Comparing speed of packages for computing ROC curves

Toby Dylan Hocking

November 13, 2019

1 Introduction

The goal of this vignette is to compare how long it takes to compute the ROC curves and AUC using different R packages: WeightedROC, ROCR, pROC.

2 Data

The data set I will analyze is the `spam` data set discussed in the book "The Elements of Statistical Learning" by Hastie, Tibshirani, and Friedman. The book is available to read for free as a downloadable PDF at

`http://statweb.stanford.edu/~tibs/ElemStatLearn/`

The data set is available in the R package ElemStatLearn:

```
> if(require(ElemStatLearn)){
+   data(spam)
+ }else{
+   spam <- unique(data.frame(
+     iris[,1:4],
+     spam=factor(
+       ifelse(iris[, "Species"]=="setosa", "spam", "email"),
+       c("email", "spam"))))
+ }
> is.label <- names(spam) == "spam"
> X <- as.matrix(spam[,!is.label])
> y <- spam[,is.label]
> set.seed(1)
> train.i <- sample(nrow(spam), nrow(spam)/2)
> sets <-
+   list(train=list(features=X[train.i, ], label=y[train.i]),
+        test=list(features=X[-train.i, ], label=y[-train.i]))
> str(sets)

List of 2
 $ train:List of 2
  ..$ features: num [1:2300, 1:57] 0 0 0 0 0 0.14 0 0 0 0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2300] "1017" "2177" "1533" "4567" ...
  .. .. ..$ : chr [1:57] "A.1" "A.2" "A.3" "A.4" ...
  ..$ label   : Factor w/ 2 levels "email","spam": 2 1 2 1 1 2 1 1 1 2 ...
 $ test :List of 2
  ..$ features: num [1:2301, 1:57] 0 0.21 0.06 0 0 0 0.06 0 0 0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2301] "1" "2" "3" "5" ...
  .. .. ..$ : chr [1:57] "A.1" "A.2" "A.3" "A.4" ...
  ..$ label   : Factor w/ 2 levels "email","spam": 2 2 2 2 2 2 2 2 2 2 ...

>
```

I divided the data set into half train, half test. I will fit a model on the training set and see if it works on the test set.

3 Model fitting

Below, I fit an L1-regularized logistic regression model to the spam training set.

```
> test.guess <- if(require(glmnet)){
+   print(system.time({
+     fit <- cv.glmnet(sets$train$features, sets$train$label, family="binomial")
+   }))
+   as.numeric(predict(fit, sets$test$features))
+ }else{
+   rnorm(nrow(sets$test$features))
+ }

   user  system elapsed
   5.84    0.00    5.84

>
```

On my Intel(R) Pentium(R) Dual CPU T2390 @ 1.86GHz, it took about 25 seconds to fit the model.

4 Timing test ROC curve computation

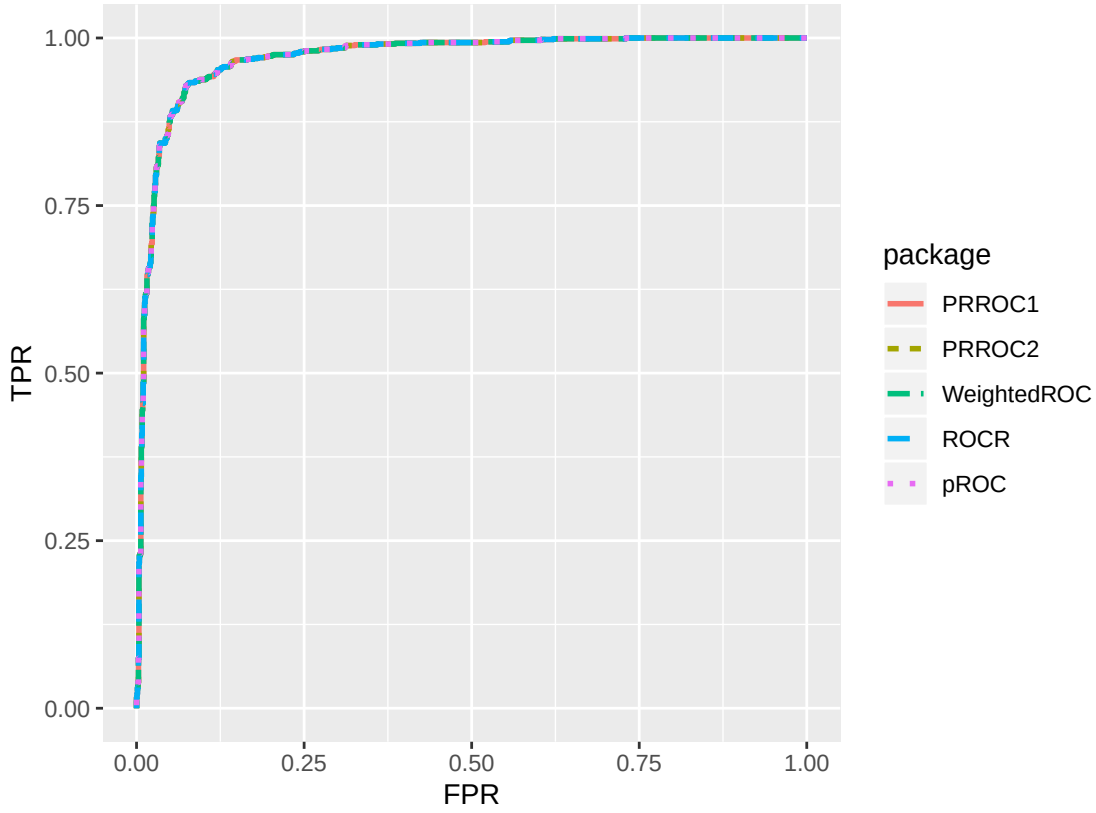
ROC analysis is useful for evaluating binary classifiers. Below we compute the ROC curves using WeightedROC, PRROC, pROC, and ROCR.

```
> bench <- function(guess.vec, label.vec){
+   label.num <- ifelse(label.vec=="spam", 1, 0)
+   df.list <- split(data.frame(guess.vec, label.num), label.vec)
+   time.fun <- if(require(microbenchmark)){
+     microbenchmark
+   }else{
+     function(..., times){
+       L <- match.call()
+       pkg.list <- L[-c(1, length(L))]
+       res.list <- list()
+       for(expr in names(pkg.list)){
+         e <- pkg.list[[expr]]
+         time <- system.time(eval(e))[["elapsed"]]*1e9
+         res.list[[expr]] <- data.frame(expr, time)
+       }
+       do.call(rbind, res.list)
+     }
+   }
+   time.res <- function(..., times){
+     L <- list(...)
+     list(time=time.fun(..., times=times), result=do.call(rbind, L))
+   }
+   time.res(PRROC1={
+     if(require(PRROC)){
+       prroc1 <- roc.curve(
+         df.list$spam$guess.vec, df.list$email$guess.vec,
+         curve=TRUE)
+       data.frame(FPR=prroc1$curve[,1], TPR=prroc1$curve[,2], package="PRROC1")
+     }
+   }, PRROC2={
```

```
+     if(require(PRROC)){
+       prroc2 <- roc.curve(guess.vec, weights.class0=label.num, curve=TRUE)
+       data.frame(FPR=prroc2$curve[,1], TPR=prroc2$curve[,2], package="PRROC2")
+     }
+   }, WeightedROC={
+     if(require(WeightedROC)){
+       wroc <- WeightedROC(guess.vec, label.vec)
+       data.frame(wroc[,c("FPR", "TPR")], package="WeightedROC")
+     }
+   }, ROCR={
+     if(require(ROCR)){
+       pred <- prediction(guess.vec, label.vec)
+       perf <- performance(pred, "tpr", "fpr")
+       data.frame(
+         FPR=perf@x.values[[1]], TPR=perf@y.values[[1]], package="ROCR")
+     }
+   }, pROC={
+     if(require(pROC)){
+       proc <- roc(label.vec, guess.vec, algorithm=2)
+       data.frame(
+         FPR=1-proc$specificities, TPR=proc$sensitivities, package="pROC")
+     }
+   }, times=10)
+ }
> L <- bench(test.guess, sets$test$label)
>
```

It is clear that PRROC is the fastest computation (on my computer, median 3.7 milliseconds), followed by WeightedROC (7.9 milliseconds), then ROCR (9.6 milliseconds), and finally pROC (52.4 milliseconds). However, all of the ROC computations are much faster than the model fitting (10 seconds). Below, we plot the ROC curves to show that they are all the same.

```
> if(require(ggplot2)){
+   rocPlot <- ggplot()+
+     geom_path(aes(FPR, TPR, color=package, linetype=package),
+               data=L$result, size=1)+
+     coord_equal()
+   print(rocPlot)
+ }else{
+   plot(1, main="ggplot2 not found")
+ }
>
```

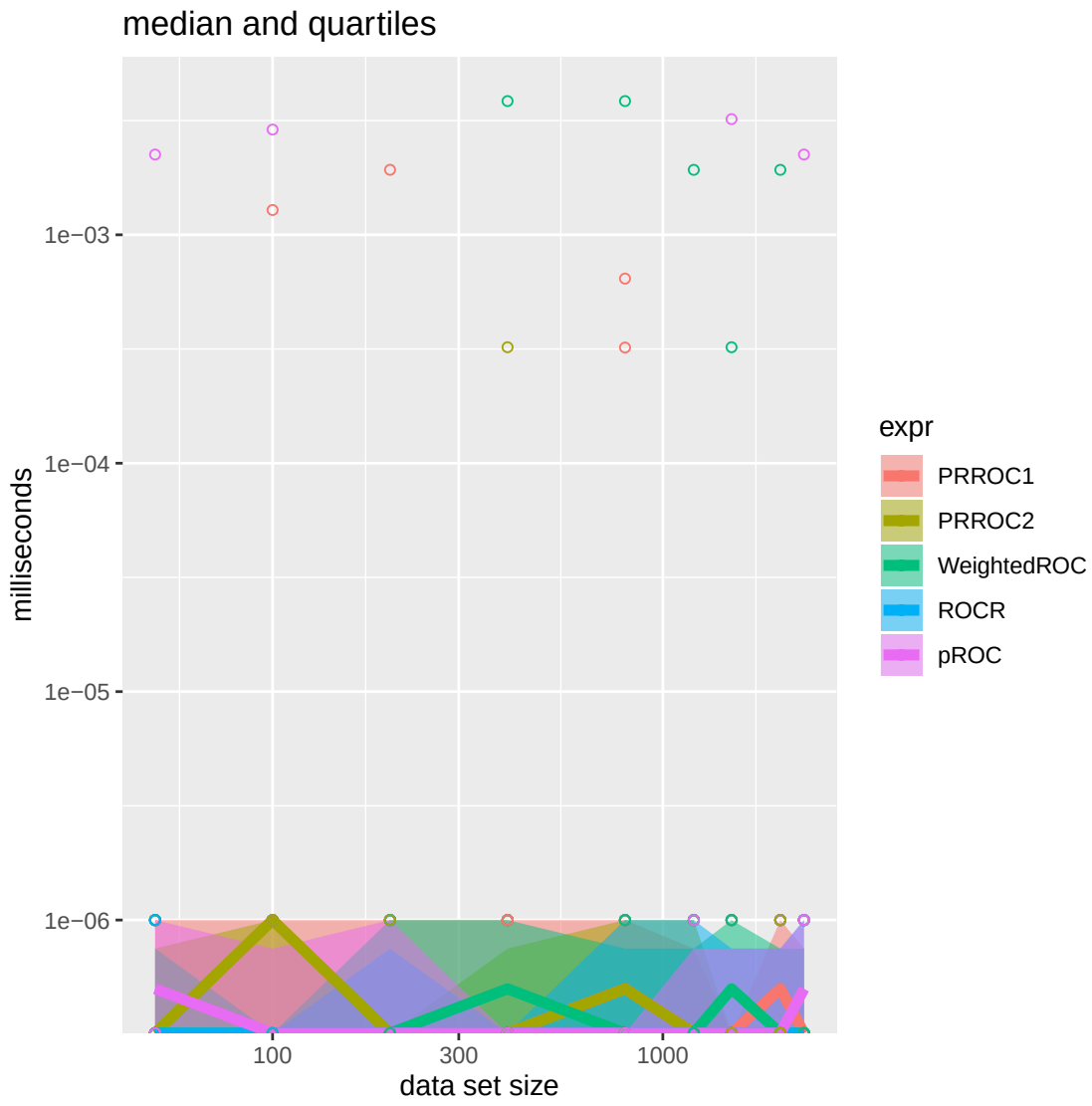


5 Scaling

In this section I was interested in seeing if there are any differences between algorithms as the number of data points changes.

```
> stats.by.size.expr <- list()
> ms.by.size <- list()
> for(size in c(50, 100, 200, 400, 800, 1200, 1500, 2000, 2300)){
+   indices <- as.integer(seq(1, length(sets$test$label), l=size))
+   y <- sets$test$label[indices]
+   y.hat <- test.guess[indices]
+   this.L <- bench(y.hat, y)
+   this.size <- this.L$time
+   this.size$milliseconds <- this.size$time/1e6
+   ms.by.size[[paste(size)]] <- data.frame(size, this.size)
+   this.by.expr <- split(this.size, this.size$expr)
+   for(expr in names(this.by.expr)){
+     stats <- with(this.by.expr[[expr]], {
+       data.frame(median=median(milliseconds),
+                  q25=quantile(milliseconds, 0.25),
```

```
+                   q75=quantile(milliseconds, 0.75))
+     })
+     stats.by.size.expr[[paste(size, expr)]] <- data.frame(size, expr, stats)
+   }
+ }
> ms <- do.call(rbind, ms.by.size)
> algo.stats <- do.call(rbind, stats.by.size.expr)
> if(require(ggplot2)){
+   timePlot <- ggplot()+
+     geom_ribbon(aes(size, ymin=q25, ymax=q75, fill=expr),
+                 data=algo.stats, alpha=1/2)+
+     geom_line(aes(size, median, color=expr), data=algo.stats, size=2)+
+     geom_point(aes(size, milliseconds, color=expr), data=ms, pch=1)+
+     scale_y_log10("milliseconds")+
+     scale_x_log10("data set size")+
+     ggtitle("median and quartiles")
+   print(timePlot)
+ }else{
+   plot(1, main="ggplot2 not available")
+ }
>
```



The figure above shows that for the spam data, the data set size does affect the speed ordering of the other algorithms.

- For large data sets (N > 1000), PRROC1 is fastest, followed by PRROC2, WeightedROC, ROCR, pROC.2, pROC.1. It makes sense that pROC.2 is faster than pROC.1, since pROC.2 uses the cumsum function, but pROC.1 does not.
- For small data sets (N < 500), WeightedROC is fastest, followed by pROC.1, then pROC2, then ROCR.