

Package ‘ofw’

April 17, 2009

Title Optimal Feature Weighting algorithm

Version 1.0-0

Date 2008-11-15

Depends R (>= 2.4.1), e1071

Author Kim-Anh Le Cao and Patrick Chabrier

Description This package implements the stochastic meta algorithm called Optimal Feature Weighting for two multiclass classifiers, CART and SVM

Maintainer Kim-Anh Le Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>

License GPL (>= 2)

URL

Repository CRAN

Date/Publication 2008-11-15 23:16:12

R topics documented:

evaluate.learn	2
evaluateCARTparallel	4
getTree	6
learn	7
ofw	9
ofw-internal	12
ofwTune	12
srbct	14

Index	15
--------------	-----------

evaluate.learn *Error rate assessment of ofw*

Description

The error rate assessment `e.632+` (Efron and Tibshirani, 1997) is performed on `ofw` applied to CART or SVM. It requires first to launch `learn`.

Usage

```
## S3 method for class 'learn':
evaluate(obj, maxvar=15, type=obj$type, nvar=if(obj$type=="CART")
        obj$nclass+1 else NULL, ntreeTest= if(obj$type=="CART") 100 else NULL,
        weight=FALSE, ...)
```

Arguments

<code>obj</code>	An object from class <code>learn</code> .
<code>maxvar</code>	Size of the evaluated variable selection.
<code>type</code>	Classifier used in the object from class <code>learn</code>
<code>nvar</code>	If CART, number of randomly sampled variables in the selection that are used to construct each tree. Should be at least <code>obj\$nclass+1</code> to ensure generalizable trees.
<code>ntreeTest</code>	If CART, number of trees aggregated to evaluate the performance of <code>ofwCART</code> when each variable enters the selection.
<code>weight</code>	Should the weighting procedure be applied during the evaluation phase ?
<code>...</code>	not used currently.

Details

In the case of data sets with a small number of samples (e.g microarray data), the use of `e.632+` bootstrap error seems appropriate to assess the performance of the algorithm. With CART, as classification trees are unstable by nature, `ntreeTest` trees are aggregated.

Value

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

<code>maxvar</code>	Size of the evaluated variable selection.
<code>nvar</code>	Number of randomly sampled variables in the selection that are used to construct each tree.
<code>weight.eval</code>	Was the weighting procedure applied during the evaluation step ?
<code>weight.learn</code>	Was the weighting procedure applied during the learning step ?
<code>ntreeTest</code>	If CART, number of aggregated trees as variable enters the selection.

matTrain	A nsample by Bsample matrix indicating the training samples in each bootstrap sample.
matProb	A nvariable by Bsample matrix for each probability distribution learnt.
error	The evaluated e.632+ bootstrap error as each variable enters the selection.
sampleWeight	if weight = TRUE, the n by Bsample matrix indicating each sample weight in each bootstrap sample.
matPredInbag	A nvariable by Bsample matrix indicating the prediction of the inbag samples.
matPredTest	A nvariable by Bsample matrix indicating the prediction of the test samples.

Note

The e.632+ code comes from the ipred package.

This type of evaluation should only be used to compare several methods and not to assess the performance of only one method.

Author(s)

Kim-Anh L\textasciicircume Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>
Patrick Chabrier <Patrick.Chabrier@toulouse.inra.fr>

References

Efron, B. and Tibshirani R.J. (1997), *Improvements on cross-validation: the e.632+ bootstrap method*, Journal of American Statistical Association 92, 548-560.

L\textasciicircume Cao, K-A., Gonçalves, O., Besse, P. and Gadat, S. (2007), *Selection of biologically relevant genes with a wrapper stochastic algorithm* Statistical Applications in Genetics and Molecular Biology: Vol. 6: Iss.1, Article 29.

See Also

[learn](#)

Examples

```
## On data set "srbcct"
#data(srbct)
#attach(srbct)
#learn.boot.cart <- learn(srbct, as.factor(class), type="CART", ntree=50, nforest=100, mtry=
#eval.boot.cart <- evaluate(learn.boot.cart, ntreeTest=50, maxvar=10)
#plot(eval.boot.cart$error, type="l")
#learn.boot.svm <- learn(srbct, as.factor(class), type="SVM", nsvm=500, mtry=5, Bsample=3)
#eval.boot.svm <- evaluate(learn.boot.svm, maxvar=10)
#plot(eval.boot.svm$error, type="l")

#detach(srbct)
```

 evaluateCARTparallel

Error rate assessment of ofwCART

Description

The error rate assessment e.632+ (Efron and Tibshirani, 1997) is performed on ofwCART. This second version of evaluateCART allows to perform the learning step independently (for example with parallel computing).

Usage

```
evaluateCARTparallel(x, y, matTrain, matProb, maxvar=15, nvar=nlevels(y)+1,
  ntreeTest=100, weight=FALSE)
```

Arguments

x	A data frame with continuous values.
y	A response vector given as a factor (classification only).
matTrain	A <code>n</code> sample by <code>B</code> sample matrix indicating the training samples in each bootstrap sample.
matProb	A <code>n</code> variable by <code>B</code> sample matrix for each probability distribution learnt.
maxvar	Size of the evaluated variable selection.
nvar	Number of randomly sampled variables in the selection that are used to construct each tree. Should be at least <code>nlevels(y)+1</code> to ensure generalizable trees.
ntreeTest	Number of trees aggregated to evaluate the performance of ofwCART when each variable enters the selection.
weight	Should the weighting procedure be applied during the evaluation phase ?

Details

see evaluateCART

Value

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

maxvar	Size of the evaluated variable selection.
nvar	Number of randomly sampled variables in the selection that are used to construct each tree.
weight.eval	Was the weighting procedure applied during the evaluation step ?
ntreeTest	Number of aggregated trees as each variable enters the selection.
matTrain	A <code>n</code> sample by <code>B</code> sample matrix indicating the training samples in each bootstrap sample.

matProb A `nvariable` by `Bsample` matrix for each probability distribution learnt.

error The evaluated `e.632+` bootstrap error as each variable enters the selection.

sampleWeight if `weight = TRUE`, a `n` by `Bsample` matrix indicating each sample weight in each bootstrap sample.

matPredInbag A `nvariable` by `Bsample` matrix indicating the prediction of the inbag samples.

matPredTest A `nvariable` by `Bsample` matrix indicating the prediction of the test samples.

Note

The `e.632+` code comes from the `ipred` package from Thorsten.

This type of evaluation should only be used to compare several methods and not to assess the performance of only one method.

Author(s)

Kim-Anh L\textasciicircume Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>

Patrick Chabrier <Patrick.Chabrier@toulouse.inra.fr>.

References

Efron, B. and Tibshirani R.J. (1997), *Improvements on cross-validation: the e.632+ bootstrap method*, Journal of American Statistical Association 92, 548-560.

L\textasciicircume Cao, K-A., Gonçalves, O., Besse, P. and Gadat, S. (2007), *Selection of biologically relevant genes with a wrapper stochastic algorithm* Statistical Applications in Genetics and Molecular Biology: Vol. 6: Iss.1, Article 29.

See Also

evaluate

Examples

```
## On data set "data"
##first learn ofwCART on Bsample bootstrap samples and store matTrain.data and matProb.data
##data.evalCARTparallel <- evaluateCARTparallel(data[,-1], as.factor(data[,1], matTrain=matT
##plot(data.evalCARTparallel$error, type="l")
```

getTree

Extract a single tree from the last iteration of ofwCART.

Description

This function extract the structure of a tree from a `ofwCART` object for the last iteration.

Usage

```
getTree(ofwobj, k=1)
```

Arguments

<code>ofwobj</code>	a <code>ofw</code> object applied with classifier CART.
<code>k</code>	which tree to extract form the last forest?

Details

This function comes from the `randomForest` package.

Value

A matrix (or data frame, if `labelVar=TRUE`) with six columns and number of rows equal to total number of nodes in the tree. The six columns are:

<code>left daughter</code>	the row where the left daughter node is; 0 if the node is terminal
<code>right daughter</code>	the row where the right daughter node is; 0 if the node is terminal
<code>split var</code>	which variable was used to split the node; 0 if the node is terminal
<code>split point</code>	where the best split is; see Details for categorical predictor
<code>status</code>	is the node terminal (-1) or not (1)
<code>prediction</code>	the prediction for the node; 0 if the node is not terminal

See Also

[ofw](#)

Examples

```
data(srbct)
attach(srbct)
##ofwCART
learn.cart.keep <- ofw(srbct, as.factor(class),type="CART", ntree=50, nforest=10, mtry=5, ke
getTree(learn.cart.keep, k=3)
detach(srbct)
```

 learn

Learning ofw for error rate assesement

Description

`learn` simply performs the learning step of `ofw` on several bootstrap samples in order to assess the error rate on the test observations (see `evaluate`)

Usage

```
## Default S3 method:
learn(x, y, type="CART", ntree= if(type=="CART") 50 else NULL,
      nforest= if(type=="CART") 100 else NULL, nsvm= if(type=="SVM")
      20000 else NULL, mtry=5, do.trace=FALSE, nstable=50, weight=FALSE,
      Bsample=5, ...)
```

Arguments

<code>x</code>	A data frame with continuous values.
<code>y</code>	A response vector given as a factor (classification only).
<code>type</code>	Classifier used: either <code>CART</code> or <code>SVM</code> .
<code>ntree</code>	If <code>CART</code> , number of trees to grow for each iteration (trees aggregation).
<code>nforest</code>	If <code>CART</code> , number of iterations to run. This should not be set to too small a number, to ensure the convergence of the algorithm.
<code>nsvm</code>	If <code>SVM</code> , number of iterations to run. This should be set to a very large number, to ensure the convergence of the algorithm.
<code>mtry</code>	Number of variables sampled according to the weight vector <code>P</code> as candidates for each tree or <code>SVM</code> . This should be small enough to ensure stable results of the algorithm.
<code>do.trace</code>	If set to some integer, then current iteration is printed for every <code>do.trace</code> iterations and the number of the first stable variables is output.
<code>nstable</code>	Need <code>do.trace</code> set to some integer. Stopping criterion before <code>nforest</code> iterations are reached: if the <code>nstable</code> first weighted variables are the same after <code>do.stable</code> iterations, then stop.
<code>weight</code>	Should the weighting procedure be applied ?
<code>Bsample</code>	Number of bootstrap samples for the learning step.
<code>...</code>	not used currently.

Details

The object from class `learn` will be used in the generic function `evaluate`

Value

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

<code>x</code>	Original input
<code>y</code>	Original predictor
<code>type</code>	Classifier used: either <code>CART</code> or <code>SVM</code>
<code>nsample</code>	Total number of samples
<code>nclass</code>	Number of levels of <code>y</code> (number of classes)
<code>weight</code>	If <code>TRUE</code> the weighted procedure was performed during the learning.
<code>Bsample</code>	Number of bootstrap samples on which <code>ofwCART</code> is learnt.
<code>matTrain</code>	A <code>n</code> by <code>Bsample</code> matrix indicating the training samples in each bootstrap sample.
<code>matProb</code>	A <code>nvariable</code> by <code>Bsample</code> matrix for each probability distribution learnt.
<code>classWeight</code>	If <code>weight = TRUE</code> class weight vector.
<code>sampleWeight</code>	If <code>weight = TRUE</code> sample weight vector.

Note

The computation of `learn` is slow as it requires to launch `ofwBsample` times.

Parallelized computations are possible with `ofw` and `Rmpi` library for the learning step and `evaluateCARTparallel` for the evaluation step.

Author(s)

Kim-Anh L\textasciicircume Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>

Patrick Chabrier <Patrick.Chabrier@toulouse.inra.fr>

References

L\textasciicircume Cao, K-A., Gonçalves, O., Besse, P. and Gadat, S. (2007), *Selection of biologically relevant genes with a wrapper stochastic algorithm* Statistical Applications in Genetics and Molecular Biology: Vol. 6: Iss.1, Article 29. L\textasciicircume Cao, K-A., Bonnet, A., and Gadat, S., *Multiclass classification and gene selection with a stochastic algorithm* <http://www.lsp.ups-tlse.fr/Recherche/Publications/2007/cao05.html>.

See Also

[evaluate.learn](#)

Examples

```
## On data set "data"
#data(srbct)
#attach(srbct)
#learn.boot.cart <- learnCART(srbct, as.factor(class), type="CART", ntree=50, nforest=200, mt
#learn.boot.svm <- learnSVM(srbct, as.factor(class), type="SVM", nsvm=500, mtry=5, Bsample=3
#detach(srbct)
```

Description

ofw implements a meta algorithm called "Optimal Feature Weighting" for multiclass classification by aggregating either CART or SVM, in the context of continuous variables.

Usage

```
## Default S3 method:
ofw(x,y,type="CART", ntree= if(type=="CART") 50 else NULL, nforest=
  if(type=="CART") 100 else NULL, nsvm= if(type=="SVM") 100 else NULL, mtry=5,
  do.trace=FALSE, nstable=25, keep.inbag=if(type=="CART") FALSE else NULL,
  keep.forest=if(type=="CART") TRUE else NULL, weight=FALSE, ...)
## S3 method for class 'ofw':
print(x, ...)
```

Arguments

x	A data frame with continuous values (for the print method, an ofw object).
y	A response vector given as a factor (classification only).
type	Classifier used: either CART or SVM
ntree	If CART, number of trees to grow for each iteration (trees aggregation).
nforest	If CART, number of iterations to run. This should not be set to too small a number, to ensure the convergence of the algorithm.
nsvm	If SVM, number of iterations to run. This should be set to a very large number, to ensure the convergence of the algorithm.
mtry	Number of variables sampled according to the weight vector P as candidates for each tree or SVM. This should be small enough to ensure stable results of the algorithm.
do.trace	If set to some integer, then current iteration is printed for every do.trace iterations and the number of the first stable variables is output.
nstable	Need do.trace set to some integer. Stopping criterion before nforest or nsvms iterations are reached: if the nstable first weighted variables are the same after do.stable iterations, then stop.
keep.inbag	If CART, should an n by ntree matrix be returned that keeps track of which samples are "in-bag" in which trees (and how many times as it is a sampling with replacement) in the last forest.
keep.forest	If CART, and if set to TRUE, the last forest (or last iteration) will be retained in the output object and the getTree function can be used to see how the trees were constructed.
weight	Should the weighting procedure be applied ?
...	not used currently.

Details

The Optimal Feature Weighting algorithm learns the probability distribution P on all variables. The more useful the variables in the classification task, the heavier their weight. When the CART classifier is used, the trees are aggregated for each iteration (bagging).

Value

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

<code>call</code>	The original call to <code>ofwCART</code> .
<code>type</code>	Classifier used.
<code>classes</code>	Level attributes of the classes in the <code>y</code> predictor.
<code>mean.error</code>	Internal mean error rate vector of length <code>nforest</code> .
<code>prob</code>	Probability distribution vector P or weighting vector of length the total number of variables.
<code>list</code>	Name of variables and their respective importance weight, sorted by decreasing order.
<code>ntree</code>	If CART, number of trees grown for each iteration.
<code>nforest</code>	If CART, number of iterations asked in the procedure.
<code>nsvm</code>	If SVM, number of iterations asked in the procedure.
<code>maxiter</code>	Actual number of iterations performed (different than <code>nforest</code> or <code>nsvm</code> if <code>nstable</code> variables are obtained after <code>do.trace</code> steps).
<code>mtry</code>	Number of predictors sampled with P for each tree for each iteration.
<code>do.trace</code>	The number of iteration is printed every <code>do.trace</code> and the stopping criterion is tested.
<code>nstable</code>	Number of stable variables obtained if <code>maxiter</code> < <code>nforest</code> .
<code>weight</code>	If <code>TRUE</code> the weighted procedure was performed.
<code>classWeight</code>	If <code>weight = TRUE</code> class weight vector.
<code>sampleWeight</code>	If <code>weight = TRUE</code> sample weight vector.
<code>forest</code>	If CART, a list that contains the entire last forest of the last iteration; <code>NULL</code> if <code>keep.forest=FALSE</code> .
<code>inbag</code>	If <code>keep.inbag = TRUE</code> a <code>n</code> by <code>ntree</code> matrix is returned with “in-bag” samples in which trees (and sampled how many times).

Note

The `ofw` for CART structure has been first largely inspired from the `randomForest` package which was itself based on Leo Breiman and Adele Cutler’s Fortran code. The code now written in C (or R).

The actual implementation of `ofw` is restrained to classification task with continuous variables. It has been especially developed to deal with $p \gg n$ data sets, such as microarrays.

Normalisation has first to be performed by the user. For extremely large data sets, a large pre processing is advised to speed up the procedure.

In contrary to CART, the `ofw` version with SVM does not provide the internal mean error rate.

Author(s)

Kim-Anh L\textasciicircume Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>

Patrick Chabrier <Patrick.Chabrier@toulouse.inra.fr>

Based on Leo Breiman and Adele Cutler Fortran code and `randomForest` package for `ofwCART` version.

References

Gadat, S. and Younes, L. (2007), *A Stochastic Algorithm for Feature Selection in Pattern Recognition*, *Journal of Machine Learning* 8, 509-548

L\textasciicircume Cao, K-A., Gonçalves, O., Besse, P. and Gadat, S. (2007), *Selection of biologically relevant genes with a wrapper stochastic algorithm* *Statistical Applications in Genetics and Molecular Biology*: Vol. 6: Iss.1, Article 29.

L\textasciicircume Cao, K-A., Bonnet, A., and Gadat, S., *Multiclass classification and gene selection with a stochastic algorithm* <http://www.lsp.ups-tlse.fr/Recherche/Publications/2007/cao05.html>.

See Also

`getTree`, `learn`, `evaluate.learn`

Examples

```
## On data set "srbcct"
data(srbct)
attach(srbct)

##ofwCART
learn.cart <- ofw(srbct, as.factor(class),type="CART", ntree=100, nforest=200, mtry=5)
print(learn.cart)
## Look at variable importance:
learn.cart$prob
## Look at the 10 most important variables:
learn.cart$list[1:10]
## Look if the internal mean error is decreasing w.r.t the number of iterations:
plot(learn.cart$mean.error, type="l")

##ofwSVM
learn.svm <- ofw(srbct, as.factor(class),type="SVM", nsvm=500, mtry=5)
print(learn.svm)
## Look at variable importance:
learn.svm$prob
## Look at the 10 most important variables:
learn.svm$list[1:10]
## to use the do.trace options:
#learn.cart <- ofw(srbct, as.factor(class),type="CART", ntree=100, nforest=200, Mtry=5, do.trace=50)
#learn.svm <- ofw(srbct, as.factor(class),type="SVM", nsvm=500, mtry=5, do.trace=50, nstable=50)
#learn.cart
#learn.svm
detach(srbct)
```

ofw-internal *Internal ofw functions*

Description

Internal functions in ofw: `learnCART`, `learnSVM`, `evaluateCART.learnCART`, `evaluateSVM.learnSVM`, `ofwCART`, `ofwSVM`, `simplexe`, `svmlearn`.

Details

These functions are not to be called by the user.

Author(s)

Kim-Anh L\textasciicircume Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>

Patrick Chabrier <Patrick.Chabrier@toulouse.inra.fr>

ofwTune *Tuning the parameters for ofwCART or ofwSVM*

Description

`ofwTune` helps to tune the main parameters: `mtry`, `nforest` and `ntree` for `ofwCART` or `mtry` and `nsvm` for `ofwSVM`.

Usage

```
## Default S3 method:
ofwTune(x, y, type="CART", ntree= if(type=="CART") 50 else NULL,
        nforest= if(type=="CART") 100 else NULL, nsvm= if(type=="SVM") 500 else NU
        mtry.test=seq(5,15,length=3), do.trace=FALSE, nstable=10, weight=FALSE, ...
```

Arguments

<code>x</code>	A data frame with continuous values.
<code>y</code>	A response vector given as a factor (classification only).
<code>type</code>	The version of OFW to perform.
<code>ntree</code>	Number of trees to grow for each iteration (trees aggregation) in case of <code>ofwCART</code> is chosen.
<code>nforest</code>	Total number of iterations to run if <code>ofwCART</code> .
<code>nsvm</code>	Total number of iterations to run if <code>ofwSVM</code> .
<code>mtry.test</code>	Vector defining the number of variables sampled according to the weight vector <code>P</code> to be tested. By default the vector is defined as <code>'seq(5,15,length=3)'</code> .

<code>do.trace</code>	If set to some integer, then current iteration is printed for every <code>do.trace</code> iterations and the number of the first stable variables is output.
<code>nstable</code>	Need <code>do.trace</code> set to some integer. Stopping criterion before <code>nforest</code> or <code>nsvm</code> iterations are reached: if the <code>nstable</code> first weighted variables are the same after <code>do.stable</code> iterations, then stop.
<code>weight</code>	Should the weighting procedure be applied ?
<code>...</code>	not used currently.

Details

`ofwTune` consists in testing either `ofwCART` or `ofwSVM` with several variable subsets sizes in the given sequence `mtry.test`. For each `mtry`, the algorithm is performed twice and the function `ofwTune` outputs the intersection length of the first `nstable` variables selected with these 2 `ofw`. The value `mtry` to choose should be the one that gives the largest intersection.

The total number of iteration to tune should then be 2 to 3 times the maximum iterations reached for the optimal `mtry`.

In case of `ofwCART`, to choose `ntree`, the user should run `ofwTune` with several values of `ntree`. Usually, the more the trees the stabler the results.

Value

A list with the following components:

<code>type</code>	The classifier applied to <code>ofw</code> (either <code>CART</code> or <code>SVM</code>).
<code>nstable</code>	The number of stable variables chosen.
<code>mtry.test</code>	The different subset size tested.
<code>param</code>	A 2 by <code>length(mtry.test)</code> matrix indicating the number of stable variables obtained for each value of <code>mtry</code> .
<code>iter.max</code>	A 2 by <code>length(mtry.test)</code> matrix indicating the maximum number of iterations reached for each value of <code>mtry</code> .
<code>weight</code>	If <code>TRUE</code> the weighted procedure was performed during the learning.

Note

The computation of `ofwTune` might be slow as it consists in launching `ofw` $2 * \text{length}(mtry.test)$.

Author(s)

Kim-Anh L\textasciicircume Cao <Kim-Anh.Le-Cao@toulouse.inra.fr>

Patrick Chabrier <Patrick.Chabrier@toulouse.inra.fr>

References

L\textasciicircume Cao, K-A., Gonçalves, O., Besse, P. and Gadat, S. (2007), *Selection of biologically relevant genes with a wrapper stochastic algorithm* Statistical Applications in Genetics and Molecular Biology: Vol. 6: Iss.1, Article 29.

See Also

[ofw](#)

Examples

```
## On data set "srbct"
#data(srbct)
#attach(srbct)
#tune.cart <- ofwTune(srbct, as.factor(class), type="CART", ntree=50, nforest=200, mtry.test)
#tune.cart
#tune.svm <- ofwTune(srbct, as.factor(class), type="SVM", nsvm=500, mtry.test=seq(5,10,length))
#tune.svm
##Using do.trace options
#tune.cart <- ofwTune(srbct, as.factor(class), type="CART", ntree=50, nforest=200, mtry.test)
#tune.cart
#tune.svm <- ofwTune(srbct, as.factor(class), type="SVM", nsvm=500, mtry.test=seq(5,10,length))
#tune.svm

#detach(srbct)
```

srbct

Small version of the small round blue cell tumors of childhood data for illustration purpose

Description

This data set gives the expression measure of a small subset of genes that were randomly sampled from the the original data set from Khan et al. 2001.

Usage

```
data(srbct)
```

Format

A data frame `srbcct` containing 200 genes and 63 microarrays (cases).

A vector `class` containing the class tumour of each case.

Source

<http://research.nhgri.nih.gov/microarray/Supplement>

References

Khan et al. (2001) Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine* 7 Number 6, June.

Index

*Topic **classif**

evaluate.learn, 1
evaluateCARTparallel, 3
learn, 6
ofw, 8
ofw-internal, 11
ofwTune, 12

*Topic **datasets**

srbct, 14

*Topic **tree**

evaluate.learn, 1
evaluateCARTparallel, 3
getTree, 5
learn, 6
ofw, 8
ofwTune, 12

class(*srbct*), 14

evaluate(*evaluate.learn*), 1
evaluate.learn, 1, 8, 11
evaluateCART(*ofw-internal*), 11
evaluateCARTparallel, 3
evaluateSVM(*ofw-internal*), 11

getTree, 5, 11

learn, 3, 6, 11
learnCART(*ofw-internal*), 11
learnSVM(*ofw-internal*), 11

ofw, 5, 6, 8, 13
ofw-internal, 11
ofwCART(*ofw-internal*), 11
ofwSVM(*ofw-internal*), 11
ofwTune, 12

print.ofw(*ofw*), 8

simplexe(*ofw-internal*), 11
srbct, 14
svmlearn(*ofw-internal*), 11