Package 'nonlinearICP'

July 31, 2017

Type Package

Title Invariant Causal Prediction for Nonlinear Models

Version 0.1.2.1

Date 2017-07-31

Author Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>, Jonas Peters <jonas.peters@math.ku.dk>

Maintainer Christina Heinze-Deml <heinzedeml@stat.math.ethz.ch>

Depends R (>= 3.1.0)

Description Performs 'nonlinear Invariant Causal Prediction' to estimate the causal parents of a given target variable from data collected in different experimental or environmental conditions, extending 'Invariant Causal Prediction' from Peters, Buehlmann and Meinshausen (2016), <arXiv:1501.01332>, to nonlinear settings. For more details, see C. Heinze-Deml, J. Peters and N. Meinshausen: 'Invariant Causal Prediction for Nonlinear Models', <arXiv:1706.08576>.

License GPL

LazyData TRUE

Imports methods, CondIndTests, data.tree, caTools, randomForest

Suggests testthat

URL https://github.com/christinaheinze/nonlinearICP-and-CondIndTests

BugReports https://github.com/christinaheinze/nonlinearICP-and-CondIndTests/issues

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-07-31 13:42:46 UTC

R topics documented:

`nonlinearICP` *Nonlinear Invariant Causal Prediction*

Description

Nonlinear Invariant Causal Prediction

Usage

```
nonlinearICP(X, Y, environment,
  condIndTest = InvariantResidualDistributionTest, argsCondIndTest = NULL,
  alpha = 0.05, varPreSelectionFunc = NULL,
  argsVarPreSelectionFunc = NULL, maxSizeSets = ncol(X),
  condIndTestNames = NULL, speedUp = FALSE, subsampleSize = c(0.1, 0.25,
  0.5, 0.75, 1), retrieveDefiningsSets = TRUE, seed = 1,
  stopIfEmpty = TRUE, testAdditionalSet = NULL, verbose = FALSE)
```

Arguments

`X`	A (nxp)-dimensional matrix (or data frame) with n observations of p variables.
`Y`	A (nx1)-dimensional response vector.
`environment`	Environment variable(s) in an (n x k)-dimensional matrix or dataframe. Note that not all nonlinear conditional independence tests may support more than one environmental variable.
`condIndTest`	Function implementing a conditional independence test (see below for the required interface). Defaults to `InvariantResidualDistributionTest` from the package `CondIndTests`.
`argsCondIndTest`	Arguments of `condIndTest`. Defaults to `NULL`.
`alpha`	Significance level to be used. Defaults to `0.05`.
`varPreSelectionFunc`	Variable selection function that is applied to pre-select a set of variables before running the ICP procedure on the resulting subset. Should be used with care as causal parents might be excluded in this step. Defaults to `NULL`.
`argsVarPreSelectionFunc`	Arguments of `varPreSelectionFunc`. Defaults to `NULL`.
`maxSizeSets`	Maximal size of sets considered as causal parents. Defaults to `ncol(X)`.

`condIndTestNames`
: Name of conditional independence test, used for printing. Defaults to `NULL`.

`speedUp`
: Use subsamples of sizes specified in `subsampleSize` to speed up the test for sets where the null hypothesis can already be rejected based on a small number of samples (a larger sample size would potentially further decrease the p-value but would not change the decision, i.e. the set is rejected in any case). Applies Bonferroni multiple testing correction. Defaults to `FALSE`.

`subsampleSize`
: Size of subsamples used in `speedUp` procedure as fraction of total sample size. Defaults to `c(0.1, 0.25, 0.5, 0.75, 1)`.

`retrieveDefiningsSets`
: Boolean variable to indicate whether defining sets should be retrieved. Defaults to `TRUE`.

`seed`
: Random seed.

`stopIfEmpty`
: Stop ICP procedure if retrieved set is empty. If `retrieveDefiningsSets` is `TRUE`, setting `stopIfEmpty` to `TRUE` results in testing further sets to retrieve the defining sets. However, setting `stopIfEmpty` to `TRUE` in this case will still speedup the procedure as some sets will not be tested (namely those where accepting/rejecting would not affect the defining sets). Setting `stopIfEmpty` to `FALSE` means that all possible subsets of the predictors are tested.

`testAdditionalSet`
: If a particular set should be tested, the corresponding indices can be provided via this argument.

`verbose`
: Boolean variable to indicate whether messages should be printed.

Details

The function provided as `condIndTest` needs to take the following arguments in the given order: `Y, environment, X, alpha, verbose`. Additional arguments can then be provided via `argsCondIndTest`.

Value

A list with the following elements:

- `retrievedCausalVars` Indices of variables in $\hat{S}$
- `acceptedSets` List of accepted sets.
- `definingSets` List of defining sets.
- `acceptedModels` List of accepted models if specified in `argsCondIndTest`.
- `pvalues.accepted` P-values of accepted sets.
- `rejectedSets` List of rejected sets.
- `pvalues.rejected` P-values of rejected sets.
- `settings` Settings provided to `nonlinearICP`.

References

Please cite C. Heinze-Deml, J. Peters and N. Meinshausen: "Invariant Causal Prediction for Nonlinear Models", arXiv:1706.08576.

See Also

The function CondIndTest from the package CondIndTests is a wrapper for a variety of nonlinear conditional independence tests that can be used in condIndTest.

Examples

```
# Example 1
require(CondIndTests)
data("simData")
targetVar <- 2
# choose environments where we did not intervene on var
useEnvs <- which(simData$interventionVar[,targetVar] == 0)
ind <- is.element(simData$environment, useEnvs)
X <- simData$X[ind,-targetVar]
Y <- simData$X[ind,targetVar]
E <- as.factor(simData$environment[ind])
result <- nonlinearICP(X = X, Y = Y, environment = E)
cat(paste("Variable",result$retrievedCausalVars, "was retrieved as the causal
parent of target variable", targetVar))

#####################################################

# Example 2
E <- rep(c(1,2), each = 500)
X1 <- E + 0.1*rnorm(1000)
X1 <- rnorm(1000)
X2 <- X1 + E^2 + 0.1*rnorm(1000)
Y <- X1 + X2 + 0.1*rnorm(1000)
resultnonlinICP <- nonlinearICP(cbind(X1,X2), Y, as.factor(E))
summary(resultnonlinICP)
```

---

simData *Example dataset for tests*

Description

Example dataset for tests

Usage

```
data("simData")
```

Format

A list with the following entries

- X Dataframe with 500 observations and three variables.
- environment A vector of length 500, indicating which environment the observations belong to.

- `interventionVar` A matrix of dimension 6 (no. of environments) x 3 (no. of variables), where entry i,j indicates whether variable j was intervened on in environment i.

`summary.nonlinICP.class` *summary function*

Description

Summary functions for 'nonlinICP.class' objects.

Usage

```
## S3 method for class 'nonlinICP.class'
summary(object, ...)
```

Arguments

`object`	object of class 'nonlinICP.class'.
`...`	Additional inputs to generic summary function (not used).

Author(s)

Christina Heinze-Deml and Jonas Peters

`varSelectionRF` *Variable selection function that can be provided to* `nonlinearICP` *- it is then applied to pre-select a set of variables before running the ICP procedure on this subset. Here, the variable selection is based on random forest variable importance measures.*

Description

Variable selection function that can be provided to `nonlinearICP` - it is then applied to pre-select a set of variables before running the ICP procedure on this subset. Here, the variable selection is based on random forest variable importance measures.

Usage

```
varSelectionRF(X, Y, env, verbose, nSelect = sqrt(ncol(X)),
  useMtry = sqrt(ncol(X)), ntree = 100)
```

Arguments

X	A (nxp)-dimensional matrix (or data frame) with n observations of p variables.
Y	Response vector (n x 1)
env	Indicator of the experiment or the intervention type an observation belongs to. A numeric vector of length n. Has to contain at least two different unique values.
verbose	If FALSE, most messages are supressed.
nSelect	Number of variables to select. Defaults to sqrt(ncol(X)).
useMtry	Random forest parameter mtry. Defaults to sqrt(ncol(X)).
ntree	Random forest parameter ntree. Defaults to 100.

Value

A vector containing the indices of the selected variables.

Examples

```
# Example 1
require(CondIndTests)
data("simData")
targetVar <- 2
# choose environments where we did not intervene on var
useEnvs <- which(simData$interventionVar[,targetVar] == 0)
ind <- is.element(simData$environment, useEnvs)
X <- simData$X[ind,-targetVar]
Y <- simData$X[ind,targetVar]
E <- as.factor(simData$environment[ind])
chosenIdx <- varSelectionRF(X = X, Y = Y, env = E, verbose = TRUE)
cat(paste("Variable(s)", paste(chosenIdx, collapse=", "), "was/were chosen."))
```

Index