

Package ‘stepPlr’

April 19, 2009

Version 0.91

Date 2007-1-28

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 the forward stepwise variable selection procedure.

License GPL-2

OS_type unix

Repository CRAN

Date/Publication 2007-01-30 08:47:00

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

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

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

Mee Young Park and Trevor Hastie (2006) Penalized Logistic Regression for Detecting Gene Interactions - available at the authors' websites, <http://stat.stanford.edu/~mypark> or <http://stat.stanford.edu/~hastie/pub.htm>.

See Also

step.plr

Examples

```

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

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

Arguments

x	matrix of features
y	binary response
weights	an optional vector of weights for observations
lambda	regularization parameter for the L2 norm of the coefficients. The minimizing criterion in plr is $-\log\text{-likelihood} + \lambda * \ \beta\ ^2$. Default is <code>lambda=1e-4</code> .
cp	complexity parameter to be used when computing the score. <code>score=deviance+cp*df</code> . If <code>cp="aic"</code> or <code>cp="bic"</code> , these are converted to <code>cp=2</code> and <code>cp=log(sample size)</code> , respectively. Default is <code>cp="bic"</code> .

Details

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 (2006)" by Park and Hastie. However, this function plr may be used for a general purpose.

We thank Michael Saunders of SOL, Stanford University for providing the solver used for the convex optimization in this function.

Value

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

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

Author(s)

Mee Young Park and Trevor Hastie

References

Mee Young Park and Trevor Hastie (2006) Penalized Logistic Regression for Detecting Gene Interactions - available at the authors' websites, <http://stat.stanford.edu/~mypark> or <http://stat.stanford.edu/~hastie/pub.htm>.

See Also

`predict.plr`, `step.plr`

Examples

```
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)
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

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

Arguments

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

Author(s)

Mee Young Park and Trevor Hastie

References

Mee Young Park and Trevor Hastie (2006) Penalized Logistic Regression for Detecting Gene Interactions - available at the authors' websites, <http://stat.stanford.edu/~mypark> or <http://stat.stanford.edu/~hastie/pub.htm>.

See Also

`plr`

Examples

```
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")

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

```

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

```

Arguments

object	a stepplr object
x	the matrix of features used for fitting object. If newx is provided, x must be provided as well.
newx	a 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 the prediction

Author(s)

Mee Young Park and Trevor Hastie

References

Mee Young Park and Trevor Hastie (2006) Penalized Logistic Regression for Detecting Gene Interactions - available at the authors' websites, <http://stat.stanford.edu/~mypark> or <http://stat.stanford.edu/~hastie/pub.htm>.

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 <- step.plr(x0, 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")

```

step.plr

*Forward stepwise selection procedure for penalized logistic regression***Description**

This function fits a series of L2 penalized logistic regression models selecting variables through the forward stepwise selection procedure.

Usage

```

step.plr(x, y, weights = rep(1, length(y)), fix.subset = NULL,
         level = NULL, lambda = 1e-4, cp = "bic",
         max.terms = 5, type = c("both", "forward"), trace = FALSE)

```

Arguments

<code>x</code>	matrix of features
<code>y</code>	binary response
<code>weights</code>	an optional vector of weights for observations
<code>fix.subset</code>	a vector of indices for the variables that are forced to be in the model
<code>level</code>	a list of length <code>ncol(x)</code> . The <code>j</code> -th element corresponds to the <code>j</code> -th column of <code>x</code> . If the <code>j</code> -th column of <code>x</code> is discrete, <code>level[[j]]</code> is the set of levels for the categorical factor. If the <code>j</code> -th column of <code>x</code> is continuous, <code>level[[j]] = NULL</code> . <code>level</code> 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 <code>level</code> . If a numeric column needs to be considered discrete, it can be done by manually providing <code>level</code> as well.

<code>lambda</code>	regularization parameter for the L2 norm of the coefficients. The minimizing criterion in <code>plr</code> is $-\log\text{-likelihood} + \lambda * \ \beta\ ^2$. Default is <code>lambda=1e-4</code> .
<code>cp</code>	complexity parameter to be used when computing the score. <code>score=deviance+cp*df</code> . If <code>cp="aic"</code> or <code>cp="bic"</code> , these are converted to <code>cp=2</code> and <code>cp=log(sample size)</code> , respectively. Default is <code>cp="bic"</code> .
<code>max.terms</code>	the maximum number of terms to be added in the forward selection procedure. Default is <code>max.terms=5</code> .
<code>type</code>	If <code>type="both"</code> , the forward selection is followed by a backward deletion. If <code>type="forward"</code> , only a forward selection is done. Default is <code>"both"</code> .
<code>trace</code>	If <code>TRUE</code> , 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 (2006)" 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.

We thank Michael Saunders of SOL, Stanford University for providing the solver used for the convex optimization in this function.

Value

A `stepplr` object is returned. `anova`, `predict`, `print`, and `summary` functions can be applied.

<code>fit</code>	a <code>plr</code> object for the optimal model selected
<code>action</code>	a list that stores the selection order of the terms in the optimal model.
<code>action.name</code>	a list of the names of the sequentially added terms - in the same order as in <code>action</code>
<code>deviance</code>	deviance of the fitted model
<code>df</code>	residual degrees of freedom of the fitted model
<code>score</code>	deviance + <code>cp*df</code> , where <code>df</code> is the model degrees of freedom
<code>group</code>	a vector of the counts for the dummy variables, to be used in <code>predict.stepplr</code>
<code>y</code>	response variable used
<code>weight</code>	weights used
<code>fix.subset</code>	<code>fix.subset</code> used
<code>level</code>	level used
<code>lambda</code>	lambda used
<code>cp</code>	complexity parameter used when computing the score
<code>type</code>	type used
<code>xnames</code>	column names of <code>x</code>

Author(s)

Mee Young Park and Trevor Hastie

References

Mee Young Park and Trevor Hastie (2006) Penalized Logistic Regression for Detecting Gene Interactions - available at the authors' websites, <http://stat.stanford.edu/~mypark> or <http://stat.stanford.edu/~hastie/pub.htm>.

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, 1
- plr, 3
- predict.plr, 4
- predict.stepplr, 6
- step.plr, 7

*Topic **regression**

- cv.step.plr, 1
- plr, 3
- predict.plr, 4
- predict.stepplr, 6
- step.plr, 7

cv.step.plr, 1

plr, 3

predict.plr, 4

predict.stepplr, 6

step.plr, 7