Package 'PPtreeExt'

May 7, 2026

Type Package

Version 0.1.0

Title Projection Pursuit Classification Tree Extensions

Maintainer Natalia da Silva <natalia.dasilva@fcea.edu.uy>

Description Implements extensions to the projection pursuit tree algorithm for supervised classification, see Lee, Y. (2013), <doi:10.1214/13-EJS810> and Lee, E-K. (2018) <doi:10.18637/jss.v083.i08>. The algorithm is changed in two ways: improving prediction boundaries by modifying the choice of split points-through class subsetting; and increasing flexibility by allowing multiple splits per group.

License GPL (>= 2)

URL https://github.com/natydasilva/PPtreeExt

LazyData yes

Depends R (>= 4.3.0)

Imports Rcpp (>= 1.1.0), ggplot2, shiny, MASS, gridExtra, MixSim, PPtreeViz

Suggests knitr, randomForest, rpart, GGally, RColorBrewer, roxygen2 (>= 7.3.0), rmarkdown, rsample

Encoding UTF-8

LinkingTo Rcpp,RcppArmadillo

RoxygenNote 7.3.3

NeedsCompilation yes

Author Natalia da Silva [aut, cre] (ORCID: <https://orcid.org/0000-0002-6031-7451>), Dianne Cook [aut], Eun-Kyung Lee [aut]

Repository CRAN

Date/Publication 2026-02-06 14:20:18 UTC

Contents

crab . . . 2
explorapp . . . 3
findproj_Ext . . . 4
fishcatch . . . 5
glass . . . 6
image . . . 7
LDAopt_Ext . . . 8
leukemia . . . 9
lymphoma . . . 10
NCI60 . . . 10
olive . . . 11
parkinson . . . 12
PDAopt_Ext . . . 13
plot.PPtreeExtclass . . . 14
PPtreeExtclass . . . 16
PPtreeExt_split . . . 19
predict.PPtreeExtclass . . . 21
print.PPtreeExtclass . . . 22
TreeExt.construct . . . 23
wine . . . 24

Index **25**

`crab` *Australian crabs*

Description

Measurements on rock crabs of the genus Leptograpsus. The data set contains 200 observations from two species of crab (blue and orange), there are 50 specimens of each sex of each species, collected on site at Fremantle, Western Australia.

Type is the class variable and has 4 classes with the combinations of specie and sex (BlueMale, BlueFemale, OrangeMale and OrangeFemale).

FL the size of the frontal lobe length, in mm

RW rear width, in mm

CL length of midline of the carapace, in mm

CW maximum width of carapace, in mm

BD depth of the body; for females, measured after displacement of the abdomen, in mm

Usage

```
data(crab)
```

Format

A data frame with 200 rows and 6 variables

Source

Campbell, N. A. & Mahon, R. J. (1974), A Multivariate Study of Variation in Two Species of Rock Crab of genus Leptograpsus, Australian Journal of Zoology 22(3), 417 - 425.

`explorapp` *Shiny app to compare PPtree, PPtreeExt and rpart boundaries in 2D with different simulation scenarios*

Description

Shiny app to compare PPtree, PPtreeExt and rpart boundaries in 2D with different simulation scenarios

Usage

```
explorapp(ui, server)
```

Arguments

`ui`	user interface
`server`	server function

Value

No return value, called for side effects. Shinyapp is launched.

Examples

```
if(interactive()){
explorapp(ui,server)
}
```

findproj_Ext *Find Optimal Projection for Class Separation*

Description

Finds an optimal 1D projection of multivariate data that best separates classes using Linear Discriminant Analysis (LDA) or Penalized Discriminant Analysis (PDA), then determines a cutpoint for classification based on entropy splitting.

Usage

```
findproj_Ext(
  origclass,
  origdata,
  PPmethod = "LDA",
  q = 1,
  weight = TRUE,
  lambda = 0.1
)
```

Arguments

origclass	Factor or numeric vector containing the class labels for each observation.
origdata	Numeric matrix or data frame containing the predictor variables. Each row represents an observation and each column represents a variable.
PPmethod	Character string specifying the projection pursuit method. Either "LDA" (Linear Discriminant Analysis, default) or "PDA" (Penalized Discriminant Analysis).
q	Integer specifying the dimension of the projected data. Default is 1 for 1D projection.
weight	Logical indicating whether to use weighted LDA index calculation. Default is TRUE.
lambda	Numeric penalty parameter for the PDA method. Default is 0.1. Only used when PPmethod = "PDA".

Details

This function performs projection pursuit to find a one-dimensional projection that optimally separates classes in multivariate data. The process involves:

1. Finding the optimal projection direction using either LDA or PDA
2. Projecting all observations onto this direction
3. Determining an optimal cutpoint using entropy-based splitting
4. Creating binary classification indicators based on the cutpoint

The cutpoint is calculated to minimize the weighted entropy of the resulting split. In edge cases where the cutpoint equals the maximum projected value, the function uses the second-largest value to ensure a valid split.

Value

A list with the following components:

`Index`	Numeric value representing the optimization criterion achieved by the best projection. Higher values indicate better class separation.
`Alpha`	Numeric vector of length `ncol(origdata)` containing the optimal projection direction coefficients. This vector defines the linear combination of original variables that maximizes class separation.
`C`	Numeric scalar representing the optimal cutpoint (threshold) on the projected data. This value is determined using entropy-based splitting and divides observations into two groups for classification.
`IOindexL`	Logical vector of length `nrow(origdata)` indicating which observations have projected values less than or equal to the cutpoint C (`projdata <= C`). These observations are assigned to the left node/class.
`IOindexR`	Logical vector of length `nrow(origdata)` indicating which observations have projected values greater than the cutpoint C (`projdata > C`). These observations are assigned to the right node/class.

Note

The vectors `IOindexL` and `IOindexR` are complementary (mutually exclusive and exhaustive), meaning every observation is assigned to exactly one group.

References

Lee, YD, Cook, D., Park JW, and Lee, EK (2013) PPtree: Projection Pursuit Classification Tree, Electronic Journal of Statistics, 7:1369-1386.

`fishcatch` *Fish catch data set*

Description

There are 159 fishes of 7 species are caught and measured. Altogether there are 7 variables. All the fishes are caught from the same lake(Laengelmavesi) near Tampere in Finland.

Type has 7 fish classes, with 35 cases of Bream, 11 cases of Parkki, 56 cases of Perch 17 cases of Pike, 20 cases of Roach, 14 cases of Smelt and 6 cases of Whitewish.

weight Weight of the fish (in grams)

length1 Length from the nose to the beginning of the tail (in cm)

length2 Length from the nose to the notch of the tail (in cm)

length3 Length from the nose to the end of the tail (in cm)

height Maximal height as % of Length3

width Maximal width as % of Length3

Usage

```
data(fishcatch)
```

Format

A data frame with 159 rows and 7 variables

Source

[http://www.amstat.org/publications/jse/jse_data_archive.htm](fishcatch)

`glass` *Glass data set*

Description

Contains measurements 214 observations of 6 types of glass; defined in terms of their oxide content.

Type has 6 types of glasses

X1 refractive index

X2 Sodium (unit measurement: weight percent in corresponding oxide).

X3 Magnesium

X4 Aluminum

X5 Silicon

X6 Potassium

X7 Calcium

X8 Barium

X9 Iron

Usage

```
data(glass)
```

Format

A data frame with 214 rows and 10 variables

`image` *The image data set*

Description

contains 2310 observations of instances from 7 outdoor images

Type has 7 types of outdoor images, brickface, cement, foliage, grass, path, sky, and window.

X1 the column of the center pixel of the region

X2 the row of the center pixel of the region.

X3 the number of pixels in a region = 9.

X4 the results of a line extraction algorithm that counts how many lines of length 5 (any orientation) with low contrast, less than or equal to 5, go through the region.

X5 measure the contrast of horizontally adjacent pixels in the region. There are 6, the mean and standard deviation are given. This attribute is used as a vertical edge detector.

X6 X5 sd

X7 measures the contrast of vertically adjacent pixels. Used for horizontal line detection.

X8 sd X7

X9 the average over the region of (R + G + B)/3

X10 the average over the region of the R value.

X11 the average over the region of the B value.

X12 the average over the region of the G value.

X13 measure the excess red: (2R - (G + B))

X14 measure the excess blue: (2B - (G + R))

X15 measure the excess green: (2G - (R + B))

X16 3-d nonlinear transformation of RGB. (Algorithm can be found in Foley and VanDam, Fundamentals of Interactive Computer Graphics)

X17 mean of X16

X18 hue mean

Usage

```
data(image)
```

Format

A data frame contains 2310 observations and 19 variables

LDAopt_Ext *PP optimization using LDA index*

Description

Projection Pursuit Optimization Using LDA Index

Usage

```
LDAopt_Ext(origclass, origdata, q = 1, weight = TRUE, ...)
```

Arguments

origclass	Factor or numeric vector containing the class labels for each observation.
origdata	Numeric matrix or data frame containing the predictor variables without class information. Each row represents an observation and each column represents a variable.
q	Integer specifying the dimension of the projection space. Default is 1 for 1-dimensional projection.
weight	Logical indicating whether to use weighted LDA index calculation. Default is TRUE.
...	Additional arguments to be passed to internal optimization methods.

Details

Finds the q-dimensional optimal projection using the Linear Discriminant Analysis (LDA) projection pursuit index. This implementation follows the method described in PPtree.

The LDA projection pursuit index measures class separation by maximizing the ratio of between-class variance to within-class variance in the projected space. This function:

1. Calls LDAopt to find the optimal q-dimensional projection directions
2. Evaluates the LDA index for the optimal projection using LDAindex2
3. Returns both the projection matrix and its associated index value

When weight = TRUE, the index calculation accounts for class proportions, giving appropriate weight to each class in the optimization.

Value

An object of class "PPoptim", which is a list containing:

indexbest	Numeric value representing the maximum LDA index achieved by the optimal projection. Higher values indicate better class separation.
projbest	Numeric matrix of optimal projection coefficients with dimensions ncol(origdata) by q. Each column represents an optimal projection direction that maximizes the LDA index for class separation.
origclass	The original class information vector passed as input, preserved for reference.
origdata	The original data matrix without class information, preserved for reference.

References

Lee, EK., Cook, D., Klinke, S., and Lumley, T. (2005) Projection Pursuit for Exploratory Supervised Classification, Journal of Computational and Graphical Statistics, 14(4):831-846.

See Also

PDAopt_Ext, findproj_Ext

`leukemia` *Leukemia data set*

Description

Leukemia data set

Usage

```
data(leukemia)
```

Format

This dataset comes from a study of gene expression in two types of acute leukemias, acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML). Gene expression levels were measured using Affymetrix high density oligonucleotide arrays containing 6817 human genes. A data set containing 72 observations from 3 leukemia types classes.

Type has 3 classes with 38 cases of B-cell ALL, 25 cases of AML and 9 cases of T-cell ALL.

Gene1, Gen2, ..., Gen40 gene expression levels.

A data frame with 72 rows and 41 variables

Source

Dudoit, S., Fridlyand, J. and Speed, T. P. (2002). Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. Journal of the American statistical Association 97 77-87.

lymphoma *Lymphoma data set*

Description

Gene expression in the three most prevalent adult lymphoid malignancies: B-cell chronic lymphocytic leukemia (B-CLL), follicular lymphoma (FL), and diffuse large B-cell lym- phoma (DLBCL). Gene expression levels were measured using a specialized cDNA microarray, the Lymphochip, containing genes that are preferentially expressed in lymphoid cells or that are of known immunologic or oncologic importance. This data set contain 80 observations from 3 lymphoma types.

Type Class variable has 3 classes with 29 cases of B-cell ALL (B-CLL), 42 cases of diffuse large B-cell lymphoma (DLBCL) and 9 cases of follicular lymphoma (FL).

Gene1 to Gen 50 gene expression

Usage

```
data(lymphoma)
```

Format

A data frame with 80 rows and 51 variables

Source

Dudoit, S., Fridlyand, J. and Speed, T. P. (2002). Comparison of Discrimination Methods for the Classification of Tumors Using Gene Ex- pression Data. Journal of the American statistical Association 97 77-87.

NCI60 *NCI60 data set*

Description

cDNA microarrays were used to examine the variation in gene expression among the 60 cell lines. The cell lines are derived from tumors with different sites of origin. This data set contain 61 observations and 30 feature variables from 8 different tissue types.

Type has 8 different tissue types, 9 cases of breast, 5 cases of central nervous system (CNS), 7 cases pf colon, 8 cases of leukemia, 8 cases of melanoma, 9 cases of non-small-cell lung carcinoma (NSCLC), 6 cases of ovarian and 9 cases of renal.

Gene1 to Gen 30 gene expression information

Usage

```
data(NCI60)
```

Format

A data frame with 61 rows and 31 variables

Source

Dudoit, S., Fridlyand, J. and Speed, T. P. (2002). Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data. Journal of the American statistical Association 97 77-87.

`olive` *The olive data set*

Description

contains 572 observations and 10 variables

Region Three super-classes of Italy: North, South and the island of Sardinia

area Nine collection areas: three from North, four from South and 2 from Sardinia

palmitic fatty acids percent x 100

palmitoleic fatty acids percent x 100

stearic fatty acids percent x 100

oleic fatty acids percent x 100

linoleic fatty acids percent x 100

linolenic fatty acids percent x 100

arachidic fatty acids percent x 100

eicosenoic fatty acids percent x 100

Usage

```
data(olive)
```

Format

A data frame contains 573 observations and 10 variables

`parkinson` *Parkinson data set*

Description

A data set containing 195 observations from 2 parkinson types.

Type Class variable has 2 classes, there are 48 cases of healthy people and 147 cases with Parkinson. The feature variables are biomedical voice measures.

X1 Average vocal fundamental frequency

X2 Maximum vocal fundamental frequency

X3 Minimum vocal fundamental frequency

X4 MDVP:Jitter(%) measures of variation in fundamental frequency

X5 MDVP:Jitter(Abs) measures of variation in fundamental frequency

X6 MDVP:RAP measures of variation in fundamental frequency

X7 MDVP:PPQ measures of variation in fundamental frequency

X8 Jitter:DDP measures of variation in fundamental frequency

X9 MDVP:Shimmer measures of variation in amplitude

X10 MDVP:Shimmer(dB) measures of variation in amplitude

X11 Shimmer:APQ3 measures of variation in amplitude

X12 Shimmer:APQ5 measures of variation in amplitude

X13 MDVP:APQ measures of variation in amplitude

X14 Shimmer:DDA measures of variation in amplitude

X15 NHR measures of ratio of noise to tonal components in the voice

X16 HNR measures of ratio of noise to tonal components in the voice

X17 RPDE nonlinear dynamical complexity measures

X18 D2 nonlinear dynamical complexity measures

X19 DFA - Signal fractal scaling exponent

X20 spread1 Nonlinear measures of fundamental frequency variation

X21 spread2 Nonlinear measures of fundamental frequency variation

X22 PPE Nonlinear measures of fundamental frequency variation

Usage

```
data(parkinson)
```

Format

A data frame with 195 rows and 23 variables

Source

[https://archive.ics.uci.edu/ml/datasets/Parkinsons](Parkinson)

PDAopt_Ext *PP optimization using PDA index*

Description

Projection Pursuit Optimization Using PDA Index

Usage

```
PDAopt_Ext(origclass, origdata, q = 1, weight = TRUE, lambda = 0.1, ...)
```

Arguments

origclass	Factor or numeric vector containing the class labels for each observation.
origdata	Numeric matrix or data frame containing the predictor variables without class information. Each row represents an observation and each column represents a variable.
q	Integer specifying the dimension of the projection space. Default is 1 for 1-dimensional projection.
weight	Logical indicating whether to use weighted PDA index calculation. Default is TRUE.
lambda	Numeric penalty parameter for the PDA index. Controls the amount of regularization applied. Default is 0.1. Higher values increase regularization, which is useful for high-dimensional or collinear data.
...	Additional arguments to be passed to internal optimization methods.

Details

Finds the q-dimensional optimal projection using the Penalized Discriminant Analysis (PDA) projection pursuit index. This implementation follows the method described in PPtree and is particularly useful for high-dimensional data (large p, small n).

The Penalized Discriminant Analysis (PDA) projection pursuit index extends LDA by incorporating a penalty term, making it particularly suitable for:

- High-dimensional data where the number of variables exceeds the number of observations (p > n)
- Data with multicollinearity among predictor variables
- Cases where standard LDA fails due to singular covariance matrices

The function performs the following steps:

1. Calls PDAopt to find the optimal q-dimensional projection directions with regularization
2. Evaluates the PDA index for the optimal projection using PDAindex2
3. Returns both the projection matrix and its associated index value

The lambda parameter controls the trade-off between maximizing class separation and regularization. When weight = TRUE, the index calculation accounts for class proportions in the optimization.

Value

An object of class "PPoptim", which is a list containing:

indexbest	Numeric value representing the maximum PDA index achieved by the optimal projection. Higher values indicate better class separation with appropriate regularization.
projbest	Numeric matrix of optimal projection coefficients with dimensions ncol(origdata) by q. Each column represents an optimal projection direction that maximizes the PDA index for class separation.
origclass	The original class information vector passed as input, preserved for reference.
origdata	The original data matrix without class information, preserved for reference.

References

Lee, EK, Cook, D. (2010) A Projection Pursuit Index for Large p Small n Data, Statistics and Computing, 20:381-392.

See Also

LDAopt_Ext, findproj_Ext

---

plot.PPtreeExtclass *Plot Projection Pursuit Classification Tree*

Description

Visualizes a Projection Pursuit (PP) classification tree using grid graphics. The function creates a hierarchical tree diagram showing the structure of splits and terminal nodes with class assignments. Supports automatic scaling for large trees.

Usage

```
## S3 method for class 'PPtreeExtclass'
plot(
  x,
  font.size = 17,
  width.size = 1,
  main = "Projection Pursuit Classification Tree",
  sub = NULL,
  auto.scale = TRUE,
  min.width = NULL,
  min.height = NULL,
  ...
)
```

Arguments

x	An object of class PPtreeclass or PPtreeExtclass containing the tree structure, projection vectors, and split points.
font.size	Numeric. Font size for text labels in the plot. Default is 17. Will be automatically reduced for large trees when auto.scale = TRUE.
width.size	Numeric. Width scaling factor for graphical elements (nodes, edges). Default is 1. Will be automatically adjusted for large trees when auto.scale = TRUE.
main	Character string. Main title for the plot. Default is "Projection Pursuit Classification Tree".
sub	Character string or NULL. Subtitle for the plot. Default is NULL (no subtitle).
auto.scale	Logical. If TRUE (default), automatically adjusts plot dimensions and font size based on tree size. Recommended for trees with more than 20 terminal nodes or depth greater than 15.
min.width	Numeric or NULL. Minimum width (number of terminal nodes) for the plot when auto.scale = FALSE. If NULL, uses the number of classes in the data. Default is NULL.
min.height	Numeric or NULL. Minimum height (tree depth) for the plot when auto.scale = FALSE. If NULL, uses calculated tree depth. Default is NULL.
...	Additional arguments (currently not used).

Details

The plot displays:

- **Internal nodes**: Shown as ellipses with the projection used for splitting (e.g., "proj1 * X")
- **Terminal nodes**: Shown as gray rectangles with the assigned class label
- **Edges**: Labeled with split rules ("< cutN" for left child, ">= cutN" for right child)
- **Node IDs**: Small boxes at the top of each node

Auto-scaling behavior: When auto.scale = TRUE and the tree has more than 20 terminal nodes or depth greater than 15:

- Font size is reduced: max(10, 17 - floor((n_terminal - 20) / 5))
- Width size is reduced: max(0.7, 1 - (n_terminal - 20) * 0.01)
- Width is set to: max(n_terminal, n_classes)
- Height is set to: tree depth

Manual scaling: When auto.scale = FALSE, you can control dimensions with min.width and min.height.

Value

Invisibly returns a list with:

width	Numeric. The width used for plotting (number of terminal nodes)
height	Numeric. The height used for plotting (tree depth)
font.size	Numeric. The final font size used (after auto-scaling)

Note

This function requires the `grid` package. It will create a new graphics page using `grid.newpage()`.

For very large trees (>50 terminal nodes), consider:

- Exporting to a large PNG or PDF file
- Manually reducing `font.size` further
- Pruning the tree before plotting

Examples

```
library(grid)
# Example with penguins dataset
data(penguins)
penguins <- na.omit(penguins[, -c(2,7)])
penguins_ppt <- PPtreeExtclass(species~bill_len + bill_dep +flipper_len +
 body_mass,  data = penguins, PPmethod = "PDA", srule = FALSE )

plot(penguins_ppt,
     main = "Penguins Classification with PPtreeExt",
      font.size = 8,
width.size = 0.7)
```

PPtreeExtclass *Projection pursuit classification tree*

Description

Projection Pursuit Classification Tree with Extensions

Usage

```
PPtreeExtclass(formula, data, PPmethod = "LDA", weight = TRUE,
                   lambda = 0.1,srule, tot = nrow(data), tol = 0.5,...)
```

Arguments

`formula`	An object of class `"formula"` of the form `class ~ x1 + x2 + ...` where `class` is the factor variable containing class labels and `x1, x2, ...` are the predictor variables. Interaction terms (using `*`) are not supported.
`data`	Data frame containing both the class variable and predictor variables specified in the formula.
`PPmethod`	Character string specifying the projection pursuit index to use. Either `"LDA"` (Linear Discriminant Analysis, default) or `"PDA"` (Penalized Discriminant Analysis).

`weight`	Logical indicating whether to use weighted index calculation in LDA and PDA. When `TRUE` (default), class proportions are accounted for in the optimization.
`lambda`	Numeric penalty parameter for the PDA index, ranging from 0 to 1. Default is 0.1. Only used when `PPmethod = "PDA"`.
`srule`	Logical flag for stopping rule. If `TRUE` (default), uses entropy-based and size-based stopping criteria. If `FALSE`, stops only when nodes are pure (single class) or empty.
`tot`	Integer specifying the total number of observations in the original dataset. Default is `nrow(data)`. Used in conjunction with stopping rules to determine minimum node sizes.
`tol`	Numeric tolerance value for the entropy-based stopping rule. Nodes with entropy below this threshold will not be split further. Default is 0.5. Lower values create deeper trees.
`...`	Additional arguments to be passed to internal tree construction methods.

Details

Constructs a projection pursuit classification tree using various projection pursuit indices (LDA or PDA) at each split. This extended version includes customizable stopping rules based on entropy and node size criteria.

This function builds a binary classification tree where each split is determined by finding an optimal projection of the data onto a one-dimensional space using either LDA or PDA indices. The algorithm works as follows:

Tree Construction Process:

1. At each node, find the optimal 1D projection that best separates classes
2. Project the data onto this direction and find an optimal cutpoint
3. Split observations based on the cutpoint into left and right child nodes
4. Recursively repeat until stopping criteria are met

Stopping Rules: When `srule = TRUE`, a node stops splitting if any of the following conditions hold:

- The node is pure (contains only one class)
- The node contains fewer than 5% of the total observations (`n/tot <= 0.05`)
- The node entropy is below the tolerance threshold (`entropy < tol`)

When `srule = FALSE`, splitting only stops for pure or empty nodes, potentially creating deeper, more complex trees.

Projection Methods:

- **LDA**: Suitable for most classification problems with moderate dimensionality
- **PDA**: Recommended for high-dimensional data (p > n) or data with multicollinearity

The `tol` parameter controls tree complexity: smaller values allow more splits (deeper trees with potentially better training accuracy but higher risk of overfitting), while larger values create simpler trees (better generalization but potentially underfitting).

Value

An object of class `c("PPtreeExtclass", "PPtreeclass")`, which is a list containing:

`Tree.Struct`	A matrix defining the tree structure. Each row represents a node with 5 columns: node ID, left child node ID, right/final node ID (class label if terminal node), coefficient ID (projection index), and optimization index value.
`projbest.node`	A matrix where each row contains the optimal 1-dimensional projection coefficients for each split node. Each row has length equal to `ncol(origdata)`, defining the projection direction used at that node.
`splitCutoff.node`	A numeric vector or matrix containing the cutoff values (thresholds) used at each split node for classification decisions.
`origclass`	Factor vector of the original class labels from the input data.
`origdata`	Matrix of the original predictor variables (without the class variable).
`terms`	The terms object from the model frame, preserving the formula structure.

Note

This function does not support interaction terms in the formula. Use only additive terms (e.g., `y ~ x1 + x2`) and not multiplicative terms (e.g., `y ~ x1 * x2`).

References

Lee, YD, Cook, D., Park JW, and Lee, EK (2013) PPtree: Projection Pursuit Classification Tree, Electronic Journal of Statistics, 7:1369-1386.

See Also

`TreeExt.construct`, `PPtreeExt_split`, `findproj_Ext`, `predict.PPtreeExtclass`

Examples

```
set.seed(234)
data(penguins)
penguins <- na.omit(penguins[, -c(2,7, 8)])
require(rsample)
penguins_spl <- rsample::initial_split(penguins, strata=species)
penguins_train <- training(penguins_spl)
penguins_test <- testing(penguins_spl)
penguins_ppt <- PPtreeExtclass(species~bill_len + bill_dep +
flipper_len + body_mass, data = penguins_train, PPmethod = "LDA", tot=nrow
(penguins_train), tol =  0.2 , srule = TRUE)
```

PPtreeExt_split *Projection Pursuit Classification Tree with Random Variable Selection*

Description

Constructs a projection pursuit classification tree using various projection pursuit indices. Optionally performs random variable selection at each split which can be used to include in a random forests methodology. When `size.p = 1`, this reduces to a PPtree algorithm.

Usage

```
PPtreeExt_split(
  formula,
  data,
  PPmethod = "LDA",
  size.p = 1,
  lambda = 0.1,
  entro = FALSE,
  entroindiv = FALSE,
  ...
)
```

Arguments

`formula`	A formula of the form `class ~ x1 + x2 + ...` where `class` is the factor variable containing class labels and `x1, x2, ...` are the predictor variables.
`data`	Data frame containing both the class variable and predictor variables.
`PPmethod`	Character string specifying the projection pursuit index to use. Either `"LDA"` (Linear Discriminant Analysis, default) or `"PDA"` (Penalized Discriminant Analysis).
`size.p`	Numeric value between 0 and 1 specifying the proportion of variables to randomly sample at each split. Default is 1, which uses all variables at each split (standard PPtree). Values less than 1 introduce randomness similar to random forests, which can improve robustness and reduce overfitting.
`lambda`	Numeric penalty parameter for the PDA index, ranging from 0 to 1. When `lambda = 0`, no penalty is applied and PDA equals LDA. When `lambda = 1`, all variables are treated as uncorrelated. Default is 0.1. Only used when `PPmethod = "PDA"`.
`entro`	Logical indicating whether to use entropy-based stopping rules for tree construction. Default is `FALSE`.
`entroindiv`	Logical indicating whether to compute entropy for each individual observation in the 1D projection. Default is `FALSE`.
`...`	Additional arguments to be passed to internal tree construction methods.

Details

This function extends the standard PPtree algorithm by incorporating random variable selection at each split, and define the split based on subsetting groups. The algorithm:

1. At each node, randomly samples `size.p * 100%` of the predictor variables
2. Finds the optimal projection using the selected variables and specified index (LDA or PDA)
3. Determines a cutpoint based on entropy splitting if entropy parameters are set
4. Recursively splits the data until stopping criteria are met

The `entro` parameter enables entropy-based stopping rules that halt splitting when nodes become sufficiently pure or small. The `entroindiv` parameter computes entropy at the individual observation level in the projected space, which can provide more refined splitting decisions.

When `size.p = 1`, all variables are used at each split and the function behaves as a standard PPtree. Values of `size.p < 1` introduce randomness that can improve model robustness, especially for high-dimensional data or when building ensemble models.

Value

An object of class `"PPtreeclass"`, which is a list containing:

`Tree.Struct` A matrix defining the tree structure of the projection pursuit classification tree. Each row represents a node with columns: node ID, left child node ID, right child node ID (or final class if terminal), coefficient ID, and index value.

`projbest.node` A matrix where each row contains the optimal 1-dimensional projection coefficients for each split node. The number of columns equals the number of predictor variables.

`splitCutoff.node`
A data frame containing the cutoff values and splitting rules for each split node. Contains 8 rule columns defining the classification boundaries.

`origclass` Factor vector of the original class labels from the input data.

`origdata` Matrix of the original predictor variables (without the class variable).

References

Lee, YD, Cook, D., Park JW, and Lee, EK (2013) PPtree: Projection pursuit classification tree, Electronic Journal of Statistics, 7:1369-1386.

See Also

`TreeExt.construct`, `findproj_Ext`, `LDAopt_Ext`, `PDAopt_Ext`

Examples

```
data(penguins)
penguins <- na.omit(penguins[, -c(2,7, 8)])
require(rsample)
penguins_spl <- rsample::initial_split(penguins, strata=species)
penguins_train <- training(penguins_spl)
```

```
penguins_test <- testing(penguins_spl)
penguins_ppt2 <- PPtreeExt_split(species~bill_len + bill_dep +
flipper_len + body_mass, data = penguins_train, PPmethod = "LDA", tot=nrow
(penguins_train), tol =  0.5 , entro=TRUE)
```

`predict.PPtreeExtclass` *Predict Method for Projection Pursuit Classification Tree Extensions*

Description

Predicts class labels for new observations using a fitted projection pursuit classification tree and optionally calculates prediction error when true class labels are provided.

Usage

```
## S3 method for class 'PPtreeExtclass'
predict(object, newdata, true.class = NULL, ...)
```

Arguments

`object`	An object of class `"PPtreeExtclass"` from `PPtreeExtclass` or `PPtreeExt_split`.
`newdata`	A data frame or matrix containing the predictor variables for which predictions are to be made. Must contain the same variables (in the same order) as used in the training data, but without the class variable.
`true.class`	Optional vector of true class labels for the test data. If provided, prediction error will be calculated. Can be either numeric or factor. Default is `NULL`.
`...`	Additional arguments (currently not used).

Value

A list with two components:

`predict.class`	A character vector of predicted class labels for each observation in `newdata`.
`predict.error`	Integer count of prediction errors (misclassifications). Only computed when `true.class` is provided; otherwise returns `NA`.

Examples

```
data(penguins)
penguins <- na.omit(penguins[, -c(2,7, 8)])
require(rsample)
penguins_spl <- rsample::initial_split(penguins, strata=species)
penguins_train <- training(penguins_spl)
penguins_test <- testing(penguins_spl)
penguins_ppt <- PPtreeExtclass(species~bill_len + bill_dep +
flipper_len + body_mass, data = penguins_train, PPmethod = "LDA", tot =nrow
(penguins_train), tol=0.5)
predict(object = penguins_ppt, newdata = penguins_test[,-1], true.class = penguins_test$species)
```

`print.PPtreeExtclass` *Print Method for PPtreeExtclass Objects*

Description

Prints a summary of a fitted projection pursuit classification tree, including the tree structure, optionally the projection coefficients and cutoff values, and the training error rate.

Usage

```
## S3 method for class 'PPtreeExtclass'
print(x, coef.print = FALSE, cutoff.print = FALSE, verbose = TRUE, ...)
```

Arguments

`x`	An object of class `"PPtreeExtclass"` from `PPtreeExtclass` or `PPtreeExt_split`.
`coef.print`	Logical indicating whether to print the projection coefficients for each split node. Default is `FALSE`.
`cutoff.print`	Logical indicating whether to print the cutoff values for each split node. Default is `FALSE`.
`verbose`	Logical indicating whether to print the tree structure and error rate. If `FALSE`, the function returns the tree structure invisibly without printing. Default is `TRUE`.
`...`	Additional arguments (currently not used).

Details

The function traverses the tree structure stored in `x$Tree.Struct` and creates a hierarchical text representation. When `coef.print = TRUE`, the projection coefficients (linear combinations of features) used at each split are displayed. When `cutoff.print = TRUE`, the threshold values used to determine left/right splits are shown.

The training error rate is computed by applying the fitted tree to the original training data.

Value

The object x, invisibly

See Also

`PPtreeExtclass`, `PPtreeExt_split`, `predict.PPtreeExtclass`

`TreeExt.construct` *Projection pursuit classification tree extensions*

Description

Construct the projection pursuit classification tree extensions

Usage

```
TreeExt.construct(origclass, origdata, Tree.Struct, id, rep, rep1, rep2,
projbest.node, splitCutoff.node, PPmethod,
lambda = NULL, q = 1, weight = TRUE, srule=TRUE, tot=NULL, tol = .5,...)
```

Arguments

`origclass`	factor or numeric vector containing the class labels for each observation.
`origdata`	data frame with the original data without class variable
`Tree.Struct`	tree structure of projection pursuit classification tree
`id`	tree node id
`rep`	internal counter for nodes
`rep1`	internal counter for nodes
`rep2`	internal counter for nodes
`projbest.node`	bests projection node
`splitCutoff.node`	cutof node
`PPmethod`	method for projection pursuit; "LDA", "PDA"
`lambda`	lambda in PDA index
`q`	numeric value with dimension of the projected data, if it is 1 then 1D projection is used
`weight`	weight flag in LDA, PDA
`srule`	stopping rule flag; if TRUE use stopping rule, if FALSE stop only for pure or empty nodes
`tot`	total number of observations
`tol`	tolerance value for entropy stopping rule for splitting a node
`...`	additional arguments to pass trough

Details

Find tree structure using various projection pursuit indices of classification in each split.

This function recursively constructs a binary classification tree using projection pursuit. At each node, it finds the optimal projection direction that best separates classes, determines a cutpoint, and creates child nodes until stopping criteria are met (pure nodes, small node size, or low entropy).

Value

A list containing the complete tree structure and node information:

`Tree.Struct` A matrix where each row represents a node in the projection pursuit classification tree. The matrix has 5 columns:

- Column 1: Node ID
- Column 2: ID of the left child node (or 0 if terminal node)
- Column 3: ID of the right child node, or the predicted class label if terminal node
- Column 4: Projection index (which projection vector is used at this node)
- Column 5: Optimization criterion value for the projection at this node

`projbest.node` A matrix where each row contains the optimal projection coefficients (Alpha vector) for each split node.

`splitCutoff.node` A matrix/vector containing the optimal cutpoint values used at each split node.

`rep` Integer counter tracking the current node being processed (internal use).

`rep1` Integer counter for assigning child node IDs (internal use).

`rep2` Integer counter for tracking projection indices (internal use).

`wine` *Wine data set*

Description

A data set containing 178 observations from 3 wine grown cultivares in Italy.

Type Class variable has 3 classes that are 3 different wine grown cultivares in Italy.

X1 to X13 Check vbles

Usage

```
data(wine)
```

Format

A data frame with 178 rows and 14 variables

Index

∗ **datasets**
  crab, 2
  fishcatch, 5
  glass, 6
  image, 7
  leukemia, 9
  lymphoma, 10
  NCI60, 10
  olive, 11
  parkinson, 12
  wine, 24
∗ **projection**
  PDAopt_Ext, 13
∗ **pursuit**
  PDAopt_Ext, 13
∗ **tree**
  PPtreeExt_split, 19
  PPtreeExtclass, 16
  predict.PPtreeExtclass, 21

crab, 2

explorapp, 3

findproj_Ext, 4, *9*, *14*, *18*, *20*
fishcatch, 5

glass, 6

image, 7

LDAopt_Ext, 8, *14*, *20*
leukemia, 9
lymphoma, 10

NCI60, 10

olive, 11

parkinson, 12
PDAopt_Ext, *9*, 13, *20*
plot.PPtreeExtclass, 14
PPtreeExt_split, *18*, 19, *21, 22*
PPtreeExtclass, 16, *21, 22*
predict.PPtreeExtclass, *18*, 21, *22*
print.PPtreeExtclass, 22

TreeExt.construct, *18*, *20*, 23

wine, 24