Package: IADT (via r-universe)

August 29, 2024

Version 1.2.1

Title Interaction Difference Test for Prediction Models

Date 2024-05-14

Description Provides functions to conduct a model-agnostic asymptotic hypothesis test for the identification of interaction effects in black-box machine learning models. The null hypothesis assumes that a given set of covariates does not contribute to interaction effects in the prediction model. The test statistic is based on the difference of variances of partial dependence functions (Friedman (2008) <doi:10.1214/07-AOAS148> and Welchowski (2022) <doi:10.1007/s13253-021-00479-7>) with respect to the original black-box predictions and the predictions under the null hypothesis. The hypothesis test can be applied to any black-box prediction model, and the null hypothesis of the test can be flexibly specified according to the research question of interest. Furthermore, the test is computationally fast to apply as the null distribution does not require resampling or refitting black-box prediction models.

Depends Rmpfr

Imports methods, mgcv, Rdpack, mvnfast

RdMacros Rdpack

Encoding UTF-8

License GPL-3

LazyLoad yes

RoxygenNote 7.2.3

NeedsCompilation no

Author Thomas Welchowski [aut, cre] (<`https://orcid.org/0000-0003-2940-647X`>)

Maintainer Thomas Welchowski <`welchow@imbie.meb.uni-bonn.de`>

Repository CRAN

Date/Publication 2024-05-14 09:30:16 UTC

Contents

`IADT-package` *Interaction Difference Test for Prediction Models*

Description

Provides functions to conduct a model-agnostic asymptotic hypothesis test for the identification of interaction effects in black-box machine learning models. The null hypothesis assumes that a given set of covariates does not contribute to interaction effects in the prediction model. The test statistic is based on the difference of variances of partial dependence functions with respect to the original black-box predictions and the predictions under the null hypothesis. The hypothesis test can be applied to any black-box prediction model, and the null hypothesis of the test can be flexibly specified according to the research question of interest. Furthermore, the test is computationally fast to apply as the null distribution does not require resampling or refitting black-box prediction models.

Details

Package:	IADT
Type:	Package
Version:	1.2.1
Date:	2024-05-14
License:	GPL-3

Author(s)

Thomas Welchowski <`welchow@imbie.meb.uni-bonn.de`>

References

Welchowski T, Maloney KO, Mitchell R, Schmid M (2022). “Techniques to Improve Ecological Interpretability of Black-Box Machine Learning Models.” *JABES*, **27**, 175-197.

Friedman JH, Popescu BE (2008). “Predictive learning via rule ensembles.” *The Annals of Applied Statistics*, **2**(3), 916-954.

`pdpEst_mpfr` *Partial dependence plot with specific numerical precision*

Description

Estimates the partial dependence plot (PDP) curve given specified numerical precision.

Usage

```
pdpEst_mpfr(
  colInd,
  object,
  predictfun,
  X,
  centering = FALSE,
  outputVector = TRUE,
  newX = NULL,
  nCores = 1,
  precBits = 53 * 2
)
```

Arguments

`colInd`	Index of columns of covariates to specify the null hypothesis set *s* (integer vector).
`object`	Prediction model object (class flexible).
`predictfun`	Prediction function to be evaluated (class function). The prediction function needs to be specified with two arguments *predictfun(object, X)*. The argument *object* is the prediction model and *X* the data on which the partial dependence functions are evaluated.
`X`	Data on that the partial dependence function is evaluated (class matrix or data.frame). The structure of the data depends on the specified argument *predictfun*.
`centering`	Should the resulting values be mean centered? (logical scalar). Default corresponds to output original values.
`outputVector`	Should be only the partial dependence function returned
`newX`	Test data set (class "data.frame")
`nCores`	Number of cores used in standard parallel computation setup based on R *parallel* package. The default value of one uses serial processing across observations.
`precBits`	Numerical precision that are used in computation after the calculation of the predictions from the estimated model. Default is defined to be double the amount of the 53 Bits usually used in R.

Value

Vector of estimated the PDP curve values for each sample in *X*.

Author(s)

Thomas Welchowski <`welchow@imbie.meb.uni-bonn.de`>

References

Friedman JH, Popescu BE (2008). “Predictive learning via rule ensembles.” *The Annals of Applied Statistics*, **2**(3), 916-954.

See Also

`testIAD_mpfr`

Examples

```
#####################
# Simulation example

# Simulate covariates from multivariate standard normal distribution
set.seed(-72498)
library(mvnfast)
X <- mvnfast::rmvn(n=1e2, mu=rep(0, 2), sigma=diag(2))

# Response generation
y <- X[, 1]^2 + rnorm(n=1e2, mean=0, sd=0.5)
trainDat <- data.frame(X, y=y)

# Estimate generalized additive model
library(mgcv)
gamFit <- gam(formula=y~s(X1)+s(X2), data=trainDat,
family=gaussian())

# Estimate PDP function
pdpEst1 <- pdpEst_mpfr(colInd=1, object=gamFit,
predictfun=function(object, X){
predict(object=object, newdata=X, type="response")
}, X=trainDat,
centering=FALSE, nCores=1, precBits=53*2)

# Convert to standard precision and order in sequence of observations
pdpEst1 <- as.numeric(pdpEst1)
ordInd <- order(X[, 1])
pdpEst1 <- pdpEst1[ordInd]

# Plot: PDP curve vs. true effect
plot(x=X[ordInd, 1], y=pdpEst1, type="l")
lines(x=X[ordInd, 1], y=X[ordInd, 1]^2, lty=2, col="red")
# -> Both curves are similiar
```

`testIAD_mpfr` *Model-agnostic interaction difference test for prediction models*

Description

Tests if a given set *s* of covariates contributes to interaction effects in the prediction model.

Usage

```
testIAD_mpfr(
  colInd,
  object,
  predictfun,
  X,
  newX = NULL,
  y = NULL,
  alternative = "twoSided",
  centering = FALSE,
  nCoresPDP = 1,
  precBits = 53 * 2
)
```

Arguments

`colInd`	Index of columns of covariates to specify the null hypothesis set *s* (integer vector).
`object`	Prediction model object (class flexible).
`predictfun`	Prediction function to be evaluated (class function). The prediction function needs to be specified with two arguments *predictfun(object, X)*. The argument *object* is the prediction model and *X* the data on which the partial dependence functions are evaluated.
`X`	Data on that the partial dependence function is evaluated (class matrix or data.frame). The structure of the data depends on the specified argument *predictfun*.
`newX`	Test data set, which is used in the evaluation of the partial dependence functions (class "matrix" or "data.frame"). Default value NULL evaluates the interaction test on training data *X*.
`y`	Response (numeric vector). Default value NULL means that the statistic is not based on response information.
`alternative`	Should the hypothesis be tested two-sided *alternative="twoSided"*, greater *alternative="greater"* or less *alternative="less"* (character vector)? Default is "twoSided".
`centering`	Should the resulting values be mean centered? (logical scalar). Default corresponds to output original values.

`nCoresPDP`	Number of cores used in standard parallel computation setup based on R *parallel* package to compute partial dependence function. The default value of one uses serial processing across observations.
`precBits`	Numerical precision that are used in computation after the calculation of the predictions from the estimated model. Default is defined to be double the amount of the 53 Bits usually used in R.

Details

The data set used to evaluate the interaction test coulbe be training or test data. The proof of the asymptotic distribution only works in case of test data. Therefore we recommend usage of test data for evaluation. Two types of hypothesis are recommended:

- Model interactions: Model does not have interaction effects between covariates in set *s* and between other covariates and those of set *s*. Argument *colInd* specifies the set s and argument *alternative* should be set to "twoSided".
- MSE improvement: Do interactions between between covariates in set *s* and between other covariates and those of set *s* descrease MSE? Argument *alternative* should be set to "less".

Value

List with following entries:

- testStat: Test statistic based on one-sample Gaussian test.
- pValue: P-value based on asymptotic normal distribution.
- IAD_f_terms: If *y=NULL* then the terms used to calculate the variance of the predictions with possible interaction effects in set *s* are returned. If *y!=NULL*, then the terms of the MSE of the prediction model with interactions is returned.
- IAD_f: If *y=NULL* then the variance of the predictions with possible interaction effects in set *s* is returned. If *y!=NULL*, then the MSE of the prediction model with possible interactions is returned.
- IAD_PD_terms: If *y=NULL* then the terms used to calculate the variance of predictions under the null hypothesis of no interaction effects in set *s* are returned. If *y!=NULL*, then the terms used to calculate the MSE of the prediction model are returned. under the null hypothesis of no interaction effects are returned.
- IAD_PD: If *y=NULL* then the variance of predictions under the null hypothesis of no interaction effects of set *s* is returned. If *y!=NULL*, then the MSE of the prediction model is returned.
- z3 Terms used to calculate the the interaction difference. under the null hypothesis of no interaction effects is returned.
- IAD: Interaction difference calculated as mean of the terms *z3*.

Note

Numerical precision does have influence on Type I error percentages. Under the null hypothesis values of the test statistic can be very small near zero. To lessen the impact of random rounding errors increased numerical precision is important.

Author(s)

Thomas Welchowski `<welchow@imbie.meb.uni-bonn.de>`

See Also

`pdpEst_mpfr`

Examples

```
#####################
# Simulation example
#####################

library(mvnfast)
library(mgcv)

############################################################################
# H0: Covariate X1 does not contribute to interaction effects in the
# prediction model.

# Train data
# Simulate covariates from multivariate standard normal distribution
set.seed(-72498)
nSize <- 100
X <- mvnfast::rmvn(n=nSize, mu=rep(0, 2), sigma=diag(2))

# Response generation
y <- X[, 1]^2 + rnorm(n=nSize, mean=0, sd=0.25)

# Complete data frame
trainDat <- data.frame(X, y=y)

# Test data
# Simulate covariates from multivariate standard normal distribution
testX <- mvnfast::rmvn(n=nSize, mu=rep(0, 2), sigma=diag(2))

# Response generation
testY <- testX[, 1]^2 + rnorm(n=nSize, mean=0, sd=0.25)
testDat <- data.frame(testX, y=testY)

# Estimate generalized additive model with training data
library(mgcv)
gamFit <- gam(formula=y~s(X1)+s(X2), data=trainDat,
              family=gaussian())

# Test interaction
testIAD_mpfr1 <- testIAD_mpfr(colInd=1, object=gamFit,
                              predictfun=function(object, X){
                                predict(object=object, newdata=X, type="response")
                              }, X=testDat)
testIAD_mpfr1$pValue
# -> H0 is not rejected with alpha=0.05
```

```
################################################################
# H1: X1 does contribute to at least one interaction effect
# in the prediction model.

# Train data
# Simulate covariates from multivariate standard normal distribution
set.seed(-72498)
nSize <- 150
X <- mvnfast::rmvn(n=nSize, mu=rep(0, 2), sigma=diag(2))

# Response generation
y <- X[, 1]^2 * X[, 2] + rnorm(n=nSize, mean=0, sd=0.25)

# Complete data frame
trainDat <- data.frame(X, y=y)

# Test data
# Simulate covariates from multivariate standard normal distribution
testX <- mvnfast::rmvn(n=nSize, mu=rep(0, 2), sigma=diag(2))

# Response generation
testY <- testX[, 1]^2 * testX[, 2] + rnorm(n=nSize, mean=0, sd=0.25)
testDat <- data.frame(testX, y=testY)

# Estimate generalized additive model with training data
library(mgcv)
gamFit <- gam(formula=y~s(X1, X2, k=25), data=trainDat,
              family=gaussian())

# Test interaction
testIAD_mpfr1 <- testIAD_mpfr(colInd=1, object=gamFit,
                              predictfun=function(object, X){
                                predict(object=object, newdata=X, type="response")
                              }, X=testDat)
testIAD_mpfr1$pValue
# -> H0 is rejected with alpha=0.05
```

Index