Package ‘obliqueRF’

February 20, 2015

Title Oblique Random Forests from Recursive Linear Model Splits

Version 0.3

Date 2012-08-10

Depends R (>= 2.0.0), stats, ROCR, pls, mda, e1071

Author Bjoern Menze and Nico Splitthoff

Description Random forest with oblique decision trees for binary classification tasks. Discriminative node models in the tree are based on: ridge regression, partial least squares regression, logistic regression, linear support vector machines, or random coefficients.

Maintainer D.N. Splitthoff `<nico.splitthoff@gmx.de>`

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-08-10 18:22:37

NeedsCompilation no

R topics documented:

`importance` *Extract variable importance measure*

Description

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

Usage

```
## S3 method for class 'obliqueRF'
importance(x, ...)
```

Arguments

x	an object of class obliqueRF.
...	not used.

Value

The variable importance

See Also

obliqueRF

Examples

```
require(obliqueRF);
data(iris);
#extract feature matrix
x<-as.matrix(iris[,1:4]);
#convert to 0/1 class labels
y<-(as.numeric(iris[,5])>1)*1;

iris.orf <- obliqueRF(x,y,training_method="log",bImportance=TRUE);
importance(iris.orf);

## Also see example given in ?obliqueRF()
```

---

obliqueRF *Classification with Oblique Random Forest*

Description

obliqueRF implements a random forest with oblique decision trees for binary classification tasks. Discriminative node models in the tree are based on: ridge regression, logistic regression, linear support vector machines, or random splits.

Usage

```
## Default S3 method:
obliqueRF(
                    x,
                    y,
                    x.test=NULL,
                    y.test=NULL,
                    mtry=NULL,
                    ntree=100,
                    training_method="ridge",
                    bImportance = F,
                    bProximity = F,
                    verbose = F,
                    ...
                    )
```

Arguments

`x` a data frame or a matrix of predictors; rows are samples, columns are featues

`y` a binary response vector (numeric, factor); for now only binary classification problems are supported

`x.test` optional: predictors of a test data set; if no test data set is given the training data set is used

`y.test` optional: binary response vector of a test data set

`mtry` the number of variables to be tested in each node; default is mtry = max(sqrt(ncol(x),2)

`ntree` the number of trees to generate in the forest; default is ntree = 100

`training_method`
specify the node model; valid models are "ridge" for fast ridge regression using SVD, "ridge_slow" for a slower version using separate explicit ridge regressions, "pls" for partial least squares regression, "svm" for a linear support vector machine, "log" for logistic regression, "rnd" for a random hyperplane; hyperparameters for constrained methods are adapted to the oob data available at the node

`bImportance` calculate the obliqueRF variable importance? default is FALSE; importance can only be calculated for unconstrained regression and sets training_method="log"; set ntree to a very large value

`bProximity` calculate the obliqueRF sample proximity? default is FALSE; be aware that the proximity matrix scales with nrow(x)^2 and may require a prohibitive large amount of memory; set ntree to a very large value

`verbose` print status messages?

`...` not used

Details

Subspace dimensionality mtry should be adjusted on a test set for optimal performance; ntree should be chosen sufficiently large.

Node models with constraint, i.e., ridge regression, partial least squares regression, linear support vector machine, are optimized in each split in a test on the out-of-bag samples available at that node. (Ridge and partial least squares regression are used without feature scaling, the support vector machine model scales feature.) Choose the logistic node model if a constrained fit is not desired or required.

The obliqueRF importance counts how often a variable was deemed relevant (at .05 level) when chosen for a split at a node (increasing the importance value by 1) and how often it was irrelevant for the split (decreasing by 1). Significance is determined through ANOVA tables for the fitted logistic node model.

This is an R implementation, C code available from the authors upon request.

Value

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

`call` the original call to `obliqueRF`

`type` for now only `classification`

`errs` list with errors

`class_names` class names referring to classes "0" and "1" in errs.

`pred` list containing the prediction result

`lab` description of the node training method

`ntree` number of trees used

`mtry` number of split variables

`importance` a vector with the variable importances - or NULL, if the importance was not calculated

`proximity` the variable proximity - or NULL, if the proximity was not calculated

`num_classes` the number of classes

`trees` the tree structure that was learned

Author(s)

Bjoern Menze <menze@csail.mit.edu> and D.N. Splitthoff <nico.splitthoff@gmx.de>.

References

Menze BH, Kelm BM, Splitthoff DN, Koethe U, Hamprecht FA. On oblique random forests. Proc ECML/PKDD 2011. LNCS 6911, 453-469 `http://people.csail.mit.edu/menze/papers/menze_11_oblique.pdf`.

See Also

`predict.obliqueRF`, `importance.obliqueRF`

Examples

```
require(obliqueRF)
data(iris)

## data
# extract feature matrix
x<-as.matrix(iris[,1:4])
# convert to 0/1 class labels
```

```
y<-as.numeric(iris[,5]=="setosa")

## train
smp<-sample(1:nrow(iris), nrow(iris)/5)
obj <- obliqueRF(x[-smp,], y[-smp])

## test
pred <- predict(obj, x[smp,], type="prob")
plot(pred[,2],col=y[smp]+1,ylab="setosa probability")
table(pred[,2]>.5,y[smp])

## example: importance
imp<-rep(0,ncol(x))
names(imp)<-colnames(x)
numIterations<-2 #increase the number of iterations for better results, e.g., numIterations=100
for(i in 1:numIterations){
 obj<-obliqueRF(x,y,
training_method="log", bImportance=TRUE,
mtry=2, ntree=20)
 imp<-imp+obj$imp
 plot(imp,t='l', main=paste("steps:", i*20), ylab="obliqueRF importance")
}
```

`orfNews` *Show the NEWS file*

Description

Show the NEWS file of the obliqueRF package.

Usage

```
orfNews()
```

Value

Package released: 05 Sept 2011.

`predict.obliqueRF` *predict method for oblique random forest objects*

Description

Prediction of test data using oblique random forest.

Usage

```
## S3 method for class 'obliqueRF'
predict(object, newdata, type="response", proximity=F,...)
```

Arguments

`object` an object of class `obliqueRF`, as that created by the function `obliqueRF`.

`newdata` a matrix containing new data.

`type` one of `response`, `prob.` or `votes`, indicating the type of output: predicted values, matrix of class probabilities, or matrix of vote counts.

`proximity` should proximity measures be computed (based on all data)?

`...` not used currently.

Value

`pred`, which can be the following, depending on `type`:

`response` `pred` is a vector with predicted classes (threshold is 0.5).

`prob` `pred` is a matrix of class probabilities (one row for each class and one column for each input).

`votes` `pred` is a matrix of vote counts (one row for each class and one column for each new input).

If `proximity=TRUE`, the returned object is a list with two components: `pred` is the prediction (as described above) and `proximity` is the proximitry matrix.

Author(s)

Bjoern Menze <menze@csail.mit.edu> and D.N. Splitthoff <nico.splitthoff@gmx.de>.

References

Menze BH, Kelm BM, Splitthoff DN, Koethe U, Hamprecht F. On oblique random forests. Proc ECML/PKDD 2011. LNAI, 16p. http://people.csail.mit.edu/menze/papers/menze_11_oblique.pdf.

See Also

`obliqueRF`

Examples

```
## Classification:
require(obliqueRF);
data(iris);
#sample some cases
s<-sample(150,100);
#extract feature matrix
x_train<-as.matrix(iris[s,1:4]);
```

```
#convert to 0/1 class labels
y_train<-(as.numeric(iris[s,5])>1)*1;
iris.orf <- obliqueRF(x_train,y_train);

#extract feature matrix
x<-as.matrix(iris[-s,1:4]);
#convert to 0/1 class labels
y<-(as.numeric(iris[-s,5])>1)*1;
pred <- predict(iris.orf,x,type="response",proximity=TRUE);
sum(pred$pred!=y)
```

Index