Package ‘cvplogistic’

February 19, 2015

Version 3.1-0
Date 2013-03-17
Title Penalized Logistic Regression Model using Majorization Minimization by Coordinate Descent (MMCD) Algorithm
Author Dingfeng Jiang <dingfengjiang@gmail.com>
Maintainer Dingfeng Jiang <dingfengjiang@gmail.com>
Depends R (>= 2.9.0)
Description The package uses majorization minimization by coordinate descent (MMCD) algorithm to compute the solution surface for concave penalized logistic regression model. The SCAD and MCP (default) are two concave penalties considered in this implementation. For the MCP penalty, the package also provides the local linear approximation by coordinate descant (LLA-CD) and adaptive rescaling algorithms for computing the solutions. The package also provides a Lasso-concave hybrid penalty for fast variable selection. The hybrid penalty applies the concave penalty only to the variables selected by the Lasso. For all the implemented methods, the solution surface is computed along kappa, which is a more smooth fit for the logistic model. Tuning parameter selection method by k-fold cross-validated area under ROC curve (CV-AUC) is implemented as well.
License GPL (>= 2)
URL http://www.r-project.org
NeedsCompilation yes
Repository CRAN
Date/Publication 2013-03-18 10:29:02

R topics documented:

  cv.cvlogistic .................................................. 2
  cv.hybrid ...................................................... 4
  cvplogistic ................................................... 6
Tuning parameter selection by k-fold cross validation for concave penalized logistic model

Description

Using k-fold cross-validated area under ROC curve (CV-AUC) to select tuning parameter for high-dimensional logistic model with concave penalty

Usage

```r
cv.cvplogistic(y, x, penalty = "mcp", approach = "mmcd", nfold = 5,
               kappa = 1/2.7, nlambda = 100, lambda.min = 0.01,
               epsilon = 1e-3, maxit = 1e+3, seed = 1000)
```

Arguments

- `y`: response vector with elements 0 or 1.
- `x`: the design matrix of penalized variables. By default, an intercept vector will be added when fitting the model.
- `penalty`: a character specifying the penalty. One of "mcp" or "scad" should be specified, with "mcp" being the default.
- `approach`: a character specifying the numerical algorithm. One of "mmcd", "adaptive" or "llacda" can be specified, with "mmcd" being the default. See following details for more information.
- `nfold`: an integer value for k-fold cross validation.
- `kappa`: a value specifying the regulation parameter kappa. The proper range for kappa is [0, 1).
- `nlambda`: an integer value specifying the number of grids along the penalty parameter lambda.
- `lambda.min`: a value specifying how to determine the minimal value of penalty parameter lambda. We define lambda_min=lambda_max*lambda.min. We suggest lambda.min=0.0001 if n>p; 0.01 otherwise.
- `epsilon`: a value specifying the converge criterion of algorithm.
- `maxit`: an integer value specifying the maximum number of iterations for each coordinate.
- `seed`: randomization seed for cross validation.
The computation for logistic model with concave penalties is not easy. The MMCD package implements the majorization minimization by coordinate descent (MMCD) algorithm for computing the solution path for logistic model with SCAD or MCP penalties. The algorithm is very efficient and stable for high-dimensional data with \( p \gg n \). For MCP penalty, the package also implements the adaptive rescaling and the local linear approximation by coordinate descent algorithms (LLA-CDA) algorithms. For SCAD, only the MMCD algorithm is implemented.

The regularization parameter controls the concavity of the penalty, with larger value of kappa being more concave. When kappa=0, both the MCP and SCAD penalty become Lasso penalty. Hence if zero is specified for kappa, the algorithm returns Lasso solutions.

To select an appropriate tuning parameter for prediction, we use k-fold cross-validated area under ROC curve (CV-AUC) approach. The CV-AUC approach calculated the predictive AUC for each validation set by using the coefficients estimated from the corresponding training set. As the cross validation proceeds, the average predictive AUC is calculated. Then the CV-AUC approach chooses the lambda corresponding to the maximum average predictive AUC as the tuning parameter.

A list of three elements is returned.

- `scvauc`: the CV-AUC corresponding to the selected lambda.
- `slambda`: the selected lambda.
- `scoef`: the regression coefficients corresponding to the selected lambda, with the first element being the intercept.

Dingfeng Jiang


See Also

cvplogistic, hybrid.logistic, cv.hybrid, path.plot
Examples

```r
set.seed(100000)
n=100
y=rbinom(n,1,.4)
p=10
x=matrix(rnorm(n*p),n,p)

## MCP penalty by MMMCD algorithm
out=cv.cvplogistic(y, x, "mcp", "mmcd")
## MCP by adaptive rescaling algorithm
## out=cv.cvplogistic(y, x, "mcp", "adaptive")
## MCP by LLA-CD algorithm,
## out=cv.cvplogistic(y, x, "mcp", "llacd")
## SCAD penalty
## out=cv.cvplogistic(y, x, "scad")

## Lasso penalty
## out=cv.cvplogistic(y, x, kappa =0)
```

---

**cv.hybrid**  
*Tuning parameter selection by k-fold cross validation for logistic models with Lasso-concave hybrid penalty*

**Description**

Using k-fold cross-validated area under ROC curve to select tuning parameter for high-dimensional logistic model with Lasso-concave hybrid penalty

**Usage**

```r
cv.hybrid(y, x, penalty = "mcp", nfold = 5,
  kappa = 1/2.7, nlambda = 100, lambda.min = 0.01,
  epsilon = 1e-3, maxit = 1e+3, seed = 1000)
```

**Arguments**

- `y`  
  response vector with elements 0 or 1.

- `x`  
  the design matrix of penalized variables. By default, an intercept vector will be added when fitting the model.

- `penalty`  
  a character specifying the penalty. One of "mcp" or "scad" should be specified, with "mcp" being the default.

- `nfold`  
  an integer value for k-fold cross validation.

- `kappa`  
  a value specifying the regulation parameter kappa. The proper range for kappa is [0, 1).

- `nlambda`  
  an integer value specifying the number of grids along the penalty parameter lambda.
lambda_min a value specifying how to determine the minimal value of penalty parameter lambda. We define lambda_min=lambda_max*lambda.min. We suggest lambda.min=0.0001 if n>p; 0.01 otherwise.

epsilon a value specifying the converge criterion of algorithm.

maxit an integer value specifying the maximum number of iterations for each coordinate.

seed randomization seed for cross validation.

Details
A Lasso-concave hybrid penalty applies SCAD or MCP penalty only to the variables selected by Lasso. The idea is to use Lasso as a screen tool to filter variables, then apply the SCAD or MCP penalty to the variables selected by Lasso for further selection. The computation for the hybrid penalty is faster than the standard concave penalty. The risk of using the hybrid penalty is that the variable missed by Lasso penalty will also not selected by the SCAD/MCP penalty.

We also use the CV-AUC approach to select tuning parameter for models using the Lasso-concave hybrid penalty.

Value
A list of three elements is returned.

scvauc the CV-AUC corresponding to the selected lambda.

slambda the selected lambda.

scoef the regression coefficients corresponding to the selected lambda, with the first element being the intercept.

Author(s)
Dingfeng Jiang

References


See Also
cvplogistic, hybrid.logistic, cv.cvplogistic, path.plot
Examples

```r
set.seed(10000)
n=100
y=rbinom(n,1,0.4)
p=10
x=matrix(rnorm(n*p),n,p)

## Lasso-concave hybrid using MCP penalty
out=cv.hybrid(y, x, "mcp")
## Lasso-concave hybrid using SCAD penalty
## out=cv.hybrid(y, x, "scad")
```

cvplogistic

Majorization minimization by coordinate descent for concave penalized logistic regression

Description

Compute solution surface for a high-dimensional logistic regression model with concave penalty using MMCD, adaptive rescaling or LLA-CD algorithms

Usage

```r
cvplogistic(y, x, penalty = "mcp", approach = "mmcd", kappa = 1/2.7, nlambda = 100, lambda.min = 0.01, epsilon = 1e-3, maxit = 1e+3)
```

Arguments

- `y` response vector with elements 0 or 1.
- `x` the design matrix of penalized variables. By default, an intercept vector will be added when fitting the model.
- `penalty` a character specifying the penalty. One of "mcp" or "scad" should be specified, with "mcp" being the default.
- `approach` a character specifying the numerical algorithm. One of "mmcd", "adaptive" or "llacda" can be specified, with "mmcd" being the default. See following details for more information.
- `kappa` a value specifying the regulation parameter kappa. The proper range for kappa is [0, 1).
- `nlambda` a integer value specifying the number of grids along the penalty parameter lambda.
- `lambda.min` a value specifying how to determine the minimal value of penalty parameter lambda. We define lambda_min=lambda_max*lambda.min. We suggest lambda.min=0.0001 if n>p; 0.01 otherwise.
- `epsilon` a value specifying the converge criterion of algorithm.
- `maxit` an integer value specifying the maximum number of iterations for each coordinate.
The computation for logistic model with concave penalties is not easy. The MMCD package implements the majorization minimization by coordinate descent (MMCD) algorithm for computing the solution path for logistic model with SCAD or MCP penalties. The algorithm is very efficient and stable for high-dimensional data with p>n. For the MCP penalty, the package also implements the adaptive rescaling and the local linear approximation by coordinate descent algorithms (LLA-CDA) algorithms. For SCAD, only the MMCD algorithm is implemented.

The regularization parameter controls the concavity of the penalty, with larger value of kappa being more concave. When kappa=0, both the MCP and SCAD penalty become Lasso penalty. Hence if zero is specified for kappa, the algorithm returns Lasso solutions.

A list of two elements is returned.

- **coef**: a matrix of dimension (p+1)*nlambda, with p the number of variables (columns) in x. The 1st row is the intercept, which is added by default.
- **lambdas**: a vector of length nlambda for the penalty parameter lambda, ranging from the largest to the smallest.

**Author(s)**

Dingfeng Jiang

**References**


**See Also**

hybrid.logistic, cv.cvplogistic, cv.hybrid, path.plot

**Examples**

```r
set.seed(10000)
n=100
y=rbinom(n,1,0.4)
p=10
x=matrix(rnorm(n*p),n,p)

## MCP penalty by MMCD algorithm
```
hybrid.logistic

A Lasso-concave hybrid penalty for logistic regression

Description

Compute solution surface for a high-dimensional logistic regression model with Lasso-concave hybrid penalty for fast variable selection

Usage

hybrid.logistic(y, x, penalty = "mcp", kappa = 1/2.7, nlambda = 100, lambda.min = 0.01, epsilon = 1e-3, maxit = 1e+3)

Arguments

y response vector with elements 0 or 1.
x the design matrix of penalized variables. By default, an intercept vector will be added when fitting the model.
penalty a character specifying the penalty. One of "mcp" or "scad" should be specified, with "mcp" being the default.
kappa a value specifying the regulation parameter kappa. The proper range for kappa is [0, 1).
nlambda a integer value specifying the number of grids along the penalty parameter lambda.
lambda.min a value specifying how to determine the minimal value of penalty parameter lambda. We define lambda.min=lambda_max*lambda.min. We suggest lambda.min=0.0001 if n>p; 0.01 otherwise.
epsilon a value specifying the converge criterion of algorithm.
maxit an integer value specifying the maximum number of iterations for each coordinate.
Details

A Lasso-concave hybrid penalty applies SCAD or MCP penalty only to the variables selected by Lasso. The idea is to use Lasso as a screen tool to filter variables, then apply the SCAD or MCP penalty to the variables selected by Lasso for further selection. The computation for the hybrid penalty is faster than the standard concave penalty. The risk of using the hybrid penalty is that the variable missed by Lasso will also not selected by the SCAD/MCP penalty.

Value

A list of two elements is returned.

- **coef**: A matrix of dimension (p+1)*nlambda, with p the number of variables (columns) in x. The 1st row is the intercept, which is added by default.
- **lambdas**: A vector of length nlambda for the penalty parameter lambda, ranging from the largest to the smallest.

Author(s)

Dingfeng Jiang

References


See Also

cvplogistic, cv.cvplogistic, cv.hybrid, path.plot

Examples

```r
set.seed(10000)
n=100
y=rbinom(n,1,.4)
p=10
x=matrix(rnorm(n*p),n,p)

## Lasso-concave hybrid using MCP penalty
out=hybrid.logistic(y, x, "mcp")
## Lasso-concave hybrid using SCAD penalty
## out=hybrid.logistic(y, x, "scad")
```
Description

Plot the path trajectories for the solutions computed by the implemented methods.

Usage

\texttt{path.plot(out)}

Arguments

\texttt{out} the object return from function \texttt{cvplogistic} or \texttt{hybrid.logistic}.

Details

The function plots the trajectories of solutions, with \texttt{x-axis} being the grids of lambda, and \texttt{y-axis} being the coefficients profile.

Author(s)

Dingfeng Jiang

References


See Also

cvplogistic, hybrid.logistic, cv.hybrid, cv.cvplogistic

Examples

\begin{verbatim}
set.seed(10000)
n=100
y=rbinom(n,1,0.4)
p=10
x=matrix(rnorm(n*p),n,p)
## MCP
\end{verbatim}
out=cvplogistic(y, x)
path.plot(out)
## hybrid penalty
## out=hybrid.logistic(y, x, "mcp")
## path.plot(out)
Index

*Topic models
   cv.cvplogistic, 2
   cv.hybrid, 4
   cvplogistic, 6
   hybrid.logistic, 8
   path.plot, 10

*Topic regression
   cv.cvplogistic, 2
   cv.hybrid, 4
   cvplogistic, 6
   hybrid.logistic, 8
   path.plot, 10

   cv.cvplogistic, 2
   cv.hybrid, 4
   cvplogistic, 6

   hybrid.logistic, 8

   path.plot, 10