Package 'bark'

May 7, 2026

Type Package

Title Bayesian Additive Regression Kernels

Version 1.0.5

Date 2024-10-05

Description Bayesian Additive Regression Kernels (BARK) provides an implementation for non-parametric function estimation using Levy Random Field priors for functions that may be represented as a sum of additive multivariate kernels. Kernels are located at every data point as in Support Vector Machines, however, coefficients may be heavily shrunk to zero under the Cauchy process prior, or even, set to zero. The number of active features is controlled by priors on precision parameters within the kernels, permitting feature selection. For more details see Ouyang, Z (2008) ``Bayesian Additive Regression Kernels'', Duke University. PhD dissertation, Chapter 3 and Wolpert, R. L, Clyde, M.A, and Tu, C. (2011) "Stochastic Expansions with Continuous Dictionaries Levy Adaptive Regression Kernels, Annals of Statistics Vol (39) pages 1916-1962 <doi:10.1214/11-AOS889>.

License GPL (>= 3)

URL https://www.R-project.org, https://github.com/merliseclyde/bark

BugReports https://github.com/merliseclyde/bark/issues

Depends R (>= 3.5.0)

Suggests BART, e1071, fdm2id, rmarkdown, knitr, roxygen2, testthat, covr

LazyData yes

Repository CRAN

NeedsCompilation yes

ByteCompile yes

Encoding UTF-8

RoxygenNote 7.3.2

Language en-US

VignetteBuilder knitr

Author Merlise Clyde [aut, cre, ths] (ORCID=0000-0002-3595-1872),
Zhi Ouyang [aut],
Robert Wolpert [ctb, ths]

Maintainer Merlise Clyde <clyde@duke.edu>

Date/Publication 2024-10-05 22:40:28 UTC

Contents

`bark-package` *bark: Bayesian Additive Regression Trees*

Description

Implementation of Bayesian Additive Regression Kernels with Feature Selection for Nonparametric Regression for Gaussian regression and classification for binary Probit models

_PACKAGE

Details

BARK is a Bayesian *sum-of-kernels* model or because of the Bayesian priors is a Bayesian Additive Regression Kernel model.
For numeric response y, we have $y = f(x) + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$.
For a binary response y, $P(Y = 1|x) = F(f(x))$, where F denotes the standard normal cdf (probit link). In both cases, f is the sum of many Gaussian kernel functions. The goal is to have very flexible inference for the unknown function f. bark uses an approximated Cauchy process as the prior distribution for the unknown function f. Feature selection can be achieved through the inference on the scale parameters in the Gaussian kernels. BARK accepts four different types of prior distributions through setting values for `selection` (TRUE or FALSE), which allows scale parameters for some variables to be set to zero, removing the variables from the kernels `selection = TRUE`; this enables either soft shrinkage or hard shrinkage for the scale parameters. The input `common_lambdas` (TRUE or FALSE) specifies whether a common scale parameter should be used for all predictors (TRUE) or if FALSE allows the scale parameters to differ across all variables in the kernel.

References

Ouyang, Zhi (2008) Bayesian Additive Regression Kernels. Duke University. PhD dissertation, Chapter 3.

See Also

Other bark functions: bark(), bark-package-deprecated, sim_Friedman1(), sim_Friedman2(), sim_Friedman3(), sim_circle()

Examples

```
# Simulate regression example
# Friedman 2 data set, 200 noisy training, 1000 noise free testing
# Out of sample MSE in SVM (default RBF): 6500 (sd. 1600)
# Out of sample MSE in BART (default):    5300 (sd. 1000)
traindata <- sim_Friedman2(200, sd=125)
testdata <- sim_Friedman2(1000, sd=0)
fit.bark.d <- bark(y ~ ., data = data.frame(traindata),
                   testdata = data.frame(testdata),
                   classification = FALSE,
                   selection = FALSE,
                   common_lambdas = TRUE)
boxplot(as.data.frame(fit.bark.d$theta.lambda))
mean((fit.bark.d$yhat.test.mean-testdata$y)^2)
# Simulate classification example
# Circle 5 with 2 signals and three noisy dimensions
# Out of sample erorr rate in SVM (default RBF): 0.110 (sd. 0.02)
# Out of sample error rate in BART (default):    0.065 (sd. 0.02)
traindata <- sim_circle(200, dim=5)
testdata <- sim_circle(1000, dim=5)
fit.bark.se <- bark(y ~ ., data= data.frame(traindata),
                    testdata= data.frame(testdata),
                    classification=TRUE,
                    selection=TRUE,
                    common_lambdas = FALSE)

boxplot(as.data.frame(fit.bark.se$theta.lambda))
mean((fit.bark.se$yhat.test.mean>0)!=testdata$y)
```

---

banknotes *Swiss Bank Notes*

Description

This data set contains six measurements on 100 genuine and 100 fradulent old Swiss banknotes

Usage

```
data(banknotes)
```

Format

a dataframe with the following variables:

Status the status of the banknote: genuine or counterfeit

Length Length of bill (mm)

Left Width of left edge (mm)

Right Width of right edge (mm)

Bottom Bottom margin width (mm)

Top Top margin width (mm)

Diagonal Length of diagonal (mm)

Source

Flury, B. and Riedwyl, H. (1988). Multivariate Statistics: A practical approach. London: Chapman & Hall, Tables 1.1 and 1.2, pp. 5-8.

---

bark *Nonparametric Regression using Bayesian Additive Regression Kernels*

Description

BARK is a Bayesian *sum-of-kernels* model.
For numeric response y, we have $y = f(x) + \epsilon$, where $\epsilon \sim N(0, \sigma^2)$.
For a binary response y, $P(Y = 1|x) = F(f(x))$, where F denotes the standard normal cdf (probit link).
In both cases, f is the sum of many Gaussian kernel functions. The goal is to have very flexible inference for the unknown function f. BARK uses an approximation to a Cauchy process as the prior distribution for the unknown function f.

Feature selection can be achieved through the inference on the scale parameters in the Gaussian kernels. BARK accepts four different types of prior distributions, *e*, *d*, enabling either soft shrinkage or *se*, *sd*, enabling hard shrinkage for the scale parameters.

Usage

```
bark(
  formula,
  data,
  subset,
  na.action = na.omit,
  testdata = NULL,
```

```
  selection = TRUE,
  common_lambdas = TRUE,
  classification = FALSE,
  keepevery = 100,
  nburn = 100,
  nkeep = 100,
  printevery = 1000,
  keeptrain = FALSE,
  verbose = FALSE,
  fixed = list(),
 tune = list(lstep = 0.5, frequL = 0.2, dpow = 1, upow = 0, varphistep = 0.5, phistep =
   1),
  theta = list()
)
```

Arguments

formula	model formula for the model with all predictors, Y ~ X. The X variables will be centered and scaled as part of model fitting.
data	a data frame. Factors will be converted to numerical vectors based on the using 'model.matrix'.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs. The default is "na.omit".
testdata	Dataframe with test data for out of sample prediction. Should have same structure as data.
selection	Logical variable indicating whether variable dependent kernel parameters λ may be set to zero in the MCMC; default is TRUE.
common_lambdas	Logical variable indicating whether kernel parameters λ should be predictor specific or common across predictors; default is TRUE. Note if *common_lambdas = TRUE* and *selection = TRUE* this applies just to the non-zero $lambda_j$.
classification	TRUE/FALSE logical variable, indicating a classification or regression problem.
keepevery	Every keepevery draw is kept to be returned to the user
nburn	Number of MCMC iterations (nburn*keepevery) to be treated as burn in.
nkeep	Number of MCMC iterations kept for the posterior inference. nkeep*keepevery iterations after the burn in.
printevery	As the MCMC runs, a message is printed every printevery draws.
keeptrain	Logical, whether to keep results for training samples.
verbose	Logical, whether to print out messages
fixed	A list of fixed hyperparameters, using the default values if not specified. alpha = 1: stable index, must be 1 currently.

eps = 0.5: approximation parameter.
gam = 5: intensity parameter.
la = 1: first argument of the gamma prior on kernel scales.
lb = 2: second argument of the gamma prior on kernel scales.
pbetaa = 1: first argument of the beta prior on plambda.
pbetab = 1: second argument of the beta prior on plambda.
n: number of training samples, automatically generates.
p: number of explanatory variables, automatically generates.
meanJ: the expected number of kernels, automatically generates.

`tune` A list of tuning parameters, not expected to change.
lstep: the stepsize of the lognormal random walk on lambda.
frequL: the frequency to update L.
dpow: the power on the death step.
upow: the power on the update step.
varphistep: the stepsize of the lognormal random walk on varphi.
phistep: the stepsize of the lognormal random walk on phi.

`theta` A list of the starting values for the parameter theta, use defaults if nothing is given.

Details

BARK is implemented using a Bayesian MCMC method. At each MCMC interaction, we produce a draw from the joint posterior distribution, i.e. a full configuration of regression coefficients, kernel locations and kernel parameters.

Thus, unlike a lot of other modelling methods in R, we do not produce a single model object from which fits and summaries may be extracted. The output consists of values $f^*(x)$ (and σ^* in the numeric case) where * denotes a particular draw. The x is either a row from the training data (x.train)

Value

`bark` returns an object of class 'bark' with a list, including:

`call` the matched call

`fixed` Fixed hyperparameters

`tune` Tuning parameters used

`theta.last` The last set of parameters from the posterior draw

`theta.nvec` A matrix with nrow(x.train)+1 rows and (nkeep) columns, recording the number of kernels at each training sample

`theta.varphi` A matrix with nrow(x.train) +1 rows and (nkeep) columns, recording the precision in the normal gamma prior distribution for the regression coefficients

`theta.beta` A matrix with nrow(x.train)+1 rows and (nkeep) columns, recording the regression coefficients

`theta.lambda` A matrix with ncol(x.train) rows and (nkeep) columns, recording the kernel scale parameters

`thea.phi` The vector of length nkeep, recording the precision in regression Gaussian noise (1 for the classification case)

`yhat.train` A matrix with nrow(x.train) rows and (nkeep) columns. Each column corresponds to a draw f^* from the posterior of f and each row corresponds to a row of x.train. The (i, j) value is $f^*(x)$ for the j^{th} kept draw of f and the i^{th} row of x.train.
For classification problems, this is the value of the expectation for the underlying normal random variable.
Burn-in is dropped

`yhat.test` Same as yhat.train but now the x's are the rows of the test data; NULL if testdata are not provided

`yhat.train.mean`
train data fits = row mean of yhat.train

`yhat.test.mean` test data fits = row mean of yhat.test

References

Ouyang, Zhi (2008) Bayesian Additive Regression Kernels. Duke University. PhD dissertation, page 58.

See Also

Other bark functions: `bark-package`, `bark-package-deprecated`, `sim_Friedman1()`, `sim_Friedman2()`, `sim_Friedman3()`, `sim_circle()`

Examples

```
##Simulated regression example
# Friedman 2 data set, 200 noisy training, 1000 noise free testing
# Out of sample MSE in SVM (default RBF): 6500 (sd. 1600)
# Out of sample MSE in BART (default):    5300 (sd. 1000)
traindata <- data.frame(sim_Friedman2(200, sd=125))
testdata <- data.frame(sim_Friedman2(1000, sd=0))
# example with a very small number of iterations to illustrate usage
fit.bark.d <- bark(y ~ ., data=traindata, testdata= testdata,
                   nburn=10, nkeep=10, keepevery=10,
                   classification=FALSE,
                   common_lambdas = FALSE,
                   selection = FALSE)
boxplot(data.frame(fit.bark.d$theta.lambda))
mean((fit.bark.d$yhat.test.mean-testdata$y)^2)

 ##Simulate classification example
 # Circle 5 with 2 signals and three noisy dimensions
 # Out of sample erorr rate in SVM (default RBF): 0.110 (sd. 0.02)
 # Out of sample error rate in BART (default):    0.065 (sd. 0.02)
 traindata <- sim_circle(200, dim=5)
 testdata <- sim_circle(1000, dim=5)
 fit.bark.se <- bark(y ~ .,
                     data=data.frame(traindata),
```

```
                    testdata= data.frame(testdata),
                    classification=TRUE,
                    nburn=100, nkeep=200, )
boxplot(as.data.frame(fit.bark.se$theta.lambda))
mean((fit.bark.se$yhat.test.mean>0)!=testdata$y)
```

`sim_circle` *Simulate Data from Hyper-Sphere for Classification Problems*

Description

The classification problem Circle is described in the BARK paper (2008). Inputs are *dim* independent variables uniformly distributed on the interval $[-1, 1]$, only the first 2 out of these *dim* are actually signals. Outputs are created according to the formula

$$y = 1(x1^2 + x2^2 \leq 2/\pi)$$

Usage

```
sim_circle(n, dim = 5)
```

Arguments

`n`	number of data points to generate
`dim`	number of dimension of the problem, no less than 2

Value

Returns a list with components

`x`	input values (independent variables)
`y`	0/1 output values (dependent variable)

References

Ouyang, Zhi (2008) Bayesian Additive Regression Kernels. Duke University. PhD dissertation, Chapter 3.

See Also

Other bark simulation functions: `sim_Friedman1()`, `sim_Friedman2()`, `sim_Friedman3()`

Other bark functions: `bark()`, `bark-package`, `bark-package-deprecated`, `sim_Friedman1()`, `sim_Friedman2()`, `sim_Friedman3()`

Examples

```
sim_circle(n=100, dim=5)
```

`sim_Friedman1` *Simulated Regression Problem Friedman 1*

Description

The regression problem Friedman 1 as described in Friedman (1991) and Breiman (1996). Inputs are 10 independent variables uniformly distributed on the interval $[0, 1]$, only 5 out of these 10 are actually used. Outputs are created according to the formula

$$y = 10 \sin(\pi x1 x2) + 20(x3 - 0.5)^2 + 10x4 + 5x5 + e$$

where e is $N(0, sd^2)$.

Usage

```
sim_Friedman1(n, sd = 1)
```

Arguments

`n`	number of data points to create
`sd`	standard deviation of noise, with default value 1

Value

Returns a list with components

`x`	input values (independent variables)
`y`	output values (dependent variable)

References

Breiman, Leo (1996) Bagging predictors. Machine Learning 24, pages 123-140.
Friedman, Jerome H. (1991) Multivariate adaptive regression splines. The Annals of Statistics 19 (1), pages 1-67.

See Also

Other bark simulation functions: `sim_Friedman2()`, `sim_Friedman3()`, `sim_circle()`

Other bark functions: `bark()`, `bark-package`, `bark-package-deprecated`, `sim_Friedman2()`, `sim_Friedman3()`, `sim_circle()`

Examples

```
sim_Friedman1(100, sd=1)
```

`sim_Friedman2` *Simulated Regression Problem Friedman 2*

Description

The regression problem Friedman 2 as described in Friedman (1991) and Breiman (1996). Inputs are 4 independent variables uniformly distributed over the ranges

$$0 \leq x1 \leq 100$$

$$40\pi \leq x2 \leq 560\pi$$

$$0 \leq x3 \leq 1$$

$$1 \leq x4 \leq 11$$

The outputs are created according to the formula

$$y = (x1^2 + (x2x3 - (1/(x2x4)))^2)^{0.5} + e$$

where e is $N(0, sd^2)$.

Usage

```
sim_Friedman2(n, sd = 125)
```

Arguments

`n`	number of data points to create
`sd`	Standard deviation of noise. The default value of 125 gives a signal to noise ratio (i.e., the ratio of the standard deviations) of 3:1. Thus, the variance of the function itself (without noise) accounts for 90% of the total variance.

Value

Returns a list with components

`x`	input values (independent variables)
`y`	output values (dependent variable)

References

Breiman, Leo (1996) Bagging predictors. Machine Learning 24, pages 123-140.
Friedman, Jerome H. (1991) Multivariate adaptive regression splines. The Annals of Statistics 19 (1), pages 1-67.

See Also

Other bark simulation functions: `sim_Friedman1()`, `sim_Friedman3()`, `sim_circle()`

Other bark functions: `bark()`, `bark-package`, `bark-package-deprecated`, `sim_Friedman1()`, `sim_Friedman3()`, `sim_circle()`

Examples

```
sim_Friedman2(100, sd=125)
```

`sim_Friedman3` *Simulated Regression Problem Friedman 3*

Description

The regression problem Friedman 3 as described in Friedman (1991) and Breiman (1996). Inputs are 4 independent variables uniformly distributed over the ranges

$$0 \leq x1 \leq 100$$

$$40\pi \leq x2 \leq 560\pi$$

$$0 \leq x3 \leq 1$$

$$1 \leq x4 \leq 11$$

The outputs are created according to the formula

$$\text{atan}((x2x3 - (1/(x2x4)))/x1) + e$$

where e is $N(0, sd^2)$.

Usage

```
sim_Friedman3(n, sd = 0.1)
```

Arguments

`n`	number of data points to create
`sd`	Standard deviation of noise. The default value of 125 gives a signal to noise ratio (i.e., the ratio of the standard deviations) of 3:1. Thus, the variance of the function itself (without noise) accounts for 90% of the total variance.

Value

Returns a list with components

`x`	input values (independent variables)
`y`	output values (dependent variable)

References

Breiman, Leo (1996) Bagging predictors. Machine Learning 24, pages 123-140.
Friedman, Jerome H. (1991) Multivariate adaptive regression splines. The Annals of Statistics 19 (1), pages 1-67.

See Also

Other bark simulation functions: sim_Friedman1(), sim_Friedman2(), sim_circle()

Other bark functions: bark(), bark-package, bark-package-deprecated, sim_Friedman1(), sim_Friedman2(), sim_circle()

Examples

```
sim_Friedman3(n=100, sd=0.1)
```

Index