Package ‘rModeling’

February 17, 2020

Type Package

Title A Framework of Cross-Validation

Version 0.0.3

Date 2020-02-06

Author Shuxia Guo [cre, aut],
Thomas Bocklitz [aut],
Juergen Popp [ctb, cph]

Maintainer Shuxia Guo <shuxia.guo@uni-jena.de>

Description A framework of cross-validation for spectral data analysis that allows for automatic tuning of model parameters and for model evaluation. It is particularly useful for applications where intra-group heterogeneity is significant due to inter individual differences. S. Guo, T. Bocklitz, U. Neugebauer, J.Popp (2017) <doi:10.1039/C7AY01363A>.

License GPL-2

Depends R (>= 2.10)

Imports MASS, caret, e1071

RoxygenNote 6.1.0

LazyData false

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-17 15:10:02 UTC

R topics documented:

rModeling-package .................................................. 2
crossValidation ..................................................... 2
DATA ................................................................. 5
dataSplit ............................................................ 6
fnPcaLda ............................................................. 7
predPcaLda .......................................................... 8
predSummary ......................................................... 9
tunePcaLda .......................................................... 10
Description

A cross-validation framework, allowing for model optimization and model evaluation based on batch-wise or normal k-fold cross-validation. It is built based on the ideas in S. Guo, T. Bocklitz, et al., Analytical Methods 2017, 9 (30): 4410-4417. In applications with significant intra-group heterogeneity, the batch-wise cross-validation ensures a robust and reliable statistical modeling and model evaluation.

Details

- Package: rModeling
- Type: Package
- Version: 0.0.1
- Date: 2020-01-23
- License: GPL-2
- Depends: MASS, caret

The main function is `crossValidation`. It can be used as an independent function for model evaluation or as a wrapper of a user-defined function to optimize the parameters of a model.

Author(s)

Shuxia Guo, Thomas Bocklitz, Juergen Popp

Maintainer: Shuxia Guo<shuxia.guo@uni-jena.de>, Thomas Bocklitz<thomas.bocklitz@uni-jena.de>, Juergen Popp<juergen.popp@ipht-jena.de>

References


Description

Conduct a cross-validation for a given classification/regression model and output the prediction results collected over the cross-validation loop. The cross-validation can be done in two ways: normal k-fold cross-validation (`batch=NULL`), or batch-wise cross-validation (`batch!=NULL`). The latter is particularly useful in the presence of significant intra-group heterogeneity.
crossValidation

Usage

crossValidation(data, label, batch = NULL,
method = lda, pred = predict, classify = TRUE,
folds = NULL, nBatch = 0, nFold = 10,
verbose = TRUE, seed = NULL, ...)

Arguments

data a data matrix, with samples saved in rows and features in columns.
label a vector of response variables (i.e., group/concentration info), must be the same length as the number of samples.
batch a vector of sample identifications (e.g., batch/patient ID), must be the same length as the number of samples. Ideally, this should be the identification of the samples at the highest hierarchy (e.g., the patient ID rather than the spectral ID). If missing, a normal k-fold cross validation will be performed (i.e., the data is split randomly into k folds). Ignored if folds is given.
method the name of the function to be performed on training data (can be any model-based procedures, like classification/regression or even pre-processings). A user-defined function is possible, see fnPcaLda as an example.
pred the name of the function to be performed on testing data (e.g. new substances) based on the model built by method. A user-defined function is possible, see predPcaLda as an example.
classify a boolean value, classify=TRUE means a classification task, otherwise a regression task. It is used in the function predSummary.
folds a list of indices specifying the sample index to be used in each fold, can be the output of function dataSplit. If missing, a data split will be done first before performing cross-validation
nBatch an integer, the number of data folds in case of batch-wise cross-validation (if nBatch=0, each batch will be used as one fold). Ignored if folds is given or if batch is missing.
nFold an integer, the value of k in case of normal k-fold cross-validation. Ignored if folds or batch is given.
verbose a boolean value, if or not to print out the logging info
seed an integer, if given, will be used as the random seed to split the data in case of k-fold cross-validation. Ignored if batch or folds is given.
... parameters to be passed to the method

Details

The cross-validation will be conducted based on the data partitions folds, each fold is predicted once using the model built on the rest folds. If folds is missing, a data split will be done first (see more in dataSplit).

The procedures to be performed within the cross-validation is given in the function method, for example, fnPcaLda. A user-defined function is possible, as long as the it follows the same structure as fnPcaLda. A two-layer cross-validation (see reference) can be done by using a tuning function
as method, such as `tunePcaLda` (see examples). In this case, the parameters of a classifier are 
optimized using the training data within `tunePcaLda` and the optimal model is tested on the testing 
data. The parameters of pre-processing can be optimized in a similar way by involving the pre-
processing steps into the function method.

NOTE: It is recommended to specify the seed for a normal k-fold cross-validation in order to get 
the same results from repeated runnings.

Value

A list with elements

- **Fold**: a list, each giving the sample indices of a fold
- **True**: a vector of characters, the groundtruth response variables, collected for each fold 
  when it is used as testing data
- **Pred**: a vector of characters, the results from prediction, collected for each fold when 
  it is used as testing data
- **Summ**: a list, the output of function `predSummary`. A confusion matrix (if `classify=TRUE`) 
  from `confusionMatrix` or RMSE (if `classify=FALSE`) calculated from each 
  fold being predicted.

Author(s)

Shuxia Guo, Thomas Bocklitz, Juergen Popp

References

S. Guo, T. Bocklitz, et al., Common mistakes in cross-validating classification models. Analytical 

See Also

dataSplit

Examples

data(DATA)
### perform batch-wise cross-validation using the function fnPcaLda
RES3 <- crossValidation(data=DATA$spec 
                        ,label=DATA$labels 
                        ,batch=DATA$batch 
                        ,method=fnPcaLda 
                        ,pred=predPcaLda 
                        ,folds=NULL 
                        ,nBatch=0 
                        ,nFold=3 
                        ,verbose=TRUE 
                        ,seed=NULL

### parameters to be passed to fnPcaLda 
                        ,center=TRUE
### perform a two-layer cross-validation using the function tunePcaLda, where the number of principal components used for LDA is optimized (i.e., internal cross-validation).

RES4 <- crossValidation(data=DATA$spec, label=DATA$labels, batch=DATA$batch, method=tunePcaLda, pred=predPcaLda, folds=NULL, nBatch=0, nFold=3, verbose=TRUE, seed=NULL

### parameters to be passed to tunePcaLda

nPC=2:4, cv=c('CV', 'BV')[2], nPart=0, optMerit=c('Accuracy', 'Sensitivity')[2], center=TRUE, scale=FALSE)

---

**DATA**

A Raman spectral data collected from cell lines.

---

**Description**

A Raman spectral data collected from cell lines composed of three cell types: MCF-7 (‘m’), Leukocytes (‘l’) and Erythrocytes (‘r’).

**Usage**

data("DATA")

**Format**

List of 3 elements: $ spec: 29 Raman spectra saved into a matrix, each row corresponding one spectrum. $ labels: a character vector of length 29, giving the cell type of each spectrum. $ batch: a character vector of length 29, giving the cultivation identification of each spectrum.

**References**

Examples

data(DATA)

dataSplit

A procedure to split whole dataset into multiple folds.

Description

The whole dataset is split into multiple folds randomly (batch=NULL) or according to the batch information (batch is specified). The number of folds are defined by nFold in the former case. In the latter case, data belonging to each batch is used as one fold if nBatch=0, otherwise the dataset is split into nBatch folds according to the batch information (i.e., data from the same batch will be used exclusively in one fold).

Usage

dataSplit(ixData, batch = NULL, nBatch = 0, nFold = 10, verbose = TRUE, seed = NULL)

Arguments

ixData a vector of integers, demonstrating the indices of spectra.
batch a vector of sample identifications (e.g., batch/patient ID), must be the same length as ixData. Ideally, this should be the identification of the samples at the highest hierarchy (e.g., the patient ID rather than the spectral ID). If missing, the data is split randomly into nFold folds.
nBatch an integer, the number of data folds in case of batch-wise cross-validation (if nBatch=0, each batch will be used as one fold). Ignored if batch is missing.
nFold an integer, the number of data folds in case of normal k-fold cross-validation. Ignored if batch is given.
verbose a boolean value, if or not to print out the logging info.
seed an integer, if given, will be used as the random seed to split the data in case of k-fold cross-validation. Ignored if batch is given.

Value

a list, of which each element representing the indices of the sample belonging to one fold.

Author(s)

Shuxia Guo, Thomas Bocklitz, Juergen Popp

References

Build a classifier using PCA-LDA

**Description**

A classification function based on PCA following LDA. This function can be cooperated into `crossValidation` by setting parameter `method=fnPcaLda`.

**Usage**

```r
fnPcaLda(data, label, batch = NULL, nPC = 10,
          cv = c("none", "CV", "BV")[1],
          nPart = 10, ...)
```

**Arguments**

- `data`: a data matrix, with samples saved in rows and features in columns.
- `label`: a vector of response variables (i.e., group/concentration info), must be the same length as the number of samples.
- `batch`: a vector of batch variables (i.e., batch/patient ID), must be given in case of `cv='BV'`. Ideally, this should be the identification of the samples at the highest hierarchy (e.g., the patient ID rather than the spectral ID). Ignored for `cv='None'` or `cv='CV'`.
- `nPC`: an integer, the number of principal components to be used in LDA.
- `cv`: a character value, specifying the type of cross-validation.
- `nPart`: an integer, the number of folds to be split for cross-validation. Equivalent to `nFold` of `crossValidation` for `cv='CV'` and to `nBatch` for `cv='BV'`. (NOTE: use `nPart=0` for leave-one-batch out cross-validation). Ignored for `cv='None'`.
- `...`: parameters for `prcomp` (cv='None') or `crossValidation`.

**Details**

Build a classifier based on the given data and return an object including the PCA and LDA models in case of `cv='none'`. Otherwise, a cross-validation is performed if `cv='CV'` or `cv='BV'`, corresponding to normal k-fold or batch-wise cross-validation, respectively. In the latter two cases, the function returns the results of the cross-validation (i.e., the output from `crossValidation`).

**Value**

For `cv='none'`, a list of elements:

- `PCA`: PCA model
- `LDA`: LDA model
- `nPC`: nPC used for modeling

For `cv='CV'` or `cv='BV'`, a list of elements:
Fold | a list, each giving the sample indices of a fold
---|---
True | a vector of characters, groundtruth response variables, collected for each fold when it is used as testing data
Pred | a vector of characters, predicted results, collected for each fold when it is used as testing data
Summ | a list, the output of function `predSummary`. A confusion matrix (if `classify=TRUE`) from `confusionMatrix` or RMSE (if `classify=FALSE`) calculated from each fold being predicted.

**Author(s)**

Shuxia Guo, Thomas Bocklitz, Juergen Popp

**References**


**See Also**

crossValidation, tunePcaLda, lda, prcomp

**Examples**

```r
data(DATA)
### perform classification with a 3-fold cross-validation
RES1 <- fnPcaLda(data=DATA$spec,
                  label=DATA$labels,
                  batch=DATA$batch,
                  nPC=3,
                  cv=c('none', 'CV', 'BV')[2],
                  nPart=3,
                  center=TRUE,
                  scale=FALSE)
```

**predPcaLda**

Predict new instances using the PCA-LDA model built from `tunePcaLda`.

**Description**

Predict new instances given in `newData` using the PCA-LDA model `objModel` built from `tunePcaLda`.

**Usage**

`predPcaLda(objModel, newData)`
Arguments

- `objModel` the classifier built from `tunePcaLda`.
- `newData` data matrix composed of samples to be predicted.

Value

a vector of characters composed of the output of the prediction.

Author(s)

Shuxia Guo, Thomas Bocklitz, Juergen Popp

---

**predSummary**

*Calculate the merit of the prediction*

Description

produce the confusion matrix using function `confusionMatrix` from package caret if `classify=TRUE`, otherwise calculate the RMSE between the predicted and groundtruth values.

Usage

```r
predSummary(reference, prediction,
lev = NULL, classify = TRUE)
```

Arguments

- `reference` groundtruth values.
- `prediction` predicted values.
- `lev` a vector of character, specifying the group names. Ignored if `classify=FALSE`.
- `classify` a boolean value, telling whether a classification or regression task.

Value

If `classify=TRUE`, a list, the output from `confusionMatrix` Otherwise a numeric value, giving the RMSE of the prediction.

Author(s)

Shuxia Guo, Thomas Bocklitz, Juergen Popp

See Also

`confusionMatrix`
tunePcaLda

Build a classifier with parameter tuning.

Description

optimize the number of principal component to be used in LDA based on a cross-validation procedure.

Usage

tunePcaLda(data, label, batch = NULL, nPC = 1:50, optMerit = c("Accuracy", "Sensitivity")[2], maximize = TRUE, cv = c("CV", "BV")[2], nPart = 10, ...)

Arguments

data

a data matrix, with samples saved in rows and features in columns.

label

a vector of response variables (i.e., group/concentration info), must be the same length as the number of samples.

batch

a vector of batch variables (i.e., batch/patient ID), must be given in case of cv='BV'. Ideally, this should be the identification of the samples at the highest hierarchy (e.g., the patient ID rather than the spectral ID). Ignored for cv='CV'.

nPC

a vector of integers, the candidate numbers of principal components to be used for LDA, out of which an optimal value will be selected.

optMerit

a character value, the name of the merit to be optimized. The mean sensitivity will be optimized if optMerit = "Sensitivity".

maximize

a boolean value, if or not maximize the merit.

cv

a character value, specifying the type of cross-validation.

nPart

an integer, the number of folds to be split for cross-validation. Equivalent to nFold of crossValidation for cv='CV' and to nBatch for cv='BV'. (NOTE: use nPart=0 for leave-one-batch out cross-validation).

... parameters for crossValidation

Details

build a classifier using each value in nPC, of which the performance is evaluated with a normal k-fold or batch-wise cross-validation. The optimal number is selected as the one giving the maximal (maximize=TRUE) or minimal (maximize=FALSE) merit.

A two-layer cross-validation can be performed by using tunePcaLda as the method in crossValidation.
### Value
A list of elements:

- **PCA**
- **LDA**
- **nPC**

- **PCA model**
- **LDA model built with the optimal number of principal components**
- **the optimal number of principal components**

### Author(s)
Shuxia Guo, Thomas Bocklitz, Juergen Popp

### References

### See Also
crossValidation, tunePcaLda, lda, prcomp

### Examples
```r
data(DATA)
### perform parameter tuning with a 3-fold cross-validation
RES2 <- tunePcaLda(data=DATA$spec,
                     label=DATA$labels,
                     batch=DATA$batch,
                     nPC=2:4,
                     cv=c('CV', 'BV')[1],
                     nPart=3,
                     optMerit=c('Accuracy','Sensitivity')[2],
                     center=TRUE,
                     scale=FALSE)
```
Index

*Topic **datasets**
  DATA, 5
  confusionMatrix, 4, 8, 9
  crossValidation, 2, 2, 7, 8, 10, 11
  DATA, 5
  dataSplit, 3, 4, 6
  fnPcaLda, 3, 7
  lda, 8, 11
  prcomp, 7, 8, 11
  predPcaLda, 3, 8
  predSummary, 3, 4, 8, 9
  rModeling (rModeling-package), 2
  rModeling-package, 2
  tunePcaLda, 4, 8, 9, 10, 11