A demonstration of the nproc package

Yang Feng, Jessica Li and Xin Tong

2016-02-21

We provide a detailed demo of the usage for `nproc` package.

Introduction

Introduction of Neyman-Pearson framework...

Back to Top

Installation and Package Load

Like many other R packages, the simplest way to obtain `nproc` is to install it directly from CRAN. Type the following command in R console:

```
install.packages("nproc", repos = "http://cran.us.r-project.org")
```

Users may change the `repos` options depending on their locations and preferences. Other options such as the directories where to install the packages can be altered in the command. For more details, see `help(install.packages)`.

Here the R package has been downloaded and installed to the default directories.

Alternatively, users can download the package source at http://cran.r-project.org/web/packages/nproc/index.html and type Unix commands to install it to the desired location.

Then we can load the `nproc` package:

```
library(nproc)
```

Back to Top

Neyman-Pearson Classification

Here, we provide a demonstration of Neyman-Pearson Classification with a type-I error control. In the first step, we create a dataset (x,y) from a logistic regression model with 2 features and sample size 1000.

```
n = 1000
set.seed(0)
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
```

Then, the `npc` function can be called to perform the Neyman-Pearson Classification (npc). If one would like to use support vector machine as the classifier, we can set `method` = "svm". The default type I error control is $\alpha = 0.05$. The α value can be changed to any desirable type I error upper bound in $(0, 1)$.

```
fit = npc(x, y, method = "svm", alpha = 0.05)
```

We can now evaluate the prediction performance of the NP classifier on a test set (xtest, ytest) generated as follows.

```
xtest = matrix(rnorm(n*2),n,2)
ctest = 1+3*xtest[,1]
ytest = rbinom(n,1,1/(1+exp(-ctest)))
```

We calculate the overall accuracy of the classifier as well as the realized Type I error. It is shown that the Type I error is smaller than α.

```
pred = predict(fit,xtest)
fit.score = predict(fit,x)
accuracy = mean(pred$pred.label==ytest)
cat("Overall Accuracy: ",  accuracy,'\n')
```

```
## Overall Accuracy:  0.521
```

```
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')
```

```
## Type I error:  0.06544503
```

The classification method implemented in the npc function includes the following options.

- logistic: `glm` function with family = 'binomial'
- penlog: `glmnet` in `glmnet` package
- svm: `svm` in `e1071` package
- randomforest: `randomForest` in `randomForest` package
- lda: `lda` in `MASS` package
- nb: `naiveBayes` in `e1071` package
- ada: `ada` in `ada` package
- custom: a custom classifier

Now, we can try the change the method to logistic regression and change α to 0.1.

```
fit = npc(x, y, method = "logistic", alpha = 0.1)
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat("Overall Accuracy: ",  accuracy,'\n')
```

```
## Overall Accuracy:  0.798
```

```
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')
```

```
## Type I error:  0.07591623
```

The package also implmented a generic classifier with the scores on each observation needed. An example is follows.

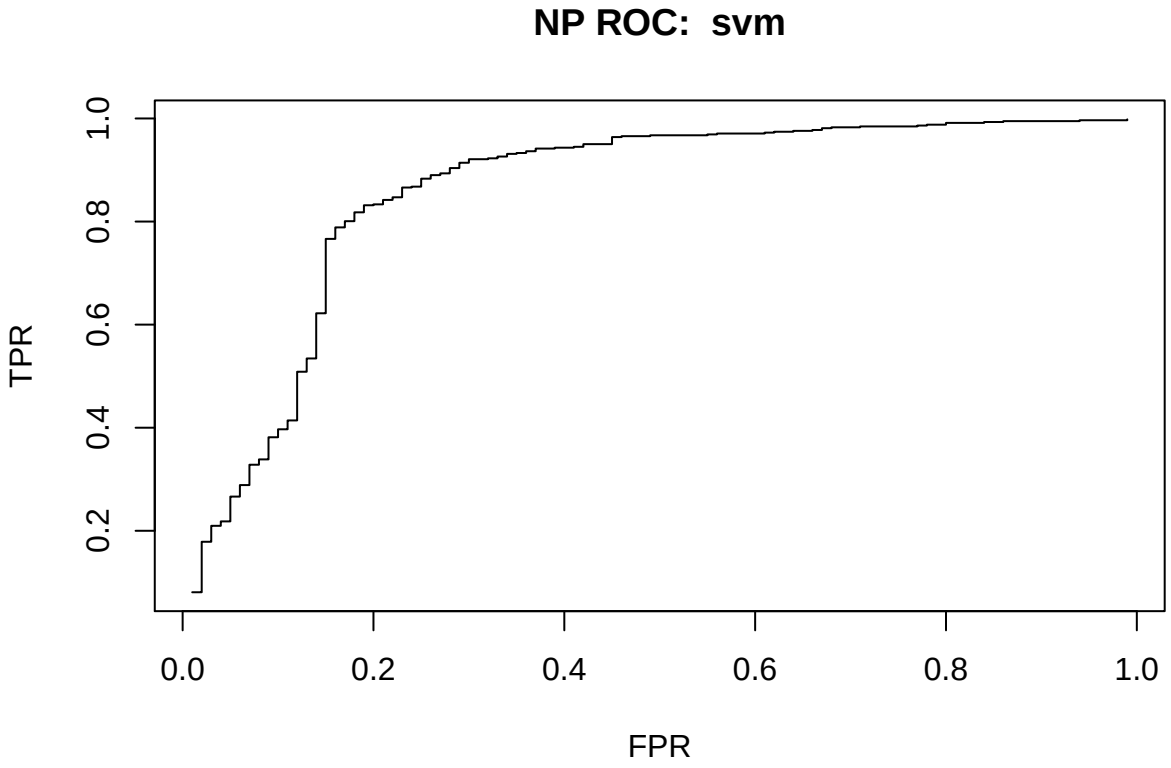
```
fit2 = npc(y = y, score = fit.score$pred.score,
pred.score = pred$pred.score, method = 'custom')
```

Back to Top

Neyman-Pearson Receiver Operator Curve

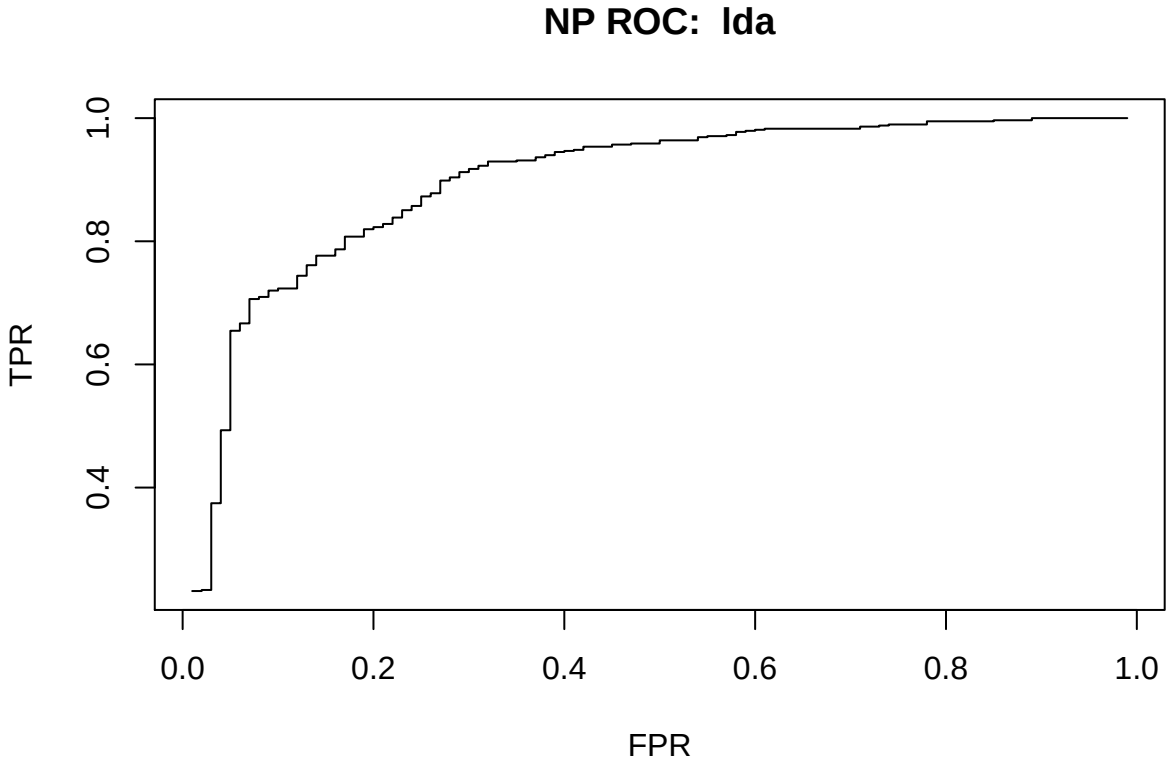
The package provides an implmentation of Neyman-Pearson Receiver Operator Curve (nproc) via the function `nproc`. Here is a brief demo. We use the same data in the NP classifier, i.e., a dataset (x,y) from a logistic regression model with 2 features and sample size 1000. Then, we can call the `nproc` function with a specificed classifier.

```
fit = nproc(x, y, method = "svm")
```



Another example with linear discriminant analysis.

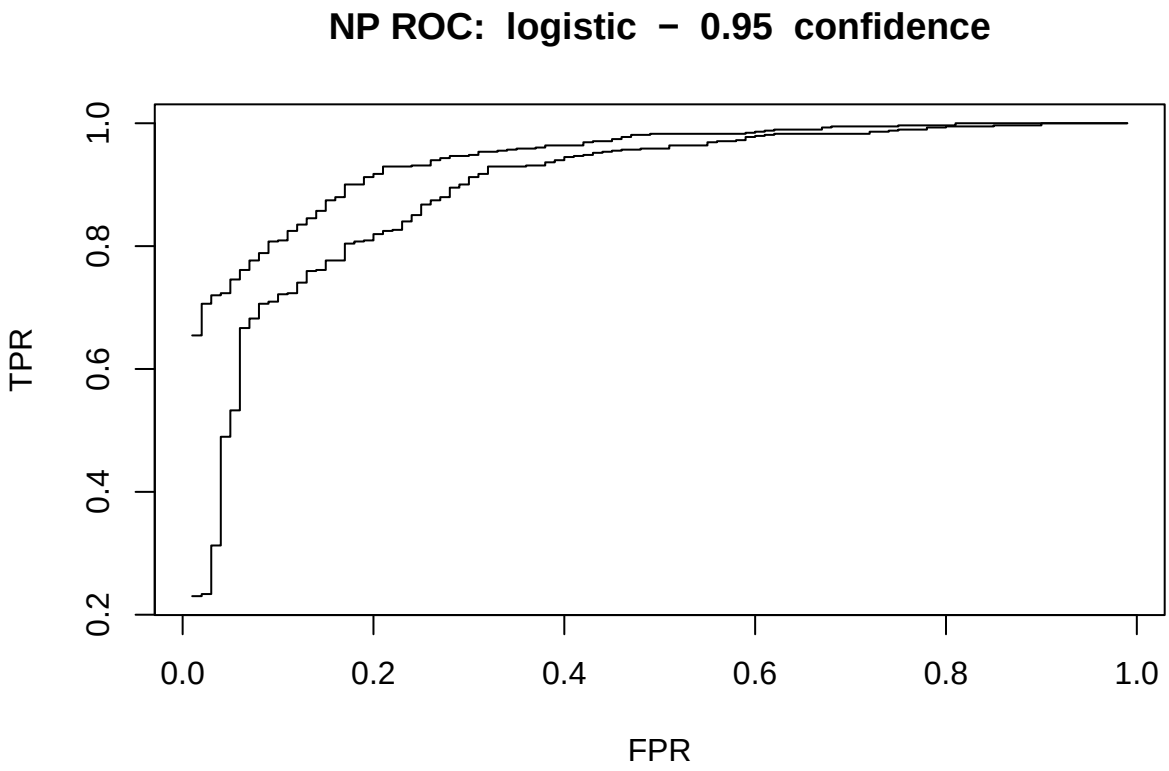
```
fit = nproc(x, y, method = "lda", n.cores = 2)
```



Another important application of the package is to create a confidence band of ROC curves with a given tolerance probability δ. The default $\delta = 0.05$ corresponds to the 95% point-wise confidence interval. Here is

one example.

```
fit = nproc(x, y, method = "logistic", conf = T)
```



Back to Top