Package ‘LDATree’

January 20, 2025

Title Oblique Classification Trees with Uncorrelated Linear Discriminant Analysis Splits

Version 0.2.0

Description A classification tree method that uses Uncorrelated Linear Discriminant Analysis (ULDA) for variable selection, split determination, and model fitting in terminal nodes. It automatically handles missing values and offers visualization tools. For more details, see Wang (2024) <doi:10.48550/arXiv.2410.23147>.

License MIT + file LICENSE

URL `https://github.com/Moran79/LDATree`, `http://iamwangsiyu.com/LDATree/`

BugReports `https://github.com/Moran79/LDATree/issues`

Imports folda, ggplot2, grDevices, magrittr, stats, utils, visNetwork

Encoding UTF-8

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Siyu Wang [aut, cre, cph] (<`https://orcid.org/0009-0005-2098-7089`>)

Maintainer Siyu Wang <`iamwangsiyu@gmail.com`>

Repository CRAN

Date/Publication 2024-10-31 08:10:07 UTC

Contents

plot.Treee 2
predict.Treee 3
Treee 4

Index **7**

`plot.Treee` *Plot a Treee Object*

Description

This function visualizes either the entire decision tree or a specific node within the tree. The tree is displayed as an interactive network of nodes and edges, while individual nodes are scatter/density plots using `ggplot2`.

Usage

```
## S3 method for class 'Treee'
plot(x, datX, response, node = -1, ...)
```

Arguments

`x`	A fitted model object of class `Treee`, typically the result of the `Treee()` function.
`datX`	A data frame of predictor variables. Required for plotting individual nodes.
`response`	A vector of response values. Required for plotting individual nodes.
`node`	An integer specifying the node to plot. If `node = -1`, the entire tree is plotted. Default is `-1`.
`...`	Additional arguments passed to the plotting functions.

Value

A `visNetwork` interactive plot of the decision tree if `node = -1`, or a `ggplot2` object if a specific node is plotted.

Overall Tree Structure

A full tree diagram is displayed using visNetwork when `node` is not specified (the default is `-1`). The color represents the most common (plurality) class within each node, and the size of each terminal node reflects its relative sample size. Below each node, the fraction of correctly predicted training samples and the total sample size for that node are shown, along with the node index. Clicking on a node opens an information panel with additional details.

Individual Node Plot

To plot a specific node, you must provide the node index along with the original training predictors (`datX`) and responses (`response`). A scatter plot is generated if more than one discriminant score is available, otherwise, a density plot is created. Samples are projected onto their linear discriminant score(s).

Examples

```
fit <- Treee(datX = iris[, -5], response = iris[, 5], verbose = FALSE)
plot(fit) # plot the overall tree
plot(fit, datX = iris, response = iris[, 5], node = 1) # plot a specific node
```

---

predict.Treee   *Predictions From a Fitted Treee Object*

Description

Generate predictions on new data using a fitted `Treee` model.

Usage

```
## S3 method for class 'Treee'
predict(object, newdata, type = c("response", "prob", "all"), ...)
```

Arguments

`object`	A fitted model object of class `Treee`, typically the result of the `Treee()` function.
`newdata`	A data frame containing the predictor variables. Missing values are allowed and will be handled according to the fitted tree's method for handling missing data.
`type`	A character string specifying the type of prediction to return. Options are: • `'response'`: returns the predicted class for each observation (default). • `'prob'`: returns a data frame of posterior probabilities for each class. • `'all'`: returns a data frame containing predicted classes, posterior probabilities, and the predicted node indices.
`...`	Additional arguments passed to or from other methods.

Value

Depending on the value of `type`, the function returns:

- If `type = 'response'`: A character vector of predicted class labels.
- If `type = 'prob'`: A data frame of posterior probabilities, where each class has its own column.
- If `type = 'all'`: A data frame containing predicted class labels, posterior probabilities, and the predicted node indices.

Note: For factor predictors, if a level not present in the training data is found in `newdata`, it will be treated as missing and handled according to the `missingMethod` specified in the fitted tree.

Examples

```
fit <- Treee(datX = iris[, -5], response = iris[, 5], verbose = FALSE)
head(predict(fit, iris)) # Predicted classes
head(predict(fit, iris[, -5], type = "prob")) # Posterior probabilities
head(predict(fit, iris[, -5], type = "all")) # Full details
```

`Treee` *Classification Trees with Uncorrelated Linear Discriminant Analysis Terminal Nodes*

Description

This function fits a classification tree where each node has a Uncorrelated Linear Discriminant Analysis (ULDA) model. It can also handle missing values and perform downsampling. The resulting tree can be pruned either through pre-pruning or post-pruning methods.

Usage

```
Treee(
  datX,
  response,
  ldaType = c("forward", "all"),
  nodeModel = c("ULDA", "mode"),
  pruneMethod = c("pre", "post"),
  numberOfPruning = 10L,
  maxTreeLevel = 20L,
  minNodeSize = NULL,
  pThreshold = NULL,
  prior = NULL,
  misClassCost = NULL,
  missingMethod = c("medianFlag", "newLevel"),
  kSample = -1,
  verbose = TRUE
)
```

Arguments

`datX`	A data frame of predictor variables.
`response`	A vector of response values corresponding to `datX`.
`ldaType`	A character string specifying the type of LDA to use. Options are `"forward"` for forward ULDA or `"all"` for full ULDA. Default is `"forward"`.
`nodeModel`	A character string specifying the type of model used in each node. Options are `"ULDA"` for Uncorrelated LDA, or `"mode"` for predicting based on the most frequent class. Default is `"ULDA"`.
`pruneMethod`	A character string specifying the pruning method. `"pre"` performs pre-pruning based on p-value thresholds, and `"post"` performs cross-validation-based post-pruning. Default is `"pre"`.

`numberOfPruning`	An integer specifying the number of folds for cross-validation during post-pruning. Default is `10`.
`maxTreeLevel`	An integer controlling the maximum depth of the tree. Increasing this value allows for deeper trees with more nodes. Default is `20`.
`minNodeSize`	An integer controlling the minimum number of samples required in a node. Setting a higher value may lead to earlier stopping and smaller trees. If not specified, it defaults to one plus the number of response classes.
`pThreshold`	A numeric value used as a threshold for pre-pruning based on p-values. Lower values result in more conservative trees. If not specified, defaults to `0.01` for pre-pruning and `0.6` for post-pruning.
`prior`	A numeric vector of prior probabilities for each class. If `NULL`, the prior is automatically calculated from the data.
`misClassCost`	A square matrix C, where each element C_{ij} represents the cost of classifying an observation into class i given that it truly belongs to class j. If `NULL`, a default matrix with equal misclassification costs for all class pairs is used. Default is `NULL`.
`missingMethod`	A character string specifying how missing values should be handled. Options include `'mean'`, `'median'`, `'meanFlag'`, `'medianFlag'` for numerical variables, and `'mode'`, `'modeFlag'`, `'newLevel'` for factor variables. `'Flag'` options indicate whether a missing flag is added, while `'newLevel'` replaces missing values with a new factor level.
`kSample`	An integer specifying the number of samples to use for downsampling during tree construction. Set to `-1` to disable downsampling.
`verbose`	A logical value. If `TRUE`, progress messages and detailed output are printed during tree construction and pruning. Default is `FALSE`.

Value

An object of class `Treee` containing the fitted tree, which is a list of nodes, each an object of class `TreeeNode`. Each `TreeeNode` contains:

- `currentIndex`: The node index in the tree.
- `currentLevel`: The depth of the current node in the tree.
- `idxRow`, `idxCol`: Row and column indices indicating which part of the original data was used for this node.
- `currentLoss`: The training error for this node.
- `accuracy`: The training accuracy for this node.
- `stopInfo`: Information on why the node stopped growing.
- `proportions`: The observed frequency of each class in this node.
- `prior`: The (adjusted) class prior probabilities used for ULDA or mode prediction.
- `misClassCost`: The misclassification cost matrix used in this node.
- `parent`: The index of the parent node.
- `children`: A vector of indices of this node's direct children.

- `splitFun`: The splitting function used for this node.
- `nodeModel`: Indicates the model fitted at the node (`'ULDA'` or `'mode'`).
- `nodePredict`: The fitted model at the node, either a ULDA object or the plurality class.
- `alpha`: The p-value from a two-sample t-test used to evaluate the strength of the split.
- `childrenTerminal`: A vector of indices representing the terminal nodes that are descendants of this node.
- `childrenTerminalLoss`: The total training error accumulated from all nodes listed in `childrenTerminal`.

References

Wang, S. (2024). FoLDTree: A ULDA-Based Decision Tree Framework for Efficient Oblique Splits and Feature Selection. *arXiv preprint arXiv:2410.23147*. Available at `https://arxiv.org/abs/2410.23147`.

Wang, S. (2024). A New Forward Discriminant Analysis Framework Based On Pillai's Trace and ULDA. *arXiv preprint arXiv:2409.03136*. Available at `https://arxiv.org/abs/2409.03136`.

Examples

```
fit <- Treee(datX = iris[, -5], response = iris[, 5], verbose = FALSE)
# Use cross-validation to prune the tree
fitCV <- Treee(datX = iris[, -5], response = iris[, 5], pruneMethod = "post", verbose = FALSE)
head(predict(fit, iris)) # prediction
plot(fit) # plot the overall tree
plot(fit, datX = iris[, -5], response = iris[, 5], node = 1) # plot a certain node
```

Index

plot.Treee, 2
predict.Treee, 3

Treee, 4
Treee(), *2, 3*

visNetwork, *2*