Package ‘sail’

December 13, 2019

Title Sparse Additive Interaction Learning

Version 0.1.0

Description Sparse additive interaction learning with the strong heredity property, i.e., an interaction is selected only if its corresponding main effects are also included. Fits a linear model with non-linear interactions via penalized maximum likelihood. Interactions are limited to a single exposure or environment variable. For more information, see the website below and the accompanying paper: Bhatnagar et al., A sparse additive model for high-dimensional interactions with an exposure variable", 2019, <DOI:10.1101/445304>.

Depends R (>= 3.4.0)

Imports glmnet, gglasso, methods

Suggests grpreg, truncnorm, foreach, doParallel, testthat, covr, vdiffr, knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

LazyData true

BugReports https://github.com/sahirbhatnagar/sail/issues

URL https://sahirbhatnagar.com/sail

RoxygenNote 6.1.1

VignetteBuilder knitr

NeedsCompilation no


Maintainer Sahir Bhatnagar <sahir.bhatnagar@gmail.com>

Repository CRAN

Date/Publication 2019-12-13 15:00:02 UTC


**R topics documented:**

createfolds ............................... 2

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>createfolds</td>
<td>2</td>
</tr>
<tr>
<td>cv.lspath</td>
<td>3</td>
</tr>
<tr>
<td>cv.sail</td>
<td>5</td>
</tr>
<tr>
<td>design_sail</td>
<td>7</td>
</tr>
<tr>
<td>gendata</td>
<td>8</td>
</tr>
<tr>
<td>gendataPaper</td>
<td>11</td>
</tr>
<tr>
<td>oasis</td>
<td>12</td>
</tr>
<tr>
<td>plot.cv.sail</td>
<td>13</td>
</tr>
<tr>
<td>plot.sail</td>
<td>14</td>
</tr>
<tr>
<td>plotInter</td>
<td>15</td>
</tr>
<tr>
<td>plotMain</td>
<td>17</td>
</tr>
<tr>
<td>predict.cv.sail</td>
<td>18</td>
</tr>
<tr>
<td>predict.sail</td>
<td>20</td>
</tr>
<tr>
<td>print.sail</td>
<td>21</td>
</tr>
<tr>
<td>Q_theta</td>
<td>22</td>
</tr>
<tr>
<td>sail</td>
<td>23</td>
</tr>
<tr>
<td>sail-internal</td>
<td>28</td>
</tr>
<tr>
<td>sailsim</td>
<td>29</td>
</tr>
<tr>
<td>standardize</td>
<td>30</td>
</tr>
</tbody>
</table>

Index 31

---

**createfolds**  
*Create CV Folds*

**Description**

createfolds splits the data into k groups. Taken from the caret package (see references for details)

**Usage**

createfolds(y, k = 10, list = FALSE, returnTrain = FALSE)

**Arguments**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>vector of response</td>
</tr>
<tr>
<td>k</td>
<td>integer for the number of folds.</td>
</tr>
<tr>
<td>list</td>
<td>logical - should the results be in a list (TRUE) or a matrix</td>
</tr>
<tr>
<td>returnTrain</td>
<td>a logical. When true, the values returned are the sample positions corresponding to the data used during training. This argument only works in conjunction with list = TRUE</td>
</tr>
</tbody>
</table>
Details

For numeric y, the sample is split into groups sections based on percentiles and sampling is done within these subgroups.

Value

A vector of CV fold ID’s for each observation in y.

References


cv.lspath

Compute cross validation error

Description

functions used to calculate cross validation error and used by the cv.sail function

Usage

cv.lspath(outlist, lambda, x, y, e, weights, foldid, type.measure, grouped, keep = FALSE)

cvcompute(mat, weights, foldid, nlams)

getmin(lambda, cvm, cvsd)

lambda.interp(lambda, s)

Arguments

outlist list of cross validated fitted models. List is of length equal to nfolds argument in cv.sail function

lambda a user supplied lambda sequence. Typically, by leaving this option unspecified users can have the program compute its own lambda sequence based on nlambdas 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. Default: NULL.

x input matrix of dimension n x p, where n is the number of subjects and p is number of X variables. Each row is an observation vector. Can be a high-dimensional (n < p) matrix. Can be a user defined design matrix of main effects only (without intercept) if expand=FALSE

y response variable. For family="gaussian" should be a 1 column matrix or numeric vector. For family="binomial", should be a 1 column matrix or numeric vector with -1 for failure and 1 for success.
exposure or environment vector. Must be a numeric vector. Factors must be converted to numeric.

weights observation weights. Default is 1 for each observation. Currently NOT IMPLEMENTED.

foldid numeric vector indicating which fold each observation belongs to

type.measure loss to use for cross-validation. Currently only 3 options are implemented. The default is type.measure="deviance", which uses squared-error for gaussian models (and is equivalent to type.measure="mse") there). type.measure="mae" (mean absolute error) can also be used which measures the absolute deviation from the fitted mean to the response (|y - \hat{y}|).

grouped This is an experimental argument, with default TRUE, and can be ignored by most users. This refers to computing nfolds separate statistics, and then using their mean and estimated standard error to describe the CV curve. If grouped=FALSE, an error matrix is built up at the observation level from the predictions from the nfolds fits, and then summarized (does not apply to type.measure="auc"). Default: TRUE.

keep If keep=TRUE, a prevalidated array is returned containing fitted values for each observation and each value of lambda. This means these fits are computed with this observation and the rest of its fold omitted. The fold vector is also returned. Default: FALSE

mat matrix of predictions

nlams number of lambdas fit

cvm mean cv error

cvse sd of cv error

s numeric value of lambda

Details

The output of the cv.lspath function only returns values for those tuning parameters that converged. cvcompute,getmin,lambda.interp are taken verbatim from the glmnet package

Functions

- cvcompute: Computations for crossvalidation error
- getmin: get lambda.min and lambda.1se
- lambda.interp: Interpolation function.

References


See Also

cv.sail
cv.sail

Cross-validation for sail

Description

Does k-fold cross-validation for sail and determines the optimal tuning parameter $\lambda$.

Usage

```r
cv.sail(x, y, e, ..., weights, lambda = NULL, type.measure = c("mse", "deviance", "class", "auc", "mae"), nfolds = 10, foldid, grouped = TRUE, keep = FALSE, parallel = FALSE)
```

Arguments

- **x**: input matrix of dimension $n \times p$, where $n$ is the number of subjects and $p$ is number of X variables. Each row is an observation vector. Can be a high-dimensional ($n < p$) matrix. Can be a user defined design matrix of main effects only (without intercept) if `expand=FALSE`
- **y**: response variable. For `family="gaussian"` should be a 1 column matrix or numeric vector. For `family="binomial"`, should be a 1 column matrix or numeric vector with -1 for failure and 1 for success.
- **e**: exposure or environment vector. Must be a numeric vector. Factors must be converted to numeric.
- **...**: other arguments that can be passed to `sail`
- **weights**: observation weights. Default is 1 for each observation. Currently NOT IMPLEMENTED.
- **lambda**: Optional user-supplied lambda sequence; default is NULL, and `sail` chooses its own sequence.
- **type.measure**: loss to use for cross-validation. Currently only 3 options are implemented. The default is `type.measure="deviance"`, which uses squared-error for gaussian models (and is equivalent to `type.measure="mse"` there). `type.measure="mae"` (mean absolute error) can also be used which measures the absolute deviation from the fitted mean to the response ($|y - \hat{y}|$).
- **nfolds**: number of folds. 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`. Default: 10
- **foldid**: an optional vector of values between 1 and `nfold` identifying what fold each observation is in. If supplied, `nfold` can be missing. Often used when wanting to tune the second tuning parameter ($\alpha$) as well (see details).
- **grouped**: This is an experimental argument, with default TRUE, and can be ignored by most users. This refers to computing `nfolds` separate statistics, and then using their mean and estimated standard error to describe the CV curve. If `grouped=FALSE`, an error matrix is built up at the observation level from the predictions from the `nfold` fits, and then summarized (does not apply to `type.measure="auc"`). Default: TRUE.
keep If keep=TRUE, a prevalidated array is returned containing fitted values for each observation and each value of lambda. This means these fits are computed with this observation and the rest of its fold omitted. The foldid vector is also returned. Default: FALSE

parallel If TRUE, use parallel foreach to fit each fold. Must register parallel beforehand using the registerDoParallel function from the doParallel package. See the example below for details. Default: FALSE

Details

The function runs sail nfolds+1 times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. Note that a new lambda sequence is computed for each of the folds and then we use the predict method to get the solution path at each value of the original lambda sequence. The error is accumulated, and the average error and standard deviation over the folds is computed. Note that cv.sail does NOT search for values for alpha. A specific value should be supplied, else alpha=0.5 is assumed by default. If users would like to cross-validate alpha as well, they should call cv.sail with a pre-computed vector foldid, and then use this same fold vector in separate calls to cv.sail with different values of alpha. Note also that the results of cv.sail are random, since the folds are selected at random. Users can reduce this randomness by running cv.sail many times, and averaging the error curves.

Value

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

- lambda the values of converged lambda used in the fits.
- cvm The mean cross-validated error - a vector of length length(lambda).
- cvsd estimate of standard error of cvm.
- cvup upper curve = cvm+cvsd.
- cvlo lower curve = cvm-cvsd.
- nzero number of non-zero coefficients at each lambda. This is the sum of the total non-zero main effects and interactions. Note that when expand=TRUE, we only count a variable once in the calculation of nzero, i.e., if a variable is expanded to three columns, then this is only counted once even though all three coefficients are estimated to be non-zero
- name a text string indicating type of measure (for plotting purposes).
- sail.fit a fitted sail object for the full data.
- lambda.min value of lambda that gives minimum cvm.
- lambda.1se largest value of lambda such that error is within 1 standard error of the minimum.
- fit.preval if keep=TRUE, this is the array of prevalidated fits. Some entries can be NA, if that and subsequent values of lambda are not reached for that fold
- foldid if keep=TRUE, the fold assignments used

Note

The skeleton of this function and the documentation were taken straight from the glmnet package. See references for details.
design_sail

References


See Also

bs sail

Examples

```r
f.basis <- function(i) splines::bs(i, degree = 3)
data("sailsim")
# Parallel
library(doParallel)
c1 <- makeCluster(2)
registerDoParallel(c1)
cvfit <- cv.sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
               parallel = TRUE, nlambda = 10,
               maxit = 25, basis = f.basis,
               nfolds = 3, dfmax = 5)
stopCluster(c1)
# plot cross validated curve
plot(cvfit)
# solution at lambda.min
coef(cvfit, s = "lambda.min")
# solution at lambda.1se
coef(cvfit, s = "lambda.1se")
# non-zero coefficients at lambda.min
predict(cvfit, s = "lambda.min", type = "nonzero")
```

---

design_sail  

Sail design matrix

Description

Create design matrix used in sail function

Usage

design_sail(x, e, expand, group, basis, nvars, vnames, center.x, center.e)
Arguments

x  
input matrix of dimension n x p, where n is the number of subjects and p is number of X variables. Each row is an observation vector. Can be a high-dimensional (n < p) matrix. Can be a user defined design matrix of main effects only (without intercept) if expand=FALSE

e  
exposure or environment vector. Must be a numeric vector. Factors must be converted to numeric.

expand  
should basis be applied to every column of x (logical). Set to FALSE if you want a user defined main effects design matrix. If FALSE the group membership argument must also be supplied. Default: TRUE.

group  
a vector of consecutive integers, starting from 1, describing the grouping of the coefficients. Only required when expand=FALSE.

basis  
user defined basis expansion function. This function will be applied to every column in x. Specify function(i) i if no expansion is desired. Default: function(i) splines::bs(i,df = 5).

nvars  
number of variables

vnames  
variable names

center.x  
should the columns of x (after basis expansion) be centered (logical). Default: TRUE.

center.e  
should exposure variable e be centered. Default: TRUE.

gendata

Simulation Scenario from Bhatnagar et al. (2018+) sail paper

Description

Function that generates data of the different simulation studies presented in the accompanying paper. This function requires the truncnorm package to be installed.

Usage

gendata(n, p, corr, E = truncnorm::rtruncnorm(n, a = -1, b = 1), betaE, SNR, parameterIndex)

Arguments

n  
number of observations

p  
number of main effect variables (X)

corr  
correlation between predictors

E  
simulated environment vector of length n. Can be continuous or integer valued. Factors must be converted to numeric. Default: truncnorm::rtruncnorm(n,a = -1,b = 1)

betaE  
exposure effect size

SNR  
signal to noise ratio

parameterIndex  
simulation scenario index. See details for more information.
Details

We evaluate the performance of our method on three of its defining characteristics: 1) the strong heredity property, 2) non-linearity of predictor effects and 3) interactions.

**Heredity Property** Truth obeys strong hierarchy (parameterIndex = 1)

\[ Y^* = \sum_{j=1}^{4} f_j(X_j) + \beta E * X_E + X_E * f_3(X_3) + X_E * f_4(X_4) \]

Truth obeys weak hierarchy (parameterIndex = 2)

\[ Y^* = f_1(X_1) + f_2(X_2) + \beta E * X_E + X_E * f_3(X_3) + X_E * f_4(X_4) \]

Truth only has interactions (parameterIndex = 3)

\[ Y^* = X_E * f_3(X_3) + X_E * f_4(X_4) \]

**Non-linearity** Truth is linear (parameterIndex = 4)

\[ Y^* = \sum_{j=1}^{4} \beta_j X_j + \beta E * X_E + X_E * X_3 + X_E * X_4 \]

**Interactions** Truth only has main effects (parameterIndex = 5)

\[ Y^* = \sum_{j=1}^{4} f_j(X_j) + \beta E * X_E \]

The functions are from the paper by Lin and Zhang (2006):

\[ f_1 \gets \text{function(t) 5 * t} \]
\[ f_2 \gets \text{function(t) 3 * (2 * t - 1)^2} \]
\[ f_3 \gets \text{function(t) 4 * sin(2 * pi * t) / (2 - sin(2 * pi * t))} \]
\[ f_4 \gets \text{function(t) 6 * (0.1 * sin(2 * pi * t) + 0.2 * cos(2 * pi * t) + 0.3 * sin(2 * pi * t)^2 + 0.4 * cos(2 * pi * t)^3 + 0.5 * sin(2 * pi * t)^3) } \]

The response is generated as

\[ Y = Y^* + k * \text{error} \]

where \( Y^* \) is the linear predictor, the error term is generated from a standard normal distribution, and \( k \) is chosen such that the signal-to-noise ratio is \( \text{SNR} = \text{Var}(Y^*)/\text{Var(error)} \), i.e., the variance of the response variable \( Y \) due to error is \( 1/\text{SNR} \) of the variance of \( Y \) due to \( Y^* \).

The covariates are simulated as follows as described in Huang et al. (2010). First, we generate \( w_1, w_p, u, v \) independently from Normal(0, 1) truncated to the interval [0, 1] for \( i = 1, \ldots, n \). Then we set \( x_j = (w_j + t * u)/(1 + t) \) for \( j = 1, \ldots, 4 \) and \( x_j = (w_j + t * v)/(1 + t) \) for \( j = 5, \ldots, p \), where the parameter \( t \) controls the amount of correlation among predictors.

This leads to a compound symmetry correlation structure where \( \text{Corr}(x_j, x_k) = t^2/(1 + t^2) \), for \( 1 \leq j \leq 4, 1 \leq k \leq 4 \), and \( \text{Corr}(x_j, x_k) = t^2/(1 + t^2) \), for \( 5 \leq j \leq p, 5 \leq k \leq p \), but the covariates of the nonzero and zero components are independent.
Value

A list with the following elements:

- **x** matrix of dimension $nxp$ of simulated main effects
- **y** simulated response vector of length $n$
- **e** simulated exposure vector of length $n$
- **Y.star** linear predictor vector of length $n$
- **f1** the function $f_1$ evaluated at $x_1$ ($f_1(x_1)$)
- **f2** the function $f_1$ evaluated at $x_1$ ($f_1(x_1)$)
- **f3** the function $f_1$ evaluated at $x_1$ ($f_1(x_1)$)
- **f4** the function $f_1$ evaluated at $x_1$ ($f_1(x_1)$)
- **betaE** the value for $\beta_E$
- **f1.f** the function $f_1$
- **f2.f** the function $f_2$
- **f3.f** the function $f_3$
- **f4.f** the function $f_4$
- **X1** an $n$ length vector of the first predictor
- **X2** an $n$ length vector of the second predictor
- **X3** an $n$ length vector of the third predictor
- **X4** an $n$ length vector of the fourth predictor
- **scenario** a character representing the simulation scenario identifier as described in Bhatnagar et al. (2018+)
- **causal** character vector of causal variable names
- **not_causal** character vector of noise variables

References


Examples

```r
DT <- gendata(n = 75, p = 100, corr = 0, betaE = 2, SNR = 1, parameterIndex = 1)
```
Simulation Scenario from Bhatnagar et al. (2018+) sail paper

description generates the different simulation scenarios. This function is not intended to be called directly by users. See gendata

Usage
gendataPaper(n, p, corr = 0, E = truncnorm::rtruncnorm(n, a = -1, b = 1), betaE = 2, SNR = 2, hierarchy = c("strong", "weak", "none"), nonlinear = TRUE, interactions = TRUE, causal, not_causal)

Arguments
n number of observations
p number of main effect variables (X)
corr correlation between predictors
E simulated environment vector of length n. Can be continuous or integer valued. Factors must be converted to numeric. Default: truncnorm::rtruncnorm(n,a = -1,b = 1)
betaE exposure effect size
SNR signal to noise ratio
hierarchy type of hierarchy. Can be one of c("strong", "weak", "none"). Default: "strong"
nonlinear simulate non-linear terms (logical). Default: TRUE
interactions simulate interaction (logical). Default: TRUE
causal character vector of causal variable names
not_causal character vector of noise variables

details Requires installation of truncnorm package. Not meant to be called directly by user. Use gendata.

Value
A list with the following elements:
x matrix of dimension n*p of simulated main effects
y simulated response vector of length n
e simulated exposure vector of length n
Y.star linear predictor vector of length n
f1 the function f1 evaluated at x_1 (f1(X1))
\( f_2 \) the function \( f_1 \) evaluated at \( x_1 \) \( (f_1(X_1)) \)
\( f_3 \) the function \( f_1 \) evaluated at \( x_1 \) \( (f_1(X_1)) \)
\( f_4 \) the function \( f_1 \) evaluated at \( x_1 \) \( (f_1(X_1)) \)
\( \text{betaE} \) the value for \( \beta_E \)
\( f_{1.f} \) the function \( f_1 \)
\( f_{2.f} \) the function \( f_2 \)
\( f_{3.f} \) the function \( f_3 \)
\( f_{4.f} \) the function \( f_4 \)
\( X_1 \) an \( n \) length vector of the first predictor
\( X_2 \) an \( n \) length vector of the second predictor
\( X_3 \) an \( n \) length vector of the third predictor
\( X_4 \) an \( n \) length vector of the fourth predictor
\( \text{scenario} \) a character representing the simulation scenario identifier as described in Bhatnagar et al. (2018+)
\( \text{causal} \) character vector of causal variable names
\( \text{not}_{-}\text{causal} \) character vector of noise variables

See Also

rnorm, cor, gendata

---

**oasis**

*OASIS Brain Data*

Description

A dataset containing a small subset of the OASIS Brain Data project. Noise variables have been added to increase the number of predictors and show the utility of the sail package.

Usage

oasis

Format

A list with 3 elements:

- \( x \) a matrix of dimension \( 136 \times 30 \) with the following columns:
  - Age Patient’s age
  - EDUC Education
  - MMSE Mini-Mental State Exam
  - eTIV Estimated Total Intracranial Volume
  - nWBV Normalized Whole Brain Volume
ASF  Atlas scaling factor
noise[1-24]  24 independent standard normal noise variables
y  a numeric vector of length 136 representing the right Hippocampal volume for each patient
e  a binary 0/1 vector of length 136, representing Dementia status. 0: Non-demented, 1: Demented

Source
https://github.com/stnava/RMI/tree/master/tomfletcher
http://www.oasis-brains.org/

Examples
oasis

-----

plot.cv.sail  
Plot the cross-validation curve produced by cv.sail

Description
Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the
lambda values used.

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

Arguments
  x  fitted cv.sail object
  sign.lambda  Either plot against log(lambda) (default) or its negative if sign.lambda=-1.
  ...  Other graphical parameters to plot

Details
This is a port of plot.cv.glmnet

Value
A plot is produced and nothing is returned

References
Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized
See Also

`sail`, `cv.sail`

Examples

data("sailsim")
f.basis <- function(i) splines::bs(i, degree = 3)
library(doParallel)
cl <- makeCluster(2)
registerDoParallel(cl)
cvfit <- cv.sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
  parallel = TRUE, nlambda = 10,
  maxit = 100, basis = f.basis,
  nfolds = 3, dfmax = 10)
stopCluster(cl)
plot(cvfit)

---

`plot.sail`  
*Plot Method for sail object*

Description

Produces a coefficient profile plot of the coefficient paths for a fitted `sail` object. Both main effects and interactions (if present) are plotted.

Usage

```r
## S3 method for class 'sail'
plot(x, type = c("both", "main", "interaction"), ...)
```

Arguments

- `x`  
  fitted sail object

- `type`  
  which type of predictors should be plotted. `type="both"` will plot the solution path for main effects and interactions, `type="main"` will only plot solution path of main effects (this also includes the exposure variable) and `type="interaction"` will only plot solution path for interaction effects. Default: `c("both", "main", "interaction")`. Default: `type="both"`.

- `...`  
  other graphical parameters passed to `plot`

Details

A coefficient profile plot is produced

Value

A plot is produced and nothing is returned
plotInter

See Also
sail, cv.sail

Examples

data("sailsim")
f.basis <- function(i) splines::bs(i, degree = 3)
fit <- sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
basis = f.basis, dfmax = 10, nlambda = 10, maxit = 100)
plot(fit)

---

plotInter

Plot Interaction Effects from sail object

Description
Takes a fitted sail object produced by sail() or cv.sail()$sail.fit and plots a persp for a pre-specified variable at a given value of lambda and on the scale of the linear predictor. Currently only implemented for type="gaussian"

Usage

plotInter(object, x, xvar, s, f.truth, interaction.only = TRUE,
  truthonly = FALSE, npoints = 30, col = c("#56B4E9", "#D55E00"),
  title_z = "", xlab, ylab, zlab, ...)

Arguments

object  a fitted sail object as produced by sail() or cv.sail()$sail.fit
x       original data supplied to the original call to sail
xvar    a character corresponding to the predictor to be plotted. Only one variable name should be supplied, if more than one is supplied, only the first element will be plotted. This variable name must be in colnames(x).
s       a single value of the penalty parameter lambda at which coefficients will be extracted via the coef method for objects of class "sail". If more than one is supplied, only the first one will be used.
f.truth true function. Only used for simulation purposes when the truth is known. The function takes as a input two numeric vectors e.g. f(x, e) corresponding the xvar column in x of length nrow(x) and the exposure variable contained in the sail object. A second persp will be plotted for the truth
interaction.only if TRUE only the interaction part is used to calculate the linear predictor, i.e., linearpredictor = E*f(X)*interaction.effects. If FALSE, then linearpredictor = E*β_E+f(X)*interaction.effects+E*f(X)*interaction.effects. Default: TRUE
truthonly  only plot the truth. f.truth must be specified if this argument is set to TRUE. Default: FALSE

npoints  number of points in the grid to calculate the perspective plot. Default: 30

col  color of the line. The first element corresponds to the color used for the estimated function and the second element is for the true function (if f.truth is specified). Default: c("#D55E00", ":009E73")

title_z  title for the plot, Default: ""

xlab  character for xlabel. if missing, variable name is used

ylab  character for ylabel. if missing, variable name is used

zlab  character for zlabel. if missing, variable name is used

...  currently ignored

Value

A plot is produced and nothing is returned

See Also

persp coef.sail predict.sail, rug

Examples

f.basis <- function(i) splines::bs(i, degree = 3)
# Parallel
library(doParallel)
c1 <- makeCluster(2)
registerDoParallel(c1)
cvfit <- cv.sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
                parallel = TRUE, nlambdas = 10,
                maxit = 100, basis = f.basis,
                nfolds = 3, dfmax = 10)
stopCluster(c1)
# plot cv-error curve
plot(cvfit)
# non-zero estimated coefficients at lambda.min
predict(cvfit, type = "nonzero", s="lambda.min")
# plot interaction effect for X4 and the true interaction effect also
plotInter(cvfit$sail.fit, x = sailsim$x, xvar = "X3",
          f.truth = sailsim$f4.inter,
          s = cvfit$lambda.min,
          title_z = "Estimated")
### plotMain

**Plot Estimated Component Smooth Functions for Main Effects**

#### Description

Takes a fitted sail object produced by `sail()` or `cv.sail()$sail.fit` and plots the component smooth function for a pre-specified variable at a given value of lambda and on the scale of the linear predictor. Currently only implemented for `type="gaussian"`

#### Usage

```r
plotMain(object, x, xvar, s, f.truth, col = c("#D55E00", "#009E73"),
         legend.position = "bottomleft", rug = TRUE, ...)
```

#### Arguments

- **object**: a fitted sail object as produced by `sail()` or `cv.sail()$sail.fit`
- **x**: original data supplied to the original call to `sail`
- **xvar**: a character corresponding to the predictor to be plotted. Only one variable name should be supplied, if more than one is supplied, only the first element will be plotted. This variable name must be in `colnames(x)`.
- **s**: a single value of the penalty parameter lambda at which coefficients will be extracted via the `coef` method for objects of class "sail". If more than one is supplied, only the first one will be used.
- **f.truth**: true function. Only used for simulation purposes when the truth is known. The function takes as a input a numeric vector corresponding the `xvar` column in `x` of length `nrow(x)`. A second line will be plotted for the truth and a legend is added to the plot.
- **col**: color of the line. The first element corresponds to the color used for the estimated function and the second element is for the true function (if `f.truth` is specified). Default: c("#D55E00","#009E73")
- **legend.position**: position of the legend. Only used when `f.truth` is specified. Default: 'bottomleft'. Can be a single keyword from the list "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". This places the legend on the inside of the plot frame at the given location. Partial argument matching is used.
- **rug**: adds a rug representation (1-d plot) of the data to the plot, logical. Default: TRUE.
- **...**: other graphical parameters passed to `plot`.

#### Details

The linear predictor $\text{basis}(xvar) \ast \beta_{xvar}$ is plotted against `xvar`, where `basis` is the expansion provided in the original call to `sail`. 
predict.cv.sail

Make predictions from a cv.sail object

Description

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

Usage

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

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

Arguments

- **object**: fitted cv.sail object
- **newx**: matrix of new values for x at which predictions are to be made. Do not include the intercept (this function takes care of that). Must be a matrix. This argument is not used for type=c("coefficients","nonzero"). This matrix will be passed to design_sail to create the design matrix necessary for predictions. This matrix must have the same number of columns originally supplied to the sail fitting function.
- **newe**: vector of new values for the exposure variable e. This is passed to the design_sail function.
- **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.
- **...**: other arguments passed to predict.sail

Details

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

Value

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

See Also

predict.sail

Examples

data("sailsim")
f.basis <- function(i) splines::bs(i, degree = 3)
library(doParallel)
cl <- makeCluster(2)
registerDoParallel(cl)
cvfit <- cv.sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
parallel = TRUE, nlambda = 10,
maxit = 20, basis = f.basis,
nfolds = 3, dfmax = 5)
stopCluster(cl)
predict(cvfit) # predict at "lambda.1se"
predict(cvfit, s = "lambda.min") # predict at "lambda.min"
predict(cvfit, s = 0.5) # predict at specific value of lambda
predict(cvfit, type = "nonzero") # non-zero coefficients at lambda.1se

# predict response for new data set
newx <- sailsim$x * 1.10
newe <- sailsim$e * 2
predict(cvfit, newx = newx, newe = newe, s = "lambda.min")

**predict.sail**

*Make predictions from a sail object*

**Description**

Similar to other predict methods, this function predicts fitted values, logits, coefficients and more from a fitted sail object.

**Usage**

```r
## S3 method for class 'sail'
predict(object, newx, newe, s = NULL, type = c("link", "response", "coefficients", "nonzero", "class"), ...)

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

**Arguments**

- **object**
  - Fitted sail model object
- **newx**
  - matrix of new values for \( x \) at which predictions are to be made. Do not include the intercept (this function takes care of that). Must be a matrix. This argument is not used for type=c("coefficients","nonzero"). This matrix will be passed to `design_sail` to create the design matrix necessary for predictions. This matrix must have the same number of columns originally supplied to the sail fitting function.
- **newe**
  - vector of new values for the exposure variable \( e \). This is passed to the `design_sail` function.
- **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" gives the linear predictors for "binomial" (not implemented yet); for "gaussian" models it gives the fitted values. Type "response" gives the fitted probabilities for "binomial" (not implemented yet), for "gaussian" type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for \( s \). Note that for "binomial" models, results are returned only for the class corresponding to the second level of the factor response (not implemented yet). Type "class" applies only to "binomial" models, and produces the class label corresponding to the maximum probability (not implemented yet). Type "nonzero" returns a list of the nonzero coefficients for each value of \( s \). Default: "link"
- **...**
  - currently ignored
Details

R Source code file for predict, coef, plot and print methods for the sail package
Author: Sahir Bhatnagar
Created: 2016
Updated: April 6, 2018

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.
coef(...) is equivalent to predict(sail.object, type="coefficients", ...)

Value

The object returned depends on type.

See Also

predict.cv.sail

Examples

f.basis <- function(i) splines::bs(i, degree = 3)
fit <- sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
basis = f.basis, dfmax = 5, nlambda = 10, maxit = 20)
predict(fit) # predicted response for whole solution path
predict(fit, s = 0.45) # predicted response for a single lambda value
predict(fit, s = c(2.15, 0.32, 0.40), type="nonzero") # nonzero coefficients

print.sail

Print Method for sail object

Description

Print a summary of the sail path at each step along the path.

Usage

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

Arguments

x
treated sail object
digits

significant digits in printout. Default: max(3,getOption("digits") -3)

... additional print arguments
Details

The call that produced the object x is printed, followed by a five-column matrix with columns df_main, df_interaction, df_environment, %Dev and Lambda. The df_ columns are the corresponding number of nonzero coefficients for main effects, interactions and exposure, respectively. %dev is the percent deviance explained (relative to the null deviance). For type="gaussian" this is the r-squared.

Value

OUTPUT_DESCRIPTION

See Also

sail, cv.sail

Examples

data("sailsim")
f.basis <- function(i) splines::bs(i, degree = 3)
fit <- sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
          basis = f.basis, dfmax = 5, nlambda = 10,
          maxit = 20)

Q_theta

Objective function

Description

calculates likelihood function. Used to assess convergence of fitting algorithm. This corresponds to the Q(theta) function in the paper

Usage

Q_theta(R, nobs, lambda, alpha, we, wj, wje, betaE, theta_list, gamma)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>residual</td>
</tr>
<tr>
<td>nobs</td>
<td>number of observations</td>
</tr>
<tr>
<td>lambda</td>
<td>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. Default: NULL.</td>
</tr>
</tbody>
</table>
alpha
the mixing tuning parameter, with $0 < \alpha < 1$. It controls the penalization strength between the main effects and the interactions. The penalty is defined as

$$
\lambda(1 - \alpha) |w_e \beta_e| + \sum w_j |\beta_j|_2 + \lambda \alpha \sum w_{je} |\gamma_j|
$$

Larger values of alpha will favor selection of main effects over interactions. Smaller values of alpha will allow more interactions to enter the final model. Default: 0.5

we
penalty factor for exposure variable

wj
penalty factor for main effects

wje
penalty factor for interactions

betaE
estimate of exposure effect

theta_list
estimates of main effects

gamma
estimates of gamma parameter

Value
value of the objective function

---

Fit Sparse Additive Interaction Model with Strong Heredity

Description
Function to fit the Sparse Additive Interaction Model with strong heredity for a sequence of tuning parameters. This is a penalized regression method that ensures the interaction term is non-zero only if its corresponding main-effects are non-zero. This model only considers the interactions between a single exposure (E) variable and a high-dimensional matrix (X). Additive (non-linear) main effects and interactions can be specified by the user. This can also be seen as a varying-coefficient model.

Usage

```r
sail(x, y, e, basis = function(i) splines::bs(i, df = 5),
    strong = TRUE, group.penalty = c("gglasso", "grMCP", "grSCAD"),
    family = c("gaussian", "binomial"), center.x = TRUE,
    center.e = TRUE, expand = TRUE, group, weights,
    penalty.factor = rep(1, 1 + 2 * nvars), lambda.factor = ifelse(nobs <
        (1 + 2 * bscols * nvars), 0.01, 1e-04), lambda = NULL, alpha = 0.5,
    nlambda = 100, thresh = 1e-04, fdev = 1e-05, maxit = 1000,
    dfmax = 2 * nvars + 1, verbose = 0)
```
Arguments

\(x\): input matrix of dimension \(n \times p\), where \(n\) is the number of subjects and \(p\) is number of X variables. Each row is an observation vector. Can be a high-dimensional (\(n < p\)) matrix. Can be a user defined design matrix of main effects only (without intercept) if \(\text{expand}=\text{FALSE}\).

\(y\): response variable. For \text{family}="gaussian" should be a 1 column matrix or numeric vector. For \text{family}="binomial", should be a 1 column matrix or numeric vector with -1 for failure and 1 for success.

\(e\): exposure or environment vector. Must be a numeric vector. Factors must be converted to numeric.

\text{basis}: user defined basis expansion function. This function will be applied to every column in \(x\). Specify \(\text{function}(i)\) if no expansion is desired. Default: \(\text{function}(i) \text{splines::bs}(i, \text{df}=5)\).

\text{strong}: Flag specifying strong hierarchy (\text{TRUE}) or weak hierarchy (\text{FALSE}). Default \text{FALSE}.

\text{group.penalty}: group lasso penalty. Can be one of "gglasso" (group lasso), "grMCP" (group MCP) or "grSCAD" (group SCAD). See references for details. Default: "gglasso".

\text{family}: response type. See \(y\) for details. Currently only \text{family}="gaussian" is implemented. Default: "gaussian".

\text{center.x}: should the columns of \(x\) (after basis expansion) be centered (logical). Default: \text{TRUE}.

\text{center.e}: should exposure variable \(e\) be centered. Default: \text{TRUE}.

\text{expand}: should \text{basis} be applied to every column of \(x\) (logical). Set to \text{FALSE} if you want a user defined main effects design matrix. If \text{FALSE} the \text{group membership} argument must also be supplied. Default: \text{TRUE}.

\text{group}: a vector of consecutive integers, starting from 1, describing the grouping of the coefficients. Only required when \text{expand}=\text{FALSE}.

\text{weights}: observation weights. Default is 1 for each observation. Currently NOT IMPLEMENTED.

\text{penalty.factor}: separate penalty factors can be applied to each coefficient. This is a number that multiplies \(\lambda\) to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. Default is 1 for all variables. Must be of length \(1 + 2 \times \text{ncol}(x)\) where the first entry corresponds to the \text{penalty.factor} for \(e\), the next \(\text{ncol}(x)\) entries correspond to main effects, and the following \(\text{ncol}(x)\) entries correspond to the interactions.

\text{lambda.factor}: the factor for getting the minimal \(\lambda\) in the lambda sequence, where \(\min(\lambda) = \text{lambda.factor} \times \max(\lambda)\). \(\max(\lambda)\) 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 \(q\) (the total number of predictors in the design matrix - including interactions). If \(N > q\), the default is \(1e^{-4}\), close to zero. If \(N < p\), the default is \(0.01\). A very small value of \(\text{lambda.factor}\) will lead to a saturated fit.
lambda: a user supplied lambda sequence. Typically, by leaving this option unspecified, users can have the program compute its own lambda sequence based on nl lambda 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. Default: NULL.

alpha: the mixing tuning parameter, with $0 < \alpha < 1$. It controls the penalization strength between the main effects and the interactions. The penalty is defined as

$$
\lambda (1 - \alpha) (w_e|\beta_e| + \sum w_j||\beta_j||_2) + \lambda \alpha (\sum w_{je}|\gamma_j|)
$$

Larger values of alpha will favor selection of main effects over interactions. Smaller values of alpha will allow more interactions to enter the final model. Default: 0.5

nl lambda: the number of lambda values. Default: 100

thresh: convergence threshold for coordinate descent. Each coordinate-descent loop continues until the change in the objective function after all coefficient updates is less than thresh. Default: $1e^{-04}$.

fdev: minimum fractional change in deviance for stopping path. Default: $1e^{-5}$.

maxit: maximum number of outer-loop iterations allowed at fixed lambda value. If models do not converge, consider increasing maxit. Default: 1000.

dfmax: limit the maximum number of variables in the model. Useful for very large q (the total number of predictors in the design matrix - including interactions), if a partial path is desired. Default: $2 * p + 1$ where p is the number of columns in x.

verbose: display progress. Can be either 0, 1 or 2. 0 will not display any progress, 2 will display very detailed progress and 1 is somewhere in between. Default: 1.

Details

The objective function for family="gaussian" is

$$
\frac{RSS}{2n} + \lambda (1 - \alpha) (w_e|\beta_e| + \sum w_j||\beta_j||_2) + \lambda \alpha (\sum w_{je}|\gamma_j|)
$$

where RSS is the residual sum of squares and n is the number of observations. See Bhatnagar et al. (2018+) for details.

It is highly recommended to specify center.x = TRUE and center.e = TRUE for both convergence and interpretation reasons. If centered, the final estimates can be interpreted as the effect of the predictor on the response while holding all other predictors at their mean value. For computing speed reasons, if models are not converging or running slow, consider increasing thresh, decreasing nl lambda, or increasing lambda.factor before increasing maxit. Then try increasing the value of alpha (which translates to more penalization on the interactions).

By default, sail uses the group lasso penalty on the basis expansions of x. To use the group MCP and group SCAD penalties (see Breheny and Huang 2015), the grpreg package must be installed.
Value

an object with S3 class "sail", "*", where "*" is "lspath" or "logitreg". Results are provided for converged values of lambda only.

call the call that produced this object

a0 intercept sequence of length nlambda

beta a (# main effects after basis expansion x nlambda) matrix of main effects coefficients, stored in sparse column format ("dgCMatrix")

alpha a (# interaction effects after basis expansion x nlambda) matrix of interaction effects coefficients, stored in sparse column format ("dgCMatrix")

gamma A p x nlambda matrix of (γ) coefficients, stored in sparse column format ("dgCMatrix")

bE exposure effect estimates of length nlambda

active list of length nlambda containing character vector of selected variables

lambda the actual sequence of lambda values used

lambda2 value for the mixing tuning parameter 0 < α < 1

dfbeta the number of nonzero main effect coefficients for each value of lambda

dfalpha the number of nonzero interaction coefficients for each value of lambda

dfenviron the number of nonzero exposure (e) coefficients for each value of lambda

dev.ratio the fraction of (null) deviance explained (for "lspath", this is the R-square). The deviance calculations incorporate weights if present in the model. The deviance is defined to be 2*(loglike_sat - loglike), where loglike_sat is the log-likelihood for the saturated model (a model with a free parameter per observation). Hence dev.ratio=1-dev/nulldev.

converged vector of logicals of length nlambda indicating if the algorithm converged

nlambda number of converged lambdas

design design matrix (X, E, X:E), of dimension n x (2*ncols*p+1) if expand=TRUE. This is used in the predict method.

nobs number of observations

nvars number of main effect variables

vnames character of variable names for main effects (without expansion)

ncols an integer of basis for each column of x if expand=TRUE, or an integer vector of basis for each variable if expand=FALSE

center.x were the columns of x (after expansion) centered?

center.e was e centered?

basis user defined basis expansion function

expand was the basis function applied to each column of x?

group a vector of consecutive integers describing the grouping of the coefficients. Only if expand=FALSE

interaction.names character vector of names of interaction variables

main.effect.names character vector of names of main effect variables (with expansion)
Author(s)
Sahir Bhatnagar
Maintainer: Sahir Bhatnagar <sahir.bhatnagar@gmail.com>

References


See Also
bs cv.sail

Examples
f.basis <- function(i) splines::bs(i, degree = 3)
# we specify dfmax to early stop the solution path to
# limit the execution time of the example
fit <- sail(x = sailsim$x, y = sailsim$y, e = sailsim$e,
  basis = f.basis, nlambda = 10, dfmax = 10,
  maxit = 100)

# estimated coefficients at each value of lambda
coeff(fit)

# predicted response at each value of lambda
predict(fit)

#predicted response at a specific value of lambda
predict(fit, s = 0.5)
# plot solution path for main effects and interactions
plot(fit)
# plot solution path only for main effects
plot(fit, type = "main")
# plot solution path only for interactions
plot(fit, type = "interaction")
Description

Internal sail helper functions

cbbPalette gives a Color Blind Palette
nonzero is to determine which coefficients are non-zero
check_col_0 is to check how many columns are 0 and is used in the fitting functions lspath

Usage

SoftThreshold(x, lambda)

l2norm(x)

cbbPalette

nonzero(beta, bystep = FALSE)

check_col_0(M)

Arguments

x numeric value of a coefficient
lambda tuning parameter value
beta vector or 1 column matrix of regression coefficients
bystep bystep = FALSE means which variables were ever nonzero, bystep = TRUE means which variables are nonzero for each step
M is a matrix

Format

An object of class character of length 8.

Details

These functions are not intended for use by users.
Description

A dataset containing simulated data used in the accompanying paper to this package

Usage

sailsim

Format

A list with 7 elements:

- `x` a matrix of dimension 100 x 20 where rows are observations and columns are predictors
- `y` a numeric response vector of length 100
- `e` a numeric exposure vector of length 100
- `f1, f2, f3, f4` the true functions

Details

The code used to simulate the data is available at [https://github.com/sahirbhatnagar/sail/blob/master/data-raw/SIMULATED_data.R](https://github.com/sahirbhatnagar/sail/blob/master/data-raw/SIMULATED_data.R). See `gendata` for more details. The true model is given by

\[ Y = f_1(X_1) + f_2(X_2) + f_3(X_3) + f_4(X_4) + E \ast (2 + f_3(X_3) + f_4(X_4)) \]

where

- \( f_1(t) = 5t \)
- \( f_2(t) = 3(2t - 1)^2 \)
- \( f_3(t) = \frac{4\sin(2\pi t)}{2 - \sin(2\pi t)} \)
- \( f_4(t) = 6(0.1\sin(2\pi t) + 0.2\cos(2\pi t) + 0.3\sin(2\pi t)^2 + 0.4\cos(2\pi t)^3 + 0.5\sin(2\pi t)^3) \)

References


Examples

sailsim
**standardize**  
*Standardize Data*

**Description**
Function that standardizes the data before running the fitting algorithm. This is used in the `sail` function.

**Usage**
```r
standardize(x, center = TRUE, normalize = FALSE)
```

**Arguments**
- `x` : data to be standardized
- `center` : Should `x` be centered. Default is `TRUE`
- `normalize` : Should `x` be scaled to have unit variance. Default is `FALSE`

**Value**
list of length 3:
- `x` : centered and possibly normalized `x` matrix
- `bx` : numeric vector of column means of `x` matrix
- `sx` : standard deviations (using a divisor of `n` observations) of columns of `x` matrix
Index

*Topic datasets
  oasis, 12
  sail-internal, 28
  sailsim, 29

bs, 7, 27

cbbPalette(sail-internal), 28
check_col_0(sail-internal), 28
coeff.cv.sail(predict.cv.sail), 18
coeff.sail, 16, 18
coeff.sail(predict.sail), 20
cor, 12
creafolds, 2
cv.lspath, 3
cv.sail, 3, 4, 5, 14, 15, 22, 27
cvcompute(cv.lspath), 3
design_sail, 7, 19, 20
gendata, 8, 11, 12, 29
gendataPaper, 11
getmin(cv.lspath), 3

l2norm(sail-internal), 28
lambda.interp(cv.lspath), 3

nonzero(sail-internal), 28

oasis, 12

persp, 15, 16
plot.cv.sail, 13
plot.sail, 14
plotInter, 15
plotMain, 17
predict.cv.sail, 18, 21
predict.sail, 16, 18, 19, 20
print.sail, 21

Q_theta, 22

registerDoParallel, 6
rnorm, 12
rug, 16, 18

sail, 5–7, 14, 15, 17, 22, 23, 30
sail-internal, 28
sailsim, 29
SoftThreshold(sail-internal), 28
standardize, 30