Package 'stepwiseCM'

April 19, 2016

Type Package

Title Stepwise Classification of Cancer Samples using High-dimensional Data Sets

Version 1.16.0

Date 2013-06-19

Author Askar Obulkasim

Depends R (>= 2.14), randomForest, MAclinical, tspair, pamr, snowfall, glmpath, penalized, e1071, Biobase

Maintainer Askar Obulkasim <askar703@gmail.com>

Description Stepwise classification of cancer samples using multiple data sets. This package implements the classification strategy using two heterogeneous data sets without actually combining them. Package uses the data type for which full measurements are available at the first stage, and the data type for which only partial measurements are available at the second stage. For incoming new samples package quantifies how much improvement will be obtained if covariates of new samples for the data types at the second stage are measured. This packages suits for the application where study goal is not only obtain high classification accuracy, but also requires economically cheap classifier.

License GPL (>2)

biocViews Classification, Microarray

NeedsCompilation no

R topics documented:

`stepwiseCM-package` *Stepwise classification of cancer samples using high-dimensional data sets.*

Description

Given two types of data, this package is designed to evaluate the classification performances of two data types independently by user define classification algorithm (s), then explore the sample distributions in the two different data spaces. Based on the exact locations of the test samples in the data space for which measreuments on all samples are available (presume that this data type is easy to obtain or relatively cheap) and the "pseudo" locations in the data space for which only partial measurements are available (presume that this data type is difficult to obtain or relatively expensive compared to former), the reclassification scores (RS) for each test sample is calculated without actually measuring the latter for large portion of samples. RS expresses our belief that a test sample is likely to be correctly classified if its covariates for the latter data types are measured. A large RS denotes, sample benefits more if classify it with latter data type and vice versa.

Details

Package:	stepwiseCM
Type:	Package
Version:	1.7.1
Date:	2013-05-20
License:	GPL (>2)
LazyLoad:	yes

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

Examples

```
data(CNS)
train.cli <- t(CNS$cli[1:40,])
test.cli <- t(CNS$cli[41:60,])
train.gen <- CNS$mrna[,1:40]
test.gen <- CNS$mrna[,41:60]
train.label <- CNS$class[1:40]
```

```
test.label <- CNS$class[41:60]
pred.cli <- Classifier(train = train.cli, train.label = train.label, test = test.cli,
            type = "GLM_L1", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
pred.gen <- Classifier(train = train.gen, train.label = train.label, test = test.gen,
            type = "GLM_L2", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
prox1 <- Proximity(train.cli, train.label, test.cli, N = 2)$prox.test
prox2 <- Proximity(train.gen, train.label, NULL, N = 2)$prox.train
RS <- RS.generator(pred.cli$P.train, pred.gen$P.train, train.label, prox1,
             prox2, type = "rank")
res <- Curve.generator(RS, pred.cli$P.test, pred.gen$P.test, test.label)
names(res)
```

`Classifier` *A function to perform classification task.*

Description

Given the set of samples for training and the type of classification algorithm, this function constructs the classifier using the training set, and predicts the class labels of the test set using the trained classifier.

Usage

```
Classifier(train, test = NULL, train.label, type = c("TSP", "GLM", "GLM_L1",
           "GLM_L2", "PAM", "SVM", "plsrf_x", "plsrf_x_pv", "RF"),
           CVtype = c("loocv", "k-fold"), outerkfold = 5, innerkfold = 5)
```

Arguments

`train`	An object of class `ExpressionSet` or data frame or matrix contains predictors for the training set, where columns correspond to samples and rows to features.
`test`	An object of class `ExpressionSet` or data frame or matrix contains predictors for the test set (optional), where columns correspond to samples and rows to features.
`train.label`	A numeric vector contains the actual class labels (0 or 1) of the training set. NOTE: class labels should be numerical not factor.
`type`	Type of classification algorithms used. Currently 9 well-known algorithm are available for user the choose from. They are: top scoring pair (TSP), logistic regression (GLM), GLM with L1 (lasso) penalty, GLM with L2 (ridge) penalty, prediction analysis for microarray (PAM), support vector machine (SVM), Random Forest combined with partial least square dimension reduction (plsrf_x), Random Forest combined with partial least square dimension reduction plus prevalidation (plsrf_x_pv), Random Forest (RF). NOTE: "TSP", "PAM", "plsrf_x" and "plsrf_x_pv" are exclusively designed for high-dimensional data.
`CVtype`	Cross-validation type to obtain predicted labels of the training set. Must be either k-fold cross-validation (k-fold), or leave-one-out-cross-validation (loocv).

`outerkfold`	Number of cross-validation used in the training phase.
`innerkfold`	Number of cross validation used to estimate the model parameters. E.g. penalty parameter in "GLM_L1".

Value

A list object contains following components:

`P.train`	predicted class labels of the training set.
`P.test`	predicted class labels of the test set if the test set is given.

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

References

Aik Choo Tan and Daniel Q. Naiman and Lei Xu and Raimond L. Winslow and Donald Geman(2005). Simple Decision Rules for Classifying Human Cancers from Gene Expression Profiles(TSP). *Bioinformatics, 21*, 3896-3904.

Anne-Laure Boulesteix and Christine Porzelius and Martin Daumer(2008). Microarray-based Classification and Clinical Predictors: on Combined Classifiers and Additional Predictive Value. *Bioinformatics, 24*, 1698–1706.

See Also

`Classifier.par`

Examples

```
data(CNS)
train <- CNS$mrna[, 1:40]
test <- CNS$mrna[, 41:60]
train.label <- CNS$class[1:40]
Pred <- Classifier(train = train, test = test, train.label = train.label,
        type = "GLM_L1", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
Pred$P.train
Pred$P.test
```

`Classifier.par` *A function to perform classification task by multi-core computation.*

Description

Classification has been embedded inside the parallel processing procedure to speed up the computation for large data set. Instead of performing sequential execution of the outer cross-validation (see below), function performs parallel execution.

Usage

```
Classifier.par(train, test = NULL, train.label, type = c("TSP", "GLM", "GLM_L1",
               "GLM_L2", "PAM", "SVM", "plsrf_x", "plsrf_x_pv", "RF"),
               CVtype = c("loocv", "k-fold"), outerkfold = 5, innerkfold = 5,
               ncpus = 2)
```

Arguments

`train`	An object of class `ExpressionSet` or data frame or matrix contains predictors for the training set, where columns correspond to samples and rows to features.
`test`	An object of class `ExpressionSet` or data frame or matrix contains predictors for the test set (optional), where columns correspond to samples and rows to features.
`train.label`	A numeric vector contains the actual class labels (0 or 1) of the training set. NOTE: values should be numeric not factor.
`type`	Type of classification algorithm used. Currently 9 well-known algorithm are available for user to choose from. They are: top scoring pair (TSP), logistic regression (GLM), GLM with L1 (lasso) penalty, GLM with L2 (ridge) penalty, prediction analysis for microarray (PAM), support vector machine (SVM), Random Forest combined with partial least square dimension reduction (plsrf_x), Random Forest combined with partial least square dimension reduction plus prevalidation (plsrf_x_pv), Random Forest (RF). NOTE: "TSP", "PAM", "plsrf_x" and "plsrf_x_pv" are exclusively for high-dimensional data.
`CVtype`	Cross-validation type used to obtain the predicted labels of the training set. Must be either k-fold cross-validation (k-fold), or leave-one-out-cross-validation (loocv).
`outerkfold`	Number of cross validation used in the training phase.
`innerkfold`	Number of cross validation used to estimate the model parameters. E.g. penalty parameter in "GLM_L1".
`ncpus`	Number of cores assign to the parallel computation.

Value

A list object contains following components:

`P.train`	A numeric vector contains the predicted labels of the training set.
`P.test`	A numeric vector contains the predicted labels of the test set (if test set is given).

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

References

Aik Choo Tan and Daniel Q. Naiman and Lei Xu and Raimond L. Winslow and Donald Geman(2005). Simple Decision Rules for Classifying Human Cancers from Gene Expression Profiles(TSP). *Bioinformatics, 21*, 3896-3904.

Anne-Laure Boulesteix and Christine Porzelius and Martin Daumer(2008). Microarray-based Classification and Clinical Predictors: on Combined Classifiers and Additional Predictive Value. *Bioinformatics, 24*, 1698–1706.

Examples

```
data(CNS)
train <- CNS$mrna[, 1:40]
test <- CNS$mrna[, 41:60]
train.label <- CNS$class[1:40]
## Not run: Pred <- Classifier.par(train = train, test = test, train.label = train.label, type = "GLM_L1",
                        CVtype = "k-fold", outerkfold = 5, innerkfold = 5, ncpus = 5)
## End(Not run)
```

CNS *Central Nervous System (CNS) cancer data set.*

Description

A list object includes the measurement of the gene expression in 7128 genes and 60 samples and the clinical risk factors which are Chang stage (nominal), sex (binary), age (nominal), chemo Cx (binary), chemo VP (binary). 21 patients died (labeled 0) and 39 patients survived (labeled 1) within 24 months after the treatment.

References

Pomeroy, SL. and Tamayo, P. and Gaasenbeek, M. and Sturla, LM. and Angelo, M. and McLaughlin, ME. and Kim, JY. and Goumnerova, LC. and Black, PM. and Lau, C. and Allen, JC. and Zagzag, D. and Olson, JM. and Curran, T. and Wetmore, C. and Biegel, JA. and Poggio, T. and Mukherjee, S. and Rifkin, R. and Califano, A. and Stolovitzky, G. and Louis, DN. and Mesirov, JP. and Lander, ES. and Golub, TR. Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression. *Nature*., 415(6870):436-442.

Examples

```
data(CNS)
names(CNS)
```

`Curve.generator` *A function to generate accuracy curve by passing different portion of samples to the data set used at the second stage.*

Description

Accuracy curve may be used as a reference for choosing the re-classification score (RS) threshold threshold for incoming samples.

Usage

```
Curve.generator(RS, pred1.test, pred2.test,
    test.label, plot.it = TRUE)
```

Arguments

`RS`	A numeric vector contains the re-classification score of the test set.
`pred1.test`	A numeric vector of contains the predicted class labels of the test set from the data set used at the first stage. Should be numeric not factor.
`pred2.test`	A numeric vector of contains the predicted class labels of the test set from the data set used at the second stage. Should be numeric not factor.
`test.label`	A vector of actual class labels (0 or 1) of the test set. Should be numeric not factor.
`plot.it`	If set to "TRUE", this function produces a plot in which Y axis denotes the accuracy and X denotes the percentage of samples passed to the second stage. In order to make this plot, class labels and molecular data for the test set must be given. Default is "TRUE".

Value

A data frame of two columns. The first column denotes the percentage of samples passed to the data used at the second stage, and the second denotes the corresponding accuracy.

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

See Also

`Classifier`, `Classifier.par`, `Proximity`, `RS.generator`

Examples

```
data(CNS)
train.cli <- t(CNS$cli[1:40,])
test.cli <- t(CNS$cli[41:60,])
train.gen <- CNS$mrna[,1:40]
test.gen <- CNS$mrna[,41:60]
train.label <- CNS$class[1:40]
test.label <- CNS$class[41:60]
pred.cli <- Classifier(train = train.cli, train.label = train.label, test = test.cli,
            type = "GLM_L1", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
pred.gen <- Classifier(train = train.gen, train.label = train.label, test = test.gen,
            type = "GLM_L1", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
prox1 <- Proximity(train.cli, train.label, test.cli, N = 2)$prox.test
prox2 <- Proximity(train.gen, train.label, NULL, N = 2)$prox.train
RS <- RS.generator(pred.cli$P.train, pred.gen$P.train, train.label, prox1,
             prox2, type = "rank")
res <- Curve.generator(RS, pred.cli$P.test, pred.gen$P.test, test.label)
```

`Proximity` *A function to calculate the proximity matrix.*

Description

This function computes the proximity matrix by Random Forest algorithm. Proximity values ranges from 0 (least similar) to 1 (perfect match).

Usage

```
Proximity(train, train.label, test = NULL, N = 50,
         Parallel = FALSE, ncpus = 2)
```

Arguments

`train`	An object of class `ExpressionSet` or data frame or matrix contains the predictors for the training set, where columns correspond to samples and rows to features.
`train.label`	A vector of actual class labels (0 or 1) of the training set. Should be numeric not factor.
`test`	An object of class `ExpressionSet` or data frame or matrix of containing predictors for the test set, where columns correspond to samples and rows to features.
`N`	Number of repetition for calculating the proximity matrix, final proximity matrix is average of these repeats. We recommend to set a large number, so that stable proximity matrix will be produced. Default is 50.
`Parallel`	Should proximity calculation use the parallel processing procedure? Default is FALSE.
`ncpus`	Number of acores assign to the parallel computation. Default is 2.

Value

A list object with following components:

prox.train	A square symmetric matrix contains the proximity values of the training set .
prox.test	A rectangular square matrix contains the proximity values between test set (rows) and training set (columns). Only returned when test set is supplied.

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

References

Breiman, L. (2001), *Random Forest, 45*, 5-32.

Examples

```
data(CNS)
train <- t(CNS$cli[1:40,])
test <- t(CNS$cli[41:60,])
train.label <- CNS$class[1:40]
##without parallel processing procedure
Prox <- Proximity(train, train.label, test, N = 2)
##with parallel processing procedure
## Not run: Prox <- Proximity(train, train.label, test,
                N = 50, Parallel = TRUE, ncpus = 10)
## End(Not run)
```

RS.generator *A function to generate the reclassification score.*

Description

This function calculates the reclassification score (RS) for the test set. See details.

Usage

```
RS.generator(pred1.train, pred2.train, train.label, prox1, prox2,
             type = c("rank", "proximity", "both"))
```

Arguments

pred1.train	A numeric vector contains the predicted class labels of the training set from the data set used at the first stage.
pred2.train	A numeric vector contains the predicted class labels of the training set from the data set used at the second stage.
train.label	A vector of actual class labels (0 or 1) of the training set. Should be numerical not factor.
prox1	A rectangular matrix contains the proximity values between the test set (rows) and the training set (columns) obtained from the data set used at the first stage.
prox2	A square matrix contains the proximity values between training set obtained from the data set used at the second stage.
type	Which values are used to construct the reclassification score (RS)? There are three options available: *proximity*, *rank* and *both*. If set to proximity, RS will be calculated directly from the proximity value. If set to rank, RS calculate from the rank of proximity values (more robust). If set to *both*, both of them will be calculated. Default is *rank*.

Details

For each test sample, RS is calculated using the given classification results from two data sets. Algorithm project each test sample onto the first stage data space to observe its neighbourhood and tries to gain some information about the test sample's "pseudo" neighbourhood in the second stage data space with the help of indirect mapping. If algorithm finds that the location of this test samples in the first stage data space are more "safe" (more neighbours are correctly classified) and the location in the second stage data space is surrounded by wrongly classified samples, then it gives this test sample lower RS score and vice versa. After obtaining the RS, user may order them in descending order and classify the top ranked certain portion of samples with the second stage data type.

Value

If the argument "type" set to "rank" ("proximity"), function returns a vector of RS calculated by rank (proximity) based approach. If set to "both" returns a matrix of two columns corresponds to the RS obtained by rank and proximity based approaches.

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

Examples

```
data(CNS)
train.cli <- t(CNS$cli[1:40,])
test.cli <- t(CNS$cli[41:60,])
train.gen <- CNS$mrna[,1:40]
train.label <- CNS$class[1:40]
```

```
pred.cli <- Classifier(train = train.cli, train.label = train.label, type = "GLM_L1",
            CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
pred.gen <- Classifier(train = train.gen, train.label = train.label, type = "GLM_L1",
            CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
prox1 <- Proximity(train.cli, train.label, test.cli, N = 2)$prox.test
prox2 <- Proximity(train.gen, train.label, NULL, N = 2)$prox.train
RS <- RS.generator(pred.cli$P.train, pred.gen$P.train, train.label, prox1,
             prox2, type = "both")
```

`Step.pred` *A function to generate RS cutoff point based the given re-classification percentage.*

Description

Based on the specified percentage, this function finds the RS threshold and recommend which test samples may benefit by classifying with the data set at the second stage.

Usage

```
Step.pred(RS, percent)
```

Arguments

`RS`	A vector of RS.
`percent`	Percentage of samples allow to pass to the second stage data set.

Value

A list object with following components:

`RS.cut`	RS threshold corresponding to the specified re-classification percentage.
`ind`	a vector of binary values. 1 denotes sample is recommend to classify with the data set at the second stage and vice versa.

Author(s)

Askar Obulkasim

Maintainer: Askar Obulkasim <askar703@gmail.com>

Examples

```
data(CNS)
train.cli <- t(CNS$cli[1:40,])
test.cli <- t(CNS$cli[41:60,])
train.gen <- CNS$mrna[,1:40]
test.gen <- CNS$mrna[,41:60]
train.label <- CNS$class[1:40]
```

```
test.label <- CNS$class[41:60]
pred.cli <- Classifier(train = train.cli, train.label = train.label, test = test.cli,
            type = "GLM_L1", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
pred.gen <- Classifier(train = train.gen, train.label = train.label, test = test.gen,
            type = "GLM_L1", CVtype = "k-fold", outerkfold = 2, innerkfold = 2)
prox1 <- Proximity(train.cli, train.label, test.cli, N = 2)$prox.test
prox2 <- Proximity(train.gen, train.label, NULL, N = 2)$prox.train
RS <- RS.generator(pred.cli$P.train, pred.gen$P.train, train.label, prox1,
             prox2, type = "rank")
res <- Step.pred(RS, 30)
```

Index