Package 'hmeasure'

February 26, 2019

Type Package

Title The H-Measure and Other Scalar Classification Performance Metrics

Version 1.0-2

Date 2019-02-26

Author Christoforos Anagnostopoulos

<christoforos.anagnostopoulos06@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Depends R (>= 2.10)

Suggests MASS, class, testthat

Maintainer Christoforos Anagnostopoulos

<christoforos.anagnostopoulos06@imperial.ac.uk>

Description Classification performance metrics that are derived from the ROC curve of a classifier. The package includes the H-measure performance metric as described in <http://link.springer.com/article/10.1007/s10994-009-5119-5>, which computes the minimum total misclassification cost, integrating over any uncertainty about the relative misclassification costs, as per a user-defined prior. It also offers a one-stop-shop for other scalar metrics of performance, including sensitivity, specificity and many others, and also offers plotting tools for ROC curves and related statistics.

License MIT + file LICENSE

URL http://www.hmeasure.net

Repository CRAN

Date/Publication 2019-02-26 15:10:03 UTC

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

`hmeasure-package` *The H-measure and other classification performance metrics*

Description

Computes the ROC curves and several related scalar performance metrics, including the H-measure, the AUC, Sensitivity while holding Specificity fixed, the Area Under the Convex Hull, and others, for several classifiers applied on a given dataset.

Details

Package:	hmeasure
Type:	Package
Version:	1.0
Date:	2012-04-30
License:	GPL (>=2)

The hmeasure package is intended as a complete solution for classification performance. Its main advantage over existing implementations is the inclusion of the H-measure for classification performance (Hand, 2009,2010), which is gradually becoming accepted in the classification literature as a coherent alternative to the AUC. Other advantages include a comprehensive list of performance metrics alongside the H-measure and AUC, plotting routines for Receiver Operating Characteristic analyses of multiple classifiers, and computational optimisation to handle large datasets.

The package contains five main functions. The function misclassCounts() takes as input a set of predicted labels and their respective true labels (presumably obtained from a test dataset), and computes the confusion matrix of misclassification counts (False Positive, False Negatives, etc.) as well as a set of commonly employed scalar summaries thereof (Sensitivity, Specificity, etc.). See help(misclassCounts) for details.

The function HMeasure() extends this functionality significantly by additionally implementing measures that do not require the user to specify a classification threshold, but rather operating on the classification scores themselves (see package vignette for an explanation of the distinction between classification scores and predicted labels). Such aggregate metrics of classification performance include the H-measure, and the AUC, as well as several others. The output of HMeasure is an object of class "hmeasure".

The function plotROC() is a plotting routine for objects of class "hmeasure", with four different options designed to give insights into the differences between the H-measure and the AUC. These include the ROC curves and their convex hulls, as well as kernel density estimators of the scoring distributions. The function relabel() is an auxillary tool which converts class labels into numeric 0s and 1s in accordance to certain conventions. Finally, the summary method for objects of class "hmeasure" report a convenient summary of the most important performance metrics in matrix format.

The package vignette provides a very detailed description of all the metrics computed by the hmeasure package, by way of a tutorial in classification performance using a real example. The most notable feature of the H-measure in particular is the possibility for the user to fix the distribution of relative misclassification severities to a classifier-independent setting on a given problem. In the absence of domain knowledge, we propose using the default prior described in [Hand and Anagnostopoulos, 2012], which takes a balanced view of misclassification costs even when faced with heavily unbalanced datasets. However, the user may freely set a prior belief for the relative cost of the two types of misclassification – see help(HMeasure) for more details.

Author(s)

Christoforos Anagnostopoulos <canagnos@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Maintainer: Christoforos Anagnostopoulos <canagnos@imperial.ac.uk>

References

Hand, D.J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, **77**, 103–123.

Hand, D.J. 2010. Evaluating diagnostic tests: the area under the ROC curve and the balance of errors. *Statistics in Medicine*, **29**, 1502–1510.

Hand, D.J. and Anagnostopoulos, C. 2012. A better Beta for the H measure of classification performance. Preprint, arXiv:1202.2564v1.

See Also

plotROC, summary.hmeasure, misclassCounts, relabel, HMeasure

Examples

```
 # load the data
library(MASS)
library(class)
data(Pima.te)

# split it into training and test
n <- dim(Pima.te)[1]
ntrain <- floor(2*n/3)
ntest <- n-ntrain
pima.train <- Pima.te[seq(1,n,3),]
pima.test <- Pima.te[-seq(1,n,3),]
true.class<-pima.test[,8]
```

```
# train an LDA classifier
pima.lda <- lda(formula=type~., data=pima.train)
out.lda <- predict(pima.lda,newdata=pima.test)

# obtain the predicted labels and classification scores
class.lda <- out.lda$class
scores.lda <- out.lda$posterior[,2]

# compute misclassification counts and related statistics
lda.counts <- misclassCounts(class.lda,true.class)
lda.counts$conf.matrix
print(lda.counts$metrics,digits=3)


# repeat for different value of the classification threshold
lda.counts.T03 <- misclassCounts(scores.lda>0.3,true.class)
lda.counts.T03$conf.matrix
lda.counts.T03$metrics[c('Sens','Spec')]


# train k-NN classifier
class.knn <- knn(train=pima.train[,-8], test=pima.test[,-8],
  cl=pima.train$type, k=9, prob=TRUE, use.all=TRUE)
scores.knn <- attr(class.knn,"prob")
# this is necessary because k-NN by default outputs
# the posterior probability of the winning class
scores.knn[class.knn=="No"] <- 1-scores.knn[class.knn=="No"]

# run the HMeasure function on the data frame of scores
scores <- data.frame(LDA=scores.lda,kNN=scores.knn)
results <- HMeasure(true.class,scores)

# report aggregate metrics
summary(results)
# additionally report threshold-specific metrics
summary(results,show.all=TRUE)


# produce the four different types of available plots
par(mfrow=c(2,2))
plotROC(results,which=1)
plotROC(results,which=2)
plotROC(results,which=3)
plotROC(results,which=4)


# experiment with different classification thresholds
HMeasure(true.class,scores,threshold=0.3)$metrics[c('Sens','Spec')]
HMeasure(true.class,scores,threshold=c(0.3,0.3))$metrics[c('Sens','Spec')]
HMeasure(true.class,scores,threshold=c(0.5,0.3))$metrics[c('Sens','Spec')]

# experiment with fixing the sensitivity (resp. specificity)
summary(HMeasure(true.class,scores,level=c(0.95,0.99)))
```

```
# experiment with non-default severity ratios
results.SR1 <- HMeasure(
  true.class, data.frame(LDA=scores.lda,kNN=scores.knn),severity.ratio=1)
results.SR1$metrics[c('H','KS','ER','FP','FN')]
```

`HMeasure` *Computes the H-measure, AUC, and several other scalar classification performance metrics.*

Description

Computes the H-measure and other scalar classification performance metrics.

Usage

```
HMeasure(true.class, scores, severity.ratio = NA, threshold=0.5, level=0.95)
```

Arguments

`true.class`	a vector/array of true labels – can be either a factor, or in numeric form. It is converted to the right format using function relabel(). Must contain at most two classes, at least one instance of each class, and no missing values.
`scores`	a matrix/vector/data frame of scores, each column corresponding to one classifier. Any missing score in any classifier will cause the respective row (i.e., the respective scores for all classifiers) to be disregarded, and produce a warning.
`severity.ratio`	an optional scalar parameter representing how much more severe misclassifying a class 0 instance is than misclassifying a class 1 instance for the computation of the H-measure and the weighted loss. See Details and/or the package vignette for an explanation of the default value for this parameter.
`threshold`	a vector containing one threshold value per classifier, or a scalar representing a common threshold for all classifiers, for use in performance metrics based on misclassification counts. It is set to 0.5 by default.
`level`	normally a scalar x that is employed in computing Sens.Spec-x (resp. Spec.Sens-x), which represents the value of sensitivity (resp. specificity) when specificity (resp. sensitivity) is held fixed at x%. If the user inputs a vector, both metrics will be computed and reported for each element in the vector.

Details

This the main function of the hmeasure package. It takes as input the vector of true class labels, as well as a matrix or data frame consisting of (column) vectors of scores obtained from deploying each of possibly several classifiers to a given dataset. It computes several scalar performance metrics, including the H-measure [Hand, 2009,2010] and the AUC.

To avoid confusion, class labels are switched to 0s (representing "negatives") and 1s (representing "positives"), according to the conventions of relabel(). It is generally understood that scores are such that class 0 objects tend to receive lower scores than class 1 objects, and, whenever AUC < 0.5, the signs of the scores of that classifier are reversed, as is customary in the literature. Any such switches produce a warning.

The HMeasure function outputs an object of class "hmeasure", with one field named "metrics" that reports several performance metrics in the form of a data frame with one row per classifier, and an attribute named "data" which preserves useful information (such as the empirical scoring distributions) for plotting purposes.

The H-measure naturally requires as input a severity ratio, which represents how much more severe misclassifying a class 0 instance is than misclassifying a class 1 instance. Formally, this determines the mode of the prior over costs that underlies the H-measure (see package vignette or references for more information). We may write SR = c_0/c_1, where c_0 > 0 is the cost of misclassifying a class 0 datapoint as class 1. It is sometimes more convenient to consider instead the normalised cost c = c_0/(c_0 + c_1), so that SR = c/(1-c) where c is in [0,1]. For instance, severity.ratio = 2 implies that a False Positive costs twice as much as a False Negative. By default the severity ratio is set to be reciprocal of relative class frequency, i.e., severity.ratio = pi1/pi0, so that misclassifying the rare class is considered a graver mistake. See Hand 2012 for a more detailed motivation of this default.

The metrics reported can be broken down into two types. The first type consists of metrics that measure the match between a set of predicted labels and the true labels. We obtain these predictions using the scores provided and employing a user-specified threshold (or thresholds, one per classifier), if provided, otherwise a default of 0.5. See help(misclassCounts) for a list of the metrics computed. The second type of measures are aggregate measures of performance that do not rely on the user to specify the threshold, and instead operate directly on the classification scores themselves. In this sense, they are more useful for performance comparisons across classifiers and datasets. The aggregate metrics currently reported include: the Area under the ROC Curve (AUC), the H-measure, the Area under the Convex Hull of the ROC Curve (AUCH), the Gini coefficient, the Kolmogorov-Smirnoff (KS) statistic, the Minimum Weighted Loss (MWL), the Minimum Total Loss (MTL), as well as the Sensitivity at 95% Specificity ("Sens95"), and the Specificity at 95% Sensitivity ("Spec95"). For these latter measures, a 95% level is the default, but alternative or additional values may be specified using the "level" argument.

The package vignette contains a very a detailed explanation of each of the above metrics, and their relationships with each other.

Value

an object of class "hmeasure", implemented as a list with a single field "metrics", and an additional attribute "data"

`stats`	A field containing a data frame of measures, where each row is a classifier
`data`	An attribute implemented as an array of lists, each containing scoring distributions and other computed quantities for each classifier to be used for further analysis, or by the plotting routine.

Author(s)

Christoforos Anagnostopoulos <canagnos@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Maintainer: Christoforos Anagnostopoulos <canagnos@imperial.ac.uk>

References

Hand, D.J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, **77**, 103–123.

Hand, D.J. 2010. Evaluating diagnostic tests: the area under the ROC curve and the balance of errors. *Statistics in Medicine*, **29**, 1502–1510.

Hand, D.J. and Anagnostopoulos, C. 2012. A better Beta for the H measure of classification performance. Preprint, arXiv:1202.2564v1

See Also

plotROC, summary.hmeasure, misclassCounts, relabel

Examples

```
# load the data
require(MASS)
require(class)
data(Pima.te)

# split it into training and test
n <- dim(Pima.te)[1]
ntrain <- floor(2*n/3)
ntest <- n-ntrain
pima.train <- Pima.te[seq(1,n,3),]
pima.test <- Pima.te[-seq(1,n,3),]
true.class<-pima.test[,8]

# train an LDA classifier
pima.lda <- lda(formula=type~., data=pima.train)
out.lda <- predict(pima.lda,newdata=pima.test)

# obtain the predicted labels and classification scores
scores.lda <- out.lda$posterior[,2]

# train k-NN classifier
class.knn <- knn(train=pima.train[,-8], test=pima.test[,-8],
  cl=pima.train$type, k=9, prob=TRUE, use.all=TRUE)
scores.knn <- attr(class.knn,"prob")
# this is necessary because k-NN by default outputs
# the posterior probability of the winning class
scores.knn[class.knn=="No"] <- 1-scores.knn[class.knn=="No"]

# run the HMeasure function on the data frame of scores
scores <- data.frame(LDA=scores.lda,kNN=scores.knn)
results <- HMeasure(true.class,scores)

# report aggregate metrics
summary(results)
# additionally report threshold-specific metrics
```

```
summary(results,show.all=TRUE)


# produce the four different types of available plots
par(mfrow=c(2,2))
plotROC(results,which=1)
plotROC(results,which=2)
plotROC(results,which=3)
plotROC(results,which=4)


# experiment with different classification thresholds
HMeasure(true.class,scores,threshold=0.3)$metrics[c('Sens','Spec')]
HMeasure(true.class,scores,threshold=c(0.3,0.3))$metrics[c('Sens','Spec')]
HMeasure(true.class,scores,threshold=c(0.5,0.3))$metrics[c('Sens','Spec')]

# experiment with fixing the sensitivity (resp. specificity)
summary(HMeasure(true.class,scores,level=c(0.95,0.99)))

# experiment with non-default severity ratios
results.SR1 <- HMeasure(
  true.class, data.frame(LDA=scores.lda,kNN=scores.knn),severity.ratio=1)
results.SR1$metrics[c('H','KS','ER','FP','FN')]
```

`misclassCounts` *Computes misclassification counts and related statistics by contrasting predicted with true class labels.*

Description

Computes a set of classification performance metrics that rely on a set of predicted labels and a set of true labels as input. This is in contrast to the HMeasure function which operates directly on the classification scores.

Usage

```
misclassCounts(predicted.class,true.class)
```

Arguments

`predicted.class`	a vector/array of predicted labels – can be either a factor, or in numeric form
`true.class`	a vector/array of true labels – can be either a factor, or in numeric form

Details

This function computes a set of classification performance metrics that rely on a set of predicted labels and a set of true labels as input. This is in contrast to the HMeasure function which operates directly on the classification scores - see help(HMeasure). All the measures computed here are scalar summaries of the confusion matrix, which consists of the number of True Positives (TPs), False Positives (FPs), True Negatives (TNs), and False Negatives (FNs). The most common such summary is the Error Rate (ER). Additionally the following metrics are reported: the True Positive Rate (TPR) and the False Positive Rate (FPR), Sensitivity (same as TPR) versus Specificity (given by 1-FPR), and yet another such pair is Precision versus Recall (same as Sensitivity). Finally, the F measure and the Youden index are scalar measures that attempt to take a more balanced view of the two different objectives than ER does. The former is given by the harmonic mean of Precision and Recall, and the latter by Sens+Spec-1.

The function misclassCounts is essentially a sub-routine of the HMeasure function. In particular, the latter reports all the above metrics, plus several more. Moreover, whereas misclassCounts can only accept a single array of predicted labels, the HMeasure function can take as input the classification scores of several classifiers simultaneously. See the package vignette for more information.

Value

a list with two fields

`metrics`	A data frame with one row of performance metrics
`conf.matrix`	The confusion matrix

Author(s)

Christoforos Anagnostopoulos <canagnos@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Maintainer: Christoforos Anagnostopoulos <canagnos@imperial.ac.uk>

References

Hand, D.J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, **77**, 103–123.

Hand, D.J. 2010. Evaluating diagnostic tests: the area under the ROC curve and the balance of errors. *Statistics in Medicine*, **29**, 1502–1510.

Hand, D.J. and Anagnostopoulos, C. 2012. A better Beta for the H measure of classification performance. Preprint, arXiv:1202.2564v1

See Also

plotROC, summary.hmeasure, relabel, HMeasure

Examples

```
# load the data
library(MASS)
library(class)
data(Pima.te)
```

```
# split it into training and test
n <- dim(Pima.te)[1]
ntrain <- floor(2*n/3)
ntest <- n-ntrain
pima.train <- Pima.te[seq(1,n,3),]
pima.test <- Pima.te[-seq(1,n,3),]
true.class<-pima.test[,8]

# train an LDA classifier
pima.lda <- lda(formula=type~., data=pima.train)
out.lda <- predict(pima.lda,newdata=pima.test)

# obtain the predicted labels and classification scores
class.lda <- out.lda$class
scores.lda <- out.lda$posterior[,2]

# compute misclassification counts and related statistics
lda.counts <- misclassCounts(class.lda,true.class)
lda.counts$conf.matrix
print(lda.counts$metrics,digits=3)


# repeat for different value of the classification threshold
lda.counts.T03 <- misclassCounts(scores.lda>0.3,true.class)
lda.counts.T03$conf.matrix
lda.counts.T03$metrics[c('Sens','Spec')]
```

`plotROC` *Plotting function illustrating the H-measure of classification performance.*

Description

Plots ROC curves, weight distributions according to the H-measure and the AUC, and smoothed scoring distributions.

Usage

```
plotROC(results, which = 1, bw = "nrd0", cols = c("red",
                "blue", "green", "magenta", "yellow", "forestgreen"),
                greyscale = FALSE, lty = c(1))
```

Arguments

`results`	An object of class "hmeasure", which will be the output of the "HMeasure" function applied on the scores of one or more classifiers on the same test dataset, and the true class labels of the test dataset.

`which`	The type of plot to be produced. Option 1 yields the ROC curves and their convex hulls; option 2 shows the weights employed by the H-measure; option 3 displays the weights over costs that are implicitly employed by the AUC in this example for each classifier; and option 4 yields the smoothed density plots of the scoring distributions per class.
`bw`	The type of bandwidth to be used for the smoothed scoring distribution plot. See help(density) for more detail on possible settings of the "bw" parameter. We employ the same default setting as 'density()', namely "nrd0".
`cols`	A list of colours to be used in order for the ROC and convex ROC hulls for each classifier.
`greyscale`	A flag which if set to TRUE prints the plot on greyscale, with as much differentiation between classifiers as possible, to support black-and-white journal publications.
`lty`	The line type.

Details

This plotting routine is meant to help users perform graphical comparisons of several classifiers, and to illustrate the difference between the H-measure and the AUC as aggregate measures of classification performance. Four types of plots are available:

Option 1 plots the Receiver Operating Characteristic curves of each classifier in a continuous colored line, as well as their respective convex hulls in dotted lines. Additionally the ROC curve of the trivial, random classifier is reported, which is a diagonal line.

Option 2 plots the prior over misclassification costs employed by the H-measure. The mode of this prior, controlled by the severity ratio parameter, is indicated by a vertical line.

Option 3 plots, for each classifier, the prior over misclassification costs implicitly employed by the AUC measure, when this latter is interpreted as an averaging operation over different choices of relative misclassification severity.

Finally, option 4 plots, for each classifier, the (smoothed) empirical probability densities of the score conditional on either class label, i.e., p(s(x) | x = 0), where x is the feature vector and s(x) the score of that feature by a given classifier. Class 0 is plotted using a continuous line, and class 1 using a dotted line.

Note

This plotting function is provided to help illustrate the H-measure, and to serve as a stepping stone in a user's development of further custom plotting functions using the "data" attribute of hmeasure objects.

Author(s)

Christoforos Anagnostopoulos <canagnos@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Maintainer: Christoforos Anagnostopoulos <canagnos@imperial.ac.uk>

References

Hand, D.J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, **77**, 103–123.

Hand, D.J. 2010. Evaluating diagnostic tests: the area under the ROC curve and the balance of errors. *Statistics in Medicine*, **29**, 1502–1510.

Hand, D.J. and Anagnostopoulos, C. 2012. A better Beta for the H measure of classification performance. Preprint, arXiv:1202.2564v1

See Also

summary.hmeasure, misclassCounts, relabel, HMeasure

Examples

```
# load the data
library(MASS)
library(class)
data(Pima.te)

# split it into training and test
n <- dim(Pima.te)[1]
ntrain <- floor(2*n/3)
ntest <- n-ntrain
pima.train <- Pima.te[seq(1,n,3),]
pima.test <- Pima.te[-seq(1,n,3),]
true.class<-pima.test[,8]

# train an LDA classifier
pima.lda <- lda(formula=type~., data=pima.train)
out.lda <- predict(pima.lda,newdata=pima.test)

# obtain the predicted labels and classification scores
scores.lda <- out.lda$posterior[,2]

# train k-NN classifier
class.knn <- knn(train=pima.train[,-8], test=pima.test[,-8],
  cl=pima.train$type, k=9, prob=TRUE, use.all=TRUE)
scores.knn <- attr(class.knn,"prob")
# this is necessary because k-NN by default outputs
# the posterior probability of the winning class
scores.knn[class.knn=="No"] <- 1-scores.knn[class.knn=="No"]

# run the HMeasure function on the data frame of scores
scores <- data.frame(LDA=scores.lda,kNN=scores.knn)
results <- HMeasure(true.class,scores)


# produce the four different types of available plots
par(mfrow=c(2,2))
```

```
plotROC(results,which=1)
plotROC(results,which=2)
plotROC(results,which=3)
plotROC(results,which=4)
```

`relabel` *Converts class labels to the default choice of numeric 0s and 1s*

Description

Converts class labels to the default choice of numeric 0s and 1s according to a specific set of conventions.

Usage

```
relabel(labels)
```

Arguments

`labels`	a vector/array of labels, with two levels

Details

This function turns a vector of class labels into the default format: a numeric array with levels 0 and 1. It accepts numeric, logical, and factor arrays, and performs the conversion in accordance with the following conventions. For a numeric vector with values a and b, the minimum of (a,b) will be mapped to 0 and their maximum to 1. For a logical vector with values TRUE and FALSE, TRUE is mapped to 1 and FALSE to 0. Levels of a factor are interpreted as character strings and then mapped to 0s and 1s in alphabetical order, e.g., "male" is mapped to 1, and "female" to 0, and similarly "true" is mapped to 1 and "false" to 0. This convention being arbitrary, the user must make sure (s)he is happy with the results. For instance, if the labels are "cases/non-cases", the default mapping maps "cases" to 0s, whereas 1s would have been the intuitive choice in this instance. To avoid confusion, a message is produced describing the switch whenever one happens.

The code will complain if the number of distinct labels in the input vector is not 2.

Value

a numeric array of 0s and 1s

Author(s)

Christoforos Anagnostopoulos <canagnos@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Maintainer: Christoforos Anagnostopoulos <canagnos@imperial.ac.uk>

See Also

plotROC, summary.hmeasure, misclassCounts, HMeasure

Examples

```
relabel(c("0","1","1"))
relabel(c("no","yes","yes"))
relabel(c(FALSE,TRUE,TRUE))

# the code complains if the number of classes present in the vector is not 2
try(relabel(c("0","1","2")))
try(relabel(c("1","1","1")))
```

`summary.hmeasure` *Report performance measures using an object of class "hmeasure"*

Description

This function retrieves a convenient numeric summary of the output of the HMeasure function.

Usage

```
## S3 method for class 'hmeasure'
summary(object, show.all, ...)
```

Arguments

`object`	an object of class hmeasure
`show.all`	when this is FALSE only aggregate metrics are reported, whereas when TRUE, threshold-specific metrics are additionally reported – see the package vignette for more details. By default this is set to FALSE.
`...`	additional arguments affecting the summary produced

Details

Objects of class "hmeasure" have a field called "metrics", which reports several performance metrics in the form of a data frame with one row per classifier. Please refer to help(HMeasure) or the package vignette to find out more information about the measures reported. The summary method for hmeasure objects retrieves and prints this field. By default only the most important aggregate metrics are reported. Additionally setting show.all=TRUE will report all available metrics.

Value

The summary method returns the "metrics" field of the original hmeasure object: i.e., a data frame where each row is a classifier, and each column a performance metric.

Author(s)

Christoforos Anagnostopoulos <canagnos@imperial.ac.uk> and David J. Hand <d.j.hand@imperial.ac.uk>

Maintainer: Christoforos Anagnostopoulos <canagnos@imperial.ac.uk>

References

Hand, D.J. 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine Learning*, **77**, 103–123.

Hand, D.J. 2010. Evaluating diagnostic tests: the area under the ROC curve and the balance of errors. *Statistics in Medicine*, **29**, 1502–1510.

Hand, D.J. and Anagnostopoulos, C. 2012. A better Beta for the H measure of classification performance. Preprint, arXiv:1202.2564v1

See Also

plotROC, misclassCounts, relabel, HMeasure

Examples

```
# load the data
library(MASS)
library(class)
data(Pima.te)

# split it into training and test
n <- dim(Pima.te)[1]
ntrain <- floor(2*n/3)
ntest <- n-ntrain
pima.train <- Pima.te[seq(1,n,3),]
pima.test <- Pima.te[-seq(1,n,3),]
true.class<-pima.test[,8]

# train an LDA classifier
pima.lda <- lda(formula=type~., data=pima.train)
out.lda <- predict(pima.lda,newdata=pima.test)

# obtain the predicted labels and classification scores
scores.lda <- out.lda$posterior[,2]

# train k-NN classifier
class.knn <- knn(train=pima.train[,-8], test=pima.test[,-8],
  cl=pima.train$type, k=9, prob=TRUE, use.all=TRUE)
scores.knn <- attr(class.knn,"prob")
# this is necessary because k-NN by default outputs
# the posterior probability of the winning class
scores.knn[class.knn=="No"] <- 1-scores.knn[class.knn=="No"]
```

```
# run the HMeasure function on the data frame of scores
scores <- data.frame(LDA=scores.lda,kNN=scores.knn)
results <- HMeasure(true.class,scores)

# report aggregate metrics
summary(results)
# additionally report threshold-specific metrics
summary(results,show.all=TRUE)

# experiment with fixing the sensitivity (resp. specificity)
summary(HMeasure(true.class,scores,level=c(0.95,0.99)))
```

Index