Package 'WeightedROC'

February 1, 2020

Maintainer Toby Dylan Hocking <toby.hocking@r-project.org>

Author Toby Dylan Hocking

Version 2020.1.31

License GPL-3

URL https://github.com/tdhock/WeightedROC

BugReports https://github.com/tdhock/WeightedROC/issues

Title Fast, Weighted ROC Curves

Description Fast computation of
Receiver Operating Characteristic (ROC) curves
and Area Under the Curve (AUC)
for weighted binary classification problems
(weights are example-specific cost values).

Suggests ROCR, pROC, PRROC, microbenchmark, glmnet, testthat, ggplot2,
GsymPoint, geometry

NeedsCompilation no

Repository CRAN

Date/Publication 2020-02-01 08:40:02 UTC

R topics documented:

WeightedAUC *WeightedAUC*

Description

Calculate the exact area under the ROC curve.

Usage

```
WeightedAUC(tpr.fpr)
```

Arguments

tpr.fpr	Output of WeightedROC: data.frame with the true positive rate (TPR) and false positive rate (FPR).

Value

Numeric scalar.

Author(s)

Toby Dylan Hocking

Examples

```
library(WeightedROC)
## Compute the AUC for this weighted data set.
y <- c(0, 0, 1, 1, 1)
w <- c(1, 1, 1, 4, 5)
y.hat <- c(1, 2, 3, 1, 1)
tp.fp <- WeightedROC(y.hat, y, w)
(wauc <- WeightedAUC(tp.fp))

## For the un-weighted ROCR example data set, verify that our AUC is
## the same as that of ROCR/pROC.
if(require(microbenchmark) && require(ROCR) && require(pROC)){
  data(ROCR.simple, envir=environment())
  microbenchmark(WeightedROC={
    tp.fp <- with(ROCR.simple, WeightedROC(predictions, labels))
    wroc <- WeightedAUC(tp.fp)
  }, ROCR={
    pred <- with(ROCR.simple, prediction(predictions, labels))
    rocr <- performance(pred, "auc")@y.values[[1]]
  }, pROC={
    proc <- pROC::auc(labels ~ predictions, ROCR.simple, algorithm=2)
  }, times=10)
  rbind(WeightedROC=wroc, ROCR=rocr, pROC=proc) #same
```

```
}

## For the un-weighted pROC example data set, verify that our AUC is
## the same as that of ROCR/pROC.
data(aSAH, envir=environment())
table(aSAH$s100b)
if(require(microbenchmark)){
  microbenchmark(WeightedROC={
    tp.fp <- with(aSAH, WeightedROC(s100b, outcome))
    wroc <- WeightedAUC(tp.fp)
  }, ROCR={
    pred <- with(aSAH, prediction(s100b, outcome))
    rocr <- performance(pred, "auc")@y.values[[1]]
  }, pROC={
    proc <- pROC::auc(outcome ~ s100b, aSAH, algorithm=2)
  }, times=10)
  rbind(WeightedROC=wroc, ROCR=rocr, pROC=proc)
}
```

`WeightedROC` *WeightedROC*

Description

Compute a weighted ROC curve.

Usage

```
WeightedROC(guess, label,
    weight = rep(1, length(label)))
```

Arguments

`guess`	Numeric vector of scores.
`label`	True positive/negative labels. A factor with 2 unique values, or integer/numeric with values all in 0=negative,1=positive or 1=negative,2=positive or -1=negative,1=positive.
`weight`	Positive weights, by default 1.

Value

data.frame with true positive rate (TPR), false positive rate (FPR), weighted false positive count (FP), weighted false negative count (FN), and threshold (smallest `guess` classified as positive).

Author(s)

Toby Dylan Hocking

Examples

```
## WeightedROC can compute ROC curves for data sets with variable
## weights.
library(WeightedROC)
y <- c(-1, -1, 1, 1, 1)
w <- c(1, 1, 1, 4, 5)
y.hat <- c(1, 2, 3, 1, 1)
tp.fp <- WeightedROC(y.hat, y, w)
if(require(ggplot2)){
  gg <- ggplot()+
    geom_path(aes(FPR, TPR), data=tp.fp)+
    coord_equal()
  print(gg)
}else{
  plot(TPR~FPR, tp.fp, type="l")
}

## The FN/FP columns can be used to plot weighted error as a
## function of threshold.
error.fun.list <- list(
  FN=function(df)df$FN,
  FP=function(df)df$FP,
  errors=function(df)with(df, FP+FN)
  )
all.error.list <- list()
for(error.type in names(error.fun.list)){
  error.fun <- error.fun.list[[error.type]]
  all.error.list[[error.type]] <-
    data.frame(tp.fp, error.type, weighted.error=error.fun(tp.fp))
}
all.error <- do.call(rbind, all.error.list)
fp.fn.colors <- c(FP="skyblue",
                  FN="#E41A1C",
                  errors="black")
ggplot()+
  scale_color_manual(values=fp.fn.colors)+
  geom_line(aes(threshold, weighted.error, color=error.type),
            data=all.error)

if(require(microbenchmark) && require(ROCR) && require(pROC)){

  data(ROCR.simple, envir=environment())
  ## Compare speed and plot ROC curves for the ROCR example data set.
  microbenchmark(WeightedROC={
    tp.fp <- with(ROCR.simple, WeightedROC(predictions, labels))
  }, ROCR={
    pred <- with(ROCR.simple, prediction(predictions, labels))
    perf <- performance(pred, "tpr", "fpr")
  }, pROC.1={
    proc <- roc(labels ~ predictions, ROCR.simple, algorithm=1)
  }, pROC.2={
```

```
  proc <- roc(labels ~ predictions, ROCR.simple, algorithm=2)
}, pROC.3={
  proc <- roc(labels ~ predictions, ROCR.simple, algorithm=3)
}, times=10)
perfDF <- function(p){
  data.frame(FPR=p@x.values[[1]], TPR=p@y.values[[1]], package="ROCR")
}
procDF <- function(p){
  data.frame(FPR=1-p$specificities, TPR=p$sensitivities, package="pROC")
}
roc.curves <- rbind(
  data.frame(tp.fp[, c("FPR", "TPR")], package="WeightedROC"),
  perfDF(perf),
  procDF(proc))
ggplot()+
  geom_path(aes(FPR, TPR, color=package, linetype=package),
            data=roc.curves, size=1)+
  coord_equal()

## Compare speed and plot ROC curves for the pROC example data set.
data(aSAH, envir=environment())
microbenchmark(WeightedROC={
  tp.fp <- with(aSAH, WeightedROC(s100b, outcome))
}, ROCR={
  pred <- with(aSAH, prediction(s100b, outcome))
  perf <- performance(pred, "tpr", "fpr")
}, pROC.1={
  proc <- roc(outcome ~ s100b, aSAH, algorithm=1)
}, pROC.2={
  proc <- roc(outcome ~ s100b, aSAH, algorithm=2)
}, pROC.3={
  proc <- roc(outcome ~ s100b, aSAH, algorithm=3)
}, times=10)
roc.curves <- rbind(
  data.frame(tp.fp[, c("FPR", "TPR")], package="WeightedROC"),
  perfDF(perf),
  procDF(proc))
ggplot()+
  geom_path(aes(FPR, TPR, color=package, linetype=package),
            data=roc.curves, size=1)+
  coord_equal()

## Compute a small ROC curve with 1 tie to show the diagonal.
y <- c(-1, -1, 1, 1)
y.hat <- c(1, 2, 3, 1)
microbenchmark(WeightedROC={
  tp.fp <- WeightedROC(y.hat, y)
}, ROCR={
  pred <- prediction(y.hat, y)
  perf <- performance(pred, "tpr", "fpr")
}, pROC.1={
  proc <- roc(y ~ y.hat, algorithm=1)
}, pROC.2={
```

```
  proc <- roc(y ~ y.hat, algorithm=2)
}, pROC.3={
  proc <- roc(y ~ y.hat, algorithm=3)
}, times=10)
roc.curves <- rbind(
  data.frame(tp.fp[, c("FPR", "TPR")], package="WeightedROC"),
  perfDF(perf),
  procDF(proc))
ggplot()+
  geom_path(aes(FPR, TPR, color=package, linetype=package),
            data=roc.curves, size=1)+
  coord_equal()

}
```

Index