Package ‘wsrf’

October 10, 2015

Type Package

Title Weighted Subspace Random Forest for Classification

Version 1.5.29

Date 2015-10-10

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

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, 8(2):44-63, 2012, proposed by Baoxun Xu, Joshua Zhexue Huang, Graham Williams, Qiang Wang, and Yunming Ye. 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 (>= 2)

Depends R (>= 3.0.0), Rcpp (>= 0.10.2), stats, parallel

LinkingTo Rcpp

Suggests rattle (>= 2.6.26), randomForest (>= 4.6.7), party (>= 1.0.7), stringr (>= 0.6.2), knitr (>= 1.5)

VignetteBuilder knitr

NeedsCompilation yes

Author Qinghan Meng [aut],
      He Zhao [aut, cre],
      Graham Williams [aut],
      Junchao Lv [ctb],
      Baoxun Xu [aut]

Repository CRAN

Date/Publication 2015-10-10 15:43:26
R topics documented:

- combine.wsrf
- correlation.wsrf
- importance.wsrf
- oob.error_rate.wsrf
- predict.wsrf
- print.wsrf
- strength.wsrf
- subset.wsrf
- varCounts.wsrf
- wsrf
- wsrfParallelInfo

Index

combine.wsrf | Combine Ensembles of Trees

Description

Combine two or 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 <- rattle::weather
target <- "RainTomorrow"
vars <- setdiff(names(ds), c("Date", "Location", "RISK_MM"))
if (sum(is.na(ds[vars]))) ds[vars] <- randomForest::na.roughfix(ds[vars])
ds[target] <- as.factor(ds[[target]])
form <- as.formula(paste(target, "~ "))
correlation.wsrf

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

# Merge two models.
model.wsrf.big <- combine.wsrf(model.wsrf.1, model.wsrf.2)
print(model.wsrf.big)
c1 <- predict(model.wsrf.big, newdata=ds[test.1, vars], type="response")
actual <- ds[test.1, target]
(accuracy.wsrf <- sum(c1==actual)/length(actual))
```

correlation.wsrf       Correlation

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

Usage
```r
## 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)
importance.wsrf

References

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

```r
## 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, `oob.error.rate.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)

#### See Also

- `wsrf`
- `predict.wsrf`

## predict.wsrf

### Predict Method for `wsrf` Model

#### Description

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

#### Usage

```r
## S3 method for class 'wsrf'
predict(object, newdata, type=c("response", "class", "prob", "vote", "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, indicating the type 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 / ntrees)
  - **aprob**: the average score from the decision trees for each class rather than the proportion of decision trees for each class (aprob = scores / ntrees)
  - **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 vector of length `nrow(newdata)` if given type of `response` or `class`, otherwise, a matrix of `nrow(newdata) * (number of class label)`. 

Author(s)

He Zhao and Graham Williams (SIAT)

See Also

- `wsrf`

print.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

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

**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)

**See Also**

- `wsrf`

---

**Description**

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

**Usage**

```r
## 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)

**References**

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     <- rattle::weather
  target <- "RainTomorrow"
  vars   <- setdiff(names(ds), c("Date", "Location", "RISK_MM"))
  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.
  model.wsrf <- wsrf(form, data=ds[train, vars])
  print(model.wsrf)
```r
# Subset.
submodel.wsrfs <- subset.wsrfs(model.wsrfs, 1:200)
print(submodel.wsrfs)
class <- predict(submodel.wsrfs, newdata=ds[test, vars], type="response")
actual <- ds[test, target]
(accuracy.wsrfs <- sum(class==actual)/length(actual))
```

<table>
<thead>
<tr>
<th>varCounts.wsrfs</th>
<th>Number of Times of Variables Selected as Split Condition</th>
</tr>
</thead>
</table>

### 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

```r
## 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)

### 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

wsrf(formula, data, nvars, mtry, ntrees=500, weights=TRUE, parallel=TRUE, na.action=na.fail, importance=FALSE, clusterlogfile)

Arguments

- **formula**: a formula, with a response but no interaction terms.
- **data**: a data frame in which to interpret the variables named in the formula.
- **ntrees**: number of trees to build on each server; By default, 500
- **nvars, mtry**: number of variables to choose, by default, being the integer less than or equal to \(\log_2(ninputs) + 1\). For compatibility with other R packages like randomforest, both nvars and mtry are supported, however, only one of them should be specified.
- **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**: indicate the behaviour when encountering NA values in data.
- **parallel**: whether to run multiple cores (TRUE), nodes, or sequentially (FALSE).
- **importance**: should importance of predictors be assessed?
- **clusterlogfile**: character. The pathname of the log file when building model in a cluster. For debug.

Details

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

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 is 2.
An object of class \texttt{wsrf}, which is a list with the following components:

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

**Author(s)**

He Zhao and Graham Williams (SIAT)

**References**


**Examples**

```r
library("rsrf")

# Prepare parameters.
ds <- rattle::weather
dim(ds)
names(ds)
target <- "RainTomorrow"
id <- c("Date", "Location")
risk <- "RISK_MM"
ignore <- c(id, if (exists("risk")) risk)
vars <- setdiff(names(ds), ignore)
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.
model.wsr <- wsr(form, data=ds[train, vars])

# View model.
print(model.wsr)
```
print(model.wsrfrf, tree=1)

# Evaluate.
strength(model.wsrfrf)
correlation(model.wsrfrf)
cl <- predict(model.wsrfrf, newdata=ds[test, vars], type="response")
actual <- ds[test, target]
(accuracy.wsrfrf <- sum(cl==actual)/length(actual))

**wsrfrfParallelInfo**  
*Query about underlying parallel implementation information*

**Description**

Give the information about underlying parallel implementation.

**Usage**

```
wsrfrfParallelInfo(...)```

**Arguments**

...  

Optional additional arguments. At present no additional arguments are used.

**Details**

The parallel implementation cannot be changed after installation of the package. So this function is used to query which techniques are actually adopted as the underlying parallel implementation, among which are C++11, Boost, or no parallelism at all.

**Value**

A diagnostic message about the parallel implementation depends on the actual situation. Currently possible messages are:

1. With parallel computing disabled
2. Use C++ standard thread library for parallel computing
3. Use C++ Boost thread library for parallel computing

**Author(s)**

He Zhao (SIAT)

**See Also**

*wsrfrf*
Index

*Topic **classif**
  importance.wsrf, 4
  wsrfs, 10
*Topic **models**
  wsrfs, 10

combine, 8
combine(combine.wsrfs), 2
combine.wsrfs, 2
correlation(correlation.wsrfs), 3
correlation.wsrfs, 3

importance(importance.wsrf), 4
importance.wsrf, 4

oob.error.rate(oob.error.rate.wsrfs), 5
oob.error.rate.wsrfs, 5

parallel(wsrfsParallelInfo), 12
predict(predict.wsrfs), 5
predict.wsrfs, 5
print(print.wsrfs), 6
print.wsrfs, 6

strength(strength.wsrfs), 7
strength.wsrfs, 7
subset, 2
subset(subset.wsrfs), 8
subset.wsrfs, 8

varCounts.wsrfs, 9

wsrfs, 4–9, 10, 12
wsrfsParallelInfo, 12