Package ‘BayesTreePrior’

July 4, 2016

Title Bayesian Tree Prior Simulation
Version 1.0.1
Date 2016-06-27
Author Alexia Jolicoeur-Martineau <alexia.jolicoeur-martineau@mail.mcgill.ca>
Maintainer Alexia Jolicoeur-Martineau <alexia.jolicoeur-martineau@mail.mcgill.ca>
Description Provides a way to simulate from the prior distribution of Bayesian trees by Chipman et al. (1998) <DOI:10.2307/2669832>. The prior distribution of Bayesian trees is highly dependent on the design matrix X, therefore using the suggested hyperparameters by Chipman et al. (1998) <DOI:10.2307/2669832> is not recommended and could lead to unexpected prior distribution. This work is part of my master thesis (expected 2016).
License GPL-3
Imports stats
Suggests tgp, BayesTree, bartMachine, MASS
RoxygenNote 5.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2016-07-04 20:28:58

R topics documented:

BayesTreePrior .................................................. 2
BayesTreePriorNotOrthogonal ................................. 4
BayesTreePriorOrthogonal .................................. 5
BayesTreePriorOrthogonalInf ................................. 6
E_alpha ............................................................. 7
GetListUniqueSplits ............................................. 8
NumBotMaxDepth ................................................ 8
NumBotMaxDepthX ................................................. 9
NumBotMaxDepth_inf .......................................... 10
p_split .......................................................... 11
Var_alpha ......................................................... 12

Index 13
BayesTreePrior

Simulation of the tree prior.

Description

This is the main function to use for simulating from the prior. There are 4 cases:

• Case #1: Unrealistic case where we assume that the number of variables and possible splits are infinite (therefore $P(T)$ is not dependent on the design matrix X) and $\beta = 0$
• Case #2: Unrealistic case where we assume that the number of variables and possible splits are infinite (therefore $P(T)$ is not dependent on the design matrix X)
• Case #3: One variable with a finite number of observations (Seems to be equivalent to the multiple variables case when all variables are continuous)
• Case #4: General case

Case #1 will be used if no design matrix X or number of observations is given and $\beta = 0$. Case #2 will be used if no design matrix X or number of observations is given and $\beta \neq 0$. Case #3 will be used if no design matrix X is given but the number of observations is given. Case #4 will be used if the design matrix X is given. Note that case #4 is always slower, so if all your variables are continuous, it would be advisable to enter the number of uniques observations rather than the design matrix X, to be able to use case #3.

Usage

BayesTreePrior(alpha, beta, X = NULL, n_obs = NULL, n_iter = 500,
minpart = 1, package = NULL, pvars = NULL, MIA = FALSE,
missingdummy = FALSE)

Arguments

alpha  
base parameter of the tree prior, $\alpha \in [0, 1)$.

beta  
power parameter of the tree prior, $\beta \geq 0$.

X  
data.frame of the design matrix (Required for case #4).

n_obs  
number of unique observations, $n_{\text{obs}} > 1$ (Required for case #3).

n_iter  
number of trees to generate, $n_{\text{iter}} > 0$ (Used for case #2, #3 or #4).

minpart  
the minimum number of observations required in one of the child to be able to split, $minpart > 0$.

package  
a optional string that can take the following values : "BayesTree", "tgp" or "bartMachine". It forces the algorithm to use the default parameters used by the package specified ($minpart = 5$ for BayesTree, $minpart = max(c(10, \text{dim}(X)[2]+1))$ for tgp and $minpart = 1$ for bartMachine).

pvars  
vector of probabilities for the choices of variables to split (Will automatically be normalized so that the sum equal to 1). It must be twice as large as the number of variables when $missingdummy$ is TRUE.
MIA set to TRUE if you want Missing Incorporated in Attributes (MIA) imputation to be used.

missingdummy set to TRUE if you want the NAs to be dummy coded.

Value

In case #1, it returns a list containing, in the following order: the expectation and the variance of the number of bottom nodes. In cases #2, #3 or #4, it returns a list containing, in the following order: the mean number of bottom nodes, the standard deviation of the number of bottom nodes, the mean of the depth, the standard deviation of the depth and a data.frame of vectors \( (b_i, d_i) \), where \( b_i \) is the number of bottom nodes and \( d_i \) is the depth of the \( i \)th generated tree \( (i = 1, \ldots, n_{iter}) \).

References


Examples

#Case 1 : Unrealistic case where we assume that the number of var/obs is infinite and beta=0 results1 = BayesTreePrior(.45,0)

#Case 2 : Unrealistic case where we assume that the number of var/obs is infinite results2 = BayesTreePrior(.95,.5)

#Case 3 : One variable with a finite number of observations results3 = BayesTreePrior(.95,.5,n_obs=150)

if (requireNamespace("MASS", quietly = TRUE)) {
    #Case 4 : General case, without missing data
    x1 = MASS::mcycle$times
    x2= MASS::mcycle$accel
    X = cbind(x1, x2)
    results4_nomiss = BayesTreePrior(.95,.5, data.frame(X), minpart=5, package="tgp")

    #Case 4 : General case, with missing data
    x1[sample(1:length(x1), 20)] <- NA
    x2[sample(1:length(x2), 20)] <- NA
    X = cbind(x1, x2)
    results4_miss = BayesTreePrior(.95,.5, data.frame(X), minpart=5, package="tgp", MIA=TRUE, missingdummy=TRUE)
}

BayesTreePriorNotOrthogonal

Simulation of the tree prior in the general case (Case #4).

Description

Generate \( n_{\text{iter}} \) trees from the prior distribution in the general case (Case #4).

Usage

```r
BayesTreePriorNotOrthogonal(alpha, beta, X, n_iter = 500, minpart = 1,
                               pvars = NULL, MIA = FALSE, missingdummy = FALSE)
```

Arguments

- `alpha` base parameter of the tree prior, \( \alpha \in [0, 1) \).
- `beta` power parameter of the tree prior, \( \beta \geq 0 \).
- `X` data.frame of the design matrix.
- `n_iter` number of trees to generate, \( n_{\text{iter}} > 0 \).
- `minpart` the minimum number of observations required in one of the child to be able to split, \( \text{minpart} > 0 \).
- `pvars` vector of probabilities for the choices of variables to split (Will automatically be normalized so that the sum equal to 1). It must be twice as large as the number of variables when `missingdummy` is TRUE.
- `MIA` set to TRUE if you want Missing Incorporated in Attributes (MIA) imputation to be used.
- `missingdummy` set to TRUE if you have dummy coded the NAs.

Value

Returns a list containing, in the following order: the mean number of bottom nodes, the standard deviation of the number of bottom nodes, the mean of the depth, the standard deviation of the depth and a data.frame of vectors \((b_i, d_i)\), where \( b_i \) is the number of bottom nodes and \( d_i \) is the depth of the \( i \)th generated tree \((i = 1, \ldots, n_{\text{iter}})\).

See Also

BayesTreePriorOrthogonalInf, BayesTreePriorOrthogonal

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  x1 = MASS::mcycle$s$times
  x1[sample(1:length(x1), 20)] <- NA
  x2 = MASS::mcycle$s$accel
  x2[sample(1:length(x2), 20)] <- NA
}
BayesTreePriorOrthogonal

Simulation of the tree prior in the case where we have one single variable (Case #3).

Description

Generate \( n_{\text{iter}} \) trees from the prior distribution in the case where we have one variable with a finite number of observations (Case #3).

Usage

\[
\text{bayestreepriororthogonal}(\alpha, \beta, n_{\text{obs}}, n_{\text{iter}} = 500)
\]

Arguments

- **alpha**: base parameter of the tree prior, \( \alpha \in (0, 1) \).
- **beta**: power parameter of the tree prior, \( \beta \geq 0 \).
- **n_obs**: number of unique observations, \( n_{\text{obs}} > 1 \).
- **n_iter**: number of trees to generate, \( n_{\text{iter}} > 0 \).

Value

Returns a list containing, in the following order: the mean number of bottom nodes, the standard deviation of the number of bottom nodes, the mean of the depth, the standard deviation of the depth and a data.frame of vectors \((b_i, d_i)\), where \( b_i \) is the number of bottom nodes and \( d_i \) is the depth of the \( i \)th generated tree \((i = 1, \ldots, n_{\text{iter}})\).

See Also

BayesTreePriorOrthogonalInf, BayesTreePriorNotOrthogonal

Examples

\[
\text{results1} = \text{bayestreepriororthogonal}(.95, .5, 100)
\]
\[
\text{results2} = \text{bayestreepriororthogonal}(.95, .5, 250)
\]
Simulation of the tree prior in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore $P(T)$ is not dependent on the design matrix $X$) (Case #2).

**Description**

Generate $n_{\text{iter}}$ trees from the prior distribution in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore $P(T)$ is not dependent on the design matrix $X$) (Case #2).

**Usage**

```r
BayesTreePriorOrthogonalInf(alpha, beta, n_iter = 500)
```

**Arguments**

- `alpha`: base parameter of the tree prior, $\alpha \in [0, 1)$.
- `beta`: power parameter of the tree prior, $\beta \geq 0$.
- `n_iter`: number of trees to generate, $n_{\text{iter}} > 0$.

**Value**

Returns a list containing, in the following order: the mean number of bottom nodes, the standard deviation of the number of bottom nodes, the mean of the depth, the standard deviation of the depth and a data.frame of vectors $(b_i, d_i)$, where $b_i$ is the number of bottom nodes and $d_i$ is the depth of the $i$th generated tree ($i = 1, \ldots, n_{\text{iter}}$).

**See Also**

`BayesTreePriorOrthogonal, BayesTreePriorNotOrthogonal`

**Examples**

```r
results = BayesTreePriorOrthogonalInf(.95, .5)
```
Expected value of the number of bottom nodes in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore $P(T)$ is not dependent on the design matrix $X$) and $\beta = 0$ (Case #1).

**Description**

Expected value of the number of bottom nodes in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore $P(T)$ is not dependent on the design matrix $X$) and $\beta = 0$ (Case #1).

**Usage**

```r
e_alpha(alpha)
```

**Arguments**

- `alpha` base parameter of the tree prior, $alpha \in [0, 1]$.

**Value**

Returns the expected value of the number of bottom nodes.

**References**


**See Also**

- `Var_alpha`

**Examples**

- `E_alpha(.30)`
- `E_alpha(.45)`
- `E_alpha(.499)`
- `E_alpha(.75)`
**GetListUniqueSplits**  
Unique splits that leads to children with more than \textit{minpart} nodes.

**Description**
Unique splits that leads to children with more than \textit{minpart} nodes.

**Usage**
GetListUniqueSplits(x, minpart = 1, MIA = FALSE)

**Arguments**
- \textit{x} vector containing the observations of a variable.
- \textit{minpart} minimum number of observations in the children nodes.
- \textit{MIA} set to TRUE if you want Missing Incorporated in Attributes (MIA) imputation to be used.

**Value**
If \textit{MIA} is TRUE and \textit{minpart} > 1, the possible splits could be different depending on whether we transfer the NAs to the left child or the right child; if this is the case then the function returns a list \((v_1,v_2)\), where \(v_1\) is the vector containing the unique splits that leads to \textit{minpart} nodes when transferring the NAs to the left child and \(v_2\) is the vector containing the unique splits that leads to children with more than \textit{minpart} nodes when transferring the NAs to the left child. Otherwise, it returns the vector containing the unique splits that leads to children with more than \textit{minpart} nodes.

**Examples**
GetListUniqueSplits(c(1,4,7,3,0,2,2,3,4,7,7,7),minpart=1)
GetListUniqueSplits(c(1,4,7,3,0,2,2,3,4,7,7,7),minpart=3)
GetListUniqueSplits(c(1,4,7,3,0,2,2,3,4,7,7,7,NA,NA),minpart=1, MIA=TRUE)
GetListUniqueSplits(c(1,4,7,3,0,2,2,3,4,7,7,7,NA,NA),minpart=3, MIA=TRUE)

**NumBotMaxDepth**  
Number of bottom nodes and depth in the case where we have one single variable (Case #3).

**Description**
Generate a tree and returns the number of bottom nodes and depth in the case where we have one variable with a finite number of observations (Case #3).

**Usage**
NumBotMaxDepth(alpha, beta, x_size, depth = 0)
NumBotMaxDepthX

Arguments

- **alpha**: base parameter of the tree prior, \( \alpha \in [0, 1) \).
- **beta**: power parameter of the tree prior, \( \beta \geq 0 \).
- **x_size**: number of possible splits, \( x_{size} > 0 \).
- **depth**: depth of the current node, \( depth \geq 0 \).

Value

Returns a vector containing the number of bottom nodes and depth.

See Also

- NumBotMaxDepth_inf, NumBotMaxDepthX

Examples

```r
numBotMaxDepth(0.95, 5, 500)
```

Description

Generate a tree and returns the number of bottom nodes and depth in the general case (Case #4).

Usage

```r
NumBotMaxDepthX(alpha, beta, X, depth = 0, minpart = 1, pvars = NULL,
                 MIA = FALSE, missingdummy = FALSE)
```

Arguments

- **alpha**: base parameter of the tree prior, \( \alpha \in [0, 1) \).
- **beta**: power parameter of the tree prior, \( \beta \geq 0 \).
- **X**: data.frame of the design matrix.
- **depth**: depth of the current node, \( depth \geq 0 \).
- **minpart**: the minimum number of observations required in one of the child to be able to split, \( minpart \geq 0 \).
- **pvars**: vector of probabilities for the choices of variables to split (Will automatically be normalized so that the sum equal to 1). It must be twice as large as the number of variables when missingdummy is TRUE.
- **MIA**: set to TRUE if you want Missing Incorporated in Attributes (MIA) imputation to be used.
- **missingdummy**: set to TRUE if you have dummy coded the NAs.
Value

Returns a vector containing the number of bottom nodes and depth

References


See Also

NumBotMaxDepth_inf, NumBotMaxDepth

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  x1 = MASS::mcycle$times
  x1[sample(1:length(x1), 20)] <- NA
  x2 = MASS::mcycle$accel
  x2[sample(1:length(x2), 20)] <- NA
  X = cbind(x1, x2)
  results1 = NumBotMaxDepthX(.95,.5, data.frame(X), minpart=5)
  X_dummies = is.na(X) + 0
  results2 = NumBotMaxDepthX(.95,.5, data.frame(cbind(X,X_dummies)), minpart=5, MIA=TRUE, missingdummy=TRUE)
}
```

NumBotMaxDepth_inf  Number of bottom nodes and depth in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore \( P(T) \) is not dependent on the design matrix \( X \) (Case #2).

Description

Generate a tree and returns the number of bottom nodes and depth in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore \( P(T) \) is not dependent on the design matrix \( X \) (Case #2).

Usage

NumBotMaxDepth_inf(alpha, beta, depth = 0)

Arguments

- **alpha**: base parameter of the tree prior, \( \alpha \in [0, 1) \).
- **beta**: power parameter of the tree prior, \( \beta \geq 0 \).
- **depth**: depth of the current node, \( \text{depth} \geq 0 \).


**p_split**

Value

Returns a vector containing the number of bottom nodes and depth.

See Also

NumBotMaxDepth, NumBotMaxDepthX

Examples

NumBotMaxDepth_inf(.95,.5)

---

**p_split**

*Probability of split of the tree prior.*

Description

Probability of split of the tree prior.

Usage

p_split(alpha, beta, depth = 0)

Arguments

- **alpha**
  - base parameter of the tree prior, \( \alpha \in [0, 1) \).
- **beta**
  - power parameter of the tree prior, \( \beta \geq 0 \).
- **depth**
  - depth of the current node, \( depth \geq 0 \).

Value

Returns the probability of split.

References


Examples

p_split(.95,.5)
p_split (.95,.5,1)
p_split (.95,.5,2)
### Description

Variance of the number of bottom nodes in the unrealistic case where we assume that the number of variables and possible splits are infinite (therefore P(T) is not dependent on the design matrix X) and $\beta = 0$ (Case #1).

### Usage

```
Var_alpha(alpha)
```

### Arguments

- **alpha**: base parameter of the tree prior, $\alpha \in [0, 1)$.

### Value

Returns the variance of the number of bottom nodes.

### References


### See Also

- `E_alpha`

### Examples

```
Var_alpha(.30)
Var_alpha(.45)
Var_alpha(.499)
Var_alpha(.75)
```
Index

BayesTreePrior, 2
BayesTreePriorNotOrthogonal, 4, 5, 6
BayesTreePriorOrthogonal, 4, 5, 6
BayesTreePriorOrthogonalInf, 4, 5, 6

E_alpha, 7, 12

GetListUniqueSplits, 8

NumBotMaxDepth, 8, 10, 11
NumBotMaxDepthInf, 9, 10, 11
NumBotMaxDepthX, 9, 9, 11

p_split, 11

Var_alpha, 7, 12