Package: treeDA (via r-universe)

September 3, 2024

Title Tree-Based Discriminant Analysis

Version 0.0.5

Description Performs sparse discriminant analysis on a combination of node and leaf predictors when the predictor variables are structured according to a tree, as described in Fukuyama et al. (2017) <doi:10.1371/journal.pcbi.1005706>.

Depends R (>= 3.4.0)

Imports sparseLDA (>= 0.1.9), Matrix (>= 1.2.10), mvtnorm (>= 1.0.6), reshape2 (>= 1.4.2), gtable (>= 0.2.0), phyloseq (>= 1.22.3), ggplot2 (>= 2.2.1), ape (>= 5.1), grid, stats

Suggests adaptiveGPCA (>= 0.1), knitr (>= 1.16), testthat (>= 2.0.0), markdown, rmarkdown

VignetteBuilder knitr

License GPL-2

URL https://github.com/jfukuyama/treeda

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Repository https://jfukuyama.r-universe.dev

RemoteUrl https://github.com/jfukuyama/treeda

RemoteRef HEAD

RemoteSha 86ae334c34a5ee5f3150ce8a4abc21d216032e70

Contents

`treeDA-package` *Tree-based discriminant analysis*

Description

A package for performing sparse, tree-based discriminant analysis.

Details

This package contains functions for building sparse, tree-structured models for classification. The method is based on the idea that when our predictors are structured according to a tree, we can create an expanded feature space containing both the original leaf predictors as well as node predictors, which correspond to sums or averages across the leaves descending from them. Without some sort of regularization this problem would be unidentifiable, but with the regularization provided by sparse discriminant analysis we get stable solutions.

The package fits a sparse discriminant model in the expanded feature space and translates the results back to the leaf space, so that the interpretation can be purely in terms of the original predictors. The package also includes functions to perform cross validation to pick the sparsity level and plotting commands to visualize the tree and the fitted coefficient vectors.

The main function in this package is `treeda`, which fits a sparse tree-based discriminant model. Additional functions provided are `treedacv`, which performs cross-validation to determine the correct sparsity level, and functions to plot the resulting coefficient vectors along the tree (`plot_coefficients`).

Author(s)

Maintainer: Julia Fukuyama <julia.fukuyama@gmail.com>

See Also

Useful links:

- https://github.com/jfukuyama/treeda

coef.treeda *Coefficients from treeda fit*

Description

Returns the coefficients from a treeda fit either in terms of the leaves only or in terms of the nodes and leaves.

Usage

```
## S3 method for class 'treeda'
coef(object, type = c("leaves", "nodes"), ...)
```

Arguments

object	An object of class treeda.
type	Should the coefficients be in the leaf space or the node space?
...	Not used.

Value

A Matrix object containing the coefficients.

Examples

```
data(treeda_example)
out.treeda = treeda(response = treeda_example$response,
    predictors = treeda_example$predictors,
    tree = treeda_example$tree,
    p = 1)
coef(out.treeda, type = "leaves")
coef(out.treeda, type = "nodes")
```

combine_plot_and_tree *Method for combining two ggplots*

Description

This method takes a ggplot of some data along the tips of the tree and a ggplot of a tree and combines them. It assumes that you are putting the tree on top and that the x axis for the plot has the leaves in the correct position (this can be found using the function get_leaf_position).

Usage

```
combine_plot_and_tree(plot, tree.plot, tree.height = 5, print = TRUE)
```

Arguments

plot	A plot of data about the leaves with the x axis corresponding to leaves.
tree.plot	A plot of the tree.
tree.height	The relative amount of space in the plot the tree should take up.
print	If true, the function will print the combined plot to a graphics device, otherwise it will just return the gtable object without printing.

Value

Returns a gtable object.

---

get_leaf_position *Get leaf positions from a tree layout*

Description

Takes a tree, returns a vector with names describing the leaves and entries giving the position of that leaf in the tree layout.

Usage

```
get_leaf_position(tree, ladderize)
```

Arguments

tree	A tree of class phylo.
ladderize	FALSE for a non-ladderzied layout, TRUE or "right" for a ladderized layout, "left" for a layout ladderized the other way.

---

makeNodeAndLeafPredictors *Make a matrix with predictors for each leaf and node*

Description

Make a matrix with one predictor for each leaf and node in the tree, where the node predictors are the sum of the leaf predictors descending from them.

Usage

```
makeNodeAndLeafPredictors(leafPredictors, tree)
```

Arguments

leafPredictors	A predictor matrix for the leaves: rows are samples, columns are leaves.
tree	A phylogenetic tree describing the relationships between the species/leaves.

Value

A predictor matrix for leaves and nodes together: rows are samples, columns are leaf/node predictors.

---

nodeToLeafCoefficients

Node coefficients to leaf coefficients

Description

General-purpose function for going from a coefficient vector on the nodes to a coefficient vector on the leaves.

Usage

```
nodeToLeafCoefficients(coef.vec, tree)
```

Arguments

coef.vec	A vector containing coefficients on internal nodes plus leaves.
tree	The phylogenetic tree.

Value

A vector containing coefficients on the leaves.

---

plot.treedacv

Plot a treedacv object

Description

Plots the cross-validation error with standard error bars.

Usage

```
## S3 method for class 'treedacv'
plot(x, ...)
```

Arguments

x	An object of class treedacv.
...	Not used.

Examples

```
data(treeda_example)
out.treedacv = treedacv(response = treeda_example$response,
    predictors = treeda_example$predictors,
    tree = treeda_example$tree,
    pvec = 1:10)
plot(out.treedacv)
```

---

plot_coefficients *Plot the discriminating axes from treeda*

Description

Plots the leaf coefficients for the discriminating axes in a fitted treeda model aligned under the tree.

Usage

```
plot_coefficients(
  out.treeda,
  remove.bl = TRUE,
  ladderize = TRUE,
  tree.height = 2
)
```

Arguments

out.treeda	The object resulting from a call to treeda.
remove.bl	A logical, TRUE if the tree should be plotted after setting all branch lengths equal to the same value or not. The plots tend to look nicer when all the branch lengths are the same, and the branch length information is not used in the model.
ladderize	Layout parameter for the tree.
tree.height	The height of the tree relative to the height of the plot below.

Value

A plot of the tree and the coefficients.

Examples

```
data(treeda_example)
out.treeda = treeda(response = treeda_example$response,
    predictors = treeda_example$predictors,
    tree = treeda_example$tree,
    p = 1)
plot_coefficients(out.treeda)
```

`predict.treeda` *Predict using new data*

Description

Given a fitted `treeda` model, get the predicted classes and projections onto the discriminating axes for new data.

Usage

```
## S3 method for class 'treeda'
predict(object, newdata, newresponse = NULL, check.consist = TRUE, ...)
```

Arguments

`object`	Output from `treeda` function.
`newdata`	New predictor matrix in the same format as the `predictor` argument to treeda. A matrix of predictor variables corresponding to the leaves of the tree and in the same order as the leaves of the tree.
`newresponse`	New response vector, not required.
`check.consist`	Check the consistency between the tree and predictor matrix?
`...`	Not used.

Value

A list containing the projections of the new data onto the discriminating axes (`projections`), the predicted classes (`classes`), and the rss (`rss`, only included if the ground truth for the responses is available).

Examples

```
data(treeda_example)
out.treeda = treeda(response = treeda_example$response,
    predictors = treeda_example$predictors,
    tree = treeda_example$tree,
    p = 1)
## Here we are predicting on the training data, in general this
## would be done on a held out test set
preds = predict(out.treeda, newdata = treeda_example$predictors,
```

```
    newresponse = treeda_example$response)
## make a confusion matrix
table(preds$classes, treeda_example$response)
```

`print.treeda` *Print a treeda object*

Description

Print a treeda object

Usage

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

Arguments

`x`	`treeda` object.
`...`	Not used.

`print.treedacv` *Print treedacv objects*

Description

Print treedacv objects

Usage

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

Arguments

`x`	`treedacv` object.
`...`	Not used

`treeda` *Tree-based sparse discriminant analysis*

Description

Performs tree-structured sparse discriminant analysis using an augmented predictor matrix with additional predictors corresponding to the nodes and then translating the parameters back in terms of only the leaves.

Usage

```
treeda(
  response,
  predictors,
  tree,
  p,
  k = nclasses - 1,
  center = TRUE,
  scale = TRUE,
  class.names = NULL,
  check.consist = TRUE,
  A = NULL,
  ...
)
```

Arguments

`response`	A factor or character vector giving the class to be predicted.
`predictors`	A matrix of predictor variables corresponding to the leaves of the tree and in the same order as the leaves of the tree.
`tree`	A tree of class `phylo`.
`p`	The number of predictors to use.
`k`	The number of components to use.
`center`	Center the predictor variables?
`scale`	Scale the predictor variables?
`class.names`	Optional argument giving the class names.
`check.consist`	Check consistency of the predictor matrix and the tree.
`A`	A matrix describing the tree structure. If it has been computed before it can be passed in here and will not be recomputed.
`...`	Additional arguments to be passed to sda

Value

An object of class `treeda`. Contains the coefficients in the original predictor space (`leafCoefficients`), the number of predictors used in the node + leaf space (`nPredictors`), number of leaf predictors used (`nLeafPredictors`), the projections of the samples onto the discriminating axes (`projections`), and the sparse discriminant analysis object that was used in the fit (`sda`).

Examples

```
data(treeda_example)
out.treeda = treeda(response = treeda_example$response,
    predictors = treeda_example$predictors,
    tree = treeda_example$tree,
    p = 1)
out.treeda
```

`treedacv` *treeda cross validation*

Description

Performs cross-validation of a `treeda` fit.

Usage

```
treedacv(
  response,
  predictors,
  tree,
  folds = 5,
  pvec = 1:tree$Nnode,
  k = nclasses - 1,
  center = TRUE,
  scale = TRUE,
  class.names = NULL,
  ...
)
```

Arguments

`response`	The classes to be predicted.
`predictors`	A matrix of predictors corresponding to the tips of the tree.
`tree`	A tree object of class `phylo`.
`folds`	Either a single number corresponding to the number of folds of cross-validation to perform or a vector of integers ranging from 1 to the number of folds desired giving the partition of the dataset.
`pvec`	The values of p to use.

k	The number of discriminating axes to keep.
center	Center the predictors?
scale	Scale the predictors?
class.names	A vector giving the names of the classes.
...	Additional arguments to be passed to treeda.

Value

A list with the value of p with minimum cv error (p.min), the minimum value of p with in 1 se of the minimum cv error (p.1se), and a data frame containing the loss for each fold, mean loss, and standard error of the loss for each value of p (loss.df).

Examples

```
data(treeda_example)
out.treedacv = treedacv(response = treeda_example$response,
    predictors = treeda_example$predictors,
    tree = treeda_example$tree,
    pvec = 1:10)
out.treedacv
```

---

treeda_example *Example dataset*

Description

A small example dataset with three components, stored as a list with a vector containing the classes (response), a matrix containing the predictor variables (predictors), and a tree describing the relationships between the predictor variables (tree). The dataset consists of 50 samples divided into two classes and 100 taxa/predictor variables, related to each other by a random tree (generated with ape::rtree). A set of 42 taxa descending from one internal node are all over-represented in one class and under-represented in the other. The predictors element in the list contains real numbers, not counts, and is supposed to reflect normalized taxon abundances (e.g., normalization using the variance-stabilizing transformation in DESeq2).

Format

A list containing response variables, predictor variables, and a tree describing the relationship between the predictor variables.

Index