iwLF manual

(R program)

R topics documented:

`iwLF.R` iterative weighted Logic Forest

Description

The iterative weighted Logic Forest (iwLF) is a supervised ensemble learning algorithm that utilizes the logic tree as its base learner model. This algorithm is suitable for binary features and, in its current version, is exclusively used for predicting binary responses.

Beyond providing predicted responses, iwLF also offers important scores for the features and their interactions (also known as Prime Implicants, PIs), which are instrumental in predicting the responses. This capability of identifying influential predictors and interactions makes iwLF a powerful tool for model interpretation.

Details

The `iwLF` package provides three primary functions to run the algorithm:

`train.iwLF`: This function is used to construct the iwLF model. Given the training data and appropriate parameters, this function will create a model that can be used for making predictions.

`predict.iwLF`: This function applies the previously constructed iwLF model to new data in order to predict responses. It takes a fitted iwLF model and the features of new dataset as input, and outputs the predicted responses.

`plot.iwLF`: This function is utilized to create plots illustrating the relationship between important PIs and their associated importance scores. It can be used for better understanding and visualization of the model's behavior and influential features.

Each of these functions and their associated settings will be explained in greater detail in the following sections.

The iwLF package depends on several other R packages, including caret, logicFS, pacman, LogicForest, parallel, PRROC, e1071, ggplot2, and dplyr. Please install and load these packages in your R environment before using iwLF.

References

Yu-Chung Wei and Ying-Chi Chen. Iterative Weighted Logic Forest Approach to Identify Important Genes and Interactions.

Examples

```
## Load the source file
source("iwLF.R")

## Read the features and responses of training data
Xs=read.table("Train.X.txt",header=T)
reps=scan("Train.Y.txt")

## Train the iwLF model
TrainModel=train.iwLF (resp=reps, Xs=Xs,
   nBS=100, nBSXVars=floor(ncol(Xs)^0.7), maxK=10,
   anneal.params= logreg.anneal.control(start=2,
   end=-1,  iter=50000))

## Read the features of new data
newdata=read.table("Test.X.txt",header=T)

## Use the trained model to predict the responses for the new
data
predict.iwLF (TrainModel, newdata)

## Plot the importance scores of the most important PIs
plot.iwLF (TrainModel, TopN=5)
```

`data` Example datasets

Description

Two datasets are provided: a training dataset (inclusive of features and responses) and a testing dataset (comprising only features).

## Training dataset

`Train.X.txt`: This file contains the features for the training samples. It includes 200 samples, each with 100 features.

`Train.Y.txt`: This file contains the corresponding binary responses for the training samples. Among the 200 samples, 100 samples have a response of 1, and the other 100 have a response of 0.

`## Testing dataset`

`Test.X.txt:` This file includes the features of 2 testing samples, each consisting of 100 features.

These datasets can be used to demonstrate the functionality of the iwLF package in the R environment. For a more comprehensive understanding of how to use the iwLF functions with these datasets, please refer to the corresponding sections of this manual.

`train.iwLF` Construct iwLF model

Description

This function is used to construct the iwLF model from the training dataset.

Usage

```
train.iwLF (resp, Xs,
   nBS=100, nBSXVars=floor(ncol(Xs)^0.7), maxK=10,
   anneal.params= logreg.anneal.control(start=2,
   end=-1,  iter=50000))
```

Argument

`resp` a numeric vector that represents the value of the response variable for the training samples. Recommended values for this binary response variable are either 0 (negative) or 1 (positive).

`Xs` A data frame that contains the values of the features for the training samples. In the data frame, each row corresponds to a sample, while each column corresponds to a feature.

The recommended values for these binary features are either 0 or 1.

`nBS`

`numeric`, number of base learner models (logic trees) used in each iteration of the iwLF algorithm. The default value is 100.

`nBSXVars`

`numeric`, number of candidate features considered by each base learner model (logic tree). The default is the 0.7 power of the total number of features.

`maxK`

`numeric`, the maxium number of iterations of the iwLF algorithm. The default value is 10.

`anneal.params`

control of simulated annealing parameters needed in the construction of each logic tree. The parameters are set using the `logreg.anneal.control` function from the `LogicReg` package.

There are three essential parameters.

`start` is the upper temperature (on a log10 scale) in the annealing chain. The default value is 2 (ie. temperature 10^2=100).

`end` is the lower temperature (on a log10 scale) in the annealing chain. The default value is -1 (ie. temperature 10^{-1}=0.1).

`iter` is the total number of iterations in the annealing chain. The default value is 50000.

Value

`bestK` `numeric`, the best iteration in the iterative process of iwLF.

`err_list` `numeric`, the out-of-bag error rate for each iteration in the iwLF.

`f1_score_list` `numeric`, the F1-score for each iteration in the iwLF.

`AUPRC_list` `numeric`, the area under the precision-recall curve (AUPRC) for each iteration in the iwLF.

`TP_list` `numeric`, the number of true positive (TP) samples for each iteration in the iwLF.

`TN_list` `numeric`, the number of true negative (TN) samples for each iteration in the iwLF.

`FP_list` `numeric`, the number of false positive (FP) samples for each iteration in the iwLF.

`FN_list` `numeric`, the number of false negative (FN) samples for each iteration in the iwLF.

Example

```
Xs=read.table("Train.X.txt",header=T)
reps=scan("Train.Y.txt")

TrainModel=train.iwLF (resp=reps, Xs=Xs,
    nBS=100, nBSXVars=floor(ncol(Xs)^0.7), maxK=10,
    anneal.params= logreg.anneal.control(start=2,
    end=-1,  iter=50000))
```

`predict.iwLF` predicting responses for new data

Description

This function is designed to apply a pre-existing iwLF model to new data to generate predictions. This function requires a fitted iwLF model and the features of a new dataset as inputs and produces the predicted responses as outputs.

Usage

```
predict.iwLF (object, newdata)
```

Argument

`object` pre-constructed iwLF model object

`newdata` a data frame that contains the values of the features for the new dataset. In the data frame, each row corresponds to a sample, while each column corresponds to a feature. The recommended values for these binary features are either 0 or 1.

Value

`LFprediction` `numeric`, the predicted value of the response variable for each sample

`proportion_one` `numeric`, the proportion of predictions which are predicted as 1 by all logic trees in the final iwLF model for each sample in the new dataset

`AllTrees` `numeric`, the predicted value of all logic trees within the final iwLF model for each sample in the new dataset

Example

```
newdata=read.table("Test.X.txt",header=T)
predict.iwLF (TrainModel, newdata)
```

`plot.iwLF` visualizing prime implicants (PIs) and importance scores

Description

This function is utilized to create bar charts illustrating the relationship between important prime implicants (PIs) and their corresponding importance scores. This function provides a valuable tool for visualizing the contribution of each PI in the model, which is essential for understanding model interpretation.

Usage

```
plot.iwLF (object, TopN)
```

Argument

`object`	pre-constructed iwLF model object
`TopN`	`numeric`, the number of PIs with the highest importance scores to be displayed in the figure