Package 'intsel'

March 10, 2025

Type Package

Title Interaction Selection in Logistic Regression

Version 1.0

Date 2025-03-06

Description Logistic regression model with data-driven screening for significant two-way interactions.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 3.5.0)

Imports Rcpp (>= 1.0.12)

LinkingTo Rcpp

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.1

Copyright file inst/COPYRIGHTS

NeedsCompilation yes

Author Yi Lian [aut, cre],
Tianze Jiao [aut],
Guanbo Wang [aut],
Archer Y. Yang [aut],
Julien Mairal [ctb],
Yuansi Chen [ctb]

Maintainer Yi Lian <yi.lian@pennmedicine.upenn.edu>

Repository CRAN

Date/Publication 2025-03-10 14:50:26 UTC

Contents

`intsel` *Logistic regression with two-way interaction screening*

Description

Fit a logistic regression model including all the two-way interaction terms between the user-specified set of variables. The method uses an overlapping group lasso penalty that respects the commonly recognized selection rule, which is that, when the interaction term is selected into the model, both main effect terms should be in the model too. The regularization path is computed at a grid of values for the regularization parameter lambda.

Usage

```
intsel(
  x,
  y,
  intercept = TRUE,
  p.screen,
  lambda,
  par_init,
  stepsize_init = 1,
  stepsize_shrink = 0.8,
  tol = 1e-05,
  maxit = 1000L,
  verbose = FALSE
)
```

Arguments

`x`	Predictor matrix with dimension $n * p$, where n is the number of subjects, and p is the number of predictors.
`y`	Binary outcome, a vector of length n.
`intercept`	Logical, indicating whether an intercept term should be included in the model. The intercept term will not be penalized. The default is `TRUE`.
`p.screen`	Number of variables of which all two-way interactions are screened. These variables should be placed in the `p.screen` left-most columns of matrix `x`.
`lambda`	Sequence of regularization coefficients λ's. Will be sorted in a decreasing order.
`par_init`	Optional, vector of initial values of the optimization algorithm. Default initial value is zero for all p variables.
`stepsize_init`	Initial value of the stepsize of the optimization algorithm. Default is 1.0.

`stepsize_shrink`
: Factor in $(0, 1)$ by which the stepsize shrinks in the backtracking linesearch. Default is 0.8.

`tol`
: Convergence criterion. Algorithm stops when the l_2 norm of the parameter update is smaller than `tol`. Default is `1e-5`.

`maxit`
: Maximum number of iterations allowed. Default is `100L`.

`verbose`
: Logical, whether progress is printed. Default is `FALSE`.

Value

A list.

`lambdas`
: The user-specified regularization coefficients `lambda` sorted in decreasing order.

`estimates`
: A matrix, with each column corresponding to the coefficient estimates at each λ in `lambdas`.

`iterations`
: A vector of number of iterations it takes to converge at each λ in `lambdas`.

`x.original`
: The input matrix `x`.

`x`
: The predictor matrix with `x` plus `p.screen` * (`p.screen` - 1)/2 interaction terms.

`y`
: The input `y`.

`p.screen`
: The input `p.screen`.

`intercept`
: The input `intercept`.

Examples

```
n <- 1000
p.int <- 5
p.noint <- 3
intercept <- TRUE
p.screen <- 5

p.int.expand <- p.int*(p.int-1)/2
p.main <- p.int + p.noint
x <- matrix(rnorm(n * p.main), nrow = n, ncol = p.main)

# true model
# logit(p) = 0.1 + 0.3 x1 + 0.3 x2 + 0.3 x8 + 0.2 * x1 * x2

beta.true <- rep(0, p.main)
beta.true[c(1, 2, p.main)] <- 0.3
eta <- x %*% beta.true + 0.2 * x[, 1] * x[, 2]

if (intercept) eta <- eta + 0.1

py <- 1/(1 + exp(-eta))

y <- rbinom(n, 1, py)

nlam <- 30
```

```
lambdas <- exp(seq(log(0.1), log(0.00005), length.out = nlam))

# All the pairwise two-way interactions for the first p.screen variables
# are included in the model and screened in a data-driven manner.
fit <- intsel(x = x,
              y = y,
              p.screen = 5,
              intercept = intercept,
              lambda = lambdas)
fit$iterations
fit$estimates[, 1]
```

`intsel_cv` *Cross-validation for logistic regression with two-way interaction screening*

Description

Cross-validation function for `intsel`

Usage

```
intsel_cv(
  x,
  y,
  intercept = TRUE,
  p.screen,
  lambda,
  par_init,
  stepsize_init = 1,
  stepsize_shrink = 0.8,
  nfolds = 10,
  foldid = NULL,
  tol = 1e-05,
  maxit = 1000L,
  verbose = FALSE
)
```

Arguments

`x`	Predictor matrix with dimension $n * p$, where n is the number of subjects, and p is the number of predictors.
`y`	Binary outcome, a vector of length n.
`intercept`	Logical, indicating whether an intercept term should be included in the model. The intercept term will not be penalized. The default is `TRUE`.
`p.screen`	Number of variables of which all two-way interactions are screened. These variables should be placed in the `p.screen` left-most columns of matrix `x`.

lambda	Sequence of regularization coefficients λ's. Will be sorted in a decreasing order.
par_init	Optional, vector of initial values of the optimization algorithm. Default initial value is zero for all p variables.
stepsize_init	Initial value of the stepsize of the optimization algorithm. Default is 1.0.
stepsize_shrink	Factor in $(0, 1)$ by which the stepsize shrinks in the backtracking linesearch. Default is 0.8.
nfolds	Optional, the folds of cross-validation. Default is 10.
foldid	Optional, user-specified vector indicating the cross-validation fold in which each observation should be included. Values in this vector should range from 1 to nfolds. If left unspecified, intsel will randomly assign observations to folds
tol	Convergence criterion. Algorithm stops when the l_2 norm of the parameter update is smaller than tol. Default is 1e-5.
maxit	Maximum number of iterations allowed. Default is 100L.
verbose	Logical, whether progress is printed. Default is FALSE.

Value

A list.

lambdas	A vector of lambda used for each cross-validation.
cvm	The cv error averaged across all folds for each lambda.
cvsd	The standard error of the cv error for each lambda.
cvup	The cv error plus its standard error for each lambda.
cvlo	The cv error minus its standard error for each lambda.
nzero	The number of non-zero coefficients at each lambda.
intsel.fit	A fitted model for the full data at all lambdas of class "intsel".
lambda.min	The lambda such that the cvm reach its minimum.
lambda.1se	The maximum of lambda such that the cvm is less than the minimum the cvup (the minmum of cvm plus its standard error).
foldid	The fold assignments used.
index	A one column matrix with the indices of lambda.min and lambda.1se
.	
iterations	A vector of number of iterations it takes to converge at each λ in lambdas
.	
x.original	The input matrix x.
x	The predictor matrix with x plus p.screen * (p.screen - 1)/2 interaction terms.
y	The input y.
p.screen	The input p.screen.
intercept	The input intercept.

Examples

```
n <- 1000
p.int <- 5
p.noint <- 3
intercept <- TRUE
p.screen <- 5

p.int.expand <- p.int*(p.int-1)/2
p.main <- p.int + p.noint
x <- matrix(rnorm(n * p.main), nrow = n, ncol = p.main)

# true model
# logit(p) = 0.1 + 0.3 x1 + 0.3 x2 + 0.3 x8 + 0.2 * x1 * x2

beta.true <- rep(0, p.main)
beta.true[c(1, 2, p.main)] <- 0.3
eta <- x %*% beta.true + 0.2 * x[, 1] * x[, 2]

if (intercept) eta <- eta + 0.1

py <- 1/(1 + exp(-eta))

y <- rbinom(n, 1, py)

nlam <- 30
lambdas <- exp(seq(log(0.1), log(0.00005), length.out = nlam))

# All the pairwise two-way interactions for the first p.screen variables
# are included in the model and screened in a data-driven manner.
cv <- intsel_cv(x = x,
                y = y,
                p.screen =5,
                intercept = intercept,
                stepsize_init = 1,
                lambda = lambdas,
                nfolds = 5,
                foldid = NULL)
cv$index
```

---

plot.intsel *Solution path plot for* intsel()

Description

Plot the solution path generated by intsel().

Usage

```
## S3 method for class 'intsel'
plot(x, type = "l", log = "x", ...)
```

Arguments

x	Fitted intsel model.
type	Graphical argument to be passed to matplot(), a character string (length 1 vector) or vector of 1-character strings indicating the type of plot for each column of y, see plot.default for all possible types. Default is "l" for lines.
log	Graphical argument to be passed to matplot(), a character string which contains "x" if the x axis is to be logarithmic, "y" if the y axis is to be logarithmic, "" if neither, "xy" or "yx" if both axes are to be logarithmic. Default is "x".
...	Further arguments of matplot() and ultimately of plot.default() for some.

Value

Produces a coefficient profile plot of the coefficient paths for a fitted intsel model.

See Also

intsel, intsel_cv.

Examples

```
n <- 1000
p.int <- 5
p.noint <- 3
intercept <- TRUE
p.screen <- 5

p.int.expand <- p.int*(p.int-1)/2
p.main <- p.int + p.noint
x <- matrix(rnorm(n * p.main), nrow = n, ncol = p.main)

# true model
# logit(p) = 0.1 + 0.3 x1 + 0.3 x2 + 0.3 x8 + 0.2 * x1 * x2

beta.true <- rep(0, p.main)
beta.true[c(1, 2, p.main)] <- 0.3
eta <- x %*% beta.true + 0.2 * x[, 1] * x[, 2]

if (intercept) eta <- eta + 0.1

py <- 1/(1 + exp(-eta))

y <- rbinom(n, 1, py)

nlam <- 30
lambdas <- exp(seq(log(0.1), log(0.00005), length.out = nlam))

# All the pairwise two-way interactions for the first p.screen variables
# are included in the model and screened in a data-driven manner.
fit <- intsel(x = x,
              y = y,
```

```
              p.screen = 5,
              intercept = intercept,
              lambda = lambdas)
plot(fit)
```

`plot.intsel_cv` *Plots for* `intsel_cv`

Description

Plot the solution path or cross-validation curves produced by `intsel_cv()`.

Usage

```
## S3 method for class 'intsel_cv'
plot(x, type = "cv-curve", ...)
```

Arguments

`x`	The `intsel_cv` object.
`type`	Character string, `"solution-path"` to generate a solution path with marks at `lambda.min` and `lambda.1se`; `"cv-curve"` to generate a cross-validation curve.
`...`	Other graphical parameters to plot

Value

The `"solution-path"` plot produces a coefficient profile plot of the coefficient paths for a fitted `intsel` model. The `"cv-curve"` plot is the `cvm` (red dot) for each lambda with its standard error (vertical bar). The two vertical dashed lines corresponds to the `lambda.min` and `lambda.1se`.

See Also

`intsel`, `intsel_cv`.

Examples

```
n <- 1000
p.int <- 5
p.noint <- 3
intercept <- TRUE
p.screen <- 5

p.int.expand <- p.int*(p.int-1)/2
p.main <- p.int + p.noint
x <- matrix(rnorm(n * p.main), nrow = n, ncol = p.main)

# true model
```

```
# logit(p) = 0.1 + 0.3 x1 + 0.3 x2 + 0.3 x8 + 0.2 * x1 * x2

beta.true <- rep(0, p.main)
beta.true[c(1, 2, p.main)] <- 0.3
eta <- x %*% beta.true + 0.2 * x[, 1] * x[, 2]

if (intercept) eta <- eta + 0.1

py <- 1/(1 + exp(-eta))

y <- rbinom(n, 1, py)

nlam <- 30
lambdas <- exp(seq(log(0.1), log(0.00005), length.out = nlam))

# All the pairwise two-way interactions for the first p.screen variables
# are included in the model and screened in a data-driven manner.
cv <- intsel_cv(x = x,
                y = y,
                p.screen =5,
                intercept = intercept,
                stepsize_init = 1,
                lambda = lambdas,
                nfolds = 5,
                foldid = NULL)
plot(cv)
plot(cv, type = "solution-path")
```

`predict.intsel` *Predict Method for intsel fits*

Description

description Obtains predictions from a fitted `intsel` object

Usage

```
## S3 method for class 'intsel'
predict(object, newx, type = "link", ...)
```

Arguments

`object`	A fitted `intsel` object
`newx`	Optional, a matrix in which to look for variables with which to predict. If omitted, the original data is used.
`type`	The type of prediction required. The default "`link`" is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
`...`	Additional arguments passed to `predict`.

Value

A matrix containing the prediction.

Examples

```
n <- 1000
p.int <- 5
p.noint <- 3
intercept <- TRUE
p.screen <- 5

p.int.expand <- p.int*(p.int-1)/2
p.main <- p.int + p.noint
x <- matrix(rnorm(n * p.main), nrow = n, ncol = p.main)

# true model
# logit(p) = 0.1 + 0.3 x1 + 0.3 x2 + 0.3 x8 + 0.2 * x1 * x2

beta.true <- rep(0, p.main)
beta.true[c(1, 2, p.main)] <- 0.3
eta <- x %*% beta.true + 0.2 * x[, 1] * x[, 2]

if (intercept) eta <- eta + 0.1

py <- 1/(1 + exp(-eta))

y <- rbinom(n, 1, py)

nlam <- 30
lambdas <- exp(seq(log(0.1), log(0.00005), length.out = nlam))

# All the pairwise two-way interactions for the first p.screen variables
# are included in the model and screened in a data-driven manner.
fit <- intsel(x = x,
              y = y,
              p.screen = 5,
              intercept = intercept,
              lambda = lambdas)
str(predict(fit))
```

---

predict.intsel_cv *Predict Method for* intsel_cv

Description

description Obtains predictions from a fitted intsel_cv object

Usage

```
## S3 method for class 'intsel_cv'
predict(object, newx, type = "link", ...)
```

Arguments

object	A fitted intsel object
newx	Optional, a matrix in which to look for variables with which to predict. If omitted, the original data is used.
type	The type of prediction required. The default "link" is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable.
...	Additional arguments passed to predict.

Value

A matrix containing the prediction.

Examples

```
n <- 1000
p.int <- 5
p.noint <- 3
intercept <- TRUE
p.screen <- 5

p.int.expand <- p.int*(p.int-1)/2
p.main <- p.int + p.noint
x <- matrix(rnorm(n * p.main), nrow = n, ncol = p.main)

# true model
# logit(p) = 0.1 + 0.3 x1 + 0.3 x2 + 0.3 x8 + 0.2 * x1 * x2

beta.true <- rep(0, p.main)
beta.true[c(1, 2, p.main)] <- 0.3
eta <- x %*% beta.true + 0.2 * x[, 1] * x[, 2]

if (intercept) eta <- eta + 0.1

py <- 1/(1 + exp(-eta))

y <- rbinom(n, 1, py)

nlam <- 30
lambdas <- exp(seq(log(0.1), log(0.00005), length.out = nlam))

# All the pairwise two-way interactions for the first p.screen variables
# are included in the model and screened in a data-driven manner.
cv <- intsel_cv(x = x,
                y = y,
                p.screen =5,
                intercept = intercept,
                stepsize_init = 1,
                lambda = lambdas,
                nfolds = 5,
                foldid = NULL)
newx <- x[sample(1:nrow(x), size = 100), ]
```

```
pred.cv.newx <- predict(cv, newx = newx, type = "link")
dim(pred.cv.newx)
```

Index