Package: ProxReg (via r-universe)

March 16, 2025

Type Package

Title Linear Models for Prediction and Classification using Proximal Operators

Version 0.1.1

Date 2025-02-27

Maintainer YingHong Chen <yinghongchen1402@gmail.com>

Description Implements optimization techniques for Lasso regression, R.Tibshirani(1996)<doi:10.1111/j.2517-6161.1996.tb02080.x> using Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) and Iterative Shrinkage-Thresholding Algorithm (ISTA) based on proximal operators, A.Beck(2009)<doi:10.1137/080716542>. The package is useful for high-dimensional regression problems and includes cross-validation procedures to select optimal penalty parameters.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Suggests knitr, rmarkdown,

VignetteBuilder knitr

Imports dplyr, EBImage, glmnet

NeedsCompilation no

Author YingHong Chen [aut, cre]

Date/Publication 2025-03-15 17:10:12 UTC

Config/pak/sysreqs libfftw3-dev make libjpeg-dev libpng-dev libtiff-dev

Repository https://doufu1402.r-universe.dev

RemoteUrl https://github.com/cran/ProxReg

RemoteRef HEAD

RemoteSha 9300ec7243c77fe434ebef8b6b3c050dbdb87add

Contents

`delete_rect` *rectangular hole in image*

Description

creates a rectangular hole in the image with the specified dimensions

Usage

```
delete_rect(image,i,j,width,height)
```

Arguments

`image`	image to be modified, it has to be a 3D array proceed with readImage function from EBImage package
`i`	row index of the upper left corner of the rectangle
`j`	column index of the upper left corner of the rectangle
`width`	width of the rectangle
`height`	height of the rectangle

Details

delete_rect

Value

a 3D array with pixels in the hole set to -100 and the rest of the image pixels unchanged

Examples

```
image<-EBImage::readImage(system.file("extdata", "bird.jpg", package = "ProxReg"))
image_noise<-delete_rect(image,160,160,20,20)
image_noise<-EBImage::Image(image_noise,colormode = "Color")
EBImage::display(image_noise)
```

`inpainting` *image recovery using Lasso regression*

Description

predicts the missing pixels in an image using Lasso regression and fills the hole in the image

Usage

```
inpainting(image,h,stride,i,j,width,height,lambda=0.1,max_iter=50000,
fista=TRUE, verbose=TRUE,ini=0,glmnet=TRUE,noise=TRUE)
```

Arguments

`image`	image to be modified, it has to be a 3D array proceed with readImage function from EBImage package
`h`	size of the patch
`stride`	stride for the patch
`i`	row index of the upper left corner of the rectangle
`j`	column index of the upper left corner of the rectangle
`width`	width of the rectangle
`height`	height of the rectangle
`lambda`	a penalized parameter for the Lasso regression, it is 0.1 by default
`max_iter`	maximum number of iterations, it is 50000 by default
`fista`	fista=TRUE: use FISTA algortihm for the pixel prediction
`verbose`	print the iteration number and the size of the boundary
`ini`	initial value for the coefficients, default is 0
`glmnet`	use glmnet package for the Lasso regression
`noise`	display the image with the hole, it is TRUE by default

Details

inpainting

Value

a 3D array with the hole filled by pixels predicted by Lasso regression

Examples

```
test_img <- EBImage::readImage(system.file("extdata", "bird.jpg", package = "ProxReg"))
image_repaired <- inpainting(
  test_img, h = 10, stride = 6, i = 160, j = 160, width = 20, height = 20,
  lambda = 0.001, max_iter = 1000, verbose = TRUE, glmnet = TRUE,noise=TRUE)
RGB_repaired<-EBImage::Image(image_repaired,colormode = "Color")
```

---

k_fold_cross *k_fold_cross*

Description

k_fold_cross splits the dataset into k parts, and uses k-1 parts to train the model and the remaining part to test the model.

Usage

```
k_fold_cross(data,k)
```

Arguments

data	dataset which will be used for K-Fols Cross Validation
k	integer

Value

a list with two sublists: training set and test set

Examples

```
df = data.frame("hours"=c(1, 2, 4, 5, 5, 6, 6, 7, 8, 10, 11, 11, 12, 12, 14),
"score"=c(64, 66, 76, 73, 74, 81, 83, 82, 80, 88, 84, 82, 91, 93, 89))
k_fold_cross(df,k=2)
```

---

lasso_fista *Lasso regression with fixed step with FISTA algorithm*

Description

the function carries out the Lasso regression using fixed step using FISTA algorithm.

Usage

```
lasso_fista(data,y,x,lambda,max_step=10000,type="Gaussian",image=TRUE,ini=0.5,tol=10^-7)
```

Arguments

data	name of the dataset
y	name of the dependent variables
x	name of the independent variable
lambda	a vector of lambda-value to be evaluated in the regression
max_step	maximum number of steps
type	type of response variable, by default, it is 'Gaussian' for continuos response and can be modified as 'Binomial' for binary response
image	logical, if TRUE, the evolution of errors in term of lambda values will be plotted
ini	initial value for the coefficients
tol	tolerance for convergence, it is 10^-7 by default

Details

lasso_fista

Value

A list containing:

- coefficients: A matrix where each column represents the estimated regression coefficients for a different lambda value.
- error_evolution: A numeric vector tracking the error at certain step.
- num_steps: An integer vector indicating the number of steps in which errors are calculated.

Examples

```
library("glmnet")
data("QuickStartExample")
test<-as.data.frame(cbind(QuickStartExample$y,QuickStartExample$x))
lasso_fista(test,"V1",colnames(test)[2:21],lambda=0.1,image=TRUE,max_step=1000)
```

---

lasso_fista_back *Lasso regression with backtraking line research with FISTA algorithm*

Description

the function carries out the Lasso regression using backtraking line research and FISTA algorithm.

Usage

```
lasso_fista_back(data,y,x,lambda,max_step=10000,tol=10^-7,
type="Gaussian",ini=0.5,image=TRUE)
```

Arguments

`data`	name of the dataset
`y`	name of the dependent variables
`x`	name of the independent variable
`lambda`	a vector of lambda-value to be evaluated in the regression
`max_step`	maximum number of steps
`tol`	tolerance for convergence, it is 10^-7 by default
`type`	type of response variable, by default, it is 'Gaussian' for continuos response and can be modified as 'Binomial' for binary response
`ini`	initial value for the coefficients, default is 0.5
`image`	plots the evolution of errors in term of lambda values

Details

lasso_fista_back

Value

A list containing:

- `coefficients`: A matrix where each column represents the estimated regression coefficients for a different lambda value.
- `error_evolution`: A numeric vector tracking the error at certain step.
- `num_steps`: An integer vector indicating the number of steps in which errors are calculated.

Examples

```
library("glmnet")
data("QuickStartExample")
test<-as.data.frame(cbind(QuickStartExample$y,QuickStartExample$x))
lasso_fista_back(test,"V1",colnames(test)[2:21],lambda=0.1,image=TRUE,type='Gaussian',max_step=1000)
```

`lasso_ista` *Lasso regression with fixed step with ISTA algorithm*

Description

the function carries out the Lasso regression using fixed step using ISTA algorithm.

Usage

```
lasso_ista(data,y,x,lambda,max_step=10000,type="Gaussian",image=TRUE,tol=10^-7,ini=0.5)
```

Arguments

`data`	name of the dataset
`y`	name of the dependent variables
`x`	name of the independent variable
`lambda`	a vector of lambda-value to be evaluated in the regression
`max_step`	maximum number of steps
`type`	type of response variable, by default, it is 'Gaussian' for continuos response and can be modified as 'Binomial' for binary response
`image`	logical, if TRUE, the evolution of errors in term of lambda values will be plotted
`tol`	tolerance for convergence, it is 10^-7 by default
`ini`	initial value for the coefficients

Details

lasso_ista

Value

A list containing:

- `coefficients`: A matrix where each column represents the estimated regression coefficients for a different lambda value.
- `error_evolution`: A numeric vector tracking the error at certain step.
- `num_steps`: An integer vector indicating the number of steps in which errors are calculated.

Examples

```
library("glmnet")
data("QuickStartExample")
test<-as.data.frame(cbind(QuickStartExample$y,QuickStartExample$x))
lasso_ista(test,"V1",colnames(test)[2:21],lambda=0.1,image=TRUE,max_step=1000)
```

`lasso_ista_back` *Lasso regression with backtraking line research*

Description

the function carries out the Lasso regression using backtraking line research and ISTA algorithm.

Usage

```
lasso_ista_back(data,y,x,lambda,max_step=10000,tol=10^-7,
type="Gaussian",ini=0.5,image=TRUE)
```

Arguments

`data`	name of the dataset
`y`	name of the dependent variables
`x`	name of the independent variable
`lambda`	a vector of lambda-value to be evaluated in the regression
`max_step`	maximum number of steps
`tol`	tolerance for convergence, it is 10^-7 by default
`type`	type of response variable, by default, it is 'Gaussian' for continuos response and can be modified as 'Binomial' for binary response
`ini`	initial value for the coefficients,dafault is 0.5
`image`	plots the evolution of errors in term of lambda values

Details

lasso_ista_back

Value

A list containing:

- `coefficients`: A matrix where each column represents the estimated regression coefficients for a different lambda value.
- `error_evolution`: A numeric vector tracking the error at certain step.
- `num_steps`: An integer vector indicating the number of steps in which errors are calculated.

Examples

```
library("glmnet")
data("QuickStartExample")
test<-as.data.frame(cbind(QuickStartExample$y,QuickStartExample$x))
lasso_ista_back(test,"V1",colnames(test)[2:21],lambda=0.1,image=TRUE,type='Gaussian',max_step=100)
```

`lasso_multi` *Lasso logistic regression for multinomial response variable with fixed step*

Description

the function realizes L1-regularized classification for multinomial response variable using ISTA / FISTA algorithm

Usage

```
lasso_multi(data,y,x,lambda,max_step=10000,image=FALSE,fista=TRUE)
```

Arguments

`data`	name of the dataset
`y`	name of the dependent variables
`x`	name of the independent variable
`lambda`	a number or a vector of lambda-value to be evaluated in the regression
`max_step`	maximum number of steps
`image`	plots the evolution of errors in term of lambda values
`fista`	fista=TRUE: use FISTA algortihm for the multiclass logistic regression; fista=FALSE: use ISTA algortihm

Details

lasso_multi

Value

A list containing:

- `coefficients`: A matrix where each column represents the estimated regression coefficients for a different lambda value.
- `error_evolution`: A numeric vector tracking the error at certain step.
- `num_steps`: An integer vector indicating the number of steps in which errors are calculated.

Examples

```
library(glmnet)
data("MultinomialExample")
x<-MultinomialExample$x
y<-MultinomialExample$y
mult<-as.data.frame(cbind(x,y))
lasso_multi(mult,y="y",x=colnames(mult)[-31],max_step = 1000,lambda=0.01,image=TRUE,fista=TRUE)
```

`lasso_multi_back` *Lasso regression with backtraking line research for multinomial response variable*

Description

the function carries out the Lasso regression for multinomial response using backtraking line research and FISTA/ISTA algorithm.

Usage

```
lasso_multi_back(data,y,x,lambda,max_step=10000,image=FALSE,fista=TRUE,tol=10^-7,ini=0)
```

Arguments

`data`	name of the dataset
`y`	name of the dependent variables
`x`	name of the independent variable
`lambda`	a vector of lambda-value to be evaluated in the regression
`max_step`	maximum number of steps
`image`	plots the evolution of errors in term of lambda values
`fista`	fista=TRUE: use FISTA algortihm for the multiclass logistic regression; fista=FALSE: use ISTA algortihm
`tol`	tolerance for the convergence
`ini`	initial value for the coefficients, default is 0 #' @examples library(glmnet) data("MultinomialExample") x<-MultinomialExample$x y<-MultinomialExample$y mult<-as.data.frame(cbind(x,y)) lasso_multi_back(mult,y="y",x=colnames(mult)[-31],max_step = 1000,lambda=0.01,image=TRUE,fista=

Details

lasso_multi_back

Value

A list containing:

- `coefficients`: A matrix where each column represents the estimated regression coefficients for a different lambda value.
- `error_evolution`: A numeric vector tracking the error at certain step.
- `num_steps`: An integer vector indicating the number of steps in which errors are calculated.

`l_CV` *K-Fold Cross validation for L1/L2 regression*

Description

the function realizes K-Fold Cross validation for ridge/Lasso regression to help to choose the lambda that minimise the RSS

Usage

```
l_CV(data,y,x,lambda,k,mode=2,binary=FALSE,step=1000,bound=0.5,fista=TRUE,tol=10^-7)
```

Arguments

`data`	name of the dataset
`y`	name of the dependent variables
`x`	name of the independent variable
`lambda`	a number or a vector of lambda-value to be evaluated in the regression
`k`	integer, which indicates how many training and test set will be splited from the dataset
`mode`	1: ridge regression; 2: lasso regression
`binary`	logical, if TRUE, the dependent variable is binary
`step`	maximum number of steps
`bound`	threshold for binary dependent variable
`fista`	logical, if TRUE, the FISTA algorithm is used
`tol`	tolerance for convergence, it is 10^-7 by default

Value

the lambda values that minimize the MSE

Examples

```
l_CV(mtcars,"hp",c("mpg","qsec","disp"),c(0.01,0.1),k=5,mode=2)
```

`ols` *Ordinary Least Square regression*

Description

This is a function that estimates coefficients for a linear model using Ordinary Least Squares (OLS) regression.

Usage

```
ols(data,y,x,alpha=0.025,verbose=TRUE)
```

Arguments

`data`	Dataset used to estimated the coefficients
`y`	name of the dependent variable
`x`	name or a vector of names of the independent variables
`alpha`	confedence level
`verbose`	logical, if TRUE, the table will be printed

Value

coefficients of the linear model, or a table with the coefficients, standard errors, t-values, p-values and confidence intervals

Examples

```
df = data.frame("hours"=c(1, 2, 4, 5, 5, 6, 6, 7, 8, 10, 11, 11, 12, 12, 14),
"score"=c(64, 66, 76, 73, 74, 81, 83, 82, 80, 88, 84, 82, 91, 93, 89))
ols(df,"score","hours")
```

`ols_KCV` *K-Fold Cross Validation for OLS*

Description

ols_KCV makes the K-Fold Cross Validation for ordinary least squared regression

Usage

```
ols_KCV(data,k,y,x)
```

Arguments

data	full dataset which will be used for KCV
k	integer, which indicates how many training and test set will be splited from the dataset
y	dependent variable
x	independent variables

Value

the root mean square error after K-Fold Cross Validation on training set

Examples

```
df<-mtcars
ols_KCV(mtcars,5,"hp",c("mpg","qsec","disp"))
```

`ridge` *Ridge regression*

Description

ridge function estimates the coefficients for a linear model using Ridge regression.

Usage

```
ridge(data,y,x,lambda)
```

Arguments

`data`	name of the dataset
`y`	name of dependent variables
`x`	name of independent variable
`lambda`	a numeric value or a numeric vector to penalize the squared residual

Value

a matrix with the coefficients for each lambda

Examples

```
ridge(mtcars,"hp",c("mpg","qsec","disp"),c(0.01,0.1))
```

`softmax` *Softmax function for multinomial response variable*

Description

the function calculates the softmax function for the multinomial response variable

Usage

```
softmax(num)
```

Arguments

`num`	A numeric matrix or vector

Details

softmax

Value

A numeric matrix or vector of the same shape as num, where each element represents a probability value between 0 and 1. The values sum to 1 across each row or the entire vector.

Index