Package 'RAMP'

December 18, 2016

Type Package

Title Regularized Generalized Linear Models with Interaction Effects

Version 1.9

Date 2016-12-17

Author Yang Feng, Ning Hao and Hao Helen Zhang

Maintainer Yang Feng <yang.feng@columbia.edu>

Description Provides an efficient procedure for fitting the entire solution path for high-dimensional regularized quadratic generalized linear models with interactions effects under the strong or weak heredity constraint.

License GPL-2

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-12-18 17:46:30

R topics documented:

`predict.RAMP` *Model prediction based on a fitted RAMP object.*

Description

Similar to the usual predict methods, this function returns predictions from a fitted `'RAMP'` object.

Usage

```
## S3 method for class 'RAMP'
predict(object, newdata = NULL, type = c("link", "response",
  "class"), allpath = FALSE, ...)
```

Arguments

object	Fitted 'RAMP' model object.
newdata	Matrix of new values for x at which predictions are to be made, without the intercept term.
type	Type of prediction required. Type 'response' gives the fitted values for 'gaussian', fitted probabilities for 'binomial', fitted mean for 'poisson', and the fitted relative risk for 'cox'. Type 'link' returns the linear predictors for 'binomial', 'poisson' and 'cox' models; for 'gaussian' models it is equivalent to type 'response'. Type 'class' applies only to 'binomial' models, and produces the class label corresponding to the maximum probability (0-1 labels).
allpath	allpath = T will output all the predictions on the solution path. allpath = FALSE will only output the one the criterion selected in the 'RAMP' object.
...	Not used. Other arguments to predict.

Value

The object returned depends on type.

See Also

RAMP,print.RAMP

---

print.RAMP *Result summary of a fitted RAMP object.*

Description

Similar to the usual print methods, this function summarize results from a fitted 'RAMP' object.

Usage

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

Arguments

x	Fitted 'RAMP' model object.
digits	The number of significant digits for the coefficient estimates.
...	Not used. Other arguments to predict.

Value

No value is returned.

See Also

RAMP

RAMP *Regularization Algorithm under Marginality Principle (RAMP) for high dimensional generalized quadratic regression.*

Description

Regularization Algorithm under Marginality Principle (RAMP) for high dimensional generalized quadratic regression.

Usage

```
RAMP(X, y, family = "gaussian", penalty = "LASSO", gamma = 3.7,
  inter = TRUE, hier = "Strong", eps = 1e-15, tune = "EBIC",
  penalty.factor = rep(1, ncol(X)), inter.penalty.factor = 1, lam.list,
  lambda.min.ratio, max.iter = 100, max.num, n.lambda = 100,
  ebic.gamma = 1, refit = TRUE, trace = FALSE)
```

Arguments

X	input matrix, of dimension nobs x nvars; each row is an observation vector.
y	response variable, of dimension nobs x 1. continous for family='gaussian', binary for family='binomial'.
family	response type. Default is 'gaussian'. The other choice is 'binomial' for logistic regression.
penalty	Choose from LASSO, SCAD and MCP. Default is 'LASSO'.
gamma	concavity parameter. If missing, the code will use 3.7 for 'SCAD' and 2.7 for 'MCP'.
inter	whether to select interaction effects. Default is TRUE.
hier	whether to enforce strong or weak heredity. Default is 'Strong'.
eps	the precision used to test the convergence. Default is 1e-15.
tune	tuning parameter selection method. 'AIC', 'BIC', 'EBIC' and 'GIC' are available options. Default is EBIC.
penalty.factor	A multiplicative factor for the penalty applied to each coefficient. If supplied, penalty.factor must be a numeric vector of length equal to the number of columns of X. The purpose of penalty.factor is to apply differential penalization if some coefficients are thought to be more likely than others to be in the model. In particular, penalty.factor can be 0, in which case the coefficient is always in the model without shrinkage.

`inter.penalty.factor`
: the penalty factor for interactions effects. Default is 1. larger value discourage interaction effects.

`lam.list`
: a user supplied λ sequence. typical usage is to have the program compute its own `lambda` sequence based on `lambda.min.ratio` and `n.lambda`. supplying a value of λ overrides this.

`lambda.min.ratio`
: optional input. smallest value for `lambda`, as a fraction of `max.lam`, the (data derived) entry value. the default depends on the sample size `n` relative to the number of variables `p`. if `n` > `p`, the default is 0.0001. otherwise, the default is 0.01.

`max.iter`
: maximum number of iteration in the computation. Default is 100.

`max.num`
: optional input. maximum number of nonzero coefficients.

`n.lambda`
: the number of `lambda` values. Default is 100.

`ebic.gamma`
: the gamma parameter value in the EBIC criteria. Default is 1.

`refit`
: whether to perform a MLE refit on the selected model. Default is TRUE.

`trace`
: whether to trace the fitting process. Default is FALSE.

Value

An object with S3 class RAMP.

`a0`
: intercept vector of length(`lambda`).

`mainInd`
: index for the selected main effects.

`interInd`
: index for the selected interaction effects

`beta.m`
: coefficients for the selected main effects.

`beta.i`
: coefficients for the selected interaction effects.

See Also

`predict.RAMP`,`print.RAMP`

Examples

```
set.seed(0)
n = 500
p = 10 #Can be changed to a much larger number say 50000
x = matrix(rnorm(n*p),n,p)
eta = 1 * x[,1] + 2 * x[,3]  + 3*x[,6]  + 4*x[,1]*x[,3] + 5*x[,1]*x[,6]
y =  eta + rnorm(n)
xtest = matrix(rnorm(n*p),n,p)
eta.test = 1 * xtest[,1] + 2 * xtest[,3]  + 3*xtest[,6] +
4*xtest[,1]*xtest[,3] + 5*xtest[,1]*xtest[,6]
ytest =  eta.test + rnorm(n)
fit1 = RAMP(x, y)
fit1    ###examine the results
ypred = predict(fit1, xtest)
```

```
mean((ypred-ytest)^2)

#fit1.scad = RAMP(x, y, penalty = 'SCAD')
#fit1.scad    ###examine the results

#fit1.mcp = RAMP(x, y, penalty = 'MCP')
#fit1.mcp    ###examine the results

##Now, try a binary response
#y = rbinom(n, 1, 1/(1+exp(-eta)))
#fit2 = RAMP(x, y, family='binomial')  ###for binary response

## Weak heredity
eta = 1 * x[,1] + 3*x[,6]  + 4*x[,1]*x[,3] + 5*x[,1]*x[,6]
y =  eta + rnorm(n)
eta.test = 1 * xtest[,1] +  3*xtest[,6] + 4*xtest[,1]*xtest[,3] +
5*xtest[,1]*xtest[,6]
ytest =  eta.test + rnorm(n)

fit3 = RAMP(x, y, hier = 'Strong')
fit3    ###examine the results
ypred3 = predict(fit3, xtest)
mean((ypred3-ytest)^2)
fit4 = RAMP(x, y, hier = 'Weak')
fit4
ypred4 = predict(fit4, xtest)
mean((ypred4-ytest)^2)
```

Index