# Function sbr

## Description

Function for scalable Bayesian regression (SBR) and sparse SBR (SSBR) in normal linear models with multiple types (sources) of feature matrices (with K being the number of sources). When K = 1, SBR corresponds to standard ridge regression using one from the three available empirical Bayes estimators (see below) for the penalty parameter. For details see Perrakis, Mukherjee and the Alzheimers Disease Neuroimaging Initiative. (2018).

## Usage

sbr(y, X, trX, G, estimator = "MAP", sparsify = FALSE, relaxed = TRUE, sparse.control = 1, p.threshold = 5000, cov.blocks = 1000, parallel = FALSE, cl, L.optim = 1e-04, U.optim = 1e+04)

## Arguments

| | |
|---|---|
| y | a standardized response vector. |
| X | a standardized feature matrix (if $K = 1$) or a list of standardized feature matrices (if $K > 1$). |
| trX | (optional) the transpose matrix of X (if $K = 1$) or a list of transpose matrices (if $K > 1$). |
| G | the inner-product Gram matrix (if $K = 1$) or a list containing the multiple Gram matrices (if $K > 1$). |
| estimator | the estimator used for tuning the shrinkage levels. Available estimates are leave-one-out cross-validation ("CV"), maximum marginal likelihood ("ML") and the maximum-a-posteriori value ("MAP", default). |
| sparsify | logical, if TRUE the SSBR solution is calculated, default option is FALSE. |
| relaxed | logical, if TRUE (default) the relaxed SSBR solution is calculated, if FALSE the general SSBR solution is calculated. |
| sparse.control | numerical value for controlling the effect of sample size ($n$) on the resulting SSBR solution when relaxed = TRUE. Default option is 1 (no control). A recommended option for sparser solutions is sparse.control = log(n). |
| p.threshold | used for block-matrix computation of the main diagonal of the covariance matrix when sparsify = TRUE and relaxed = TRUE. It will be triggered for any source-matrix whose number of columns is larger than p.threshold. |
| cov.blocks | argument corresponding to block size (not the number of blocks) when the block-matrix computation is triggered (see above). Default option is 1000, i.e. blocks of dimensionality $1000 \times 1000$. |

parallel — logical, if parallel = TRUE the calculation of variance components required for the relaxed SSBR solution is performed in parallel. Default is FALSE.

cl — the number of cores to use when parallel = TRUE. Must be provided by the user.

L.optim — lower bound for the optimization procedure used to tune the shrinkage levels, default is 1e-04.

U.optim — upper bound for the optimization procedure used to tune the shrinkage levels, default is 1e+04.

## Value

coefficients — a 1-column matrix with the SBR beta estimates (when sparsify = FALSE) or a 2-column matrix with the SBR and SSBR beta estimates (when sparsify = TRUE). Note that the coefficients correspond to the standardized response variable and feature matrix.

sigma2 — the variance component (at the posterior mode).

lambda — the vector of penalty parameters.

lambdaEstimator — the estimator used for lambda.

duration — reported runtime.

## References

Perrakis, K., Mukherjee, S. and the Alzheimers Disease Neuroimaging Initiative. (2018). *Scalable Bayesian regression in high dimensions with multiple data sources*, arXiv:1710.00596 [**stat.ME**].

## Examples:

```
################## Example with 3 data sources #################

library(mvtnorm)
library(MCMCpack)

### GENERATION OF DATA ###

## sample size and number of predictors
n <- 50
p1 <- 10
p2 <- 100
p3 <- 300

## generation of covariance matrices and feature matrices
S1 <- riwish(p1, diag(p1))
S2 <- riwish(p2, diag(p2))
S3 <- riwish(p3, diag(p3))
X1 <- matrix(rmvnorm(n * p1, rep(0, p1), S1), n, p1)
X2 <- matrix(rmvnorm(n * p2, rep(0, p2), S2), n, p2)
X3 <- matrix(rmvnorm(n * p3, rep(0, p3), S3), n, p3)
```

```
## sparsity and generation of betas
s2 <- p2 * 0.3
s3 <- p3 * 0.01
non.zero2 <- sample(1:p2, s2)
non.zero3 <- sample(1:p3, s3)
b1 <- rnorm(10, 0, 2.5)
b2 <- rep(0, p2)
b2[non.zero2] <- rnorm(s2)
b3 <- rep(0, p3)
b3[non.zero3] <- rnorm(s3)

## generation of responce
mu <- X1 %*% b1 + X2 %*% b2 + X3 %*% b3
y <- rnorm(n, mu, sd=0.5)

## standardize
y <- scale(y)
X1 <- scale(X1)
X2 <- scale(X2)
X3 <- scale(X3)

## calculation of gram matrices
G1 <- X1 %*% t(X1); G2 <- X2 %*% t(X2); G3 <- X3 %*% t(X3)

## make lists
G <- list(G1, G2, G3)
X <- list(X1, X2, X3)

### RUN SBR/SSBR ###

# 1) SBR with the ML lambda-estimator

model1 <- sbr(y = y, X = X,G = G, estimator = 'ML')

# 2) relaxed SSBR with the ML lambda-estimator using block-matrix computations for the
# variances of X3 (since p3=300)

model2 <- sbr(y = y, X = X,G = G, estimator = 'ML', sparsify = TRUE, p.threshold =
    100, cov.blocks = 100)

# 3) SSBR with the ML lambda-estimator

model3 <- sbr(y = y, X = X, G = G, estimator = 'ML', sparsify = TRUE, relaxed = FALSE)

# 4) parallel computing for the configuration of model2

cores <- detectCores() - 1
cores <- makeCluster(cores)
registerDoParallel(cores)
model4 <- sbr(y = y, X = X, G = G, estimator = 'ML', parallel = TRUE, cl = cores,
    sparsify = TRUE, p.threshold = 100, cov.blocks = 100)
stopCluster(cores)

### EXTRACTING OUTPUT FROM A MODEL ###

coef(model3)    # SBR/SSBR coefficients (or alternatively model3$coeffients)
model3$lambda   # vector of lambdas
model3$sigma2   # error variance
model3$duration # runtime
```

# Function gram

## Description

Function for calculating the (inner-product) Gram matrix that allows for block-matrix multiplication.

## Usage

gram(X, trX, block = FALSE, block.size = 1000, show.progress = FALSE)

## Arguments

| | |
|---|---|
| X | a standardized feature matrix. |
| trX | (optional) the transpose matrix of X. |
| block | logical, block matrix computation is performed when TRUE, default option is FALSE. |
| block.size | used when block = TRUE. Default option is 1000, i.e. blocks of dimensionality $1000 \times 1000$. |
| show.progress | logical, when TRUE (and block = TRUE) the progress of the calculations is reported on the console. |

## Value

Returns the inner-product Gram matrix.

## Examples:

```
X <- matrix(rnorm(100 * 300), 100, 300)
G0 <- gram(X)                              # usual matrix multiplication
G1 <- gram(X, block=TRUE,block.size=100)  # block matrix multiplication
all.equal(G0, G1)
```

# Function gram.parallel

## Description

Function for calculating the (inner-product) Gram matrix that allows for block-matrix multiplication performed in parallel.

## Usage

gram.parallel(X, cl, ...)

### Arguments

X a standardized feature matrix.

cl the number of cores to use. Must be provided by the user.

... additional arguments passed from function gram. One can additionanly use block matrix multiplication within each core. Warning: in this case argument block.size must not exceed floor(ncol(X)/number of cores)−1.

### Value

Returns the inner-product Gram matrix.

### Examples:

```
X <- matrix(rnorm(100 * 300), 100, 300)
G0 <- gram(X)                          # usual matrix multiplication
cores <- detectCores() - 1
cores <- makeCluster(cores)
registerDoParallel(cores)
G1 <- gram.parallel(X, cl = cores)   # parallel matrix multiplication
stopCluster(cores)
all.equal(G0, G1)
```

## Function predict.sbr

### Description

Predict S3 method for objects of class 'sbr'.

### Usage

predict.sbr(object, newdata, coef = 'sbr')

### Arguments

object object an object of class 'sbr'.

newdata a (standardized) data matrix from which to predict.

coef choose whether to use the SBR beta estimates (default option "sbr") or the SSBR beta estimates (option "ssbr").

### Value

Returns a vector of predictions.

## Examples:

```
y <- rnorm(100)
X <- matrix(rnorm(100*300), 100, 300)
G <- gram(X)
model <- sbr(y = y, X = X, G = G, sparsify = TRUE)
predict1 <- predict(model, X[1:10, ], coef = 'sbr')
predict2 <- predict(model, X[1:10, ], coef = 'ssbr')
```