Package ‘binda’

July 8, 2015

Version 1.0.3
Date 2015-07-07
Title Multi-Class Discriminant Analysis using Binary Predictors
Author Sebastian Gibb and Korbinian Strimmer.
Maintainer Korbinian Strimmer <strimmerlab@gmail.com>
Depends R (>= 3.0.2), entropy (>= 1.2.1)
Imports graphics, stats, utils
Suggests crossval
Description The "binda" package implements functions for multi-class
discriminant analysis using binary predictors, for corresponding
variable selection, and for dichotomizing continuous data.
License GPL (>= 3)
URL http://strimmerlab.org/software/binda/
NeedsCompilation no
Repository CRAN
Date/Publication 2015-07-08 00:33:03

R topics documented:

binda-package ........................................... 2
binda ...................................................... 2
binda.ranking ............................................. 4
chances ................................................... 6
dichotomize ............................................. 7
is.binaryMatrix ....................................... 9

Index 10
The binda Package

Description

The "binda" package implements functions for multi-class discriminant analysis using binary predictors, for corresponding variable selection, and for dichotomizing continuous data.

Author(s)

Sebastian Gibb and Korbinian Strimmer (http://strimmerlab.org/)

References


Website: http://strimmerlab.org/software/binda/

See Also

binda, binda.ranking, dichotomize, chances.

binda

Binary Discriminant Analysis: Model Fit and Class Prediction

Description

binda trains a diagonal multivariate Bernoulli model. predict.binda performs corresponding class prediction.

Usage

binda(Xtrain, L, lambda.freqs, verbose=TRUE)
## S3 method for class 'binda'
predict(object, Xtest, verbose=TRUE, ...)

Arguments

Xtrain  A matrix containing the training data set. Note that the rows correspond to observations and the columns to variables.
L  A factor with the class labels of the training samples.
lambda.freqs  Shrinkage intensity for the frequencies. If not specified it is estimated from the data. lambda.freqs=0 implies no shrinkage (i.e. empirical frequencies) and lambda.freqs=1 complete shrinkage (i.e. uniform frequencies).
verbose: Report shrinkage intensities (binda) and number of used features (predict.binda).

object: An binda fit object obtained from the function binda.

Xtest: A matrix containing the test data set. Note that the rows correspond to observations and the columns to variables.

...: Additional arguments for generic predict.

Details
For detailed description of binary discriminant analysis as implemented in binda see Gibb and Strimmer (2015).

Value
predict.binda predicts class probabilities for each test sample and returns a list with two components:

class: a factor with the most probable class assignment for each test sample, and

posterior: a matrix containing the respective class posterior probabilities.

Author(s)

References

See Also
binda.ranking.

Examples

# load "binda" library
library("binda")

# training data set with labels
Xtrain = matrix(c(1, 1, 0, 1, 0, 0,
                  1, 1, 1, 0, 0, 0,
                  1, 0, 0, 1, 1, 1,
                  1, 0, 0, 1, 1, 1), nrow=4, byrow=TRUE)

colnames(Xtrain) = paste0("v", 1:ncol(Xtrain))
is.binaryMatrix(Xtrain) # TRUE
L = factor(c("Treatment", "Treatment", "Control", "Control") )

# learn predictor
binda.fit = binda(Xtrain, L)
binda.ranking

# predict classes using new test data
Xtest = matrix(c(1, 1, 0, 1, 1,
                 1, 0, 0, 1, 1), nrow=2, byrow=TRUE)
colnames(Xtest) = paste0("v", 1:nrow(Xtest))
predict(binda.fit, Xtest)

binda.ranking

Binary Discriminant Analysis: Variable Ranking

Description
binda.ranking determines a ranking of predictors by computing corresponding t-scores between
the group means and the pooled mean.
plot.binda.ranking provides a graphical visualization of the top ranking variables

Usage
binda.ranking(Xtrain, L, lambda.freqs, verbose=TRUE)
## S3 method for class 'binda.ranking'
plot(x, top=40, arrow.col="blue", zeroaxis.col="red", ylab="Variables", main, ...)

Arguments

- **Xtrain** A matrix containing the training data set. Note that the rows correspond to
  observations and the columns to variables.
- **L** A factor with the class labels of the training samples.
- **lambda.freqs** Shrinkage intensity for the class frequencies. If not specified it is estimated from
  the data. lambda.freqs=0 implies no shrinkage (i.e. empirical frequencies) and
  lambda.freqs=1 complete shrinkage (i.e. uniform frequencies).
- **verbose** Print out some info while computing.
- **x** A "binda.ranking" object – this is produced by the binda.ranking() function.
- **top** The number of top-ranking variables shown in the plot (default: 40).
- **arrow.col** Color of the arrows in the plot (default is "blue").
- **zeroaxis.col** Color for the center zero axis (default is "red").
- **ylab** Label written next to feature list (default is "Variables").
- **main** Main title (if missing, "The", top, "Top Ranking Variables" is used).
- **...** Other options passed on to generic plot().

Details
The overall ranking of a feature is determined by computing a weighted sum of the squared t-scores.
This is approximately equivalent to the mutual information between the response and each variable.
The same criterion is used in dichotomize. For precise details see Gibb and Strimmer (2015).
**Value**

`binda.ranking` returns a matrix with the following columns:

- **idx**: original feature number
- **score**: the score determining the overall ranking of a variable
- **t**: for each group and feature the t-score of the class mean versus the pooled mean

**Author(s)**

Sebastian Gibb and Korbinian Strimmer ([http://strimmerlab.org](http://strimmerlab.org)).

**References**


**See Also**

`binda`, `predict.binda`, `dichotomize`.

**Examples**

```r
# load "binda" library
library("binda")

# training data set with labels
Xtrain = matrix(c(1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1), nrow=4, byrow=TRUE)
colnames(Xtrain) = paste0("v", 1:ncol(Xtrain))
is.binaryMatrix(Xtrain) # TRUE
L = factor(c("Treatment", "Treatment", "Control", "Control"))

# ranking variables
br = binda.ranking(Xtrain, L)

# show plot
plot(br)
```

# result: variable V1 is irrelevant for distinguishing the two groups

## Description

`chances` estimates Bernoulli parameters (=chances) from a binary matrix and associated class labels.

### Usage

```r
chances(x, L, lambda.freqs, verbose = TRUE)
```

### Arguments

- `x`: data matrix (columns correspond to variables, rows to samples).
- `L`: factor containing the class labels, one for each sample (row).
- `lambda.freqs`: shrinkage parameter for class frequencies (if not specified it is estimated).
- `verbose`: report shrinkage intensity and other information.

### Details

The class-specific chances are estimated using the empirical means over the 0s and 1s in each class. For estimating the pooled mean the class-specific means are weighted using the estimated class frequencies. Class frequencies are estimated using `freqs.shrink`.

### Value

- `samples`: the samples in each class,
- `regularization`: the shrinkage intensity used to estimate the class frequencies,
- `freqs`: the estimated class frequencies,
- `means`: the estimated chances (parameters of Bernoulli distribution, expectations of 1s) for each variable conditional on class, as well as the marginal changes (pooled means).

### Author(s)


### See Also

`is.binaryMatrix`
Examples

# load binda library
library("binda")

# example binary matrix with 6 variables (in columns) and 4 samples (in rows)
Xb = matrix(c(1, 1, 0, 1, 0, 0,
              1, 1, 1, 0, 0, 0,
              1, 0, 0, 1, 1,
              1, 0, 0, 1, 1), nrow=4, byrow=TRUE)
colnames(Xb) = paste0("v", 1:ncol(Xb))

# Test for binary matrix
is.binaryMatrix(Xb) # TRUE

L = factor(c("Treatment", "Treatment", "Control", "Control")

chances(Xb, L)

dichotomize

Dichotomize Continuous Data Set With Labels

Description

dichotomize converts a matrix containing continuous measurements into a binary matrix. optimizeThreshold determines optimal thresholds for dichotomization.

Usage

dichotomize(X, thresh)
optimizeThreshold(X, L, lambda.freqs, verbose=FALSE)

Arguments

X data matrix (columns correspond to variables, rows to samples).
thresh vector of thresholds, one for each variable (column).
L factor containing the class labels, one for each sample (row).
lambda.freqs shrinkage parameter for class frequencies (if not specified it is estimated).
verbose report shrinkage intensity and other information.

Details

dichotomize assigns 0 if a matrix entry is lower than given column-specific threshold, otherwise it assigns 1.
optimizeThreshold uses (approximate) mutual information to determine the optimal thresholds. Specifically, the thresholds are chosen to maximize the mutual information between response and each variable. The same criterion is also used in binda.ranking. For detailed description of the dichotomization procedure see Gibb and Strimmer (2015).

Class frequencies are estimated using freqs.shrink.
### Value

dichotomize returns a binary matrix.

optimizeThreshold returns a vector containing the variable thresholds.

### Author(s)


### References


### See Also

binda.ranking, freqs.shrink, mi.plugin, is.binaryMatrix.

### Examples

```r
# load binda library
library("binda")

# example data with 6 variables (in columns) and 4 samples (in rows)
X = matrix(c(1L, 1L, 1L, 1.7UL, 0.TL, 0L,
            1L, 1L, 2, 0.4, 0.09,
            1L, 0L, 1L, 1.5, 0.1,
            1L, 0L, 1L, 6, 0.1), nrow=4, byrow=TRUE)

# class labels
L = factor(c("Treatment", "Treatment", "Control", "Control") )
rownames(X) = paste0(L, rep(1:2, times=2))

X
# V1 V2 V3 V4 V5 V6
#Treatment1 1 1 1 1.75 0.4 0.00
#Treatment2 1 1 2 2.00 0.4 0.09
#Control1 1 0 1 1.00 0.5 0.10
#Control2 1 0 1 0.50 0.6 0.10

# find optimal thresholds (one for each variable)
thr = optimizeThreshold(X, L)

# V1 V2 V3 V4 V5 V6
#1.00 1.00 2.00 1.75 0.50 0.10

# convert into binary matrix
# if value is lower than threshold -> 0 otherwise -> 1
Xb = dichotomize(X, thr)
is.binaryMatrix(Xb) # TRUE
```
is.binaryMatrix  

Description

is.binaryMatrix tests whether \( m \) is a matrix and whether it contains only 0s and 1s. Note that functions like binda.ranking and binda require a binary matrix as input.

Usage

```r
is.binaryMatrix(m)
```

Arguments

- \( m \): a matrix.

Value

is.binaryMatrix returns either TRUE or FALSE.

Author(s)


Examples

```r
# load binda library
library("binda")

# test matrix
m = matrix(c(1, 1, 0, 1, 0, 0,
            1, 1, 1, 0, 0,
            1, 0, 0, 0, 1, 1,
            1, 0, 0, 0, 1, 1), nrow=4, byrow=TRUE)

# Test for binary matrix
is.binaryMatrix(m) # TRUE
```
Index

*Topic multivariate
  binda, 2
  binda-package, 2
  binda.ranking, 4
  chances, 6
*Topic univar
  dichotomize, 7
  is.binaryMatrix, 9
binda, 2, 2, 5, 9
binda-package, 2
binda.ranking, 2, 3, 4, 7–9
chances, 2, 6
dichotomize, 2, 4, 5, 7
freqs.shrink, 6–8
is.binaryMatrix, 6, 8, 9
mi.plugin, 8
optimizeThreshold(dichotomize), 7
plot.binda.ranking (binda.ranking), 4
predict.binda, 5
predict.binda (binda), 2