Package 'msgps'

May 10, 2022

Type Package

Title Degrees of Freedom of Elastic Net, Adaptive Lasso and Generalized Elastic Net

Version 1.3.4

Date 2022-5-10

Author Kei Hirose

Maintainer Kei Hirose <mail@keihirose.com>

Description Computes the degrees of freedom of the lasso, elastic net, generalized elastic net and adaptive lasso based on the generalized path seeking algorithm. The optimal model can be selected by model selection criteria including Mallows' Cp, bias-corrected AIC (AICc), generalized cross validation (GCV) and BIC.

License GPL (>= 2)

URL https://keihirose.com/

Encoding UTF-8

Repository CRAN

NeedsCompilation yes

Date/Publication 2022-05-10 07:10:02 UTC

R topics documented:

msgps *msgps (Degrees of Freedom of Elastic Net, Adaptive Lasso and Generalized Elastic Net)*

Description

This package computes the degrees of freedom of the lasso, elastic net, generalized elastic net and adaptive lasso based on the generalized path seeking algorithm. The optimal model can be selected by model selection criteria including Mallows' Cp, bias-corrected AIC (AICc), generalized cross validation (GCV) and BIC.

Usage

```
msgps(X,y,penalty="enet", alpha=0, gamma=1, lambda=0.001, tau2, STEP=20000,
STEP.max=200000, DFtype="MODIFIED", p.max=300, intercept=TRUE, stand.coef=FALSE)
```

Arguments

X	predictor matrix
y	response vector
penalty	The penalty term. The "enet" indicates the elastic net: $$\alpha/2\|\|\beta\|\|_2^2 + (1-\alpha)\|\|\beta\|\|_1.$$ Note that alpha=0 is the lasso penalty. The "genet" is the generalized elastic net: $$log(\alpha + (1-\alpha)\|\|\beta\|\|_1).$$ The "alasso" is the adaptive lasso, which is a weighted version of the lasso given by $$w_i\|\|\beta\|\|_1,$$ where w_i is $1/(\hat{\beta}_i)^\gamma$. Here $\gamma > 0$ is a tuning parameter, and $\hat{\beta}_i$ is the ridge estimate with regularization parameter being $\lambda \geq 0$.
alpha	The value of α on "enet" and "genet" penalty.
gamma	The value of γ on "alasso".
lambda	The value of regularization parameter $\lambda \geq 0$ for ridge regression, which is used to calculate the weight vector of "alasso" penalty. Note that the ridge estimates can be ordinary least squared estimates when lambda=0.
tau2	Estimator of error variance for Mallows' Cp. The default is the unbiased estimator of error vairance of the most complex model. When the unbiased estimator of error vairance of the most complex model is not available (e.g., the number of variables exceeds the number of samples), tau2 is the variance of response vector.
STEP	The approximate number of steps.

STEP.max	The number of steps in this algorithm can often exceed STEP. When the number of steps exceeds STEP.max, this algorithm stops.
DFtype	"MODIFIED" or "NAIVE". The "MODIFIED" update is much more efficient thatn "NAIVE" update.
p.max	If the number of selected variables exceeds p.max, the algorithm stops.
intercept	When intercept is TRUE, the result of intercept is included.
stand.coef	When stand.coef is TRUE, the standardized coefficient is displayed.

Author(s)

Kei Hirose
<mail@keihirose.com>

References

Friedman, J. (2008). Fast sparse regression and classification. $Technicalreport$, Standford University.
Hirose, K., Tateishi, S. and Konishi, S.. (2011). Efficient algorithm to select tuning parameters in sparse regression modeling with regularization. arXiv:1109.2411 (arXiv).

See Also

coef.msgps, plot.msgps, predict.msgps and summary.msgos objects.

Examples

```
#data
X <- matrix(rnorm(100*8),100,8)
beta0 <- c(3,1.5,0,0,2,0,0,0)
epsilon <- rnorm(100,sd=3)
y <- X %*% beta0 + epsilon
y <- c(y)

#lasso
fit <- msgps(X,y)
summary(fit)
coef(fit) #extract coefficients at t selected by model selection criteria
coef(fit,c(0, 0.5, 2.5)) #extract coefficients at some values of t
predict(fit,X[1:10,]) #predict values at t selected by model selection criteria
predict(fit,X[1:10,],c(0, 0.5, 2.5)) #predict values at some values of t
plot(fit,criterion="cp") #plot the solution path with a model selected by Cp criterion

#elastic net
fit2 <- msgps(X,y,penalty="enet",alpha=0.5)
summary(fit2)

#generalized elastic net
fit3 <- msgps(X,y,penalty="genet",alpha=0.5)
summary(fit3)
```

```
#adaptive lasso
fit4 <- msgps(X,y,penalty="alasso",gamma=1,lambda=0)
summary(fit4)
```

`plot.msgps` *plot the solution path from a "msgps" object.*

Description

This functions predicts fitted values from a "msgps" object.

Usage

```
## S3 method for class 'msgps'
plot(x, criterion="cp", xvar="norm", yvar="coef", yvar.dflasso=TRUE,
stand.coef=TRUE, plot.step = 1000, col=TRUE,...)
```

Arguments

`x`	Fitted `"msgps"` model object.
`criterion`	The code `criterion` plots the value of tuning parameter of each criterion (`"cp"`, `"aicc"`, `"gcv"`, `"bic"`). The code `"none"` does not depict the tuning parameter.
`xvar`	The type of x variable. `"xvar=norm"` is max|beta|/|beta|, `"xvar=sum"` is max|beta|, `"xvar=step"` is the number of steps, and `"xvar=t"` is tuning parameter.
`yvar`	The type of y variable. `"yvar=coef"` is the standardized coefficients, and `"tvar=df"` is the degrees of freedom.
`yvar.dflasso`	For lasso penalty, the degrees of freedom of the lasso (the number of non-zero parameters) is given when `"yvar=df"` and `"yvar.dflasso=TRUE"`.
`stand.coef`	The standardized coefficients and tuning parameters are dipicted if "stand.coef=TRUE".
`plot.step`	The number of steps to plot the solution of df. As `plot.step` increases, the picture will be well-looking whereas the file size of the picture will increase.
`col`	The color option.
`...`	Other graphical parameters to plot

Value

The object returned depends on type.

Author(s)

Kei Hirose
<mail@keihirose.com>

See Also

`coef.msgps`, `predict.msgps` and `summary.msgps` objects.

Examples

```
#data
X <- matrix(rnorm(100*8),100,8)
beta0 <- c(3,1.5,0,0,2,0,0,0)
epsilon <- rnorm(100,sd=3)
y <- X %*% beta0 + epsilon
y <- c(y)

#fit
fit <- msgps(X,y)
plot(fit,criterion="cp") #plot the solution path with a model selected by Cp criterion
```

`predict.msgps` *make predictions from a "msgps" object.*

Description

This functions predicts fitted values via msgps function.

Usage

```
## S3 method for class 'msgps'
predict(object, X, tuning,...)
## S3 method for class 'msgps'
coef(object, tuning,...)
```

Arguments

`object`	Fitted `"msgps"` model object.
`X`	Matrix of vector of new input `x`.
`tuning`	Tuning parameter vector `t` where predictions are required. If `tuning` is missing, solutions selected by Cp, bias-corrected AIC (AICC), generalized cross validation (GCV) and BIC are displayed.
`...`	Other parameters

Value

The object returned depends on type.

Author(s)

Kei Hirose
<mail@keihirose.com>

Examples

```
#data
X <- matrix(rnorm(100*8),100,8)
beta0 <- c(3,1.5,0,0,2,0,0,0)
epsilon <- rnorm(100,sd=3)
y <- X %*% beta0 + epsilon
y <- c(y)

#fit
fit <- msgps(X,y)
coef(fit) #extract coefficients at t selected by model selection criteria
coef(fit,c(0, 0.5, 2.5)) #extract coefficients at some values of t
predict(fit,X[1:10,]) #predict values at t selected by model selection criteria
predict(fit,X[1:10,],c(0, 0.5, 2.5)) #predict values at some values of t
```

`summary.msgps` *A summary of "msgps" object..*

Description

This functions summarizes the "msgps" object.

Usage

```
## S3 method for class 'msgps'
summary(object, digits=max(3, getOption("digits") - 3), num.result = 20,
coef.result=100,...)
```

Arguments

`object`	Fitted `"msgps"` model object.
`digits`	The digits of the output.
`num.result`	The number of tuning parameter and the corresponding degrees of freedom displayed in this code.
`coef.result`	If the coef.result exceeds the number of variables, the result of coefficient is not described in this code.
`...`	Other parameters on summary

Value

`df`	The degrees of freedom for each tuning parameter.
`tuning.max`	Maximum value of tuning parameter.
`ms.coef`	The coefficient selected by each model selection criterion.
`ms.tuning`	The values of tuning parameter of models selected by each model selection criterion.
`ms.df`	The degerees of freedom selected of models each model selection criterion.

Author(s)

Kei Hirose
<mail@keihirose.com>

Examples

```
#data
X <- matrix(rnorm(100*8),100,8)
beta0 <- c(3,1.5,0,0,2,0,0,0)
epsilon <- rnorm(100,sd=3)
y <- X %*% beta0 + epsilon
y <- c(y)

#fit
fit <- msgps(X,y)
summary(fit)
```

Index