Package 'horserule'

March 21, 2018

Type Package

Title Flexible Non-Linear Regression with the HorseRule Algorithm

Version 1.0.0

Date 2018-03-21

Author Malte Nalenz <malte.nlz@googlemail.com>, Mattias Villani <mattias.villani@liu.se>

Maintainer Malte Nalenz <malte.nlz@googlemail.com>

Description
Implementation of the HorseRule model a flexible tree based Bayesian regression method for linear and nonlinear regression described in Nalenz & Villani (2017) <arXiv:1702.05008>.

Imports mvnfast, MASS, randomForest, gbm, inTrees, ggplot2, grDevices, Rdpack, RColorBrewer, stats

License GPL-3

RoxygenNote 6.0.1

RdMacros Rdpack

NeedsCompilation no

Repository CRAN

Date/Publication 2018-03-21 14:59:48 UTC

R topics documented:

convergence_plot *convergence_plot*

Description

Can be used to check model convergence.

Usage

```
convergence_plot(model, Xtest, ytest, burnin = 0)
```

Arguments

model	list containing a model of class "hs_rulefit".
Xtest	Out of bag sample to check error.
ytest	response of test data.
burnin	Number of samples disregarded as burnin.

Details

Convergence is checked by the convergence of the prediction error on unseen test data, to find a suitable number of iterations, in the spirit of gradient boosting. To check convergence on the Training data just use training X and y instead of Xtest and ytest.

Examples

```
library(MASS)
data(Boston)
#Split in train and test data
N = nrow(Boston)
train = sample(1:N, 400)
Xtrain = Boston[train,-14]
ytrain = Boston[train, 14]
Xtest = Boston[-train, -14]
ytest = Boston[-train, 14]

hrres = HorseRuleFit(X = Xtrain, y=ytrain,
                     thin=1, niter=100, burnin=10,
                     L=5, S=6, ensemble = "both", mix=0.3, ntree=100,
                     intercept=FALSE, linterms=1:13, ytransform = "log",
                     alpha=1, beta=2, linp = 1, restricted = 0)

#Check the model convergence out of sample
convergence_plot(hrres, Xtest, ytest, burnin = 10)
```

HorseRuleFit *Horseshoe RuleFit*

Description

Fits the Horseshoe Rulefit model described in Nalenz M and Villani M (2017). "TREE ENSEMBLES WITH RULE STRUCTURED HORSESHOE REGULARIZATION." In *https://arxiv.org* . (https://arxiv.org/abs/1702.05008)

Usage

```
HorseRuleFit(X = NULL, y = NULL, Xtest = NULL, ytest = NULL,
  niter = 1000, burnin = 100, thin = 1, restricted = 0.001,
  shrink.prior = "HS", beta = 2, alpha = 1, linp = 1, ensemble = "RF",
  L = 4, S = 6, ntree = 250, minsup = 0.025, mix = 0.5,
  linterms = NULL, intercept = F, ytransform = "linear")
```

Arguments

X	A matrix or dataframe containing the predictor variables to be used.
y	A vector containing the response variables. If numeric regression is performed and classification otherwise.
Xtest	optional matrix or dataframe containing predictor variables of test set.
ytest	optional vector containing the response values of the test set.
niter	number of iterations for the horseshoe sampling.
burnin	number of initial samples to be disregarded as burnin.
thin	thinning parameter.
restricted	Threshold for restricted Gibbs sampling. In each iteration only coefficients with scale > restricted are updated. Set restricted = 0 for unrestricted Gibbs sampling.
shrink.prior	Specifies the shrinkage prior to be used for regularization. Currently the options "HS" and "HS+" for the Horseshoe+ are supported.
beta	Hyperparameter to control the extra shrinkage on the rule complexity meassured as the rule length.
alpha	Hyperparameter to control the extra shrinkage on the rules that cover only few observations. Set alpha = beta = 0 for the standard horseshoe without rule structure prior.
linp	Hyperparameter to control prior shrinkage of linear terms. Set linp > 1 if strong linear effects are assumed.
ensemble	Which ensemble method should be used to generate the rules? Options are "RF","GBM" or "both".
L	Parameter controling the complexity of the generated rules. Higher values lead to more complex rules.

S	Parameter controlling the minimum number of observations in the tree growing process.
ntree	Number of trees in the ensemble step from which the rules are extracted.
minsup	Rules with support < minsup are removed. Can be used to prevent overfitting.
mix	If ensemble = "both" mix*ntree are generated via random forest and (1-mix)*ntree trees via gradient boosting.
linterms	specifies the columns in X which should be included as linear terms in the hs rulefit model. Specified columns need to be numeric. Categorical variables have to be transformed (e.g. to dummies) before included as linear effects.
intercept	If TRUE an intercept is included. Note that the y by default is shifted to have 0 mean therefor not necessary for regression. For classification highly recommended.
ytransform	Choose "log" for logarithmic transform of y.

Value

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

bhat	Posterior mean of the regression coefficients.
postdraws	List contraining the Posterior samples of the regression coefficients, error variance sigma and shrinkage tau.
rules	Vector containing the decision rules.
df	Matrix containing original training data and the decision rule covariates (normalized).
y	Response in train data.
prior	Vector rule structure prior for the individual rules.
modelstuff	List contraining the parameters used and values used for the normalization (means and sds).
pred	If Test data was supplied, gives back the predicted values.
err	If y-test was also supplies additionally gives back a test error score (RMSE for regression, Missclassificationrate for Classficitaion).

Examples

```
library(MASS)
library(horserule)
data(Boston)
# Split in train and test data
N = nrow(Boston)
train = sample(1:N, 400)
Xtrain = Boston[train,-14]
ytrain = Boston[train, 14]
Xtest = Boston[-train, -14]
ytest = Boston[-train, 14]

# Run the HorseRuleFit with 100 trees
```

```
# Increase Number of trees and the number of posterior samples for better modelfit
hrres = HorseRuleFit(X = Xtrain, y=ytrain,
                     thin=1, niter=100, burnin=10,
                     L=5, S=6, ensemble = "both", mix=0.3, ntree=100,
                     intercept=FALSE, linterms=1:13, ytransform = "log",
                     alpha=1, beta=2, linp = 1, restricted = 0)

# Calculate the error
pred = predict(hrres, Xtest, burnin=100, postmean=TRUE)
sqrt(mean((pred-ytest)^2))

# Look at the most important rules/linear effects.
importance_hs(hrres)

# Look at the input variable importance.
Variable_importance(hrres, var_names=colnames(Xtrain))
```

---

hs *Horseshoe regression Gibbs-sampler*

Description

Generates posterior samples using the horseshoe prior. The Gibbs sampling method from Makalic E and Schmidt DF (2016). "A Simple Sampler for the Horseshoe Estimator." *IEEE Signal Process. Lett.*, **23**(1), pp. 179–182. is used to generate the posterior samples.

Usage

```
hs(X, y, niter = 1000, hsplus = F, prior = NULL, thin = 1,
  restricted = 0)
```

Arguments

X	A matrix containing the predictor variables to be used.
y	The vector of numeric responses.
niter	Number of posterior samples.
hsplus	If "hsplus=T" the horseshoe+ extension will be used.
prior	Prior for the individual predictors. If all 1 a standard horseshoe model is fit.
thin	If > 1 thinning is performed to reduce autocorrelation.
restricted	Threshold for restricted Gibbs sampling. In each iteration only coefficients with scale > restricted are updated. Set restricted = 0 for unrestricted Gibbs sampling.

Value

A list containing the posterior samples of the following parameters:

beta	Matrix containing the posterior samples for the regression coefficients.
sigma	Vector contraining the Posterior samples of the error variance.
tau	Vector contraining the Posterior samples of the overall shrinkage.
lambda	Matrix containing the posterior samples for the individual shrinkage parameter.

Examples

```
x = matrix(rnorm(1000), ncol=10)
y = apply(x,1,function(x)sum(x[1:5])+rnorm(1))
hsmod = hs(X=x, y=y, niter=100)
```

---

importance_hs *Most important Rules/terms*

Description

Produces a table containing the most important Rules or linear terms

Usage

```
importance_hs(model, k = 10)
```

Arguments

model	list containing a model of class "HorseRuleFit".
k	number of most important rules to be shown in the table.

Examples

```
library(MASS)
library(horserule)
data(Boston)
# Split in train and test data
N = nrow(Boston)
train = sample(1:N, 400)
Xtrain = Boston[train,-14]
ytrain = Boston[train, 14]
Xtest = Boston[-train, -14]
ytest = Boston[-train, 14]
hrres = HorseRuleFit(X = Xtrain, y=ytrain,
                     thin=1, niter=100, burnin=10,
                     L=5, S=6, ensemble = "both", mix=0.3, ntree=100,
                     intercept=FALSE, linterms=1:13, ytransform = "log",
                     alpha=1, beta=2, linp = 1, restricted = 0)

#Create an importance table containing the 10 most important rules and linear terms
importance_hs(hrres, k=10)
```

predict.HorseRulemodel *predict.hs*

Description

Predict unseen data with the horseshoe model.

Usage

```
## S3 method for class 'HorseRulemodel'
predict(object, newdata, burnin = 100,
  postmean = TRUE, ...)
```

Arguments

object	list containing a model of class "hs_rulefit".
newdata	Dataframe containing the unseen data to predict.
burnin	Number of samples that is disregarded as burnin. Increase number in case of slow convergence.
postmean	If true returns the Predictive-Posterior mean value. If False returns the full predictive posterior distribution.
...	additional arguments

Value

Returns the predictive posterior distribution matrix. Column i contains the predictive posterior of observation i.

Examples

```
x = matrix(rnorm(1000), ncol=10)
y = apply(x,1,function(x)sum(x[1:5])+rnorm(1))
hrresmod = HorseRuleFit(X=x, y=y, niter=100, burnin=10)
#predict training data to obtain the fitted values
predict(hrresmod, x, burnin=10)
```

ruleheat *RuleHeatmap*

Description

Produces a heatmap that allows to identify what observations are covered by the most important decision rules. Details can be found in Nalenz & Villani (2017).

Usage

```
ruleheat(model, k)
```

Arguments

model	list containing a model of class "HorseRuleFit".

k	number of most important rules to be shown in the RuleHeat plot. library(MASS) data(Boston) # Split in train and test data N = nrow(Boston) train = sample(1:N, 400) Xtrain = Boston[train,-14] ytrain = Boston[train, 14] Xtest = Boston[-train, -14] ytest = Boston[-train, 14]
	hrres = HorseRuleFit(X = Xtrain, y=ytrain, thin=1, niter=200, burnin=10, L=5, S=6, ensemble = "both", mix=0.3, ntree=100, intercept=FALSE, linterms=1:13, ytransform = "log", alpha=1, beta=2, linp = 1, restricted = 0)
	#Create a ruleheat plot. ruleheat(hrres = 10)

Variable_importance *Variable Importance plot*

Description

Creates a input variable importance plot

Usage

```
Variable_importance(model, top = NULL, var_names = NULL)
```

Arguments

model	list containing a model of class "hs_rulefit".
top	If a integer number is given only shows the top most important variables in the plot
var_names	optional vector with the variable names to be shown in plot.

Examples

```
#Fit HorseRuleFit
x = matrix(rnorm(5000), ncol=10)
y = apply(x,1,function(x)sum(x[1:5])+rnorm(1))
hrres = HorseRuleFit(X = x, y=y,
                     thin=1, niter=100, burnin=10,
                     L=5, S=6, ensemble = "both", mix=0.3, ntree=100,
                     intercept=FALSE, linterms=1:10,
                     alpha=1, beta=2, linp = 1, restricted = 0)
Variable_importance(hrres)
```

Index