# Package 'LDATree'

August 26, 2023

**Title** Classification Trees with Linear Discriminant Analysis at Terminal Nodes

**Version** 0.1.2

**Description** A classification tree method that uses LDA (Linear Discriminant Analysis) for variable selection, split determination, and model fitting in terminal nodes. It automatically handles missing values and offers visualization tools.

**License** MIT + file LICENSE

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

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

**Imports** ggplot2, lifecycle, magrittr, scales, stats, visNetwork

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

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

**Maintainer** Siyu Wang <swang739@wisc.edu>

**Repository** CRAN

**Date/Publication** 2023-08-25 22:40:10 UTC

## R topics documented:

---

`ldaGSVD` *Linear Discriminant Analysis using the Generalized Singular Value Decomposition*

---

## Description

**[Experimental]** Fit an LDA/GSVD model.

## Usage

```
ldaGSVD(formula, data)
```

## Arguments

| | |
|---|---|
| `formula` | an object of class formula, which has the form `class ~ x1 + x2 + ...` |
| `data` | a data frame that contains both predictors and the response. Missing values are NOT allowed. |

## Details

Traditional Fisher's Linear Discriminant Analysis (LDA) ceases to work when the within-class scatter matrix is singular. The Generalized Singular Value Decomposition (GSVD) is used to address this issue. GSVD simultaneously diagonalizes both the within-class and between-class scatter matrices without the need to invert a singular matrix. This method is believed to be more accurate than PCA-LDA (as in `MASS::lda`) because it also considers the information in the between-class scatter matrix.

## Value

An object of class `ldaGSVD` containing the following components:

- `scaling`: a matrix which transforms the training data to LD scores, normalized so that the within-group scatter matrix is proportional to the identity matrix.
- `formula`: the formula passed to the `ldaGSVD()`
- `terms`: a object of class `terms` derived using the input `formula` and the training data
- `prior`: a `table` of the estimated prior probabilities.
- `groupMeans`: a matrix that records the group means of the training data on the transformed LD scores.
- `xlevels`: a list records the levels of the factor predictors, derived using the input `formula` and the training data

## References

Ye, J., Janardan, R., Park, C. H., & Park, H. (2004). *An optimization criterion for generalized discriminant analysis on undersampled problems*. IEEE Transactions on Pattern Analysis and Machine Intelligence

Howland, P., Jeon, M., & Park, H. (2003). *Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition*. SIAM Journal on Matrix Analysis and Applications

## Examples

```
fit <- ldaGSVD(Species~., data = iris)
# prediction
predict(fit,iris)
```

---

`plot.Treee` *Plot a Treee object*

---

## Description

Provide a diagram of the whole tree structure or a scatter/density plot for a specific tree node.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a fitted model object of class `Treee`, which is assumed to be the result of the `Treee()` function. |
| `data` | the original data you used to fit the `Treee` object if you want the individual plot for each node. Otherwise, you can leave this parameter blank if you only need the overall tree structure diagram. |
| `node` | the node index that you are interested in. By default, it is set to `-1` and the overall tree structure is drawn. |
| `...` | further arguments passed to or from other methods. |

## Value

For overall tree structure (`node = -1`), A figure of class `visNetwork` is drawn. Otherwise, a figure of class `ggplot` is drawn.

### Overall tree structure

A full tree diagram (via the R package visNetwork) is shown if node is not provided (default is -1). The color shows the most common (plurality) class inside each node. The size of each terminal node is based on its relative sample size. Under every node, you see the plurality class, the fraction of the correctly predicted training sample vs. the node's sample size, and the node index, respectively. When you click on the node, an information panel with more details will appear.

### Individual plot for each node

The node index and the original training data are required to return a more detailed plot within a specific node. The density plot will be provided when only two levels are left for the response variable in a node (like in a binary classification problem). Samples are projected down to their first linear discriminant scores (LD1). A scatter plot will be provided if a node contains more than two classes. Samples are projected down to their first and second linear discriminant scores.

### Examples

```
fit <- Treee(Species~., data = iris)
# plot the overall tree
plot(fit)
# plot a certain node
plot(fit, iris, node = 1)
```

---

## predict.ldaGSVD *Predictions from a fitted ldaGSVD object*

---

### Description

Prediction of test data using a fitted ldaGSVD object

### Usage

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

### Arguments

| | |
|---|---|
| object | a fitted model object of class ldaGSVD, which is assumed to be the result of the ldaGSVD() function. |
| newdata | data frame containing the values at which predictions are required. Missing values are NOT allowed. |
| type | character string denoting the type of predicted value returned. The default is to return the predicted class (type = 'response'). The predicted posterior probabilities for each class will be returned if type = 'prob'. |
| ... | further arguments passed to or from other methods. |

## Details

Unlike the original paper, which uses the k-nearest neighbor (k-NN) as the classifier, we use a faster and more straightforward likelihood-based method. One limitation of the traditional likelihood-based method for LDA is that it ceases to work when there are Linear Discriminant (LD) directions with zero variance in the within-class scatter matrix. However, when using LDA/GSVD, all chosen LD directions possess non-zero variance in the between-class scatter matrix. This implies that LD directions with zero variance in the within-class scatter matrix will yield the highest Fisher's ratio. Therefore, to get these directions higher weights, we manually adjust the zero variance to `1e-15` for computational reasons.

## Value

The function returns different values based on the `type`, if

- `type = 'response'`: vector of predicted responses.
- `type = 'prob'`: a data frame of the posterior probabilities. Each class takes a column.

## References

Ye, J., Janardan, R., Park, C. H., & Park, H. (2004). *An optimization criterion for generalized discriminant analysis on undersampled problems*. IEEE Transactions on Pattern Analysis and Machine Intelligence

Howland, P., Jeon, M., & Park, H. (2003). *Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition*. SIAM Journal on Matrix Analysis and Applications

## Examples

```
fit <- ldaGSVD(Species~., data = iris)
predict(fit,iris)
# output prosterior probabilities
predict(fit,iris,type = "prob")
```

---

`predict.Treee` *Predictions from a fitted Treee object*

---

## Description

Prediction of test data using a fitted Treee object

## Usage

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

### Arguments

| | |
|---|---|
| object | a fitted model object of class Treee, which is assumed to be the result of the Treee() function. |
| newdata | data frame containing the values at which predictions are required. Missing values are allowed. |
| type | character string denoting the type of predicted value returned. The default is to return the predicted class (type = 'response'). The predicted posterior probabilities for each class will be returned if type = 'prob'. 'all' returns a data frame with predicted classes, posterior probabilities, and the predicted node indices. If cross-validation is carried out during the LDATree fitting, 'grove' option is available and will output an ensemble result from k LDATrees where k is the number of cross-validation. |
| ... | further arguments passed to or from other methods. |

### Value

The function returns different values based on the type, if

- type = 'response': vector of predicted responses.
- type = 'prob': a data frame of the posterior probabilities. Each class takes a column.
- type = 'all': a data frame contains the predicted responses, posterior probabilities, and the predicted node indices.
- type = 'grove': vector of predicted responses using the ensemble method. Only available when cross-validation is carried out during the tree generating process.

Note: for factor predictors, if it contains a level which is not used to grow the tree, it will be converted to missing and will be imputed according to the missingMethod in the fitted tree.

### Examples

```
fit <- Treee(Species~., data = iris)
predict(fit,iris)
# output prosterior probabilities
predict(fit,iris,type = "prob")
```

---

## Treee *Classification trees with Linear Discriminant Analysis terminal nodes*

---

### Description

**[Experimental]** Fit an LDATree model.

## Usage

```
Treee(
  formula,
  data,
  missingMethod = c("meanFlag", "newLevel"),
  splitMethod = "LDscores",
  pruneMethod = "none",
  numberOfPruning = 10,
  maxTreeLevel = 4,
  minNodeSize = NULL,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| formula | an object of class formula, which has the form `class ~ x1 + x2 + ...` |
| data | a data frame that contains both predictors and the response. Missing values are allowed in predictors but not in the response. |
| missingMethod | Missing value solutions for numerical variables and factor variables. `'mean'`, `'median'`, `'meanFlag'`, `'medianFlag'` are available for numerical variables. `'mode'`, `'modeFlag'`, `'newLevel'` are available for factor variables. The word `'Flag'` in the methods indicates whether a missing flag is added or not. The `'newLevel'` method means that all missing values are replaced with a new level rather than imputing them to another existing value. |
| splitMethod | the splitting rule in LDATree growing process. For now, `'LDscores'` is the only available option. |
| pruneMethod | the model selection method in the LDATree growing process, which controls the size of the tree. By default, it's set to `'none'`, which applies a direct stopping rule. Alternatively, `'CV'` uses the alpha-pruning process from CART. Although `'CV'` is often more accurate, it can be slower, especially with large datasets. |
| numberOfPruning | controls the number of cross-validation in the pruning. It is 10 by default. |
| maxTreeLevel | controls the largest tree size possible for either a direct-stopping tree or a CV-pruned tree. Adding one extra level (depth) introduces an additional layer of nodes at the bottom of the current tree. e.g., when the maximum level is 1 (or 2), the maximum tree size is 3 (or 7). |
| minNodeSize | controls the minimum node size. Think carefully before changing this value. Setting a large number might result in early stopping and reduced accuracy. By default, it's set to one plus the number of classes in the response variable. |
| verbose | a logical. If TRUE, the function provides additional diagnostic messages or detailed output about its progress or internal workings. Default is FALSE, where the function runs silently without additional output. |

## Details

Unlike other classification trees, LDATree integrates LDA throughout the entire tree-growing process. Here is a breakdown of its distinctive features:

- The tree searches for the best binary split based on sample quantiles of the first linear discriminant score.
- An LDA/GSVD model is fitted for each terminal node (For more details, refer to `ldaGSVD()`).
- Missing values can be imputed using the mean, median, or mode, with optional missing flags available.
- By default, the tree employs a direct-stopping rule. However, cross-validation using the alpha-pruning from CART is also provided.

## Value

An object of class `Treee` containing the following components:

- `formula`: the formula passed to the `Treee()`
- `treee`: a list of all the tree nodes, and each node is an object of class `TreeeNode`.
- `missingMethod`: the missingMethod passed to the `Treee()`

  An object of class `TreeeNode` containing the following components:
- `currentIndex`: the node index of the current node
- `currentLevel`: the level of the current node in the tree
- `idxRow`, `idxCol`: the row and column indices showing which portion of data is used in the current node
- `currentLoss`: the training error (number of misclassified sample) of the current node
- `accuracy`: the training accuracy of the current node
- `proportions`: shows the observed frequency for each class
- `parent`: the node index of its parent
- `children`: the node indices of its direct children (not including its children's children)
- `misReference`: a data frame, serves as the reference for missing value imputation
- `splitCut`: the cut point of the split
- `nodeModel`: one of `'mode'` or `'LDA'`. It shows the type of predictive model fitted in the current node
- `nodePredict`: the fitted predictive model in the current node. It is an object of class `ldaGSVD` if LDA is fitted. If `nodeModel = 'mode'`, then it is a vector of length one, showing the plurality class.
- `offsprings`: (available only if `pruneMethod = 'CV'`) showing all terminal descendant nodes of the current node
- `offspringLoss`: (available only if `pruneMethod = 'CV'`) sum of the `currentLoss` of the `offsprings` of the current node
- `alpha`: (available only if `pruneMethod = 'CV'`) the alpha in alpha-pruning from CART

## Examples

```
fit <- Treee(Species~., data = iris)
# Use cross-validation to prune the tree
fitCV <- Treee(Species~., data = iris, pruneMethod = "CV")
# prediction
predict(fit,iris)
# plot the overall tree
plot(fit)
# plot a certain node
plot(fit, iris, node = 1)
```

# Index