

# Package ‘bark’

February 9, 2012

**Type** Package

**Title** Bayesian Additive Regression Kernels

**Version** 0.1-0

**Date** 2008-07-16

**Author** Zhi Ouyang <zo2@stat.duke.edu>, Merlise Clyde  
<clyde@stat.duke.edu>, Robert Wolpert <wolpert@stat.duke.edu>

**Maintainer** Zhi Ouyang <zo2@stat.duke.edu>

**Description** Impelementation of BARK: Bayesian Additive Regression  
Kernels with Feature Selection, Zhi Ouyang (2008), Ph.D. thesis, chapter 3.

**License** GPL (>= 2)

**URL** <http://www.r-project.org>, <http://www.stat.duke.edu/~zo2/research.html>

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2008-07-16 18:42:10

## R topics documented:

bark-package . . . . .	2
bark . . . . .	3
sim.Circle . . . . .	6
sim.Friedman1 . . . . .	7
sim.Friedman2 . . . . .	8
sim.Friedman3 . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

bark-package

*Bayesian Additive Regression Kernels*

---

## Description

Implementation of BARK: Bayesian Additive Regression Kernels with Feature Selection, Zhi Ouyang, Ph.D. thesis, Duke University

## Details

Package: bark  
Type: Package  
Version: 0.1-0  
Date: 2008-07-16  
License: GPL version 2 or newer  
LazyLoad: yes

### Overview:

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  $f$ . It 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,  $e$ ,  $d$ ,  $se$ ,  $sd$ , enabling either soft shrinkage or hard shrinkage for the scale parameters.

### Functions:

bark()  
sim.Friedman1()  
sim.Friedman2()  
sim.Friedman3()  
sim.Circle()

## Author(s)

Zhi Ouyang <zo2@stat.duke.edu>, Merlise Clyde <clyde@stat.duke.edu>, Robert Wolpert <wolpert@stat.duke.edu>

Maintainer: Zhi Ouyang <zo2@stat.duke.edu>

## References

Ouyang, Zhi (2008) Bayesian Additive Regression Kernels. Duke University. Ph.D. dissertation, Chapter 3.  
at: <http://stat.duke.edu/people/theses/OuyangZ.html>

## 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(traindata$x, traindata$y, testdata$x, classification=FALSE, type="d")
boxplot(as.data.frame(fit.bark.d$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 error 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(traindata$x, traindata$y, testdata$x, classification=TRUE, type="se")
boxplot(as.data.frame(fit.bark.se$theta.lambda))
mean((fit.bark.se$yhat.test.mean>0)!=testdata$y)
```

---

bark

*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  $f$ . It 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,  $e$ ,  $d$ ,  $se$ ,  $sd$ , enabling either soft shrinkage or hard shrinkage for the scale parameters.

## Usage

```
bark(x.train, y.train, x.test = matrix(0, 0, 0),
     type = "se", classification = FALSE,
```

```

keepevery = 100, nburn = 100, nkeep = 100,
printevery = 1000, keeptrain = FALSE,
fixed = list(),
tune = list(lstep=0.5, frequL=.2,
            dpow=1, upow=0, varphistep=.5, phistep=1),
theta = list()
)

```

### Arguments

<code>x.train</code>	Explanatory variables for training (in sample) data. Must be a matrix of doubles, with (as usual) rows corresponding to observations and columns to variables.
<code>y.train</code>	Dependent variable for training (in sample) data. If <code>y</code> is numeric a continuous response model is fit (normal errors). If <code>y</code> is a logical (or just has values 0 and 1), then a binary response model with a probit link is fit.
<code>x.test</code>	Explanatory variables for test (out of sample) data. Should have same structure as <code>x.train</code> .
<code>type</code>	BARK type, <i>e</i> , <i>d</i> , <i>se</i> , or <i>sd</i> , default choice is <i>se</i> . <i>e</i> : BARK with equal weights. <i>d</i> : BARK with different weights. <i>se</i> : BARK with selection and equal weights. <i>sd</i> : BARK with selection and different weights.
<code>classification</code>	True false logical variable, indicating a classification or regression problem.
<code>keepevery</code>	Every <code>keepevery</code> draw is kept to be returned to the user.
<code>nburn</code>	Number of MCMC iterations ( <code>nburn</code> × <code>keepevery</code> ) to be treated as burn in.
<code>nkeep</code>	Number of MCMC iterations kept for the posterior inference. <code>nkeep</code> × <code>keepevery</code> iterations after the burn in.
<code>printevery</code>	As the MCMC runs, a message is printed every <code>printevery</code> draws.
<code>keeptrain</code>	Logical, whether to keep results for training samples.
<code>fixed</code>	A list of fixed hyperparameters, using the default values if not specified. <code>alpha = 1</code> : stable index, must be 1. <code>eps = 0.5</code> : approximation parameter. <code>gam = 5</code> : intensity parameter. <code>la = 1</code> : first argument of the gamma prior on kernel scales. <code>lb = 2</code> : second argument of the gamma prior on kernel scales. <code>pbetaa = 1</code> : first argument of the beta prior on <code>plambda</code> . <code>pbetab = 1</code> : second argument of the beta prior on <code>plambda</code> . <code>n</code> : number of training samples, automatically generates. <code>p</code> : number of explanatory variables, automatically generates. <code>meanJ</code> : 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 the defaults if nothing is given.

### Details

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

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  $\sigma^*$  in the numeric case) where  $*$  denotes a particular draw. The  $x$  is either a row from the training data (`x.train`) or the test data (`x.test`).

### Value

bark returns a list, including:

fixed	Fixed hyperparameters.
tune	Tuning parameters used.
theta.last	The last set of parameters from the posterior draw.
theta.nvec	A matrix with <code>nrow(x.train)+1</code> rows and <code>(nkeep)</code> columns, recording the number of kernels at each training sample.
theta.varphi	A matrix with <code>nrow(x.train)+1</code> rows and <code>(nkeep)</code> columns, recording the precision in the normal gamma prior distribution for the regression coefficients.
theta.beta	A matrix with <code>nrow(x.train)+1</code> rows and <code>(nkeep)</code> columns, recording the regression coefficients.
theta.lambda	A matrix with <code>ncol(x.train)</code> rows and <code>(nkeep)</code> columns, recording the kernel scale parameters.
thea.phi	The vector of length <code>nkeep</code> , recording the precision in regression Gaussian noise (1 for the classification case).
yhat.train	A matrix with <code>nrow(x.train)</code> rows and <code>(nkeep)</code> columns. Each column corresponds to a draw $f^*$ from the posterior of $f$ and each row corresponds to a row of <code>x.train</code> . The $(i, j)$ value is $f^*(x)$ for the $j^{th}$ kept draw of $f$ and the $i^{th}$ row of <code>x.train</code> . For classification problems, this is the value of the expectation for the underlying normal random variable. Burn-in is dropped.
yhat.test	Same as <code>yhat.train</code> but now the <code>x</code> 's are the rows of the test data.

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

### Note

*The code is not efficient in terms of calculating the kernel matrix in the updating. This is expected to be fixed in the next version.*

### References

Ouyang, Zhi (2008) Bayesian Additive Regression Kernels. Duke University. Ph.D. dissertation, page 58.  
at: <http://stat.duke.edu/people/theses/OuyangZ.html>

### 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(traindata$x, traindata$y, testdata$x, classification=FALSE, type="d")
boxplot(as.data.frame(fit.bark.d$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 error 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(traindata$x, traindata$y, testdata$x, classification=TRUE, type="se")
boxplot(as.data.frame(fit.bark.se$theta.lambda))
mean((fit.bark.se$yhat.test.mean>0)!=testdata$y)
```

---

sim.Circle

*Simulated Classification Problem Circle*

---

### 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

$$y = 1(x_1^2 + x_2^2 \leq 2/\pi)$$

**Usage**

```
sim.Circle(n, dim=5)
```

**Arguments**

n                    number of data points to create  
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. Ph.D. dissertation, page 58. at: <http://stat.duke.edu/people/theses/OuyangZ.html>

---

sim.Friedman1	<i>Simulated Regression Problem Friedman 1</i>
---------------	--

---

**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

$$y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e$$

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

**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.

---

 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

$$0 \leq x_1 \leq 100$$

$$40\pi \leq x_2 \leq 560\pi$$

$$0 \leq x_3 \leq 1$$

$$1 \leq x_4 \leq 11$$

The outputs are created according to the formula

$$y = (x_1^2 + (x_2x_3 - (1/(x_2x_4)))^2)^{0.5} + e$$

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

## 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.

---

`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

$$0 \leq x_1 \leq 100$$

$$40\pi \leq x_2 \leq 560\pi$$

$$0 \leq x_3 \leq 1$$

$$1 \leq x_4 \leq 11$$

The outputs are created according to the formula

$$y = \text{atan}((x_2 x_3 - (1/(x_2 x_4)))/x_1) + e$$

where  $e$  is  $N(0, \text{sd})$ .

**Usage**

```
sim.Friedman3(n, sd=0.1)
```

**Arguments**

<code>n</code>	number of data points to create
<code>sd</code>	Standard deviation of noise. The default value of 0.1 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

<code>x</code>	input values (independent variables)
<code>y</code>	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.

# Index

bark, [3](#)  
bark-package, [2](#)

sim.Circle, [6](#)  
sim.Friedman1, [7](#)  
sim.Friedman2, [8](#)  
sim.Friedman3, [9](#)