Package 'BSCRN'

December 7, 2019

Type Package

Title Classical and Bayesian Screening for Binary Classification

Version 1.0

Date 2019-11-14

Description This package is intended to be used for screening important variables in the binary classification setting. This should be useful for data sets with very large amounts of predictors. Also provides implementation of non parametric Bayesian tests for equality of distributions, and methods to draw from their predictive posterior.

License GPL-2

Imports Rcpp

Depends stats, rootSolve, parallel, foreach, doParallel

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.0.0

Suggests knitr, rmarkdown

VignetteBuilder knitr

LazyData true

Encoding UTF-8

R topics documented:

`BSCRN-package` *Classical and Bayesian Screening for Binary Classification*

Description

This package is intended to be used for screening important variables in the binary classification setting. This should be useful for data sets with very large amounts of predictors. Also provides implementation of non parametric Bayesian tests for equality of distributions, and methods to draw from their predictive posterior.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
~~ An overview of how to use the package, including the most important functions ~~

Author(s)

NA

Maintainer: NA

References

Original paper on SIS: Fan, J., & Lv, J. (2008). Sure independence screening for ultrahigh dimensional feature space. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70(5), 849-911.

First paper on Kolmogorov Smirnov Screening: Mai, Q., & Zou, H. (2012). The Kolmogorov filter for variable screening in high-dimensional binary classification. Biometrika, 100(1), 229-234. ~~ Literature or other references for background information ~~

Hanson's paper on inference of Polya Trees Hanson, T. (2006). Inference for Mixtures of Finite Polya Tree Models. Journal of the American Statistical Association, 101(476), 1548-1565. Retrieved from www.jstor.org/stable/27639772

Holmes et al.'s POlya Tree test Holmes, C. C., Caron, F., Griffin, J. E., & Stephens, D. A. (2015). Two-sample Bayesian nonparametric hypothesis testing. Bayesian Analysis, 10(2), 297-320.

Hanson and Chen's Polya Tree test Chen, Y., & Hanson, T. E. (2014). Bayesian nonparametric k-sample tests for censored and uncensored data. Computational Statistics & Data Analysis, 71, 335-346.

CVBF manuscript is in preperation, please contact author for current manuscript.

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

Examples

```
~~ simple examples of the most important functions ~~
```

`CVBFtestrsplit` *Compute a CVBF, a Bayes factor that checks if two data sets share the same distribution.*

Description

Compute a CVBF, a Bayes factor that checks if two data sets share the same distribution.

Usage

```
CVBFtestrsplit(
  dataset1,
  dataset2,
  trainsize1,
  trainsize2,
  seed = NULL,
  train1_ids = NULL,
  train2_ids = NULL
)
```

Arguments

`dataset1`	One dataset that we want to check if it has the same distribution as another data set
`dataset2`	Another dataset that we want to check if it has the same distribution as another data set
`trainsize1`	The training set size of dataset 1
`trainsize2`	The training set size of dataset 2
`seed`	The seed used to generate training set and validation sets for both of the data sets
`train1_ids`	Indices for the training set of dataset 1
`train2_ids`	Indices for the training set of dataset 2

Value

A list containing a log BF that tests whether two distributions are the same via CVBF, and the training sets used to generate the CVBF.

Examples

```
set.seed(100)
dataset1 = rnorm(200)
dataset2 = rnorm(200)
CVBF1 = CVBFtestrsplit(dataset1, dataset2, trainsize1 = 100, trainsize2 = 100)
CVBF1$logBF #Gives back the log Bayes factor
CVBF1$train1_ids #Gives back the training set of the first data set
CVBF1$train2_ids #Gives back the training set of the second data set
```

`gisettetrainlabs` *Labels for data that represent 4 or 9 in image recognition. It is unknown which label corresponds to which image. This data set was given as a blind classification problem by NIPS in 2003. The competition involves guessing what some validation labels were. These were the training labels ued for statisticains to train their models before submission.*

Description

Labels for data that represent 4 or 9 in image recognition. It is unknown which label corresponds to which image. This data set was given as a blind classification problem by NIPS in 2003. The competition involves guessing what some validation labels were. These were the training labels ued for statisticains to train their models before submission.

Usage

```
gisettetrainlabs
```

Format

A named matrix with 6000 rows and 1 column, each value is -1 or 1 and corresponds to whether or not an image is a 4 or 9 in the training data set.

Source

https://archive.ics.uci.edu/ml/datasets/Gisette

`gisettetrainpreds` *Predictors of the labels in gisettetrainlabs. Some of these correspond to pixel intensity measurements of some images, while others are random noise. It is uncertain which are noisy probes and which are pixel measurements.*

Description

Predictors of the labels in gisettetrainlabs. Some of these correspond to pixel intensity measurements of some images, while others are random noise. It is uncertain which are noisy probes and which are pixel measurements.

Usage

```
gisettetrainpreds
```

Format

A named matrix with 6000 rows and 5000 columns. It is zero inflated, as not every pixel is touched when a person writes a number.

Source

https://archive.ics.uci.edu/ml/datasets/Gisette

`gisettevalidlabs` *More labels for data that represent 4 or 9 in image recognition. It's unknown which label corresponds to which image. This data set was given as a blind classification problem by NIPS in 2003. The competition involves guessing what the validation labels were. These were released after the competition.*

Description

More labels for data that represent 4 or 9 in image recognition. It's unknown which label corresponds to which image. This data set was given as a blind classification problem by NIPS in 2003. The competition involves guessing what the validation labels were. These were released after the competition.

Usage

```
gisettevalidlabs
```

Format

A named matrix with 1000 rows and 1 column.

Source

https://archive.ics.uci.edu/ml/datasets/Gisette

`gisettevalidpreds` *Predictors of the labels in gisettevalidlabs. Some of these correspond to pixel intensity measurements of some images, while others are random noise. It is uncertain which are noisy probes and which are pixel measurements.*

Description

Predictors of the labels in gisettevalidlabs. Some of these correspond to pixel intensity measurements of some images, while others are random noise. It is uncertain which are noisy probes and which are pixel measurements.

Usage

```
gisettevalidpreds
```

Format

A named matrix with 1000 rows and 5000 columns.

Source

https://archive.ics.uci.edu/ml/datasets/Gisette

HallKernel *Compute Hall Kernel density estimate, given a training set and a set of values to evaluate on*

Description

Compute Hall Kernel density estimate, given a training set and a set of values to evaluate on

Usage

```
HallKernel(h, datagen2, x)
```

Arguments

h	Bandwidth parameter
datagen2	Training set for Hall KDE
x	Value (or values) to evaluate KDE on. Can be a scalar, a matrix, or a vector.

Value

An object of the same dimension as x that evaluates a KDE trained on datagen2 on everyone of the points in x

---

KHall *Title Hall Kernel evaluation*

Description

Title Hall Kernel evaluation

Usage

```
KHall(x)
```

Arguments

x	The parameter to evaluate the Hall Kernel density on

Value

Evaluation of K(X), where K is a heavy tailed kernel. For more information on K(x), see K_0 on Hall's paper on Kullback Leibler loss (Annals of Statistics 1957)

Examples

```
Khall(.1)
```

`laplace.kernH2c` *Compute Marginal Likelihoods for CVBF*

Description

Compute Marginal Likelihoods for CVBF

Usage

```
laplace.kernH2c(y, x, hhat, c)
```

Arguments

`y`	Validation Set
`x`	Training set
`hhat`	Bandwidth parameter that maximizes the log likelihood
`c`	A constant that is equal to the log likelihood + log prior evaluated at the maximum

Value

Evaluation of the CVBF marginal likelihood via Laplace Approximation. Also returns bandwidth that maximized log integrand, and hessian of log integrand at maximum.

Examples

```
dataset1 = rnorm(100)
DT = dataset1[1:50]
DV = dataset1[51:100]
likvec = function(h) {sum(log(HallKernel(h,datagen2 = DT, x = DT)))}
bwlikcy = optimize(f = function(h){  likvec(h)}, lower = 0, upper = 10, maximum = TRUE)
ExpectedKernML2 = laplace.kernH2c(y = DT, x = DV, hhat = bwlikcy$maximum, c= bwlikcy$objective + logpriorused(h
ExpectedKernML2
```

logintegrand.Hall *Compute log of the integrand of the Marginal likelihood for CVBF*

Description

Compute log of the integrand of the Marginal likelihood for CVBF

Usage

```
logintegrand.Hall(h, y, x, hhat)
```

Arguments

h	Bandwidth parameter, this is the variable that is being integrated over
y	Validation set to evaluate the likelihood over
x	Training set to build the KDE
hhat	Parameter that specifies where prior should be centered

Value

Evaluation of the integrand at a particular log likelihood value.

Examples

```
dataset1 = rnorm(100)
DT = dataset1[1:50]
DV = dataset1[51:100]
logintegrand.Hall(.01, DT, DV, .1)
```

loglike.KHall *Evaluate a log likelihood using the Hall Kernel*

Description

Evaluate a log likelihood using the Hall Kernel

Usage

```
loglike.KHall(h, y, x)
```

Arguments

h	A bandwidth or vector of bandwidths tot ry out
y	A validation set to evaluate the KDE on
x	A training set to build the Hall kernel density estimate on

Value

-log likelihood evaluation, where the likelihood is constructed using the training data

Examples

```
dataset1 = rnorm(100)
DT = dataset1[1:50]
DV = dataset1[51:100]
loglike.Khall(.01, DT, DV)
```

`logpriorused` *Evaluate the log prior on one of the data sets. Called by other functions.*

Description

Evaluate the log prior on one of the data sets. Called by other functions.

Usage

```
logpriorused(h, hhat)
```

Arguments

`h`	A value to evaluate the prior on
`hhat`	Tuning parameter for the prior

Value

Evaluates a particular type of prior placed on the bandwidth for CVBF.

Examples

```
logpriorused(.1,.1)
```

`ParScreenVars` *Screen a data set for important functions in parallel*

Description

Screen a data set for important functions in parallel

Usage

```
ParScreenVars(
  datasetX,
  datasetY,
  method = "SIS",
  ncores = 1,
  cutoff = NULL,
  train1ids = NULL,
  trainsize1 = NULL,
  trainsize2 = NULL,
  train2ids = NULL,
```

```
  seed = NULL,
  Ginv = NULL,
  c = NULL,
  leveltot = NULL,
  PTscale = TRUE
)
```

Arguments

datasetX	A matrix containing values that are predictors for the Y values
datasetY	A vector containg the class that each predictor corresponds to. For now can only handle binary responses.
method	A string containing the type of screening to do. Can be "SIS", "KS", "CVBF" or "PT"
ncores	A integer that corresponds to the number of cores to be used for parallelizing computation
cutoff	A real number that corresponds either to an alpha value for testing or a cutoff value on how large the Bayes factor needs to be to conclude a difference exists.
train1ids	A vector of ids that correspond to which observations to use for the training set for the first data set
trainsize1	Size of the training set for one of the classes for CVBF
trainsize2	Size of the training set for the other one of the classes for CVBF
train2ids	A vector of ids that correspond to which observations to use for the training set for the second data set
seed	A seed for CVBF based screening, can use this to reproduce results instead of train_ids.
Ginv	A function to compute quantiles with for Polya tree.
c	Tuning parameter for Polya tree, signifies how impactful prior should be.
leveltot	Depth of Polya tree to construct if Polya tree based screening is type of screening chosen
PTscale	A True / false variable. Should columns be standardized before proceeding with Polya tree based screening? Default is to screen as recommended by authors.

Value

A list of variables that are interpreted to be important

Examples

```
data(gisettetrainlabs)
data(gisettetrainpreds)
nworkers = detectCores()
ImpVarsSIS1 = ParScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method = "
length(ImpVarsSIS1$varspicked)
ImpVarsKS1 = ParScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method = "K
length(ImpVarsKS1$varspicked)
ImpVarsPT1 = ParScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method = "P
#Only do on first 500
length(ImpVarsPT1$varspicked)
hist(ImpVarsPT1$logBFlist)
```

```
ImpVarsCVBF1 = ParScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method =
length(ImpVarsCVBF1$varspicked)
```

`PolyaTreeBFcons` *Polya Tree test for checking if two data sets share the same distribution using Polya tree objects.*

Description

Polya Tree test for checking if two data sets share the same distribution using Polya tree objects.

Usage

```
PolyaTreeBFcons(PolyaTreePriorLik1, PolyaTreePriorLik2)
```

Arguments

`PolyaTreePriorLik1`
: An object constructed by PolyaTreePriorLikCons for a dataset.

`PolyaTreePriorLik2`
: Another object constructed by PolyaTreePriorLikCons for another dataset.

Value

A scalar, which corresponds to the log BF of the test. Another vector is given which corresponds to the contribution of the log BF at each level.

Examples

```
set.seed(100)
dataset1 = rnorm(200)
dataset2 = rnorm(200)
PTmodel1 = PolyaTreePriorLikCons(datasetX = dataset1, Ginv = qnorm)
PTmodel2 = PolyaTreePriorLikCons(datasetX = dataset2, Ginv = qnorm)
PTresults = PolyaTreeBFcons(PTmodel1, PTmodel2)
```

`PolyaTreePredDraws` *Generate draws from a Polya Tree model's Predictive distribution*

Description

Generate draws from a Polya Tree model's Predictive distribution

Usage

```
PolyaTreePredDraws(PolyaTreePriorLik, ndraw = 2000)
```

Arguments

PolyaTreePriorLik	An object constructed by PolyaTreePriorLikCons for a dataset.
ndraw	Number of draws desired from Predictive Polya Tree distribution.

Value

A list of draws from the Predictive Polya Tree Posterior

Examples

```
set.seed(100)
dataset1 = rnorm(200)
PTmodel1 = PolyaTreePriorLikCons(datasetX = dataset1, Ginv = qnorm)
Predictiveposteriordraws = PolyaTreePredDraws(PTmodel1, ndraw = 400)
plot(density(Predictiveposteriordraws))
```

PolyaTreePriorLikCons *Construct a Polya tree object for a data set*

Description

Construct a Polya tree object for a data set

Usage

```
PolyaTreePriorLikCons(datasetX, Ginv = NULL, c = 1, leveltot = NULL)
```

Arguments

datasetX	A dataset to compute the Polya Tree prior on
Ginv	A quantile function of some distribution, use to make bins
c	A scalar. The higher this is, the more influential the prior is on the data set.
leveltot	The number of levels the Polya Tree should go down.

Value

A list of vectors called alphalist, a list of vectors called splitlist, c, leveltot, and Ginv. Alphalist is used to construct a prior. Splitlist is used to construct a likelihood. Call the collection of these a Polya Tree object.

Examples

```
set.seed(100)
dataset1 = rnorm(200)
PTmodel1 = PolyaTreePriorLikCons(datasetX = dataset1)
#Ptmodel1 can be called by other methods in the package to use it
```

PolyaTreetest *Compute a Bayes factor that checks to see if two data sets share the same distribution by Polya Tree*

Description

Compute a Bayes factor that checks to see if two data sets share the same distribution by Polya Tree

Usage

```
PolyaTreetest(datasetX, datasetY, Ginv = NULL, c = NULL, leveltot = NULL)
```

Arguments

datasetX	A set of data (or in the case of screening a predictor corresponding to one class), one of the data sets we want to check if its distribution is the same as dataset Y's.
datasetY	Another set of data (or in the case of screening a predictor corresponding to another class), the other data set we're comparing to datasetX
Ginv	A function that can compute quantiles of some distribution. The default is qnorm. Can specify another function, needs to be able to take in a number between 0 and 1 and return back some positive value. It needs to be a quantile function.
c	A tuning parameter corresponding to how influential the prior should be. Authors recommend to set to 1. Can change from 1. The larger the tuning parameter the more influential the prior. The smaller the tuning parameter the less influential the prior.
leveltot	Total number of levels deep the tree should go. 9 is given as a default. Some authors recommend going to $log_2(sample size)$, but doesn't need to be done. The deeper the tree the more computation that is required.

Value

Returns a list. The log BF component is a scalar that corresponds to the log BF of the computed test, and a vector which correspons to the contribution of the log BF at each level.

Examples

```
set.seed(100)
dataset1 = rnorm(200)
dataset2 = rnorm(200)
PTtest1 = PolyaTreetest(dataset1, dataset2)
PTtest1$logBF #Gives back the log Bayes factor
PTtest1$logBFcont #Gives back the contributions of the log Bayes factor for different levels of the tree. Summin
PTtest2 = PolyaTreetest(dataset1, dataset2, Ginv = qnorm, leveltot = 10) #Can fill in arguments to suit data set
```

PredCVBFDens *Compute a Predictive Posterior function given a sequence of bandwidths from the predictive posterior*

Description

Compute a Predictive Posterior function given a sequence of bandwidths from the predictive posterior

Usage

```
PredCVBFDens(bwvec, XT1)
```

Arguments

bwvec	A vector of bandwidths, can come from either of the methods that draw bandwidths from the posterior.
XT1	The training set

Value

A function that evaluates the predictive posterior at particular values

Examples

```
set.seed(500)
datasetsample1 = rnorm(600)
trainingindices1 = sample(1:600, size = 300)
XT1 = datasetsample1[trainingindices1]
XV1 = datasetsample1[-trainingindices1]
predbwvec1 = PredCVBFIndepMHbw(ndraw = 500, maxIter = 5000, XT1 = XT1, XV1 = XV1)
predpost = PredCVBFDens(predbwvec1, XT1)
plot(seq(from = -3, to = 3, by = .1), Predpost(predbwvec1, XT1)(seq(from = -3, to = 3, by = .1)))
```

PredCVBFIndepMHbw *Draw bandwidths from CVBF predictive posterior by independent Metropolis Hasting sampling*

Description

Draw bandwidths from CVBF predictive posterior by independent Metropolis Hasting sampling

Usage

```
PredCVBFIndepMHbw(
  ndraw = 100,
  propsd = NULL,
  maxIter = 10000,
  XT1,
  XV1,
  startingbw = NULL
)
```

Arguments

ndraw	Number of unique draws desired for the bandwidth parameter from the posterior.
propsd	A tuning parameter, corresponds to what proposal standard deviation should be for when using MH to traverse the posterior. We give a decent theoretical default. May need to be altered if performance is bad.
maxIter	The max number of MH iterations to try. Do not set to be too large. It will kick the code out if acceptance rates for MH are small.
XT1	Training set for a data set
XV1	Validation set for a data set
startingbw	A value to start the MH chain at. If not provided, starts at posterior mode. All proposals will be drawn from a distribution whose center is startingbw. This is normally a bad idea, but the posterior is some type of unimodal distribution, so this is actually effective.

Value

A list of bandwidths that come from the posterior distribution. This will be larger than ndraw, as some draws will be repeats.

Examples

```
set.seed(500)
datasetsample1 = rnorm(600)
trainingindices1 = sample(1:600, size = 300)
XT1 = datasetsample1[trainingindices1]
XV1 = datasetsample1[-trainingindices1]
predbwvec1 = PredCVBFIndepMHbw(ndraw = 500, maxIter = 5000, XT1 = XT1, XV1 = XV1)
```

PredCVBFMHbw *Draw bandwidths from CVBF predictive posterior by Metropolis Hasting sampling*

Description

Draw bandwidths from CVBF predictive posterior by Metropolis Hasting sampling

Usage

```
PredCVBFMHbw(
  ndraw = 100,
  propsd = NULL,
  maxIter = 10000,
  XT1,
  XV1,
  startingbw = NULL
)
```

Arguments

ndraw	Number of unique draws desired for the bandwidth parameter from the posterior.
propsd	A tuning parameter, corresponds to what proposal standard deviation should be for when using MH to traverse the posterior. Should be chosen with care to ensure good mixing.
maxIter	The max number of MH iterations to try. Do not set to be too large. It will kick the code out if acceptance rates for MH are small.
XT1	Training set for a data set
XV1	Validation set for a data set
startingbw	A value to start the MH chain at. If not provided, starts at posterior mode.

Value

A list of bandwidths that come from the posterior distribution. This will be larger than ndraw, as some draws will be repeats.

Examples

```
set.seed(500)
datasetsample1 = rnorm(600)
trainingindices1 = sample(1:600, size = 300)
XT1 = datasetsample1[trainingindices1]
XV1 = datasetsample1[-trainingindices1]
predbwvec1 = PredCVBFMHbw(ndraw = 500, maxIter = 5000, XT1 = XT1, XV1 = XV1)
```

RcppArmadillo-Functions

Set of functions in example RcppArmadillo package

Description

These four functions are created when `RcppArmadillo.package.skeleton()` is invoked to create a skeleton packages.

Usage

```
rcpparma_hello_world()
rcpparma_outerproduct(x)
rcpparma_innerproduct(x)
rcpparma_bothproducts(x)
```

Arguments

x	a numeric vector

Details

These are example functions which should be largely self-explanatory. Their main benefit is to demonstrate how to write a function using the Armadillo C++ classes, and to have to such a function accessible from R.

Value

`rcpparma_hello_world()` does not return a value, but displays a message to the console.

`rcpparma_outerproduct()` returns a numeric matrix computed as the outer (vector) product of `x`.

`rcpparma_innerproduct()` returns a double computer as the inner (vector) product of `x`.

`rcpparma_bothproducts()` returns a list with both the outer and inner products.

Author(s)

Dirk Eddelbuettel

References

See the documentation for Armadillo, and RcppArmadillo, for more details.

Examples

```
x <- sqrt(1:4)
rcpparma_innerproduct(x)
rcpparma_outerproduct(x)
```

SeqScreenVars *Screen a data set for important functions sequentially (not in parallel)*

Description

Screen a data set for important functions sequentially (not in parallel)

Usage

```
SeqScreenVars(
  datasetX,
  datasetY,
  method = "SIS",
  cutoff = NULL,
  train1ids = NULL,
  trainsize1 = NULL,
  trainsize2 = NULL,
  train2ids = NULL,
  seed = NULL,
  Ginv = NULL,
  c = NULL,
  leveltot = NULL,
  PTscale = TRUE
)
```

Arguments

datasetX	A matrix containing values that are predictors for the Y values
datasetY	A vector containg the class that each predictor corresponds to. For now can only handle binary responses.
method	A string containing the type of screening to do. Can be "SIS", "KS", "CVBF" or "PT"
cutoff	A real number that corresponds either to an alpha value for testing or a cutoff value on how large the Bayes factor needs to be to conclude a difference exists.
train1ids	A vector of ids that correspond to which observations to use for the training set for the first data set
trainsize1	Size of the training set for one of the classes for CVBF
trainsize2	Size of the training set for the other one of the classes for CVBF
train2ids	A vector of ids that correspond to which observations to use for the training set for the second data set
seed	A seed for CVBF based screening, can use this to reproduce results instead of train1ids or train2ids.
Ginv	A function to compute quantiles with for Polya tree.
c	Tuning parameter for Polya tree, signifies how impactful prior should be.
leveltot	Depth of Polya tree to construct if Polya tree based screening is type of screening chosen
PTscale	A True / false variable. Should columns be standardized before proceeding with Polya tree based screening? Default is to screen as recommended by authors.

Value

A list of variables that are interpreted to be important

Examples

```
data(gisettetrainlabs)
data(gisettetrainpreds)
ImpVarsSIS1 = SeqScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method = "
length(ImpVarsSIS1$varspicked)
ImpVarsKS1 = SeqScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method = "K
length(ImpVarsKS1$varspicked)
ImpVarsPT1 = SeqScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method = "P
#Only do on first 500
length(ImpVarsPT1$varspicked)
hist(ImpVarsPT1$logBFlist)
ImpVarsCVBF1 = SeqScreenVars(datasetX = gisettetrainpreds[, 1:500], datasetY = gisettetrainlabs[,1], method =
length(ImpVarsCVBF1$varspicked)
```

Index