Package: fdaSP (via r-universe)

August 31, 2024

Type Package

Title Sparse Functional Data Analysis Methods

Version 1.1.1

Date 2023-09-12

Author Mauro Bernardi [aut, cre], Marco Stefanucci [aut], Antonio Canale [ctb]

Maintainer Mauro Bernardi <mauro.bernardi@unipd.it>

Description Provides algorithms to fit linear regression models under several popular penalization techniques and functional linear regression models based on Majorizing-Minimizing (MM) and Alternating Direction Method of Multipliers (ADMM) techniques. See Boyd et al (2010) <doi:10.1561/2200000016> for complete introduction to the method.

License GPL (>= 3)

Suggests RColorBrewer, gglasso, glmnet, latex2exp, utils

Imports Rcpp, Rdpack, grDevices, graphics, stats, parallel, doParallel, foreach, splines, ks, pracma

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.2.3.9000

RdMacros Rdpack

NeedsCompilation yes

Date/Publication 2023-10-05 18:00:02 UTC

Repository https://maurobernardi.r-universe.dev

RemoteUrl https://github.com/cran/fdaSP

RemoteRef HEAD

RemoteSha 95f90ec8645b06dbe1dd0ffe0680e143dbf02541

Contents

`fdaSP-package` *Sparse Functional Data Analysis Methods*

Description

Provides algorithms to fit linear regression models under several popular penalization techniques and functional linear regression models based on Majorizing-Minimizing (MM) and Alternating Direction Method of Multipliers (ADMM) techniques. See Boyd et al (2010) <doi:10.1561/2200000016> for complete introduction to the method.

Package Content

Index of help topics:

```
confband                Function to plot the confidence bands
f2fSP                   Overlap Group Least Absolute Shrinkage and
                        Selection Operator for function-on-function
                        regression model
f2fSP_cv                Cross-validation for Overlap Group Least
                        Absolute Shrinkage and Selection Operator for
                        function-on-function regression model
f2sSP                   Overlap Group Least Absolute Shrinkage and
                        Selection Operator for scalar-on-function
                        regression model
f2sSP_cv                Cross-validation for Overlap Group Least
                        Absolute Shrinkage and Selection Operator on
                        scalar-on-function regression model
fdaSP-package           Sparse Functional Data Analysis Methods
lmSP                    Sparse Adaptive Overlap Group Least Absolute
                        Shrinkage and Selection Operator
lmSP_cv                 Cross-validation for Sparse Adaptive Overlap
                        Group Least Absolute Shrinkage and Selection
                        Operator
softhresh               Function to solve the soft thresholding problem
```

Maintainer

Mauro Bernardi <mauro.bernardi@unipd.it>

Author(s)

Mauro Bernardi [aut, cre], Marco Stefanucci [aut], Antonio Canale [ctb]

`confband` *Function to plot the confidence bands*

Description

Function to plot the confidence bands

Usage

```
confband(xV, yVmin, yVmax)
```

Arguments

`xV`	the values for the x-axis.
`yVmin`	the minimum values for the y-axis.
`yVmax`	the maximum values for the y-axis.

Value

a polygon.

`f2fSP` *Overlap Group Least Absolute Shrinkage and Selection Operator for function-on-function regression model*

Description

Overlap Group-LASSO for function-on-function regression model solves the following optimization problem

$$\min_\psi \frac{1}{2}\sum_{i=1}^{n}\int\left(y_i(s)-\int x_i(t)\psi(t,s)dt\right)^2 ds+\lambda\sum_{g=1}^{G}\|S_g T\psi\|_2$$

to obtain a sparse coefficient vector $\psi = \mathsf{vec}(\Psi) \in \mathbb{R}^{ML}$ for the functional penalized predictor $x(t)$, where the coefficient matrix $\Psi \in \mathbb{R}^{M\times L}$, the regression function $\psi(t,s) = \varphi(t)^\intercal\Psi\theta(s)$, $\varphi(t)$ and $\theta(s)$ are two B-splines bases of order d and dimension M and L, respectively. For each group g, each row of the matrix $S_g \in \mathbb{R}^{d\times ML}$ has non-zero entries only for those bases belonging to that group. These values are provided by the arguments groups and group_weights (see below). Each basis function belongs to more than one group. The diagonal matrix $T \in \mathbb{R}^{ML \times ML}$ contains the basis-specific weights. These values are provided by the argument var_weights (see below). The regularization path is computed for the overlap group-LASSO penalty at a grid of values for the regularization parameter λ using the alternating direction method of multipliers (ADMM). See Boyd et al. (2011) and Lin et al. (2022) for details on the ADMM method.

Usage

```
f2fSP(
  mY,
  mX,
  L,
  M,
  group_weights = NULL,
  var_weights = NULL,
  standardize.data = TRUE,
  splOrd = 4,
  lambda = NULL,
  lambda.min.ratio = NULL,
  nlambda = 30,
  overall.group = FALSE,
  control = list()
)
```

Arguments

- `mY` an $(n \times r_y)$ matrix of observations of the functional response variable.
- `mX` an $(n \times r_x)$ matrix of observations of the functional covariate.
- `L` number of elements of the B-spline basis vector $\theta(s)$.
- `M` number of elements of the B-spline basis vector $\varphi(t)$.
- `group_weights` a vector of length G containing group-specific weights. The default is square root of the group cardinality, see Bernardi et al. (2022).
- `var_weights` a vector of length ML containing basis-specific weights. The default is a vector where each entry is the reciprocal of the number of groups including that basis. See Bernardi et al. (2022) for details.
- `standardize.data` logical. Should data be standardized?
- `splOrd` the order d of the spline basis.
- `lambda` either a regularization parameter or a vector of regularization parameters. In this latter case the routine computes the whole path. If it is NULL values for lambda are provided by the routine.
- `lambda.min.ratio` smallest value for lambda, as a fraction of the maximum lambda value. If $nr_y > LM$, the default is 0.0001, and if $nr_y < LM$, the default is 0.01.
- `nlambda` the number of lambda values - default is 30.

`overall.group` logical. If it is TRUE, an overall group including all penalized covariates is added.

`control` a list of control parameters for the ADMM algorithm. See ‘Details’.

Value

A named list containing

sp.coefficients an $(M \times L)$ solution matrix for the parameters Ψ, which corresponds to the minimum in-sample MSE.

sp.coef.path an $(n_\lambda \times M \times L)$ array of estimated Ψ coefficients for each lambda.

sp.fun an $(r_x \times r_y)$ matrix providing the estimated functional coefficient for $\psi(t, s)$.

sp.fun.path an $(n_\lambda \times r_x \times r_y)$ array providing the estimated functional coefficients for $\psi(t, s)$ for each lambda.

lambda sequence of lambda.

lambda.min value of lambda that attains the minimum in-sample MSE.

mse in-sample mean squared error.

min.mse minimum value of the in-sample MSE for the sequence of lambda.

convergence logical. 1 denotes achieved convergence.

elapsedTime elapsed time in seconds.

iternum number of iterations.

When you run the algorithm, output returns not only the solution, but also the iteration history recording following fields over iterates,

objval objective function value.

r_norm norm of primal residual.

s_norm norm of dual residual.

eps_pri feasibility tolerance for primal feasibility condition.

eps_dual feasibility tolerance for dual feasibility condition.

Iteration stops when both `r_norm` and `s_norm` values become smaller than `eps_pri` and `eps_dual`, respectively.

Details

The control argument is a list that can supply any of the following components:

adaptation logical. If it is TRUE, ADMM with adaptation is performed. The default value is TRUE. See Boyd et al. (2011) for details.

rho an augmented Lagrangian parameter. The default value is 1.

tau.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 2. See Boyd et al. (2011) and Lin et al. (2022) for details.

mu.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 10. See Boyd et al. (2011) and Lin et al. (2022) for details.

abstol absolute tolerance stopping criterion. The default value is sqrt(sqrt(.Machine$double.eps)).

reltol relative tolerance stopping criterion. The default value is sqrt(.Machine$double.eps).

maxit maximum number of iterations. The default value is 100.

print.out logical. If it is TRUE, a message about the procedure is printed. The default value is TRUE.

References

Bernardi M, Canale A, Stefanucci M (2022). "Locally Sparse Function-on-Function Regression." *Journal of Computational and Graphical Statistics*, **0**(0), 1-15. doi:10.1080/10618600.2022.2130926, https://doi.org/10.1080/10618600.2022.2130926.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends® in Machine Learning*, **3**(1), 1-122. ISSN 1935-8237, doi:10.1561/2200000016, http://dx.doi.org/10.1561/2200000016.

Jenatton R, Audibert J, Bach F (2011). "Structured variable selection with sparsity-inducing norms." *J. Mach. Learn. Res.*, **12**, 2777–2824. ISSN 1532-4435.

Lin Z, Li H, Fang C (2022). *Alternating direction method of multipliers for machine learning*. Springer, Singapore. ISBN 978-981-16-9839-2; 978-981-16-9840-8, doi:10.1007/9789811698408, With forewords by Zongben Xu and Zhi-Quan Luo.

Examples

```
## generate sample data
set.seed(4321)
s  <- seq(0, 1, length.out = 100)
t  <- seq(0, 1, length.out = 100)
p1 <- 5
p2 <- 6
r  <- 10
n  <- 50

beta_basis1 <- splines::bs(s, df = p1, intercept = TRUE)    # first basis for beta
beta_basis2 <- splines::bs(s, df = p2, intercept = TRUE)    # second basis for beta

data_basis <- splines::bs(s, df = r, intercept = TRUE)    # basis for X

x_0   <- apply(matrix(rnorm(p1 * p2, sd = 1), p1, p2), 1,
               fdaSP::softhresh, 1.5)  # regression coefficients
x_fun <- beta_basis2 %*% x_0 %*%  t(beta_basis1)

fun_data <- matrix(rnorm(n*r), n, r) %*% t(data_basis)
b        <- fun_data %*% x_fun + rnorm(n * 100, sd = sd(fun_data %*% x_fun )/3)

## set the hyper-parameters
maxit          <- 1000
rho_adaptation <- FALSE
rho            <- 1
reltol         <- 1e-5
```

```
abstol          <- 1e-5

## fit functional regression model
mod <- f2fSP(mY = b, mX = fun_data, L = p1, M = p2,
           group_weights = NULL, var_weights = NULL, standardize.data = FALSE, splOrd = 4,
             lambda = NULL, nlambda = 30, lambda.min.ratio = NULL,
             control = list("abstol" = abstol,
                            "reltol" = reltol,
                            "maxit" = maxit,
                            "adaptation" = rho_adaptation,
                            rho = rho,
             "print.out" = FALSE))

mycol <- function (n) {
palette <- colorRampPalette(RColorBrewer::brewer.pal(11, "Spectral"))
palette(n)
}
cols <- mycol(1000)

oldpar <- par(mfrow = c(1, 2))
image(x_0, col = cols)
image(mod$sp.coefficients, col = cols)
par(oldpar)

oldpar <- par(mfrow = c(1, 2))
image(x_fun, col = cols)
contour(x_fun, add = TRUE)
image(beta_basis2 %*% mod$sp.coefficients %*% t(beta_basis1), col = cols)
contour(beta_basis2 %*% mod$sp.coefficients %*% t(beta_basis1), add = TRUE)
par(oldpar)
```

`f2fSP_cv` *Cross-validation for Overlap Group Least Absolute Shrinkage and Selection Operator for function-on-function regression model*

Description

Overlap Group-LASSO for function-on-function regression model solves the following optimization problem

$$\min_\psi \frac{1}{2}\sum_{i=1}^n \int \left(y_i(s) - \int x_i(t)\psi(t,s)dt \right)^2 ds + \lambda \sum_{g=1}^G \|S_g T\psi\|_2$$

to obtain a sparse coefficient vector $\psi = \mathsf{vec}(\Psi) \in \mathbb{R}^{ML}$ for the functional penalized predictor $x(t)$, where the coefficient matrix $\Psi \in \mathbb{R}^{M\times L}$, the regression function $\psi(t,s) = \varphi(t)^\intercal \Psi \theta(s)$, $\varphi(t)$ and $\theta(s)$ are two B-splines bases of order d and dimension M and L, respectively. For each group g, each row of the matrix $S_g \in \mathbb{R}^{d\times ML}$ has non-zero entries only for those bases belonging to that group. These values are provided by the arguments `groups` and `group_weights` (see below).

Each basis function belongs to more than one group. The diagonal matrix $T \in \mathbb{R}^{ML \times ML}$ contains the basis-specific weights. These values are provided by the argument `var_weights` (see below). The regularization path is computed for the overlap group-LASSO penalty at a grid of values for the regularization parameter λ using the alternating direction method of multipliers (ADMM). See Boyd et al. (2011) and Lin et al. (2022) for details on the ADMM method.

Usage

```
f2fSP_cv(
  mY,
  mX,
  L,
  M,
  group_weights = NULL,
  var_weights = NULL,
  standardize.data = FALSE,
  splOrd = 4,
  lambda = NULL,
  lambda.min.ratio = NULL,
  nlambda = NULL,
  cv.fold = 5,
  overall.group = FALSE,
  control = list()
)
```

Arguments

`mY`	an $(n \times r_y)$ matrix of observations of the functional response variable.
`mX`	an $(n \times r_x)$ matrix of observations of the functional covariate.
`L`	number of elements of the B-spline basis vector $\theta(s)$.
`M`	number of elements of the B-spline basis vector $\varphi(t)$.
`group_weights`	a vector of length G containing group-specific weights. The default is square root of the group cardinality, see Bernardi et al. (2022).
`var_weights`	a vector of length ML containing basis-specific weights. The default is a vector where each entry is the reciprocal of the number of groups including that basis. See Bernardi et al. (2022) for details.
`standardize.data`	logical. Should data be standardized?
`splOrd`	the order d of the spline basis.
`lambda`	either a regularization parameter or a vector of regularization parameters. In this latter case the routine computes the whole path. If it is NULL values for lambda are provided by the routine.
`lambda.min.ratio`	smallest value for lambda, as a fraction of the maximum lambda value. If $nr_y > LM$, the default is 0.0001, and if $nr_y < LM$, the default is 0.01.
`nlambda`	the number of lambda values - default is 30.

`cv.fold`	the number of folds - default is 5.
`overall.group`	logical. If it is TRUE, an overall group including all penalized covariates is added.
`control`	a list of control parameters for the ADMM algorithm. See 'Details'.

Value

A named list containing

sp.coefficients an $(M \times L)$ solution matrix for the parameters Ψ, which corresponds to the minimum cross-validated MSE.

sp.fun an $(r_x \times r_y)$ matrix providing the estimated functional coefficient for $\psi(t, s)$ corresponding to the minimum cross-validated MSE.

lambda sequence of lambda.

lambda.min value of lambda that attains the cross-validated minimum mean squared error.

indi.min.mse index of the lambda sequence corresponding to lambda.min.

mse cross-validated mean squared error.

min.mse minimum value of the cross-validated MSE for the sequence of lambda.

mse.sd standard deviation of the cross-validated mean squared error.

convergence logical. 1 denotes achieved convergence.

elapsedTime elapsed time in seconds.

iternum number of iterations.

Iteration stops when both `r_norm` and `s_norm` values become smaller than `eps_pri` and `eps_dual`, respectively.

Details

The control argument is a list that can supply any of the following components:

adaptation logical. If it is TRUE, ADMM with adaptation is performed. The default value is TRUE. See Boyd et al. (2011) for details.

rho an augmented Lagrangian parameter. The default value is 1.

tau.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 2. See Boyd et al. (2011) and Lin et al. (2022) for details.

mu.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 10. See Boyd et al. (2011) and Lin et al. (2022) for details.

abstol absolute tolerance stopping criterion. The default value is sqrt(sqrt(.Machine$double.eps)).

reltol relative tolerance stopping criterion. The default value is sqrt(.Machine$double.eps).

maxit maximum number of iterations. The default value is 100.

print.out logical. If it is TRUE, a message about the procedure is printed. The default value is TRUE.

References

Bernardi M, Canale A, Stefanucci M (2022). “Locally Sparse Function-on-Function Regression.” *Journal of Computational and Graphical Statistics*, **0**(0), 1-15. doi:10.1080/10618600.2022.2130926, https://doi.org/10.1080/10618600.2022.2130926.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011). “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.” *Foundations and Trends® in Machine Learning*, **3**(1), 1-122. ISSN 1935-8237, doi:10.1561/2200000016, http://dx.doi.org/10.1561/2200000016.

Jenatton R, Audibert J, Bach F (2011). “Structured variable selection with sparsity-inducing norms.” *J. Mach. Learn. Res.*, **12**, 2777–2824. ISSN 1532-4435.

Lin Z, Li H, Fang C (2022). *Alternating direction method of multipliers for machine learning*. Springer, Singapore. ISBN 978-981-16-9839-2; 978-981-16-9840-8, doi:10.1007/9789811698408, With forewords by Zongben Xu and Zhi-Quan Luo.

Examples

```
## generate sample data
set.seed(4321)
s  <- seq(0, 1, length.out = 100)
t  <- seq(0, 1, length.out = 100)
p1 <- 5
p2 <- 6
r  <- 10
n  <- 50

beta_basis1 <- splines::bs(s, df = p1, intercept = TRUE)    # first basis for beta
beta_basis2 <- splines::bs(s, df = p2, intercept = TRUE)    # second basis for beta

data_basis <- splines::bs(s, df = r, intercept = TRUE)    # basis for X

x_0   <- apply(matrix(rnorm(p1 * p2, sd = 1), p1, p2), 1,
               fdaSP::softhresh, 1.5)  # regression coefficients
x_fun <- beta_basis2 %*% x_0 %*%  t(beta_basis1)

fun_data <- matrix(rnorm(n*r), n, r) %*% t(data_basis)
b        <- fun_data %*% x_fun + rnorm(n * 100, sd = sd(fun_data %*% x_fun )/3)

## set the hyper-parameters
maxit          <- 1000
rho_adaptation <- FALSE
rho            <- 0.01
reltol         <- 1e-5
abstol         <- 1e-5

## fit functional regression model
mod_cv <- f2fSP_cv(mY = b, mX = fun_data, L = p1, M = p2,
                   group_weights = NULL, var_weights = NULL,
                   standardize.data = FALSE, splOrd = 4,
                   lambda = NULL, nlambda = 30, cv.fold = 5,
                   lambda.min.ratio = NULL,
```

```
                    control = list("abstol" = abstol,
                                   "reltol" = reltol,
                                   "maxit" = maxit,
                                   "adaptation" = rho_adaptation,
                                   "rho" = rho,
                                   "print.out" = FALSE))

### graphical presentation
plot(log(mod_cv$lambda), mod_cv$mse, type = "l", col = "blue", lwd = 2, bty = "n",
     xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "Prediction Error",
     ylim = range(mod_cv$mse - mod_cv$mse.sd, mod_cv$mse + mod_cv$mse.sd),
     main = "Cross-validated Prediction Error")
fdaSP::confband(xV = log(mod_cv$lambda), yVmin = mod_cv$mse - mod_cv$mse.sd,
                yVmax = mod_cv$mse + mod_cv$mse.sd)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]), col = "red", lwd = 1.0)

### comparison with oracle error
mod <- f2fSP(mY = b, mX = fun_data, L = p1, M = p2,
             group_weights = NULL, var_weights = NULL,
             standardize.data = FALSE, splOrd = 4,
             lambda = NULL, nlambda = 30, lambda.min.ratio = NULL,
             control = list("abstol" = abstol,
                            "reltol" = reltol,
                            "maxit" = maxit,
                            "adaptation" = rho_adaptation,
                            "rho" = rho,
                            "print.out" = FALSE))
err_mod <- apply(mod$sp.coef.path, 1, function(x) sum((x - x_0)^2))
plot(log(mod$lambda), err_mod, type = "l", col = "blue", lwd = 2,
     xlab = latex2exp::TeX("$\\log(\\lambda)$"),
     ylab = "Estimation Error", main = "True Estimation Error", bty = "n")
abline(v = log(mod$lambda[which(err_mod == min(err_mod))]), col = "red", lwd = 1.0)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
       col = "red", lwd = 1.0, lty = 2)
```

`f2sSP` *Overlap Group Least Absolute Shrinkage and Selection Operator for scalar-on-function regression model*

Description

Overlap Group-LASSO for scalar-on-function regression model solves the following optimization problem

$$\min_{\psi,\gamma} \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \int x_i(t)\psi(t)dt - z_i^\intercal\gamma\right)^2 + \lambda\sum_{g=1}^{G}\|S_g T\psi\|_2$$

to obtain a sparse coefficient vector $\psi \in \mathbb{R}^M$ for the functional penalized predictor $x(t)$ and a coefficient vector $\gamma \in \mathbb{R}^q$ for the unpenalized scalar predictors $z_1, \ldots, z_q$. The regression function is $\psi(t) = \varphi(t)^\intercal\psi$ where $\varphi(t)$ is a B-spline basis of order d and dimension M. For each group g, each row of the matrix $S_g \in \mathbb{R}^{d\times M}$ has non-zero entries only for those bases belonging to that group. These values are provided by the arguments `groups` and `group_weights` (see below). Each basis function belongs to more than one group. The diagonal matrix $T \in \mathbb{R}^{M\times M}$ contains the basis-specific weights. These values are provided by the argument `var_weights` (see below). The regularization path is computed for the overlap group-LASSO penalty at a grid of values for the regularization parameter λ using the alternating direction method of multipliers (ADMM). See Boyd et al. (2011) and Lin et al. (2022) for details on the ADMM method.

Usage

```
f2sSP(
  vY,
  mX,
  mZ = NULL,
  M,
  group_weights = NULL,
  var_weights = NULL,
  standardize.data = TRUE,
  splOrd = 4,
  lambda = NULL,
  nlambda = 30,
  lambda.min.ratio = NULL,
  intercept = FALSE,
  overall.group = FALSE,
  control = list()
)
```

Arguments

`vY` a length-n vector of observations of the scalar response variable.

`mX` a $(n \times r)$ matrix of observations of the functional covariate.

`mZ` an $(n \times q)$ full column rank matrix of scalar predictors that are not penalized.

`M` number of elements of the B-spline basis vector $\varphi(t)$.

`group_weights` a vector of length G containing group-specific weights. The default is square root of the group cardinality, see Bernardi et al. (2022).

`var_weights` a vector of length M containing basis-specific weights. The default is a vector where each entry is the reciprocal of the number of groups including that basis. See Bernardi et al. (2022) for details.

`standardize.data` logical. Should data be standardized?

`splOrd` the order d of the spline basis.

`lambda` either a regularization parameter or a vector of regularization parameters. In this latter case the routine computes the whole path. If it is NULL values for lambda are provided by the routine.

`nlambda` the number of lambda values - default is 30.

`lambda.min.ratio`
: smallest value for lambda, as a fraction of the maximum lambda value. If $n > M$, the default is 0.0001, and if $n < M$, the default is 0.01.

`intercept`
: logical. If it is TRUE, a column of ones is added to the design matrix.

`overall.group`
: logical. If it is TRUE, an overall group including all penalized covariates is added.

`control`
: a list of control parameters for the ADMM algorithm. See 'Details'.

Value

A named list containing

sp.coefficients a length-M solution vector for the parameters ψ, which corresponds to the minimum in-sample MSE.

sp.coef.path an $(n_\lambda \times M)$ matrix of estimated ψ coefficients for each lambda.

sp.fun a length-r vector providing the estimated functional coefficient for $\psi(t)$.

sp.fun.path an $(n_\lambda \times r)$ matrix providing the estimated functional coefficients for $\psi(t)$ for each lambda.

coefficients a length-q solution vector for the parameters γ, which corresponds to the minimum in-sample MSE. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

coef.path an $(n_\lambda \times q)$ matrix of estimated γ coefficients for each lambda. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

lambda sequence of lambda.

lambda.min value of lambda that attains the minimum in-sample MSE.

mse in-sample mean squared error.

min.mse minimum value of the in-sample MSE for the sequence of lambda.

convergence logical. 1 denotes achieved convergence.

elapsedTime elapsed time in seconds.

iternum number of iterations.

When you run the algorithm, output returns not only the solution, but also the iteration history recording following fields over iterates,

objval objective function value.

r_norm norm of primal residual.

s_norm norm of dual residual.

eps_pri feasibility tolerance for primal feasibility condition.

eps_dual feasibility tolerance for dual feasibility condition.

Iteration stops when both `r_norm` and `s_norm` values become smaller than `eps_pri` and `eps_dual`, respectively.

Details

The control argument is a list that can supply any of the following components:

adaptation logical. If it is TRUE, ADMM with adaptation is performed. The default value is TRUE. See Boyd et al. (2011) for details.

rho an augmented Lagrangian parameter. The default value is 1.

tau.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 2. See Boyd et al. (2011) and Lin et al. (2022) for details.

mu.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 10. See Boyd et al. (2011) and Lin et al. (2022) for details.

abstol absolute tolerance stopping criterion. The default value is sqrt(sqrt(.Machine$double.eps)).

reltol relative tolerance stopping criterion. The default value is sqrt(.Machine$double.eps).

maxit maximum number of iterations. The default value is 100.

print.out logical. If it is TRUE, a message about the procedure is printed. The default value is TRUE.

References

Bernardi M, Canale A, Stefanucci M (2022). "Locally Sparse Function-on-Function Regression." *Journal of Computational and Graphical Statistics*, **0**(0), 1-15. doi:10.1080/10618600.2022.2130926, https://doi.org/10.1080/10618600.2022.2130926.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends® in Machine Learning*, **3**(1), 1-122. ISSN 1935-8237, doi:10.1561/2200000016, http://dx.doi.org/10.1561/2200000016.

Jenatton R, Audibert J, Bach F (2011). "Structured variable selection with sparsity-inducing norms." *J. Mach. Learn. Res.*, **12**, 2777–2824. ISSN 1532-4435.

Lin Z, Li H, Fang C (2022). *Alternating direction method of multipliers for machine learning*. Springer, Singapore. ISBN 978-981-16-9839-2; 978-981-16-9840-8, doi:10.1007/9789811698408, With forewords by Zongben Xu and Zhi-Quan Luo.

Examples

```
## generate sample data
set.seed(1)
n     <- 40
p     <- 18                                 # number of basis to GENERATE beta
r     <- 100
s     <- seq(0, 1, length.out = r)

beta_basis <- splines::bs(s, df = p, intercept = TRUE)    # basis
coef_data  <- matrix(rnorm(n*floor(p/2)), n, floor(p/2))
fun_data   <- coef_data %*% t(splines::bs(s, df = floor(p/2), intercept = TRUE))

x_0  <- apply(matrix(rnorm(p, sd=1),p,1), 1, fdaSP::softhresh, 1)  # regression coefficients
x_fun <- beta_basis %*% x_0
```

```
b     <- fun_data %*% x_fun + rnorm(n, sd = sqrt(crossprod(fun_data %*% x_fun ))/10)
l     <- 10^seq(2, -4, length.out = 30)
maxit <- 1000


## set the hyper-parameters
maxit          <- 1000
rho_adaptation <- TRUE
rho            <- 1
reltol         <- 1e-5
abstol         <- 1e-5

mod <- f2sSP(vY = b, mX = fun_data, M = p,
           group_weights = NULL, var_weights = NULL, standardize.data = FALSE, splOrd = 4,
             lambda = NULL, nlambda = 30, lambda.min = NULL, overall.group = FALSE,
             control = list("abstol" = abstol,
                            "reltol" = reltol,
                            "adaptation" = rho_adaptation,
                            "rho" = rho,
                            "print.out" = FALSE))

# plot coefficiente path
matplot(log(mod$lambda), mod$sp.coef.path, type = "l",
        xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "", bty = "n", lwd = 1.2)
```

`f2sSP_cv` *Cross-validation for Overlap Group Least Absolute Shrinkage and Selection Operator on scalar-on-function regression model*

Description

Overlap Group-LASSO for scalar-on-function regression model solves the following optimization problem

$$\min_{\psi,\gamma} \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \int x_i(t)\psi(t)dt - z_i^\intercal\gamma\right)^2 + \lambda\sum_{g=1}^{G}\|S_gT\psi\|_2$$

to obtain a sparse coefficient vector $\psi \in \mathbb{R}^M$ for the functional penalized predictor $x(t)$ and a coefficient vector $\gamma \in \mathbb{R}^q$ for the unpenalized scalar predictors $z_1, \dots, z_q$. The regression function is $\psi(t) = \varphi(t)^\intercal\psi$ where $\varphi(t)$ is a B-spline basis of order d and dimension M. For each group g, each row of the matrix $S_g \in \mathbb{R}^{d\times M}$ has non-zero entries only for those bases belonging to that group. These values are provided by the arguments `groups` and `group_weights` (see below). Each basis function belongs to more than one group. The diagonal matrix $T \in \mathbb{R}^{M\times M}$ contains the basis specific weights. These values are provided by the argument `var_weights` (see below). The regularization path is computed for the overlap group-LASSO penalty at a grid of values for the regularization parameter λ using the alternating direction method of multipliers (ADMM). See Boyd et al. (2011) and Lin et al. (2022) for details on the ADMM method.

Usage

```
f2sSP_cv(
  vY,
  mX,
  mZ = NULL,
  M,
  group_weights = NULL,
  var_weights = NULL,
  standardize.data = FALSE,
  splOrd = 4,
  lambda = NULL,
  lambda.min.ratio = NULL,
  nlambda = NULL,
  cv.fold = 5,
  intercept = FALSE,
  overall.group = FALSE,
  control = list()
)
```

Arguments

`vY`	a length-n vector of observations of the scalar response variable.
`mX`	a $(n \times r)$ matrix of observations of the functional covariate.
`mZ`	an $(n \times q)$ full column rank matrix of scalar predictors that are not penalized.
`M`	number of elements of the B-spline basis vector $\varphi(t)$.
`group_weights`	a vector of length G containing group-specific weights. The default is square root of the group cardinality, see Bernardi et al. (2022).
`var_weights`	a vector of length M containing basis-specific weights. The default is a vector where each entry is the reciprocal of the number of groups including that basis. See Bernardi et al. (2022) for details.
`standardize.data`	logical. Should data be standardized?
`splOrd`	the order d of the spline basis.
`lambda`	either a regularization parameter or a vector of regularization parameters. In this latter case the routine computes the whole path. If it is NULL values for lambda are provided by the routine.
`lambda.min.ratio`	smallest value for lambda, as a fraction of the maximum lambda value. If $n > M$, the default is 0.0001, and if $n < M$, the default is 0.01.
`nlambda`	the number of lambda values - default is 30.
`cv.fold`	the number of folds - default is 5.
`intercept`	logical. If it is TRUE, a column of ones is added to the design matrix.
`overall.group`	logical. If it is TRUE, an overall group including all penalized covariates is added.
`control`	a list of control parameters for the ADMM algorithm. See 'Details'.

Value

A named list containing

sp.coefficients a length-M solution vector solution vector for the parameters ψ, which corresponds to the minimum cross-validated MSE.

sp.fun a length-r vector providing the estimated functional coefficient for $\psi(t)$ corresponding to the minimum cross-validated MSE.

coefficients a length-q solution vector for the parameters γ, which corresponds to the minimum cross-validated MSE. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

lambda sequence of lambda.

lambda.min value of lambda that attains the minimum cross-validated MSE.

mse cross-validated mean squared error.

min.mse minimum value of the cross-validated MSE for the sequence of lambda.

convergence logical. 1 denotes achieved convergence.

elapsedTime elapsed time in seconds.

iternum number of iterations.

Iteration stops when both `r_norm` and `s_norm` values become smaller than `eps_pri` and `eps_dual`, respectively.

Details

The control argument is a list that can supply any of the following components:

adaptation logical. If it is TRUE, ADMM with adaptation is performed. The default value is TRUE. See Boyd et al. (2011) for details.

rho an augmented Lagrangian parameter. The default value is 1.

tau.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 2. See Boyd et al. (2011) and Lin et al. (2022) for details.

mu.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 10. See Boyd et al. (2011) and Lin et al. (2022) for details.

abstol absolute tolerance stopping criterion. The default value is sqrt(sqrt(.Machine$double.eps)).

reltol relative tolerance stopping criterion. The default value is sqrt(.Machine$double.eps).

maxit maximum number of iterations. The default value is 100.

print.out logical. If it is TRUE, a message about the procedure is printed. The default value is TRUE.

References

Bernardi M, Canale A, Stefanucci M (2022). "Locally Sparse Function-on-Function Regression." *Journal of Computational and Graphical Statistics*, **0**(0), 1-15. doi:10.1080/10618600.2022.2130926, https://doi.org/10.1080/10618600.2022.2130926.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends® in Machine Learning*, **3**(1), 1-122. ISSN 1935-8237, doi:10.1561/2200000016, http://dx.doi.org/10.1561/2200000016.

Jenatton R, Audibert J, Bach F (2011). "Structured variable selection with sparsity-inducing norms." *J. Mach. Learn. Res.*, **12**, 2777–2824. ISSN 1532-4435.

Lin Z, Li H, Fang C (2022). *Alternating direction method of multipliers for machine learning*. Springer, Singapore. ISBN 978-981-16-9839-2; 978-981-16-9840-8, doi:10.1007/9789811698408, With forewords by Zongben Xu and Zhi-Quan Luo.

Examples

```
## generate sample data and functional coefficients
set.seed(1)
n     <- 40
p     <- 18
r     <- 100
s     <- seq(0, 1, length.out = r)

beta_basis <- splines::bs(s, df = p, intercept = TRUE)    # basis
coef_data  <- matrix(rnorm(n*floor(p/2)), n, floor(p/2))
fun_data   <- coef_data %*% t(splines::bs(s, df = floor(p/2), intercept = TRUE))

x_0   <- apply(matrix(rnorm(p, sd=1),p,1), 1, fdaSP::softhresh, 1)
x_fun <- beta_basis %*% x_0

b     <- fun_data %*% x_fun + rnorm(n, sd = sqrt(crossprod(fun_data %*% x_fun ))/10)
l     <- 10^seq(2, -4, length.out = 30)
maxit <- 1000


## set the hyper-parameters
maxit          <- 1000
rho_adaptation <- TRUE
rho            <- 1
reltol         <- 1e-5
abstol         <- 1e-5

## run cross-validation
mod_cv <- f2sSP_cv(vY = b, mX = fun_data, M = p,
             group_weights = NULL, var_weights = NULL, standardize.data = FALSE, splOrd = 4,
              lambda = NULL, lambda.min = 1e-5, nlambda = 30, cv.fold = 5, intercept = FALSE,
                   control = list("abstol" = abstol,
                                  "reltol" = reltol,
                                  "adaptation" = rho_adaptation,
                                  "rho" = rho,
                                  "print.out" = FALSE))

### graphical presentation
plot(log(mod_cv$lambda), mod_cv$mse, type = "l", col = "blue", lwd = 2, bty = "n",
     xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "Prediction Error",
```

```
     ylim = range(mod_cv$mse - mod_cv$mse.sd, mod_cv$mse + mod_cv$mse.sd),
     main = "Cross-validated Prediction Error")
fdaSP::confband(xV = log(mod_cv$lambda), yVmin = mod_cv$mse - mod_cv$mse.sd,
                yVmax = mod_cv$mse + mod_cv$mse.sd)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
       col = "red", lwd = 1.0)

### comparison with oracle error
mod <- f2sSP(vY = b, mX = fun_data, M = p,
             group_weights = NULL, var_weights = NULL,
             standardize.data = FALSE, splOrd = 4,
             lambda = NULL, nlambda = 30,
             lambda.min = 1e-5, intercept = FALSE,
             control = list("abstol" = abstol,
                            "reltol" = reltol,
                            "adaptation" = rho_adaptation,
                            "rho" = rho,
                            "print.out" = FALSE))

err_mod <- apply(mod$sp.coef.path, 1, function(x) sum((x - x_0)^2))
plot(log(mod$lambda), err_mod, type = "l", col = "blue",
     lwd = 2, xlab = latex2exp::TeX("$\\log(\\lambda)$"),
     ylab = "Estimation Error", main = "True Estimation Error", bty = "n")
abline(v = log(mod$lambda[which(err_mod == min(err_mod))]), col = "red", lwd = 1.0)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
       col = "red", lwd = 1.0, lty = 2)
```

`lmSP` *Sparse Adaptive Overlap Group Least Absolute Shrinkage and Selection Operator*

Description

Sparse Adaptive overlap group-LASSO, or sparse adaptive group L_2-regularized regression, solves the following optimization problem

$$\min_{\beta,\gamma} \frac{1}{2}\|y - X\beta - Z\gamma\|_2^2 + \lambda\Big[(1-\alpha)\sum_{g=1}^{G}\|S_g T\beta\|_2 + \alpha\|T_1\beta\|_1\Big]$$

to obtain a sparse coefficient vector $\beta \in \mathbb{R}^p$ for the matrix of penalized predictors X and a coefficient vector $\gamma \in \mathbb{R}^q$ for the matrix of unpenalized predictors Z. For each group g, each row of the matrix $S_g \in \mathbb{R}^{n_g \times p}$ has non-zero entries only for those variables belonging to that group. These values are provided by the arguments `groups` and `group_weights` (see below). Each variable can belong to more than one group. The diagonal matrix $T \in \mathbb{R}^{p \times p}$ contains the variable-specific weights. These values are provided by the argument `var_weights` (see below). The diagonal matrix $T_1 \in \mathbb{R}^{p \times p}$ contains the variable-specific L_1 weights. These values are provided by the argument `var_weights_L1` (see below). The regularization path is computed for the sparse adaptive overlap group-LASSO penalty at a grid of values for the regularization parameter λ using the alternating direction method of multipliers (ADMM). See Boyd et al. (2011) and Lin et al. (2022) for details on the ADMM method. The regularization is a combination of L_2 and L_1 simultaneous constraints. Different specifications of the `penalty` argument lead to different models choice:

LASSO The classical Lasso regularization (Tibshirani, 1996) can be obtained by specifying $\alpha = 1$ and the matrix T_1 as the $p \times p$ identity matrix. An adaptive version of this model (Zou, 2006) can be obtained if T_1 is a $p \times p$ diagonal matrix of adaptive weights. See also Hastie et al. (2015) for further details.

GLASSO The group-Lasso regularization (Yuan and Lin, 2006) can be obtained by specifying $\alpha = 0$, non-overlapping groups in S_g and by setting the matrix T equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrix T is a $p \times p$ diagonal matrix of adaptive weights. See also Hastie et al. (2015) for further details.

spGLASSO The sparse group-Lasso regularization (Simon et al., 2011) can be obtained by specifying $\alpha \in (0, 1)$, non-overlapping groups in S_g and by setting the matrices T and T_1 equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrices T and T_1 are $p \times p$ diagonal matrices of adaptive weights.

OVGLASSO The overlap group-Lasso regularization (Jenatton et al., 2011) can be obtained by specifying $\alpha = 0$, overlapping groups in S_g and by setting the matrix T equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrix T is a $p \times p$ diagonal matrix of adaptive weights.

spOVGLASSO The sparse overlap group-Lasso regularization (Jenatton et al., 2011) can be obtained by specifying $\alpha \in (0, 1)$, overlapping groups in S_g and by setting the matrices T and T_1 equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrices T and T_1 are $p \times p$ diagonal matrices of adaptive weights.

Usage

```
lmSP(
  X,
  Z = NULL,
  y,
  penalty = c("LASSO", "GLASSO", "spGLASSO", "OVGLASSO", "spOVGLASSO"),
  groups,
  group_weights = NULL,
  var_weights = NULL,
  var_weights_L1 = NULL,
  standardize.data = TRUE,
  intercept = FALSE,
  overall.group = FALSE,
  lambda = NULL,
  alpha = NULL,
  lambda.min.ratio = NULL,
  nlambda = 30,
  control = list()
)
```

Arguments

`X` an $(n \times p)$ matrix of penalized predictors.

`Z` an $(n \times q)$ full column rank matrix of predictors that are not penalized.

`y` a length-n response vector.

`penalty` choose one from the following options: 'LASSO', for the or adaptive-Lasso penalties, 'GLASSO', for the group-Lasso penalty, 'spGLASSO', for the sparse group-Lasso penalty, 'OVGLASSO', for the overlap group-Lasso penalty and 'spOVGLASSO', for the sparse overlap group-Lasso penalty.

`groups` either a vector of length p of consecutive integers describing the grouping of the coefficients, or a list with two elements: the first element is a vector of length $\sum_{g=1}^{G} n_g$ containing the variables belonging to each group, where n_g is the cardinality of the g-th group, while the second element is a vector of length G containing the group lengths (see example below).

`group_weights` a vector of length G containing group-specific weights. The default is square root of the group cardinality, see Yuan and Lin (2006).

`var_weights` a vector of length p containing variable-specific weights. The default is a vector of ones.

`var_weights_L1` a vector of length p containing variable-specific weights for the L_1 penalty. The default is a vector of ones.

`standardize.data`
logical. Should data be standardized?

`intercept` logical. If it is TRUE, a column of ones is added to the design matrix.

`overall.group` logical. This setting is only available for the overlap group-LASSO and the sparse overlap group-LASSO penalties, otherwise it is set to NULL. If it is TRUE, an overall group including all penalized covariates is added.

`lambda` either a regularization parameter or a vector of regularization parameters. In this latter case the routine computes the whole path. If it is NULL values for lambda are provided by the routine.

`alpha` the sparse overlap group-LASSO mixing parameter, with $0 \leq \alpha \leq 1$. This setting is only available for the sparse group-LASSO and the sparse overlap group-LASSO penalties, otherwise it is set to NULL. The LASSO and group-LASSO penalties are obtained by specifying $\alpha = 1$ and $\alpha = 0$, respectively.

`lambda.min.ratio`
smallest value for lambda, as a fraction of the maximum lambda value. If $n > p$, the default is 0.0001, and if $n < p$, the default is 0.01.

`nlambda` the number of lambda values - default is 30.

`control` a list of control parameters for the ADMM algorithm. See 'Details'.

Value

A named list containing

sp.coefficients a length-p solution vector for the parameters β. If $n_\lambda > 1$ then the provided vector corresponds to the minimum in-sample MSE.

coefficients a length-q solution vector for the parameters γ. If $n_\lambda > 1$ then the provided vector corresponds to the minimum in-sample MSE. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

sp.coef.path an $(n_\lambda \times p)$ matrix of estimated β coefficients for each lambda of the provided sequence.

coef.path an $(n_\lambda \times q)$ matrix of estimated γ coefficients for each lambda of the provided sequence. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

lambda sequence of lambda.

lambda.min value of lambda that attains the minimum in sample MSE.

mse in-sample mean squared error.

min.mse minimum value of the in-sample MSE for the sequence of lambda.

convergence logical. 1 denotes achieved convergence.

elapsedTime elapsed time in seconds.

iternum number of iterations.

When you run the algorithm, output returns not only the solution, but also the iteration history recording following fields over iterates:

objval objective function value

r_norm norm of primal residual

s_norm norm of dual residual

eps_pri feasibility tolerance for primal feasibility condition

eps_dual feasibility tolerance for dual feasibility condition.

Iteration stops when both `r_norm` and `s_norm` values become smaller than `eps_pri` and `eps_dual`, respectively.

Details

The control argument is a list that can supply any of the following components:

adaptation logical. If it is TRUE, ADMM with adaptation is performed. The default value is TRUE. See Boyd et al. (2011) for details.

rho an augmented Lagrangian parameter. The default value is 1.

tau.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 2. See Boyd et al. (2011) for details.

mu.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 10. See Boyd et al. (2011) for details.

abstol absolute tolerance stopping criterion. The default value is sqrt(sqrt(.Machine$double.eps)).

reltol relative tolerance stopping criterion. The default value is sqrt(.Machine$double.eps).

maxit maximum number of iterations. The default value is 100.

print.out logical. If it is TRUE, a message about the procedure is printed. The default value is TRUE.

References

Bernardi M, Canale A, Stefanucci M (2022). "Locally Sparse Function-on-Function Regression." *Journal of Computational and Graphical Statistics*, **0**(0), 1-15. doi:10.1080/10618600.2022.2130926, https://doi.org/10.1080/10618600.2022.2130926.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends® in Machine Learning*, **3**(1), 1-122. ISSN 1935-8237, doi:10.1561/2200000016, `http://dx.doi.org/10.1561/2200000016`.

Hastie T, Tibshirani R, Wainwright M (2015). *Statistical learning with sparsity: the lasso and generalizations*, number 143 in Monographs on statistics and applied probability. CRC Press, Taylor & Francis Group, Boca Raton. ISBN 978-1-4987-1216-3.

Jenatton R, Audibert J, Bach F (2011). "Structured variable selection with sparsity-inducing norms." *J. Mach. Learn. Res.*, **12**, 2777–2824. ISSN 1532-4435.

Lin Z, Li H, Fang C (2022). *Alternating direction method of multipliers for machine learning*. Springer, Singapore. ISBN 978-981-16-9839-2; 978-981-16-9840-8, doi:10.1007/9789811698408, With forewords by Zongben Xu and Zhi-Quan Luo.

Simon N, Friedman J, Hastie T, Tibshirani R (2013). "A sparse-group lasso." *J. Comput. Graph. Statist.*, **22**(2), 231–245. ISSN 1061-8600, doi:10.1080/10618600.2012.681250.

Yuan M, Lin Y (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**(1), 49–67.

Zou H (2006). "The adaptive lasso and its oracle properties." *J. Amer. Statist. Assoc.*, **101**(476), 1418–1429. ISSN 0162-1459, doi:10.1198/016214506000000735.

Examples

```
### generate sample data
set.seed(2023)
n    <- 50
p    <- 30
X    <- matrix(rnorm(n*p), n, p)

### Example 1, LASSO penalty

beta <- apply(matrix(rnorm(p, sd = 1), p, 1), 1, fdaSP::softhresh, 1.5)
y    <- X %*% beta + rnorm(n, sd = sqrt(crossprod(X %*% beta)) / 20)

### set regularization parameter grid
lam   <- 10^seq(0, -2, length.out = 30)

### set the hyper-parameters of the ADMM algorithm
maxit      <- 1000
adaptation <- TRUE
rho        <- 1
reltol     <- 1e-5
abstol     <- 1e-5

### run example
mod <- lmSP(X = X, y = y, penalty = "LASSO", standardize.data = FALSE, intercept = FALSE,
```

```
            lambda = lam, control = list("adaptation" = adaptation, "rho" = rho,
                                         "maxit" = maxit, "reltol" = reltol,
                                         "abstol" = abstol, "print.out" = FALSE))

### graphical presentation
matplot(log(lam), mod$sp.coef.path, type = "l", main = "Lasso solution path",
        bty = "n", xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "")

### Example 2, sparse group-LASSO penalty

beta <- c(rep(4, 12), rep(0, p - 13), -2)
y    <- X %*% beta + rnorm(n, sd = sqrt(crossprod(X %*% beta)) / 20)

### define groups of dimension 3 each
group1 <- rep(1:10, each = 3)

### set regularization parameter grid
lam   <- 10^seq(1, -2, length.out = 30)

### set the alpha parameter
alpha <- 0.5

### set the hyper-parameters of the ADMM algorithm
maxit         <- 1000
adaptation    <- TRUE
rho           <- 1
reltol        <- 1e-5
abstol        <- 1e-5

### run example
mod <- lmSP(X = X, y = y, penalty = "spGLASSO", groups = group1, standardize.data = FALSE,
            intercept = FALSE, lambda = lam, alpha = 0.5,
            control = list("adaptation" = adaptation, "rho" = rho,
                           "maxit" = maxit, "reltol" = reltol, "abstol" = abstol,
                           "print.out" = FALSE))

### graphical presentation
matplot(log(lam), mod$sp.coef.path, type = "l", main = "Sparse Group Lasso solution path",
        bty = "n", xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "")
```

lmSP_cv — *Cross-validation for Sparse Adaptive Overlap Group Least Absolute Shrinkage and Selection Operator*

Description

Sparse Adaptive overlap group-LASSO, or sparse adaptive group L_2-regularized regression, solves the following optimization problem

$$\min_{\beta,\gamma} \frac{1}{2}\|y - X\beta - Z\gamma\|_2^2 + \lambda\Big[(1-\alpha)\sum_{g=1}^{G}\|S_g T\beta\|_2 + \alpha\|T_1\beta\|_1\Big]$$

to obtain a sparse coefficient vector $\beta \in \mathbb{R}^p$ for the matrix of penalized predictors X and a coefficient vector $\gamma \in \mathbb{R}^q$ for the matrix of unpenalized predictors Z. For each group g, each row of the matrix $S_g \in \mathbb{R}^{n_g \times p}$ has non-zero entries only for those variables belonging to that group. These values are provided by the arguments `groups` and `group_weights` (see below). Each variable can belong to more than one group. The diagonal matrix $T \in \mathbb{R}^{p\times p}$ contains the variable-specific weights. These values are provided by the argument `var_weights` (see below). The diagonal matrix $T_1 \in \mathbb{R}^{p\times p}$ contains the variable-specific L_1 weights. These values are provided by the argument `var_weights_L1` (see below). The regularization path is computed for the sparse adaptive overlap group-LASSO penalty at a grid of values for the regularization parameter λ using the alternating direction method of multipliers (ADMM). See Boyd et al. (2011) and Lin et al. (2022) for details on the ADMM method. The regularization is a combination of L_2 and L_1 simultaneous constraints. Different specifications of the `penalty` argument lead to different models choice:

LASSO The classical Lasso regularization (Tibshirani, 1996) can be obtained by specifying $\alpha = 1$ and the matrix T_1 as the $p \times p$ identity matrix. An adaptive version of this model (Zou, 2006) can be obtained if T_1 is a $p \times p$ diagonal matrix of adaptive weights. See also Hastie et al. (2015) for further details.

GLASSO The group-Lasso regularization (Yuan and Lin, 2006) can be obtained by specifying $\alpha = 0$, non-overlapping groups in S_g and by setting the matrix T equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrix T is a $p \times p$ diagonal matrix of adaptive weights. See also Hastie et al. (2015) for further details.

spGLASSO The sparse group-Lasso regularization (Simon et al., 2011) can be obtained by specifying $\alpha \in (0, 1)$, non-overlapping groups in S_g and by setting the matrices T and T_1 equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrices T and T_1 are $p \times p$ diagonal matrices of adaptive weights.

OVGLASSO The overlap group-Lasso regularization (Jenatton et al., 2011) can be obtained by specifying $\alpha = 0$, overlapping groups in S_g and by setting the matrix T equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrix T is a $p \times p$ diagonal matrix of adaptive weights.

spOVGLASSO The sparse overlap group-Lasso regularization (Jenatton et al., 2011) can be obtained by specifying $\alpha \in (0, 1)$, overlapping groups in S_g and by setting the matrices T and T_1 equal to the $p \times p$ identity matrix. An adaptive version of this model can be obtained if the matrices T and T_1 are $p \times p$ diagonal matrices of adaptive weights.

Usage

```
lmSP_cv(
  X,
  Z = NULL,
  y,
```

```
  penalty = c("LASSO", "GLASSO", "spGLASSO", "OVGLASSO", "spOVGLASSO"),
  groups,
  group_weights = NULL,
  var_weights = NULL,
  var_weights_L1 = NULL,
  cv.fold = 5,
  standardize.data = TRUE,
  intercept = FALSE,
  overall.group = FALSE,
  lambda = NULL,
  alpha = NULL,
  lambda.min.ratio = NULL,
  nlambda = 30,
  control = list()
)
```

Arguments

X	an $(n \times p)$ matrix of penalized predictors.
Z	an $(n \times q)$ full column rank matrix of predictors that are not penalized.
y	a length-n response vector.
penalty	choose one from the following options: 'LASSO', for the or adaptive-Lasso penalties, 'GLASSO', for the group-Lasso penalty, 'spGLASSO', for the sparse group-Lasso penalty, 'OVGLASSO', for the overlap group-Lasso penalty and 'spOVGLASSO', for the sparse overlap group-Lasso penalty.
groups	either a vector of length p of consecutive integers describing the grouping of the coefficients, or a list with two elements: the first element is a vector of length $\sum_{g=1}^{G} n_g$ containing the variables belonging to each group, where n_g is the cardinality of the g-th group, while the second element is a vector of length G containing the group lengths (see example below).
group_weights	a vector of length G containing group-specific weights. The default is square root of the group cardinality, see Yuan and Lin (2006).
var_weights	a vector of length p containing variable-specific weights. The default is a vector of ones.
var_weights_L1	a vector of length p containing variable-specific weights for the L_1 penalty. The default is a vector of ones.
cv.fold	the number of folds - default is 5.
standardize.data	logical. Should data be standardized?
intercept	logical. If it is TRUE, a column of ones is added to the design matrix.
overall.group	logical. This setting is only available for the overlap group-LASSO and the sparse overlap group-LASSO penalties, otherwise it is set to NULL. If it is TRUE, an overall group including all penalized covariates is added.
lambda	either a regularization parameter or a vector of regularization parameters. In this latter case the routine computes the whole path. If it is NULL values for lambda are provided by the routine.

`alpha` the sparse overlap group-LASSO mixing parameter, with $0 \leq \alpha \leq 1$. This setting is only available for the sparse group-LASSO and the sparse overlap group-LASSO penalties, otherwise it is set to NULL. The LASSO and group-LASSO penalties are obtained by specifying $\alpha = 1$ and $\alpha = 0$, respectively.

`lambda.min.ratio` smallest value for lambda, as a fraction of the maximum lambda value. If $n > p$, the default is 0.0001, and if $n < p$, the default is 0.01.

`nlambda` the number of lambda values - default is 30.

`control` a list of control parameters for the ADMM algorithm. See 'Details'.

Value

A named list containing

sp.coefficients a length-p solution vector for the parameters β. If $n_\lambda > 1$ then the provided vector corresponds to the minimum cross-validated MSE.

coefficients a length-q solution vector for the parameters γ. If $n_\lambda > 1$ then the provided vector corresponds to the minimum cross-validated MSE. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

sp.coef.path an $(n_\lambda \times p)$ matrix of estimated β coefficients for each lambda of the provided sequence.

coef.path an $(n_\lambda \times q)$ matrix of estimated γ coefficients for each lambda of the provided sequence. It is provided only when either the matrix Z in input is not NULL or the intercept is set to TRUE.

lambda sequence of lambda.

lambda.min value of lambda that attains the minimum cross-validated MSE.

mse cross-validated mean squared error.

min.mse minimum value of the cross-validated MSE for the sequence of lambda.

convergence logical. 1 denotes achieved convergence.

elapsedTime elapsed time in seconds.

iternum number of iterations.

When you run the algorithm, output returns not only the solution, but also the iteration history recording following fields over iterates:

objval objective function value

r_norm norm of primal residual

s_norm norm of dual residual

eps_pri feasibility tolerance for primal feasibility condition

eps_dual feasibility tolerance for dual feasibility condition.

Iteration stops when both `r_norm` and `s_norm` values become smaller than `eps_pri` and `eps_dual`, respectively.

Details

The control argument is a list that can supply any of the following components:

adaptation logical. If it is TRUE, ADMM with adaptation is performed. The default value is TRUE. See Boyd et al. (2011) for details.

rho an augmented Lagrangian parameter. The default value is 1.

tau.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 2. See Boyd et al. (2011) for details.

mu.ada an adaptation parameter greater than one. Only needed if adaptation = TRUE. The default value is 10. See Boyd et al. (2011) for details.

abstol absolute tolerance stopping criterion. The default value is sqrt(sqrt(.Machine$double.eps)).

reltol relative tolerance stopping criterion. The default value is sqrt(.Machine$double.eps).

maxit maximum number of iterations. The default value is 100.

print.out logical. If it is TRUE, a message about the procedure is printed. The default value is TRUE.

References

Bernardi M, Canale A, Stefanucci M (2022). "Locally Sparse Function-on-Function Regression." *Journal of Computational and Graphical Statistics*, **0**(0), 1-15. doi:10.1080/10618600.2022.2130926, https://doi.org/10.1080/10618600.2022.2130926.

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011). "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers." *Foundations and Trends® in Machine Learning*, **3**(1), 1-122. ISSN 1935-8237, doi:10.1561/2200000016, `http://dx.doi.org/10.1561/2200000016`.

Hastie T, Tibshirani R, Wainwright M (2015). *Statistical learning with sparsity: the lasso and generalizations*, number 143 in Monographs on statistics and applied probability. CRC Press, Taylor & Francis Group, Boca Raton. ISBN 978-1-4987-1216-3.

Jenatton R, Audibert J, Bach F (2011). "Structured variable selection with sparsity-inducing norms." *J. Mach. Learn. Res.*, **12**, 2777–2824. ISSN 1532-4435.

Lin Z, Li H, Fang C (2022). *Alternating direction method of multipliers for machine learning*. Springer, Singapore. ISBN 978-981-16-9839-2; 978-981-16-9840-8, doi:10.1007/9789811698408, With forewords by Zongben Xu and Zhi-Quan Luo.

Simon N, Friedman J, Hastie T, Tibshirani R (2013). "A sparse-group lasso." *J. Comput. Graph. Statist.*, **22**(2), 231–245. ISSN 1061-8600, doi:10.1080/10618600.2012.681250.

Yuan M, Lin Y (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**(1), 49–67.

Zou H (2006). "The adaptive lasso and its oracle properties." *J. Amer. Statist. Assoc.*, **101**(476), 1418–1429. ISSN 0162-1459, doi:10.1198/016214506000000735.

Examples

```
### generate sample data
set.seed(2023)
```

```
n    <- 50
p    <- 30
X    <- matrix(rnorm(n * p), n, p)

### Example 1, LASSO penalty

beta <- apply(matrix(rnorm(p, sd = 1), p, 1), 1, fdaSP::softhresh, 1.5)
y    <- X %*% beta + rnorm(n, sd = sqrt(crossprod(X %*% beta)) / 20)

### set the hyper-parameters of the ADMM algorithm
maxit      <- 1000
adaptation <- TRUE
rho        <- 1
reltol     <- 1e-5
abstol     <- 1e-5

### run cross-validation
mod_cv <- lmSP_cv(X = X, y = y, penalty = "LASSO",
                  standardize.data = FALSE, intercept = FALSE,
                  cv.fold = 5, nlambda = 30,
                  control = list("adaptation" = adaptation,
                                 "rho" = rho,
                                 "maxit" = maxit, "reltol" = reltol,
                                 "abstol" = abstol,
                                 "print.out" = FALSE))

### graphical presentation
plot(log(mod_cv$lambda), mod_cv$mse, type = "l", col = "blue", lwd = 2, bty = "n",
     xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "Prediction Error",
     ylim = range(mod_cv$mse - mod_cv$mse.sd, mod_cv$mse + mod_cv$mse.sd),
     main = "Cross-validated Prediction Error")
fdaSP::confband(xV = log(mod_cv$lambda), yVmin = mod_cv$mse - mod_cv$mse.sd,
                yVmax = mod_cv$mse + mod_cv$mse.sd)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
       col = "red", lwd = 1.0)

### comparison with oracle error
mod <- lmSP(X = X, y = y, penalty = "LASSO",
            standardize.data = FALSE,
            intercept = FALSE,
            nlambda = 30,
            control = list("adaptation" = adaptation,
                           "rho" = rho,
                           "maxit" = maxit, "reltol" = reltol,
                           "abstol" = abstol,
                           "print.out" = FALSE))

err_mod <- apply(mod$sp.coef.path, 1, function(x) sum((x - beta)^2))
plot(log(mod$lambda), err_mod, type = "l", col = "blue", lwd = 2,
     xlab = latex2exp::TeX("$\\log(\\lambda)$"),
     ylab = "Estimation Error", main = "True Estimation Error", bty = "n")
abline(v = log(mod$lambda[which(err_mod == min(err_mod))]), col = "red", lwd = 1.0)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
```

```
       col = "red", lwd = 1.0, lty = 2)

### Example 2, sparse group-LASSO penalty

beta <- c(rep(4, 12), rep(0, p - 13), -2)
y    <- X %*% beta + rnorm(n, sd = sqrt(crossprod(X %*% beta)) / 20)

### define groups of dimension 3 each
group1 <- rep(1:10, each = 3)

### set regularization parameter grid
lam   <- 10^seq(1, -2, length.out = 30)

### set the alpha parameter
alpha <- 0.5

### set the hyper-parameters of the ADMM algorithm
maxit         <- 1000
adaptation    <- TRUE
rho           <- 1
reltol        <- 1e-5
abstol        <- 1e-5

### run cross-validation
mod_cv <- lmSP_cv(X = X, y = y, penalty = "spGLASSO",
                  groups = group1, cv.fold = 5,
                  standardize.data = FALSE,  intercept = FALSE,
                  lambda = lam, alpha = 0.5,
                  control = list("adaptation" = adaptation,
                                 "rho" = rho,
                                 "maxit" = maxit, "reltol" = reltol,
                                 "abstol" = abstol,
                                 "print.out" = FALSE))

### graphical presentation
plot(log(mod_cv$lambda), mod_cv$mse, type = "l", col = "blue", lwd = 2, bty = "n",
     xlab = latex2exp::TeX("$\\log(\\lambda)$"), ylab = "Prediction Error",
     ylim = range(mod_cv$mse - mod_cv$mse.sd, mod_cv$mse + mod_cv$mse.sd),
     main = "Cross-validated Prediction Error")
fdaSP::confband(xV = log(mod_cv$lambda), yVmin = mod_cv$mse - mod_cv$mse.sd,
                yVmax = mod_cv$mse + mod_cv$mse.sd)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
       col = "red", lwd = 1.0)

### comparison with oracle error
mod <- lmSP(X = X, y = y,
            penalty = "spGLASSO",
            groups = group1,
            standardize.data = FALSE,
            intercept = FALSE,
            lambda = lam,
            alpha = 0.5,
            control = list("adaptation" = adaptation, "rho" = rho,
```

```
                        "maxit" = maxit, "reltol" = reltol, "abstol" = abstol,
                        "print.out" = FALSE))

err_mod <- apply(mod$sp.coef.path, 1, function(x) sum((x - beta)^2))
plot(log(mod$lambda), err_mod, type = "l", col = "blue", lwd = 2,
     xlab = latex2exp::TeX("$\\log(\\lambda)$"),
     ylab = "Estimation Error", main = "True Estimation Error", bty = "n")
abline(v = log(mod$lambda[which(err_mod == min(err_mod))]), col = "red", lwd = 1.0)
abline(v = log(mod_cv$lambda[which(mod_cv$lambda == mod_cv$lambda.min)]),
       col = "red", lwd = 1.0, lty = 2)
```

`softhresh` *Function to solve the soft thresholding problem*

Description

Function to solve the soft thresholding problem

Usage

```
softhresh(x, lambda)
```

Arguments

`x`	the data value.
`lambda`	the lambda value.

Value

the solution to the soft thresholding operator.

References

Hastie T, Tibshirani R, Wainwright M (2015). *Statistical learning with sparsity: the lasso and generalizations*, number 143 in Monographs on statistics and applied probability. CRC Press, Taylor & Francis Group, Boca Raton. ISBN 978-1-4987-1216-3.

Index