Package 'FarmSelect'

April 19, 2018

Title Factor Adjusted Robust Model Selection

Version 1.0.2

Description Implements a consistent model selection strategy for high dimensional sparse regression when the covariate dependence can be reduced through factor models. By separating the latent factors from idiosyncratic components, the problem is transformed from model selection with highly correlated covariates to that with weakly correlated variables. It is appropriate for cases where we have many variables compared to the number of samples. Moreover, it implements a robust procedure to estimate distribution parameters wherever possible, hence being suitable for cases when the underlying distribution deviates from Gaussianity. See the paper on the 'FarmSelect' method, Fan et al.(2017) <arXiv:1612.08490>, for detailed description of methods and further references.

Depends R (>= 3.3.0)

License GPL-2

Encoding UTF-8

LazyData true

Imports graphics, stats, grDevices, utils, methods, Rcpp, ncvreg, fBasics

URL https://kbose28.github.io/FarmSelect/

RoxygenNote 6.0.1

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Koushiki Bose [aut, cre], Yuan Ke [aut], Kaizheng Wang [aut]

Maintainer Koushiki Bose <bose@princeton.edu>

Repository CRAN

Date/Publication 2018-04-19 21:46:04 UTC

R topics documented:

`farm.res` *Adjusting a data matrix for underlying factors*

Description

Given a matrix of covariates, this function estimates the underlying factors and computes data residuals after regressing out those factors.

Usage

```
farm.res(X, K.factors = NULL, robust = TRUE, cv = FALSE, tau = 2,
  verbose = TRUE)
```

Arguments

`X`	an n x p data matrix with each row being a sample.
`K.factors`	a *optional* number of factors to be estimated. Otherwise estimated internally. K>0.
`robust`	a boolean, specifying whether or not to use robust estimators for mean and variance. Default is TRUE.
`cv`	a boolean, specifying whether or not to run cross-validation for the tuning parameter. Default is FALSE. Only used if `robust` is TRUE.
`tau`	`>0` multiplier for the tuning parameter for Huber loss function. Default is 2. Only used if `robust` is TRUE and `cv` is FALSE. See details.
`verbose`	a boolean specifying whether to print runtime updates to the console. Default is TRUE.

Details

For details about the method, see Fan et al.(2017).

Using `robust = TRUE` uses the Huber's loss to estimate parameters robustly. For details of covariance estimation method see Fan et al.(2017).

Number of rows and columns of the data matrix must be at least 4 in order to be able to calculate latent factors.

Number of latent factors, if not provided, is estimated by the eignevalue ratio test. See Ahn and Horenstein(2013). The maximum number is taken to be min(n,p)/2. User can supply a larger number is desired.

The tuning parameter = `tau * sigma * optimal rate` where `optimal rate` is the optimal rate for the tuning parameter. For details, see Fan et al.(2017). `sigma` is the standard deviation of the data.

Value

A list with the following items

`residual`	the data after being adjusted for underlying factors
`loadings`	estimated factor loadings
`factors`	estimated factors
`nfactors`	the number of (estimated) factors

References

Ahn, S. C., and A. R. Horenstein (2013): "Eigenvalue Ratio Test for the Number of Factors," Econometrica, 81 (3), 1203–1227.

Fan J., Ke Y., Wang K., "Decorrelation of Covariates for High Dimensional Sparse Regression." https://arxiv.org/abs/1612.08490

See Also

`farm.select`

Examples

```
set.seed(100)
P = 200 #dimension
N = 50 #samples
K = 3 #nfactors
Q = 3 #model size
Lambda = matrix(rnorm(P*K, 0,1), P,K)
F = matrix(rnorm(N*K, 0,1), N,K)
U = matrix(rnorm(P*N, 0,1), P,N)
X = Lambda%*%t(F)+U
X = t(X)
output = farm.res(X) #default options
output$nfactors
output = farm.res(X, K.factors = 10) #inputting factors
names(output) #list of output
```

`farm.select` *Factor-adjusted robust model selection*

Description

Given a covariate matrix and output vector, this function first adjusts the covariates for underlying factors and then performs model selection.

Usage

```
farm.select(X, Y, loss = c("scad", "mcp", "lasso"), robust = TRUE,
  cv = FALSE, tau = 2, lin.reg = TRUE, K.factors = NULL,
  max.iter = 10000, nfolds = ceiling(length(Y)/3), eps = 1e-04,
  verbose = TRUE)
```

Arguments

`X`	an n x p covariate matrix with each row being a sample. Must have same number of rows as the size of `Y`.
`Y`	a size n outcome vector.
`loss`	a character string specifying the loss function to be minimized. Must be one of "scad" (default) "mcp" or "lasso". You can just specify the initial letter.
`robust`	a boolean, specifying whether or not to use robust estimators for mean and variance. Default is TRUE.
`cv`	a boolean, specifying whether or not to run cross-validation for the tuning parameter. Default is FALSE. Only used if `robust` is TRUE.
`tau`	`>0`, multiplier for the tuning parameter for Huber loss function. Default is 2. Only used if `robust` is TRUE and `cv` is FALSE. See details.
`lin.reg`	a boolean, specifying whether or not to assume that we have a linear regression model (TRUE) or a logit model (FALSE) structure. Default is TRUE.
`K.factors`	number of factors to be estimated. Otherwise estimated internally. K>0.
`max.iter`	maximum number of iterations across the regularization path. Default is 10000.
`nfolds`	the number of cross-validation folds. Default is ceiling(samplesize/3).
`eps`	Convergence threshhold for model fitting using `ncvreg`. The algorithm iterates until the RMSD for the change in linear predictors for any coefficient is less than eps. Default is `1e-4`.
`verbose`	a boolean specifying whether to print runtime updates to the console. Default is TRUE.

Details

For formula of how the covariates are adjusted for latent factors, see Section 3.2 in Fan et al.(2017).

The tuning parameter = `tau * sigma * optimal rate` where `optimal rate` is the optimal rate for the tuning parameter. For details, see Fan et al.(2017). `sigma` is the standard deviation of the data.

`ncvreg` is used to fit the model after decorrelation. This package may output its own warnings about failures to converge and model saturation.

Value

A list with the following items

`model.size`	the size of the model
`beta.chosen`	the indices of the covariates chosen in the model
`coef.chosen`	the coefficients of the chosen covariates
`X.residual`	the residual covariate matrix after adjusting for factors
`nfactors`	number of (estimated) factors
`n`	number of observations
`p`	number of dimensions
`robust`	whether robust parameters were used
`loss`	loss function used

#' @details Number of rows and columns of the covariate matrix must be at least 4 in order to be able to calculate latent factors.

References

Fan J., Ke Y., Wang K., "Decorrelation of Covariates for High Dimensional Sparse Regression." https://arxiv.org/abs/1612.08490

See Also

`print.farm.select` `farm.res`

Examples

```
##linear regression
set.seed(100)
P = 200 #dimension
N = 50 #samples
K = 3 #nfactors
Q = 3 #model size
Lambda = matrix(rnorm(P*K, 0,1), P,K)
F = matrix(rnorm(N*K, 0,1), N,K)
U = matrix(rnorm(P*N, 0,1), P,N)
X = Lambda%*%t(F)+U
X = t(X)
```

```
beta_1 = rep(5,Q)
beta = c(beta_1, rep(0,P-Q))
eps = rt(N, 2.5)
Y = X%*%beta+eps

##with default options
output = farm.select(X,Y) #robust, no cross-validation
output$beta.chosen #variables selected
output$coef.chosen #coefficients of selected variables

#examples of other robustification options
output = farm.select(X,Y,robust = FALSE) #non-robust
output = farm.select(X,Y, tau = 3) #robust, no cross-validation, specified tau
#output = farm.select(X,Y, cv= TRUE) #robust, cross-validation: LONG RUNNING!

##changing the loss function and inputting factors
output = farm.select(X, Y,loss = "mcp", K.factors = 4)

##use a logistic regression model, a larger sample size is desired.
## Not run:
set.seed(100)
P = 400 #dimension
N = 300 #samples
K = 3 #nfactors
Q = 3 #model size
Lambda = matrix(rnorm(P*K, 0,1), P,K)
F = matrix(rnorm(N*K, 0,1), N,K)
U = matrix(rnorm(P*N, 0,1), P,N)
X = Lambda%*%t(F)+U
X = t(X)
beta_1 = rep(5, Q)
beta = c(beta_1, rep(0,P-Q))
eps = rnorm(N)
Prob = 1/(1+exp(-X%*%beta))
Y = rbinom(N, 1, Prob)

output = farm.select(X,Y, lin.reg=FALSE, eps=1e-3)
output$beta.chosen
output$coef.chosen

## End(Not run)
```

FarmSelect *FarmSelect: Factor Adjusted Robust Model Selection*

Description

This R package implements a consistent model selection strategy for high dimensional sparse regression when the covariate dependence can be reduced through factor models. By separating the latent factors from idiosyncratic components, the problem is transformed from model selection

with highly correlated covariates to that with weakly correlated variables. It is appropriate for cases where we have many variables compared to the number of samples. Moreover, it implements a robust procedure to estimate distribution parameters wherever possible, hence being suitable for cases when the underlying distribution deviates from Gaussianity, which is commonly assumed in the literature. See the paper on this method, Fan et al.(2017) <https://arxiv.org/abs/1612.08490>, for detailed description of methods and further references. For detailed information on how to use and install see https://kbose28.github.io/FarmSelect.

`print.farm.select` *Summarize and print the results of the model selection*

Description

Print method for `farm.select` objects

Usage

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

Arguments

`x`	A `farm.select` object.
`...`	Further arguments passed to or from other methods.

Value

A list with the following items:

`model.size`	the size of the model
`beta.chosen`	the indices of the covariates chosen in the model
`coef.chosen`	the coefficients of the chosen covariates
`X.residual`	the residual covariate matrix after adjusting for factors
`nfactors`	number of (estimated) factors
`n`	number of observations
`p`	number of dimensions
`robust`	whether robust parameters were used
`loss`	loss function used

See Also

`farm.select`

Examples

```
set.seed(100)
P = 200 #dimension
N = 50 #samples
K = 3 #nfactors
Q = 3 #model size
Lambda = matrix(rnorm(P*K, 0,1), P,K)
F = matrix(rnorm(N*K, 0,1), N,K)
U = matrix(rnorm(P*N, 0,1), P,N)
X = Lambda%*%t(F)+U
X = t(X)
beta_1 = rep(5,Q)
beta = c(beta_1, rep(0,P-Q))
eps = rt(N, 2.5)
Y = X%*%beta+eps

##with default options
output = farm.select(X,Y)
output
```

Index