

Package ‘PerfMeas’

November 8, 2011

Type Package

Title PerfMeas: Performance Measures for ranking and classification tasks

Version 1.0

Date 2011-11-07

Author Giorgio Valentini, Matteo Re -- Universita' degli Studi di Milano

Maintainer Giorgio Valentini<valentini@dsi.unimi.it>

Description Package that implements different performance measures for classification and ranking tasks. AUC, precision at a given recall, F-score for single and multiple classes are available.

License GPL (>= 2)

LazyLoad yes

Depends limma, graph, RBGL

URL <http://homes.dsi.unimi.it/~valenti/SW/PerfMeas>

R topics documented:

PerfMeas-package	2
AUC.measures	2
F.measures	4
get.all.nodes.by.depth	6
precision.recall.curves.plot	7
PXR	8
Index	11

PerfMeas-package	<i>PerfMeas: Performance Measures for ranking and classification tasks</i>
------------------	--

Description

Metrics for ranking and classification tasks: Area Under the ROC Curve (AUC), F-scores, and precision at given recall level are implemented.

Details

Package:	PerfMeas
Type:	Package
Version:	1.0
Date:	2011-11-07
License:	GPL (>= 2)
LazyLoad:	yes

This package implements a set of functions to estimate the AUC, F-score, precision, recall, specificity, accuracy according to the 0/1 loss, and precision at given recall level for ranking and classification problems.

Functions to compute the above measures for single classes or for sets of classes are provided.

Author(s)

Giorgio Valentini and Matteo Re
 DSI, Dipartimento di Scienze dell'Informazione
 Universita' degli Studi di Milano
 <{valentini,re}@dsi.unimi.it>
 Maintainer: *Giorgio Valentini*

AUC.measures	<i>AUC measures</i>
--------------	---------------------

Description

Set of functions to compute the Area Under the ROC Curve (AUC)

Usage

```
AUC.single(pred, labels)
AUC.single.over.classes(target, predicted, g, root = "00")
compute.mean.AUC.single.over.classes(y)
```

Arguments

<code>pred</code>	numeric vector (scores) of the values of the predicted labels
<code>labels</code>	numeric vector of the true labels (0 negative, 1 positive examples)
<code>target</code>	matrix with the target multilabels: rows correspond to examples and columns to classes. <code>target[i,j] = 1</code> if example <i>i</i> belongs to class <i>j</i> , <code>target[i,j] = 0</code> otherwise.
<code>predicted</code>	a numeric matrix with predicted values (scores): rows correspond to examples and columns to classes.
<code>g</code>	a graph of class <i>graphNEL</i> (package graph) of the classes. If <i>g</i> is missing no per.level results are computed
<code>root</code>	the name of the root node (def. "00")
<code>y</code>	a list of lists. The components of the outer list is a list returned from the function <code>AUC.single.over.classes</code>

Details

`AUC.single` computes the AUC for a single class.

`AUC.single.over.classes` computes AUC for a set of classes, including their average values across classes and the average values across the levels of the hierarchy (if any); level 1 classes are at distance 1 from the root, level 2 the second level, till to last level corresponding to the leaves. Note that if the argument *g* is missing no per-level values are computed.

`compute.mean.AUC.single.over.classes` compute means across folds of `AUC.single.over.classes`. It can be used to automatically computed average values (for each class, level, or average across classes) across folds.

Value

`AUC.single` returns a numeric value corresponding to the AUC.

`AUC.single.over.classes` returns a list with three elements:

- `average` the average AUC across classes
- `per.level` a named vector with average AUC for each level of the hierarchy; names correspond to levels
- `per.class` a named vector with AUC for each class; names correspond to classes

`compute.mean.AUC.single.over.classes` returns a list obtained by averaging the results across folds of the input *y*. The components are:

- `average` the average AUC across classes
- `per.level` a named vector with average AUC for each level of the hierarchy; names correspond to levels
- `per.class` a named vector with AUC for each class; names correspond to classes

See Also

[F.measures](#), [PXR](#)

Examples

```
# preparing pseudo.random scores and target-labels for examples: 100 examples
# and 10 classes
Scores <- matrix(runif(1000),nrow=100);
Targets <- matrix(integer(1000),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(1000, sd=0.3),nrow=100);
colnames(Scores) <-colnames(Targets) <- LETTERS[1:10];
# getting scores and labels of class "A"
scores <- Scores[, "A"];
labels <- Targets[, "A"];
# AUC for a single class
AUC.single(scores,labels);
# AUC for the 10 classes
AUC.single.over.classes(Targets, Scores);
```

F.measures

F-measures

Description

Set of functions to compute the F-measure, precision, recall, specificity and 0/1 loss accuracy.

Usage

```
F.measure.single(pred, labels)
F.measure.single.over.classes(target, predicted, g, root = "00")
compute.mean.F.measure.single.over.classes(y)
```

Arguments

pred	vector of the predicted labels. 0 stands for negative and 1 for positive
labels	vector of the true labels. 0 stands for negative and 1 for positive
target	matrix with the target multilabels. 0 stands for negative and 1 for positive. Rows correspond to examples and columns to classes.
predicted	matrix with the predicted multilabels. 0 stands for negative and 1 for positive. Rows correspond to examples and columns to classes.
g	graph of the classes (object of class graphNEL, package graph). If missing, no per level results are computed.
root	the name of the root node (def. "00") of the graph g.
y	a list of lists. The components of the outer list is a list returned from the function F.measure.single.over.classes

Details

`F.measure.single` computes the F.score, precision, recall, specificity and accuracy for a single class.

`F.measure.single.over.classes` computes precision, recall, specificity, accuracy and F-measure for a set of classes. In particular it computes the corresponding average values across classes, the average values across levels of the hierarchy of the classes (if any), and the values of the measures for each class. Note that if there is no hierarchy between classes (represented by the graph `g`), you can miss the `g` parameter and no per-level values are computed.

`compute.mean.F.measure.single.over.classes` computes means across folds of `F.measure.single.over.classes`. This function could be useful in cross-validated or multiple hold-out experimental settings.

Value

`F.measure.single` returns a named numeric vector with six elements:

- `P` precision
- `R` recall (sensitivity)
- `S` specificity
- `F` F measure
- `A` 0/1 loss accuracy
- `npos` number of positive examples

`F.measure.single.over.classes` returns a list with three elements:

- `average` a named vector with the average precision, recall, specificity, F-measure, accuracy and average number of positive examples across classes.
- `per.level` a named matrix with average precision, recall, specificity, F-measure and accuracy for each level of the hierarchy. Named rows correspond to levels, named columns correspond respectively to precision, recall, specificity, F-measure, accuracy and number of positive examples.
- `per.class` a named matrix with precision, recall, specificity, F-measure, accuracy and number of positive examples for each class. Named rows correspond to classes, named columns correspond respectively to precision, recall, specificity, F-measure, accuracy and number of positive examples.

`compute.mean.F.measure.single.over.classes` returns a list obtained by averaging the results across folds of the input `y`. The components are:

- `average` a named vector with the average precision, recall, specificity, F-measure and accuracy across classes across folds.
- `per.level` a named matrix with average precision, recall, specificity, F-measure and accuracy for each level of the hierarchy across folds. Named rows correspond to levels, named columns correspond respectively to precision, recall, specificity, F-measure and accuracy
- `per.class` a named matrix with precision, recall, specificity, F-measure and accuracy for each class across folds. Named rows correspond to classes, named columns correspond respectively to precision, recall, specificity, F-measure and accuracy.

See Also

[AUC.measures](#), [PXR](#)

Examples

```
# preparing pseudo-random predictions and target-labels for examples: 100 examples
# and 10 classes
Scores <- matrix(runif(1000),nrow=100);
Targets <- Pred <- matrix(integer(1000),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(1000, sd=0.3),nrow=100);
Pred[Scores>0.5] <- 1;
colnames(Pred) <- colnames(Targets) <- LETTERS[1:10];
# getting predictions and labels of class "A"
pred <- Pred[, "A"];
labels <- Targets[, "A"];
# F.score and other metrics for a single class
F.measure.single(pred, labels);
# F.score and other metrics for the 10 classes
F.measure.single.over.classes(Targets, Pred);
```

```
get.all.nodes.by.depth
```

Getting nodes by their depth

Description

Grouping classes by level in a given hierarchy.

Usage

```
get.all.nodes.by.depth(g, root = "00")
```

Arguments

<code>g</code>	graph of the classes (object of class <code>graphNEL</code> , package graph).
<code>root</code>	name of the root node (def. 00)

Details

The minimum paths between the “root” and all the other classes/nodes are computed. Levels are numbered from 1 in increasing order by their distance from the “root” class.

Value

a list of the nodes, grouped w.r.t. the distance from the root. The first element of the list corresponds to the nodes at distance 1, the second to nodes at distance 2 and so on.

```
precision.recall.curves.plot
```

Plotting precisions at given recall levels

Description

Function to plot multiple precision recall curves

Usage

```
precision.recall.curves.plot(y, curve.names = 1:length(y), f = "",
  range = seq(from = 0, to = 1, by = 0.1), cex.val = 0.6, height = 9,
  width = 11, col = c("black", "red1", "blue1", "green1", "darkgrey",
    "brown1", "yellow1", "orange1", "red4", "blue4", "green4",
    "lightgrey", "brown4", "yellow4", "orange4"), line.type = 1,
  leg=TRUE, pos=c(range[length(range)-2], y=range[length(range)]))
```

Arguments

y	a list of lists. Each component list is a list returned from <code>precision.at.all.recall.levels</code> that reports precision and recall results at different levels for different methods or tasks
curve.names	names of the compared methods to be reported in the legenda (def: numbers)
f	file name. If is given, an encapsulated postscript file is created, otherwise the output is rendered on a window.
range	numeric vector of the precision/recall values to be represented (def: values between 0 and 1 step 0.1)
cex.val	magnification value for characters (def. 0.6)
height	realstive heighth of the graph (def. 9)
width	relstive width of the graph (def. 11)
col	colors of the lines. 14 different colors are given as default, but any vector of color from <code>colors()</code> (package graphics) can be used. Colors are recycled if <code>length(col) < length(y)</code> .
line.type	type of the line. Any valid vector of integer can be assigned (values between 1 and 6, see <code>lty</code> in <code>par()</code> , package graphics for details). Values are recycled if <code>length(line.type) < length(y)</code> . Def.: 1 (solid lines).
leg	boolean: if TRUE (def.) a legend is depicted.
pos	coordinates of the position of the legend.

Details

It plot multiple curves with a given recall in abscissa and the corresponding precision in ordinate. The curves can differ by color and tpe of line. A legend is automatically constructed.

Value

It outputs a graphics file either on a window or on an encapsuled postscript file.

Examples

```
# preparing pseudo-random predictions and target-labels to plot results of 5 methods
n=500;
Scores <- matrix(runif(n),nrow=100);
Targets <- matrix(integer(n),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(n, sd=0.7),nrow=100);
colnames(Scores) <-colnames(Targets) <- paste("Method", 1:5);
res=list();
for (j in 1:(n/100))
  res=c(res, list(precision.at.all.recall.levels(Scores[,j], Targets[,j])));
precision.recall.curves.plot(res,curve.names=paste("Method", 1:5), pos=c(0.7,0.3));
```

PXR

Precision at a given recall level measures

Description

Set of functions to compute the precision at fixed recall levels.

Usage

```
precision.at.recall.level(scores, labels, rec.level = 0.2)
precision.at.recall.level.over.classes(target, predicted,
                                       g, rec.level = 0.2, root = "00")
precision.at.multiple.recall.level(scores, labels,
                                   rec.levels = seq(from = 0.1, to = 1, by = 0.1))
precision.at.multiple.recall.level.over.classes(target,
                                                predicted, rec.levels = seq(from = 0.1, to = 1, by = 0.1))
precision.at.all.recall.levels(scores, labels)
```

Arguments

scores	vector of the predicted scores in [0,1]
labels	0/1 vector of the true labels
rec.level	rec.level: the desired recall level (def: 0.2)
target	matrix with the target multilabels; rows correspond to examples, columns to classes
predicted	matrix with the predicted multilabels; rows correspond to examples, columns to classes
g	graph of the classes (object of class graphNEL, package graph). If missing, no per level results are computed.

`root` the name of the root node (def. "00") of the graph `g`.
`rec.levels` a vector with the desired recall levels (def. 0.1 to 1 by 0.1 step)

Details

`precision.at.recall.level` computes the precision at a given recall level for a single class.
`precision.at.recall.level.over.classes` computes precision at a given recall level for a set of classes.
`precision.at.multiple.recall.level` computes the precision at multiple levels of recall for a single class.
`precision.at.multiple.recall.level.over.classes` computes the precision at multiple levels of recall for multiple classes.
`precision.at.all.recall.levels` compute the precision at all recall levels for a single class. It returns a pair of precision and recall values by moving a threshold from the lowest to the highest score: a number of precision and recall values equal to the number of available examples is returned.

Value

`precision.at.recall.level` returns the precision at the requested recall
`precision.at.recall.level.over.classes` a list with three elements:

- `average` the average precision at a given recall level across classes.
- `per.level` a named vector with average precision at a given recall level for each level of the hierarchy; names correspond to levels
- `per.class` a named vector with precision at a given recall level for each class. Names correspond to classes

`precision.at.multiple.recall.level` a list with 2 elements:

- `precisions` a vector with the precision at different recall levels
- `f.score` a vector with the f-score at different recall levels

`precision.at.multiple.recall.level.over.classes`

- `PXR` a matrix with the precisions at different recall levels: rows are classes, columns precisions at different recall levels
- `avgPXR` a vector with the the average precisions at different recall levels across classes

`precision.at.all.recall.levels` a list with 3 elements:

- `precision` precision at different thresholds
- `recall` recall at different thresholds
- `f.score` f.score at different thresholds

See Also

[AUC.measures](#), [F.measures](#)

Examples

```
# preparing pseudo-random predictions and target-labels for examples:
# 100 examples and 10 classes
Scores <- matrix(runif(1000),nrow=100);
Targets <- matrix(integer(1000),nrow=100);
Targets[Scores>0.5] <- 1;
# adding noise to scores
Scores <- Scores + matrix(rnorm(1000, sd=0.3),nrow=100);
colnames(Scores) <- colnames(Targets) <- LETTERS[1:10];
# getting scores and labels of class "A"
scores <- Scores[, "A"];
labels <- Targets[, "A"];
# precision at 0.4 recall level for class A
precision.at.recall.level(scores, labels, rec.level=0.4);
# precision at 0.4 recall level for all the 10 classes
precision.at.recall.level.over.classes(Targets, Scores, rec.level=0.4);
# precision at multiple recall levels for class A
levels <- seq(from=0.1, to=1, by=0.1);
precision.at.multiple.recall.level(scores, labels, rec.levels=levels);
# precision at multiple recall levels for all the 10 classes
precision.at.multiple.recall.level.over.classes(Targets, Scores);
# precision, recall and f-score for a single class obtained
# by moving the threshold across the examples
precision.at.all.recall.levels(scores, labels);
```

Index

*Topic **manip**

AUC.measures, [2](#)

F.measures, [4](#)

*Topic **package**

PerfMeas-package, [2](#)

AUC.measures, [2](#), [6](#), [10](#)

AUC.single(AUC.measures), [2](#)

compute.mean.AUC.single.over.classes
(AUC.measures), [2](#)

compute.mean.F.measure.single.over.classes
(F.measures), [4](#)

F.measure.single(F.measures), [4](#)

F.measures, [3](#), [4](#), [10](#)

get.all.nodes.by.depth, [6](#)

PerfMeas(PerfMeas-package), [2](#)

PerfMeas-package, [2](#)

precision.at.all.recall.levels
(PXR), [8](#)

precision.at.multiple.recall.level
(PXR), [8](#)

precision.at.recall.level(PXR), [8](#)

precision.recall.curves.plot, [7](#)

PXR, [3](#), [6](#), [8](#)