Package 'kernelPSI'

September 8, 2019

Title Post-Selection Inference for Nonlinear Variable Selection

Version 1.1.0

Date 2019-09-08

Description Different post-selection inference strategies for kernel selection, as described in ``kernelPSI: a Post-Selection Inference Framework for Nonlinear Variable Selection'', Slim et al., Proceedings of Machine Learning Research, 2019, <http://proceedings.mlr.press/v97/slim19a/slim19a.pdf>. The strategies rest upon quadratic kernel association scores to measure the association between a given kernel and an outcome of interest. The inference step tests for the joint effect of the selected kernels on the outcome. A fast constrained sampling algorithm is proposed to derive empirical p-values for the test statistics.

URL http://proceedings.mlr.press/v97/slim19a.html

Depends R (>= 3.5.0)

License GPL (>= 2)

Imports Rcpp (>= 1.0.1), CompQuadForm, pracma, kernlab, lmtest

Suggests bindata, knitr, rmarkdown, MASS, testthat

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

RoxygenNote 6.1.1

NeedsCompilation yes

Author Lotfi Slim [aut, cre],
Clément Chatelain [ctb],
Chloé-Agathe Azencott [ctb],
Jean-Philippe Vert [ctb]

Maintainer Lotfi Slim <lotfi.slim@mines-paristech.fr>

Repository CRAN

Date/Publication 2019-09-08 17:00:02 UTC

R topics documented:

`adaFOHSIC` *adaptively selects a subset of kernels in a forward fashion.*

Description

This function is similar to the `FOHSIC` function. The only difference lies in the adaptive selection of the number of causal kernels. First, similarly to `FOHSIC`, the order of selection of the n kernels in `K` is determined, and then, the size of the subset of ordered kernels is chosen. The size is chosen as to maximize the overall association with the kernel L.

Usage

```
adaFOHSIC(K, L)
```

Arguments

`K`	list of kernel similarity matrices
`L`	kernel similarity matrix for the outcome

Value

a list where the the first item `selection` is the order of selection of all kernels in the list `K` and the second item is the number of selected kernels.

Examples

```
n <- 50
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
L <- matrix(rnorm(n*p), nrow = n, ncol = p)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
L <-  L %*% t(L) / p
adaS <- adaFOHSIC(K, L)
print(names(adaS) == c("selection", "n"))
```

---

adaQ *models the forward selection of the kernels for the adaptive variant*

Description

Similarly to the fixed variant, the adaptive selection of the kernels in a forward fashion can also be modeled with a set of quadratic constraints. The constraints for adaptive selection can be split into two subsets. The first subset encodes the order of selection of the kernels, while the second subset encodes the selection of the number of the kernels. The two subsets are equally sized (`length(K) -1`) and are sequentially included in the output list.

Usage

```
adaQ(K, select, n)
```

Arguments

`K`	list kernel similarity matrices
`select`	integer vector containing the order of selection of the kernels in `K`. Typically, the `selection` field of the output of `FOHSIC`.
`n`	number of selected kernels. Typically, the `n` field of the output of `adaFOHSIC`.

Value

list of matrices modeling the quadratic constraints of the adaptive selection event

References

Loftus, J. R., & Taylor, J. E. (2015). Selective inference in regression models with groups of variables.

Examples

```
n <- 50
p <- 20
K <- replicate(8, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
L <- matrix(rnorm(n*p), nrow = n, ncol = p)
L <-  L %*% t(L) / p
adaS <- adaFOHSIC(K, L)
listQ <- adaQ(K, select = adaS[["selection"]], n = adaS[["n"]])
```

---

anovaLR *implements a scaled variant of the maximum likelihood ratio test*

Description

Compared to maxLR, the residual sum of squares (RSS) is scaled by the degrees of freedom of the model $df = n - k$, where n is the number of samples and k is the number of covariates. In maxLR, the RSS is instead averaged over n. Both estimators are asymptotically equivalent, with minor differences for finite samples. Further details in this link.

Usage

```
anovaLR(X, Y)
```

Arguments

X	covariate matrix
Y	response vector

Value

p-value of the test

See Also

Other LR test: maxLR

Examples

```
n <- 50
p <- 20
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- rnorm(n)
stat.anova <- anovaLR(X, Y)
```

FOHSIC *selects a fixed number of kernels which are most associated with the outcome kernel.*

Description

This function implements a forward algorithm for kernel selection. In the first step, the kernel which maximizes the HSIC measure with the outcome kernel `L` is selected. In the subsequent iterations, the kernel which, combined with the selected kernels maximizes the HSIC measure is selected. For the sum kernel combination rule, the forward algorithm can be simplified. The kernels which maximize the HSIC measure with the kernel `L` are selected in a descending order.

Usage

```
FOHSIC(K, L, mKernels = 1L)
```

Arguments

`K`	list of kernel similarity matrices
`L`	kernel similarity matrix for the outcome
`mKernels`	number of kernels to be selected

Details

`FOHSIC` implements the forward algorithm with a predetermined number of kernels `mKernels`. If the exact number of causal kernels is unavailable, the adaptive version `adaFOHSIC` should be preferred.

Value

an integer vector containing the indices of the selected kernels

Examples

```
n <- 50
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
L <- matrix(rnorm(n*p), nrow = n, ncol = p)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
L <-  L %*% t(L) / p
selection <- FOHSIC(K, L, 2)
```

`forwardQ` *models the forward selection event of a fixed number of kernels as a succession of quadratic constraints*

Description

The selection of the kernels with the forward algorithm implemented in `FOHSIC` can be represented as a set of quadratic constraints. This is owed to the quadratic form of the HSIC criterion. In this function, we determine the matrices of the corresponding constraints. The output is a list of matrices where the order is identical to the order of selection of the kernels. The matrices are computed such the associated constraint is nonnegative. For a length n of the list K, the total number of constraints is $n - 1$.

Usage

```
forwardQ(K, select)
```

Arguments

`K`	list kernel similarity matrices
`select`	integer vector containing the indices of the selected kernels

Value

list of matrices modeling the quadratic constraints of the selection event

Examples

```
n <- 50
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
listQ <- forwardQ(K, select = c(4, 1))
```

`HSIC` *Computes the HSIC criterion for two given kernels*

Description

The Hilbert-Schmidt Independence Criterion (HSIC) is a measure of independence between two random variables. If characteristic kernels are used for both variables, the HSIC is zero iff the variables are independent. In this function, we implement an unbiased estimator for the HSIC measure. Specifically, for two positive-definite kernels K and L and a sample size n, the unbiased HSIC estimator is:

$$HSIC(K, L) = \frac{1}{n(n-3)} \left[trace(KL) + \frac{1^\top K 1 1^\top L 1}{(n-1)(n-2)} - \frac{2}{n-2} 1^\top KL \right]$$

Usage

```
HSIC(K, L)
```

Arguments

K	first kernel similarity matrix
L	second kernel similarity matrix

Value

an unbiased estimate of the HSIC measure.

References

Song, L., Smola, A., Gretton, A., Borgwardt, K., & Bedo, J. (2007). Supervised Feature Selection via Dependence Estimation. https://doi.org/10.1145/1273496.1273600

Examples

```
n <- 50
p <- 20
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- matrix(rnorm(n*p), nrow = n, ncol = p)
K <-  X %*% t(X) / p
L <-  Y %*% t(Y) / p
uHSIC <- HSIC(K, L)
```

---

kernelPSI *computes a valid significance value for the effect of the selected kernels on the outcome*

Description

In this function, we compute an empirical p-value for the effect of a subset of kernels on the outcome. A number of statistics are supported in this function : ridge regression, kernel PCA and the HSIC criterion. The p-values are determined by comparing the statistic of the original response vector to those of the replicates. We use the sampleH function to sample replicates of the response in the acceptance region of the selection event.

Usage

```
kernelPSI(Y, K_select, constraints, method = "all", mu = 0,
  sigma = 1, lambda = 1, n_replicates = 5000, burn_in = 1000)
```

Arguments

Y	the response vector
K_select	list of selected kernel
constraints	list of quadratic matrices modeling the selection of the kernels in K_select
method	test statistic. Must be one of the following: ridge for log-likelihood ratio for ridge regression, pca for log-likelihood for kernel PCA, hsic for HSIC measures, or all to obtain significance values for all three former methods.
mu	mean of the response
sigma	standard deviation of the response
lambda	regularization parameter for ridge regression.
n_replicates	number of replicates for the hit-and-run sampler in sampleH
burn_in	number of burn_in iteration in sampleH

Details

For valid inference on hundreds of samples, we recommend setting the number of replicates to 50000 and the number of burn-in iterations to 10000. These ranges are to be increased for higher sample sizes.

Value

p-values for the chosen methods

Examples

```
n <- 30
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
Y <- rnorm(n)
L <- Y %*% t(Y)
selectK <- FOHSIC(K, L, mKernels = 2)
constraintFO <- forwardQ(K, selectK)
kernelPSI(Y, K[selectK], constraintFO, method = "ridge")
```

---

maxLR *implements the maximum likelihood ratio test*

Description

The maximum likelihood ratio test is a classical goodness-of-fit test for linear models. Mathematically speaking, the average residual sum of squares for an ordinary least squares (OLS) is approximated as a chi-square distribution to generate a p-value.

Usage

```
maxLR(X, Y)
```

Arguments

X	covariate matrix
Y	response vector

Details

The test is valid when the number of samples is larger than the number of covariates.

Value

p-value of the test

See Also

Other LR test: anovaLR

Examples

```
n <- 50
p <- 20
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
Y <- rnorm(n)
stat.likelihood <- maxLR(X, Y)
```

---

pcaLR *generates a closure for the computation of the likelihood ratio statistic for the kernel PCA prototype.*

Description

This function implements the same prototype statistics in the ridgeLR function, but for kernel principal component regression (see reference). In our simulations, we observed that this method underperforms the ridge prototype. The main benefit of this approach is the possibility of exact post-selection without the need for replicates sampling.

Usage

```
pcaLR(K, mu = 0, sigma = 1)
```

Arguments

K	a single or a list of selected kernel similarity matrices.
mu	marginal mean of the response Y
sigma	standard deviation of the response

Value

a closure for the calculation of the LR statistic for the kernel PCA prototype

References

Rosipal, R., Girolami, M., Trejo, L. J., & Cichocki, A. (2001). Kernel PCA for feature extraction and de-noising in nonlinear regression. Neural Computing and Applications, 10(3), 231–243.

See Also

Other prototype: ridgeLR

Examples

```
n <- 30
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
print(typeof(pcaLR(K, mu = 0, sigma = 1)) == "closure")
```

---

quadHSIC *Determines the quadratic form of the HSIC unbiased estimator*

Description

For a linear kernel of the outcome $L = Y^\top Y$, the unbiased HSIC estimator implemented in HSIC can be expressed as a quadratic form of the outcome Y i.e. $HSIC(K, L) = Y^\top Q(K) Y$. Here, the matrix Q only depends on the kernel similarity matrix K.

Usage

```
quadHSIC(K)
```

Arguments

K	kernel similarity matrix

Value

the matrix of the HSIC estimator quadratic form

Examples

```
n <- 50
p <- 20
X <- matrix(rnorm(n*p), nrow = n, ncol = p)
K <-  X %*% t(X) / p
Q <- quadHSIC(K)
```

`ridgeLR` *generates a closure for the computation of the likelihood ratio statistic for the ridge prototype.*

Description

The main inspiration for the kernel ridge prototype is the prototype concept developed in Reid (2018, see references). A prototype is a synthetic scalar variable that aggregates the effect of a set of variables in the outcome. Here, we extend this concept to kernels, where the prototype is the prediction of ridge regression with the selected kernels. In this function, we implement a likelihood ratio (LR) statistic to test for the effect of the the prototype on the outcome Y.

Usage

```
ridgeLR(K, mu = 0, sigma = 1, lambda = 1, tol = 1e-06,
  n_iter = 10000)
```

Arguments

`K`	a single or a list of selected kernel similarity matrices.
`mu`	mean of the response Y
`sigma`	standard deviation of the response
`lambda`	regularization parameter for the ridge prototype
`tol`	convergence tolerance used a stopping criterion for the Newton- Raphson algorithm
`n_iter`	maximum number of iterations for the Newton-Raphson algorithm

Details

To maximize the likelihood objective function, we implement in the output closure a Newton-Raphson algorithm that determines the maximum for each input vector Y.

For our post-selection inference framework, The output closure is used to compute the test statistics for both the replicates and the original outcome in order to derive empirical p-values.

Value

a closure for the calculation of the LR statistic for the ridge prototype

References

Reid, S., Taylor, J., & Tibshirani, R. (2018). A General Framework for Estimation and Inference From Clusters of Features. Journal of the American Statistical Association, 113(521), 280–293.

See Also

`pcaLR`

Other prototype: `pcaLR`

Examples

```
n <- 30
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
print(typeof(ridgeLR(K, mu = 0, sigma = 1, lambda = .1)) == "closure")
```

`sampleH` *samples within the acceptance region defined by the kernel selection event*

Description

To approximate the distribution of the test statistics, we iteratively sample replicates of the response in order to generate replicates of the test statistics. The response replicates are iteratively sampled within the acceptance region of the selection event. The goal of the constrained sampling is to obtain a valid post-selection distribution of the test statistic. To perform the constrained sampling, we develop a hit-and-run sampler based on the hypersphere directions algorithm (see references).

Usage

```
sampleH(A, initial, n_replicates, mu = 0, sigma = 1, n_iter = 1e+05,
  burn_in = 1000)
```

Arguments

`A`	list of matrices modeling the quadratic constraints of the selection event
`initial`	initialization sample. This sample must belong to the acceptance region given by `A`. In practice, this parameter is set to the outcome of the original dataset.
`n_replicates`	total number of replicates to be generated
`mu`	mean of the outcome
`sigma`	standard deviation of the outcome
`n_iter`	maximum number of rejections for the parameter λ in a single iteration
`burn_in`	number of burn-in iterations

Details

Given the iterative nature of the sampler, a large number of `n_replicates` and `burn_in` iterations is needed to correctly approximate the test statistics distributions.

For high-dimensional responses, and depending on the initialization, the sampler may not scale well to generate tens of thousands of replicates because of an intermediate rejection sampling step.

Value

a matrix with `n_replicates` columns where each column contains a sample within the acceptance region

References

Berbee, H. C. P., Boender, C. G. E., Rinnooy Ran, A. H. G., Scheffer, C. L., Smith, R. L., & Telgen, J. (1987). Hit-and-run algorithms for the identification of non-redundant linear inequalities. Mathematical Programming, 37(2), 184–207.

Belisle, C. J. P., Romeijn, H. E., & Smith, R. L. (2016). HIT-AND-RUN ALGORITHMS FOR GENERATING MULTIVARIATE DISTRIBUTIONS, 18(2), 255–266.

Examples

```
n <- 30
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
Y <- rnorm(n)
L <- Y %*% t(Y)
selection <- FOHSIC(K, L, 2)
constraintQ <- forwardQ(K, select = selection)
samples <- sampleH(A = constraintQ, initial = Y,
                   n_replicates = 50, burn_in = 20)
```

SKAT *implements the sequence kernel association test for GWAS data*

Description

The SKAT test is a quadratic test of association between a phenotype of interest and a genomic region. One of the main benefits of the SKAT test is the incorporation of nonlinear effects through the use of a kernel similarity matrix in the quadratic form. For instance, the identical-by-state (IBS) kernel which computes the number of identical alleles between two samples can be used.

Usage

```
SKAT(Y, K, sigma = 1)
```

Arguments

Y	response vector
K	list of kernel similarity matrices. The sum kernel is used in the quadratic form.
sigma	standard deviation of the response Y

Details

The null hypothesis in the SKAT test is the absence of effects of the SNPs within the region of interest and the outcome. Under the null, the distribution of the test statistic is a weighted sum of chi-square distributions whose quantiles are computed using the davies formula.

Value

p-value of the SKAT test

References

Wu, M. C., Lee, S., Cai, T., Li, Y., Boehnke, M., & Lin, X. (2011). Rare-variant association testing for sequencing data with the sequence kernel association test. American Journal of Human Genetics, 89(1), 82–93.

Examples

```
n <- 30
p <- 20
K <- replicate(5, matrix(rnorm(n*p), nrow = n, ncol = p), simplify = FALSE)
K <-  sapply(K, function(X) return(X %*% t(X) / dim(X)[2]), simplify = FALSE)
Y <- rnorm(n)
SKAT(Y, K)
```

Index