

Package ‘partDSA’

February 15, 2012

Type Package

Title Partitioning using deletion, substitution, and addition moves

Version 0.8.4

Author Annette Molinaro <annette.molinaro@ucsf.edu>, Karen Lostritto
<karen.lostritto@yale.edu>, Gregory Ryslik <gregory.rysluk@yale.edu>, Steve Weston
<stephen.weston@yale.com>

Maintainer Annette Molinaro <annette.molinaro@ucsf.edu>

Description partDSA is a novel tool for generating a piecewise
constant estimation list of increasingly complex predictors
based on an intensive and comprehensive search over the entire covariate space.

Depends R (>= 2.5), survival

Suggests nws, MASS, ipred, VGAM

License GPL-2

Repository CRAN

Date/Publication 2012-01-25 06:44:30

R topics documented:

dumpDSA	2
loglevel	2
partDSA	3
predict.dsa	5
print.dsa	5
showDSA	6
trim	7

Index	8
--------------	----------

dumpDSA	<i>dumpDSA</i>
---------	----------------

Description

dumpDSA is used to create an XML file containing visualization information for a partDSA object.

Usage

```
dumpDSA(x, file=stdout())
```

Arguments

x	partDSA or dsa object to be dumped.
file	file to write data to.

loglevel	<i>Set Logging Level</i>
----------	--------------------------

Description

loglevel is used to set the logging level for the DSA package.

Usage

```
loglevel(lev)
```

Arguments

lev	The default level to use. The level can be specified as error, warn, info, debug, or verbose.
-----	---

Examples

```
loglevel(debug)
```

partDSA	<i>partDSA</i>
---------	----------------

Description

partDSA is a novel tool for generating a piecewise constant estimation sieve of candidate estimators based on an intensive and comprehensive search over the entire covariate space. The strength of this algorithm is that it builds 'and' and 'or' statements. This allows combinations and substitutions of regions for the purpose of discovering intricate correlations patterns and interactions in addition to main effects. Depending on the application, this approach will supersede methods such as CART by being not only more aggressive but also more flexible. As such, *partDSA* provides the user an additional tool for their statistical toolbox.

Usage

```
partDSA(x, y, wt=rep(1, nrow(x)), x.test=x, y.test=y, wt.test,
        control=DSA.control(), sleigh)
DSA.control(vfold=10, minbuck=6, cut.off.growth=10, MPD=0.1,
            missing="impute.at.split", loss.function="default",
            wt.method="KM", brier.vec=NULL,
            leafy=0, leafy.random.num.variables.per.split=4,
            leafy.num.trees=50, save.input=FALSE, cox.vec=NULL, IBS.wt=NULL)
```

Arguments

x	The matrix or data frame of predictor variables for the training set, used to build the model. Each row corresponds to an observation, and each column corresponds to a variable.
y	The outcome (response) vector, either continuous or categorical, representing the true response values for observations in x. The length of this vector should equal the number of rows in x.
wt	Optional vector of training weights with length equal to the number of observations in x. Default is a vector of ones with length equal to the number of training set observations.
x.test	The matrix or data frame of predictor variables used to build the model. The number of columns (variables) of x.test should equal the number of columns as x. The default is x.
y.test	The outcome (response) vector, either continuous or categorical, representing the true response values for observations in x.test. The length of this vector should equal the number of rows in x.test. The default value is y.
wt.test	Optional vector of test weights with length equal to the number of test set observations. Default value is wt if x.test wasn't specified, otherwise it is a vector of ones with length equal to the number of test set observations.
control	A list object used to specify additional control parameters. This is normally created by calling the DSA.control function. Default value is the result of calling DSA.control with no arguments.

sleigh	Optional sleigh object to allow the cross-validation to be performed in parallel using the nws package. If not specified, the cross-validation will be executed sequentially.
vfold	The number of folds of cross-validation for the model building process. The default value is 10.
minbuck	The minimum number of observations in any terminal partition. The default value is 6.
cut.off.growth	The maximum number of terminal partitions to be considered when building the model. The default value is 10.
MPD	Minimum Percent Difference. The model fit must improve by this percentage in order to be considered. This saves time in the model building process. The default value is 0.1.
missing	Character string specifying how missing data should be handled. The default value is "no." See the details section for more information.
loss.function	The function to be minimized when building the model. For categorical outcomes, "entropy" (default) or "gini" can be specified. For continuous outcomes, the L2 loss function is used.
wt.method	Not documented yet.
brier.vec	Not documented yet.
cox.vec	Not documented yet.
IBS.wt	Not documented yet.
leafy	Not documented yet.
leafy.random.num.variables.per.split	Not documented yet.
leafy.num.trees	Not documented yet.
save.input	Indicates if x and y should be saved in the object returned by partDSA. If FALSE, x and y are set to NULL. The default value is FALSE.

Details

missing set to "no" indicates that there is no missing data and will create an error if missing data is found in the dataset. Setting missing="impute.at.split" will use a data imputation method similar to that in CRUISE (Kim and Loh, 2001). At each split, the non-missing observations for a given variable will be used to find the best split, and the missing observations will be imputed based on the mean or mode (depending on whether the variable is categorical or continuous) of the non-missing observations in that node. Once the node assignment of these missing observations is determined using the imputed values, the imputed values are returned to their missing status. For missing values in the test set, the grand mean or mode from the corresponding variables in the training set are used. Including variables which are entirely missing will result in an error.

Examples

```
library(MASS)
set.seed(6442)
```

```

n <- nrow(Boston)
tr.n <- floor(n / 2)
train.index <- sample(1:n, tr.n, replace=FALSE)
test.index <- (1:n)[-train.index]

x <- Boston[train.index, -14]
y <- Boston[train.index, 14]
x.test <- Boston[test.index, -14]
y.test <- Boston[test.index, 14]

control <- DSA.control(vfold=1) # no cross-validation
partDSA(x, y, x.test=x.test, y.test=y.test, control=control)

```

predict.dsa

predict.dsa

Description

Predicted values based on the DSA algorithm.

Usage

```

## S3 method for class 'dsa'
predict(object, newdata1, ...)

```

Arguments

object	a partDSA or dsa object.
newdata1	a data frame in which to look for variables with which to predict.
...	unused.

print.dsa

Print a partDSA Object

Description

These functions print partDSA and dsa objects.

Usage

```

## S3 method for class 'partDSA'
print(x, ...)
## S3 method for class 'dsa'
print(x, ...)

```

Arguments

x	fitted model object of class partDSA or dsa. These are the result of calling the partDSA or rss.dsa functions.
...	unused.

showDSA	<i>showDSA</i>
---------	----------------

Description

showDSA is a convenience function that starts the Java visualization program (included in the partDSA package) to display information about the model created by partDSA function. There are two methods: one taking a partDSA/dsa object, the other taking the name of a visualization file created via the dumpDSA function.

Usage

```
showDSA(x, javacmd=getOption("javacmd"), quietly=FALSE, ...)
## S3 method for class 'dsa'
showDSA(x, javacmd=getOption("javacmd"), quietly=FALSE, ...)
## S3 method for class 'character'
showDSA(x, javacmd=getOption("javacmd"), quietly=FALSE, wait=FALSE, ...)
```

Arguments

x	dsa object or the name of the visualization file to be shown.
javacmd	A character string giving the path of the Java interpreter. Defaults to getOption("javacmd"), which is not set by default, in which case, "java" is used.
quietly	A logical value indicating if messages should be suppressed. Defaults to FALSE.
wait	A logical value indicating if the R session should wait until the the Java program exits before continuing. Defaults to FALSE.
...	Not currently used.

Note

Java 1.5 or greater is required to run these functions, although Java 1.6 is recommended. The Java interpreter should be in the command search path, unless either the javacmd option is set, or the javacmd argument is specified.

trim	<i>trim</i>
------	-------------

Description

trim is used to trim, or prune, the model object returned by the `partDSA` function.

Usage

```
trim(object, ...)  
## S3 method for class 'partDSA'  
trim(object, cut.off.growth, ...)  
## S3 method for class 'dsa'  
trim(object, cut.off.growth, ...)
```

Arguments

object	partDSA or dsa object to be trimmed.
cut.off.growth	number of the level to trim to.
...	unused.

Index

*Topic **utilities**

- dumpDSA, [2](#)
- loglevel, [2](#)
- partDSA, [3](#)
- predict.dsa, [5](#)
- print.dsa, [5](#)
- showDSA, [6](#)
- trim, [7](#)

DSA.control (partDSA), [3](#)
dumpDSA, [2](#)

loglevel, [2](#)

partDSA, [3](#)
predict.dsa, [5](#)
print.dsa, [5](#)
print.partDSA (print.dsa), [5](#)

showDSA, [6](#)

trim, [7](#)