Package 'rFerns'

May 9, 2026

Type Package

Title Random Ferns Classifier

Version 6.0.0

Description Provides the random ferns classifier by Ozuysal, Calonder, Lepetit and Fua (2009) <doi:10.1109/TPAMI.2009.23>, modified for generic and multi-label classification and featuring OOB error approximation and importance measure as introduced in Kursa (2014) <doi:10.18637/jss.v061.i10>.

Encoding UTF-8

URL https://gitlab.com/mbq/rFerns

BugReports https://gitlab.com/mbq/rFerns/-/issues

License GPL (>= 2)

RoxygenNote 7.3.3

NeedsCompilation yes

Author Miron Bartosz Kursa [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7672-648X>)

Maintainer Miron Bartosz Kursa <m@mbq.me>

Repository CRAN

Date/Publication 2026-04-23 10:30:02 UTC

Contents

`merge.rFerns` *Merge two random ferns models*

Description

This function combines two compatible (same decision, same training data structure and same depth) models into a single ensemble. It can be used to distribute model training, perform it on batches of data, save checkouts or precisely investigate its course.

Usage

```
## S3 method for class 'rFerns'
merge(
  x,
  y,
  dropModel = FALSE,
  ignoreObjectConsistency = FALSE,
  trueY = NULL,
  ...
)
```

Arguments

`x`	Object of a class `rFerns`; a first model to be merged.
`y`	Object of a class `rFerns`; a second model to be merged. Can also be `NULL`, `x` is immediately returned in that case. Has to have be built on the same kind of training data as `x`, with the same depth.
`dropModel`	If `TRUE`, model structure will be dropped to save size. This disallows prediction using the merged model, but retains importance and OOB approximations.
`ignoreObjectConsistency`	If `TRUE`, merge will be done even if both models were built on a different sets of objects. This drops OOB approximations.
`trueY`	Copy of the training decision, used to re-construct OOB error and confusion matrix. Can be omitted, OOB error and confusion matrix will disappear in that case; ignored when `ignoreObjectConsistency` is `TRUE`.
`...`	Ignored, for S3 gerneric/method consistency.

Value

An object of class `rFerns`, which is a list with the following components:

`model`	The merged model in case both `x` and `y` had model structures included and `dropModel` was `FALSE`. Otherwise `NULL`.
`oobErr`	OOB approximation of accuracy, if can be computed. Namely, when `oobScores` could be and `trueY` is provided.

importance	The merged importance scores in case both x and y had importance calculated. Shadow importance appears only if both models had it enabled.
oobScores	OOB scores, if can be computed; namely if both models had it calculated and ignoreObjectConsistency was not used.
oobPreds	A vector of OOB predictions of class for each object in training set, if can be computed.
oobConfusionMatrix	OOB confusion matrix, if can be computed. Namely, when oobScores could be and trueY is provided.
timeTaken	Time used to train the model, calculated as a sum of training times of x and y.
parameters	Numerical vector of three elements: classes, depth and ferns.
classLabels	Copy of levels(Y) after purging unused levels.
isStruct	Copy of the train set structure.
merged	Set to TRUE to mark that merging was done.

Note

In case of different training object sets were used to build the merged models, merged importance is calculated but mileage may vary; for substantially different sets it may become biased. Your have been warned.

Shadow importance is only merged when both models have shadow importance and the same consistentSeed value; otherwise shadow importance would be biased down.

The order of objects in x and y is not important; the only exception is merging with NULL, in which case x must be an rFerns object for R to use proper merge method.

Examples

```
set.seed(77)
#Fetch Iris data
data(iris)
#Build models
rFerns(Species~.,data=iris)->modelA
rFerns(Species~.,data=iris)->modelB
modelAB<-merge(modelA,modelB)
print(modelA)
print(modelAB)
```

---

naiveWrapper *Naive feature selection method utilising the rFerns shadow imporance*

Description

Proof-of-concept ensemble of rFerns models, built to stabilise and improve selection based on shadow importance. It employs a super-ensemble of iterations small rFerns forests, each built on a subspace of size attributes, which is selected randomly, but with a higher selection probability for attributes claimed important by previous sub-models. Final selection is a group of attributes which hold a substantial weight at the end of the procedure.

Usage

```
naiveWrapper(
  x,
  y,
  iterations = 1000,
  depth = 5,
  ferns = 100,
  size = 30,
  lambda = 5,
  threads = 0,
  saveHistory = FALSE
)
```

Arguments

x	Data frame containing attributes; must have unique names and contain only numeric, integer or (ordered) factor columns. Factors must have less than 31 levels. No NA values are permitted.
y	A decision vector. Must a factor of the same length as nrow(X) for ordinary many-label classification, or a logical matrix with each column corresponding to a class for multi-label classification.
iterations	Number of iterations i.e., the number of sub-models built.
depth	The depth of the ferns; must be in 1–16 range. Note that time and memory requirements scale with 2^depth.
ferns	Number of ferns to be build in each sub-model. This should be a small number, around 3-5 times size.
size	Number of attributes considered by each sub-model.
lambda	Lambda parameter driving the re-weighting step of the method.
threads	Number of parallel threads, copied to the underlying rFerns call.
saveHistory	Should weight history be stored.

Value

An object of class naiveWrapper, which is a list with the following components:

found	Names of all selected attributes.
weights	Vector of weights indicating the confidence that certain feature is relevant.
timeTaken	Time of computation.
weightHistory	History of weights over all iterations, present if saveHistory was TRUE.
params	Copies of algorithm parameters, iterations, depth, ferns and size, as a named vector.

References

Kursa MB (2017). *Efficient all relevant feature selection with random ferns*. In: Kryszkiewicz M., Appice A., Slezak D., Rybinski H., Skowron A., Ras Z. (eds) Foundations of Intelligent Systems. ISMIS 2017. Lecture Notes in Computer Science, vol 10352. Springer, Cham.

Examples

```
set.seed(77)
#Fetch Iris data
data(iris)
#Extend with random noise
noisyIris<-cbind(iris[,-5],apply(iris[,-5],2,sample))
names(noisyIris)[5:8]<-sprintf("Nonsense%d",1:4)
#Execute selection
naiveWrapper(noisyIris,iris$Species,iterations=50,ferns=20,size=8)
```

`predict.rFerns` *Prediction with random ferns model*

Description

This function predicts classes of new objects with given `rFerns` object.

Usage

```
## S3 method for class 'rFerns'
predict(object, x, scores = FALSE, ...)
```

Arguments

`object`	Object of a class `rFerns`; a model that will be used for prediction.
`x`	Data frame containing attributes; must have corresponding names to training set (although order is not important) and do not introduce new factor levels. If this argument is not given, OOB predictions on the training set will be returned.
`scores`	If `TRUE`, the result will contain score matrix instead of simple predictions.
`...`	Additional parameters.

Value

Predictions. If `scores` is `TRUE`, a factor vector (for many-class classification) or a logical data.frame (for multi-class classification) with predictions, else a data.frame with class' scores.

Examples

```
set.seed(77)
#Fetch Iris data
data(iris)
#Split into tRain and tEst set
iris[c(TRUE,FALSE),]->irisR
iris[c(FALSE,TRUE),]->irisE
#Build model
rFerns(Species~.,data=irisR)->model
print(model)
```

```
#Test
predict(model,irisE)->p
print(table(
 Predictions=p,
 True=irisE[["Species"]]))
err<-mean(p!=irisE[["Species"]])
print(paste("Test error",err,sep=" "))

#Show first OOB scores
head(predict(model,scores=TRUE))
```

---

rFerns *Classification with random ferns*

Description

This function builds a random ferns model on the given training data.

Usage

```
rFerns(x, ...)

## S3 method for class 'formula'
rFerns(formula, data = .GlobalEnv, ...)

## S3 method for class 'matrix'
rFerns(x, y, ...)

## Default S3 method:
rFerns(
  x,
  y,
  depth = 5,
  ferns = 1000,
  importance = "none",
  saveForest = TRUE,
  consistentSeed = NULL,
  threads = 0,
  ...
)
```

Arguments

x	Data frame containing attributes; must have unique names and contain only numeric, integer or (ordered) factor columns. Factors must have less than 31 levels. No NA values are permitted.
...	For formula and matrix methods, a place to state parameters to be passed to default method. For the print method, arguments to be passed to print.

formula	alternatively, formula describing model to be analysed.
data	in which to interpret formula.
y	A decision vector. Must a factor of the same length as nrow(X) for ordinary many-label classification, or a logical matrix with each column corresponding to a class for multi-label classification.
depth	The depth of the ferns; must be in 1–16 range. Note that time and memory requirements scale with 2^depth.
ferns	Number of ferns to be build.
importance	Set to calculate attribute importance measure (VIM); "simple" will calculate the default mean decrease of true class score (MDTS, something similar to Random Forest's MDA/MeanDecreaseAccuracy), "shadow" will calculate MDTS and additionally MDTS of this attribute shadow, an implicit feature build by shuffling values within it, thus stripping it from information (which is slightly slower). Shadow importance is useful as a reference to judge significance of a regular importance. "none" turns importance calculation off, for a slightly faster execution. For compatibility with pre-1.2 rFerns, TRUE will resolve to "simple" and FALSE to "none". Abbreviation can be used instead of a full value.
saveForest	Should the model be saved? It must be TRUE if you want to use the model for prediction; however, if you are interested in importance or OOB error only, setting it to FALSE significantly improves memory requirements, especially for large depth and ferns.
consistentSeed	PRNG seed used for shadow importance *only*. Must be either a 2-element integer vector or NULL, which corresponds to seeding from the default PRNG.
threads	Number or OpenMP threads to use. The default value of 0 means all available to OpenMP. It should be set to the same value in two merged models to make shadow importance meaningful.

Value

An object of class rFerns, which is a list with the following components:

model	The built model; NULL if saveForest was FALSE.
oobErr	OOB approximation of accuracy. Ignores never-OOB-tested objects (see oobScores element).
importance	The importance scores or NULL if importance was set to "none". In a first case it is a data.frame with two or three columns: MeanScoreLoss which is a mean decrease of a score of a correct class when a certain attribute is permuted, Tries which is number of ferns which utilised certain attribute, and, only when importance was set to "shadow", Shadow, which is a mean decrease of accuracy for the correct class for a permuted copy of an attribute (useful as a baseline for normal importance). The rownames are set and equal to the names(x).
oobScores	A matrix of OOB scores of each class for each object in training set. Rows correspond to classes in the same order as in levels(Y). If the ferns is too small, some columns may contain NAs, what means that certain objects were never in test set.

`oobPreds` A vector of OOB predictions of class for each object in training set. Never-OOB-tested objects (see above) have predictions equal to `NA`.

`oobConfusionMatrix` Confusion matrix build from `oobPreds` and `y`.

`timeTaken` Time used to train the model (smaller than wall time because data preparation and model final touches are excluded; however it includes the time needed to compute importance, if it applies). An object of `difftime` class.

`parameters` Numerical vector of three elements: `classes`, `depth` and `ferns`, containing respectively the number of classes in decision and copies of `depth` and `ferns` parameters.

`classLabels` Copy of `levels(Y)` after purging unused levels.

`consistentSeed` Consistent seed used; only present for `importance="shadow"`. Can be used to seed a new model via `consistentSeed` argument.

`isStruct` Copy of the train set structure, required internally by predict method.

Note

The unused levels of the decision will be removed; on the other hand unused levels of categorical attributes will be preserved, so that they could be present in the data later predicted with the model. The levels of ordered factors in training and predicted data must be identical.

Do not use formula interface for a data with large number of attributes; the overhead from handling the formula may be significant.

References

Ozuysal M, Calonder M, Lepetit V & Fua P. (2009). *Fast Keypoint Recognition using Random Ferns*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(3), 448-461.

Kursa MB (2014). *rFerns: An Implementation of the Random Ferns Method for General-Purpose Machine Learning*, Journal of Statistical Software, 61(10), 1-13.

Examples

```
set.seed(77)
#Fetch Iris data
data(iris)
#Build model
rFerns(Species~.,data=iris)
##Importance
rFerns(Species~.,data=iris,importance="shadow")->model
print(model$imp)
```

Index