Package 'CrossValidate'

August 4, 2017

Version 2.3.1

Date 2017-08-04

Title Classes and Methods for Cross Validation of ``Class Prediction'' Algorithms

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 various binary classification algorithms used for ``class prediction'' problems.

License Apache License (== 2.0)

LazyLoad yes

biocViews Microarray, Clustering

URL http://oompa.r-forge.r-project.org

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 profiling. 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 `Modeler-package` contains numerous classification methods that have been adapted to work within this general cross-validation framework, including: K nearest neighbors (`learnKNN`), recursive partitioning and regression trees (`learnRPART`), random forests (`learnRF`) neural networks (`learnNNET`), support vector machines (`learnSVM`), compound covariate predictors (`learnCCP`), and the TailRank test (`learnTailRank`).

`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 factor. This function implements such a balanced split.

Usage

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

Arguments

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

Details

This function randomly samples the same fraction of items from each level of a factor to include in a training set. In most cases, this will be a binary factor (and might even be the outcome that one wants to predict). However, the implementation works for factors with an arbitrary number of levels.

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-validation 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, prune=keepAll, verbose=TRUE)
```

Arguments

model	An element of the Modeler class.
data	A matrix containing the data to be used for cross-validation. As with most gene expression data, columns are the independent samples or observations and rows are the measured features.
status	A binary-valued factor with the classes to be predicted.
frac	A number between 0 and 1; the fraction of the data that should be used in each iteration to train the model.
nLoop	An integer; the number of times to split the data into training and test sets.
prune	A function that takes two inoputs, a data matrix and a factor with two levels, and rteturns a logical vector whose length equals the number of rows in the data matrix.
verbose	A logical value; should the cross-validation routine report interim progress.

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

```
dataset <- matrix(rnorm(50*100), nrow=50)
pseudoclass <- factor(rep(c("A", "B"), each=50))
model <- modelerCCP # obviously, other models can be used
numTimes <- 10 # and more is probably better
cv <- CrossValidate(model, dataset, pseudoclass, 0.5, numTimes)
summary(cv)
```

CrossValidate-class *Class "CrossValidate"*

Description

A class that contains the results of internal cross-validation (by multiple splits into training and test sets) of an algorithm that builds a model to predict a binary outcome.

Objects from the Class

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

Slots

`nIterations`: An integer; the number of times the data was split into training and test sets.

`trainPercent`: A number between 0 and 1; the fraction of data used in each training set.

`outcome`: A binary factor containing the true outcome for each sample.

`trainOutcome`: A data frame containing the true outcomes for each member of the training set. The value 'NA' is used for samples that were reserved for testing. Each column is a different split into training and test sets.

`trainPredict`: A data frame containing the predicted outcome from the model for each member of the training set. The value 'NA' is used for samples that were reserved for testing. Each column is a different split into training and test sets.

`validOutcome`: A data frame containing the true outcomes for each member of the test set. The value 'NA' is used for samples that were used for training. Each column is a different split into training and test sets.

`validPredict`: A data frame containing the predicted outcome from the model for each member of the test set. The value 'NA' is used for samples that were used for training. Each column is a different split into training and test sets.

`extras`: A list, whose length equals the number of plsits into trainin and test sets. Each entry contains any "extra" information collected during the fitting of the model; the kinds of items stored here depend on the actual classification algorithm used.

Methods

summary `signature(object = "CrossValidate")`: Produces a summary of the performance of the algorithm on both the trinaing sets and the test sets, in terms of specificity, sensitivity, and positive or negative predictive value. Specifically, this method returns an object of the `CrossValSummary-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 `CrossValidate-package` for an overview, and see `CrossValidate` for the constructor function.

Examples

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

CrossValSummary-class *Class "CrossValSummary"*

Description

Represents the effect of summarizing a CrossValidate-class object by computing the performance of predictions on each split into training and test sets.

Objects from the Class

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

Slots

call: A "call" object recoding how the summary method was invoked.

parent: A character vector containing the name of the CrossValidate object being summarized.

trainAcc: A "list" containing five numeric vectors: the sensitivity, specificity, accuracy, PPV, and NPV on each training data set.

validAcc: A "list" containing five numeric vectors: the sensitivity, specificity, accuracy, PPV, and NPV on each test data set.

Methods

show signature(object = "CrossValSummary"): Summarizes the algorithm performance.

Author(s)

Kevin R. Coombes <krcoombes@mdanderson.org>

See Also

CrossValidate-class

Examples

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

Index