all other regression models.
`B`	How many bootstrap samples to generate. The gOMP will be performed for each of these samples.
`ncores`	How many cores to use. This argument is valid only if you have a multi-threaded machine.

Value

A list including:

`runtime`	The runtime of the algorithm
`phi`	The *phi* coefficient, returned in the quasi binomial (qlogistic), quasi Poisson (qpoisson), Gamma (gamma) and Gaussian with log link (normlog). In all other cases this is NULL.
`res`	For the case of "gomp" a matrix with two columns. The selected variable(s) and the criterion value at every step. For the case of "gomp.path" a matrix with many columns. Every column contains the selected variables for each tolerance calue, starting from the smallest value (which selected most variables). The final column is the deviance of the model at each step. For the "boot.gomp" this is a matrix with two columns. The first one is the selected variables and the second column is their proportion of selection.

Author(s)

Michail Tsagris

R implementation and documentation: Michail Tsagris `<mtsagris@uoc.gr>`.

References

Pati Y. C., Rezaiifar R. and Krishnaprasad P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In Signals, Systems and Computers. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on. IEEE.

Davis G. (1994). Adaptive Nonlinear Approximations. PhD thesis. http://www.geoffdavis.net/papers/dissertation.pdf

Mallat S. G. and Zhang Z. (1993). Matching pursuits with time-frequency dictionaries. IEEE Transactions on signal processing, 41(12), 3397–3415. https://www.di.ens.fr/~mallat/papiers/MallatPursuit93.pdf

Gharavi-Alkhansari M. and Huang T. S. (1998, May). A fast orthogonal matching pursuit algorithm. In Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on (Vol. 3, pp. 1389–1392). IEEE.

Chen S., Billings S. A. and Luo W. (1989). Orthogonal least squares methods and their application to non-linear system identification. International Journal of control, 50(5), 1873–1896.

Lozano A., Swirszcz G. and Abe N. (2011). Group orthogonal matching pursuit for logistic regression. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.

Razavi S. A. Ollila E. and Koivunen V. (2012). Robust greedy algorithms for compressed sensing. In Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European. IEEE.

Mazin Abdulrasool Hameed (2012). Comparative analysis of orthogonal matching pursuit and least angle regression. MSc thesis, Michigan State University. https://www.google.gr/url?sa=t&rct=j&q=&esrc=s&source=web&c

Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2018). Efficient feature selection on gene expression data: Which algorithm to use? BioRxiv.

Tsagris M., Papadovasilakis Z., Lakiotaki K. and Tsamardinos I. (2022). The γ-OMP algorithm for feature selection with application to gene expression data. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 19(2): 1214–1224.

Alharbi N. (2024). Variable selection with time-to-event data: Cox or Weibull regression? Communications in Statistics: Case Studies, Data Analysis and Applications (accepted for publication).

See Also

cv.gomp

Examples

```
x <- matrix( rnorm(500 * 50), ncol = 50)
y <- rnorm(500)
b <- gomp::gomp(y, x, test = "cor")
```

Index