Package ‘ExtremesBinaryClassifier’

December 21, 2021

Type Package

Description Provides functions for the empirical risk estimation of binary classifiers for extremes, as proposed in Legrand, J., Naveau, P., and Oesting, M. (2021) <add doi here>.

Title Evaluation of binary classifiers for extremes

Version 0.1.0

Date 2021-10-22

Author Juliette Legrand [aut, cre]

Maintainer Juliette Legrand <juliette.legrand@lsce.ipsl.fr>

Depends R (>= 3.1.0)

Imports graphicalExtremes

Remotes github::sebastian-engelke/graphicalExtremes

Suggests rpart, randomForest, glmnet, ...

License to add (ex. GPL (>= 2))

Encoding UTF-8

URL to add

RoxygenNote 7.1.2

LazyData TRUE

R topics documented:

`dataDanube` *Danube river discharges*

Description

Daily river discharges, measured in m^3/s, at 31 stations spread over the upper Danube basin. The data set covers the period from 1960 to 2010 but only the months of June, July, and August are retained. These data are already declustered following Mhalla et al.(2020) methodology.

Usage

```
graphicalExtremes::danube
```

Source

Bavarian Environmental Agency (http://www.gkd.bayern.de)

References

Asadi, P., Davison, A.C., and Engelke, S. (2015). Extremes on river networks. The Annals of Applied Statistics, 9(4), 2023-2050.

Examples

```
graphicalExtremes::danube
```

`EmpiricalRisk` *A Risk estimation function*

Description

Compute the empirical risk defined in Proposition 5 given the output of a classifier g and the true binary outcomes Y. If epsilon>0, supply the values of the trained classifier on the thresholded data g.eps and the true binary outcomes Y.eps := +1 if H > eps_u and -1 otherwise.

Usage

```
EmpiricalRisk(Y, Y.eps = NULL, g, g.eps = NULL, epsilon = TRUE)
```

Arguments

`Y`	vector of the true binary outcomes
`Y.eps`	vector of the true binary outcomes in the extreme region
`g`	vector of the predicted binary outcomes from a given classifier
`g.eps`	vector of the predicted binary outcomes in the extreme region from the same classifier
`epsilon`	logical value indicating whether the classic risk function should be used or the extended version

Details

to add

References

Legrand et al.

See Also

LinearClassifier

Examples

```
require(rpart)

set.seed(123)
## Reproduce the simulation example from Legrand et al.
nsim <- 1e4
X1 <- 1/(runif(nsim)^(1/3))
X2 <- 1/(runif(nsim)^(1/2))
P <- 1/(runif(nsim)^(1/2))
H <- X1 + P
## Compute the two thresholds u and epsilon_u
u <- quantile(H,probs=0.97)
eps <- 0.6
eps_u <- u*eps
## Split data between training and testing sets
ii <- sample.int(length(H), size = 0.7*length(H), replace=F)
Xtrain <- cbind(X1[ii], X2[ii])
Xtest <- cbind(X1[-ii], X2[-ii])
Htrain <- H[ii]
Htest <- H[-ii]

## Linear classifier
#~~~~~~~~~~~~~~~~~~~
## Train the linear classifier with mass
init0 <- lm(H ~ X1 + X2, data = data.frame(H = H, X1 = X1, X2 = X2))$coefficients[2:3]
linclass <- LinearClassifier(X = Xtrain, thresh = u, H = Htrain, initials = init0, epsilon=0)$theta
## Compute the predicted binary outcome on the test set from all the data
glin <- 2*(as.vector(linclass %*% t(Xtest)) > u) -1
#~~~~~~~~~~~~~~~~~~~
## Train the linear classifier without mass
glin0 <- init0 %*% t(cbind(X1,X2))
init <- lm(H ~ X1 + X2, data=data.frame(H = H[H>eps_u & glin0>eps_u], X1 = X1[H>eps_u & glin0>eps_u],
          X2 = X2[H>eps_u & glin0>eps_u]))$coefficients[2:3]
linclass.eps <- LinearClassifier(X = Xtrain[,c(1,2)], thresh = u, H = Htrain, initials = init,
                  epsilon = eps)$theta
## Compute the predicted binary outcome on the test set from the extreme region data
glineps <- 2*(as.vector(linclass.eps %*% t(Xtest)) > eps_u) - 1
#~~~~~~~~~~~~~~~~~~~
## Compute the true binary outcome on the test set from all the data and only with the extreme region data
Ytesteps <- 2*(Htest > eps_u) - 1
```

```
Ytest <- 2*(Htest > u) - 1
#~~~~~~~~~~~~~~~~~~~
## Compute the associated risk
EmpiricalRisk(Y = Ytest, Y.eps = Ytesteps, g = glin, g.eps = glineps, epsilon = TRUE)

## Comparison with regression tree
#~~~~~~~~~~~~~~~~~~~
## Train the tree classifier with mass
Ytrain <- 2*(Htrain > u) - 1
treeclass <- rpart(y~., data=data.frame(x = Xtrain, y = as.factor(Ytrain)), method = "class")
## Compute the predicted binary outcome on the test set from all the data
gtree <- as.numeric(as.character(predict(treeclass, newdata = data.frame(x = Xtest, y = Ytest),
           type="class")))
#~~~~~~~~~~~~~~~~~~~
## Train the tree classifier without mass
Ytraineps <- 2*(Htrain > eps_u) - 1
treeclasseps <- rpart(y~., data=data.frame(x = Xtrain, y = as.factor(Ytraineps)), method = "class")
## Compute the predicted binary outcome on the test set from the extreme region data
gtreeeps <- as.numeric(as.character(predict(treeclasseps, newdata = data.frame(x = Xtest, y = Ytesteps),
              type = 'class')))
#~~~~~~~~~~~~~~~~~~~
## Compute the associated risk
EmpiricalRisk(Y = Ytest, Y.eps = Ytesteps, g = gtree, g.eps = gtreeeps, epsilon = TRUE)
```

`LinearClassifier` *Optimal linear classifier*

Description

Compute the optimal linear classifier as defined in Appendix B by minimizing the empirical risk (defined by emp.risk.lin). Initial values must be provided which can be estimated by performing a classical linear regression (lm) for example.

Usage

```
LinearClassifier(X, thresh, H, initials, epsilon)
```

Arguments

`X`	numeric matrix corresponding to the input data we want to classify
`thresh`	single numeric giving the threshold over which an extreme event is defined
`H`	numeric vector corresponding to the latent variable that we wish to predict
`initials`	initial values for the parameters of the linear classifier to be optimized over
`epsilon`	single numeric giving the amount of data to remove

Value

theta value of the linear classifier

`theta`	the optimal parameters for the linear classifier
`Risk`	the value of the risk corresponding theta

References

Legrand et al.

Examples

```
set.seed(123)
## Reproduce the simulation example from Legrand et al.
nsim <- 1e4
X1 <- 1/(runif(nsim)^(1/3))
X2 <- 1/(runif(nsim)^(1/2))
P <- 1/(runif(nsim)^(1/2))
H <- X1 + P
u <- quantile(H,probs=0.97)
init <- lm(H ~ X1 + X2, data=data.frame(H = H, X1 = X1, X2 = X2))$coefficients[2:3]
LinearClassifier(X = cbind(X1, X2), thresh = u, H = H, initials = init, epsilon = 0)
```

Index