Package 'SoftRandomForest'

May 15, 2019

Version 0.1.0

Date 2019-05-05

Title Classification Random Forests for Soft Decision Trees

Author Gregory Hilleren <Gregory.Hilleren@gmail.com>

Maintainer Gregory Hilleren <Gregory.Hilleren@gmail.com>

Depends R (>= 3.4.0)

Description Performs random forests for soft decision trees for a classification problem. Current limitations are for a maximum depth of 5 resulting in 16 terminal nodes. Some data cleaning is required before input. Final graphic output requires currently requires exporting to 'Microsoft Excel' for visualization. Method based on Irsoy, Yildiz and Alpaydin (2012, ISBN: 978-4-9906441-1-6).

Imports boot, utils

License CC0

URL https://github.com/GregHilleren/SoftRandomForest

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-05-15 13:20:03 UTC

R topics documented:

`BestForestSplit` *Choosing the best variable for splitting.*

Description

`BestForestSplit` searches through possible variables in order to find the most accurate split. It returns the variable chosen, the model, and the two sets of fitted values where both 0 or 1 are considered a "success."

Usage

```
BestForestSplit(response, data, num.features, ntry, weights = rep(1,
  nrow(data)))
```

Arguments

`response`	Logical vector of 0 and 1 denoting the binomial response.
`data`	A data frame or matrix consisting of all possible variables to attempt.
`num.features`	A numeric of the number of variables in the dataset to possibly try. The leftmost number of variables in the dataset are the variables chosen.
`ntry`	A numeric of the number of variables from the `num.features` to attempt to split. This is useful for building random forests. For a standard tree, choose `ntry = num.features`.
`weights`	A vector of weights for use in Weighted Least Squares. Defaults to a vector of 1.

Details

`BestForestSplit` searches through possible variables to split using single variable logistic regression with prior weights in the iteratively reweighted least squares procedure. The variable minimizing residual deviance is chosen. Note, this is a valid choice since all models being compared are using the same Null Model containing only the intercept with equal weights.

Value

List of elements

`Feature`	Returns the variable chosen for best split.
`fit`	A `glm` object of the fit with the chosen variable.
`weights0`	A vector of the weights if response `0` was considered a success. Calculated as $1 - weights1$.
`weights1`	A vector of the weights if response `1` was considered a success.

ClassMode *Determining the mode from a vector of numbers.*

Description

ClassMode returns the most commonly occurring number from a vector of numbers.

Usage

```
ClassMode(classes)
```

Arguments

classes	A character vector of elements to find the mode.

Details

ClassMode creates a frequency table based on a vector. This table is then sorted by highest frequency and returns the top element. Note, in the case of a tie, the first element is chosen as opposed to a random tiebreaker or returning both elements. For the sake of developing Random Forests of SDTs, this is a safer option.

Value

The mode found.

Examples

```
Input = c("A", "C", "B", "B", "A", "B")
ClassMode(Input)
```

SoftClassForest *Implementing a Random Forest of SDTs.*

Description

SoftClassForest creates categorical Random Forests of Soft Decision Trees while returning the fitted classification given by the majority vote of individual SDTs.

Usage

```
SoftClassForest(trainresponses, train, test, ntry, ntrees, depth,
  bag = TRUE)
```

Arguments

trainresponses	A matrix or data frame of responses 0 and 1 for the training set with length equal to the number of observations in the training set and width equal to the number of possible classifications.
train	A matrix or data frame consisting of all possible variables to attempt for the training set.
test	A matrix or data frame consisting of all possible variables to attempt for the test set.
ntry	A numeric of the number of variables from the num.features to attempt to split. This is useful for building random forests. For a standard tree, choose ntry = num.features.
ntrees	A numeric of the number of SDTs to build in the Random Forest.
depth	A numeric of the number of the depth each SDT should be. Here this ends with $2^{depth-1}$ terminal nodes.
bag	Logical if Random Forests should be built with bootstrap aggregating (TRUE) or raw data (FALSE).

Details

SoftClassForest individually fits a Random Forest for each possible classification response using SoftForestPredFeeder function one classification at a time. The result from each one of these SDTs is a fitted probability of 0 or 1. Once all classifications have a fitted probability, the observation is classified as the maximum a posteriori probability. Given a Random Forest of SDTs, the final Random Forest classification goes to the majority vote from the SDTs.

Value

A vector of the final classifications based on the Random Forest generated.

Examples

```
Responses = SoftClassMatrix(as.vector(iris$Species))
SoftClassForest(trainresponses = Responses, train = iris[,1:4], test = iris[,1:4],
ntry = 2, ntrees = 15, depth = 2, bag = TRUE)
```

SoftClassMatrix *Converting response vector to sparse matrix.*

Description

SoftClassMatrix converts a classification response matrix into a sparse matrix that can be used for inputs into the SoftRandomForest function.

Usage

```
SoftClassMatrix(responses, classes = NA)
```

Arguments

responses	A vector of classification responses.
classes	A vector of possible classifications with a manually specified order. Must contain all elements in the responses vector.

Details

SoftClassMatrix runs through each line of a classification vector and creates a sparse matrix where each column represents an individual classification. The output matrix has number of rows equal to the number of rows of the input vector and number of columns equal to the number of unique entries in the input vector. The order is determined by the order they appear in the vector. Adjust this afterwards if another order is desired.

Value

A matrix where 1 indicates that observation was classified as that column's response and 0 if not.

Examples

```
Input = c("A", "C", "B", "B", "A", "B")
SoftClassMatrix(Input, classes = c("A", "B", "C", "D"))
```

SoftForestPredDepth1 *Building a single level for the Random Forest of SDTs.*

Description

Building a single level for the Random Forest of SDTs.

Usage

```
SoftForestPredDepth1(trainresponse, train, test, num.features, ntry,
  keep = FALSE)
```

Arguments

trainresponse	A vector of responses 0 and 1 for the training set with length equal to the number of observations in the training set.
train	A matrix or data frame consisting of all possible variables to attempt for the training set.
test	A matrix or data frame consisting of all possible variables to attempt for the test set.
num.features	The number of variables in the dataset to possibly try. The leftmost number of variables in the dataset are the variables chosen.
ntry	The number of variables from the num.features to attempt to split. This is useful for building random forests. For a standard tree, choose ntry = num.features.
keep	Logical if weights from a single observation should be kept. Keep FALSE if a Random Forest is to be built.

Value

A vector of the final fitted probabilities for this classification.

`SoftForestPredFeeder` *Choosing the appropriate depth function.*

Description

`SoftForestPredFeeder` is used inside the user interface function to choose the appropriate depth function since the SDT's depth are not generated dynamically.

Usage

```
SoftForestPredFeeder(trainresponse, train, test, num.features, ntry, depth)
```

Arguments

`trainresponse`	A vector of responses `0` and `1` for the training set with length equal to the number of observations in the training set.
`train`	A matrix or data frame consisting of all possible variables to attempt for the training set.
`test`	A matrix or data frame consisting of all possible variables to attempt for the test set.
`num.features`	The number of variables in the dataset to possibly try. The leftmost number of variables in the dataset are the variables chosen.
`ntry`	The number of variables from the `num.features` to attempt to split. This is useful for building random forests. For a standard tree, choose `ntry = num.features`.
`depth`	The number of the depth each SDT should be. Here this ends with $2^{depth-1}$ terminal nodes.

Details

`SoftForestPredDepth` chooses the correct of five possible depths that has functioning code. Any invalid attempt returns an error.

Value

The output from the chosen function.

`SoftObservation` *Recording the prediction weights to analyze observation-level patterns*

Description

`SoftObservation` runs the depth given of a single SDT of a single response focusing on a single (or a few) observations in order to make inference from the Prediction Weights found.

Usage

```
SoftObservation(response, responselabel = "No Variable Label", train,
  depth, keep = TRUE, observation, export = FALSE, path = NA)
```

Arguments

`response`	A vector of responses `0` and `1` for a single classification for the training set with length equal to the number of observations in the training set and width `1`.
`responselabel`	A character string of the title of the response variable used.
`train`	A matrix or data frame consisting of the entire dataset to train.
`depth`	A numeric of the number of the depth each SDT should be. Here this ends with $2^{depth-1}$ terminal nodes.
`keep`	A logical passed to the internal functions to keep the prediction and weights (TRUE) or discard the weights and keep only prediction (FALSE).
`observation`	A numeric or vector containing observations of interest to keep the fitted prediction probability and weights.
`export`	A logical indicating if results should be printed directly (FALSE) or exported to csv (TRUE).
`path`	A directory location to save the exported csv file. Must be provided if export = TRUE.

Details

`SoftObservation` runs the internal SoftForestPredDepth functions so that a single SDT's weights can be recorded. This can then be exported to any tree visual representation to see how the observation(s) of interest pass through the SDT This follows from the other user interface function SoftClassForest where the test set is the observation of interest instead of being used for testing misclassification. Exporting these weights for visual representation is possible and recommended.

Value

A list of possible elements

`Prediction`	A vector of fitted probabilities for the given classification and observation(s).
`AllFeatures`	A numeric list of Features chosen at each node where the number represents the column number in the data.

`AllWeights` A matrix of weights where the rows represent the observations and columns represent the weights used at different stages.

`SoftObservationDataOutput.csv`
If export = TRUE, this csv file can be used with an Excel supplement to create visual displays of a single observation for a single response.

Examples

```
Responses = SoftClassMatrix(as.vector(iris$Species))
SoftObservation(response = Responses[,1], responselabel = "setosa", train = iris[,1:4],
depth = 2, keep = TRUE, observation = 34, export = TRUE, path = tempdir())
```

Index