Package 'randomGLM'

October 14, 2022

Version 1.10-1

Date 2022-04-08

Title Random General Linear Model Prediction

Author Lin Song, Peter Langfelder

Maintainer Peter Langfelder <peter.langfelder@gmail.com>

Depends R (>= 4.0.0), MASS, foreach, doParallel,

Imports Hmisc, geometry, survival, matrixStats, parallel

ZipData no

License GPL (>= 2)

Description A bagging predictor based on generalized linear models (GLMs) is implemented. The method is published in
Song, Langfelder and Horvath (2013) <doi:10.1186/1471-2105-14-5>.

URL https://horvath.genetics.ucla.edu/rglm/

NeedsCompilation no

Repository CRAN

Date/Publication 2022-04-10 23:00:02 UTC

R topics documented:

accuracyMeasures *Accuracy measures for a 2x2 confusion matrix or for vectors of predicted and observed values.*

Description

The function calculates various prediction accuracy statistics for predictions of binary or quantitative (continuous) responses. For binary classification, the function calculates the error rate, accuracy, sensitivity, specificity, positive predictive value, and other accuracy measures. For quantitative prediction, the function calculates correlation, R-squared, error measures, and the C-index.

Usage

```
accuracyMeasures(
  predicted,
  observed = NULL,
  type = c("auto", "binary", "quantitative"),
  levels = if (isTRUE(all.equal(dim(predicted), c(2,2)))) colnames(predicted)
    else if (is.factor(predicted))
      sort(unique(c(as.character(predicted), as.character(observed))))
    else sort(unique(c(observed, predicted))),
  negativeLevel = levels[2],
  positiveLevel = levels[1] )
```

Arguments

predicted
either a a 2x2 confusion matrix (table) whose entries contain non-negative integers, or a vector of predicted values. Predicted values can be binary or quantitative (see type below). If a 2x2 matrix is given, it must have valid column and row names that specify the levels of the predicted and observed variables whose counts the matrix is giving (e.g., the function table sets the names appropriately.) If it is a 2x2 table and the table contains non-negative real (non-integer) numbers the function outputs a warning.

observed
if predicted is a vector of predicted values, this (observed) must be a vector of the same length giving the "gold standard" (or observed) values. Ignored if predicted is a 2x2 table.

type
character string specifying the type of the prediction problem (i.e., values in the predicted and observed vectors). The default "auto" decides type automatically: if predicted is a 2x2 table or if the number of unique values in the concatenation of predicted and observed is 2, the prediction problem (type) is assumed to be binary, otherwise it is assumed to be quantitative. Inconsistent specification (for example, when predicted is a 2x2 matrix and type is "quantitative") trigger errors.

levels
a 2-element vector specifying the two levels of binary variables. Only used if type is "binary" (or "auto" that results in the binary type). Defaults to either the column names of the confusion matrix (if the matrix is specified) or to the sorted unique values of observed and opredicted.

`negativeLevel` the binary value (level) that corresponds to the negative outcome. Note that the default is the second of the sorted levels (for example, if levels are 1,2, the default negative level is 2). Only used if `type` is `"binary"` (or `"auto"` that results in the binary type).

`positiveLevel` the binary value (level) that corresponds to the positive outcome. Note that the default is the second of the sorted levels (for example, if levels are 1,2, the default negative level is 2). Only used if `type` is `"binary"` (or `"auto"` that results in the binary type).

Details

The rows of the 2x2 table tab must correspond to a test (or predicted) outcome and the columns to a true outcome ("gold standard"). A table that relates a predicted outcome to a true test outcome is also known as confusion matrix. Warning: To correctly calculate sensitivity and specificity, the positive and negative outcome must be properly specified so they can be matched to the appropriate rows and columns in the confusion table.

Interchanging the negative and positive levels swaps the estimates of the sensitivity and specificity but has no effect on the error rate or accuracy. Specifically, denote by `pos` the index of the positive level in the confusion table, and by `neg` th eindex of the negative level in the confusion table. The function then defines number of true positives=TP=tab[pos, pos], no.false positives =FP=tab[pos, neg], no.false negatives=FN=tab[neg, pos], no.true negatives=TN=tab[neg, neg]. Then Specificity= TN/(FP+TN) Sensitivity= TP/(TP+FN) NegativePredictiveValue= TN/(FN + TN) PositivePredictiveValue= TP/(TP + FP) FalsePositiveRate = 1-Specificity FalseNegativeRate = 1-Sensitivity Power = Sensitivity LikelihoodRatioPositive = Sensitivity / (1-Specificity) LikelihoodRatioNegative = (1-Sensitivity)/Specificity. The naive error rate is the error rate of a constant (naive) predictor that assigns the same outcome to all samples. The prediction of the naive predictor equals the most frequenly observed outcome. Example: Assume you want to predict disease status and 70 percent of the observed samples have the disease. Then the naive predictor has an error rate of 30 percent (since it only misclassifies 30 percent of the healthy individuals).

Value

Data frame with two columns:

`Measure` this column contais character strings that specify name of the accuracy measure.

`Value` this column contains the numeric estimates of the corresponding accuracy measures.

Author(s)

Steve Horvath and Peter Langfelder

References

http://en.wikipedia.org/wiki/Sensitivity_and_specificity

Examples

```
m=100
trueOutcome=sample( c(1,2),m,replace=TRUE)
predictedOutcome=trueOutcome
# now we noise half of the entries of the predicted outcome
predictedOutcome[ 1:(m/2)] =sample(predictedOutcome[ 1:(m/2)] )
tab=table(predictedOutcome, trueOutcome)
accuracyMeasures(tab)

# Same result:
accuracyMeasures(predictedOutcome, trueOutcome)
```

`brainCancer` *The brain cancer data set*

Description

2 sets containing the gene expression profiles of 55 and 65 brain cancer patients respectively.

Usage

```
data(brainCancer)
```

Format

`brainCancer` is a list of 2 components: train and test. "train" is a numeric matrix with 55 samples (rows) across 5000 genes (columns). "test" is a numeric matrix with 65 samples (rows) across the same 5000 genes (columns).

Author(s)

Lin Song, Steve Horvath

Source

Horvath S, Zhang B, Carlson M, Lu K, Zhu S, Felciano R, Laurance M, Zhao W, Shu Q, Lee Y, Scheck A, Liau L, Wu H, Geschwind D, Febbo P, Kornblum H, TF C, Nelson S, Mischel P: Analysis of Oncogenic Signaling Networks in Glioblastoma Identifies ASPM as a Novel Molecular Target. Proc Natl Acad Sci U S A 2006, 103(46):17402-7.

References

Lin Song, Peter Langfelder, Steve Horvath: Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics (2013)

Examples

```
data(brainCancer)
```

mini *An example data set derived from the brain cancer data set*

Description

This example contains one training set, one test set, a corresponding binary outcome and a corresponding continuous outcome. Outcomes are gene traits derived from the brain cancer data set.

Usage

```
data(mini)
```

Format

`mini` is a list of 6 components: x, xtest, yB, yBtest, yC and yCtest. "x" is a numeric matrix with 55 samples (rows) across 4999 genes (columns). "xtest" is a numeric matrix with 65 samples (rows) across the same 4999 genes (columns). They are subsets of the original `brainCancer` data. One gene is left out. "yC" and "yCtest" are continuous outcomes equal to the expression values of the left out gene in the training and test set respectively. "yB" and "yBtest" are binary outcomes dichotomized at the median based on "yC" and "yCtest".

Author(s)

Lin Song, Steve Horvath

Source

Horvath S, Zhang B, Carlson M, Lu K, Zhu S, Felciano R, Laurance M, Zhao W, Shu Q, Lee Y, Scheck A, Liau L, Wu H, Geschwind D, Febbo P, Kornblum H, TF C, Nelson S, Mischel P: Analysis of Oncogenic Signaling Networks in Glioblastoma Identifies ASPM as a Novel Molecular Target. Proc Natl Acad Sci U S A 2006, 103(46):17402-7.

References

Lin Song, Peter Langfelder, Steve Horvath: Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics (2013)

Examples

```
data(mini)
```

predict.randomGLM *Prediction from a random generalized linear model predictor*

Description

Implements a predict method on a previously-constructed random generalized linear model predictor and new data.

Usage

```
## S3 method for class 'randomGLM'
predict(object, newdata, type=c("response", "class"),
                 thresholdClassProb = object$details$thresholdClassProb, ...)
```

Arguments

`object`	a `randomGLM` object such as one returned by `randomGLM`.
`newdata`	specification of test data for which to calculate the prediction.
`type`	type of prediction required. Type "response" gives the fitted probabilities for classification, the fitted values for regression. Type "class" applies only to classification, and produces the predicted class labels.
`thresholdClassProb`	the threshold of predictive probabilities to arrive at classification. Takes values between 0 and 1. Only used for binary outcomes.
`...`	other arguments that may be passed to and from methods. Currently unused.

Details

The function calculates prediction on new test data. It only works if `object` contains the regression models that were used to construct the predictor (see argument `keepModels` of the function `randomGLM`).

If the predictor was trained on a multi-class response, the prediction is applied to each of the representing binary variables (see `randomGLM` for details).

Value

For continuous prediction, the predicted values. For classification of binary response, predicted class when `type="class"`; or a two-column matrix giving the class probabilities if `type="response"`.

If the predictor was trained on a multi-class response, the returned value is a matrix of "cbind"-ed results for the representing individual binary variables (see `randomGLM` for details).

Author(s)

Lin Song, Steve Horvath and Peter Langfelder.

References

Lin Song, Peter Langfelder, Steve Horvath: Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics (2013)

Examples

```
## binary outcome prediction
# data generation
data(iris)
# Restrict data to first 100 observations
iris=iris[1:100,]
# Turn Species into a factor
iris$Species = as.factor(as.character(iris$Species))
# Select a training and a test subset of the 100 observations
set.seed(1)
indx = sample(100, 67, replace=FALSE)
xyTrain = iris[indx,]
xyTest = iris[-indx,]
xTrain = xyTrain[, -5]
yTrain = xyTrain[, 5]

xTest = xyTest[, -5]
yTest = xyTest[, 5]

# predict with a small number of bags
# - normally nBags should be at least 100.
RGLM = randomGLM(
   xTrain, yTrain,
   nCandidateCovariates=ncol(xTrain),
   nBags=30,
   keepModels = TRUE, nThreads = 1)

predicted = predict(RGLM, newdata = xTest, type="class")
table(predicted, yTest)

## continuous outcome prediction

x=matrix(rnorm(100*20),100,20)
y=rnorm(100)

xTrain = x[1:50,]
yTrain = y[1:50]
xTest = x[51:100,]
yTest = y[51:100]

RGLM = randomGLM(
   xTrain, yTrain,
   classify=FALSE,
   nCandidateCovariates=ncol(xTrain),
   nBags=10,
   keepModels = TRUE, nThreads = 1)
```

```
predicted = predict(RGLM, newdata = xTest)
```

randomGLM *Random generalized linear model predictor*

Description

Ensemble predictor comprised of individual generalized linear model predictors.

Usage

```
randomGLM(
  # Input data
  x, y, xtest = NULL,
  weights = NULL,

  # Which columns in x are categorical?
  categoricalColumns = NULL,
  maxCategoricalLevels = 2,

  # Include interactions?
  maxInteractionOrder = 1,
  includeSelfinteractions = TRUE,

  # Prediction type: type can be used to set
  # the prediction type in a simplified way...
  type = c("auto", "linear", "binary", "count", "general", "survival"),

  # classify is retained mostly for backwards compatibility
  classify = switch(type,
    auto = !is.Surv(y) & (is.factor(y) | length(unique(y)) < 4),
    linear = FALSE,
    binary = TRUE ,
    count = FALSE,
    general = FALSE,
    survival = FALSE),

  # family can be used to fine-tune the underlying regression model
  family = switch(type,
    auto = NULL,
    linear = gaussian(link="identity"),
    binary = binomial(link=logit),
    count = poisson(link = "log"),
    general = NULL,
    survival = NULL),
  # Multi-level classification options - only apply to classification
  # with multi-level response
  multiClass.global = TRUE,
  multiClass.pairwise = FALSE,
  multiClass.minObs = 1,
  multiClass.ignoreLevels = NULL,

  # Sampling options
  nBags = 100,
  replace = TRUE,
  sampleBaggingWeights = NULL,
  nObsInBag = if (replace) nrow(x) else as.integer(0.632 * nrow(x)),
  nFeaturesInBag = ceiling(ifelse(ncol(x)<=10, ncol(x),
        ifelse(ncol(x)<=300, (1.0276-0.00276*ncol(x))*ncol(x), ncol(x)/5))),
  minInBagObs = min( max( nrow(x)/2, 5), 2*nrow(x)/3),
  maxBagAttempts = 100*nBags,
  replaceBadBagFeatures = TRUE,

  # Individual ensemble member predictor options
  nCandidateCovariates=50,
  corFncForCandidateCovariates= cor,
  corOptionsForCandidateCovariates = list(method = "pearson", use="p"),
  mandatoryCovariates = NULL,
  interactionsMandatory = FALSE,
  keepModels = is.null(xtest),

  # Miscellaneous options
  thresholdClassProb = 0.5,
  interactionSeparatorForCoefNames = ".times.",
  randomSeed = 12345,
  nThreads = NULL,
  verbose =0 )
```

Arguments

`x` a matrix whose rows correspond to observations (samples) and whose columns correspond to features (also known as covariates or variables). Thus, `x` contains the training data sets.

`y` outcome variable corresponding to the rows of `x`: at this point, one can either use a binary class outcome (factor variable) or a quantitative outcome (numeric variable).

`xtest` an optional matrix of a second data set (referred to as test data set while the data in `x` are interpreted as training data). The number of rows can (and typically will) be different from the number of rows in `x`.

`weights` optional specifications of sample weights for the regression models. Not to be confused with `sampleBaggingWeights` below.

`categoricalColumns`
optional specifications of columns that are to be treated as categorical. If not

given, columns with at most `maxCategoricalLevels` (see below) unique values will be considered categorical.

`maxCategoricalLevels`
: columns with no more than this number of unique values will be considered categorical.

`maxInteractionOrder`
: integer specifying the maximum interaction level. The default is to have no interactions; numbers higher than 1 specify interactions up to that order. For example, 3 means quadratic and cubic interactions will be included. Warning: higher order interactions greatly increase the computation time. We see no benefit of using maxInteractionOrder>2.

`includeSelfinteractions`
: logical: should self-interactions be included?

`type`
: character string specifying the type of the response variable. Recognized values are (unique abbreviations of) `"auto"`, `"linear"`, `"binary"`, `"count"`, `"general"`, and `"survival"`. See Details for what the individual types mean.

`classify`
: logical indicating whether the response is a categorical variable. This argument is present mainly for backwards compatibility; please use `type` above to specify the type of the response variable. If `TRUE` the response `y` will be interpreted as a binary variable and logistic regression will be used. If `FALSE` the response `y` will be interpreted as a quantitative numeric variable and a least squares regression model will be used to arrive at base learners. Multi-level classification is split into a series of binary classification problems according to the `multiClass...` arguments described below.

`family`
: Specification of family (see `family`) for general linear model fitting (see `glm`). Default values are provided for most of the specific types but can be overriden here (for example, if a different link function is desired). There is no default value for `type = "general"` and the user must specify a valid family. In contrast, this argument must be `NULL` when `type = "survival"`.

`multiClass.global`
: for multi-level classification, this logical argument controls whether binary variables of the type "level vs. all others" are included in the series of binary variables to which classification is applied.

`multiClass.pairwise`
: for multi-level classification, this logical argument controls whether binary variables of the type "level A vs. level B" are included in the series of binary variables to which classification is applied.

`multiClass.minObs`
: an integer specifying the minimum number of observations for each level for the level to be considered when creating "level vs. all" and "level vs. level" binary variables.

`multiClass.ignoreLevels`
: optional specifications of the values (levels) of the input response `y` that are to be ignored when constructing level vs. all and level vs. level binary responses. Note that observation with these values will be included in the "all" but will not have their own "level vs. all" variables.

`nBags` number of bags (bootstrap samples) for defining the ensemble predictor, i.e. this also corresponds to the number of individual GLMs.

`replace` logical. If `TRUE` then each bootstrap sample (bag) is defined by sampling with replacement. Otherwise, sampling is carried out without replacement. We recommend to choose `TRUE`.

`sampleBaggingWeights`
weights assigned to each observations (sample) during bootstrap sampling. Default `NULL` corresponds to equal weights.

`nObsInBag` number of observations selected for each bag. Typically, a bootstrap sample (bag) has the same number of observations as in the original data set (i.e. the rows of `x`).

`nFeaturesInBag` number of features randomly selected for each bag. Features are randomly selected without replacement. If there are no interaction terms, then this number should be smaller than or equal to the number of rows of `x`.

`minInBagObs` minimum number of unique observations that constitute a valid bag. If the sampling produces a bag with fewer than this number of unique observations, the bag is discarded and re-sampled again until the number of unique observations is at least `minInBagObs`. This helps prevent too few unique observations in a bag which would lead to problems with model selection.

`maxBagAttempts` Maximum number of bagging attempts.

`replaceBadBagFeatures`
If a feature in a bag contains missing data, should it be replaced?

`nCandidateCovariates`
Positive integer. The number of features that are being considered for forward selection in each GLM (and in each bag). For each bag, the covariates are being chosen according their highest absolute correlation with the outcome. In case of a binary outcome, it is first turned into a binary numeric variable.

`corFncForCandidateCovariates`
the correlation function used to select candidate covariates. Choices include `cor` or biweight midcorrelation, `bicor`, implemented in the package WGCNA. The biweight mid-correlation is a robust alternative to the Pearson correlation.

`corOptionsForCandidateCovariates`
list of arguments to the correlation function. Note that robust correlations are sometimes problematic for binary class outcomes. When using the robust correlation `bicor`, use the argument `"robustY=FALSE"`.

`mandatoryCovariates`
indices of features that are included as mandatory covariates in each GLM model. The default is no mandatory features. This allows the user to "force" variables into each GLM.

`interactionsMandatory`
logical: should interactions of mandatory covariates be mandatory as well? Interactions are only included up to the level specified in `maxInteractionOrder`.

`keepModels` logical: should the regression models for each bag be kept? The models are necessary for future predictions using the `predict` function, predict() generic.

thresholdClassProb
: number in the interval [0,1]. Recommended value 0.5. This parameter is only relevant in case of a binary outcome, i.e. for a logistic regression model. Then this threshold will be applied to the predictive class probabilities to arrive at binary outcome (class outcome).

interactionSeparatorForCoefNames
: a character string that will be used to separate feature names when forming names of interaction terms. This is only used when interactions are actually taken into account (see maxInteractionLevel above) and only affects coefficient names in models and columns names in returned featuresInForwardRegression (see output value below). We recommend setting it so the interaction separator does not conflict with any feature name since this may improve interpretability of the results.

randomSeed
: NULL or integer. The seed for the random number generator. If NULL, the seed will not be set. If non-NULL and the random generator has been initialized prior to the function call, the latter's state is saved and restored upon exit.

nThreads
: number of threads (worker processes) to perform the calculation. If not given, will be determined automatically as the number of available cores if the latter is 3 or less, and number of cores minus 1 if the number of available cores is 4 or more. Invalid entries (missing value, zero or negative values etc.) are changed to 1, with a warning.

verbose
: value 0 or 1 which determines the level of verbosity. Zero means silent, 1 reports the bag number the function is working on. At this point verbose output only works if nThreads=1

Details

The function randomGLM can be used to predict a variety of different types of outcomes. The outcome type is specified by the argument type as follows:

If type = "auto", the function will attempt to determine the response type automatically. If the response is not a Surv object and is a factor or has 3 or fewer unique values, it is assumed to be categorical. If the number of unique values is 2, the function uses logistic regression; for categorical responses with more 3 or more possible values, see below.

If type = "linear", the responses is assumed to be numeric with Gaussian errors, and the function will use linear regression.

If type = "binary", the response is assumed to be categorical (at this point not necessarily binary but that may change in the future). If the response has 2 levels, logistic regression (binomial family with the logit link) is used. If the response has more than 2 levels, see below.

If type = "count", the response is assumed to represent counts with Poisson-distributed errors, and Poisson regression (poisson family with the logarithmic link) is used.

If type = "general", the function does not make assumption about the response type and the user must specify an appropriate family (see stats{family}).

If type = "survival", the function assumes the response is a censored time, that is a Surv object. In this case the argument family must be NULL (the default) and the function uses Cox proportional hazard regression implemented in function coxph

The function proceeds along the following steps:

Step 1 (bagging): nBags bootstrapped data sets are being generated based on random sampling from the original training data set (x,y). If a bag contains less than minInBagObs unique observations or it contains all observations, it is discarded and re-sampled again.

Step 2 (random subspace): For each bag, nFeaturesInBag features are randomly selected (without replacement) from the columns of x. Optionally, interaction terms between the selected features can be formed (see the argument maxInteractionOrder).

Step 3 (feature ranking): In each bag, features are ranked according to their correlation with the outcome measure. Next the top nCandidateCovariates are being considered for forward selection in each GLM (and in each bag).

Step 4 (forward selection): Forward variable selection is employed to define a multivariate GLM model of the outcome in each bag.

Step 5 (aggregating the predictions): Prediction from each bag are aggregated. In case, of a quantitative outcome, the predictions are simply averaged across the bags.

Generally, nCandidateCovariates>100 is not recommended, because the forward selection process is time-consuming. If arguments "nBags=1, replace=FALSE, nObsInBag=nrow(x)" are used, the function becomes a forward selection GLM predictor without bagging.

Classification of multi-level categorical responses is performed indirectly by turning the single multi-class response into a set of binary variables. The set can include two types of binary variables: Level vs. all others (this binary variable is 1 when the original response equals the level and zero otherwise), and level A vs. level B (this binary variable is 0 when the response equals level A, 1 when the response equals level B, and NA otherwise). For example, if the input response y contains observations with values (levels) "A", "B", "C", the binary variables will have names "all.vs.A" (1 means "A", 0 means all others), "all.vs.B", "all.vs.C", and optionally also "A.vs.B" (0 means "A", 1 means "B", NA means neither "A" nor "B"), "A.vs.C", and "B.vs.C". Note that using pairwise level vs. level binary variables be very time-consuming since the number of such binary variables grows quadratically with the number of levels in the response. The user has the option to limit which levels of the original response will have their "own" binary variables, by setting the minimum observations a level must have to qualify for its own binary variable, and by explicitly enumerating levels that should not have their own binary variables. Note that such "ignored" levels are still included on the "all" side of "level vs. all" binary variables.

At this time the predictor does not attempt to summarize the binary variable classifications into a single multi-level classification.

Training this predictor on data with fewer than 8 observations is not recommended (and the function will warn about it). Due to the bagging step, the number of unique observations in each bag is less than the number of observations in the input data; the low number of unique observations can (and often will) lead to an essentially perfect fit which makes it impossible to perfrom meaningful stepwise model selection.

Feature names: In general, the column names of input x are assumed to be the feature names. If x has no column names (i.e., colnames(x) is NULL), stadard column names of the form "F01", "F02", ... are used. If x has non-NULL column names, they are turned into valid and unique names using the function make.names. If the function make.names returns names that are not the same as the column names of x, the component featureNamesChanged will be TRUE and the component nameTranslationTable contains the information about input and actual used feature names. The feature names are used as predictor names in the individual models in each bag.

Value

The function returns an object of class randomGLM. For continuous prediction or two-level classification, this is a list with the following components:

predictedOOB
: the continuous prediction (if classify is FALSE) or predicted classification (if classify is TRUE) of the input data based on out-of-bag samples.

predictedOOB.response
: In case of a binary outcome, this is the predicted probability of each outcome specified by y based on out-of-bag samples. In case of a continuous outcome, this is the predicted value based on out-of-bag samples (i.e., a copy of predictedOOB).

predictedTest.cont
: if test set is given, the predicted probability of each outcome specified by y for test data for binary outcomes. In case of a continuous outcome, this is the test set predicted value.

predictedTest
: if test set is given, the predicted classification for test data. Only for binary outcomes.

candidateFeatures
: candidate features in each bag. A list with one component per bag. Each component is a matrix with maxInteractionOrder rows and nCandidateCovariates columns. Each column represents one interaction obtained by multiplying the features indicated by the entries in each column (0 means no feature, i.e. a lower order interaction).

featuresInForwardRegression
: features selected by forward selection in each bag. A list with one component per bag. Each component is a matrix with maxInteractionOrder rows. Each column represents one interaction obtained by multiplying the features indicated by the entries in each column (0 means no feature, i.e. a lower order interaction). The column names contain human-readable names for the terms.

coefOfForwardRegression
: coefficients of forward regression. A list with one component per bag. Each component is a vector giving the coefficients of the model determined by forward selection in the corresponding bag. The order of the coefficients is the same as the order of the terms in the corresponding component of featuresInForwardRegression.

interceptOfForwardRegression
: a vector with one component per bag giving the intercept of the regression model in each bag.

bagObsIndx
: a matrix with nObsInBag rows and nBags columns, giving the indices of observations selected for each bag.

timesSelectedByForwardRegression
: a matrix of maxInteractionOrder rows and number of features columns. Each entry gives the number of times the corresponding feature appeared in a predictor model at the corresponding order of interactions. Interactions where a single feature enters more than once (e.g., a quadratic interaction of the feature with itself) are counted once.

models
: the regression models for each bag. Predictor features in each bag model are named using their

`featureNamesChanged`
: logical indicating whether feature names were copied verbatim from column names of `x` (`FALSE`) or whether they had to be changed to make them valid and unique names (`TRUE`).

`nameTranslationTable`
: only present if above `featureNamesChanged` is `TRUE`. A data frame with three columns and one row per input feature (column of input `x`) giving the feature number, original feature name, and modified feature name that is used for model fitting.

In addition, the output value contains a copy of several input arguments. These are included to facilitate prediction using the `predict` method. These returned values should be considered undocumented and may change in the future.

In the multi-level classification classification case, the returned list (still considered a valid `randomGLM` object) contains the following components:

`binaryPredictors`
: a list with one component per binary variable, containing the `randomGLM` predictor trained on that binary variable as the response. The list is named by the corresponding binary variable. For example, if the input response `y` contains observations with values (levels) "A", "B", "C", the binary variables (and components of this list) will have names "all.vs.A" (1 means "A", 0 means all others), "all.vs.B", "all.vs.C", and optionally also "A.vs.B" (0 means "A", 1 means "B", NA means neither "A" nor "B"), "A.vs.C", and "B.vs.C".

`predictedOOB`
: a matrix in which columns correspond to the binary variables and rows to samples, containing the predicted binary classification for each binary variable. Columns names and meaning of 0 and 1 are described above.

`predictedOOB.response`
: a matrix with two columns per binary variable, giving the class probabilities for each of the two classes in each binary variables. Column names contain the variable and class names.

`levelMatrix`
: a character matrix with two rows and one column per binary variable, giving the level corresponding to value 0 (row 1) and level corresponding to value 1 (row 2). This encodes the same information as the names of the `binaryPredictors` list but in a more programmer-friendly way.

If input `xTest` is non-NULL, the components `predictedTest` and `predictedTest.response` contain test set predictions analogous to `predictedOOB` and `predictedOOB.response`.

Author(s)

Lin Song, Steve Horvath, Peter Langfelder. The function makes use of the `glm` function and other standard R functions.

References

Lin Song, Peter Langfelder, Steve Horvath: Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics 2013 Jan 16;14:5. doi: 10.1186/1471-2105-14-5.

Examples

```
## binary outcome prediction
# data generation
data(iris)
# Restrict data to first 100 observations
iris=iris[1:100,]
# Turn Species into a factor
iris$Species = as.factor(as.character(iris$Species))
# Select a training and a test subset of the 100 observations
set.seed(1)
indx = sample(100, 67, replace=FALSE)
xyTrain = iris[indx,]
xyTest = iris[-indx,]
xTrain = xyTrain[, -5]
yTrain = xyTrain[, 5]

xTest = xyTest[, -5]
yTest = xyTest[, 5]

# predict with a small number of bags
# - normally nBags should be at least 100.
RGLM = randomGLM(
   xTrain, yTrain,
   xTest,
   nCandidateCovariates=ncol(xTrain),
   nBags=30, nThreads = 1)

yPredicted = RGLM$predictedTest
table(yPredicted, yTest)


## continuous outcome prediction

x=matrix(rnorm(100*20),100,20)
y=rnorm(100)

xTrain = x[1:50,]
yTrain = y[1:50]
xTest = x[51:100,]
yTest = y[51:100]

RGLM = randomGLM(
  xTrain, yTrain,
  xTest, classify=FALSE,
  nCandidateCovariates=ncol(xTrain),
  nBags=10,
  keepModels = TRUE, nThreads = 1)
```

thinRandomGLM *Random generalized linear model predictor thinning*

Description

This function allows the user to define a "thinned" version of a random generalized linear model predictor by focusing on those features that occur relatively frequently.

Usage

```
thinRandomGLM(rGLM, threshold)
```

Arguments

`rGLM`	a `randomGLM` object such as one returned by `randomGLM`.
`threshold`	integer specifying the minimum of times a feature was selected across the bags in `rGLM` for the feature to be kept. Note that only features selected `threshold +1` times and more are retained. For the purposes of this count, appearances in interactions are not counted. Features that appear `threshold` times or fewer are removed from the underlying regression models when the models are re-fit.

Details

The function "thins out" (reduces) a previously-constructed random generalized linear model predictor by removing rarely selected features and refitting each (generalized) linear model (GLM). Each GLM (per bag) is refit using only those features that occur more than `threshold` times across the `nBags` number of bags. The occurrence count excludes interactions (in other words, the threshold will be applied to the first row of `timesSelectedByForwardRegression`).

Value

The function returns a valid `randomGLM` object (see `randomGLM` for details) that can be used as input to the predict() method (see `predict.randomGLM`). The returned object contains a copy of the input `rGLM` in which the following components were modified:

`predictedOOB`
: the updated continuous prediction (if `classify` is `FALSE`) or predicted classification (if `classify` is `TRUE`) of the input data based on out-of-bag samples.

`predictedOOB.response`
: In case of a binary outcome, the updated predicted probability of each outcome specified by `y` based on out-of-bag samples. In case of a continuous outcome, this is the predicted value based on out-of-bag samples (i.e., a copy of `predictedOOB`).

`featuresInForwardRegression`
: features selected by forward selection in each bag. A list with one component per bag. Each component is a matrix with `maxInteractionOrder` rows. Each column represents one interaction obtained by multiplying the features indicated by the entries in each column (0 means no feature, i.e. a lower order interaction).

`coefOfForwardRegression`
: coefficients of forward regression. A list with one component per bag. Each component is a vector giving the coefficients of the model determined by forward selection in the corresponding bag. The order of the coefficients is the same as the order of the terms in the corresponding component of `featuresInForwardRegression`.

`interceptOfForwardRegression`
: a vector with one component per bag giving the intercept of the regression model in each bag.

`timesSelectedByForwardRegression`
: a matrix of `maxInteractionOrder` rows and number of features columns. Each entry gives the number of times the corresponding feature appeared in a predictor model at the corresponding order of interactions. Interactions where a single feature enters more than once (e.g., a quadratic interaction of the feature with itself) are counted once.

`models`
: the "thinned" regression models for each bag.

Author(s)

Lin Song, Steve Horvath, Peter Langfelder

References

Lin Song, Peter Langfelder, Steve Horvath: Random generalized linear model: a highly accurate and interpretable ensemble predictor. BMC Bioinformatics (2013)

Examples

```
## binary outcome prediction
# data generation
data(iris)
# Restrict data to first 100 observations
iris=iris[1:100,]
# Turn Species into a factor
iris$Species = as.factor(as.character(iris$Species))
# Select a training and a test subset of the 100 observations
set.seed(1)
indx = sample(100, 67, replace=FALSE)
xyTrain = iris[indx,]
xyTest = iris[-indx,]
xTrain = xyTrain[, -5]
yTrain = xyTrain[, 5]

xTest = xyTest[, -5]
yTest = xyTest[, 5]

# predict with a small number of bags - normally nBags should be at least 100.
RGLM = randomGLM(
   xTrain, yTrain,
   nCandidateCovariates=ncol(xTrain),
   nBags=30,
   keepModels = TRUE, nThreads = 1)
table(RGLM$timesSelectedByForwardRegression[1, ])
# 0  7 23
# 2  1  1
```

```
thinnedRGLM = thinRandomGLM(RGLM, threshold=7)
predicted = predict(thinnedRGLM, newdata = xTest, type="class")
predicted = predict(RGLM, newdata = xTest, type="class")
```

Index