Package ‘gensemble’

February 19, 2015

Type Package
Title generalized ensemble methods
Version 1.0
Date 2013-01-25
Author Peter Werner, Eugene Dubossarsky
Maintainer Peter Werner <gensemble.r@gmail.com>
Description Generalized ensemble methods allowing arbitrary underlying
       models to be used. Currently only bagging is supported.
License GPL-2
Depends methods
Repository CRAN
Date/Publication 2013-02-11 06:57:24
NeedsCompilation no

R topics documented:

ab.arglist ................................................................. 2
ab.create ................................................................. 3
ab.model ................................................................. 4
ab.predict ................................................................. 4
AbstractModel-class ...................................................... 5
gensemble ............................................................... 7
gensemble-class ......................................................... 10
mksampsize ............................................................. 11
predict.gensemble ....................................................... 12

Index 14
ab.arglist

Get the arglist for the given model

Description

Generate a list containing the arguments to be passed to the underlying model building function. Most interaction with AbstractModel-class is done via ab.create, ab.model and ab.predict, however it may be useful for debugging purposes to call this function directly.

Usage

ab.arglist(ab, X, Y)

Arguments

ab An object of class AbstractModel-class
X The X values. Will be passed through xtrans.
Y The Y values. Will be passed through ytrans.

Details

Both X and Y must be present. They will be passed through the xtransfrom and ytransform functions respectively, which default to a passthrough. If a formula is to be used, the X and Y data will be combined via cbind.

Value

Returns a list of values that will be passed to the underlying model function

Author(s)

Peter Werner

See Also

AbstractModel-class, gensemble.
**ab.create**

Create an Abstract Model instance

**Description**

This function provides a wrapper around instantiating an object of class `AbstractModel-class`. Please see the documentation for `AbstractModel-class` for example usage.

**Usage**

```
ab.create(model.call, model.args = list(), predict.args = list(), formula = NA, ...)
```

**Arguments**

- `model.call` The name of the model function as a string.
- `model.args` A named list of arguments. If used with `gensemble` please note classification requires a probability matrix to be returned.
- `predict.args` A named list of arguments to be passed to the predict method for the model selected.
- `formula` A logical indicating if the formula method of the underlying model should be used or not. If this is NA (i.e. unspecified), `ab.create` will examine `formals` of `model.call` to determine if formula syntax is required.
- `...` Additional arguments to be passed to `AbstractModel-class` creation.

**Value**

Returns an object of class `AbstractModel-class`.

**Author(s)**

Peter Werner

**See Also**

`ab.model`, `ab.predict`, `AbstractModel-class`, `gensemble`.
**ab.predict**

---

**ab.predict**

---

**Description**

This function can be used to call the underlingly model of an *AbstractModel-class* instance. Please see the documentation for *AbstractModel-class* for example usage.

**Usage**

```
ab.predict(ab, mod, X)
```

**Arguments**

- `ab` An object of type *AbstractModel-class*.
- `mod` A trained instance of the underlying model.
- `X` The X values to be passed to the model function.

**Value**

Returns a trained instance of whatever underlying model is in use. Suitable to be passed to `predict` or `ab.predict`.

**Author(s)**

Peter Werner

**See Also**

`ab.create, ab.predict, AbstractModel-class, gensemble`.

---

**ab.predict**

---

**Description**

Take a model trained by ab.model and use it for prediction.

Please see the documentation for *AbstractModel-class* for example usage.

**Usage**

```
ab.predict(ab, mod, X)
```
AbstractModel-class

Arguments

ab  An object of type AbstractModel-class.
mod A trained model object, possibly the return value from a call to ab.model.
X   The input predictors.

Value

This will return whatever a call to predict would return for the given model in use by the AbstractModel instance passed in ab.

Author(s)

Peter Werner

See Also

ab.model, ab.create, AbstractModel-class, gensemble.

AbstractModel-class  Class "AbstractModel"

Description

AbstractModel is an abstraction of R modelling functions/packages. Designed to be used with gensemble.

Objects from the Class

It is best to use ab.create to instantiate an object of this class.

Slots

model: The model function to call e.g. "ksvm" "nnet"
model_args: Named list of arguments to be passed to the model call, excluding X and Y
predict: The model prediction function, if different from predict
predict_args: Named list of arguments to be passed to the predict function
xtrans: Function that will be passed the predictor matrix, prior to any model or predict call
ytrans: Function that will be passed the response vector, prior to any model or predict call
formula: A logical indicating formula syntax should be used

Author(s)

Peter Werner <gensemble.r@gmail.com>
AbstractModel-class

See Also

ab.model, ab.predict, ab.model.gensemble

Examples

```r
## Not run:
# ksvm classification
library(kernlab)
# note we pass prob.model=TRUE as gensemble requires the probabilities for classification.
ksvm_model_args <- list(prob.model=TRUE, type="C-svc", C=1, epsilon=0.1)
# create the abstract model instance
abm <- ab.create(model.call="ksvm", model.args=ksvm_model_args, predict.args=list(type="probabilities"), xtrans=as.matrix)

# nnet classification
library(nnet)
# use the formula
abm <- ab.create(model.call="nnet", model.args=list(size=3), formula=TRUE)

# rpart classification
library(rpart)
abm <- ab.create(model.call="rpart", model.args=list(control=rpart.control(minsplit=0)), predict.args=list(type="probabilities"), xtrans=as.matrix)

# classification test stub (try with the different abm's from above)
X <- iris[,1:4]
Y <- iris[,5]
# generate train/test samples
cnt <- nrow(iris)
samp <- sample(1:cnt, cnt * 0.7, rep=FALSE)
# train the model
mod <- ab.model(abm, X[samp,], Y[samp])
# get the predictions
preds <- ab.predict(abm, mod, X[-samp,])
# compare to actual classes
cbind(apply(preds, 1, which.max), Y[-samp])

# ksvm regression
library(kernlab)
abm <- ab.create(model.call="ksvm", xtrans=as.matrix)

# nnet regression
library(nnet)
abm <- ab.create(model.call="nnet", model.args=list(size=3, linout=TRUE, maxit=400, rang=0.001, decay=0.0001), xtrans=as.matrix)

# rpart regression
library(rpart)
abm <- ab.create(model.call="rpart", model.args=list(method='anova', control=rpart.control(minsplit=2, cp=1e-03)), xtrans=as.matrix)

# regression test stub
X <- trees[,1:2]
Y <- trees[,3]
# generate train/test samples
cnt <- nrow(trees)
```
gensemble

```
samp <- sample(1:cnt, cnt * 0.7L, rep=FALSE)
# build the model
mod <- ab.model(abm, X[samp,], Y[samp])
# try some predictions
preds <- ab.predict(abm, mod, X[-samp,])
# compare vs actual values
cbind(preds, Y[-samp])
```

```
## End(Not run)
```

---

### Gensemble

**Generalized ensemble methods**

#### Description

Gensemble is a generalisation of random forests allowing allowing arbitrary use of underlying models.

#### Usage

```
gensemble(abm, X, Y, sampsize = NULLL, sampsize_prop = FALSEL, nmods = 100L, 
perturb_val = 0.1L, Xtest = NULLL, Ytest = NULLL, do.trace = TRUEL, 
stepsize = 10L)
```

#### Arguments

- **abm**: An object of type `AbstractModel-class`
- **X**: A data frame or matrix of predictors
- **Y**: A response vector. If `Y` is a factor classification is assumed, otherwise regression. See the notes for more details.
- **sampsize**: A list or vector of sample sizes used when creating a bagged sample. If not supplied, all input data will be used to build the models. See `mksampsize` for details on how this will be interpreted.
- **sampsize_prop**: A boolean indicating the values in `sampsize` should be interpreted as proportions.
- **nmods**: How many models to build.
- **perturb_val**: The proportion of input data to perturb.
- **Xtest**: Optional test set of `X` values.
- **Ytest**: Optional test set of `Y` values.
- **do.trace**: If `TRUE`, summary statistics will be printed. The information printed is as follows:
  1. For classification, the per-class accuracy is printed, along with the proportion of training points not yet included in any model, and the total accuracy.
  2. For regression, the variance, `mse`, scaled `mse`, estimated R^2 and proportion of training points not yet included in any model.
stepsizesize If do.trace is TRUE, specifies how often to print trace information. For example, a value of 10 will print every 10 models. A value of 1 will print after every model.

Details

This is a general implementation of bagging. It enables (in theory) any underlying modelling/learning algorithm to be used, via the AbstractModel-class.

Value

An object of class gensemble-class uncertainty gensemble-class.

Wrapping the model function

The first argument to gensemble is an instance of an AbstractModel-class. You will need to wrap the model you wish to use in this class before using gensemble.

First off, you should probably make sure the model function works for the data you will pass to gensemble. For example let’s say we are using ksvm from kernlab, on the iris data set. You might have something that looks like this:

```r
library(kernlab)
x <- iris[,1:4]
y <- iris[,5]
cnt <- nrow(iris)
samp <- sample(1:cnt, cnt * 0.7)
mod <- ksvm(as.matrix(x[samp]), y[samp], type="C-svc", C=1, epsilon=0.1)
preds <- predict(mod, x[-samp])
```

We can wrap this up in an instance of AbstractModel-class as follows:

```r
abm <- ab.create(model.call="ksvm", model.args=list(type="C-svc", C=1, epsilon=0.1), xtrans=as.matrix)
```

We now pass the arguments we would pass to ksvm via the model.args argument to ab.create. It is simply list of the arguments and their values.

Note we define the X transform to be as.matrix, which means the X values passed to ksvm by AbstractModel will first be run through as.matrix.

We can check this is working as expected using ab.model and ab.predict.

```r
mod <- ab.model(abm, X[samp], Y[samp])
preds <- ab.predict(abm, mod, X[-samp])
```

Classification with gensemble requires a probability matrix to be returned by the underlying model. We will need to pass some extra arguments to ksvm to make sure this is present.
We have added two extra things. First we pass prob.model=TRUE to the ksvm model function, telling it to generate probabilities. We also added predict.args to AbstractModel, so when the predict function for ksvm is called, it will be passed type="probabilities", telling it to return a matrix of class probabilities.

We now have an AbstractModel-class instance ready to use with gensemble. Please see the documentation for AbstractModel-class for further examples and information.

Note
This is still relatively experimental code. In particular I expect AbstractModel to not be abstract enough at some point in the near future, and fail to be able to model normal usage. We welcome bug reports or any other feedback.

Author(s)
Peter Werner and Eugene Dubossarsky <gensemble.R@gmail.com>

References
http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

See Also
mksampsize, AbstractModel-class, predict.gensemble

Examples
## Not run:
#classification with kernlab
library(kernlab)
#make our abstract model object
abm <- ab.create(model.call="ksvm", model.args=list(prob.model=TRUE,
type="C-svc", C=1, epsilon=0.1), predict.args=list(type="probabilities"),
xtrans=as.matrix)

#the example data
X <- iris[,1:4]
Y <- iris[,5]
#create a training/test set
samp <- sample(1:nrow(iris), nrow(iris) * 0.8)
#train the model
gmod <- gensemble(abm, X[samp,], Y[samp], sampsize=0.8, sampsize_prop=TRUE)
#test it out
gpreds <- predict(gmod, X[-samp,])
#compare
cbind(apply(gpreds, 1, which.max), Y[-samp])
#regression with rpart
library(rpart)
abm <- ab.create(model.call="rpart", model.args=list(control=rpart.control(minsplit=2)))
X <- trees[,1:2]
Y <- trees[,3]
#generate a training set
samp <- sample(1:nrow(trees), nrow(trees) * 0.8)
#build the model
Gmod <- gensemble(abm, X[samp], Y[samp])
#use it to predict with the test set
gpreds <- predict(Gmod, X[-samp])
#compare
cbind(gpreds, Y[-samp])

## End(Not run)

gensemble-class Class "gensemble"

Description

The gensemble class is returned by a call to gensemble. It should be passed to predict.gensemble for prediction.
Details on the slots are provided below, but in general it should be treated as an opaque data structure.

Slots

abm: The AbstractModel-class object used to build the model
dclass: logical TRUE when classification was performed
nlev: For classification, a numeric indicating how many levels were detected
ylevels: For classification, a vector containing the levels
mods: The list of models built
nmods: numeric indication the number of models built
bagmat: A matrix containing which samples were used in which iteration of model building
oopred: The aggregated OOB predictions for all iterations
oopredmat: A matrix of per iteration OOB predictions
accmat: A matrix tracking per iteration accuracy
test_oobpred: The aggregated OOB predictions of the test set
test_oopredmat: A matrix for the test set equivalent to oopredmat
test_accmat: A matrix for the test set equivalent to accmat
Note
The test_items will only make sense if a test set was provided to the call to gensemble.

Author(s)
Peter Werner and Eugene Dubossarsky <gensemble.r@gmail.com>

See Also
gensemble, predict.gensemble

mksampsize
Generate sample size information for use with gensemble

Description
This translates the sampsize argument to gensemble to a form for use internally.

Usage
mksampsize(Y, sampsize = NULL, proportion = FALSE)

Arguments
Y The response vector.
sampsize The desired sample size(s). Can be NULL, a single value, a vector or a list. See the details section for more information.
proportion A logical indicating the values in sampsize represent proportions.

Details
For regression, sampsize indicates how much of the underlying data should be used in the bagged model. It should either be NULL or a single value. If it is NULL, roughly 80

For classification, the internals of gensemble require a list of each class and the size of the sample from each class. If sampsize is NULL, this list will be built using the levels present in Y, and roughly 80

Value
If Y is a factor, will return a list of each class and the number of data points to sample for that class. Otherwise it will return a single value.

Author(s)
Peter Werner <gensemble.r@gmail.com>
# predict.gensemble

**Summary**

Predict method for generalized ensemble methods.

**Description**

Prediction of data using a model built with `gensemble`.

**Usage**

```r
## S3 method for class 'gensemble'
predict(object, X, type = c("prob", "class"), method = c("prob", "vote"), return.all = F, ...)```

**Arguments**

- **object**: An instance of `gensemble-class`.
- **X**: The input predictors.
- **type**: For classification, either probabilities or the class name can be returned.
- **method**: For classification, if method is `prob`, the assigned class will be the highest probability. If method is `vote`, the assigned class will be the class with the highest number of votes across all underlying models.
- **return.all**: For regression, if TRUE, a matrix with the output of each underlying model will be returned in addition to the output vector.
- **...**: Present for compatibility. You could put stuff here but it won’t be used.

**Examples**

```r
# regression
Y <- trees[,3]
# use roughly 80% for each training iteration
mksampsiz(Y)
# the same thing using proportion
mksampsiz(Y, 0.8, TRUE)

# classification
Y <- iris[,5]
# use roughly 80% of each class
mksampsize(Y)
# specify the size of each class in absolute terms
mksampsize(Y, list(setosa=20, versicolor=30, virginica=40))
# use about 70% of each class
mksampsize(Y, 0.7, proportion=TRUE)
# specify the proportion for each class
mksampsize(Y, c(0.5, 0.6, 0.7), proportion=TRUE)
```
Details

Please see \texttt{gensemble} for examples.

Value

The return value will vary depending on classification or regression, and if \texttt{return.all} was \texttt{FALSE} (default) or \texttt{TRUE}.

For classification, the return value will be a matrix of probabilities where each row corresponds to the input \texttt{X} and the columns are the classes and the probabilities aggregated from the underlying models.

For regression, the return value will be a vector of the predictor values. If \texttt{return.all} is \texttt{TRUE}, the return value will be a list. The first item is the vector of predicted values, and the second is a matrix where each row corresponds to the input \texttt{X} and each column is the predicted value generated from each model. For example an input \texttt{X} with 150 rows, using a \texttt{gensemble} with 100 underlying models would return a 150 row x 100 column matrix.

Author(s)

Peter Werner and Eugene Dubossarsky <gensemble.r@gmail.com>

See Also

\texttt{gensemble, gensemble-class}. 
Index

*Topic AbstractModel
   ab.create, 3
   ab.model, 4
   ab.predict, 4
*Topic \textasciitilde kwd1
   ab.arglist, 2
*Topic \textasciitilde kwd2
   ab.arglist, 2
*Topic classes
   AbstractModel-class, 5
   gensemble-class, 10
*Topic gensemble
   ab.create, 3
   ab.model, 4
   ab.predict, 4
   ab.arglist, 2
   ab.create, 2, 3, 4, 5
   ab.model, 2, 3, 4, 5, 6, 8
   ab.predict, 2–4, 4, 6, 8
   AbstractModel-class, 5

cbind, 2

formals, 3

gensemble, 2–6, 7, 10–13
   gensemble-class, 8, 10
mksampsize, 7, 9, 11

predict, 4, 5
predict.gensemble, 9–11, 12
print.gensemble(gensemble), 7