A demonstration of the nproc package

Yang Feng, Jessica Li and Xin Tong

2017-03-02

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

Introduction

Installation and Package Load

Neyman-Pearson Classification

Neyman-Pearson Receiver Operator Curve

Introduction

Let \((X, Y)\) be random variables where \(X \in \mathcal{X}\) is a vector of \(d\) features and \(Y \in \{0, 1\}\) represents a binary class label. A data set that contains independent observations \(\{(x_i, y_i)\}_{i=1}^{n+m}\) sampled from the joint distribution of \((X, Y)\) are often divided into training data \(\{(x_i, y_i)\}_{i=1}^{n}\) and test data \(\{(x_i, y_i)\}_{i=n+1}^{n+m}\).

Based on training data, a classifier \(\phi(\cdot)\) is a mapping \(\phi: \mathcal{X} \rightarrow \{0, 1\}\) that returns the predicted class label given \(X\). Classification error occurs when \(\phi(X) \neq Y\), and the binary loss is defined as \(R(\phi) = E[1(\phi(X) \neq Y)] = P(\phi(X) \neq Y)\), which can be expressed as a weighted sum of type I and II errors: \(R(\phi) = P(Y = 0)R_0(\phi) + P(Y = 1)R_1(\phi)\), where \(R_0(\phi) = P(\phi(X) \neq Y | Y = 0)\) denotes the type I error, and \(R_1(\phi) = P(\phi(X) \neq Y | Y = 1)\) denotes the type II error.

The classical classification paradigm aims to find an oracle classifier \(\phi^*\) by minimizing the risk, \(\phi^* = \arg \min_\phi R(\phi)\).

In contrast, the NP classification paradigm aims to mimic the NP oracle classifier \(\phi^*_\alpha\) with respect to a pre-specified type I error upper bound \(\alpha\), \(\phi^*_\alpha = \arg \min_\phi R_0(\phi) \leq \alpha R_1(\phi)\), where \(\alpha\) reflects users’ conservative attitude (priority) towards the type I error.

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:

\[
\text{install.packages("nproc", repos = "http://cran.us.r-project.org")}
\]

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:

\[
\text{library(nproc)}
\]
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.

```r
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)))
```

A visualization of the two classes.

```r
plot(x[y==1,],col=1,xlim=c(-4,4),xlab="x1",ylab="x2")
points(x[y==0,],col=2,pch=2)
legend("topright",legend=c("Class 1","Class 0"),col=1:2,pch=c(1,2))
```

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

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

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

```r
xtest = matrix(rnorm(n*2),n,2)
cetest = 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 \(\alpha\).
pred = predict(fit,xtest)
fitted.score = predict(fit,x)
accuracy = mean(pred$pred.label==ytest)
cat("Overall Accuracy: ", accuracy,\n"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
"
fit = npc(x, y, method = method, alpha = 0.05)
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat(method, ': Overall Accuracy: ', accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat(method, ': Type I error: ', typeI, '\n')
}

## logistic : Overall Accuracy: 0.708
## logistic : Type I error: 0.02216749
## penlog : Overall Accuracy: 0.709
## penlog : Type I error: 0.02216749
## svm : Overall Accuracy: 0.52
## svm : Type I error: 0.05665025
## randomforest : Overall Accuracy: 0.613
## randomforest : Type I error: 0.01477833
## lda : Overall Accuracy: 0.708
## lda : Type I error: 0.02216749
## nb : Overall Accuracy: 0.713
## nb : Type I error: 0.02216749
## ada : Overall Accuracy: 0.731
## ada : Type I error: 0.03448276

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

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

Neyman-Pearson Receiver Operator Curve

The package provides an implementation 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 specified classifier.

fit = nproc(x, y, method = "svm")
plot(fit)
The implementation with linear discriminant analysis is as follows.

```r
fit = nproc(x, y, method = "lda")
plot(fit)
```

Another important usage of the package is to create a point-wise confidence interval of NP-ROC curves with a given tolerance probability `delta`. The default `delta = 0.05` corresponds to the 95% point-wise confidence interval. Here is one example.

```r
fit = nproc(x, y, method = "logistic", band = TRUE)
plot(fit)
```
The package provides comparison ability between two different NP-ROC curves. The following compares the NP-ROC curves generated by logistic regression with the two predictors. The graph shows the regions of the alpha values where one NP-ROC curve is significantly better than the other one.

```r
n = 1000
set.seed(0)
x1 = c(rnorm(n), rnorm(n) + 1)
x2 = c(rnorm(n), rnorm(n)*sqrt(6) + 1)
y = c(rep(0,n), rep(1,n))
fit1 = nproc(x1, y, split = 11, method = 'lda')
fit2 = nproc(x2, y, split = 11, method = 'lda')
v = compare(fit1, fit2)
legend('topleft', legend = c(expression(X[1]),expression(X[2])),col = 1:2,lty = c(1,1))
```