Package: sprintr (via r-universe)

September 4, 2024

Type Package

Title Sparse Reluctant Interaction Modeling

Version 0.9.1

Date 2019-08-08

Description An implementation of a computationally efficient method to
fit large-scale interaction models based on the reluctant
interaction selection principle. The method and its properties
are described in greater depth in Yu, G., Bien, J., and
Tibshirani, R.J. (2019) ``Reluctant interaction modeling'', which
is available at <arXiv:1907.08414>.

BugReports `https://github.com/hugogogo/sprintr/issues`

License GPL-3

Imports Rcpp (>= 0.12.16), glmnet

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://hugogogo.r-universe.dev

RemoteUrl https://github.com/hugogogo/sprintr

RemoteRef HEAD

RemoteSha 7a4120c862e455f922a4ff85ac561bb238475df1

Contents

`cv.sprinter` *Running sprinter with cross-validation*

Description

The main cross-validation function to select the best sprinter fit for a path of tuning parameters.

Usage

```
cv.sprinter(
  x,
  y,
  square = FALSE,
  num_keep = NULL,
  lambda1 = NULL,
  lambda3 = NULL,
  cv_step1 = FALSE,
  nlam1 = 10,
  nlam3 = 100,
  lam_min_ratio = ifelse(nrow(x) < ncol(x), 0.01, 1e-04),
  nfold = 5,
  foldid = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	An `n` by `p` design matrix of main effects. Each row is an observation of `p` main effects.
`y`	A response vector of size `n`.
`square`	Indicator of whether squared effects should be fitted in Step 1. Default to be FALSE.
`num_keep`	A user specified number of candidate interactions to keep in Step 2. If `num_keep` is not specified (as default), it will be set to `round[n / log n]`.
`lambda1`	Tuning parameter values for Step 1. `lambda1` is a vector. Default to be NULL, and the program will compute its own `lambda1` based on `nlam1` and `lam_min_ratio`.

`lambda3`	Tuning parameter values for Step 3. `lambda3` is a matrix, where the k-th column is the list of tuning parameter in Step 3 corresponding to Step 1 using `lambda1[k]`. Default to be NULL, and the program will compute its own `lambda3` based on `nlam3` and `lam_min_ratio`.
`cv_step1`	Indicator of whether cross-validation of `lambda1` should be carried out in Step 1 before subsequent steps. Default is `FALSE`.
`nlam1`	the number of values in `lambda1`. If not specified, they will be all set to `10`.
`nlam3`	the number of values in each column of `lambda3`. If not specified, they will be all set to `100`.
`lam_min_ratio`	The ratio of the smallest and the largest values in `lambda1` and each column of `lambda2`. The largest value is usually the smallest value for which all coefficients are set to zero. Default to be `1e-2` in the `n` < `p` setting.
`nfold`	Number of folds in cross-validation. Default value is 5. If each fold gets too view observation, a warning is thrown and the minimal `nfold = 3` is used.
`foldid`	A vector of length `n` representing which fold each observation belongs to. Default to be `NULL`, and the program will generate its own randomly.
`verbose`	If `TRUE`, a progress bar shows the progress of the fitting.
`...`	other arguments to be passed to the `glmnet` calls, such as `alpha` or `penalty.factor`

Value

An object of S3 class "`sprinter`".

`n` The sample size.

`p` The number of main effects.

`square` The `square` parameter passed into sprinter.

`a0_step3` Estimate of intercept corresponding to the CV-selected model.

`compact` A compact representation of the selected variables. `compact` has three columns, with the first two columns representing the indices of a selected variable (main effects with first index = 0), and the last column representing the estimate of coefficients.

`fit` The whole `glmnet` fit object.

`fitted` fitted value of response corresponding to the CV-selected model.

`num_keep` The value of `num_keep`.

`cvm` The averaged estimated prediction error on the test sets over K folds.

`cvse` The standard error of the estimated prediction error on the test sets over K folds.

`foldid` Fold assignment. A vector of length `n`.

`i_lambda1_best` The index in `lambda1` that is chosen by CV by minimizing cvm.

`i_lambda3_best` The index in `lambda3` that is chosen by CV by minimizing cvm.

`lambda1_best` The value of `lambda1` that is chosen by CV by minimizing cvm.

`lambda3_best` The value of `lambda3` that is chosen by CV by minimizing cvm.

`call` Function call.

See Also

predict.cv.sprinter

Examples

```
n <- 100
p <- 100
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
mod <- cv.sprinter(x = x, y = y)
```

`hier_lasso` *Two-stage hierarchical lasso*

Description

An implementation of the two-stage lasso studied in Hao et, al (2018).

Usage

```
hier_lasso(
  x,
  y,
  lambda = NULL,
  nlam = 100,
  lam_choice = "min",
  lam_min_ratio = ifelse(nrow(x) < ncol(x), 0.01, 1e-04),
  nfold = 5,
  foldid = NULL,
  ...
)
```

Arguments

`x`	An `n` by `p` design matrix of main effects. Each row is an observation of `p` main effects.
`y`	A response vector of size `n`.
`...`	other arguments to be passed to the `glmnet` calls, such as `alpha` or `penalty.factor`

Value

An object of S3 class "`cv.hier`".

`n` The sample size.

`p` The number of main effects.

`fit` The whole `cv.glmnet` fit object.

`compact` A compact representation of the selected variables. `compact` has three columns, with the first two columns representing the indices of a selected variable (main effects with first index = 0), and the last column representing the estimate of coefficients.

Examples

```
set.seed(123)
n <- 100
p <- 200
# dense input
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
mod <- hier_lasso(x = x, y = y)
```

`plot.cv.sprinter` *Plot function of cv.sprinter fit*

Description

This function produces plots of cross-validation for cv.sprinter.

Usage

```
## S3 method for class 'cv.sprinter'
plot(fit)
```

Arguments

`fit`	A "`cv.sprinter`" object.

Details

The orange pairs on the top of the plot shows the number of non-zero (main effects, interactions) selected by each value of lambda. Adopted from the function `plot.cv.rgam` from package `relgam` by Kenneth Tay and Robert Tibshirani.

See Also

`cv.sprinter`.

Examples

```
set.seed(123)
n <- 100
p <- 200
# dense input
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
```

```
mod <- cv.sprinter(x = x, y = y)

plot(mod)
```

---

plot.sprinter *Plot function of sprinter fit*

Description

Produces a two-panel plot of the sprinter object showing coefficient paths for both main effects and interactions.

Usage

```
## S3 method for class 'sprinter'
plot(fit, which = 1, label = TRUE, index = NULL)
```

Arguments

`fit`	Fitted `sprinter` object.
`which`	The tuning parameter considered in Step 2.
`label`	If `TRUE` (default), annotate the plot with variable labels.
`index`	Lambda indices to plot

Details

A two panel plot is produced, that summarizes the main effects (left) and interaction (right) coefficients, as a function of lambda. Adopted from the function `summary.rgam` from package `relgam` by Kenneth Tay and Robert Tibshirani.

Examples

```
set.seed(123)
n <- 100
p <- 100
# dense input
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
fit <- sprinter(x = x, y = y)

plot(fit)
```

predict.cv.sprinter *Calculate prediction from a* cv.sprinter *object.*

Description

Calculate prediction from a cv.sprinter object.

Usage

```
## S3 method for class 'cv.sprinter'
predict(object, newdata, ...)
```

Arguments

object	a fitted cv.sprinter object.
newdata	a design matrix of all the p main effects of some new observations of which predictions are to be made.
...	additional argument (not used here, only for S3 generic/method consistency)

Value

The prediction of newdata by the cv.sprinter fit object.

Examples

```
n <- 100
p <- 200
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] + 2 * x[, 2] - 3 * x[, 1] * x[, 2] + rnorm(n)
mod <- cv.sprinter(x = x, y = y)
fitted <- predict(mod, newdata = x)
```

predict.other *Calculate prediction from a* other *object.*

Description

Calculate prediction from a other object.

Usage

```
## S3 method for class 'other'
predict(object, newdata, ...)
```

Arguments

`object`	a fitted `other` object.
`newdata`	a design matrix of all the `p` main effects of some new observations of which predictions are to be made.
`...`	additional argument (not used here, only for S3 generic/method consistency)

Value

The prediction of `newdata` by the cv.sprinter fit `object`.

Examples

```
n <- 100
p <- 200
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] + 2 * x[, 2] - 3 * x[, 1] * x[, 2] + rnorm(n)
mod <- cv.sprinter(x = x, y = y)
fitted <- predict(mod, newdata = x)
```

`predict.sprinter` *Calculate prediction from a* `sprinter` *object.*

Description

Calculate prediction from a `sprinter` object.

Usage

```
## S3 method for class 'sprinter'
predict(object, newdata, ...)
```

Arguments

`object`	a fitted `sprinter` object.
`newdata`	a design matrix of all the `p` main effects of some new observations of which predictions are to be made.
`...`	additional argument (not used here, only for S3 generic/method consistency)

Value

The prediction of `newdata` by the sprinter fit `object`.

Examples

```
n <- 100
p <- 200
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] + 2 * x[, 2] - 3 * x[, 1] * x[, 2] + rnorm(n)
mod <- sprinter(x = x, y = y)
fitted <- predict(mod, newdata = x)
```

`print.cv.sprinter` *Print the cross validation information of cv.sprinter*

Description

Print a summary of the cross-validation information for running cv.sprinter.

Usage

```
## S3 method for class 'cv.sprinter'
print(fit, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`fit`	A fitted `cv.sprinter` object.
`digits`	Significant digits in printout.

Details

This function takes in a `cv.sprinter` object and produces summary of the cross-validation informationabout the tuning parameters (in Step 3) selected by `lambda.min` and `lambda.1se`. Adopted from the function `print.cv.rgam` from package `relgam` by Kenneth Tay and Robert Tibshirani.

See Also

`cv.sprinter`, `print.printer`.

Examples

```
set.seed(123)
n <- 100
p <- 100
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)

fit.cv <- cv.sprinter(x = x, y = y)
print(fit.cv)
```

`print.sprinter` *Print a summary of the sprinter fit*

Description

Print a summary of the sprinter fit at each step along the path of tuning parameters used in Step 3, for any given tuning parameter in Step 1.

Usage

```
## S3 method for class 'sprinter'
print(fit, which = 1, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`fit`	A `sprinter` object.
`which`	Which tuning parameter of Step 1 to print. Default is 1.
`digits`	Significant digits in printout.
`...`	Additional print arguments.

Details

The function produces a three-column matrix with tuning parameter values (in Step 3), number of nonzero main effects, and the number of nonzero interactions. Adopted from the function `print.rgam` from package `relgam` by Kenneth Tay and Robert Tibshirani.

See Also

`sprinter`.

Examples

```
set.seed(123)
n <- 100
p <- 100
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
fit <- sprinter(x = x, y = y)

print(fit, which = 3)
```

`screen_cpp` *Sure Independence Screening in Step 2*

Description

Sure Independence Screening in Step 2

Usage

```
screen_cpp(x, y, num_keep, square = FALSE, main_effect = FALSE)
```

Arguments

`x`	a n-by-p matrix of main effects, with i.i.d rows, and each row represents a vector of observations of p main-effects
`y`	a vector of length n. In sprinter, y is the residual from step 1
`num_keep`	the number of candidate interactions in Step 2. Default to be n / [log n]
`square`	An indicator of whether squared effects should be considered in Step 1 (NOT Step 2!). square == TRUE if squared effects have been considered in Step 1, i.e., squared effects will NOT be considered in Step 2.
`main_effect`	An indicator of whether main effects should also be screened. Default to be false. The functionality of main_effect = true is not used in sprinter, but for SIS_lasso.

Value

an matrix of 3 columns, representing the index pair of the selected interactions, and the corresponding absolute correlation with the residual.

`screen_sparse_cpp` *Sure Independence Screening in Step 2 for sparse design matrix*

Description

Sure Independence Screening in Step 2 for sparse design matrix

Usage

```
screen_sparse_cpp(x, y, num_keep, square = FALSE, main_effect = FALSE)
```

Arguments

x	a n-by-p sparse matrix of main effects
y	a vector of length n. In sprinter, y is the residual from step 1
num_keep	the number of candidate interactions in Step 2. Default to be n / [log n]
square	An indicator of whether squared effects should be considered in Step 1 (NOT Step 2!). square == TRUE if squared effects have been considered in Step 1, i.e., squared effects will NOT be considered in Step 2.
main_effect	An indicator of whether main effects should also be screened. Default to be false. The functionality of main_effect = true is not used in sprinter, but for SIS_lasso.

Value

an matrix of 3 columns, representing the index pair of the selected interactions, and the corresponding absolute correlation with the residual.

---

sis_lasso *Sure independence screening followed by lasso*

Description

Sure independence screening followed by lasso

Usage

```
sis_lasso(
  x,
  y,
  num_keep = NULL,
  lam_min_ratio = ifelse(nrow(x) < ncol(x), 0.01, 1e-04),
  nfold = 5,
  foldid = NULL,
  ...
)
```

Arguments

x	An n by p design matrix of main effects. Each row is an observation of p main effects.
y	A response vector of size n.
num_keep	Number of variables to keep in the screening phase
...	other arguments to be passed to the glmnet calls, such as alpha or penalty.factor

Value

An object of S3 class "`cv.hier`".

`n` The sample size.

`p` The number of main effects.

`fit` The whole `cv.glmnet` fit object.

`compact` A compact representation of the selected variables. `compact` has three columns, with the first two columns representing the indices of a selected variable (main effects with first index = 0), and the last column representing the estimate of coefficients.

Examples

```
set.seed(123)
n <- 100
p <- 200
# dense input
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
mod <- hier_lasso(x = x, y = y)
```

`sprinter` *Reluctant Interaction Modeling*

Description

This is the main function that fits interaction models with a path of tuning parameters (for Step 3).

Usage

```
sprinter(
  x,
  y,
  square = FALSE,
  num_keep = NULL,
  lambda1 = NULL,
  lambda3 = NULL,
  cv_step1 = FALSE,
  nlam1 = 10,
  nlam3 = 100,
  lam_min_ratio = ifelse(nrow(x) < ncol(x), 0.01, 1e-04),
  ...
)
```

Arguments

`x`	An `n` by `p` design matrix of main effects. Each row is an observation of `p` main effects.
`y`	A response vector of size `n`.
`square`	Indicator of whether squared effects should be fitted in Step 1. Default to be FALSE.
`num_keep`	A user specified number of candidate interactions to keep in Step 2. If `num_keep` is not specified (as default), it will be set to `round[n / log n]`.
`lambda1`	Tuning parameter values for Step 1. `lambda1` is a vector. Default to be NULL, and the program will compute its own `lambda1` based on `nlam1` and `lam_min_ratio`.
`lambda3`	Tuning parameter values for Step 3. `lambda3` is a matrix, where the k-th column is the list of tuning parameter in Step 3 corresponding to Step 1 using `lambda1[k]`. Default to be NULL, and the program will compute its own `lambda3` based on `nlam3` and `lam_min_ratio`.
`cv_step1`	Indicator of whether cross-validation of `lambda1` should be carried out in Step 1 before subsequent steps. Default is `FALSE`.
`nlam1`	the number of values in `lambda1`. If not specified, they will be all set to `10`.
`nlam3`	the number of values in each column of `lambda3`. If not specified, they will be all set to `100`.
`lam_min_ratio`	The ratio of the smallest and the largest values in `lambda1` and each column of `lambda2`. The largest value is usually the smallest value for which all coefficients are set to zero. Default to be `1e-2` in the `n` < `p` setting.
`...`	other arguments to be passed to the `glmnet` calls, such as `alpha` or `penalty.factor`

Value

An object of S3 class "`sprinter`".

`square` The `square` parameter passed into sprinter

`n` The number of observations in the dataset

`p` The number of main effects

`step1` The output from fitting Step 1

`lambda1` The path of tuning parameters passed into / computed for fitting Step 1

`step2` The output from the screening Step 2

`num_keep` The path of tuning parameters for Step 2

`step3` The output from fitting Step 3

`lambda3` The path of tuning parameters passed into / computed for fitting Step 3

`main_center` Column centers of the input main effects

`main_scale` Column scales of the input main effects

`call` Function call.

See Also

cv.sprinter

Examples

```
set.seed(123)
n <- 100
p <- 100
# dense input
x <- matrix(rnorm(n * p), n, p)
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
mod <- sprinter(x = x, y = y)

# sparse input
library(Matrix)
x <- Matrix::Matrix(0, n, p)
idx <- cbind(sample(seq(n), size = 10, replace = TRUE), sample(seq(p), size = 10, replace = TRUE))
x[idx] <- 1
y <- x[, 1] - 2 * x[, 2] + 3 * x[, 1] * x[, 3] - 4 * x[, 4] * x[, 5] + rnorm(n)
mod <- sprinter(x = x, y = y)
```

Index