Package ‘HPLB’

January 20, 2025

Type Package

Title High-Probability Lower Bounds for the Total Variance Distance

Version 1.0.0

Author Loris Michel, Jeffrey Naef

Maintainer Loris Michel <`michel@stat.math.ethz.ch`>

Description An implementation of high-probability lower bounds for the total variance distance as introduced in Michel & Naef & Meinshausen (2020) <arXiv:2005.06006>. An estimated lower-bound (with high-probability) on the total variation distance between two probability distributions from which samples are observed can be obtained with the function HPLB.

License GPL-3

Encoding UTF-8

LazyData true

Depends data.table, stats, graphics

Suggests testthat, fields, ranger, distrEx

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-09 12:50:02 UTC

Contents

`boundingOperation` *Bounding Operation*

Description

Bounding Operation

Usage

```
boundingOperation(v, left, right, m, n)
```

Arguments

`v`	a a numeric value giving an ordering permutation of 1 to m+n.
`left`	a numeric value giving the number of witnesses left.
`right`	a numeric value giving the number of witnesses right.
`m`	a numeric value, the number of observations left.
`n`	a numeric value, the number of observations right.

Value

a cumulative counting function represented as a numeric vector.

`empiricalBF` *Empirical Bounding Functions*

Description

Empirical Bounding Functions

Usage

```
empiricalBF(tv.seq, nsim = 1000, m = 100, n = 100, alpha = 0.05)
```

Arguments

`tv.seq`	a vector of total variation values between 0 and 1.
`nsim`	a numeric value giving the number of repetitions.
`m`	a numeric value, the number of observations left.
`n`	a numeric value, the number of observations right.
`alpha`	a numeric value giving the type-I error level.

Value

a list of empirical bounding functions indexed by the tv.seq (in the respective order).

`HPLB`	*High Probability Lower Bounds (HPLB) for the Total Variation Distance (TV) Based on Finite Samples*

Description

Implementations of different HPLBs for TV as described in (Michel et al., 2020).

Usage

```
HPLB(
  t,
  rho,
  s = 0.5,
  estimator.type = "adapt",
  alpha = 0.05,
  tv.seq = seq(from = 0, to = 1, by = 1/length(t)),
  custom.bounding.seq = NULL,
  direction = rep("left", length(s)),
  cutoff = 0.5,
  verbose.plot = FALSE,
  seed = 0,
  ...
)
```

Arguments

`t`	a numeric vector value corresponding to a natural ordering of the observations. For a two-sample test 0-1 numeric values values should be provided.
`rho`	a numeric vector value providing an ordering. This could be a binary classifier, a regressor, a witness function from a MMD kernel or anything else that would witness a distributional difference.
`s`	a numeric vector value giving split points on t.
`estimator.type`	a character value indicating which estimator to use. One option out of: • `adapt`:adaptive binary classification estimator (asymptotic bounding function) • `bayes`:binary classification estimator • `bayes_finite_sample`:binary classification finite sample estimator • `adapt_empirical`:adaptive binary classification estimator (simulation-based bounding function) • `adapt_custom`:adaptive binary classificatrion estimator (user-defined bounding function) • `adapt_dwit`:adaptive binary classificatrion estimator (for distributional witnesses estimation)
`alpha`	a numeric value giving the overall type-I error control level.

`tv.seq`	a sequence of values between 0 and 1 used as the grid search for the total variation distance in case of tv-search.
`custom.bounding.seq`	a list of bounding functions respecting the order of tv.seq used in case of estimator.type "custom-tv-search".
`direction`	a character vector value made of "left" or "right" giving which distribution witness count to estimate (t<=s or t>s?).
`cutoff`	a numeric value. This is the cutoff used if bayes estimators are used. The theory suggests to use 1/2 but this can be changed.
`verbose.plot`	a boolean value for additional plots.
`seed`	an integer value. The seed for reproducibility.
`...`	additional parameters for the function `empiricalBF`.

Value

a `list` containing the relevant lower bounds estimates. For the total variation distance the relevant entry is `tvhat`.

Author(s)

Loris Michel, Jeffrey Naef

References

L. Michel, J. Naef and N. Meinshausen (2020). High-Probability Lower Bounds for the Total Variation Distance

Examples

```
## libs
library(HPLB)
library(ranger)
library(distrEx)

## reproducibility
set.seed(0)

## Example 1: TV lower bound based on two samples (bayes estimator), Gaussian mean-shift example

n <- 100
means <- rep(c(0,2), each = n / 2)
x <- stats::rnorm(n, mean = means)
t <- rep(c(0,1), each = n / 2)

bayesRate <- function(x) {
  return(stats::dnorm(x, mean = 2) /
    (stats::dnorm(x, mean = 2) + stats::dnorm(x, mean = 0)))
}
```

```
# estimated HPLB
tvhat <- HPLB(t = t, rho = bayesRate(x), estimator.type = "bayes")
# true TV
TotalVarDist(e1 = Norm(2,1), e2 = Norm(0,1))

## Example 2: optimal mixture detection (adapt estimator), Gaussian mean-shift example

n <- 100
mean.shift <- 2
t.train <- runif(n, 0 ,1)
x.train <- ifelse(t.train>0.5, stats::rnorm(n, mean.shift), stats::rnorm(n))
rf <- ranger::ranger(t~x, data.frame(t=t.train,x=x.train))

n <- 100
t.test <- runif(n, 0 ,1)
x.test <- ifelse(t.test>0.5, stats::rnorm(n, mean.shift), stats::rnorm(n))
rho <- predict(rf, data.frame(t=t.test,x=x.test))$predictions

## out-of-sample
tv.oos <- HPLB(t = t.test, rho = rho, s = seq(0.1,0.9,0.1), estimator.type = "adapt")


## total variation values
tv <- c()
for (s in seq(0.1,0.9,0.1)) {

 if (s<=0.5) {

   D.left <- Norm(0,1)
 } else {

   D.left <- UnivarMixingDistribution(Dlist = list(Norm(0,1),Norm(mean.shift,1)),
               mixCoeff = c(ifelse(s<=0.5, 1, 0.5/s), ifelse(s<=0.5, 0, (s-0.5)/s)))
 }
 if (s < 0.5) {

   D.right <- UnivarMixingDistribution(Dlist = list(Norm(0,1),Norm(mean.shift,1)),
           mixCoeff = c(ifelse(s<=0.5, (0.5-s)/(1-s), 0), ifelse(s<=0.5, (0.5/(1-s)), 1)))
 } else {

   D.right <- Norm(mean.shift,1)
 }
tv <- c(tv, TotalVarDist(e1 = D.left, e2 = D.right))
}

## plot
oldpar <- par(no.readonly =TRUE)
par(mfrow=c(2,1))
plot(t.test,x.test,pch=19,xlab="t",ylab="x")
plot(seq(0.1,0.9,0.1), tv.oos$tvhat,type="l",ylim=c(0,1),xlab="t", ylab="TV")
lines(seq(0.1,0.9,0.1), tv, col="red",type="l")
par(oldpar)
```

`HPLBmatrix` *Pairwise Total Variation Distance Lower Bound Matrix for the Multi-Class Setting*

Description

Pairwise Total Variation Distance Lower Bound Matrix for the Multi-Class Setting

Usage

```
HPLBmatrix(
  labels,
  ordering.array,
  alpha = 0.05,
  computation.type = "non-optimized",
  seed = 0,
  ...
)
```

Arguments

`labels`	a numeric vector value. The labels of the classes, should be encoded in [0,nclass-1].
`ordering.array`	a numeric array of size (nclass, nclass, nobs) such that the value (i,j,k) represents a propensity of being of class j instead of i for observation k.
`alpha`	a numeric value. The type-I error level.
`computation.type`	a character value. For the moment only "non-optimized" (default) available.
`seed`	an integer value. The seed for reproducility.
`...`	additional parameters to be passed to the HPLB function.

Value

a numeric matrix of size (nclass, nclass) giving the matrix of pairwise total variation lower bounds.

Author(s)

Loris Michel, Jeffrey Naef

References

L. Michel, J. Naef and N. Meinshausen (2020). High-Probability Lower Bounds for the Total Variation Distance

Examples

```
# iris example
require(HPLB)
require(ranger)

# training a multi-class classifier on iris and getting tv lower bounds between classes
data("iris")

ind.train <- sample(1:nrow(iris), size = nrow(iris)/2, replace = FALSE)

rf <- ranger(Species~., data = iris[ind.train, ], probability = TRUE)
preds <- predict(rf, iris[-ind.train,])$predictions

# creating the ordering array based on prediction differences
ar <- array(dim = c(3, 3, nrow(preds)))
for (i in 1:3) {
  for (j in 1:3) {
   ar[i,j,] <- preds[,j] - preds[,i]
  }
}

# encoding the class response
y <- factor(iris$Species)
levels(y) <- c(0,1,2)
y <- as.numeric(y)-1

# getting the lower bound matrix
tvhat.iris <- HPLBmatrix(labels = y[-ind.train], ordering.array = ar)
tvhat.iris
```

Index