Package: nproc (via r-universe)

September 15, 2024

Type Package

Title Neyman-Pearson (NP) Classification Algorithms and NP Receiver Operating Characteristic (NP-ROC) Curves

Version 2.1.5

Date 2020-01-13

Imports glmnet, e1071, randomForest, naivebayes, MASS, parallel, ada, stats, graphics, ROCR, tree

Description In many binary classification applications, such as disease diagnosis and spam detection, practitioners commonly face the need to limit type I error (i.e., the conditional probability of misclassifying a class 0 observation as class 1) so that it remains below a desired threshold. To address this need, the Neyman-Pearson (NP) classification paradigm is a natural choice; it minimizes type II error (i.e., the conditional probability of misclassifying a class 1 observation as class 0) while enforcing an upper bound, alpha, on the type I error. Although the NP paradigm has a century-long history in hypothesis testing, it has not been well recognized and implemented in classification schemes. Common practices that directly limit the empirical type I error to no more than alpha do not satisfy the type I error control objective because the resulting classifiers are still likely to have type I errors much larger than alpha. As a result, the NP paradigm has not been properly implemented for many classification scenarios in practice. In this work, we develop the first umbrella algorithm that implements the NP paradigm for all scoring-type classification methods, including popular methods such as logistic regression, support vector machines and random forests. Powered by this umbrella algorithm, we propose a novel graphical tool for NP classification methods: NP receiver operating characteristic (NP-ROC) bands, motivated by the popular receiver operating characteristic (ROC) curves. NP-ROC bands will help choose in a data adaptive way and compare different NP classifiers.

License GPL-2

LazyData TRUE

RoxygenNote 6.1.0

URL `http://advances.sciencemag.org/content/4/2/eaao1659`

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Yang Feng [aut, cre], Jessica Li [aut], Xin Tong [aut], Ye Tian [ctb]

Maintainer Yang Feng <`yangfengstat@gmail.com`>

Date/Publication 2020-01-13 19:20:05 UTC

Repository https://yangfengstat.r-universe.dev

RemoteUrl https://github.com/cran/nproc

RemoteRef HEAD

RemoteSha 79ae465ef7b47e6ddd557c8e15dd603cfbd3fd6f

Contents

compare 2
lines.nproc 4
npc 4
nproc 7
plot.nproc 9
predict.npc 9
print.npc 11
print.nproc 11
rocCV 12

Index **14**

`compare` *Compare two NP classification methods at different type I error upper bounds.*

Description

`compare` compares NP classification methods and provides the regions where one method is better than the other.

Usage

```
compare(roc1, roc2, plot = TRUE, col1 = "black", col2 = "red")
```

Arguments

`roc1`	the first nproc object.
`roc2`	the second nproc object.
`plot`	whether to generate the two NP-ROC plots and mark the area of significant difference. Default = 'TRUE'.
`col1`	the color of the region where roc1 is significantly better than roc2. Default = 'black'.
`col2`	the color of the region where roc2 is significantly better than roc1. Default = 'red'.

Value

A list with the following items.

`alpha1`	the alpha values where roc1 is significantly better than roc2.
`alpha2`	the alpha values where roc2 is significantly better than roc1.
`alpha3`	the alpha values where roc1 and roc2 are not significantly different.

References

Xin Tong, Yang Feng, and Jingyi Jessica Li (2018), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), *Science Advances*, **4**, 2, eaao1659.

See Also

`npc`, `nproc`, `predict.npc` and `plot.nproc`

Examples

```
n = 1000
set.seed(1)
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, method = 'lda')
fit2 = nproc(x2, y, method = 'lda')
v = compare(fit1, fit2)
legend('topleft',legend=c('x1','x2'),col=1:2,lty=c(1,1))
```

`lines.nproc` *Add NP-ROC curves to the current plot object.*

Description

Add NP-ROC curves to the current plot object.

Usage

```
## S3 method for class 'nproc'
lines(x, ...)
```

Arguments

`x`	fitted NP-ROC object using `nproc`.
`...`	additional arguments.

See Also

`npc`, `nproc` and `plot.nproc`.

Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = nproc(x, y, method = 'nb')
plot(fit)
fit2 = nproc(x, y, method = 'lda')
lines(fit2, col = 2)
```

`npc` *Construct a Neyman-Pearson Classifier from a sample of class 0 and class 1.*

Description

Given a type I error upper bound alpha and a violation upper bound delta, npc calculates the Neyman-Pearson Classifier which controls the type I error under alpha with probability at least 1-delta.

Usage

```
npc(x = NULL, y, method = c("logistic", "penlog", "svm", "randomforest",
  "lda", "slda", "nb", "nnb", "ada", "tree"), alpha = 0.05, delta = 0.05,
  split = 1, split.ratio = 0.5, n.cores = 1, band = FALSE,
  nfolds = 10, randSeed = 0, warning = TRUE, ...)
```

Arguments

`x`	n * p observation matrix. n observations, p covariates.
`y`	n 0/1 observatons.
`method`	base classification method. • logistic: Logistic regression. glm function with family = 'binomial' • penlog: Penalized logistic regression with LASSO penalty. `glmnet` in `glmnet` package • svm: Support Vector Machines. `svm` in `e1071` package • randomforest: Random Forest. `randomForest` in `randomForest` package • lda: Linear Discriminant Analysis. `lda` in `MASS` package • slda: Sparse Linear Discriminant Analysis with LASSO penalty. • nb: Naive Bayes. `naiveBayes` in `e1071` package • nnb: Nonparametric Naive Bayes. `naive_bayes` in `naivebayes` package • ada: Ada-Boost. `ada` in `ada` package
`alpha`	the desirable upper bound on type I error. Default = 0.05.
`delta`	the violation rate of the type I error. Default = 0.05.
`split`	the number of splits for the class 0 sample. Default = 1. For ensemble version, choose split > 1.
`split.ratio`	the ratio of splits used for the class 0 sample to train the base classifier. The rest are used to estimate the threshold. Can also be set to be "adaptive", which will be determined using a data-driven method implemented in `find.optim.split`. Default = 0.5.
`n.cores`	number of cores used for parallel computing. Default = 1. WARNING: windows machine is not supported.
`band`	whether to generate both lower and upper bounds of type II error. Default = FALSE.
`nfolds`	number of folds for performing adaptive split ratio selection. Default = 10.
`randSeed`	the random seed used in the algorithm.
`warning`	whether to show various warnings in the program. Default = TRUE.
`...`	additional arguments.

Value

An object with S3 class npc.

`fits`	a list of length max(1,split), represents the fit during each split.
`method`	the base classification method.
`split`	the number of splits used.

References

Xin Tong, Yang Feng, and Jingyi Jessica Li (2018), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), *Science Advances*, **4**, 2, eaao1659.

See Also

nproc and predict.npc

Examples

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

##Use lda classifier and the default type I error control with alpha=0.05, delta=0.05
fit = npc(x, y, method = 'lda')
pred = predict(fit,xtest)
fit.score = predict(fit,x)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ',  accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')

## Not run:
##Ensembled lda classifier with split = 11,  alpha=0.05, delta=0.05
fit = npc(x, y, method = 'lda', split = 11)
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ',  accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')

##Now, change the method to logistic regression and change alpha to 0.1
fit = npc(x, y, method = 'logistic', alpha = 0.1)
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ',  accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')

##Now, change the method to adaboost
fit = npc(x, y, method = 'ada', alpha = 0.1)
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ',  accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')
```

```
##Now, try the adaptive splitting ratio
fit = npc(x, y, method = 'ada', alpha = 0.1, split.ratio = 'adaptive')
pred = predict(fit,xtest)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ',  accuracy,'\n')
ind0 = which(ytest==0)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')
cat('Splitting ratio:', fit$split.ratio)

## End(Not run)
```

`nproc` *Calculate the Neyman-Pearson Receiver Operating Characteristics*

Description

`nproc` calculates the Neyman-Pearson Receiver Operating Characteristics band for a given sequence of type I error values.

Usage

```
nproc(x = NULL, y, method = c("logistic", "penlog", "svm", "randomforest",
  "lda", "nb", "nnb", "ada", "tree"), delta = 0.05, split = 1,
  split.ratio = 0.5, n.cores = 1, randSeed = 0, ...)
```

Arguments

`x`	n * p observation matrix. n observations, p covariates.
`y`	n 0/1 observatons.
`method`	base classification method(s). • logistic: Logistic regression. glm function with family = 'binomial' • penlog: Penalized logistic regression with LASSO penalty. `glmnet` in `glmnet` package • svm: Support Vector Machines. `svm` in `e1071` package • randomforest: Random Forest. `randomForest` in `randomForest` package • Linear Discriminant Analysis. lda: `lda` in `MASS` package • nb: Naive Bayes. `naiveBayes` in `e1071` package • nnb: Nonparametric Naive Bayes. `naive_bayes` in `naivebayes` package • ada: Ada-Boost. `ada` in `ada` package
`delta`	the violation rate of the type I error. Default = 0.05.
`split`	the number of splits for the class 0 sample. Default = 1. For ensemble version, choose split > 1.
`split.ratio`	the ratio of splits used for the class 0 sample to train the classifier. Default = 0.5.
`n.cores`	number of cores used for parallel computing. Default = 1.
`randSeed`	the random seed used in the algorithm.
`...`	additional arguments.

Value

An object with S3 class nproc.

`typeI.u`	sequence of upper bound of type I error.
`typeII.l`	sequence of lower bound of type II error.
`typeII.u`	sequence of upper bound of type II error.
`auc.l`	the auc value of the lower NP-ROC curve.
`auc.u`	the auc value of the upper NP-ROC curve.
`method`	the base classification method implemented.
`delta`	the violation rate.

References

Xin Tong, Yang Feng, and Jingyi Jessica Li (2018), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), *Science Advances*, **4**, 2, eaao1659.

See Also

`npc`

Examples

```
n = 200
x = matrix(rnorm(n*2),n,2)
c = 1 - 3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
#fit = nproc(x, y, method = 'svm')
fit2 = nproc(x, y, method = 'penlog')
##Plot the nproc curve
plot(fit2)

## Not run:
fit3 = nproc(x, y, method = 'penlog',  n.cores = 2)
#In practice, replace 2 by the number of cores available 'detectCores()'
fit4 = nproc(x, y, method = 'penlog', n.cores = detectCores())

#Confidence nproc curves
fit6 = nproc(x, y, method = 'lda')
plot(fit6)
nproc ensembled version
fit7 = nproc(x, y, method = 'lda', split = 11)
plot(fit7)

## End(Not run)
```

plot.nproc — *Plot the nproc band(s).*

Description

Plot the nproc band(s).

Usage

```
## S3 method for class 'nproc'
plot(x, ...)
```

Arguments

`x`	fitted nproc object using `nproc`.
`...`	additional arguments.

See Also

`npc`, `nproc`

Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = nproc(x, y, method = 'lda')
plot(fit)
```

predict.npc — *Predicting the outcome of a set of new observations using the fitted npc object.*

Description

Predicting the outcome of a set of new observations using the fitted npc object.

Usage

```
## S3 method for class 'npc'
predict(object, newx = NULL, ...)
```

Arguments

`object`	fitted npc object using `npc`.
`newx`	a set of new observations.
`...`	additional arguments.

Value

A list containing the predicted label and score.

`pred.label`	Predicted label vector.
`pred.score`	Predicted score vector.

See Also

`npc` and `nproc`

Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
xtest = matrix(rnorm(n*2),n,2)
ctest = 1+3*xtest[,1]
ytest = rbinom(n,1,1/(1+exp(-ctest)))


## Not run:
##Use logistic classifier and the default type I error control with alpha=0.05
fit = npc(x, y, method = 'logistic')
pred = predict(fit,xtest)
fit.score = predict(fit,x)
accuracy = mean(pred$pred.label==ytest)
cat('Overall Accuracy: ',  accuracy,'\n')
ind0 = which(ytest==0)
ind1 = which(ytest==1)
typeI = mean(pred$pred.label[ind0]!=ytest[ind0]) #type I error on test set
cat('Type I error: ', typeI, '\n')
typeII = mean(pred$pred.label[ind1]!=ytest[ind1]) #type II error on test set
cat('Type II error: ', typeII, '\n')

## End(Not run)
```

print.npc *Print the npc object.*

Description

Print the npc object.

Usage

```
## S3 method for class 'npc'
print(x, ...)
```

Arguments

x	fitted npc object using npc.
...	additional arguments.

See Also

npc, nproc

Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = npc(x, y, method = 'lda')
print(fit)
```

print.nproc *Print the nproc object.*

Description

Print the nproc object.

Usage

```
## S3 method for class 'nproc'
print(x, ...)
```

Arguments

x	fitted nproc object using nproc.
...	additional arguments.

See Also

npc, nproc

Examples

```
n = 1000
x = matrix(rnorm(n*2),n,2)
c = 1+3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = nproc(x, y, method = 'lda')
print(fit)
```

---

rocCV *Calculate the Receiver Operating Characteristics with Cross-validation or Subsampling*

Description

rocCV calculates the receiver operating characterisitc with cross-validation

Usage

```
rocCV(x = NULL, y, method = c("logistic", "penlog", "svm", "randomforest",
  "lda", "nb", "ada", "tree"), metric = "CV", n.folds = 5,
  train.frac = 0.5, n.cores = 1, randSeed = 0, ...)
```

Arguments

x	n * p observation matrix. n observations, p covariates.
y	n 0/1 observatons.
method	classification method(s). • logistic: Logistic regression. glm function with family = 'binomial' • penlog: Penalized logistic regression with LASSO penalty. glmnet in glmnet package • svm: Support Vector Machines. svm in e1071 package • randomforest: Random Forest. randomForest in randomForest package • Linear Discriminant Analysis. lda: lda in MASS package • nb: Naive Bayes. naiveBayes in e1071 package • ada: Ada-Boost. ada in ada package
metric	metric used for averging performance. Includes 'CV' and 'SS' as options. Default = 'CV'.
n.folds	number of folds used for cross-validation or the number of splits in the subsampling. Default = 5.
train.frac	fraction of training data in the subsampling process. Default = 0.5.
n.cores	number of cores used for parallel computing. Default = 1.
randSeed	the random seed used in the algorithm. Default = 0.
...	additional arguments.

Value

A list.

`fpr`	sequence of false positive rate.
`tpr`	sequence of true positive rate.

References

Xin Tong, Yang Feng, and Jingyi Jessica Li (2018), Neyman-Pearson (NP) classification algorithms and NP receiver operating characteristic (NP-ROC), *Science Advances*, **4**, 2, eaao1659.

See Also

`nproc`

Examples

```
n = 200
x = matrix(rnorm(n*2),n,2)
c = 1 - 3*x[,1]
y = rbinom(n,1,1/(1+exp(-c)))
fit = rocCV(x, y, method = 'svm')
fit2 = rocCV(x, y, method = 'penlog')
fit3 = rocCV(x, y, method = 'penlog', metric = 'SS')
```

Index

ada, *5*, *7*, *12*

compare, 2

glm, *5*, *7*, *12*
glmnet, *5*, *7*, *12*

lda, *5*, *7*, *12*
lines.nproc, 4

naive_bayes, *5*, *7*
naiveBayes, *5*, *7*, *12*
npc, *3*, *4*, 4, *8–12*
nproc, *3*, *4*, *6*, 7, *9–13*

plot.nproc, *3*, *4*, 9
predict.npc, *3*, *6*, 9
print.npc, 11
print.nproc, 11

randomForest, *5*, *7*, *12*
rocCV, 12

svm, *5*, *7*, *12*