Package 'listdtr'

March 5, 2021

Type Package

Title List-Based Rules for Dynamic Treatment Regimes

Version 1.1

Date 2021-03-04

Author Yichi Zhang

Maintainer Yichi Zhang <yzhang52@ncsu.edu>

Description
Construction of list-based rules, i.e. a list of if-then clauses, to estimate the optimal dynamic treatment regime. Details can be found at Zhang et al (2018) <doi:10.1080/01621459.2017.1345743>.

License GPL (>= 2)

Depends ggplot2, grid

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-03-04 23:20:02 UTC

R topics documented:

`listdtr-package` *List-Based Rules for Dynamic Treatment Regimes*

Description

Construction of list-based rules, i.e. a list of if-then clauses, to estimate the optimal dynamic treatment regime. Details can be found at Zhang et al (2018) <doi:10.1080/01621459.2017.1345743>.

Details

The DESCRIPTION file:

Package:	listdtr
Type:	Package
Title:	List-Based Rules for Dynamic Treatment Regimes
Version:	1.1
Date:	2021-03-04
Author:	Yichi Zhang
Maintainer:	Yichi Zhang <yzhang52@ncsu.edu>
Description:	Construction of list-based rules, i.e. a list of if-then clauses, to estimate the optimal dynamic treatment regime.
License:	GPL (>=2)
Depends:	ggplot2, grid

Index of help topics:

Main function is `listdtr`. Useful functions include `build.rule`, `krr`.

Author(s)

Yichi Zhang

Maintainer: Yichi Zhang <yzhang52@ncsu.edu>

References

https://arxiv.org/abs/1606.01472

`build.rule` *Low-level Functions for Handling List-based Rules*

Description

Build, apply and visualize list-based rules directly using features and losses.

Usage

```
build.rule(x, y,
  maxlen = 10L, zeta = 0.1 * mean(y), eta = 0.05 * sum(y))

build.rule.cv(x, y,
  kfolds = 5L, fold = NULL,
  maxlen = 10L, zeta.choices = NULL, eta.choices = NULL,
  cv.only = FALSE)

apply.rule(object, xnew, what = c("label", "index"))

show.rule(object, digits = 3L)

verbalize.rule(object, digits = 3L)

draw.rule(object, digits = 3L, filepath = NULL)
```

Arguments

`x`	A matrix of features.
`y`	A matrix of losses; `y[i,j]` gives the loss if the `i`-th observation receives the `j`-th treatment.
`maxlen`	A scalar for the maximum length of the list.
`zeta`	A scalar for tuning paramter zeta. Larger zeta tends to construct condition that covers more observations in each if-then clause.
`eta`	A scalar for tuning paramter eta. Larger eta tends to construct condition that uses less features in each if-then clause.
`kfolds`	A scalar for the number of folds for cross validation.
`fold`	An integer vector consisting of fold membership.
`zeta.choices`	A numeric vector for possible values of zeta in cross validation.
`eta.choices`	A numeric vector for possible values of eta in cross validation.
`cv.only`	A boolean scalar. If true, only cross validated losses are computed. Otherwise, the list built using the optimal zeta and eta is also computed.

object	Return value of build.rule, or build.rule.cv with cv.only = FALSE.
xnew	A matrix of features for prediction.
what	A scalar that determines the form in which the recommended treatment is represented.
digits	A scalar for the number of decimal digits to show.
filepath	A character scalar, if not null, that gives the location that the diagram will save to.

Details

See the reference if interested in the algorithm.

Value

build.rule returns a list.

build.rule.cv returns a list as well as cross validated losses.

apply.rule returns a vector of recommended actions.

show.rule prints the rule in words and returns it invisibly.

verbalize.rule returns a data.frame that contains conditions and actions separately for each if-then clause.

draw.rule returns a ggplot2 object that contains the diagram.

Note

Use these functions only when it is really necessary.

References

https://arxiv.org/abs/1606.01472

See Also

listdtr

Examples

```
x <- matrix(rnorm(200 * 10), 200, 10)
y <- cbind(
  a1 = as.double(x[, 1] < 0) + rnorm(200, 0, 0.1),
  a2 = as.double(x[, 2] > 0) + rnorm(200, 0, 0.1))
y[y < 0] <- 0

obj <- build.rule(x, y)
show.rule(obj)
draw.rule(obj)

xnew <- matrix(rnorm(1000 * 10), 1000, 10)
ynew <- apply.rule(obj, xnew)
table(factor(xnew[, 1] < 0) : factor(xnew[, 2] < 0), ynew)
```

krr *Kernel Ridge Regression*

Description

Fit kernel ridge regression, i.e. reproducing kernel Hilbert space regression.

Usage

```
krr(x, y, group = NULL)
```

Arguments

x	a matrix of predictors.
y	a vector of response.
group	an optional vector of the same length as y which specifies the group membership. The regression model is fitted separately for each group of observations but with the same scaling factors as well as penalty amount. If omitted, a single group is assumed.

Details

krr minimizes the sum of squared loss plus a penalty term on the squared norm of the regression function. Gaussian kernel is used. Tuning parameters are chosen by minimizing the leave-one-out cross validated mean squared error. See the mathematical formulation in the reference.

Value

An object of class krr.

References

https://arxiv.org/abs/1606.01472

See Also

predict.krr

Examples

```
x <- matrix(rnorm(200 * 10), 200, 10)
y <- x[, 1] + x[, 2] ^ 2 + x[, 3] * x[, 4] + rnorm(200)
obj <- krr(x, y)

xnew <- matrix(rnorm(1000 * 10), 1000, 10)
ynew <- predict(obj, xnew)

ytrue <- xnew[, 1] + xnew[, 2] ^ 2 + xnew[, 3] * xnew[, 4]
mean((ynew - ytrue) ^ 2)  # MSE
```

`listdtr` *Estimation of List-based Optimal Dynamic Treatment Regime*

Description

Estimate the optimal dynamic treatment regime / individualized treatment rule, in the form of decision list, namely, a sequence of if-then clauses.

Usage

```
listdtr(y, a, x, stage.x,
  seed = NULL, kfolds = 5L, fold = NULL,
  maxlen = 10L, zeta.choices = NULL, eta.choices = NULL)
```

Arguments

`y`	a matrix of immediate outcomes, of size `n.obs` by `n.stage`, where `n.obs` is the number of observations and `n.stage` is the number of stages. Assume larger outcomes are more favorable.
`a`	a matrix of treatments/interventions actually received at each stage, of size `n.obs` by `n.stage`.
`x`	a matrix of features (such as demographics, biomarkers, confounders), of size `n.obs` by `n.feature`, where `n.feature` is the number of features measured at any of the stages.
`stage.x`	a vector of length `n.feature`, with values in 1, ..., `n.stage` that gives the stage at which each feature is measured.
`seed`	seed for random number generator to obtain `fold`. Omitted if `fold` is not null.
`kfolds`	number of folds to perform cross validation.
`fold`	a vector of length `n.obs` that specifies fold membership for each observation.
`maxlen`	maximum length of the decision list in each stage. Should be a scalar.
`zeta.choices`	Choices for the tuning parameter zeta. Larger value of zeta tends to construct a condition that covers more observations in each if-then clause. Should be null or a numeric vector.
`eta.choices`	Choices for the tuning parameter eta. Larger value of eta tends to construct a condition that uses less features in each if-then clause. Should be null or a numeric vector.

Details

The algorithm is quite complicated. See the reference if interested.

Value

An object of class `listdtr`.

References

https://arxiv.org/abs/1606.01472

See Also

predict.listdtr, print.listdtr, plot.listdtr, build.rule.cv

Examples

```
# an example for one-stage study
x <- matrix(rnorm(500), 100, 5)
stage.x <- rep(1, 5)
a <- rbinom(100, 1, 0.5)
y <- a * x[, 1] + rnorm(100, 0, 0.1)
dtr <- listdtr(y, a, x, stage.x)

dtr               # display the regime in words
plot(dtr)         # display the regime in diagrams
yrec <- predict(dtr, x, 1)
yopt <- ifelse(x[, 1] > 0, 1, 0)
table(yrec, yopt)  # discrepancy between recommended and optimal

# an example for two-stage study
x <- matrix(rnorm(500), 100, 5)
stage.x <- c(1, 1, 1, 2, 2)
a1 <- rbinom(100, 1, 0.5)
a2 <- rbinom(100, 1, 0.5)
y1 <- rep(0, 100)
y2 <- 9 * a1 * sin(x[, 1] * pi / 8) - 2 * a2 * x[, 4] + rnorm(100)
dtr <- listdtr(cbind(y1, y2), cbind(a1, a2), x, stage.x)

dtr               # display the regime in words
plot(dtr)         # display the regime in diagrams
yrec <- predict(dtr, x, 1)
yopt <- ifelse(x[, 1] > 0, 1, 0)
table(yrec, yopt)  # discrepancy between recommended and optimal
```

---

plot.listdtr *Representation of List-based Regimes in Diagrams*

Description

Describe the given regime in diagrams and plot them.

Usage

```
## S3 method for class 'listdtr'
plot(x, stages = NULL, digits = 3L, ...)
```

Arguments

x	an object of class listdtr.
stages	an integer scalar / vector that specifies the stage(s) of interest. Default to all stages.
digits	number of decimal digits to show.
...	further arguments passed to or from other methods.

Value

Return object invisibly.

See Also

listdtr, draw.rule

Examples

```
# see examples for listdtr
```

---

predict.krr *Prediction for Kernel Ridge Regression*

Description

Predict response given predictors, based on a krr object.

Usage

```
## S3 method for class 'krr'
predict(object, xnew, ...)
```

Arguments

object	an object of classkrr, usually the return value of function krr
xnew	a matrix of predictors.
...	further arguments passed to or from other methods.

Value

A vector of predicted response.

See Also

krr

Examples

```
# see examples for krr
```

predict.listdtr — *Treatment Recommendation Dictated by List-based Optimal Dynamic Treatment Regime*

Description

Provide treatment recommendation at a given stage using the features, based on a given `listdtr` object.

Usage

```
## S3 method for class 'listdtr'
predict(object, xnew, stage, ...)
```

Arguments

`object`	an object of class `listdtr`, usually the return value of function `listdtr`.
`xnew`	a matrix of features.
`stage`	an integer that specifies the stage.
`...`	further arguments passed to or from other methods.

Value

A factor vector that gives the estimated optimal treatment for the features presented in each row of `xnew`. The length is the same as the number of rows in `xnew`. The levels are the unique values of treatments actually received at that stage.

See Also

`listdtr`, `apply.rule`

Examples

```
# see examples for listdtr
```

print.listdtr — *Representation of List-based Regimes in Words*

Description

Describe the given regime in words and print them.

Usage

```
## S3 method for class 'listdtr'
print(x, stages = NULL, digits = 3L, ...)
```

Arguments

x	an object of class listdtr.
stages	an integer scalar / vector that specifies the stage(s) of interest. Default to all stages.
digits	number of decimal digits to show.
...	further arguments passed to or from other methods.

Value

Return object invisibly.

See Also

listdtr, show.rule

Examples

```
# see examples for listdtr
```

Index