Package 'CrossValidate'

July 6, 2017

Version 2.2.2

Date 2016-05-09

Title Classes and Methods for Cross Validation of ``Class Prediction'' with Microarrays or Proteomics

Author Kevin R. Coombes

Maintainer Kevin R. Coombes <krc@silicovore.com>

Depends R (>= 3.0), Modeler

Imports methods, oompaBase (>= 3.0.1)

Suggests Biobase

Description Defines classes and methods to cross validate methods for ``class prediction''.

License Apache License (== 2.0)

LazyLoad yes

biocViews Microarray, Clustering

URL

NeedsCompilation no

R topics documented:

CrossValidate-package *Cross validation of high-throughput prediction algorithms*

Description

The CrossValidate package provides generic tools for performing cross-validation on classification methods in the context of high-throughput data sets such as those produced by gene expression microarrays. In order to use a classifier with this implementation of cross-validation, you must first prepare a pair of functions (one for learning models from training data, and one for making predictions on test data). These functions, along with any required meta-parameters, are used to create an object of the Modeler-class. That object is then passed to the CrossValidate function along with the full training data set. The full data set is then repeatedly split into its own training and test sets; you can specify the fraction to be used for training and the number of iterations. The result is a detailed look at the accuracy, sensitivity, specificity, and positive and negative predictive value of the model, as estimated by cross-validation.

Details

Package:	CrossValidate
Type:	Package
Version:	1.0.1
Date:	2012-05-04
License:	Artistic-2.0
LazyLoad:	yes

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Braga-Neto U, Dougherty ER.
Is cross-validation valid for small-sample microarray classification?
Bioinformatics, 2004; **20**:374–380.

Jiang W, Varma S, Simon R.
Calculating confidence intervals for prediction error in microarray classification using resampling.
Stat Appl Genet Mol Biol. 2008; **7**:Article8.

Fu LM, Youn ES.
Improving reliability of gene selection from microarray functional genomics data.
IEEE Trans Inf Technol Biomed. 2003; **7**:191–6.

Man MZ, Dyson G, Johnson K, Liao B.
Evaluating methods for classifying expression data.
J Biopharm Stat. 2004; **14**:1065–84.

Fu WJ, Carroll RJ, Wang S.
Estimating misclassification error with small samples via bootstrap cross-validation.
Bioinformatics, 2005; **21**:1979–86.

Ancona N, Maglietta R, Piepoli A, D'Addabbo A, Cotugno R, Savino M, Liuni S, Carella M, Pesole G, Perri F.
On the statistical assessment of classifiers using DNA microarray data.
BMC Bioinformatics, 2006; **7**:387.

Lecocke M, Hess K.
An empirical study of univariate and genetic algorithm-based feature selection in binary classification with microarray data.
Cancer Inform, 2007; **2**:313–27.

Lee S.
Mistakes in validating the accuracy of a prediction classifier in high-dimensional but small-sample microarray data.
Stat Methods Med Res, 2008; **17**:635–42.

See Also

The following classification methods have been adapted to work within the general cross-validation framework: K nearest neighbors (`learnKNN`), recursive partitioning and regression trees (`learnRPART`),

---

`balancedSplit` *Split a dataset into training and testing sets, balancing a factor*

Description

When performing cross-validation on a dataset, it often becomes necessary to split the data into training and test sets that are balanced for a certain binary outcome. This function implements such a balanced split.

Usage

```
balancedSplit(fac, size)
```

Arguments

`fac`	A factor that should be balnced between the two subsets.
`size`	A number between 0 and 1 indicating the fraction of the dataset to be used for training.

Details

Stuff should go here

Value

Returns a logical vector with length equal to the length of `fac`. TRUE values designate samples selected for the training set.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

CrossValidate, CrossValidate-class, CrossValidate-package.

Examples

```
nFeatures <- 40
nSamples <- 2*10
dataset <- matrix(rnorm(nSamples*nFeatures), ncol=nSamples)
groups <- factor(rep(c("A", "B"), each=10))
balancedSplit(dataset, groups)
```

CrossValidate *Creating CrossValidate objects*

Description

Given a model classifier and a data set, this function performs cross-valiadtion by repeatedly splitting the data into training and testing subsets in order to estimate the performance of this kind of classifer on new data.

Usage

```
CrossValidate(model, data, status, frac, nLoop, verbose=TRUE)
```

Arguments

model

data

status

frac

nLoop

verbose

Details

The CrossValidate package provides generic tools for performing cross-validation on classificaiton methods in the context of high-throughput data sets such as those produced by gene expression microarrays. In order to use a classifier with this implementaiton of cross-validation, you must first prepare a pair of functions (one for learning models from training data, and one for making predictions on test data). These functions, along with any required meta-parameters, are used to create an object of the Modeler-class. That object is then passed to the CrossValidate function along with the full training data set. The full data set is then repeatedly split into its own training and test sets; you can specify the fraction to be used for training and the number of iterations. The result is a detailed look at the accuracy, sensitivity, specificity, and positive and negative predictive value of the model, as estimated by cross-validation.

Value

An object of the CrossValidate-class.

Author(s)

Kevin R. Coombes <krcoombes@mdanderson.org>

References

See the manual page for the CrossValidate-package for a list of related references.

See Also

See the manual page for the CrossValidate-package for a list of classifiers that have been adapted to work with this cross-validation mechanism.

See CrossValidate-class for a description of the slots in the object created by this function.

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==>  Define data, use random,
##--or do  help(data=index)  for the standard data sets.

## The function is currently defined as
function(model, data, status, frac, nLoop, verbose=TRUE) {
  if (length(status) != ncol(data)) {
    stop("The length of the status vector must match the size of the data set.")
  }

  temp <- balancedSplit(status, frac) # just to compute sizes
  nTrain <- sum(temp)
  nTest <- sum(!temp)

  # allocate space to hold the results
  trainOutcome <- data.frame(matrix(NA, ncol=nLoop, nrow=nTrain))
  validOutcome <- data.frame(matrix(NA, ncol=nLoop, nrow=nTest))
  trainPredict <- data.frame(matrix(NA, ncol=nLoop, nrow=nTrain))
  validPredict <- data.frame(matrix(NA, ncol=nLoop, nrow=nTest))
  extras <- list()

  for (i in 1:nLoop) {
    # show that we are still alive
    if(verbose) print(i)
    # split into training and test
    tr <- balancedSplit(status, frac)
    # record the true status for each split so we can get
    # statistics on the performance later
    trainOutcome[,i] <- status[tr]
    validOutcome[,i] <- status[!tr]
    # train the model
    thisModel <- learn(model, data[,tr], status[tr])
    # record anything interesting about the model
    extras[[i]] <- thisModel@extras
    # save the predictions on the training set
    trainPredict[,i] <- predict(thisModel)
    # now make the predictions using the logistic model
    validPredict[,i] <- predict(thisModel, newdata=data[, !tr])
  }
  new("CrossValidate",
      nIterations=nLoop,
```

```
        trainPercent=frac,
        outcome=status,
        trainOutcome=trainOutcome,
        validOutcome=validOutcome,
        trainPredict=trainPredict,
        validPredict=validPredict,
        extras=extras)
    }
```

`CrossValidate-class` *Class "CrossValidate"*

Description

~~ A concise (1-5 lines) description of what the class is. ~~

Objects from the Class

Objects can be created by calls to the constructor function, `CrossValidate`.

Slots

`nIterations`: Object of class `"numeric"` ~~

`trainPercent`: Object of class `"numeric"` ~~

`outcome`: Object of class `"factor"` ~~

`trainOutcome`: Object of class `"data.frame"` ~~

`trainPredict`: Object of class `"data.frame"` ~~

`validOutcome`: Object of class `"data.frame"` ~~

`validPredict`: Object of class `"data.frame"` ~~

`extras`: Object of class `"list"` ~~

Methods

summary `signature(object = "CrossValidate")`: ...

Author(s)

Kevin R. Coombes <krcoombes@mdanderson.org>

References

See the manual page for the `CrossValidate-package` for a list of related references.

See Also

See `CrossValidate-package` for an overview, and see `CrossValidate` for the constructor function.

Examples

```
showClass("CrossValidate")
```

`CrossValSummary-class` *Class "CrossValSummary"*

Description

~~ A concise (1-5 lines) description of what the class is. ~~

Objects from the Class

Objects are almost always created automatically by applying the `summary` method to an object oif the `CrossValidate-class`.

Slots

`call`: Object of class `"call"` ~~

`parent`: Object of class `"character"` ~~

`trainAcc`: Object of class `"list"` ~~

`validAcc`: Object of class `"list"` ~~

Methods

show `signature(object = "CrossValSummary")`: ...

Author(s)

Kevin R. Coombes <krcoombes@mdanderson.org>

See Also

`CrossValidate-class`

Examples

```
showClass("CrossValSummary")
```

Index