Package: tabularMLC (via r-universe)

October 19, 2026

Type Package

Title Tabular Maximum Likelihood Classifier

Version 0.0.4

Description The maximum likelihood classifier (MLC) is one of the most common classifiers used for remote sensing imagery. This package uses 'RcppArmadillo' to provide a fast implementation of the MLC to train and predict over tabular data (data.frame). The algorithms were based on Mather (1985) <doi:10.1080/01431168508948456> method.

License GPL-3

Depends Rcpp, methods

Imports stats

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.2

Encoding UTF-8

URL https://github.com/caiohamamura/tabularMLC

BugReports https://github.com/caiohamamura/tabularMLC/issues

NeedsCompilation yes

Author Caio Hamamura [aut, cre] (ORCID: <https://orcid.org/0000-0001-6149-5885>)

Maintainer Caio Hamamura <caiohamamura@gmail.com>

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

Date/Publication 2025-09-22 19:40:03 UTC

RemoteUrl https://github.com/cran/tabularMLC

RemoteRef HEAD

RemoteSha c4245e443f9f9fc35c5c324e759b813c89f09632

Contents

tabularMLC-package 2
MLC 3
MLC.model-class 4
predict.MLC.model 4

Index **6**

`tabularMLC-package` *Tabular maximum likelihood classifier*

Description

Maximum likelihood is a common classifier used for land use classification. It calculates the likelihood of an object to belong to each class based on an expected distribution and a metric of distance.

Details

The most common implementation, like in this package, will assume normal distributed variables within classes, and calculate the distance, based on Mahalanobis distance.

Author(s)

Maintainer: Caio Hamamura <caiohamamura@gmail.com> (ORCID)

References

Mather, P. M. (1985). Remote sensing letters: A computationally efficient maximum-likelihood classifier employing prior probabilities for remotely-sensed data. International Journal of Remote Sensing, 6(2), 369–376. doi:10.1080/01431168508948456

Imports

See Also

Useful links:

- https://github.com/caiohamamura/tabularMLC
- Report bugs at https://github.com/caiohamamura/tabularMLC/issues

MLC *Maximum Likelihood Classifier*

Description

Function to create the classifier class from the training set

Usage

```
MLC(x, ...)

## S3 method for class 'formula'
MLC(formula, data = NULL, ...)

## Default S3 method:
MLC(x, y = NULL, ...)
```

Arguments

x	feature vector for the training set
...	for other signatures
formula	formula. The formula for defining the model.
data	the dataset
y	factor vector with the training set labels

Value

An object of class MLC.model parameters used for the model

Examples

```
data(iris)

x = iris[, -5]
y = iris$Species

# Default x y interface
mlcModel1 = MLC(x, y)

# Formula interface
mlcModel2 = MLC(Species ~ Petal.Length + Petal.Width, iris)

# Formula except one column
mlcModel3 = MLC(Species ~ . - Sepal.Length, iris)
```

MLC.model-class *Maximum likelihood model class*

Description

Maximum likelihood model class

Slots

`k` the constant fraction to be used in model $\frac{1}{(2\pi)^{\frac{L}{2}}\sqrt{|\Sigma_i|}}$

`mu` mean (μ_i) list for each variable and class

`inverseCovarianceMatrices` inverted covariance matrix (Σ_i) for each class

`groups` the classification levels

`vars` the variables used for trainning the model

See Also

`MLC` which creates this class

predict.MLC.model *Predict function for MLC.model-class*

Description

`predict` is inherited from the generic function for predictions from the results.

Usage

```
## S3 method for class 'MLC.model'
predict(object, x = NULL, likelihood = FALSE, ...)
```

Arguments

`object`	`MLC.model-class` model class to use for prediction
`x`	data.frame. The feature vector to predict
`likelihood`	logical. Whether to return or not the likelihood values, default FALSE.
`...`	inherited from generic function (not in use)

Value

a factor vector with the predicted value. If likelihood is TRUE, then it will also return the calculated likelihoods.

Examples

```
data(iris)

n = length(iris$Species)

# Split training by sample
training = sample(1:n, size=n*0.7)
validation = (1:n)[-training]

# Train model with training dataset
mlcModel = MLC(Species ~ ., iris[training,])

# Predict using validation dataset
predict = predict(mlcModel, iris[validation,])

# Print confusion matrix
confusionMatrix = table(predicted=predict, observed=iris$Species[validation])
print(confusionMatrix)
```

Index

MLC, 3, *4*
MLC.model, *3*
MLC.model-class, 4

predict.MLC.model, 4

tabularMLC *(*tabularMLC-package*)*, 2
tabularMLC-package, 2