Package 'hdme'

April 12, 2019

Type Package

Title High-Dimensional Regression with Measurement Error

Version 0.2.3

Encoding UTF-8

Author Oystein Sorensen

Maintainer Oystein Sorensen <oystein.sorensen.1985@gmail.com>

Description Penalized regression for generalized linear models for measurement error problems (aka. errors-in-variables). The package contains a version of the lasso (L1-penalization) which corrects for measurement error (Sorensen et al. (2015) <doi:10.5705/ss.2013.180>). It also contains an implementation of the Generalized Matrix Uncertainty Selector, which is a version the (Generalized) Dantzig Selector for the case of measurement error (Sorensen et al. (2018) <doi:10.1080/10618600.2018.1425626>).

License GPL-3

LazyData TRUE

RoxygenNote 6.1.1

Imports glmnet (>= 2.0-13), ggplot2 (>= 2.2.1), Rdpack, Rglpk (>= 0.6-1), Rcpp (>= 0.12.15)

URL https://github.com/osorensen/hdme

RdMacros Rdpack

Suggests knitr, rmarkdown, testthat, flare, dplyr, tidyr, covr

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-12 10:03:24 UTC

R topics documented:

`cv_corrected_lasso` *Cross-validated Corrected lasso*

Description

Cross-validated Corrected lasso

Usage

```
cv_corrected_lasso(W, y, sigmaUU, n_folds = 10, family = "gaussian",
  radii = NULL, no_radii = 100, alpha = 0.1, maxits = 5000)
```

Arguments

`W`	Design matrix, measured with error.
`y`	Vector of the continuous response value.
`sigmaUU`	Covariance matrix of the measurement error.
`n_folds`	Number of folds to use in cross-validation. Default is 100.
`family`	Only "gaussian" is implemented at the moment.
`radii`	Optional vector containing the set of radii of the l1-ball onto which the solution is projected.
`no_radii`	Length of vector radii, i.e., the number of regularization parameters to fit the corrected lasso for.
`alpha`	Optional step size of the projected gradient descent algorithm. Default is 0.1.
`maxits`	Optional maximum number of iterations of the project gradient descent algorithm for each radius. Default is 5000.

Details

Corrected version of the lasso for the case of linear regression, estimated using cross-validation. The method does require an estimate of the measurement error covariance matrix.

Value

A list

References

Loh P, Wainwright MJ (2012). "High-dimensional regression with noisy and missing data: Provable guarantees with nonconvexity." *Ann. Statist.*, **40**(3), 1637–1664.

Sorensen O, Frigessi A, Thoresen M (2015). "Measurement error in lasso: Impact and likelihood bias correction." *Statistica Sinica*, **25**(2), 809-829.

Examples

```
# Gaussian
set.seed(100)
n <- 100; p <- 50 # Problem dimensions
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
# (typically estimated by replicate measurements)
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaUU))
# Coefficient
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the corrected lasso
cvfit <- cv_corrected_lasso(W, y, sigmaUU, no_radii = 5, n_folds = 3)
plot(cvfit)
# Run the standard lasso using the radius found by cross-validation
fit <- fit_corrected_lasso(W, y, sigmaUU, family = "gaussian",
radii = cvfit$radius_min)
```

---

fit_corrected_lasso *Corrected Lasso*

Description

Lasso (L1-regularization) for generalized linear models with measurement error.

Usage

```
fit_corrected_lasso(W, y, sigmaUU, family = c("gaussian", "binomial",
  "poisson"), radii = NULL, no_radii = NULL, alpha = 0.1,
  maxits = 5000)
```

Arguments

W	Design matrix, measured with error. Must be a numeric matrix.
y	Vector of responses.
sigmaUU	Covariance matrix of the measurement error.
family	Response type. Character string of length 1. Possible values are "gaussian", "binomial" and "poisson".
radii	Vector containing the set of radii of the l1-ball onto which the solution is projected. If not provided, the algorithm will select an evenly spaced vector of 20 radii.
no_radii	Length of vector radii, i.e., the number of regularization parameters to fit the corrected lasso for.
alpha	Step size of the projected gradient descent algorithm. Default is 0.1.
maxits	Maximum number of iterations of the project gradient descent algorithm for each radius. Default is 5000.

Details

Corrected version of the lasso for generalized linear models. The method does require an estimate of the measurement error covariance matrix.

Value

Returns a list containing a matrix whose columns represent the corrected beta estimates at each radius, as well as the vector of radii used.

References

Loh P, Wainwright MJ (2012). "High-dimensional regression with noisy and missing data: Provable guarantees with nonconvexity." *Ann. Statist.*, **40**(3), 1637–1664.

Sorensen O, Frigessi A, Thoresen M (2015). "Measurement error in lasso: Impact and likelihood bias correction." *Statistica Sinica*, **25**(2), 809-829.

Examples

```
# Example with linear regression
# Number of samples
n <- 100
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
# (typically estimated by replicate measurements)
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaUU))
# Coefficient
```

```
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the corrected lasso
fit <- fit_corrected_lasso(W, y, sigmaUU, family = "gaussian")
plot(fit)

# Binomial, logistic regression
# Number of samples
n <- 1000
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaUU))
logit <- function(x) (1+exp(-x))^(-1)
# Response
y <- rbinom(n, size = 1, prob = logit(X %*% c(rep(5, 5), rep(0, p-5))))
fit <- fit_corrected_lasso(W, y, sigmaUU, family = "binomial")
plot(fit)
```

`fit_gds` *Generalized Dantzig Selector*

Description

Generalized Dantzig Selector

Usage

```
fit_gds(X, y, lambda = NULL, family = c("gaussian", "binomial"))
```

Arguments

`X`	Design matrix.
`y`	Vector of the continuous response value.
`lambda`	Regularization parameter. Only a single value is supported.
`family`	Use "gaussian" for linear regression and "binomial" for logistic regression.

Value

Intercept and coefficients at the values of lambda specified.

References

Candes E, Tao T (2007). "The Dantzig selector: Statistical estimation when p is much larger than n." *Ann. Statist.*, **35**(6), 2313–2351.

James GM, Radchenko P (2009). "A generalized Dantzig selector with shrinkage tuning." *Biometrika*, **96**(2), 323-337.

Examples

```
# Example with logistic regression
n <- 1000  # Number of samples
p <- 10 # Number of covariates
X <- matrix(rnorm(n * p), nrow = n) # True (latent) variables # Design matrix
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5)) # True regression coefficients
y <- rbinom(n, 1, (1 + exp(-X %*% beta))^(-1)) # Binomially distributed response
gds <- fit_gds(X, y, family = "binomial")
```

`fit_gmus` *Generalized Matrix Uncertainty Selector*

Description

Generalized Matrix Uncertainty Selector

Usage

```
fit_gmus(W, y, lambda = NULL, delta = NULL, family = c("gaussian",
  "binomial", "poisson"))
```

Arguments

`W`	Design matrix, measured with error. Must be a numeric matrix.
`y`	Vector of responses.
`lambda`	Regularization parameter.
`delta`	Additional regularization parameter, bounding the measurement error.
`family`	"gaussian" for linear regression, "binomial" for logistic regression or "poisson" for Poisson regression.

Value

List object with intercept and coefficients at the values of lambda and delta specified, as well as regularization parameters.

References

Rosenbaum M, Tsybakov AB (2010). "Sparse recovery under matrix uncertainty." *Ann. Statist.*, **38**(5), 2620–2651.

Sorensen O, Hellton KH, Frigessi A, Thoresen M (2018). "Covariate Selection in High-Dimensional Generalized Linear Models With Measurement Error." *Journal of Computational and Graphical Statistics*, **27**(4), 739-749. doi: 10.1080/10618600.2018.1425626, https://doi.org/10.1080/10618600.2018.1425626, https://doi.org/10.1080/10618600.2018.1425626.

Examples

```
# Example with linear regression
set.seed(1)
n <- 100 # Number of samples
p <- 50 # Number of covariates
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement matrix (this is the one we observe)
W <- X + matrix(rnorm(n*p, sd = 1), nrow = n, ncol = p)
# Coefficient vector
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the MU Selector
gmus1 <- fit_gmus(W, y)
# Draw an elbow plot to select delta
plot(gmus1)

# Now, according to the "elbow rule", choose
# the final delta where the curve has an "elbow".
# In this case, the elbow is at about delta = 0.08,
# so we use this to compute the final estimate:
gmus2 <- fit_gmus(W, y, delta = 0.08)
# Plot the coefficients
plot(gmus2)
```

`fit_gmu_lasso` *Generalized Matrix Uncertainty Lasso*

Description

Generalized Matrix Uncertainty Lasso

Usage

```
fit_gmu_lasso(W, y, lambda = NULL, delta = NULL, family = "binomial",
  active_set = TRUE)
```

Arguments

W	Design matrix, measured with error. Must be a numeric matrix.
y	Vector of responses.
lambda	Regularization parameter. If not set, lambda.min from glmnet::cv.glmnet is used.
delta	Additional regularization parameter, bounding the measurement error.
family	Character string. Currently "binomial" and "poisson" are supported.
active_set	Logical. Whether or not to use an active set strategy to speed up coordinate descent algorithm.

Value

List object with intercept and coefficients at the values of lambda and delta specified, as well as regularization parameters.

References

Rosenbaum M, Tsybakov AB (2010). "Sparse recovery under matrix uncertainty." *Ann. Statist.*, **38**(5), 2620–2651.

Sorensen O, Hellton KH, Frigessi A, Thoresen M (2018). "Covariate Selection in High-Dimensional Generalized Linear Models With Measurement Error." *Journal of Computational and Graphical Statistics*, **27**(4), 739-749. doi: 10.1080/10618600.2018.1425626, https://doi.org/10.1080/10618600.2018.1425626, https://doi.org/10.1080/10618600.2018.1425626.

Examples

```
set.seed(1)
# Number of samples
n <- 200
# Number of covariates
p <- 100
# Number of nonzero features
s <- 10
# True coefficient vector
beta <- c(rep(1,s),rep(0,p-s))
# Standard deviation of measurement error
sdU <- 0.2
# True data, not observed
X <- matrix(rnorm(n*p),nrow = n,ncol = p)
# Measured data, with error
W <- X + sdU * matrix(rnorm(n * p), nrow = n, ncol = p)
# Binomial response
y <- rbinom(n, 1, (1 + exp(-X%*%beta))**(-1))
# Run the GMU Lasso
gmu_lasso <- fit_gmu_lasso(W, y, delta = NULL)
# Get an elbow plot, in order to choose delta.
plot(gmu_lasso)
```

`fit_mus` *Matrix Uncertainty Selector*

Description

Matrix Uncertainty Selector

Usage

```
fit_mus(W, y, lambda = NULL, delta = NULL)
```

Arguments

`W`	Design matrix, measured with error. Must be a numeric matrix.
`y`	Vector of responses.
`lambda`	Regularization parameter.
`delta`	Additional regularization parameter, bounding the measurement error.

Value

Intercept and coefficients at the values of lambda and delta specified.

References

Rosenbaum M, Tsybakov AB (2010). "Sparse recovery under matrix uncertainty." *Ann. Statist.*, **38**(5), 2620–2651.

Sorensen O, Hellton KH, Frigessi A, Thoresen M (2018). "Covariate Selection in High-Dimensional Generalized Linear Models With Measurement Error." *Journal of Computational and Graphical Statistics*, **27**(4), 739-749. doi: 10.1080/10618600.2018.1425626, https://doi.org/10.1080/10618600.2018.1425626, https://doi.org/10.1080/10618600.2018.1425626.

Examples

```
# Example with Gaussian response
set.seed(1)
# Number of samples
n <- 100
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement matrix (this is the one we observe)
W <- X + matrix(rnorm(n*p, sd = 1), nrow = n, ncol = p)
# Coefficient vector
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
```

```
# Run the MU Selector
mus1 <- fit_mus(W, y)
# Draw an elbow plot to select delta
plot(mus1)

# Now, according to the "elbow rule", choose the final delta where the curve has an "elbow".
# In this case, the elbow is at about delta = 0.08, so we use this to compute the final estimate:
mus2 <- fit_mus(W, y, delta = 0.08)
plot(mus2) # Plot the coefficients
```

`plot.corrected_lasso` *plot.corrected_lasso*

Description

Plot the output of fit_corrected_lasso

Usage

```
## S3 method for class 'corrected_lasso'
plot(x, type = "nonzero", ...)
```

Arguments

`x`	Object of class corrected_lasso, returned from calling fit_corrected_lasso()
`type`	Type of plot. Either "nonzero" or "path".
`...`	Other arguments to plot (not used)

Examples

```
# Example with linear regression
n <- 100 # Number of samples
p <- 50 # Number of covariates
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement error covariance matrix
# (typically estimated by replicate measurements)
sigmaUU <- diag(x = 0.2, nrow = p, ncol = p)
# Measurement matrix (this is the one we observe)
W <- X + rnorm(n, sd = diag(sigmaUU))
# Coefficient
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the corrected lasso
fit <- fit_corrected_lasso(W, y, sigmaUU, family = "gaussian")
plot(fit)
```

plot.cv_corrected_lasso *plot.cv_corrected_lasso*

Description

Plot the output of cv_corrected_lasso

Usage

```
## S3 method for class 'cv_corrected_lasso'
plot(x, ...)
```

Arguments

x	The object to be plotted, returned from cv_corrected_lasso
...	Other arguments to plot (not used).

plot.gds *Plot the estimates returned by fit_gds*

Description

Plot the number of nonzero coefficients at the given lambda.

Usage

```
## S3 method for class 'gds'
plot(x, ...)
```

Arguments

x	An object of class gds
...	Other arguments to plot (not used).

Examples

```
set.seed(1)
# Example with logistic regression
# Number of samples
n <- 1000
# Number of covariates
p <- 10
# True (latent) variables (Design matrix)
X <- matrix(rnorm(n * p), nrow = n)
# True regression coefficients
```

```
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Binomially distributed response
y <- rbinom(n, 1, (1 + exp(-X %*% beta))^(-1))
# Fit the generalized Dantzig Selector
gds <- fit_gds(X, y, family = "binomial")
# Plot the estimated coefficients at the chosen lambda
plot(gds)
```

`plot.gmus` *Plot the estimates returned by fit_gmus and fit_mus*

Description

Plot the number of nonzero coefficients along a range of delta values if delta has length larger than 1, or the estimated coefficients if delta has length 1.

Usage

```
## S3 method for class 'gmus'
plot(x, ...)
```

Arguments

`x`	An object of class gmus
`...`	Other arguments to plot (not used).

Examples

```
# Example with linear regression
set.seed(1)
# Number of samples
n <- 100
# Number of covariates
p <- 50
# True (latent) variables
X <- matrix(rnorm(n * p), nrow = n)
# Measurement matrix (this is the one we observe)
W <- X + matrix(rnorm(n*p, sd = 0.4), nrow = n, ncol = p)
# Coefficient vector
beta <- c(seq(from = 0.1, to = 1, length.out = 5), rep(0, p-5))
# Response
y <- X %*% beta + rnorm(n, sd = 1)
# Run the MU Selector
mus1 <- fit_mus(W, y)
# Draw an elbow plot to select delta
plot(mus1)

# Now, according to the "elbow rule", choose the final
```

```
# delta where the curve has an "elbow".
# In this case, the elbow is at about delta = 0.08, so
# we use this to compute the final estimate:
mus2 <- fit_mus(W, y, delta = 0.08)
# Plot the coefficients
plot(mus2)
```

`plot.gmu_lasso` *Plot the estimates returned by fit_gmu_lasso*

Description

Plot the number of nonzero coefficients along a range of delta values if delta has length larger than 1, or the estimated coefficients of delta has length 1.

Usage

```
## S3 method for class 'gmu_lasso'
plot(x, ...)
```

Arguments

`x`	An object of class gmu_lasso
`...`	Other arguments to plot (not used).

Examples

```
set.seed(1)
n <- 200
p <- 50
s <- 10
beta <- c(rep(1,s),rep(0,p-s))
sdU <- 0.2

X <- matrix(rnorm(n*p),nrow = n,ncol = p)
W <- X + sdU * matrix(rnorm(n * p), nrow = n, ncol = p)

y <- rbinom(n, 1, (1 + exp(-X%*%beta))**(-1))
gmu_lasso <- fit_gmu_lasso(W, y)

plot(gmu_lasso)
```

Index