Package ‘PRIMAL’

January 22, 2020

Type  Package
Title  Parametric Simplex Method for Sparse Learning
Version  1.0.2
Date  2020-01-21
Author  Zichong Li, Qianli Shen
Maintainer  Zichong Li <zichongli5@gmail.com>
LinkingTo  Rcpp, RcppEigen
Description  Implements a unified framework of parametric simplex method for a variety of sparse learning problems (e.g., Dantzig selector (for linear regression), sparse quantile regression, sparse support vector machines, and compressive sensing) combined with efficient hyper-parameter selection strategies. The core algorithm is implemented in C++ with Eigen3 support for portable high performance linear algebra. For more details about parametric simplex method, see Haotian Pang (2017) <https://papers.nips.cc/paper/6623-parametric-simplex-method-for-sparse-learning.pdf>.

Imports  Matrix
License  GPL (>= 2)
NeedsCompilation  yes
RoxygenNote  6.1.1
Repository  CRAN
Date/Publication  2020-01-22 11:10:02 UTC

R topics documented:

- PRIMAL-package ......................................................... 2
- coef.primal .......................................................... 2
- CompressedSensing_solver ........................................... 3
- Dantzig_solver ......................................................... 4
- plot.primal ............................................................ 6
- print.primal ........................................................... 6
- PSM_solver ............................................................. 7
- QuantileRegression_solver .......................................... 8
- SparseSVM_solver ..................................................... 10
Description

A package for parametric simplex method for sparse learning

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>PRIMAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.0.0</td>
</tr>
<tr>
<td>Date:</td>
<td>2019-08-15</td>
</tr>
</tbody>
</table>

The package "PRIMAL" provides 5 main functions:
1) The dantzig selector solver applying simplex method. Please refer to `Dantzig_solver`.
2) The sparse SVM solver applying simplex method. Please refer to `SparseSVM_solver`.
3) The compressed sensing solver. Please refer to `CompressedSensing_solver`.
4) The quantile regression solver. Please refer to `QuantileRegression_solver`.
5) The solver for standard formulation of parametric simplex method. Please refer to `PSM_solver`.

Author(s)

Qianli Shen, Zichong Li

See Also

`plot.primal`, `print.primal`, `coef.primal`

Description

Print the estimated solution correspond to a specific parameter.

Usage

```r
## S3 method for class 'primal'
coef(object, n = NULL, ...)
```
CompressedSensing_solver

Arguments

object An object with S3 class "primal".

n The index of the wanted parameter.

... System reserved (No specific usage)

See Also

Dantzig_solver, SparseSVM_solver

CompressedSensing_solver

Solve given compressed sensing problem in parametric simplex method

Description

Solve given compressed sensing problem in parametric simplex method

Usage

CompressedSensing_solver(X, y, max_it = 50, lambda_threshold = 0.01)

Arguments

X x is an n by d data matrix

y y is a length n response vector

max_it This is the number of the maximum path length one would like to achieve. The
default length is 50.

lambda_threshold The parametric simplex method will stop when the calculated parameter is smaller
than lambda. The default value is 0.01.

Value

An object with S3 class "primal" is returned:

data The n by d data matrix from the input

response The length n response vector from the input

beta A matrix of regression estimates whose columns correspond to regularization
parameters for parametric simplex method.

df The degree of freedom (number of nonzero coefficients) along the solution path.

value The sequence of optimal value of the object function corresponded to the se-
quence of lambda.

iterN The number of iteration in the program.

lambda The sequence of regularization parameters lambda obtained in the program.

type The type of the problem, such as Dantzig and SparseSVM.
See Also

primal-package.Dantzig_solver

Examples

```r
## Compressed Sensing
## We set X to be standard normal random matrix and generate Y using gaussian noise.
## Generate the design matrix and coefficient vector
n = 100 # sample number
d = 250 # sample dimension
c = 0.5 # correlation parameter
s = 20  # support size of coefficient
set.seed(1024)
X = scale(matrix(rnorm(n*d),n,d)+c*rnorm(n))/sqrt(n-1)*sqrt(n)
beta = c(rnorm(s), rep(0, d-s))
## Generate response using Gaussian noise, and solve the solution path
noise = rnorm(n)
Y = X%*%beta + noise
## Compressed Sensing solved with parametric simplex method
fit.compressed = CompressedSensing_solver(X, Y, max_it = 100, lambda_threshold = 0.01)
### lambdas used
print(fit.compressed$lambda)
## number of nonzero coefficients for each lambda
print(fit.compressed$df)
## Visualize the solution path
plot(fit.compressed)
```
**Value**

An object with S3 class "primal" is returned:

- **data**: The $n$ by $d$ data matrix from the input.
- **response**: The length $n$ response vector from the input.
- **beta**: A matrix of regression estimates whose columns correspond to regularization parameters for parametric simplex method.
- **df**: The degree of freedom (number of nonzero coefficients) along the solution path.
- **value**: The sequence of optimal value of the object function corresponded to the sequence of lambda.
- **iterN**: The number of iteration in the program.
- **lambda**: The sequence of regularization parameters lambda obtained in the program.
- **type**: The type of the problem, such as Dantzig and SparseSVM.

**See Also**

primal-package

**Examples**

```r
## Dantzig
## We set X to be standard normal random matrix and generate Y using gaussian noise.
## Generate the design matrix and coefficient vector
n = 100 # sample number
d = 250 # sample dimension
c = 0.5 # correlation parameter
s = 20 # support size of coefficient
set.seed(1024)
X = scale(matrix(rnorm(n*d),n,d)+c*rnorm(n))/sqrt(n-1)*sqrt(n)
beta = c(rnorm(s), rep(0, d-s))
## Generate response using Gaussian noise, and solve the solution path
noise = rnorm(n)
Y = X%*%beta + noise
## Dantzig selection solved with parametric simplex method
fit.dantzig = Dantzig_solver(X, Y, max_it = 100, lambda_threshold = 0.01)
### lambdas used
print(fit.dantzig$lambda)
## number of nonzero coefficients for each lambda
print(fit.dantzig$df)
## Visualize the solution path
plot(fit.dantzig)
```
plot.primal  \textit{Plot function for S3 class "primal"}

\textbf{Description}

Plot regularization path and parameter obtained from the algorithm.

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'primal'
plot(x, n = NULL, ...)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textbf{x} \hspace{1cm} An object with S3 class "primal"
  \item \textbf{n} \hspace{1cm} If \( n = \text{NULL} \), three graph will be shown together. If \( n \) is a number, then the corresponding graph will be shown.
  \item ... \hspace{1cm} System reserved (No specific usage)
\end{itemize}

\textbf{See Also}

\texttt{Dantzig_solver, SparseSVM_solver}

print.primal  \textit{Print function for S3 class "primal"}

\textbf{Description}

Print the information about the model usage, the parameter path, degree of freedom of the solution path.

\textbf{Usage}

\begin{verbatim}
## S3 method for class 'primal'
print(x, ...)\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textbf{x} \hspace{1cm} An object with S3 class "primal".
  \item ... \hspace{1cm} System reserved (No specific usage)
\end{itemize}

\textbf{See Also}

\texttt{Dantzig_solver, SparseSVM_solver}
PSM_solver

Solve given problem in parametric simplex method

Description

Solve given problem in parametric simplex method

Usage

PSM_solver(A, b, b_bar, c, c_bar, B_init = NULL, max_it = 50, lambda_threshold = 0.01)

Arguments

A
b
b_bar
c
c_bar
B_init
max_it
lambda_threshold

A is an n by d data matrix
b is a length n response vector
b_bar is a length n vector time to parameter in constraints.
c is a length d vector in target function.
c_bar is a length d vector time to parameter in target function
B_init is the index of initial basic column.
This is the number of the maximum path length one would like to achieve. The default length is 50.
The parametric simplex method will stop when the calculated parameter is smaller than lambda. The default value is 0.01.

Value

An object with S3 class "primal" is returned:
data
response
beta
beta0
df
value
iterN
lambda
type

The n by d data matrix from the input
The length n response vector from the input
A matrix of regression estimates whose columns correspond to regularization parameters for parametric simplex method.
A vector of regression estimates whose index correspond to regularization parameters for parametric simplex method.
The degree of freecom (number of nonzero coefficients) along the solution path.
The sequence of optimal value of the object function corresponded to the sequence of lambda.
The number of iteration in the program.
The sequence of regularization parameters lambda obtained in the program.
The type of the problem, such as Dantzig and SparseSVM.
QuantileRegression_solver

Solve given quantile regression problem in parametric simplex method

Description

Solve given quantile regression problem in parametric simplex method

Usage

QuantileRegression_solver(X, y, max_it = 50, lambda_threshold = 0.01, tau = 0.5)
Arguments

- **X**: \( x \) is an \( n \) by \( d \) data matrix
- **y**: \( y \) is a length \( n \) response vector
- **max_it**: This is the number of the maximum path length one would like to achieve. The default length is 50.
- **lambda_threshold**: The parametric simplex method will stop when the calculated parameter is smaller than \( \lambda \). The default value is 0.01.
- **tau**: The quantile number you want. The default quantile is 0.5

Value

An object with S3 class "primal" is returned:

- **data**: The \( n \) by \( d \) data matrix from the input
- **response**: The length \( n \) response vector from the input
- **beta**: A matrix of regression estimates whose columns correspond to regularization parameters for parametric simplex method.
- **df**: The degree of freedom (number of nonzero coefficients) along the solution path.
- **value**: The sequence of optimal value of the object function corresponded to the sequence of \( \lambda \).
- **iterN**: The number of iteration in the program.
- **lambda**: The sequence of regularization parameters \( \lambda \) obtained in the program.
- **type**: The type of the problem, such as Dantzig and SparseSVM.

See Also

- primal-package, Dantzig_solver

Examples

```r
## Quantile Regression
## We set X to be standard normal random matrix and generate Y using gaussian noise
## with default quantile number to be 0.5.
## Generate the design matrix and coefficient vector
n = 100 # sample number
d = 250 # sample dimension
c = 0.5 # correlation parameter
s = 20 # support size of coefficient
set.seed(1024)
X = scale(matrix(rnorm(n*d),n,d)+c*rnorm(n))/sqrt(n-1)*sqrt(n)
beta = c(rnorm(s), rep(0, d-s))
## Generate response using Gaussian noise, and solve the solution path
noise = rnorm(n)
Y = X%*%beta + noise
## Quantile Regression problem solved with parametric simplex method
fit.quantile = QuantileRegression_solver(X, Y, max_it = 100, lambda_threshold = 0.01)
```
### SparseSVM_solver

**Description**

Solve given Sparse SVM problem in parametric simplex method

**Usage**

SparseSVM_solver(X, y, max_it = 50, lambda_threshold = 0.01)

**Arguments**

- **X**
  - `x` is an n by d data matrix
- **y**
  - `y` is a length n response vector
- **max_it**
  - This is the number of the maximum path length one would like to achieve. The default length is 50.
- **lambda_threshold**
  - The parametric simplex method will stop when the calculated parameter is smaller than lambda. The default value is 0.01.

**Value**

An object with S3 class "primal" is returned:

- **data**
  - The n by d data matrix from the input
- **response**
  - The length n response vector from the input
- **beta**
  - A matrix of regression estimates whose columns correspond to regularization parameters for parametric simplex method.
- **beta0**
  - A vector of regression estimates whose index correspond to regularization parameters for parametric simplex method.
- **df**
  - The degree of freecom (number of nonzero coefficients) along the solution path.
- **value**
  - The sequence of optimal value of the object function corresponded to the sequence of lambda.
- **iterN**
  - The number of iteration in the program.
- **lambda**
  - The sequence of regularization parameters `lambda` obtained in the program.
- **type**
  - The type of the problem, such as Dantzig and SparseSVM.
## SparseSVM

We set the X matrix to be normal random matrix and Y is a vector consists of -1 and 1 with the number of iteration to be 1000.

Generate the design matrix and coefficient vector

```r
n = 200  # sample number
d = 100  # sample dimension
c = 0.5  # correlation parameter
s = 20   # support size of coefficient
set.seed(1024)
X = matrix(rnorm(n*d),n,d)+c*rnorm(n)
```  

Generate response and solve the solution path

```r
Y <- sample(c(-1,1),n,replace = TRUE)
fit.SVM = SparseSVM_solver(X, Y, max_it = 1000, lambda_threshold = 0.01)
```

Lambdas used

```r
print(fit.SVM$lambda)
```

Visualize the solution path

```r
plot(fit.SVM)
```
Index

_PACKAGE (PRIMAL-package), 2

coef.primal, 2, 2
CompressedSensing_solver, 2, 3

Dantzig_solver, 2–4, 4, 6, 9

plot.primal, 2, 6
PRIMAL-package, 2
primal-package (PRIMAL-package), 2
print.primal, 2, 6
PSM_solver, 2, 7

QuantileRegression_solver, 2, 8

SparseSVMSolver, 2, 3, 6, 10