Package 'logitr'

February 28, 2018

Type Package

Title Penalized Logistic Regression

Version 0.1.0

Description This is an R package for linear and logistic regression with optional ridge and bridge regularization penalties.

URL https://github.com/MGallow/logitr

BugReports https://github.com/MGallow/logitr/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends Rcpp (>= 0.12.10),
RcppArmadillo,
dplyr

LinkingTo Rcpp,
RcppArmadillo

RoxygenNote 6.0.1

Suggests testthat

R topics documented:

CV_linearc *CV Linearc (c++)*

Description

Computes the coefficient estimates for linear regression. ridge regularization and bridge regularization optional. This function is to be used with the "linearc" function

Usage

```
CV_linearc(X, y, lam = 0L, alpha = 0L, penalty = "none", weights = 0L,
  intercept = TRUE, kernel = FALSE, method = "SVD", tol = 1e-05,
  maxit = 10000, vec = 0L, init = 0L, K = 5L)
```

Arguments

X	matrix
y	matrix or vector of response values 0,1
lam	vector of tuning parameters for ridge regularization term. Defaults to 'lam = 0'
alpha	vector of tuning parameters for bridge regularization term. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
intercept	Defaults to TRUE
method	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
tol	tolerance - used to determine algorithm convergence. Defaults to 1e-5
maxit	maximum iterations. Defaults to 1e5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm
K	specify number of folds in cross validation, if necessary

Value

returns best lambda, best alpha, cv.errors

Examples

```
CV_linearc(X, y, lam = seq(0.1, 2, 0.1), alpha = seq(1.1, 1.9, 0.1), penalty = "bridge", vec = c(0,1,1,1))
```

CV_logisticc *CV Logisticc (c++)*

Description

Computes the coefficient estimates for logistic regression. ridge regularization and bridge regularization optional. This function is to be used with the "logisticc" function.

Usage

```
CV_logisticc(X, y, lam = 0L, alpha = 0L, penalty = "none",
  intercept = TRUE, method = "IRLS", tol = 1e-05, maxit = 10000,
  vec = 0L, init = 0L, criteria = "logloss", K = 5L)
```

Arguments

X	matrix
y	matrix or vector of response values 0,1
lam	vector of tuning parameters for ridge regularization term. Defaults to 'lam = 0'
alpha	vector of tuning parameters for bridge regularization term. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
intercept	Defaults to TRUE
method	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
tol	tolerance - used to determine algorithm convergence. Defaults to 1e-5
maxit	maximum iterations. Defaults to 1e5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm
criteria	specify the criteria for cross validation. Choose from c("mse", "logloss", "misclass"). Defauls to "logloss"
K	specify number of folds in cross validation, if necessary

Value

returns best lambda, best alpha, and cross validation errors

Examples

```
CV_logisticc(X, y, lam = seq(0.1, 2, 0.1), alpha = c(1.1, 1.9, 0.1), penalty = "bridge", method = "MM", vec = c
```

linearc *Linearc (c++)*

Description

Computes the linear regression coefficient estimates (ridge and bridge penalization and weights, optional)

Usage

```
linearc(X, y, lam = 0, alpha = 1.5, penalty = "none", weights = 0L,
  intercept = TRUE, kernel = FALSE, method = "SVD", tol = 1e-05,
  maxit = 1e+05, vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to "alpha = 1.5"
penalty	choose from c("none", "ridge", "bridge"). Defaults to "none"
weights	optional vector of weights for weighted least squares
intercept	add column of ones if not already present. Defaults to TRUE
kernel	use linear kernel to compute ridge regression coefficeients. Defaults to TRUE when p » n (for "SVD")
method	optimization algorithm. Choose from "SVD" or "MM". Defaults to "SVD"
tol	tolerance - used to determine algorithm convergence for "MM". Defaults to 10^-5
maxit	maximum iterations for "MM". Defaults to 10^5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm

Value

returns the coefficient estimates

Examples

```
Weighted ridge regression
library(dplyr)
X = dplyr::select(iris, -c(Species, Sepal.Length))
y = dplyr::select(iris, Sepal.Length)
linearc(X, y, lam = 0.1, penalty = "ridge", weights = rep(1:150), vec = c(0,1,1,1))

Kernelized ridge regression
linearc(X, y, lam = 0.1, penalty = "ridge", kernel = T, vec = c(0,1,1,1))
```

linearr *Linear*

Description

Computes the linear regression coefficient estimates (ridge-penalization and weights, optional)

Usage

```
linearr(X, y, lam = seq(0, 2, 0.1), alpha = 1.5, penalty = "none",
  weights = NULL, intercept = TRUE, kernel = FALSE, method = "SVD",
  tol = 1e-05, maxit = 1e+05, vec = NULL, init = 1, K = 5)
```

Arguments

X	matrix or data frame
y	matrix or data frame of response values
lam	optional tuning parameter for ridge regularization term. If passing a list of values, the function will choose the optimal value based on K-fold cross validation. Defaults to 'lam = seq(0, 2, 0.1)'
alpha	optional tuning parameter for bridge regularization term. If passing a list of values, the function will choose the optimal value based on K-fold cross validation. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
weights	optional vector of weights for weighted least squares
intercept	add column of ones if not already present. Defaults to TRUE
kernel	use linear kernel to compute ridge regression coefficeients. Defaults to TRUE when p » n (for 'SVD')
method	optimization algorithm. Choose from 'SVD' or 'MM'. Defaults to 'SVD'
tol	tolerance - used to determine algorithm convergence for 'MM'. Defaults to 10^-5
maxit	maximum iterations for 'MM'. Defaults to 10^5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm
K	specify number of folds for cross validation, if necessary

Value

returns the selected tuning parameters, coefficient estimates, MSE, and gradients

Examples

```
Weighted ridge regression
library(dplyr)
X = dplyr::select(iris, -c(Species, Sepal.Length))
y = dplyr::select(iris, Sepal.Length)
linearr(X, y, lam = 0.1, penalty = 'ridge', weights = rep(1:150))
```

```
Kernelized ridge regression
linearr(X, y, lam = 0.1, penalty = 'ridge', kernel = T)
```

---

logisticc *Logistic Regression (c++)*

Description

Computes the coefficient estimates for logistic regression. ridge regularization and bridge regularization optional.

Usage

```
logisticc(X, y, lam = 0, alpha = 1.5, penalty = "none",
  intercept = TRUE, method = "IRLS", tol = 1e-05, maxit = 1e+05,
  vec = 0L, init = 0L)
```

Arguments

X	matrix
y	matrix or vector of response values 0,1
lam	optional tuning parameter for ridge regularization term. Defaults to 'lam = 0'
alpha	optional tuning parameter for bridge regularization term. Defaults to 'alpha = 1.5'
penalty	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
intercept	Defaults to TRUE
method	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
tol	tolerance - used to determine algorithm convergence. Defaults to 1e-5
maxit	maximum iterations. Defaults to 1e5
vec	optional vector to specify which coefficients will be penalized
init	optional initialization for MM algorithm

Value

returns beta estimates (includes intercept), total iterations, and gradients.

Examples

```
Logistic Regression
library(dplyr)
X = as.matrix(dplyr::select(iris, -Species))
y = as.matrix(dplyr::select(iris, Species))
y = ifelse(y == 'setosa', 1, 0)
logisticc(X, y, vec = c(0,1,1,1))

ridge Logistic Regression with IRLS
logisticc(X, y, lam = 0.1, penalty = 'ridge', vec = c(0,1,1,1))

ridge Logistic Regression with MM
```

```
logisticc(X, y, lam = 0.1, penalty = 'ridge', method = 'MM', vec = c(0,1,1,1))

bridge Logistic Regression
logisticc(X, y, lam = 0.1, alpha = 1.5, penalty = 'bridge', method = "MM", vec = c(0,1,1,1))
```

`logisticr` *Logistic Regression*

Description

Computes the coefficient estimates for logistic regression. ridge regularization and bridge regularization optional.

Usage

```
logisticr(X, y, lam = seq(0, 2, 0.1), alpha = 1.5, penalty = "none",
  intercept = TRUE, method = "IRLS", tol = 1e-05, maxit = 1e+05,
  vec = NULL, init = 1, criteria = "logloss", K = 5)
```

Arguments

`X`	matrix or data frame
`y`	matrix or vector of response values 0,1
`lam`	optional tuning parameter(s) for ridge regularization term. If passing a list of values, the function will choose optimal value based on K-fold cross validation. Defaults to 'lam = seq(0, 2, 0.1)'
`alpha`	optional tuning parameter for bridge regularization term. If passing a list of values, the function will choose the optimal value based on K-fold cross validation. Defaults to 'alpha = 1.5'
`penalty`	choose from c('none', 'ridge', 'bridge'). Defaults to 'none'
`intercept`	Defaults to TRUE
`method`	optimization algorithm. Choose from 'IRLS' or 'MM'. Defaults to 'IRLS'
`tol`	tolerance - used to determine algorithm convergence. Defaults to 10^-5
`maxit`	maximum iterations. Defaults to 10^5
`vec`	optional vector to specify which coefficients will be penalized
`init`	optional initialization for MM algorithm
`criteria`	specify the criteria for cross validation. Choose from c('mse', 'logloss', 'misclass'). Defauls to 'logloss'
`K`	specify number of folds for cross validation, if necessary

Value

returns selected tuning parameters, beta estimates (includes intercept), MSE, log loss, misclassification rate, total iterations, and gradients.

Examples

```
Logistic Regression
library(dplyr)
X = dplyr::select(iris, -Species)
y = dplyr::select(iris, Species)
y$Species = ifelse(y$Species == 'setosa', 1, 0)
logisticr(X, y)

ridge Logistic Regression with IRLS
logistir(X, y, lam = 0.1, penalty = 'ridge')

ridge Logistic Regression with MM
logisticr(X, y, lam = 0.1, penalty = 'ridge', method = 'MM')

bridge Logistic Regression
(Defaults to MM -- IRLS will return error)
logisticr(X, y, lam = 0.1, alpha = 1.5, penalty = 'bridge')
```

---

logitc *Logitc (c++)*

Description

Computes the logit for u

Usage

```
logitc(u)
```

Arguments

u	some number

Value

returns the logit of u

Examples

```
logit(X*beta)
```

predict.linearr *Predict Linear Regression*

Description

Generates prediction for linear regression. Note that one can either input a 'linearr' object or a matrix of beta coefficients.

Usage

```
## S3 method for class 'linearr'
predict(object, X, y = NULL)
```

Arguments

object	'linearr' object or matrix of betas
X	matrix or data frame of (new) observations
y	optional, matrix or vector of response values

Value

predictions and loss metrics

Examples

```
fitted = linearr(X, y, lam = 0.1)
predict_linearr(fitted, X)
```

predict.logisticr *Predict Logistic Regression*

Description

Generates prediction for logistic regression. Note that one can either input a 'logisticr' object or a matrix of beta coefficients.

Usage

```
## S3 method for class 'logisticr'
predict(object, X, y = NULL)
```

Arguments

object	'logisticr' object or matrix of betas
X	matrix or data frame of (new) observations
y	optional, matrix or vector of response values 0,1

Value

predictions and loss metrics

Examples

```
fitted = logisticr(X, y, lam = 0.1, penalty = 'ridge', method = 'MM')
predict_logisticr(fitted, X)
```

---

predict_linearc *Predict Linear Regression*

Description

Generates prediction for linear regression

Usage

```
predict_linearc(betas, X, y = 0L)
```

Arguments

betas	'linearr' object or matrix of betas
X	matrix of (new) observations
y	matrix of response values

Value

predictions and loss metrics

Examples

```
fitted = linearr(X, y, penalty = "ridge")
predict_linearr(fitted$coefficients, X)
```

---

predict_logisticc *Predict Logistic Regression (c++)*

Description

Generates prediction for logistic regression

Usage

```
predict_logisticc(betas, X, y = 0L)
```

Arguments

betas	matrix of coefficientts
X	matrix of (new) observations
y	matrix of response values 0,1

Value

predictions and loss metrics

Examples

```
fitted = logisticr(X, y, lam = 0.1, penalty = 'ridge', method = 'MM')
predict_logisticr(fitted$coefficients, X)
```

---

print.linearr *Print logitr object*

Usage

```
## S3 method for class 'linearr'
print(x, ...)
```

Arguments

x	logitr class object

---

print.logisticr *Print logitr object*

Usage

```
## S3 method for class 'logisticr'
print(x, ...)
```

Arguments

x	logitr class object

Index