Package 'OPCreg'

April 24, 2025

Title Online Principal Component Regression for Online Datasets

Date 2025-04-03

Version 0.2.0

Description
The online principal component regression method can process the online data set. 'OPCreg' implements the online principal component regression method, which is specifically designed to process online datasets efficiently. This method is particularly useful for handling large-scale, streaming data where traditional batch processing methods may be computationally infeasible.The philosophy of the package is described in 'Guo' (2025) <doi:10.1016/j.physa.2024.130308>.

RoxygenNote 7.3.2

Imports MASS, stats, Matrix, car

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Author Guangbao Guo [aut, cre] (<`https://orcid.org/0000-0002-4115-6218`>),
Chunjie Wei [aut]

Maintainer Guangbao Guo <`ggb11111111@163.com`>

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2025-04-24 17:50:03 UTC

Encoding UTF-8

License GPL-3

Contents

`IPC` *The incremental principal component method can handle online data sets.*

Description

The incremental principal component method can handle online data sets.

Usage

```
IPC(data, m, eta)
```

Arguments

`data`	is an online data set
`m`	is the number of principal component
`eta`	is the proportion of online data to total data

Value

T2,T2k,V,Vhat,lambdahat,time

Examples

```
library(MASS)
n=2000;p=20;m=9;
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
D=as.matrix(diag(rep(runif(p,0,1))))
epsilon=matrix(mvrnorm(n,rep(0,p),D),nrow=n)
data=mu+F%*%t(A)+epsilon
IPC(data=data,m=m,eta=0.8)
```

IPCR *Incremental Principal Component Regression for Online Datasets*

Description

The IPCR function implements an incremental Principal Component Regression (PCR) method designed to handle online datasets. It updates the principal components recursively as new data arrives, making it suitable for real-time data processing.

Usage

```
IPCR(data, eta, m, alpha)
```

Arguments

`data`	A data frame where the first column is the response variable and the remaining columns are predictor variables.
`eta`	The proportion of the initial sample size used to initialize the principal components ($0 < \text{eta} < 1$). Default is 0.0035.
`m`	The number of principal components to retain. Default is 3.
`alpha`	The significance level used for calculating critical values. Default is 0.05.

Details

The IPCR function performs the following steps: 1. Standardizes the predictor variables. 2. Initializes the principal components using the first `n0 = round(eta * n)` samples. 3. Recursively updates the principal components as each new sample arrives. 4. Fits a linear regression model using the principal component scores. 5. Back-transforms the regression coefficients to the original scale.

This method is particularly useful for datasets where new observations are continuously added, and the model needs to be updated incrementally.

Value

A list containing the following elements:

`Bhat`	The estimated regression coefficients, including the intercept.
`RMSE`	The Root Mean Square Error of the regression model.
`summary`	The summary of the linear regression model.
`yhat`	The predicted values of the response variable.

See Also

`lm`: For fitting linear models.

`eigen`: For computing eigenvalues and eigenvectors.

Examples

```
set.seed(1234)
library(MASS)
n <- 2000
p <- 10
mu0 <- as.matrix(runif(p, 0))
sigma0 <- as.matrix(runif(p, 0, 10))
ro <- as.matrix(c(runif(round(p / 2), -1, -0.8), runif(p - round(p / 2), 0.8, 1)))
R0 <- ro %*% t(ro)
diag(R0) <- 1
Sigma0 <- sigma0 %*% t(sigma0) * R0
x <- mvrnorm(n, mu0, Sigma0)
colnames(x) <- paste("x", 1:p, sep = "")
e <- rnorm(n, 0, 1)
B <- sample(1:3, (p + 1), replace = TRUE)
en <- matrix(rep(1, n * 1), ncol = 1)
y <- cbind(en, x) %*% B + e
colnames(y) <- paste("y")
data <- data.frame(cbind(y, x))
IPCR(data = data, m = 3, eta = 0.0035, alpha = 0.05)
```

`PCR` *Principal Component Regression (PCR)*

Description

The `PCR` function performs Principal Component Regression (PCR) on a given dataset. It standardizes the predictor variables, determines the number of principal components to retain based on a specified threshold, and fits a linear regression model using the principal component scores.

Usage

```
PCR(data, threshold)
```

Arguments

`data`	A data frame where the first column is the response variable and the remaining columns are predictor variables.
`threshold`	The proportion of variance to retain in the principal components (default is 0.8).

Details

The function performs the following steps: 1. Standardize the predictor variables. 2. Compute the covariance matrix of the standardized predictors. 3. Perform eigen decomposition on the covariance matrix to obtain principal components. 4. Determine the number of principal components to retain based on the cumulative explained variance exceeding the specified threshold. 5. Project the standardized predictors onto the retained principal components. 6. Fit a linear regression model using the principal component scores. 7. Back-transform the regression coefficients to the original scale.

Value

A list containing the following elements:

`Bhat`	The estimated regression coefficients, including the intercept.
`RMSE`	The Root Mean Square Error of the regression model.
`summary`	The summary of the linear regression model.
`yhat`	The predicted values of the response variable.

See Also

`lm`: For fitting linear models.

`eigen`: For computing eigenvalues and eigenvectors.

Examples

```
# Example data
library(MASS)
set.seed(1234)
n <- 2000
p <- 30
mu0 <- as.matrix(runif(p, 0))
sigma0 <- as.matrix(runif(p, 0, 10))
ro <- matrix(c(runif(round(p / 2), -1, -0.8), runif(p - round(p / 2), 0.8, 1)))
R0 <- ro %*% t(ro)
diag(R0) <- 1
Sigma0 <- sigma0 %*% t(sigma0) * R0
x <- mvrnorm(n, mu0, Sigma0)
colnames(x) <- paste("x", 1:p, sep = "")
e <- rnorm(n, 0, 1)
B <- sample(1:3, (p + 1), replace = TRUE)
en <- matrix(rep(1, n * 1), ncol = 1)
y <- cbind(en, x) %*% B + e
colnames(y) <- paste("y")
data <- data.frame(cbind(y, x))
# Call the PCR function
PCR(data, threshold = 0.9)
```

`PPC` *The perturbation principal component method can handle online data sets.*

Description

The perturbation principal component method can handle online data sets.

Usage

```
PPC(data, m, eta)
```

Arguments

`data`	is an online data set
`m`	is the number of principal component
`eta`	is the proportion of online data to total data

Value

T2,T2k,V,Vhat,lambdahat,time

Examples

```
library(MASS)
library(OPCreg)
n=2000;p=20;m=9;
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
D=as.matrix(diag(rep(runif(p,0,1))))
epsilon=matrix(mvrnorm(n,rep(0,p),D),nrow=n)
data=mu+F%*%t(A)+epsilon
PPC(data=data,m=m,eta=0.8)
```

PPCR *Perturbation-based Principal Component Regression*

Description

This function performs Perturbation-based Principal Component Regression (PPCR) on the provided dataset. It combines Principal Component Analysis (PCA) with linear regression, incorporating perturbation to enhance robustness.

Usage

```
PPCR(data, eta = 0.0035, m = 3, alpha = 0.05, perturbation_factor = 0.1)
```

Arguments

`data`	A data frame containing the response variable and predictors.
`eta`	A proportion (between 0 and 1) determining the initial sample size for PCA.
`m`	The number of principal components to retain.
`alpha`	Significance level (currently not used in the function).
`perturbation_factor`	A factor controlling the magnitude of perturbation added to the principal components.

Details

The function first standardizes the predictors, then performs PCA on an initial subset of the data. It iteratively updates the principal components by incorporating new observations and adding random perturbations. Finally, it fits a linear regression model using the principal components as predictors and transforms the coefficients back to the original space.

Value

A list containing the following components:

`Bhat`	Estimated regression coefficients in the original space.
`RMSE`	Root Mean Squared Error of the regression model.
`summary`	Summary of the linear regression model.
`Vhat`	Estimated principal components.
`lambdahat`	Estimated eigenvalues.
`yhat`	Predicted values from the regression model.

See Also

`lm`: For linear regression models.

`prcomp`: For principal component analysis.

Examples

```
set.seed(1234)
library(MASS)
n <- 2000
p <- 10
mu0 <- as.matrix(runif(p, 0))
sigma0 <- as.matrix(runif(p, 0, 10))
ro <- as.matrix(c(runif(round(p / 2), -1, -0.8), runif(p - round(p / 2), 0.8, 1)))
R0 <- ro %*% t(ro)
diag(R0) <- 1
Sigma0 <- sigma0 %*% t(sigma0) * R0
x <- mvrnorm(n, mu0, Sigma0)
colnames(x) <- paste("x", 1:p, sep = "")
e <- rnorm(n, 0, 1)
B <- sample(1:3, (p + 1), replace = TRUE)
en <- matrix(rep(1, n * 1), ncol = 1)
y <- cbind(en, x) %*% B + e
colnames(y) <- paste("y")
data <- data.frame(cbind(y, x))
PPCR(data, eta = 0.0035, m = 3, alpha = 0.05, perturbation_factor = 0.1)
```

SAPC *The stochastic approximate component method can handle online data sets.*

Description

The stochastic approximate component method can handle online data sets.

Usage

```
SAPC(data, m, eta, alpha)
```

Arguments

data	is a online data set
m	is the number of principal component
eta	is the proportion of online data to total data
alpha	is the step size

Value

T2,T2k,V,Vhat,lambdahat,time

Examples

```
library(MASS)
n=2000;p=20;m=9;
mu=t(matrix(rep(runif(p,0,1000),n),p,n))
mu0=as.matrix(runif(m,0))
sigma0=diag(runif(m,1))
F=matrix(mvrnorm(n,mu0,sigma0),nrow=n)
A=matrix(runif(p*m,-1,1),nrow=p)
D=as.matrix(diag(rep(runif(p,0,1))))
epsilon=matrix(mvrnorm(n,rep(0,p),D),nrow=n)
data=mu+F%*%t(A)+epsilon
SAPC(data=data,m=m,eta=0.8,alpha=1)
```

SPCR *Stochastic Principal Component Regression*

Description

The Stochastic Principal Component Regression (SPCR) function performs principal component regression on an online dataset using a stochastic update rule. It is designed to handle large datasets efficiently by incrementally updating the principal components as new data arrives.

Usage

```
SPCR(data, eta, m)
```

Arguments

`data`	A data frame containing the response variable and predictors.
`eta`	A proportion (between 0 and 1) determining the initial sample size for PCA.
`m`	The number of principal components to retain.

Details

The function first standardizes the predictors, then performs PCA on an initial subset of the data. It iteratively updates the principal components using a stochastic approximation method. Finally, it fits a linear regression model using the principal components as predictors and transforms the coefficients back to the original space.

Value

A list containing the following components:

`Bhat`	Estimated regression coefficients in the original space.
`RMSE`	Root Mean Squared Error of the regression model.
`summary`	Summary of the linear regression model.
`Vhat`	Estimated principal components.
`lambdahat`	Estimated eigenvalues.
`time`	Computation time of the function.
`yhat`	Predicted values from the regression model.

Examples

```
# Load necessary libraries
library(MASS)
library(Matrix)
library(car)
# Set seed for reproducibility
set.seed(1234)

# Define sample size and number of variables
n = 2000
p = 10
# Mean vector
mu0 = runif(p, 0)

# Method 1: Generate a positive-definite matrix using the Wishart distribution
Sigma0 = rWishart(1, df = p, Sigma = diag(p))[,,1]

# Method 2: Manually construct a positive-definite matrix
# A = matrix(rnorm(p^2), nrow = p)
# Sigma0 = A %*% t(A) + diag(p) * 10

# Method 3: Adjust an existing matrix to be positive-definite
# Sigma0 = nearPD(Sigma0)$mat

# Generate multivariate normal data
x = mvrnorm(n, mu0, Sigma0)
colnames(x)<-paste("x",1:p,sep="")
e=rnorm(n,0,1)
B = sample(1:3,(p+1),replace = TRUE)
en<-matrix(rep(1,n*1),ncol=1)
y=cbind(en,x)
colnames(y)<-paste("y", 1:ncol(y), sep="")
data<-data.frame(cbind(y,x))
#lm.sol<-lm(y~.,data=data)
#summary(lm.sol)
#VIF<-mean(vif(lm.sol));VIF

X<-scale(data[,-1])
p<-ncol(X)
n<-nrow(X)
s=Sys.time()
S<-cov(X)
eig<-eigen(S)
diag_S<-diag(S)
sum_rank<-sum(diag_S)
m=0
if (m==0){
eig<-eigen(S)
sum_eig<-sum(diag(S))
for (i in 1:p){
if (sum(eig$values[1:i])/sum_eig>0.9){
m<-i;break
}
}
}
# Example usage of SPCR function
SPCR(data, eta = 0.0035, m = 3)
```

---

spcrl *The stochastic principal component regression with varying learning-rate can handle online data sets.*

Description

The stochastic principal component regression with varying learning-rate can handle online data sets.

Usage

```
spcrl(data, m, eta, alpha)
```

Arguments

data	is a online data set
m	is the number of principal component
eta	is the proportion of online data to total data
alpha	is the step size

Value

T2,T2k,V,Vhat,lambdahat,time

Examples

```
# Load necessary libraries
library(MASS)
library(Matrix)
library(car)
# Set seed for reproducibility
set.seed(1234)

# Define sample size, number of variables and number of principal component
n = 2000
p = 10
m=9
# Mean vector
mu0 = runif(p, 0)

# Method 1: Generate a positive-definite matrix using the Wishart distribution
Sigma0 = rWishart(1, df = p, Sigma = diag(p))[,,1]

# Method 2: Manually construct a positive-definite matrix
# A = matrix(rnorm(p^2), nrow = p)
# Sigma0 = A %*% t(A) + diag(p) * 10

# Method 3: Adjust an existing matrix to be positive-definite
# Sigma0 = nearPD(Sigma0)$mat

# Generate multivariate normal data
x = mvrnorm(n, mu0, Sigma0)
colnames(x)<-paste("x",1:p,sep="")
e=rnorm(n,0,1)
B = sample(1:3,(p+1),replace = TRUE)
en<-matrix(rep(1,n*1),ncol=1)
y=cbind(en,x)
colnames(y)<-paste("y", 1:ncol(y), sep="")
data<-data.frame(cbind(y,x))
spcrl(data=data,m=m,eta=0.8,alpha=0.5)
```

Index