Package ‘stepPlr’

July 2, 2014

Version 0.92

Date 2009-12-17

Title L2 penalized logistic regression with a stepwise variable selection

Author Mee Young Park, Trevor Hastie

Maintainer Mee Young Park <meeyoung@google.com>

Depends R (>= 2.0)

Description L2 penalized logistic regression for both continuous and discrete predictors, with forward stagewise/forward stepwise variable selection procedure.

License GPL (>= 2)

Repository CRAN

Date/Publication 2010-07-26 13:59:19

NeedsCompilation yes

R topics documented:

cv.step.plr .......................................................... 2
plr ................................................................. 3
predict.plr .......................................................... 5
predict.stepplr ..................................................... 6
step.plr ............................................................ 7

Index 10
cv.step.plr  Computes cross-validated deviance or prediction errors for step.plr

Description
This function computes cross-validated deviance or prediction errors for step.plr. The parameters that can be cross-validated are lambda and cp.

Usage
.cv.step.plr(x, y, weights = rep(1, length(y)),
nfold = 5, folds = NULL, lambda = c(1e-4, 1e-2, 1),
cp = c("aic", "bic"), cv.type=c("deviance", "class"),
trace = TRUE, ...)

Arguments
x          matrix of features
y          binary response
weights    optional vector of weights for observations
nfold      number of folds to be used in cross-validation. Default is nfold=5.
folds      list of cross-validation folds. Its length must be nfold. If NULL, the folds are randomly generated.
lambda     vector of the candidate values for lambda in step.plr
cp         vector of the candidate values for cp in step.plr
cv.type    If cv.type=deviance, cross-validated deviances are returned. If cv.type=class, cross-validated prediction errors are returned.
trace      If TRUE, the steps are printed out.
...        other options for step.plr

Details
This function computes cross-validated deviance or prediction errors for step.plr. The parameters that can be cross-validated are lambda and cp. If both are input as vectors (of length greater than 1), then a two-dimensional cross-validation is done. If either one is input as a single value, then the cross-validation is done only on the parameter with multiple inputs.

Author(s)
Mee Young Park and Trevor Hastie

References
See Also

   step.plr

Examples

```r
n <- 100
p <- 5
x <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
y <- sample(c(0,1),n,replace=TRUE)
level <- vector("list",length=p)
for (i in 1:p) level[[i]] <- seq(3)
cvfit1 <- cv.step.plr(x,y,level=level,lambda=c(1e-4,1e-2,1),cp="bic")
cvfit2 <- cv.step.plr(x,y,level=level,lambda=1e-4,cp=c(2,3,4))
cvfit3 <- cv.step.plr(x,y,level=level,lambda=c(1e-4,1e-2,1),cp=c(2,3,4))
```

---

**plr**

*Logistic regression with a quadratic penalization on the coefficients*

**Description**

This function fits a logistic regression model penalizing the size of the L2 norm of the coefficients.

**Usage**

```r
plr(x, y, weights = rep(1,length(y)),
    offsetNsubset = NULL, offsetNcoefficients = NULL,
    lambda = 1e-4, cp = "bic")
```

**Arguments**

- **x**: matrix of features
- **y**: binary response
- **weights**: optional vector of weights for observations
- **offsetNsubset**: optional vector of indices for the predictors for which the coefficients are preset to **offsetNcoefficients**. If **offsetNcoefficients** is not **NULL**, **offsetNsubset** must be provided.
- **offsetNcoefficients**: optional vector of preset coefficient values for the predictors in **offsetNsubset**. If **offsetNcoefficients** is not **NULL**, **offsetNsubset** must be provided.
- **lambda**: regularization parameter for the L2 norm of the coefficients. The minimizing criterion in **plr** is -log-likelihood+\(\lambda \times ||\beta||^2\). Default is lambda=1e-4.
- **cp**: complexity parameter to be used when computing the score. score=deviance+cp*df. If cp="aic" or cp="bic", these are converted to cp=2 or cp=log(sample size), respectively. Default is cp="bic".
We proposed using logistic regression with a quadratic penalization on the coefficients for detecting gene interactions as described in "Penalized Logistic Regression for Detecting Gene Interactions (2008)" by Park and Hastie. However, this function plr may be used for a general purpose.

A plr object is returned. predict, print, and summary functions can be applied.

coefficients vector of the coefficient estimates
covariance sandwich estimate of the covariance matrix for the coefficients
deviance deviance of the fitted model
null.deviance deviance of the null model
df degrees of freedom of the fitted model
score deviance + cp*df
nobs number of observations
cp complexity parameter used when computing the score
fitted.values fitted probabilities
linear.predictors linear predictors computed with the estimated coefficients
level If any categorical factors are input, level - the list of level sets - is automatically generated and returned. See step.plr for details of how it is generated.

Mee Young Park and Trevor Hastie


predict.plr, step.plr

n <- 100
p <- 10
x <- matrix(rnorm(n*p), nrow=n)
y <- sample(c(0,1), n, replace=TRUE)
fit <- plr(x,y, lambda=1)

p <- 3
z <- matrix(sample(seq(3), n*p, replace=TRUE), nrow=n)
```r
x <- data.frame(x1=factor(z[,1]), x2=factor(z[,2]), x3=factor(z[,3]))
y <- sample(c(0,1), n, replace=TRUE)
fit <- plr(x, y, lambda=1)
# 'level' is automatically generated. Check 'fit$level'.
```

---

**predict.plr**

**prediction function for plr**

**Description**

This function computes the linear predictors, probability estimates, or the class labels for new data, using a plr object.

**Usage**

```r
predict.plr(object, newx = NULL,
            type = c("link", "response", "class"), ...)
```

**Arguments**

- **object**: plr object
- **newx**: matrix of features at which the predictions are made. If newx=NULL, predictions for the training data are returned.
- **type**: If type=link, the linear predictors are returned; if type=response, the probability estimates are returned; and if type=class, the class labels are returned. Default is type=link.
- **...**: other options for prediction

**Author(s)**

Mee Young Park and Trevor Hastie

**References**


**See Also**

plr
Examples

```r
n <- 100
p <- 10
x0 <- matrix(rnorm(n*p),nrow=n)
y <- sample(c(0,1),n,replace=TRUE)
fit <- plr(x0,y,lambda=1)
x1 <- matrix(rnorm(n*p),nrow=n)
pred1 <- predict(fit,x1,type="link")
pred2 <- predict(fit,x1,type="response")
pred3 <- predict(fit,x1,type="class")
```

```r
p <- 3
z <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
x0 <- data.frame(x1=factor(z[,1]),x2=factor(z[,2]),x3=factor(z[,3]))
y <- sample(c(0,1),n,replace=TRUE)
fit <- plr(x0,y,lambda=1)
z <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
x1 <- data.frame(x1=factor(z[,1]),x2=factor(z[,2]),x3=factor(z[,3]))
pred1 <- predict(fit,x1,type="link")
pred2 <- predict(fit,x1,type="response")
pred3 <- predict(fit,x1,type="class")
```

---

**predict.stepplr**  
*prediction function for step.plr*

### Description
This function computes the linear predictors, probability estimates, or the class labels for new data, using a `stepplr` object.

### Usage
```r
predict.stepplr(object, x = NULL, newx = NULL,
                type = c("link", "response", "class"), ...)
```

### Arguments
- **object**  
  `stepplr` object
- **x**  
  matrix of features used for fitting object. If `newx` is provided, `x` must be provided as well.
- **newx**  
  matrix of features at which the predictions are made. If `newx=NULL`, predictions for the training data are returned.
- **type**  
  If `type=link`, the linear predictors are returned; if `type=response`, the probability estimates are returned; and if `type=class`, the class labels are returned. Default is `type=link`.
- **...**  
  other options for prediction
Author(s)
Mee Young Park and Trevor Hastie

References

See Also
stepplr

Examples

n <- 100
p <- 5
x0 <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
x0 <- cbind(rnorm(n),x0)
y <- sample(c(0,1),n,replace=TRUE)
level <- vector("list",length=6)
for (i in 2:6) level[[i]] <- seq(3)
fit <- stepNplr(x,y,level=level)
x1 <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
x1 <- cbind(rnorm(n),x1)
pred1 <- predict(fit,x0,x1,type="link")
pred2 <- predict(fit,x0,x1,type="response")
pred3 <- predict(fit,x0,x1,type="class")
Arguments

- **x**: matrix of features
- **y**: binary response
- **weights**: optional vector of weights for observations
- **fix.subset**: vector of indices for the variables that are forced to be in the model
- **level**: list of length `ncol(x)`. The j-th element corresponds to the j-th column of x. If the j-th column of x is discrete, `level[[j]]` is the set of levels for the categorical factor. If the j-th column of x is continuous, `level[[j]] = NULL`. level is automatically generated in the function; however, if any levels of the categorical factors are not observed, but still need to be included in the model, then the user must provide the complete sets of the levels through level. If a numeric column needs to be considered discrete, it can be done by manually providing level as well.
- **lambda**: regularization parameter for the L2 norm of the coefficients. The minimizing criterion in `plr` is -log-likelihood+λ*||β||^2. Default is `lambda=1e-4`.
- **cp**: complexity parameter to be used when computing the score. score=deviance+cp*df. If `cp="aic"` or `cp="bic"`, these are converted to cp=2 or cp=log(sample size), respectively. Default is `cp="bic"`.
- **max.terms**: maximum number of terms to be added in the forward selection procedure. Default is `max.terms=5`.
- **type**: If `type="both"`, forward selection is followed by a backward deletion. If `type="forward"`, only a forward selection is done. If `type="forward.stagewise"`, variables are added in the forward-stagewise method. Default is "both".
- **trace**: If TRUE, the variable selection procedure prints out its progress.

Details

This function implements an L2 penalized logistic regression along with the stepwise variable selection procedure, as described in "Penalized Logistic Regression for Detecting Gene Interactions (2008)" by Park and Hastie.

If `type="forward"`, max.terms terms are sequentially added to the model, and the model that minimizes score is selected as the optimal fit. If `type="both"`, a backward deletion is done in addition, which provides a series of models with a different combination of the selected terms. The optimal model minimizing score is chosen from the second list.

Value

A `stepplr` object is returned. `anova`, `predict`, `print`, and `summary` functions can be applied.
score  deviance + cp*df, where df is the model degrees of freedom
group  vector of the counts for the dummy variables, to be used in predict.stepplr
y  response variable used
weight  weights used
fix.subset  fix.subset used
level  level used
lambda  lambda used
cp  complexity parameter used when computing the score
type  type used
xnames  column names of x

Author(s)
Mee Young Park and Trevor Hastie

References

See Also
cv.step.plr, plr, predict.stepplr

Examples
n <- 100
p <- 3
z <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
x <- data.frame(x1=factor(z[,1]),x2=factor(z[,2]),x3=factor(z[,3]))
y <- sample(c(0,1),n,replace=TRUE)
fit <- step.plr(x,y)
# 'level' is automatically generated. Check 'fit$level'.

p <- 5
x <- matrix(sample(seq(3),n*p,replace=TRUE),nrow=n)
x <- cbind(rnorm(n),x)
y <- sample(c(0,1),n,replace=TRUE)
level <- vector("list",length=6)
for (i in 2:6) level[[i]] <- seq(3)
fit1 <- step.plr(x,y,level=level,cp="aic")
fit2 <- step.plr(x,y,level=level,cp=4)
fit3 <- step.plr(x,y,level=level,type="forward")
fit4 <- step.plr(x,y,level=level,max.terms=10)
# This is an example in which 'level' was input manually.
# level[[1]] should be either 'NULL' or 'NA' since the first factor is continuous.
Index

*Topic models
  cv.step.plr, 2
  plr, 3
  predict.plr, 5
  predict.stepplr, 6
  step.plr, 7

*Topic regression
  cv.step.plr, 2
  plr, 3
  predict.plr, 5
  predict.stepplr, 6
  step.plr, 7

  cv.step.plr, 2
  plr, 3
  predict.plr, 5
  predict.stepplr, 6
  step.plr, 7