Package 'APPEstimation'

January 5, 2018

Type Package

Title Adjusted Prediction Model Performance Estimation

Version 0.1.1

Depends densratio

Date 2018-1-4

Author Eisuke Inoue, Hajime Uno

Maintainer Eisuke Inoue <eisuke.inoue@marianna-u.ac.jp>

Description Calculating predictive model performance measures adjusted for predictor distributions using density ratio method (Sugiyama et al., (2012, ISBN:9781139035613)). L1 and L2 error for continuous outcome and C-statistics for binomial outcome are computed.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2018-01-05 12:30:40 UTC

R topics documented:

APPEstimation-package *R function to calculate model performance measure adjusted for predictor distributions.*

Description

This package provides the function to estimate model performance measures (L_1, L_2, C-statistics). The difference in the distribution of predictors between two datasets (training and validation) is adjusted by a density ratio estimate.

Details

Package:	APPEstimation
Type:	Package
Title:	Adjusted Prediction Model Performance Estimation
Version:	0.1.1
Depends:	densratio
Date:	2018-1-4
Author:	Eisuke Inoue, Hajime Uno
Maintainer:	Eisuke Inoue <eisuke.inoue@marianna-u.ac.jp>
Description:	Calculating predictive model performance measures adjusted for predictor distributions using density ratio met
License:	GPL-2

Index of help topics:

Author(s)

Eisuke Inoue, Hajime Uno

Maintainer: Eisuke Inoue <eisuke.inoue@marianna-u.ac.jp>

References

Sugiyama, M., Suzuki, T. & Kanamori, T. Density Ratio Estimation in Machine Learning. Cambridge University Press 2012. ISBN:9781139035613.

Examples

```
set.seed(100)

# generating learning data
n0  = 100
Z   = cbind(rbeta(n0, 5, 5), rbeta(n0, 5, 5))
Y   = apply(Z, 1, function (xx) {
        rbinom(1, 1, (1/(1+exp(-(sum(c(-2,2,2) * c(1,xx)))))))})
dat = data.frame(Y=Y, Za=Z[,1], Zb=Z[,2])

# the model to be evaluated
mdl = glm(Y~., binomial, data=dat)

# validation dataset, with different centers on predictors
n1   = 100
Z1   = cbind(rbeta(n1, 6, 4), rbeta(n1, 6, 4))
Y1   = apply(Z1, 1, function (xx) {
         rbinom(1, 1, (1/(1+exp(-(sum(c(-2,2,2) * c(1,xx)))))))})
dat1 = data.frame(Y=Y1, Za=Z1[,1], Zb=Z1[,2])

# calculation of L1 and L2 for this model
appe.glm(mdl, dat, dat1, reps=0)
```

`appe.glm` *C-statistics adjusted for predictor distributions*

Description

Calculates adjusted C statistics by predictor distributions for a generalized linear model with binary outcome.

Usage

```
appe.glm(mdl, dat.train, dat.test, method = "uLSIF", sigma = NULL,
         lambda = NULL, kernel_num = NULL, fold = 5, stabilize = TRUE,
         qstb = 0.025, reps = 2000, conf.level = 0.95)
```

Arguments

`mdl`	a `glm` object describing a prediction model to be evaluated.
`dat.train`	a dataframe used to construct a prediction model (specified in `mdl`), corresponding to a training data. Need to include outcome and all predictors.
`dat.test`	a dataframe corresponding to a validation (testing) data. Need to include outcome and all predictors.
`method`	uLSIF or KLIEP. Same as the argument in `densratio` function from densratio package.

sigma	a positive numeric vector corresponding to candidate values of a bandwidth for Gaussian kernel. Same as the argument in densratio function from densratio package.
lambda	a positive numeric vector corresponding to candidate values of a regularization parameter. Same as the argument in densratio function from densratio package.
kernel_num	a positive integer corresponding to number of kernels. Same as the argument in densratio function from densratio package.
fold	a positive integer corresponding to a number of the folds of cross-validation in the KLIEP method. Same as the argument in densratio function from densratio package.
stabilize	a logical value as to whether tail weight stabilization is performed or not. If TRUE, both tails of the estimated density ratio distribution are replaced by the constant value which is specified at qstb option.
qstb	a positive numerical value less than 1 to control the degree of weight stabilization. Default value is 0.025, indicating estimated density ratio values less than the 2.5 percentile and more than the 97.5 percentile are set to 2.5 percentile and 97.5 percentile, respectively.
reps	a positive integer to specify bootstrap repetitions. If 0, bootstrap calculations are not performed.
conf.level	a numerical value indicating a confidence level of interval.

Value

Adjusted and non-adjusted estimates of C-statistics are provided as matrix form. "Cstat" indicates non-adjusted version, "C adjusted by score" indicates adjusted version by linear predictors distribution, and "C adjusted by predictors" indicates adjusted version by predictor distributions (multidimensionally). For confidence intervals, "Percentile" indicates a confidence interval by percentile method and "Approx" indicates approximated versions by Normal distribution.

Examples

```
set.seed(100)

# generating learning data
n0  = 100
Z   = cbind(rbeta(n0, 5, 5), rbeta(n0, 5, 5))
Y   = apply(Z, 1, function (xx) {
        rbinom(1, 1, (1/(1+exp(-(sum(c(-2,2,2) * c(1,xx)))))))})
dat = data.frame(Y=Y, Za=Z[,1], Zb=Z[,2])

# the model to be evaluated
mdl = glm(Y~., binomial, data=dat)

# validation dataset, with different centers on predictors
n1   = 100
Z1   = cbind(rbeta(n1, 6, 4), rbeta(n1, 6, 4))
Y1   = apply(Z1, 1, function (xx) {
```

```
        rbinom(1, 1, (1/(1+exp(-(sum(c(-2,2,2) * c(1,xx)))))))})
dat1 = data.frame(Y=Y1, Za=Z1[,1], Zb=Z1[,2])

# calculation of L1 and L2 for this model
appe.glm(mdl, dat, dat1, reps=0)
```

`appe.lm` *L_1 and L_2 errors adjusted for predictor distributions*

Description

Calculates adjusted L_1 and L_2 errors by predictor distributions for a linear model.

Usage

```
appe.lm(mdl, dat.train, dat.test, method = "uLSIF", sigma = NULL,
        lambda = NULL, kernel_num = NULL, fold = 5, stabilize = TRUE,
        qstb = 0.025, reps = 2000, conf.level = 0.95)
```

Arguments

`mdl`	a `lm` object describing a prediction model to be evaluated.
`dat.train`	same as in `appe.glm`.
`dat.test`	same as in `appe.glm`.
`method`	same as in `appe.glm`.
`sigma`	same as in `appe.glm`.
`lambda`	same as in `appe.glm`.
`kernel_num`	same as in `appe.glm`.
`fold`	same as in `appe.glm`.
`stabilize`	same as in `appe.glm`.
`qstb`	same as in `appe.glm`.
`reps`	same as in `appe.glm`.
`conf.level`	same as in `appe.glm`.

Value

Adjusted and non-adjusted estimates of L_1 and L_2 errors are provided as matrix form. "L1" and "L2" indicate non-adjusted versions, "L1 adjusted by score" and "L2 adjusted by score" indicate adjusted versions by linear predictors distribution, "L1 adjusted by predictors" and "L2 adjusted by predictors" indicate adjusted versions by predictor distributions (multi-dimensionally). For confidence intervals, "Percentile" indicates a confidence interval by percentile method and "Approx" indicates approximated versions by Normal distribution.

Examples

```
set.seed(100)

# generating development data
n0  = 100
Z   = cbind(rbeta(n0, 3, 3), rbeta(n0, 3, 3))
Y   = apply(Z, 1, function(xx) { rlnorm(1, sum(c(1, 1) * xx), 0.3) })
dat = data.frame(Za=Z[,1], Zb=Z[,2], Y=Y)

# the model to be evaluated
mdl = lm(Y~ Za + Zb, data=dat)

# generating validation dataset
n1   = 100
Z1   = cbind(rbeta(n0, 3.5, 2.5), rbeta(n0, 3.5, 2.5))
Y1   = apply(Z1, 1, function(xx) { rlnorm(1, sum(c(1, 1) * xx), 0.3) })
dat1 = data.frame(Za=Z1[,1], Zb=Z1[,2], Y=Y1)

# calculation of L1 and L2 for this model
appe.lm(mdl, dat, dat1, reps=0)
```

`cvalest.bin` *Estimation of C-statistics*

Description

Calculates C-statistics. Individual case weight can be incorporated.

Usage

```
cvalest.bin(Y, scr, wgt = NULL)
```

Arguments

`Y`	a numerical vector of inary outcome, either 0 or 1.
`scr`	a numerical vector of continuous variable.
`wgt`	a numerical vector corresponding to individuatl weight.

Value

C-statistics is provided.

`densratio.appe` *A wrapper function*

Description

A wrapper function to use "densratio" function from the densratio package.

Usage

```
densratio.appe(xtrain, xtest, method = "uLSIF", sigma = NULL,
               lambda = NULL, kernel_num = NULL, fold = 5,
               stabilize = TRUE,  qstb = 0.025)
```

Arguments

`xtrain`	a dataframe used to construct a prediction model.
`xtest`	a dataframe corresponding to a validation (testing) data.
`method`	same as in `appe.glm`.
`sigma`	same as in `appe.glm`.
`lambda`	same as in `appe.glm`.
`kernel_num`	same as in `appe.glm`.
`fold`	same as in `appe.glm`.
`stabilize`	same as in `appe.glm`.
`qstb`	same as in `appe.glm`.

Index