Package ‘adass’

November 18, 2021

Type Package
Title Adaptive Smoothing Spline (AdaSS) Estimator for the Function-on-Function Linear Regression
Version 1.0.0
Description Implements the adaptive smoothing spline estimator for the function-on-function linear regression model described in Centofanti et al. (2020) <arXiv:2011.12036>.
License GPL-3
Encoding UTF-8
Imports fda, parallel, matrixcalc, SparseM, mvtnorm, Rfast, plot3D
URL https://github.com/unina-sfere/adass
BugReports https://github.com/unina-sfere/adass
SystemRequirements GNU make
Suggests knitr, rmarkdown, testthat
RoxygenNote 7.1.1
NeedsCompilation no
Author Fabio Centofanti [cre, aut], Antonio Lepore [aut], Alessandra Menafoglio [aut], Biagio Palumbo [aut], Simone Vantini [aut]
Maintainer Fabio Centofanti <fabio.centofanti@unina.it>
Repository CRAN
Date/Publication 2021-11-18 14:40:05 UTC

R topics documented:

adass-package .......................................................... 2
adass.fr ................................................................. 3
adass.fr_eaass .......................................................... 5
plot.adass ............................................................... 7
simulate_data .......................................................... 8
adass-package

Adaptive smoothing spline estimator for the function-on-function linear regression model

Description

Implements the adaptive smoothing spline estimator for the function-on-function linear regression model described in Centofanti et al. (2020) <arXiv:2011.12036>.

Details

Package: adass
Type: Package
Version: 1.0.0
Date: 2021-11-16
License: GPL-3

Author(s)

Fabio Centofanti, Antonio Lepore, Alessandra Menafoglio, Biagio Palumbo, Simone Vantini

References


See Also

adass.fr, adass.fr_eaass

Examples

library(adass)
data<-simulate_data("Scenario HAT",n_obs=100)
X_fd=data$X_fd
Y_fd=data$Y_fd
basis_s <- fda::create.bspline.basis(c(0,1),nbasis = 10,norder = 4)
basis_t <- fda::create.bspline.basis(c(0,1),nbasis = 10,norder = 4)
mod_smooth <-adass.fr(Y_fd,X_fd,basis_s = basis_s,basis_t = basis_t,tun_par=c(10^-6,10^-6,0,0,0,0))
grid_s<-seq(0,1,length.out = 10)
grid_t<-seq(0,1,length.out = 10)
beta_der_eval_s<-fda::eval.bifd(grid_s,grid_t,mod_smooth$Beta_hat_fd,sLfdobj = 2)
beta_der_eval_t <- fda::eval.bifd(grid_s, grid_t, mod_smooth$Beta_hat_fd, tLfdobj = 2)
mod_adsm <- adass.fr_eaass(Y_fd, X_fd, basis_s, basis_t,
        beta_ders = beta_der_eval_s, beta_dert = beta_der_eval_t,
        rand_search_par = list(c(-8, 4), c(-8, 4), c(0, 0.1), c(0, 4), c(0, 0.1), c(0, 4)),
        grid_eval_ders = grid_s, grid_eval_dert = grid_t,
        popul_size = 2, ncores = 1, iter_num = 1)

mod_opt <- adass.fr(Y_fd, X_fd, basis_s = basis_s, basis_t = basis_t,
        tun_par = mod_adsm$tun_par_opt, beta_ders = beta_der_eval_s,
        beta_dert = beta_der_eval_t, grid_eval_ders = grid_s, grid_eval_dert = grid_t)
plot(mod_opt)

---

adass.fr

Adaptive smoothing spline estimator for the function-on-function linear regression model

Description

The adaptive smoothing spline (AdaSS) estimator for the function-on-function linear regression proposed in Centofanti et al., 2020.

Usage

adass.fr(
    Y_fd,
    X_fd,
    basis_s,
    basis_t,
    beta_ders = NULL,
    beta_dert = NULL,
    grid_eval_ders = NULL,
    grid_eval_dert = NULL,
    tun_par = c(lambda_s = 10^4, lambda_t = 10^4, delta_s = 0, gamma_s = 1, delta_t = 0,
                delta_t = 1),
    CV = FALSE,
    K = 10,
    X_fd_test = NULL,
    Y_fd_test = NULL
)

Arguments

Y_fd An object of class fd corresponding to the response functions.
X_fd An object of class fd corresponding to the covariate functions.
basis_s B-splines basis along the s-direction of class basisfd.
basis_t B-splines basis along the t-direction of class basisfd.
beta_ders  Initial estimate of the partial derivative of the coefficient function along the s-direction. Either a matrix or a class basisfd object. If NULL no adaptive penalty is used along the s-direction.

beta_dert  Initial estimate of the partial derivative of the coefficient function along the t-direction. Either a matrix or a class basisfd object. If NULL no adaptive penalty is used along the t-direction.

grid_eval_ders  Grid of evaluation of the partial derivatives along the s-direction.

grid_eval_dert  Grid of evaluation of the partial derivatives along the t-direction.

tun_par  Vector of tuning parameters.

CV  If TRUE the K-fold cross-validation prediction error is calculated. Default is FALSE. If X_fd_test and Y_fd_test are both provided the prediction error on the test set is calculated in place of the cross-validation prediction error when CV is TRUE.

K  Number of folds. Default is 10.

X_fd_test  Test set covariate functions. Default is NULL.

Y_fd_test  Test set response functions. Default is NULL.

Value

A list containing the following arguments:

- B: The basis coefficients matrix estimate of the coefficient function.
- Beta_hat_fd: The coefficient function estimate of class bifd.
- alpha: The intercept function estimate.
- tun_par: Vector of tuning parameters.
- CV: Estimated prediction error.
- CV_sd: Standard error of the estimated prediction error.
- Y_fd: The response functions.
- X_fd: The covariate functions.

References


See Also

- adass.fr_eaass

Examples

```r
library(adass)
data<-simulate_data("Scenario HAT",n_obs=100)
X_fd=data$X_fd
Y_fd=data$Y_fd
```
```r
basis_s <- fda::create.bspline.basis(c(0,1), nbasis = 10, norder = 4)
basis_t <- fda::create.bspline.basis(c(0,1), nbasis = 10, norder = 4)
mod_smooth <- adass.fr(Y_fd, X_fd, basis_s = basis_s, basis_t = basis_t, tun_par = c(10^-6, 10^-6, 0, 0, 0, 0))
grid_s <- seq(0, 1, length.out = 10)
grid_t <- seq(0, 1, length.out = 10)
beta_der_eval_s <- fda::eval.bifd(grid_s, grid_t, mod_smooth$Beta_hat_fd, slfdobj = 2)
beta_der_eval_t <- fda::eval.bifd(grid_s, grid_t, mod_smooth$Beta_hat_fd, tlfdobj = 2)
mod_adass <- adass.fr(Y_fd, X_fd, basis_s = basis_s, basis_t = basis_t,
tun_par = c(10^-6, 10^-6, 0, 1, 0, 1),
beta_ders = beta_der_eval_s,
beta_dert = beta_der_eval_t,
grid_eval_ders = grid_s, grid_eval_dert = grid_t)
```

---

**adass.fr_eaass**

Evolutionary algorithm for the adaptive smoothing spline estimator (EAASS).

**Description**

EAASS algorithm to choose the tuning parameters for the AdaSS estimator (Centofanti et al., 2020).

**Usage**

```r
adass.fr_eaass(Y_fd, X_fd, basis_s, basis_t, beta_ders = NULL, beta_dert = NULL, grid_eval_ders = NULL, grid_eval_dert = NULL, rand_search_par = list(c(-4, 4), c(-4, 4), c(0, 1, 5, 10, 15), c(0, 1, 2, 3, 4), c(0, 1, 5, 10, 15), c(0, 1, 2, 3, 4)), popul_size = 12, iter_num = 10, r = 0.2, pert_vec = c(0.8, 1.2), X_fd_test = NULL, Y_fd_test = NULL, progress = TRUE, ncores = 1, K = 10)
```

**Arguments**

- **Y_fd** An object of class fd corresponding to the response functions.
- **X_fd** An object of class fd corresponding to the covariate functions.
basis_s  B-splines basis along the s-direction of class basisfd.
basis_t  B-splines basis along the t-direction of class basisfd.

beta_ders Initial estimate of the partial derivative of the coefficient function along the s-direction. Either a matrix or a class basisfd object. If NULL no adaptive penalty is used along the s-direction.

beta_dert Initial estimate of the partial derivative of the coefficient function along the t-direction. Either a matrix or a class basisfd object. If NULL no adaptive penalty is used along the t-direction.

grid_eval_ders Grid of evaluation of the partial derivatives along the s-direction.

grid_eval_dert Grid of evaluation of the partial derivatives along the t-direction.

rand_search_par List containing the initial population ranges for the tuning parameters.

popul_size Initial population size.

iter_num Algorithm iterations.

r Truncation parameter in the exploitation phase.

pert_vec Perturbation parameters in the exploration phase.

X_fd_test Test set covariate functions. Default is NULL. If X_fd_test and Y_fd_test are both provided the prediction error on the test set is used as performance metric in place of the cross-validation prediction error.

Y_fd_test Test set response functions. Default is NULL. If X_fd_test and Y_fd_test are both provided the prediction error on the test set is used as performance metric in place of the cross-validation prediction error.

progress If TRUE a progress bar is printed. Default is TRUE.

ncores If ncores>1, then parallel computing is used, with ncores cores. Default is 1.

K Number of folds. Default is 10.

Value

A list containing the following arguments:

- tun_par_opt: Vector of optimal tuning parameters.
- CV: Estimated prediction errors.
- CV_sd: Standard errors of the estimated prediction errors.
- comb_list: The combinations of tuning parameters explored.
- Y_fd: The response functions.
- X_fd: The covariate functions.

References

See Also

adass.fr_eaass

Examples

library(adass)
data<-simulate_data("Scenario HAT",n_obs=100)
X_fd=data$X_fd
Y_fd=data$Y_fd
basis_s <- fda::create.bspline.basis(c(0,1),nbasis = 5,norder = 4)
basis_t <- fda::create.bspline.basis(c(0,1),nbasis = 5,norder = 4)
mod_smooth <- adass.fr(Y_fd,X_fd,basis_s,basis_t,tun_par=c(10^-6,10^-6,0,0,0,0))
grid_s<-seq(0,1,length.out = 5)
grid_t<-seq(0,1,length.out = 5)
beta_der_eval_s<-fda::eval.bifd(grid_s,grid_t,mod_smooth$Beta_hat_fd,sLfdobj = 2)
beta_der_eval_t<-fda::eval.bifd(grid_s,grid_t,mod_smooth$Beta_hat_fd,tLfdobj = 2)
mod_adsm<-adass.fr_eaass(Y_fd,X_fd,basis_s,basis_t,
  beta_ders=beta_der_eval_s, beta_dert=beta_der_eval_t,
  rand_search_par=list(c(-8,4),c(-8,4),c(0,0.1),c(0,4),c(0,0.1),c(0,4)),
  grid_eval_ders=grid_s, grid_eval_dert=grid_t,
  popul_size = 1,ncores=1,iter_num=1)

plot.adass

Plot the results of the AdaSS method

Description

This function provides plots of the AdaSS coefficient function estimate when applied to the output of adass.fr.

Usage

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

Arguments

x The output of adass.fr.
...

No additional parameters, called for side effects.

Value

No return value, called for side effects.
Examples

```r
library(adass)
data<-simulate_data("Scenario HAT", n_obs=100)
X_fd=data$X_fd
Y_fd=data$Y_fd
basis_s <- fda::create.bspline.basis(c(0,1), nbasis = 10, norder = 4)
basis_t <- fda::create.bspline.basis(c(0,1), nbasis = 10, norder = 4)
mod_adass <- adass.fr(Y_fd,X_fd, basis_s = basis_s, basis_t = basis_t, 
tun_par=c(10^-6,10^-6,0,0,0,0))
plot(mod_adass)
```

**simulate_data**  
*Simulate data through the function-on-function linear regression model*

Description

Generate synthetic data as in the simulation study of Centofanti et al. (2020).

Usage

```r
simulate_data(scenario, n_obs = 3000)
```

Arguments

- `scenario` A character strings indicating the scenario considered. It could be "Scenario HAT", "Scenario DAMP", or "Scenario RCHANGE".
- `n_obs` Number of observations.

Value

A list containing the following arguments:
- `X`: Covariate matrix, where the rows correspond to argument values and columns to replications.
- `Y`: Response matrix, where the rows correspond to argument values and columns to replications.
- `X_fd`: Covariate functions.
- `Y_fd`: Response functions.
- `Beta_vero_fd`: The true coefficient function.

References


Examples

```r
library(adass)
data<-simulate_data("Scenario HAT", n_obs=100)
```
Index

adass (adass-package), 2
adass-package, 2
adass.fr, 2, 3
adass.fr_eaass, 2, 4, 5, 7
plot.adass, 7
simulate_data, 8