# Package ‘imptree’

October 13, 2022

**Type**  Package

**Title**  Classification Trees with Imprecise Probabilities

**Version**  0.5.1

**Date**  2018-08-16

**Description**  Creation of imprecise classification trees. They rely on probability estimation within each node by means of either the imprecise Dirichlet model or the nonparametric predictive inference approach. The splitting variable is selected by the strategy presented in Fink and Crossman (2013) [http://www.sipta.org/isipta13/index.php?id=paper&paper=014.html], but also the original imprecise information gain of Abellan and Moral (2003) [doi:10.1002/int.10143] is covered.

**License**  GPL (>= 2)

**Encoding**  UTF-8

**Imports**  Rcpp (>= 0.12.5)

**LinkingTo**  Rcpp

**SystemRequirements**  C++11

**RoxygenNote**  6.1.0

**Suggests**  testthat

**NeedsCompilation**  yes

**Author**  Paul Fink [aut, cre]

**Maintainer**  Paul Fink <paul.fink@stat.uni-muenchen.de>

**Repository**  CRAN

**Date/Publication**  2018-08-17 08:50:06 UTC

## R topics documented:

- `imptree-package`
- `carEvaluation`
- `imptree`

2 3 4
imptree-package

imptree-package  imptree: Classification Trees with Imprecise Probabilities

Description

The imptree package implements the creation of imprecise classification trees based on algorithm developed by Abellan and Moral. The credal sets of the classification variable within each node are estimated by either the imprecise Dirichlet model (IDM) or the nonparametric predictive inference (NPI). As split possible split criteria serve the ‘information gain’, based on the maximal entropy distribution, and the adaptable entropy-range based criterion proposed by Fink and Crossman. It also implements different correction terms for the entropy.

The performance of the tree can be evaluated with respect to the common criteria in the context of imprecise classification trees.

It also provides the functionality for estimating credal sets via IDM or NPI and obtain their minimal/maximal entropy (distribution) to be used outside the tree growing process.

References


See Also

imptree for tree creation, probInterval for the credal set and entropy estimation functionality
Examples

data("carEvaluation")

## create a tree with IDM (s=1) to full size
## carEvaluation, leaving the first 10 observations out
ip <- imptree(acceptance~., data = carEvaluation[-(1:10),],
               method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
               control = list(depth = NULL, minbucket = 1))

## summarize the tree and show performance on training data
summary(ip)

## predict the first 10 observations
## Note: The result of the prediction is return invisibly
pp <- predict(ip, dominance = "max", data = carEvaluation[(1:10),])

## print the general evaluation statistics
print(pp)

## display the predicted class labels
pp$classes

---

**carEvaluation**

*Car Evaluation Database*

Description

This data.frame contains the 'Car Evaluation' data set from the UCI Machine Learning Repository. The 'Car Evaluation data' set gives the acceptance of a car directly related to the six input attributes: buying, maint, doors, persons, lug_boot, safety.

Usage

data(carEvaluation)

Format

A data frame with 1728 observations on the following 7 variables, where each row contains information on one car. All variables are factor variables.

- **buying**: Buying price of the car (Levels: high, low, med, vhigh)
- **maint**: Price of the maintenance (Levels: high, low, med, vhigh)
- **doors**: Number of doors (Levels: 2, 3, 4, 5more)
- **persons**: Capacity in terms of persons to carry (Levels: 2, 4, more)
- **lug_boot**: Size of luggage boot (Levels: big, med, small)
- **safety**: Estimated safety of the car (Levels: high, low, med)
- **acceptance**: Acceptance of the car (target variable) (Levels: acc, good, unacc, vgood)
Details

Car Evaluation Database was derived from a simple hierarchical decision model originally developed for the demonstration of DEX.

The model evaluates cars according to the following concept structure:

\[
\begin{align*}
\text{CAR} & : \quad \text{car acceptability} \\
. \text{PRICE} & : \quad \text{overall price} \\
. . \text{buying} & : \quad \text{buying price} \\
. . \text{maint} & : \quad \text{price of the maintenance} \\
. \text{TECH} & : \quad \text{technical characteristics} \\
. . \text{COMFORT} & : \quad \text{comfort} \\
. . . \text{doors} & : \quad \text{number of doors} \\
. . . \text{persons} & : \quad \text{capacity in terms of persons to carry} \\
. . . \text{lug\_boot} & : \quad \text{the size of luggage boot} \\
. . . \text{safety} & : \quad \text{estimated safety of the car}
\end{align*}
\]

Input attributes are printed in lowercase. Besides the target concept (CAR), the model includes three intermediate concepts: PRICE, TECH, COMFORT.

The Car Evaluation Database contains examples with the structural information removed, i.e., directly relates CAR to the six input attributes: buying, maint, doors, persons, lug_boot, safety.

Source

The original data were taken from the UCI Machine Learning repository (https://archive.ics.uci.edu/ml/datasets/Car+Evaluation) and were converted into R format by Paul Fink.

References


Examples

```r
data("carEvaluation")
summary(carEvaluation)
```
Description

`imptree` implements Abellan and Moral's tree algorithm (based on Quinlan's ID3) for classification. It employs either the imprecise Dirichlet model (IDM) or nonparametric predictive inference (NPI) to generate the imprecise probability distribution of the classification variable within a node.

Usage

```r
## S3 method for class 'formula'
imptree(formula, data = NULL, weights, control,
        method = c("IDM", "NPI", "NPIapprox"), method.param, ...)

## Default S3 method:
imptree(x, y, ...)
```

Arguments

- `formula`: Formula describing the structure (class variable ~ feature variables). Any interaction terms trigger an error.
- `data`: Data.frame to evaluate supplied formula on. If not provided the the formula is evaluated on the calling environment.
- `weights`: Individual weight of the observations (default: 1 to each). This argument is ignored at the moment.
- `control`: A named (partial) list according to the result of `imptree_control`.
- `method`: Method applied for calculating the probability intervals of the class probability. "IDM" for the imprecise Dirichlet model (default), "NPI" for use of the nonparametric predictive inference approach and "NPIapprox" for use of the approximate algorithm obtaining maximal entropy of NPI generated probability intervals.
- `method.param`: Named list providing the method specific parameters. See `imptree_params`.
- `...`: Optional parameters to be passed to the main function `imptree.formula` or to the call of `imptree_control`.
- `x`: A data.frame or a matrix of feature variables. The columns are required to be named.
- `y`: The classification variable as a factor.

Value

An object of class `imptree`, which is a list with the following components:

- `call`: Original call to `imptree`.
- `tree`: Object reference to the underlying C++ tree object.
- `train`: Training data in the form required by the workhorse C++ function. It is an integer matrix containing the internal factor representations, adjusted for the C++ specific indexing starting at 0 and not at 1 as in R. Further attributes of
the matrix, hold the names of the variables, the C++ adjusted index of the classification variable, as well as the levels and number of levels for each variable.

**formula**
The formula describing the data structure

**Author(s)**
Paul Fink <Paul.Fink@stat.uni-muenchen.de>, based on algorithms by J. Abellán and S. Moral for the IDM and R. M. Baker for the NPI approach.

**References**
Baker, R. M. (2010), *Multinomial Nonparametric Predictive Inference: Selection, Classification and Subcategory Data*.

**See Also**
`predict.imptree` for prediction, `summary.imptree` for summary information, `imptree_params` and `imptree_control` for arguments controlling the creation, `node_imptree` for accessing a specific node in the tree

**Examples**
```r
data("carEvaluation")

## create a tree with IDM (s=1) to full size on
## carEvaluation, leaving the first 10 observations out
imptree(acceptance~., data = carEvaluation[-(1:10),],
    method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
    control = list(depth = NULL, minbucket = 1)) # control args as list

## same setting as above, now passing control args in '...
imptree(acceptance~., data = carEvaluation[-(1:10),],
    method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
    depth = NULL, minbucket = 1)
```

**imptree_control**
Control parameters for generating imptree objects

**Description**
Initializing and validating the tree generation parameters
Usage

imptree_control(splitmetric, controlList = NULL, tbase = 1, gamma = 1, depth = NULL, minbucket = 1L, ...)

Arguments

- **splitmetric**: Choosen split metric as integer: 0 means "globalmax" and 1L "range", respectively. See imptree_params
- **controlList**: Named list containing the processed arguments. See details.
- **tbase**: Value that needs to be at least attained to qualify for splitting (default: 1)
- **gamma**: Weighting factor of the maximum entropy (default: 1)
- **depth**: Integer limiting the tree to the given depth, with 0 indicating to perform no splitting at all. If not supplied, NULL (default) or negative the tree is grown to maximal size, the latter triggering a warning.
- **minbucket**: Positive integer as minimal leaf size (default: 1)
- **...**: Argument gobbling; is not processed

Details

The argument controlList may be a named list with names in c("tbase", "gamma", "depth", "minbucket") Any values in this list will overwrite those supplied in named arguments. When controlList = NULL (default) only the supplied arguments are checked.

In case controlList contains an argument named splitmetric, this will be ignored. If splitmetric is 0L, i.e. "globalmax", the values for gamma and tbase are set to their default values, even if the user supplied different values.

Value

A list containing the options. Missing options are set to their default value.

Author(s)

Paul Fink <Paul.Fink@stat.uni-muenchen.de>

See Also

imptree, imptree_params

Examples

```r
## Check performed for splitmetric 'globalmax',
## tbase is default generated and 'gamma' is overwritten
## (see Details), tree is grown to full depth and
## at least 5 observations are needed to be within each node
imptree_control(splitmetric = 0, gamma = 0.5,
    depth = NULL, minbucket = 5)

## Passing some control arguments in a list
```
```r
# As splitmetric is 'range', gamma is respected
imptree_control(splitmetric = 1, minbucket = 5,
                controlList = list(gamma = 0.5, depth = NULL))
```

## node_imptree

### Classification with Imprecise Probabilities

#### Description

Access probability information of nodes

#### Usage

```r
node_imptree(x, idx = NULL)
```

#### Arguments

- `x`: An object of class `imptree` or `node_imptree`, respectively. See details.
- `idx`: numeric or integer vector of indices specifying the sequential node access from the root node. Numeric values are coerced to integer as by `as.integer` (and hence truncated towards zero).
  - If `NULL` the probability information of the root node are accessed.
- `...`: Further arguments passed to `print` methods

#### Details

This function accesses the properties of a specific node of an imprecise tree. An existence check on the stored C++ object reference is carried out at first. If the reference is not valid the original call for "x" is printed as error.

#### Value

An object of class `node_imptree` containing information on the properties of the node as a list:

- `probint`: matrix containing the bounds of the imprecise probability distribution and the absolute observed frequencies of the classification variable within the node.
- `depth`: The depth of the node with the tree.
- `splitter`: The name of the variable used for splitting as character; NA if node is a leaf.
- `children`: The number of children of the node.
- `traindataIdx`: Vector giving the indexes of the training data contained within the node
- `ipmodel`: List giving details about the used imprecise probability model to obtain the credal set:
iptype used IP model: "IDM", "NPI" or "NPIapprox"

s If iptype == "IDM" the IDM's parameter 's', otherwise this list entry is missing

The printing function returns the node_imptree object invisibly.

Author(s)

Paul Fink <Paul.Fink@stat.uni-muenchen.de>

See Also

imptree, for global information on the generated tree summary.imptree

Examples

data("carEvaluation")

## create a tree with IDM (s=1) to full size
## carEvaluation, leaving the first 10 observations out
ip <- imptree(acceptance~., data = carEvaluation[-(1:10),],
method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
control = list(depth = NULL, minbucket = 1))

## obtain information on the root node
node_imptree(x = ip, idx = NULL)

## obtain information on the 2nd note in the 1st level
node_imptree(x = ip, idx = c(1, 2))

## reference to an invalid index and/or level generates error
## Not run:
node_imptree(x = ip, idx = c(1,10)) # no 10th node on 1st level

## End(Not run)
predict.imptree

Usage

## S3 method for class 'imptree'
predict(object, data, dominance = c("strong", "max"),
    utility = 0.65, ...)

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

Arguments

object An object of class imptree. See details.
data Data.frame containing observations to be predicted. If NULL the observations in the training set of "object" are employed.
dominance Dominance criterion to be applied when predicting classes. This may either be "strong" (default) or "max". See details.
utility Utility for the utility based accuracy measure for a vacuous prediction result (default: 0.65).
... Additional arguments for data. May be "weights", "subset", "na.action", any further are discarded.
x an object of class evaluation_imptree

details

This function carries out the prediction of an imprecise tree. An existence check on the stored C++ object reference is carried out at first. If the reference is not valid the original call for "object" is printed as error.

There are currently 2 different dominance criteria available:

max Maximum frequency criterion. Dominance is decided only by the upper bound of the probability interval, ie. a state \( C_i \) is dominated if there exists any \( j \neq i \) with \( u(C_i) < u(C_j) \)

strong Interval dominance criterion. For the IDM it coincides with the strong dominance criterion. Here a state \( C_i \) is dominated if there exists any \( j \neq i \) with \( u(C_i) < l(C_j) \)

value

predict.imptree() return an object of class evaluation_imptree, which is a named list containing predicted classes, predicted probability distribution and accuracy evaluation

probintlist List of the imprecise probability distributions of the class variable. One matrix per observation in the test data.
classes Predicted class(es) of the observations as boolean matrix
evaluation Result of accuracy evaluation
    • nObs: Number of observations
    • deter: Determinacy
    • nObsIndet: Number of observations with indeterminate prediction
• indetSize: Average number of classes when predicting indeterminate (NA when no indeterminate observation)
• acc_single: Single-set accuracy (NA when no determine observation)
• acc_set: Set-accuracy (NA when no indeterminate observation)
• acc_disc: Discounted-accuracy
• acc_util: Utility based (discounted) accuracy

The printing function returns the evaluation_imptree object invisibly.

Author(s)
Paul Fink <Paul.Fink@stat.uni-muenchen.de>

See Also
imptree, node_imptree

Examples

data("carEvaluation")

## create a tree with IDM (s=1) to full size
## carEvaluation, leaving the first 10 observations out
ip <- imptree(acceptance~., data = carEvaluation[-(1:10),],
               method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
               control = list(depth = NULL, minbucket = 1))

## predict the first 10 observations with 'max' dominance
pp <- predict(ip, dominance = "max", data = carEvaluation[(1:10),])
print(pp)
pp$classes          ## predicted classes as logical matrix

## predict the first 10 observations with 'strong' dominance and
## use a different level of utility
predict(ip, dominance = "strong", data = carEvaluation[(1:10),],
        utility = 0.5)

print.imptree
Classification with Imprecise Probabilities

Description

Printing the imptree object to console

Usage

## S3 method for class 'imptree'
print(x, digits = getOption("digits"), sep = "\t",
       ...)
print.imptree

Arguments

x
Object of class imptree. See details.
digits
a non-null value for digits specifies the minimum number of significant digits to be printed in values. The default uses `getOption("digits")`. Non-integer values will be rounded down, and only values greater than or equal to 1 and no greater than 17 are accepted.
sep
Separator between the displayed IPDistribution objects. (Default: '\t')
...
Additional arguments; ignored at the moment

Details

An existence check on the stored C++ object reference is carried out at first. If the reference is not valid the original call for "object" is printed as error.

For a more detailed summary of the tree `summary.imptree`.

Value

Returns the calling object invisible.

Author(s)

Paul Fink <Paul.Fink@stat.uni-muenchen.de>

See Also

`imptree`, `summary.imptree`

Examples

data("carEvaluation")

## create a tree with IDM (s=1) to full size
## carEvaluation, leaving the first 10 observations out
ip <- imptree(acceptance~., data = carEvaluation[-(1:10),],
              method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
              control = list(depth = NULL, minbucket = 1))

ip          ## standard printing; same as 'print(ip)'
print(ip, sep = ";")  ## probability intervals are separated by ';'
probInterval

Various method around IPIntervals

Description
Calculation of probability intervals, and their maximal and minimal entropy

Usage
probInterval(table, iptype = c("IDM", "NPI", "NPIapprox"),
entropymin = TRUE, entropymax = TRUE, correction = c("no",
"strobl", "abellan"), s = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>integer vector of absolute frequencies</td>
</tr>
<tr>
<td>iptype</td>
<td>method for calculating the probability intervals of table. &quot;IDM&quot; for the imprecise Dirichlet model (default), &quot;NPI&quot; for use of the nonparametric predictive inference approach and &quot;NPIapprox&quot; for use of the approximate algorithm obtaining maximal entropy of NPI generated probability intervals.</td>
</tr>
<tr>
<td>entropymin</td>
<td>Calculation of one distribution with minimal entropy, including the actual value of the minimal entropy (default: TRUE)</td>
</tr>
<tr>
<td>entropymax</td>
<td>Calculation of the distribution with maximal entropy, including the actual value of the maximal entropy (default: TRUE)</td>
</tr>
<tr>
<td>correction</td>
<td>Entropy correction to be carried out, igorned if (entropymin</td>
</tr>
<tr>
<td>s</td>
<td>Hyperparamter of the IDM (s &gt;= 0), see imptree_params (ignored for iptype == &quot;NPI&quot;)</td>
</tr>
</tbody>
</table>

Value
A list with 5 named entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>probint</td>
<td>matrix with 3 rows and length(table) columns: in the rows are the absosulte frequencies, the lower bound (&quot;lower&quot;) and the upper bound (&quot;upper&quot;) of the event-wise probabilities.</td>
</tr>
<tr>
<td>maxEntDist</td>
<td>The (unique) probability distribution with maximal entropy</td>
</tr>
<tr>
<td>maxEntCorr</td>
<td>The value of the (corrected) maximal entropy</td>
</tr>
<tr>
<td>minEntDist</td>
<td>A probability distribution with minimal entropy, as it is not necessarily unique there may be others</td>
</tr>
<tr>
<td>minEntCorr</td>
<td>The value of the (corrected) minimal entropy</td>
</tr>
</tbody>
</table>

Author(s)
Paul Fink <Paul.Fink@stat.uni-muenchen.de>
See Also

imptree_params

Examples

```r
## Artificial vector of absolute frequencies
obs <- c(a = 1, b = 2, c = 10, d = 30, e = 5)

## probability interval by NPI, including only information on the
## minimum entropy distribution, using no entropy correction
probInterval(obs, iptype = "NPI", entropymax = FALSE)

## probability interval by IDM, including information on the
## minimum and maximum entropy distribution with s = 2 and correction
## according to 'strobl'
probInterval(obs, iptype = "IDM", correction = "strobl", s = 2)
```

summary.imptree

Classification with Imprecise Probabilities

Description

Summary function for an imptree object, assesses accuracy achieved on training data and further
tree properties.

Usage

```r
## S3 method for class 'imptree'
summary(object, utility = 0.65,
         dominance = c("strong", "max"), ...)

## S3 method for class 'summary.imptree'
print(x, ...)
```

Arguments

- `object`: An object of class imptree. See details.
- `utility`: Utility for the utility based accuracy measure for a vacuous prediction result (default: 0.65).
- `dominance`: Dominance criterion to be applied when predicting classes. This may either be "strong" (default) or "max". See details at `predict.imptree`.
- `...`: Further arguments are ignored at the moment.
- `x`: An object of class summary.imptree
Details

An existence check on the stored C++ object reference is carried out at first. If the reference is not valid the original call for "object" is printed as error.

Value

A named list of class summary.imptree containing the tree creation call, accuracy on the training data, meta data and supplied the utility and dominance criterion for evaluation.

call Call to create the tree
utility Supplied utility, or its default value
dominance Supplied dominace criterion, or its default value
sizes List containing the overall number and number of indeterminate predictions on training data
acc named vector containing the accuracy measures on training data with nicer names (without size information) (see predict.imptree)
meta named vector containing the tree’s depth, number of leaves and number of nodes

The printing function returns the summary.imptree object invisibly.

Author(s)

Paul Fink <Paul.Fink@stat.uni-muenchen.de>

See Also

imptree, predict.imptree, for information on a single node node_imptree

Examples

data("carEvaluation")

## create a tree with IDM (s=1) to full size
## carEvaluation, leaving the first 10 observations out
ip <- imptree(acceptance~., data = carEvaluation[-(1:10),],
     method="IDM", method.param = list(splitmetric = "globalmax", s = 1),
     control = list(depth = NULL, minbucket = 1))

## summary including prediction on training data
summary(ip) # default prediction
summary(ip, dominance = "max") # different prediction parameter
Index

* datasets
  carEvaluation, 3
* tree
  imptree, 4
  imptree-package, 2
  imptree_control, 6
  node_imptree, 8
  predict.imptree, 9
  print.imptree, 11
  summary.imptree, 14

as.integer, 8

carEvaluation, 3

ggetOption, 12

imptree, 2, 4, 7, 9, 11, 12, 15
imptree-package, 2
imptree_control, 5, 6, 6
imptree_params, 5–7, 13, 14

node_imptree, 6, 8, 11, 15

predict.imptree, 6, 9, 14, 15
print.evaluation_imptree
  (predict.imptree), 9
print.imptree, 11
print.node_imptree (node_imptree), 8
print.summary.imptree
  (summary.imptree), 14
probInterval, 2, 13

summary.imptree, 6, 9, 12, 14