Package 'wsrf'

May 8, 2026

Type Package

Title Weighted Subspace Random Forest for Classification

Version 1.7.32

Date 2026-02-21

Description A parallel implementation of Weighted Subspace Random Forest. The Weighted Subspace Random Forest algorithm was proposed in the International Journal of Data Warehousing and Mining by Baoxun Xu, Joshua Zhexue Huang, Graham Williams, Qiang Wang, and Yunming Ye (2012) <DOI:10.4018/jdwm.2012040103>. The algorithm can classify very high-dimensional data with random forests built using small subspaces. A novel variable weighting method is used for variable subspace selection in place of the traditional random variable sampling.This new approach is particularly useful in building models from high-dimensional data.

License GPL (>= 3)

URL https://github.com/SimonYansenZhao/wsrf, https://togaware.com

BugReports https://github.com/SimonYansenZhao/wsrf/issues

Depends parallel, R (>= 3.3.0), Rcpp (>= 0.10.2), stats

LinkingTo Rcpp

Suggests knitr (>= 1.5), randomForest (>= 4.6.7), stringr (>= 0.6.2), rmarkdown (>= 1.6)

VignetteBuilder knitr

NeedsCompilation yes

Classification/ACM-2012 Computing methodologies ~ Classification and regression trees, Computing methodologies ~ Supervised learning by classification, Computing methodologies ~ Massively parallel and high-performance simulations, Computing methodologies ~ Distributed simulation

Author Qinghan Meng [aut], He Zhao [aut, cre] (ORCID: <https://orcid.org/0000-0001-5763-9743>), Graham J. Williams [aut] (ORCID:

<https://orcid.org/0000-0001-7041-4127>),
Junchao Lv [aut],
Baoxun Xu [aut],
Joshua Zhexue Huang [aut] (ORCID:
<https://orcid.org/0000-0002-6797-2571>)

Maintainer He Zhao <Simon.Yansen.Zhao@gmail.com>

Repository CRAN

Date/Publication 2026-02-22 15:30:02 UTC

Contents

`combine.wsrf` *Combine Ensembles of Trees*

Description

Combine two more more ensembles of trees into one.

Usage

```
combine(...)
```

Arguments

`...`	two or more objects of class `randomForest`, to be combined into one.

Value

An object of class `wsrf`.

See Also

`subset`

Examples

```
library("wsrf")

# Prepare parameters.
ds     <- iris
target <- "Species"
vars   <- names(ds)
if (sum(is.na(ds[vars]))) ds[vars] <- randomForest::na.roughfix(ds[vars])
ds[target] <- as.factor(ds[[target]])
form <- as.formula(paste(target, "~ ."))
set.seed(42)
train.1 <- sample(nrow(ds), 0.7*nrow(ds))
test.1  <- setdiff(seq_len(nrow(ds)), train.1)

set.seed(49)
train.2 <- sample(nrow(ds), 0.7*nrow(ds))
test.2  <- setdiff(seq_len(nrow(ds)), train.2)

# Build model.  We disable parallelism here, since CRAN Repository
# Policy (https://cran.r-project.org/web/packages/policies.html)
# limits the usage of multiple cores to save the limited resource of
# the check farm.

model.wsrf.1 <- wsrf(form, data=ds[train.1, vars], parallel=FALSE)
model.wsrf.2 <- wsrf(form, data=ds[train.2, vars], parallel=FALSE)


# Merge two models.
model.wsrf.big <- combine.wsrf(model.wsrf.1, model.wsrf.2)
print(model.wsrf.big)
cl <- predict(model.wsrf.big, newdata=ds[test.1, vars], type="response")$response
actual <- ds[test.1, target]
(accuracy.wsrf <- mean(cl==actual))
```

`correlation.wsrf` *Correlation*

Description

Give the measure for the diversity of the trees in the forest model built from `wsrf`.

Usage

```
## S3 method for class 'wsrf'
correlation(object, ...)
```

Arguments

`object`	object of class `wsrf`.
`...`	optional additional arguments. At present no additional arguments are used.

Details

The measure was introduced in Breiman (2001).

Value

A numeric value.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

References

Breiman, L. 2001 "Random forests". *Machine learning*, **45(1)**, 5–32.

See Also

wsrf

---

importance.wsrf *Extract Variable Importance Measure*

Description

This is the extractor function for variable importance measures as produced by wsrf.

Usage

```
## S3 method for class 'wsrf'
importance(x, type=NULL, class=NULL, scale=TRUE, ...)
```

Arguments

x	an object of class wsrf.
type	either 1 or 2, specifying the type of importance measure (1=mean decrease in accuracy, 2=mean decrease in node impurity).
class	for classification problem, which class-specific measure to return.
scale	for permutation based measures, should the measures be divided their "standard errors"?
...	not used.

Details

Here are the definitions of the variable importance measures. The first measure is computed from permuting OOB data: For each tree, the prediction error on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences.

The second measure is the total decrease in node impurities from splitting on the variable, averaged over all trees. The node impurity is measured by the Information Gain Ratio index.

Value

A matrix of importance measure, one row for each predictor variable. The column(s) are different importance measures.

See Also

`randomForest`

`oob.error.rate.wsrf` *Out-of-Bag Error Rate*

Description

Return out-of-bag error rate for "`wsrf`" model.

Usage

```
## S3 method for class 'wsrf'
oob.error.rate(object, tree, ...)
```

Arguments

`object`	object of class `wsrf`.
`tree`	logical or an integer vector for the index of a specific tree in the forest model. If provided as an integer vector, `oobErrorRate.wsrf` will give the corresponding out-of-bag error rates of the exact trees specified by `tree`. If TRUE, all error rates will be presented. If FALSE or missing, the gross error rate for the forest will be given.
`...`	not used.

Value

return a vector of error rates.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

See Also

wsrf

---

predict.wsrf *Predict Method for* wsrf *Model*

Description

Give the predictions for the new data by the forest model built from wsrf.

Usage

```
 ## S3 method for class 'wsrf'
predict(object, newdata, type=c("response",
  "class", "vote", "prob", "aprob", "waprob"), ...)
```

Arguments

object	object of class wsrf.
newdata	the data that needs to be predicted. Its format should be the same as that for wsrf.
type	the type of prediction required, a character vector indicating the types of output, and can be one of the values below: **vote** matrix of vote counts **response** predicted values. **class** the same as response. **prob** matrix of class probabilities. The probability is the proportion of trees in the forest voting for the particular outcome (prob = votes / ntree) **aprob** the average score from the decision trees for each class rather than the proportion of decision trees for each class (aprob = scores / ntree) **waprob** the weighted average, weighted by the accuracy of the tree (waprob = scores * accuracy / sum(accuracy))
...	optional additional arguments. At present no additional arguments are used.

Value

a list of predictions for the new data with corresponding components for each type of predictions. For type=class or type=class, a vector of length nrow(newdata), otherwise, a matrix of nrow(newdata) * (number of class label). For example, if given type=c("class", "prob") and the return value is res, then res$class is a vector of predicted class labels of length nrow(newdata), and res$prob is a matrix of class probabilities.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

See Also

wsrf

`print.wsrf` *Print Method for* `wsrf` *Model*

Description

Print a summary of the forest model or one specific tree in the forest model built from `wsrf`.

Usage

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

Arguments

`x`	object of class `wsrf`.
`trees`	the index of a specific tree. If missing, `print` will print a summary of the model.
`...`	optional additional arguments. At present no additional arguments are used.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

See Also

wsrf

`strength.wsrf` *Strength*

Description

Give the measure for the collective performance of individual trees in the forest model built from `wsrf`.

Usage

```
## S3 method for class 'wsrf'
strength(object, ...)
```

Arguments

`object`	object of class `wsrf`.
`...`	optional additional arguments. At present no additional arguments are used.

Details

The measure was introduced in Breiman (2001).

Value

A numeric value.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

References

Breiman, L. 2001 "Random forests". *Machine learning*, **45(1)**, 5–32.

See Also

wsrf

---

subset.wsrf *Subset of a Forest*

Description

Obtain a subset of a forest.

Usage

```
## S3 method for class 'wsrf'
subset(x, trees, ...)
```

Arguments

x	an object of class wsrf.
trees	which trees should be included in the sub-forest. An integer vector, which indicates the index of the trees.
...	not used.

Value

An object of class wsrf.

See Also

combine

Examples

```
library("wsrf")

# Prepare parameters.
ds     <- iris
target <- "Species"
vars   <- names(ds)
if (sum(is.na(ds[vars]))) ds[vars] <- randomForest::na.roughfix(ds[vars])
ds[target] <- as.factor(ds[[target]])
form <- as.formula(paste(target, "~ ."))
set.seed(42)
train <- sample(nrow(ds), 0.7*nrow(ds))
test  <- setdiff(seq_len(nrow(ds)), train)

# Build model.  We disable parallelism here, since CRAN Repository
# Policy (https://cran.r-project.org/web/packages/policies.html)
# limits the usage of multiple cores to save the limited resource of
# the check farm.

model.wsrf <- wsrf(form, data=ds[train, vars], parallel=FALSE)
print(model.wsrf)

# Subset.
submodel.wsrf <- subset.wsrf(model.wsrf, 1:200)
print(submodel.wsrf)
cl <- predict(submodel.wsrf, newdata=ds[test, vars], type="response")$response
actual <- ds[test, target]
(accuracy.wsrf <- mean(cl==actual))
```

---

varCounts.wsrf *Number of Times of Variables Selected as Split Condition*

Description

Return the times of each variable being selected as split condition. For evaluating the bias of wsrf towards attribute types (categorical and numerical) and the number of values each attribute has.

Usage

```
## S3 method for class 'wsrf'
varCounts(object)
```

Arguments

`object`	object of class `wsrf`.

Value

A vector of integer. The length is the same as the training data for building that `wsrf` model.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

See Also

wsrf

---

wsrf *Build a Forest of Weighted Subspace Decision Trees*

Description

Build weighted subspace C4.5-based decision trees to construct a forest.

Usage

```
## S3 method for class 'formula'
wsrf(formula, data, ...)
## Default S3 method:
wsrf(x, y, mtry=floor(log2(length(x))+1), ntree=500,
                        weights=TRUE, parallel=TRUE, na.action=na.fail,
                        importance=FALSE, nodesize=2, clusterlogfile, ...)
```

Arguments

x, formula	a data frame or a matrix of predictors, or a formula with a response but no interaction terms.
y	a response vector.
data	a data frame in which to interpret the variables named in the formula.
ntree	number of trees to grow. By default, 500
mtry	number of variables to choose as candidates at each node split, by default, floor(log2(length(x))+1).
weights	logical. TRUE for weighted subspace selection, which is the default; FALSE for random selection, and the trees are based on C4.5.
na.action	a function indicate the behaviour when encountering NA values in data. By default, na.fail. If NULL, do nothing.
parallel	whether to run multiple cores (TRUE), nodes, or sequentially (FALSE).
importance	should importance of predictors be assessed?
nodesize	minimum size of leaf node, i.e., minimum number of observations a leaf node represents. By default, 2.
clusterlogfile	character. The pathname of the log file when building model in a cluster. For debug.
...	optional parameters to be passed to the low level function wsrf.default.

Details

See Xu, Huang, Williams, Wang, and Ye (2012) for more details of the algorithm, and Zhao, Williams, Huang (2017) for more details of the package.

Currently, **wsrf** can only be used for classification. When `weights=FALSE`, C4.5-based trees (Quinlan (1993)) are grown by `wsrf`, where binary split is used for continuous predictors (variables) and *k*-way split for categorical ones. For continuous predictors, each of the values themselves is used as split points, no discretization used. The only stopping condition for split is the minimum node size must not less than `nodesize`.

Value

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

`confusion`	the confusion matrix of the prediction (based on OOB data).
`oob.times`	number of times cases are 'out-of-bag' (and thus used in computing OOB error estimate)
`predicted`	the predicted values of the input data based on out-of-bag samples.
`useweights`	logical. Whether weighted subspace selection is used? NULL if the model is obtained by combining multiple **wsrf** model and one of them has different value of 'useweights'.
`mtry`	integer. The number of variables to be chosen when splitting a node.

Author(s)

He Zhao and Graham Williams (SIAT, CAS)

References

Xu, B. and Huang, J. Z. and Williams, G. J. and Wang, Q. and Ye, Y. 2012 "Classifying very high-dimensional data with random forests built from small subspaces". *International Journal of Data Warehousing and Mining (IJDWM)*, **8(2)**, 44–63.

Quinlan, J. R. 1993 *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Zhao, H. and Williams, G. J. and Huang, J. Z. 2017 "wsrf: An R Package for Classification with Scalable Weighted Subspace Random Forests". *Journal of Statistical Software*, **77(3)**, 1–30. doi:10.18637/jss.v077.i03

Examples

```
library("wsrf")

# Prepare parameters.
ds <- iris
dim(ds)
names(ds)
target <- "Species"
vars   <- names(ds)
if (sum(is.na(ds[vars]))) ds[vars] <- randomForest::na.roughfix(ds[vars])
ds[target] <- as.factor(ds[[target]])
```

```
(tt  <- table(ds[target]))
form <- as.formula(paste(target, "~ ."))
set.seed(42)
train <- sample(nrow(ds), 0.7*nrow(ds))
test  <- setdiff(seq_len(nrow(ds)), train)

# Build model.  We disable parallelism here, since CRAN Repository
# Policy (https://cran.r-project.org/web/packages/policies.html)
# limits the usage of multiple cores to save the limited resource of
# the check farm.

model.wsrf <- wsrf(form, data=ds[train, vars], parallel=FALSE)

# View model.
print(model.wsrf)
print(model.wsrf, tree=1)

# Evaluate.
strength(model.wsrf)
correlation(model.wsrf)
res <- predict(model.wsrf, newdata=ds[test, vars], type=c("response", "waprob"))
actual <- ds[test, target]
(accuracy.wsrf <- mean(res$response==actual))

# Different type of prediction.
cl <- apply(res$waprob, 1, which.max)
cl <- factor(cl, levels=1:ncol(res$waprob), labels=levels(actual))
(accuracy2.wsrf <- mean(cl==actual))
```

Index