Package 'folda'

July 22, 2025

Title Forward Stepwise Discriminant Analysis with Pillai's Trace

Version 0.2.0

Description A novel forward stepwise discriminant analysis framework
that integrates Pillai's trace with Uncorrelated Linear Discriminant Analysis (ULDA),
providing an improvement over traditional stepwise LDA methods that rely on Wilks' Lambda.
A stand-alone ULDA implementation is also provided, offering a more general solution
than the one available in the 'MASS' package. It automatically handles missing values and
provides visualization tools. For more de-
tails, see Wang (2024) <doi:10.48550/arXiv.2409.03136>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports ggplot2, grDevices, Rcpp, stats

URL https://github.com/Moran79/folda, http://iamwangsiyu.com/folda/

BugReports https://github.com/Moran79/folda/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

LinkingTo Rcpp, RcppEigen

VignetteBuilder knitr

NeedsCompilation yes

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

Maintainer Siyu Wang <iamwangsiyu@gmail.com>

Repository CRAN

Date/Publication 2024-10-29 22:20:02 UTC

Contents

`checkPriorAndMisClassCost`

Check and Normalize Prior Probabilities and Misclassification Costs

---

Description

This function verifies and normalizes the provided prior probabilities and misclassification cost matrix for a given response variable. It ensures that the lengths of the prior and the dimensions of the misclassification cost matrix match the number of levels in the response variable. If `prior` or `misClassCost` are not provided, default values are used: the prior is set to the observed frequencies of the response, and the misclassification cost matrix is set to 1 for all misclassifications and 0 for correct classifications.

Usage

```
checkPriorAndMisClassCost(prior, misClassCost, response)
```

Arguments

`prior`	A numeric vector representing the prior probabilities for each class in the response variable. If `NULL`, the observed frequencies of the response are used as the default prior.
`misClassCost`	A square matrix representing the misclassification costs for each pair of classes in the response variable. If `NULL`, a default misclassification matrix is created where all misclassifications have a cost of 1 and correct classifications have a cost of 0.
`response`	A factor representing the response variable with multiple classes.

Value

A list containing:

`prior`	A normalized vector of prior probabilities for each class.
`misClassCost`	A square matrix representing the misclassification costs, with rows and columns labeled by the levels of the response variable.

Examples

```
# Example 1: Using default prior and misClassCost
response <- factor(c('A', 'B', 'A', 'B', 'C', 'A'))
checkPriorAndMisClassCost(NULL, NULL, response)

# Example 2: Providing custom prior and misClassCost
prior <- c(A = 1, B = 1, C = 2)
misClassCost <- matrix(c(0, 2, 10,
                         1, 0, 10,
                         1, 2, 0), nrow = 3, byrow = TRUE)
checkPriorAndMisClassCost(prior, misClassCost, response)
```

---

folda *Forward Uncorrelated Linear Discriminant Analysis*

Description

This function fits a ULDA (Uncorrelated Linear Discriminant Analysis) model to the provided data, with an option for forward selection of variables based on Pillai's trace or Wilks' Lambda. It can also handle missing values, perform downsampling, and compute the linear discriminant scores and group means for classification. The function returns a fitted ULDA model object.

Usage

```
folda(
  datX,
  response,
  subsetMethod = c("forward", "all"),
  testStat = c("Pillai", "Wilks"),
  correction = TRUE,
  alpha = 0.1,
  prior = NULL,
  misClassCost = NULL,
  missingMethod = c("medianFlag", "newLevel"),
  downSampling = FALSE,
  kSample = NULL
)
```

Arguments

datX	A data frame of predictor variables.
response	A factor representing the response variable with multiple classes.
subsetMethod	A character string specifying the method for variable selection. Options are "forward" for forward selection or "all" for using all variables. Default is "forward".
testStat	A character string specifying the test statistic to use for forward selection. Options are "Pillai" or "Wilks". Default is "Pillai".

correction	A logical value indicating whether to apply a multiple comparison correction during forward selection. Default is TRUE.
alpha	A numeric value between 0 and 1 specifying the significance level for the test statistic during forward selection. Default is 0.1.
prior	A numeric vector representing the prior probabilities for each class in the response variable. If NULL, the observed class frequencies are used as the prior. Default is NULL.
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 vector of length 2 specifying how to handle missing values for numerical and categorical variables, respectively. Default is c("medianFlag", "newLevel").
downSampling	A logical value indicating whether to perform downsampling to balance the class distribution in the training data or to improve computational efficiency. Default is FALSE. Note that if downsampling is applied and the prior is NULL, the class prior will be calculated based on the downsampled data. To retain the original prior, please specify it explicitly using the prior parameter.
kSample	An integer specifying the maximum number of samples to take from each class during downsampling. If NULL, the number of samples is limited to the size of the smallest class. Default is NULL.

Value

A list of class ULDA containing the following components:

scaling	The matrix of scaling coefficients for the linear discriminants.
groupMeans	The group means of the linear discriminant scores.
prior	The prior probabilities for each class.
misClassCost	The misclassification cost matrix.
misReference	A reference for handling missing values.
terms	The terms used in the model formula.
xlevels	The levels of the factors used in the model.
varIdx	The indices of the selected variables.
varSD	The standard deviations of the selected variables.
varCenter	The means of the selected variables.
statPillai	The Pillai's trace statistic.
pValue	The p-value associated with Pillai's trace.
predGini	The Gini index of the predictions on the training data.
confusionMatrix	The confusion matrix for the training data predictions.
forwardInfo	Information about the forward selection process, if applicable.
stopInfo	A message indicating why forward selection stopped, if applicable.

References

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

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 the ULDA model
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "all")

# Fit the ULDA model with forward selection
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "forward")
```

---

getChiSqStat *Compute Chi-Squared Statistics for Variables*

Description

This function calculates the chi-squared statistic for each column of `datX` against the response variable `response`. It supports both numerical and categorical predictors in `datX`. For numerical variables, it automatically discretizes them into factor levels based on standard deviations and mean, using different splitting criteria depending on the sample size.

Usage

```
getChiSqStat(datX, response)
```

Arguments

`datX`	A matrix or data frame containing predictor variables. It can consist of both numerical and categorical variables.
`response`	A factor representing the class labels. It must have at least two levels for the chi-squared test to be applicable.

Details

For each variable in `datX`, the function first checks if the variable is numerical. If so, it is discretized into factor levels using either two or three split points, depending on the sample size and the number of levels in the `response`. Missing values are handled by assigning them to a new factor level.

The chi-squared statistic is then computed between each predictor and the `response`. If the chi-squared test has more than one degree of freedom, the Wilson-Hilferty transformation is applied to adjust the statistic to a 1-degree-of-freedom chi-squared distribution.

Value

A vector of chi-squared statistics, one for each predictor variable in datX. For numerical variables, the chi-squared statistic is computed after binning the variable.

References

Loh, W. Y. (2009). Improving the precision of classification trees. *The Annals of Applied Statistics*, 1710–1737. JSTOR.

Examples

```
datX <- data.frame(var1 = rnorm(100), var2 = factor(sample(letters[1:3], 100, replace = TRUE)))
y <- factor(sample(c("A", "B"), 100, replace = TRUE))
getChiSqStat(datX, y)
```

---

getDataInShape *Align Data with a Missing Reference*

Description

This function aligns a given dataset (data) with a reference dataset (missingReference). It ensures that the structure, column names, and factor levels in data match the structure of missingReference. If necessary, missing columns are initialized with NA, and factor levels are adjusted to match the reference. Additionally, it handles the imputation of missing values based on the reference and manages flag variables for categorical or numerical columns.

Usage

```
getDataInShape(data, missingReference)
```

Arguments

data	A data frame to be aligned and adjusted according to the missingReference.
missingReference	A reference data frame that provides the structure (column names, factor levels, and missing value reference) for aligning data.

Value

A data frame where the structure, column names, and factor levels of data are aligned with missingReference. Missing values in data are imputed based on the first row of the missingReference, and flag variables are updated accordingly.

Examples

```
data <- data.frame(
  X1_FLAG = c(0, 0, 0),
  X1 = factor(c(NA, "C", "B"), levels = LETTERS[2:3]),
  X2_FLAG = c(NA, 0, 1),
  X2 = c(2, NA, 3)
)

missingReference <- data.frame(
  X1_FLAG = 1,
  X1 = factor("A", levels = LETTERS[1:2]),
  X2 = 1,
  X2_FLAG = 1
)

getDataInShape(data, missingReference)
```

---

getMode *Calculate the Mode of a Factor Variable with Optional Priors*

Description

This function calculates the mode of a given factor or vector that can be coerced into a factor. You can optionally provide prior weights for each level of the factor.

Usage

```
getMode(v, prior)
```

Arguments

v	A factor or vector that can be coerced into a factor. The mode will be calculated from the levels of this factor.
prior	A numeric vector of prior weights for each level of the factor. If not provided, all levels will be given equal weight.

Value

The mode of the factor v as a character string. If all values are NA, the function returns NA.

Examples

```
# Example 1: Mode without priors
v <- factor(c("apple", "banana", "apple", "orange", NA))
getMode(v)

# Example 2: Mode with priors
v <- factor(c("apple", "banana", "apple", "orange", NA))
prior <- c(apple = 0.5, banana = 1.5, orange = 1)
getMode(v, prior)
```

`missingFix` *Impute Missing Values and Add Missing Flags to a Data Frame*

Description

This function imputes missing values in a data frame based on specified methods for numerical and categorical variables. Additionally, it can add flag columns to indicate missing values. For numerical variables, missing values can be imputed using the mean or median. For categorical variables, missing values can be imputed using the mode or a new level. This function also removes constant columns (all NAs or all observed but the same value).

Usage

```
missingFix(data, missingMethod = c("medianFlag", "newLevel"))
```

Arguments

`data`	A data frame containing the data to be processed. Missing values (`NA`) will be imputed based on the methods provided in `missingMethod`.
`missingMethod`	A character vector of length 2 specifying the methods for imputing missing values. The first element specifies the method for numerical variables (`"mean"`, `"median"`, `"meanFlag"`, or `"medianFlag"`), and the second element specifies the method for categorical variables (`"mode"`, `"modeFlag"`, or `"newLevel"`). If `"Flag"` is included, a flag column will be added for the corresponding variable type.

Value

A list with two elements:

`data`	The original data frame with missing values imputed, and flag columns added if applicable.
`ref`	A reference row containing the imputed values and flag levels, which can be used for future predictions or reference.

Examples

```
dat <- data.frame(
  X1 = rep(NA, 5),
  X2 = factor(rep(NA, 5), levels = LETTERS[1:3]),
  X3 = 1:5,
  X4 = LETTERS[1:5],
  X5 = c(NA, 2, 3, 10, NA),
  X6 = factor(c("A", NA, NA, "B", "B"), levels = LETTERS[1:3])
)
missingFix(dat)
```

plot.ULDA *Plot Decision Boundaries and Linear Discriminant Scores*

Description

This function plots the decision boundaries and linear discriminant (LD) scores for a given ULDA model. If it is a binary classification problem, a density plot is created. Otherwise, a scatter plot with decision boundaries is generated.

Usage

```
## S3 method for class 'ULDA'
plot(x, datX, response, ...)
```

Arguments

x	A fitted ULDA model object.
datX	A data frame containing the predictor variables.
response	A factor representing the response variable (training labels) corresponding to datX.
...	Additional arguments.

Value

A ggplot2 plot object, either a density plot or a scatter plot with decision boundaries.

Examples

```
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "all")
plot(fit, iris[, -5], iris[, 5])
```

predict.ULDA *Predict Method for ULDA Model*

Description

This function predicts the class labels or class probabilities for new data using a fitted ULDA model. The prediction can return either the most likely class ("response") or the posterior probabilities for each class ("prob").

Usage

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

Arguments

object	A fitted ULDA model object.
newdata	A data frame containing the new predictor variables for which predictions are to be made.
type	A character string specifying the type of prediction to return. "response" returns the predicted class labels, while "prob" returns the posterior probabilities for each class. Default is "response".
...	Additional arguments.

Value

If type = "response", the function returns a vector of predicted class labels. If type = "prob", it returns a matrix of posterior probabilities, where each row corresponds to a sample and each column to a class.

Examples

```
fit <- folda(datX = iris[, -5], response = iris[, 5], subsetMethod = "all")

# Predict class labels
predictions <- predict(fit, iris, type = "response")

# Predict class probabilities
prob_predictions <- predict(fit, iris, type = "prob")
```

Index