The tgp Package

August 23, 2006

Title  Bayesian treed Gaussian process models

Version  1.1-7

Date  2006-08-22

Author  Robert B. Gramacy <rbgramacy@ams.ucsc.edu>

Depends  R (>= 2.1)

Suggests  akima, maptree, combinat

Description  Bayesian semiparametric and nonstationary regression by treed Gaussian processes with
jumps to the limiting linear model (LLM). Special cases also implemented include Bayesian
linear models, linear CART, stationary separable and isotropic Gaussian process regression.
Provides 1-d and 2-d plotting functions (with projection and slice capabilities) and tree drawing,
designed for visualization of tgp-class output.

Maintainer  Robert B. Gramacy <rbgramacy@ams.ucsc.edu>

License  LGPL (Lesser GNU Public License)

URL  http://www.ams.ucsc.edu/~rbgramacy/tgp.html

R topics documented:

btgp ................................................................. 2
dopt.gp ............................................................ 7
exp2d ............................................................. 9
exp2d.rand ....................................................... 10
friedman.1.data ............................................... 11
interp.loess .................................................... 12
lhs ................................................................. 14
mapT ............................................................... 15
partition ......................................................... 16
plot.tgp .......................................................... 17
tgp-internal ..................................................... 19
tgp-package ...................................................... 20
tgp ............................................................... 20
tgp.default.params ........................................... 25
tgp.design ........................................................ 26
tgp.get.partitions ............................................. 28
tgp.trees ........................................................ 29
One of Six Bayesian Nonparametric & Nonstationary Regression Models

Description

The six functions described below implement Bayesian regression models of varying complexity: linear model, linear CART, Gaussian process (GP), GP with jumps to the limiting linear model (LLM), treed GP, and treed GP LLM. They are provided as a streamlined interface to the \texttt{tgp} function of which each of the functions herein represents a special case.

Usage

\begin{verbatim}
blm(X, Z, XX = NULL, bprior = "bflat", BTE = c(1000, 4000, 3),
    R = 1, m0r1 = FALSE, pred.n = TRUE, ds2x = FALSE,
    ego = FALSE, traces = FALSE, verb = 1)
blm(X, Z, XX = NULL, bprior = "bflat", tree = c(0.25, 2, 10),
    BTE = c(2000, 7000, 2), R = 1, m0r1 = FALSE,
    pred.n = TRUE, ds2x = FALSE, ego=FALSE, traces = FALSE,
    verb = 1)
bgp(X, Z, XX = NULL, bprior = "bflat", corr = "expsep",
    BTE = c(1000, 4000, 2), R = 1, m0r1 = FALSE,
    pred.n = TRUE, ds2x = FALSE, ego = FALSE, nu = 0.5,
    traces = FALSE, verb = 1)
bgpllm(X, Z, XX = NULL, bprior = "bflat", corr = "expsep",
    gamma=c(10,0.2,0.7), BTE = c(1000, 4000, 2), R = 1,
    m0r1 = FALSE, pred.n = TRUE, ds2x = FALSE,
    ego = FALSE, nu = 0.5, traces = FALSE, verb = 1)
btp(X, Z, XX = NULL, bprior = "bflat", corr = "expsep",
    tree = c(0.25, 2, 10), BTE = c(2000, 7000, 2), R = 1,
    m0r1 = FALSE, linburn = FALSE, pred.n = TRUE, ds2x = FALSE,
    ego = FALSE, nu = 0.5, traces = FALSE, verb = 1)
btppllm(X, Z, XX = NULL, bprior = "bflat", corr = "expsep",
    tree = c(0.25, 2, 10), gamma=c(10,0.2,0.7),
    BTE = c(2000, 7000, 2), R = 1, m0r1 = FALSE,
    linburn = FALSE, pred.n = TRUE, ds2x = FALSE,
    ego = FALSE, nu = 0.5, traces = FALSE, verb = 1)
\end{verbatim}

Arguments

Each of the above functions takes some subset of the following arguments...

\begin{verbatim}
X        data.frame, matrix, or vector of inputs X
Z        Vector of output responses Z of length equal to the leading dimension (rows) of X, i.e., length(Z) == dim(X)[1]
XX       Optional data.frame, matrix, or vector of predictive input locations with the same number of columns as X, i.e., dim(XX)[2] == dim(X)[2]
\end{verbatim}
btgp

bprior  Linear (beta) prior, default is "bflat"; alternates include "b0" hierarchical Normal prior, "bmle" empirical Bayes Normal prior, "bcart" Bayesian linear CART style prior from Chipman et al., "b0tau" a independent Normal prior with inverse-gamma variance.

tree  3-vector of tree process prior parameterization c(\alpha, \beta, nmin) specifying 

\[ p_{\text{om}}(\eta, \mathcal{T}) = \alpha \times (1 + \eta)^\beta \]

giving zero probability to trees with partitions containing less than nmin data points.

gamma  Limiting linear model parameters c(g, t1, t2), with growth parameter 
g > 0 minimum parameter t1 >= 0 and maximum parameter t1 >= 0, where t1 + t2 <= 1 specifies 

\[ p(b|d) = t_{1} + \exp\left\{-g(t_{2} - t_{1})/(d - 0.5)\right\} \]

corr  Gaussian process correlation model. Choose between the isotropic power exponential family ("exp") or the separable power exponential family ("expsep", default); the current version also supports the isotropic Matern ("matern") as “beta” functionality

BTE  3-vector of Monte-carlo parameters (B)urn in, (T)otal, and (E)very. Predictive samples are saved every E MCMC rounds starting at round B, stopping at T.

R  Number of repeats or restarts of BTE MCMC rounds, default R=1 is no restarts

m0r1  If TRUE the responses Z will be scaled to have a mean of zero and a range of 1; default is FALSE

linburn  If TRUE initializes MCMC with B (additional) rounds of Bayesian Linear CART (btlm); default is FALSE

pred.n  TRUE (default) value results in prediction at the inputs X; FALSE skips prediction at X resulting in a faster implementation

ds2x  TRUE results in ALC (Active Learning–Cohn) computation of expected reduction in uncertainty calculations at the X locations, which can be used for adaptive sampling; FALSE (default) skips this computation, resulting in a faster implementation

ego  TRUE results in EGO (Expected Global Optimization) computation of expected information about the location of the minimum reduction in uncertainty calculations at the XX locations, which can be used for adaptive sampling; FALSE (default) skips this computation, resulting in a faster implementation

nu  “beta” functionality: fixed smoothness parameter for the Matern correlation function; nu+0.5 times differentiable

traces  TRUE results in a saving of samples from the posterior distribution for most of the parameters in the model. The default is FALSE for speed/storage reasons. See note below

verb  Level of verbosity of R-console print statements: from 0 (none); 1 (default) which shows the “progress meter”; 2 includes an echo of initialization parameters; up to 3 and 4 (max) with more info about successful tree operations

Details

The functions and their arguments can be categorized by whether or not they use treed partitioning (T), GP models, and jumps to the LLM
Each function implements a special case of the generic function \texttt{tgp} which is an interface to C/C++ code for treed Gaussian process modeling of varying parameterization. For each of the examples, below, see \texttt{help(tgp)} for the direct \texttt{tgp} implementation. Only functions in the T (tree) category take the \texttt{tree} argument; GP category functions take the \texttt{corr} argument; and LLM category functions take the \texttt{gamma} argument. Non-tree class functions omit the \texttt{parts} and \texttt{trees} outputs, see below.

Please see \texttt{vignette("tgp")} for detailed illustration.

**Value**

\texttt{bgp} returns an object of class "\texttt{tgp}". The function \texttt{plot.tgp} can be used to help visualize results.

An object of class "\texttt{tgp}" is a list containing at least the following components... The final two (\texttt{parts} \& \texttt{trees}) are tree-related outputs unique to the T (tree) category functions. Tree viewing is supported by \texttt{tgp.trees}.

- \texttt{state} unsigned short[3] random number seed to C
- \texttt{X} Input argument: \texttt{data.frame} of inputs \(X\)
- \texttt{n} Number of rows in \(X\), i.e., \texttt{dim(X)[1]}.
- \texttt{d} Number of cols in \(X\), i.e., \texttt{dim(X)[2]}.
- \texttt{Z} Vector of output responses \(Z\).
- \texttt{XX} Input argument: \texttt{data.frame} of predictive locations \(XX\).
- \texttt{nn} Number of rows in \(XX\), i.e., \texttt{dim(XX)[1]}.
- \texttt{BTE} Input argument: Monte-carlo parameters.
- \texttt{R} Input argument: restarts.
- \texttt{linburn} Input argument: initialize MCMC with linear CART.
- \texttt{params} list of model parameters generated by \texttt{tgp.default.params} and passed to \texttt{tgp}.
- \texttt{dparams} Double-representation of model input parameters used by the C-code.
- \texttt{Zp.mean} Vector of mean predictive estimates at \(X\) locations.
- \texttt{Zp.q1} Vector of 5\% predictive quantiles at \(X\) locations.
- \texttt{Zp.q2} Vector of 95\% predictive quantiles at \(X\) locations.
- \texttt{Zp.q} Vector of quantile norms \(Zp.q2-Zp.q1\).
- \texttt{ZZ.q1} Vector of 5\% predictive quantiles at \(XX\) locations.
- \texttt{ZZ.q2} Vector of 95\% predictive quantiles at \(XX\) locations.
- \texttt{ZZ.q} Vector of quantile norms \(ZZ.q2-ZZ.q1\), used by the Active Learning–MacKay (ALM) adaptive sampling algorithm.
- \texttt{Ds2x} If argument \texttt{ds2x=TRUE}, this vector contains ALC statistics for \(XX\) locations.
- \texttt{ego} If argument \texttt{ego=TRUE}, this vector contains EGO statistics for \(XX\) locations.
response  Name of response \( Z \) if supplied by \texttt{data.frame} in argument, or "z" if none provided

parts  Internal representation of the regions depicted by partitions of the maximum a’ posteriori (MAP) tree

trees  list of trees (\texttt{maptree} representation) which were MAP as a function of each tree height sampled between MCMC rounds \( B \) and \( T \)

traces  If input argument \texttt{traces=}TRUE, this is list containing traces of most of the model parameters and posterior predictive distributions at input locations \( XX \). Otherwise the entry is FALSE. See note below

verb  Input argument: verbosity level

Note

Inputs \( X, \ XX, \ Z \) containing NaN, NA, Inf are discarded with non-fatal warnings

Upon execution, MCMC reports are made every 1,000 rounds to indicate progress

Stationary (non-treed) processes on larger inputs (e.g., \( X, Z \)) of size greater than 500, *might* be slow in execution, especially on older machines. Once the C code starts executing, it can be interrupted in the usual way: either via Ctrl-C (Unix-alikes) or pressing the Stop button in the \texttt{R}-GUI. When this happens, interrupt messages will indicate which required cleanup measures completed before returning control to \( R \).

Regarding \texttt{traces=}TRUE: Samples from the posterior will be collected for all parameters in the model, except those of the hierarchical priors, e.g., \( b0, \) etc. Traces for some parameters are stored in memory, others in files. GP parameters are collected with reference to the locations in \( XX \), resulting \( \texttt{nn}=\text{dim}\{XX\}[2] \) traces of \( d,g,s2,tau2,\text{ etc} \). Therefore, it is recommended that \( \texttt{nn} \) is chosen to be a small, representative, set of input locations. Besides GP parameters, traces are saved for the tree partitions, areas under the LLM, log posterior (as a function of tree height), and samples \( ZZ \) from the posterior predictive distribution at \( XX \)

Author(s)

Robert B. Gramacy \{rbgramacy@ams.ucsc.edu\}

References


http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

\texttt{tgp, plot.tgp, tgp.trees}
Examples

## Many of the examples below illustrate the above function(s) on random data. Thus it can be fun (and informative) to run them several times.

```
# simple linear response
#
# input and predictive data
X <- seq(0,1,length=50)
XX <- seq(0,1,length=99)
Z <- 1 + 2*X + rnorm(length(X),sd=0.25)
out <- blm(X=X, Z=Z, XX=XX) # try Linear Model
plot(out) # plot the surface

# 1-d Example
#
# construct some 1-d nonstationary data
X <- seq(0,20,length=100)
XX <- seq(0,20,length=99)
Z <- (sin(pi*X/5) + 0.2*cos(4*pi*X/5)) * (X <= 9.6)
lin <- X>9.6;
Z[lin] <- -1 + X[lin]/10
Z <- Z + rnorm(length(Z), sd=0.1)
out <- btlm(X=X, Z=Z, XX=XX) # try Linear CART
plot(out) # plot the surface
tgp.trees(out) # plot the MAP trees

out <- btgp(X=X, Z=Z, XX=XX) # use a treed GP
plot(out) # plot the surface
tgp.trees(out) # plot the MAP trees

# 2-d example
# (using the isotropic correlation function)
#
# construct some 2-d nonstationary data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
XX <- exp2d.data$XX

# try a GP
out <- bgp(X=X, Z=Z, XX=XX, corr="exp")
plot(out) # plot the surface

# try a treed GP LLM
out <- btgpllm(X=X, Z=Z, XX=XX, corr="exp")
plot(out) # plot the surface
```
## Motorcycle Accident Data

# get the data
# and scale the response to zero mean and a range of 1 (m0r1)
require(MASS)

# try a GP
out <- bgp(X=mcycle[,1], Z=mcycle[,2], m0r1=TRUE)
plot(out)  # plot the surface

# try a treed GP LLM
# best to use the "b0" beta linear prior to capture common
# common linear process throughout all regions
out <- btgpllm(X=mcycle[,1], Z=mcycle[,2], bprior="b0", m0r1=TRUE)
plot(out)  # plot the surface
tgp.trees(out)  # plot the MAP trees

# Actually, instead of using m0r1, the mcycle data is best fit
# with using a mixture prior for the nugget due to its input-
# dependent noise. See the examples for the tgp function

# for other examples try the demos or the vignette

---

**dopt gp**

**Sequential D-Optimal Design for a Stationary Gaussian Process**

**Description**

Create sequential D-Optimal design for a stationary Gaussian process model of fixed parameterization by subsampling from a list of candidates

**Usage**

```
dopt.gp(nn, X, Xcand)
```

**Arguments**

- `nn` Number of new points in the design. Must be less than or equal to the number of candidates contained in `Xcand`, i.e., `nn <= dim(Xcand)`
- `X` data.frame, matrix or vector of input locations which are forced into (already in) the design
- `Xcand` data.frame, matrix or vector of candidates from which new design points are subsampled. Must have the same dimension as `X`, i.e., `dim(X)[2] == dim(Xcand)[2]`
Details

Design is based on a stationary Gaussian process model with stationary isotropic exponential correlation function with parameterization fixed as a function of the dimension of the inputs. The algorithm implemented is a simple stochastic ascent which maximizes $\det(K)$ – the covariance matrix constructed with locations $X$ and a subset of $X_{\text{cand}}$ of size $nn$. The selected design is locally optimal.

Value

The output is a list which contains the inputs to, and outputs of, the C code used to find the optimal design. The chosen design locations can be accessed as list members $XX$ or equivalently $X_{\text{cand}}[fi,]$.

```r
state
X
nn
n
m
X_{\text{cand}}
n_{\text{cand}}
fi
XX
```

state: unsigned short[3] random number seed to C

X: Input argument: data.frame of inputs $X$, can be NULL

nn: Input argument: number new points in the design

n: Number of rows in $X$, i.e., $n = \text{dim}(X)[1]$ ...

m: Number of cols in $X$, i.e., $m = \text{dim}(X)[2]$ ...

X_{\text{cand}}: Input argument: data.frame of candidate locations $X_{\text{cand}}$

n_{\text{cand}}: Number of rows in $X_{\text{cand}}$, i.e., $n_{\text{cand}} = \text{dim}(X_{\text{cand}})[1]$ ...

fi: Vector of length $nn$ describing the selected new design locations as indices into $X_{\text{cand}}$

XX: data.frame of selected new design locations, i.e., $XX = X_{\text{cand}}[fi,]$ ...

Note

Inputs $X$, $X_{\text{cand}}$ containing NaN, NA, Inf are discarded with non-fatal warnings. If $nn > \text{dim}(X_{\text{cand}})[1]$ then a non-fatal warning is displayed and execution commences with $nn = \text{dim}(X_{\text{cand}})[1]$ ...

In the current version there is no progress indicator. You will have to be patient. Creating D-optimal designs is no speedy task.

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References


See Also

tgp.design, lhs
Examples

```r
# 2-d Exponential data
# (This example is based on random data.
# It might be fun to run it a few times)
#
# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX

# find a treed sequential D-Optimal design
# with 10 more points
dgp <- dopt.gp(10, X, Xcand)

# plot the d-optimally chosen locations
# Contrast with locations chosen via
# the tgp.design function
plot(X, pch=19, xlim=c(-2,6), ylim=c(-2,6))
points(dgp$XX)
```

exp2d  2-d Exponential Data

Description

A 2-dimensional data set that can be used to validate non-stationary models.

Usage

data(exp2d)

Format

A data frame with 441 observations on the following 4 variables.

- **X1** Numeric vector describing the first dimension of X inputs
- **X2** Numeric vector describing the second dimension of X inputs
- **Z** Numeric vector describing the response $Z(X) + \mathcal{N}(0, \text{sd}=0.001)$
- **Ztrue** Numeric vector describing the true response $Z(X)$, without noise

Details

The response is evaluated as

$$Z(X) = x_1 \cdot \exp(x_1^2 - x_2^2).$$

Zero-mean normal noise with $\text{sd}=0.001$ has been added to the response

Note

This data is used in the examples of the functions listed above in the “See Also” section via the `exp2d.rand` function.
exp2d.rand

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

exp2d.rand, tgp, bgpllm, btlm, blm, bgp, btgpllm, bgp

Description

A Random subsample of data(exp2d)

Usage

exp2d.rand(n1 = 50, n2 = 30)

Arguments

n1 Number of samples from the first, interesting, quadrant
n2 Number of samples from the other three, uninteresting, quadrants

Details

Data is subsampled without replacement from data(exp2d). Of the n1 + n2 >= 441 input/response pairs, n1 are taken from the first quadrant, i.e., where the response is interesting, and the remaining n1 are taken from the other three quadrant. The remaining 441 - (n1 + n2) are treated as predictive locations.

Value

Output is a list with entries:

- X 2-d data.frame with n1 + n2 input locations
- Z Numeric vector describing the responses (with noise) at the X input locations
- Ztrue Numeric vector describing the true responses (without noise) at the X input locations
- XX 2-d data.frame containing the remaining 441 - (n1 + n2) input locations
- ZZ Numeric vector describing the responses (with noise) at the XX predictive locations
- ZZtrue Numeric vector describing the responses (without noise) at the XX predictive locations
friedman.1.data

Note

This data is used in the examples of the functions listed above in the “See Also” section, below

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

exp2d, tgp, bgpllm, btlm, blm, bgp, btgpllm, bgp

Examples

# random data
ed <- exp2d.rand()

# higher span = 0.5 required because the data is sparse
# and was generated randomly
ed.g <- interp.loess(ed$X[,1], ed$X[,2], ed$Z, span=0.5)

# perspective plot, and plot of the input (X) locations
par(mfrow=c(1,2), bty="n")
persp(ed.g, main="loess surface", theta=-30, phi=20,
      xlab="X[,1]", ylab="X[,2]", zlab="Z")
plot(ed$X, main="Randomly Subsampled Inputs")

friedman.1.data  First Friedman Dataset

Description

Function to generate X and Y values from the 10-dim “first” Friedman data set used to validate the Multivariate Adaptive Regression Splines (MARS) model. This function is stationary, with three non-linear and interacting variables, along with two linear, and five irrelevant effects.

Usage

friedman.1.data(n = 100)

Arguments

n  Number of samples

Details

10-dim inputs X are drawn from N(0,1), and responses are N(m(X),1) where m(X) = E[X] and

E[X] = 10 sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5
Value

Output is a data frame with columns

- X1...X10 describing the 10-d sampled inputs
- Y sample responses (with N(0,1) noise)
- Ytruth true responses (without noise)

Note

An example using this data is contained in the package vignette: vignette("tgp").

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

tgp, bgpllm, btlm, blm, bgp, btgpllm, bgp

Description

Use the loess function to interpolate the two-dimensional x, y, z data onto a uniform grid. The output produced is an object directly usable by the plotting functions persp, image, and contour, etc.

This function is designed as an alternative to the interp functions from the akima library.

Usage

interp.loess(x, y, z, gridlen = 40, span = 0.1, ...)
interp.loess

Arguments

x Vector of X spatial input locations
y Vector of Y spatial input locations
z Vector of Z responses interpreted as \( Z = f(X, Y) \)
gridlen Size of the interpolated grid to be produced. The default of gridlen = 40 causes a 40 * 40 grid of X, Y, and Z values to be computed.
span Kernel span argument to the loess function with default setting span = 0.1 set significantly lower than the the loess default – see note below.

Details

Uses expand.grid function to produce a uniform grid of size gridlen with domain equal to the rectangle implied by X and Y. Then, a loess a smoother is fit to the data \( Z = f(X, Y) \). Finally, predict.loess is used to predict onto the grid.

Value

The output is a list compatible with the 2-d plotting functions persp, image, and contour, etc. The list contains...

x Vector of with length(x) == gridlen of increasing X grid locations
y Vector of with length(y) == gridlen of increasing Y grid locations
z matrix of interpolated responses \( Z = f(X, Y) \) where \( z[i,j] \) contains an estimate of \( f(x[i], y[j]) \)

Note

As mentioned above, the default span = 0.1 parameter is signifigantly smaller that the default loess setting. This asserts a tacit assumption that the input is densely packed and that the variance in the data is be small. Such should be the case when the data are output from a tgp regression – this function was designed specifically for this situation. For data that is random or sparse, simply choose higher setting, e.g., the default interp setting of span = 0.75, or a more intermediate setting of span = 0.5 as in the example below

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

interp, loess, persp, image, contour
Examples

```r
# random data
ed <- exp2d.rand()

# higher span = 0.5 required because the data is sparse
# and was generated randomly
ed.g <- interp.loess(ed$X[,1], ed$X[,2], ed$Z, span=0.5)

# perspective plot
persp(ed.g)
```

lhs

*Latin Hypercube sampling*

Description

Draw a (random) Latin Hypercube (LH) sample of size \( n \) from in the region outlined by the provided rectangle

Usage

```r
lhs(n, rect)
```

Arguments

- `n` Size of the LH sample
- `rect` Rectangle describing the domain from which the LH sample is to be taken. The rectangle should have exactly two columns (i.e., `dim(rect)[2] = 2`), and number of rows equal to the dimension of the domain. For 1-d data, a vector of length 2 is allowed

Value

The output is a matrix with \( n \) rows and `dim(rect)[1]` columns. Each of the \( n \) rows represents a sample

Note

The domain bounds specified by the rows of `rect` can be specified backwards with no change in effect

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

mapT

**See Also**

tgp.design, dopt.gp

**Examples**

```r
# get and plot a 2-d LH design
s1 <- lhs(10, rbind(c(-2,3), c(0.5, 0.8)))
plot(s1)

# plot a grid to show that there is one sample
# in each grid location
abline(v=seq(-2,3,length=11), lty=2, col=3)
abline(h=seq(0.5,0.8,length=11), lty=2, col=3)
```

---

**mapT**

Plot the MAP partition, or add one to an existing plot

**Description**

Plot the maximum a’ posteriori (MAP) tree from a "tgp"-class object, or add one on top of an existing plot. Like plot.tgp, projections and slices of trees can be plotted as specified

**Usage**

```r
mapT(out, proj = NULL, slice = NULL, add = FALSE, lwd = 2, ...)
```

**Arguments**

- `out`: "tgp"-class object which is the output of one the model functions with tree support (e.g. btgpllm)
- `proj`: 1-or-2-Vector describing the dimensions to be shown in a projection. The argument is ignored for 1-d data, i.e., if x$d == 1. For 2-d data, no projection needs be specified—the default argument (proj = NULL) will result in a 2-d perspective or image plot. 1-d projections of 2-d or higher data are are supported, e.g., proj = c(2) would show the second variable projection. For 3-d data or higher, proj=NULL defaults to proj = c(1,2) which plots a 2-d projection for the first two variables. Slices have priority over projections—see next argument (slice)—when non-null arguments are provided for both.
- `slice`: list object with x and z fields, which are vectors of equal length describing the slice to be plotted, i.e., which z-values of the x$d - 2 inputs x$X and x$XX should be fixed to in order to obtain a 2-d visualization. For example, for 4-d data, slice = list(x=(2,4), z=c(0.2, 1.5) will result in a 2-d plot of the first and third dimensions which have the second and fourth slice fixed at 0.5 and 1.5. The default is NULL, yielding to the proj argument. Argument is ignored for 1-d data, i.e., if x$d == 1
- `add`: Specify whether the to add partitions to an existing plot (add = TRUE) or to make a new plot showing the data out$X along with the partitions (default add = FALSE)
- `lwd`: Plotting argument specifying the width of the lines used to depict the partitions
- `...`: Additional arguments to plot used when add = FALSE
**partition**

**Value**

The only output of this function is a beautiful region-representation of the MAP tree.

**Note**

For examples, see `vignette("tgp")` and the examples provided in the documentation for the `tgp.design` function

**Author(s)**

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

**References**

[http://www.ams.ucsc.edu/~rbgramacy/tgp.html](http://www.ams.ucsc.edu/~rbgramacy/tgp.html)

**See Also**

`plot.tgp`, `tgp.trees`, `tgp.design`, `vignette("tgp")`

---

**partition**  
*Partition data according to the MAP tree*

**Description**

Partition data according to the maximum a’ posteriori (MAP) tree contained in a "tgp"-class object.

**Usage**

`partition(X, out)`

**Arguments**

- **X**  
  data.frame, matrix, or vector of inputs `X` with the same dimension of `out$X`, i.e., `dim(X)[2] == dim(out$X)[2]`

- **out**  
  "tgp"-class object which is the output of one the model functions with tree support (e.g. `btgpllm`, `btgp`, `btlm`, or `tgp`)

**Value**

Output is a list of `data.frames` populated with the inputs `X` contained in each region of the partition of the MAP tree in the "tgp"-class object `out`

**Author(s)**

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

**References**

[http://www.ams.ucsc.edu/~rbgramacy/tgp.html](http://www.ams.ucsc.edu/~rbgramacy/tgp.html)
See Also

tgp.design, tgp.trees

Examples

# 2-d Exponential data
# (This example is based on random data.
# It might be fun to run it a few times)
#
# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX

# fit treed GP LLM model to data w/o prediction
# basically just to get MAP tree (and plot it)
out <- btgpllm(X=X, Z=Z, pred.n=FALSE, BTE=c(2000,3000,2))
tgp.trees(out)

# find a treed sequential D-Optimal design
# with 10 more points
Xcand.parts <- partition(Xcand, out)

plot.tgp

Plotting for Treed Gaussian Process Models

Description

A generic function for plotting of "tgp" class objects. 1-d posterior mean and error plots, 2-d posterior mean and error image and perspective plots, and 3+-dimensional mean and error image and perspective plots are supported via projection and slicing.

Usage

plot.tgp(x, pparts = TRUE, proj = NULL, slice = NULL, map = NULL,
as = NULL, layout = "both", main = NULL, xlab = NULL,
 ylab = NULL, zlab = NULL, pc = "pc", method = "loess",
gridlen = 40, span = 0.1, ...)

Arguments

x
"tgp" class object that is the output of one of the tgp functions: blm, btlm,
bgp, bgpllm, btgp, or btgpllm

pparts
If TRUE, partition-regions are plotted (default), otherwise they are not

proj
1-or-2-Vector describing the dimensions to be shown in a projection. The argument is ignored for 1-d data, i.e., if x$d == 1. For 2-d data, no projection needs be specified—the default argument (proj = NULL) will result in a 2-d perspective or image plot. 1-d projections of 2-d or higher data are supported, e.g., proj = c(2) would show the second variable projection. For 3-d data or higher, proj=NULL defaults to proj = c(1,2) which plots a 2-d
projection for the first two variables. Slices have priority over the projections—see next argument (slice)—when non-null arguments are provided for both.

\textbf{slice}  
\texttt{list} object with \texttt{x} and \texttt{z} fields, which are vectors of equal length describing the slice to be plotted, i.e., which \texttt{z}-values of the \texttt{x$d} - 2 inputs \texttt{x$X} and \texttt{x$XX} should be fixed to in order to obtain a 2-d visualization. For example, for 4-d data, \texttt{slice = list(x=(2,4), z=c(0.2, 1.5))} will result in a 2-d plot of the first and third dimensions which have the second and fourth slice fixed at 0.5 and 1.5. The default is \texttt{NULL}, yielding to the \texttt{proj} argument. Argument is ignored for 1-d data, i.e., if \texttt{x$d == 1}

\textbf{map}  
Optional 2-d map (longitude and latitude) from \texttt{library(maps)} to be shown on top of image plots

\textbf{as}  
Optional string indicator for plotting of adaptive sampling statistics: specifying \texttt{as = "alm"} for ALM, \texttt{as = "alc"} for ALC, and \texttt{as = "ego"} for EGO. The default \texttt{as = NULL} plots error-bars (1d-plots) or error magnitudes (2d-plots), which is essentially the same as \texttt{as = "alm"}

\textbf{layout}  
Specify whether to plot the mean predictive surface (\texttt{layout = "surf"}), the error or adaptive sampling statistics (\texttt{layout = "as"}), or default (\texttt{layout = "both"}) which shows both

\textbf{main}  
Optional character string to add to the main title of the plot

\textbf{xlab}  
Optional character string to add to the x label of the plots

\textbf{ylab}  
Optional character string to add to the y label of the plots

\textbf{zlab}  
Optional character string to add to the z label of the plots; ignored unless \texttt{pc = "p"}

\textbf{pc}  
Selects perspective-posterior mean and image-error plots (\texttt{pc = "pc"}, the default) or a double-image plot (\texttt{pc = "c"})

\textbf{method}  
Method of interpolation onto a regular grid for perspective and image plotting of 2-d slices and projections. Ignored for 1-d plots and projections. Default is \texttt{method = "loess"} which results in the use of the built-in \texttt{loess} function via the provided \texttt{interp.loess} interface. The lowess kernel span can be specified with the \texttt{span} parameter to this function. Another option, \texttt{method = "akima"} uses the \texttt{akima} function \texttt{interp.old}. In general, \texttt{akima} provides a better (linear) interpolation, but the current version(s) has been buggy (see notes below), and so is not the default

\textbf{gridlen}  
Number of regular grid points for 2-d slices and projections. Default is \texttt{gridlen = 40}. Ignored for 1-d plots and projections

\textbf{span}  
Span for \texttt{loess} kernel. See \texttt{help(loess)} for mode details. This argument is ignored for 1-d plots and projections, and 2-d plots with \texttt{method = "akima"}. The \texttt{tgp} package default (\texttt{span = 0.1}) is set lower than the \texttt{loess} default. Decreasing the span any further, or when the data or predictive locations are sparse, warnings may be given and ugly plots may result. In this case, try increasing the span, or use the \texttt{method = "akima"} option

\textbf{...}  
Extra arguments to 1-d (\texttt{plot}) and 2-d plotting functions \texttt{persp} and \texttt{image}

\section{Value}

The only output of this function is beautiful plots.
Note

This plotting function is provided with the intention that it will be used as an aid in the visualization of "tgp"-class objects. Users are encouraged to use the source code for this function in order to develop custom plotting functions.

There seems to be two bugs in the `interp` function of the `akima` package. The first, benign, bug causes NAs to appear when plotting 2-d surfaces. The result is white rectangles (or “holes”) in the surfaces. This affect is most severe when x$x and x$XX together form a regular grid, or close to a regular grid. The second, malign, bug can cause segmentation faults when plotting projections, causing the entire R session to crash. Please note this is not a bug in the `tgp` package, rather it is a bug in the `akima` package. Instead, interpolation by the `loess` function is provided as a workaround, and is the default (method = "loess"). 1-d projections for 3-d or higher data are also available by specifying a 1-d projection argument (e.g. proj=c(1) for projecting onto the first input variable).

For examples, see `vignette("tgp")` and the help files of those functions in "See Also", below

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

`plot, bgpllm, btlm, blm, bgp, btgpllm, tgp, tgp.trees, mapT, interp, loess`

tgp-internal  Internal Treed Gaussian Process Model Functions

Description

Internal Treed Gaussian Process Model functions

Details

These are not to be called by the user (or in some cases are just waiting for proper documentation to be written :)).

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html
The Treed Gaussian Process Model Package

Description

A Bayesian nonstationary nonparametric regression and design package implementing an array of models of varying flexibility and complexity.

Details

This package implements Bayesian nonparametric and nonstationary regression with “treed Gaussian process models”. The package contains functions which facilitate inference for six regression models of varying complexity using Markov chain Monte Carlo (MCMC): linear model, linear CART (Classification and Regression Tree), Gaussian process (GP), GP with jumps to the limiting linear model (LLM), treed GP, and treed GP LLM. R provides an interface to the C/C++ backbone, and also provides a mechanism for graphically visualizing the results of inference and posterior predictive surfaces under the models. A limited set of experimental design and adaptive sampling functions are also provided.

For a complete list of functions, use library(help="tgp").

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References


http://www.ams.ucsc.edu/~rbgramacy/tgp.html

generic interface to treed Gaussian process models

Description

A generic interface to treed Gaussian process models used by many of the functions of class "tgp": bgpllm, btlm, blm, bgp, btgpllm bgp, and plot.tgp.tgp.trees. This more complicated interface is provided for a finer control of the model parameterization.

Usage

tgp(X, Z, XX = NULL, BTE = c(2000, 7000, 2), R = 1, m0r1 = FALSE, linburn = FALSE, params = NULL, pred.n = TRUE, ds2x = FALSE, ego = FALSE, traces = FALSE, verb = 1)
**tgp**

Arguments

- **X**
  - data.frame, matrix, or vector of inputs \( X \)

- **Z**
  - Vector of output responses \( Z \) of length equal to the leading dimension (rows) of \( X \)

- **XX**
  - Optional data.frame, matrix, or vector of predictive input locations with the same number of columns as \( X \)

- **BTE**
  - 3-vector of Monte-carlo parameters (B)urn in, (T)otal, and (E)very. Predictive samples are saved every E MCMC rounds starting at round B, stopping at T.

- **R**
  - Number of repeats or restarts of BTE MCMC rounds, default R=1 is no restarts

- **m0r1**
  - If TRUE the responses \( Z \) will be scaled to have a mean of zero and a range of 1; default is FALSE

- **linburn**
  - If TRUE initializes MCMC with B (additional) rounds of Bayesian linear CART (bcart); default is FALSE

- **params**
  - Generic parameters list which can be provided for a more flexible model. See tgp.default.params for more details about the parameter list

- **pred.n**
  - TRUE (default) value results in prediction at the inputs \( X \); FALSE skips prediction at \( X \) resulting in a faster implementation

- **ds2x**
  - TRUE results in ALC (Active Learning–Cohn) computation of expected reduction in uncertainty calculations at the \( XX \) locations, which can be used for adaptive sampling; FALSE (default) skips this computation, resulting in a faster implementation

- **ego**
  - TRUE results in EGO (Expected Global Optimization) computation of expected information about the location of the minimum reduction in uncertainty calculations at the \( XX \) locations, which can be used for adaptive sampling; FALSE (default) skips this computation, resulting in a faster implementation

- **traces**
  - TRUE results in a saving of samples from the posterior distribution for most of the parameters in the model. The default is FALSE for speed/storage reasons. See note below

- **verb**
  - Level of verbosity of R-console print statements: from 0 (none); 1 (default) which shows the “progress meter”; 2 includes an echo of initialization parameters; up to 3 and 4 (max) with more info about successful tree operations.

Value

tgp returns an object of class "tgp". The function plot.tgp can be used to help visualize results.

An object of type "tgp" is a list containing at least the following components... The final two (parts & trees) are tree-related outputs unique to the T (tree) class functions-- those which have a positive first (alpha) parameter in params$tree <- c(alpha, beta, minpart. Tree viewing is supported by tgp.trees.

- **state**
  - unsigned short[3] random number seed to C

- **X**
  - Input argument: data.frame of inputs \( X \)

- **n**
  - Number of rows in \( X \), i.e., dim(\( X \))[1]

- **d**
  - Number of cols in \( X \), i.e., dim(\( X \))[2]

- **Z**
  - Vector of output responses \( Z \)

- **XX**
  - Input argument: data.frame of predictive locations \( XX \)
Number of rows in XX, i.e., \( \text{dim}(XX) \) [1]

Input argument: Monte-carlo parameters

Input argument: restarts

Input argument: initialize MCMC with linear CART

list of model parameters generated by \texttt{tgp.default.params}

Double-representation of model input parameters used by C-code

Vector of mean predictive estimates at X locations

Vector of 5% predictive quantiles at X locations

Vector of 95% predictive quantiles at X locations

Vector of quantile norms \( Zp.q2 - Zp.q1 \)

Vector of 5% predictive quantiles at XX locations

Vector of 95% predictive quantiles at XX locations

Vector of quantile norms \( ZZ.q2 - ZZ.q1 \), used by the Active Learning–MacKay (ALM) adaptive sampling algorithm

If argument \texttt{ds2x=TRUE}, this vector contains ALC statistics for XX locations

If argument \texttt{ego=TRUE}, this vector contains EGO statistics for XX locations

Name of response Z if supplied by \texttt{data.frame} in argument, or “z” if none provided

Internal representation of the regions depicted by partitions of the maximum a’ posteriori (MAP) tree

list of trees (\texttt{maptree} representation) which were MAP as a function of each tree height sampled between MCMC rounds B and T

list containing traces of most of the model parameters and posterior predictive distributions at input locations XX. See note below

Input argument: verbosity level

Inputs X, XX, Z containing NaN, NA, Inf are discarded with non-fatal warnings

Upon execution, MCMC reports are made every 1,000 rounds to indicate progress

Stationary (non-treed) processes on larger inputs (e.g., X, Z) of size greater than 500, *might* be slow in execution, especially on older machines. Once the C code starts executing, it can be interrupted in the usual way: either via Ctrl-C (Unix-alikes) or pressing the Stop button in the R-GUI. When this happens, interrupt messages will indicate which required cleanup measures completed before returning control to R

Regarding \texttt{traces=TRUE}: Samples from the posterior will be collected for all parameters in the model, except those of the hierarchical priors, e.g., \( b_0 \), etc. Traces for some parameters are stored in memory, others in files. GP parameters are collected with reference to the locations in XX, resulting \( nn=\text{dim}(XX) \) [2] traces of d, g, s2, tau2, etc. Therefore, it is recommended that \( nn \) is chosen to be a small, representative, set of input locations. Besides GP parameters, traces are saved for the tree partitions, areas under the LLM, log posterior (as a function of tree height), and samples ZZ from the posterior predictive distribution at XX

Robert B. Gramacy \{rbgramacy@ams.ucsc.edu\}
References


http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

tgp.default.params, bgpllm, btlm, blm, bgp, btgppllm, bgp, plot.tgp, tgp.trees

Examples

```r
## Many of the examples below illustrate the above
## function(s) on random data. Thus it can be fun
## (and informative) to run them several times.
##
# simple linear response
#
# input and predictive data
X <- seq(0,1,length=50)
XX <- seq(0,1,length=99)
Z <- 1 + 2*X + rnorm(length(X),sd=0.25)
# out <- blm(X=X, Z=Z, XX=XX) # try Linear Model with tgp
p <- tgp.default.params(2)
p$tree <- c(0,0,10) # no tree
p$gamma <- c(-1,0.2,0.7) # force llm
out <- tgp(X=X,Z=Z,XX=XX,params=p)
plot(out) # plot the surface

# 1-d Example
#
# construct some 1-d nonstationary data
X <- seq(0,20,length=100)
XX <- seq(0,20,length=99)
Z <- (sin(pi*X/5) + 0.2*cos(4*pi*X/5)) * (X <= 9.6)
lin <- X>9.6;
Z[lin] <- -1 + X[lin]/10
Z <- Z + rnorm(length(Z), sd=0.1)
# out <- btlm(X=X, Z=Z, XX=XX) # try Linear CART with tgp
```
p <- tgp.default.params(2)
p$gamma <- c(-1,0.2,0.7)  # force llm
out <- tgp(X=X,Z=Z,XX=XX,params=p)
plot(out)  # plot the surface

tgp.trees(out)  # plot the MAP trees

# out <- btgp(X=X, Z=Z, XX=XX)  # use a treed GP with tgp
p <- tgp.default.params(2)
p$gamma <- c(0,0.2,0.7)  # force no llm
out <- tgp(X=X,Z=Z,XX=XX,params=p)
plot(out)  # plot the surface
tgp.trees(out)  # plot the MAP trees

#
# 2-d example
# (using the isotropic correlation function)
#
# construct some 2-d nonstationary data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
XX <- exp2d.data$XX

# try a GP with tgp
# out <- bgp(X=X, Z=Z, XX=XX, corr="exp")
p <- tgp.default.params(3)
p$tree <- c(0,0,10)  # no tree
p$gamma <- c(0,0.2,0.7)  # no llm
p$corr <- "exp"
out <- tgp(X=X,Z=Z,XX=XX,params=p)
plot(out)  # plot the surface

# try a treed GP LLM with tgp
# out <- btgppllm(X=X, Z=Z, XX=XX, corr="exp")
p <- tgp.default.params(3)
p$corr <- "exp"
out <- tgp(X=X,Z=Z,XX=XX,params=p)
plot(out)  # plot the surface
tgp.trees(out)  # plot the MAP trees

#
# Motorcycle Accident Data
#
# get the data
require(MASS)

# try a custom treed GP LLM with tgp, without m0r1
p <- tgp.default.params(2)
p$bprior <- "b0"  # beta linear prior for common mean
p$nug.p <- c(1.0,0.1,1.0,0.1)  # mixture nugget prior
out <- tgp(X=mcycle[,1], Z=mcycle[,2], params=p,
BTE=c(2000,22000,2))  # run mcmc longer
plot(out)  # plot the surface
tgp.trees(out)  # plot the MAP trees

# for other examples try the demos or the vignette
tgp.default.params  Default Treed Gaussian Process Model Parameters

Description

Construct a default list of parameters to the tgp function– the generic interface to treed Gaussian process modeling

Usage

tgp.default.params(col, base = "gp")

Arguments

col  number of input dimensions dim(X) [2] plus 1
base  Base model to be used. Right now, the only supported option is the default, base = "gp". Future versions of this package will support other base models.

Value

The output is the following list of params...

corr  "expsep" separable power exponential family correlation model; alternate is "exp" isotropic power family
bprior  Linear (beta) prior, default is "bflat"; alternates include "b0" hierarchical Normal prior, "bmle" empirical Bayes Normal prior, "bcart" Bayesian linear CART style prior from Chipman et al. "b0tau" a independent Normal prior with inverse-gamma variance.
start  c(0.5,0.1,1.0,1.0) starting values for range d, nugget g, σ², and τ²
beta  rep(0,d) starting values for beta linear parameters
tree  c(0.25,2,10) tree prior process parameters c(alpha, beta, nmin) specifying

   \[ p_{\text{mix}}(\eta, T) = \alpha \ast (1 + \eta)^3 \]

with zero probability to trees with partitions containing less than nmin data points

s2.p  c(5,10) σ² inverse-gamma prior parameters c(a0, g0) where g0 is scale (1/rate) parameter
tau2.p  c(5,10) τ² inverse-gamma prior parameters c(a0, g0) where g0 is scale (1/rate) parameter
d.p  c(1.0,20.0,10.0,10.0) Mixture of gamma prior parameter (initial values) for for the range parameter c(a1,g1,a2,g2) where g1 and g2 are scale (1/rate) parameters
d.p  codec(1,1,1,1) Mixture of gamma prior parameter (initial values) for for the range parameter c(a1,g1,a2,g2) where g1 and g2 are scale (1/rate) parameters; default reduces to simple exponential prior
Limiting Linear model parameters \( c(g, t_1, t_2) \), with growth parameter \( g > 0 \) minimum parameter \( t_1 \geq 0 \) and maximum parameter \( t_1 \geq 0 \), where \( t_1 + t_2 \leq 1 \) specifies

\[
p(b|d) = t_1 + \exp\left\{ -g(t_2 - t_1) \right\}
\]

Hierarchical exponential distribution parameters to \( a_1, g_1, a_2, \) and \( g_2 \) of the prior distribution for the range parameter \( d.p \); fixed indicates that the hierarchical prior is “turned off”

Hierarchical exponential distribution parameters to \( a_1, g_1, a_2, \) and \( g_2 \) of the prior distribution for the nug parameter \( nug.p \); "fixed" indicates that the hierarchical prior is “turned off”

Hierarchical exponential distribution prior for \( a_0 \) and \( g_0 \) of the prior distribution for the \( s_2 \) parameter \( s_2.p \); "fixed" indicates that the hierarchical prior is “turned off”

Hierarchical exponential distribution prior for \( a_0 \) and \( g_0 \) of the prior distribution for the \( s_2 \) parameter \( tau2.p \); "fixed" indicates that the hierarchical prior is “turned off”

Note

Please refer to the examples for the functions in the “See Also” list below and vignette("tgp")

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References


http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

tgp

tgp.design

Sequential Treed D-Optimal Design for Treed Gaussian Process Models

Description

Based on the maximum a’ posteriori (MAP) treed partition extracted from a "tgp"-class object, calculate independent sequential treed D-Optimal designs in each of the regions.

Usage

tgp.design(howmany, Xcand, out)
Arguments

- **howmany**: Number of new points in the design. Must be less than the number of candidates contained in Xcand, i.e., howmany <= dim(Xcand)[1]

- **Xcand**: data.frame, matrix or vector of candidates from which new design points are subsampled. Must have the same dimension as out$X

- **out**: "tgp" class object which is the output of one of the model functions which has tree support, e.g., btgpllm, btgp, btlm, or tgp

Details

This function partitions Xcand and out$X based on the MAP tree (obtained on "tgp"-class out with `partition`) and calls dopt.gp in order to obtain a D-optimal design under independent stationary Gaussian processes models defined in each region. The aim is to obtain a design where new points from Xcand are spaced out relative to themselves, and relative to the existing locations (out$X) in the region. The number of new points from each region is proportional to the number of candidates Xcand in the region.

Value

Output is a list of data.frames containing XX design points for each region of the MAP tree in out

Note

Input Xcand containing NaN, NA, Inf are discarded with non-fatal warnings

D-Optimal computation in each region is preceded by a print statement indicated the number of new locations to be chosen and the number of candidates in the region. Other than that, there are no other indicators of progress. You will have to be patient. Creating treed sequential D-optimal designs is no speedy task. At least it faster than the non-treed version (see dopt.gp).

This function is still considered experimental. (Pardon the pun.)

The example below is also part of vignette("tgp")

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References


See Also

- bgpllm, btlm, blm, bgp, btgpllm, tgp, plot.tgp, dopt.gp, lhs, partition
Examples

# 2-d Exponential data
# (This example is based on random data.
# It might be fun to run it a few times)
#
# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX

# fit treed GP LLM model to data w/o prediction
# basically just to get MAP tree (and plot it)
out <- btgpllm(X=X, Z=Z, pred.n=FALSE, corr="exp")
tgp.trees(out)

# find a treed sequential D-Optimal design
# with 10 more points. It is interesting to
# contrast this design with one obtained via
# the dopt.gp function
XX <- tgp.design(10, Xcand, out)

# now fit the model again in order to assess
# the predictive surface at those new design points
dout <- btgpllm(X=X, Z=Z, XX=XX, corr="exp")
plot(dout)

---

tgp.get.partitions  
Get partition of data from maximum a’ posteriori tree

Description

Partition data according to the maximum a’ posteriori (MAP) tree contained in a "tgp"-class object.

Usage

tgp.get.partitions(X, out)

Arguments

X data.frame, matrix, or vector of inputs X with the same dimension of out$X, i.e., \( \text{dim}(X)[2] == \text{dim}(\text{out}$X)[2] \)

out "tgp"-class object which is the output of one the model functions with tree support (e.g. btgpllm, btgp, btlm, or tgp)

Value

Output is a list of data.frames populated with the inputs X contained in each region of the partition of the MAP tree in the "tgp"-class object out
tgp.trees

Author(s)

Robert B. Gramacy ⟨rbgramacy@ams.ucsc.edu⟩

References


http://www.ams.ucsc.edu/~rbgramacy/tgp.php

See Also
tgp.design, tgp.trees

Examples

```r
# # 2-d Exponential data  # (This example is based on random data.)  # It might be fun to run it a few times)
#
# get the data
exp2d.data <- exp2d.rand()
X <- exp2d.data$X; Z <- exp2d.data$Z
Xcand <- exp2d.data$XX

# fit treed GP LLM model to data w/o prediction
# basically just to get MAP tree (and plot it)
out <- btgpllm(X=X, Z=Z, pred.n=FALSE, BTE=c(2000,3000,2))
tgp.trees(out)

# find a treed sequential D-Optimal design
# with 10 more points
Xcand.parts <- tgp.get.partitions(Xcand, out)
```

See Also
tgp.design, tgp.trees

Description

Plot the maximum a’ posteriori (MAP) tree as a function of tree height, and show the log posterior probabilities for comparison.

Usage

tgp.trees(out, which = NULL, main = NULL, ...)

Plot the MAP Tree for each height encountered by the Markov Chain
Arguments

out    "tgp"-class object which is the output of one the model functions with tree support (e.g. btgpllm)
which  Index vector of length less than length(out$trees) describing trees to plot. Default (NULL) is to plot all trees, one for each height encountered when sampling from the Markov chain of the tree posterior. This is equivalent to which = 1:length(out$trees). For example, the first, third, and forth trees would result if which = c(1,3,4)
main   Optional character string to add to the main title of the plot
...    Extra arguments to the draw.tree function from maptree

Value

The only output of this function is beautiful tree diagrams.

Note

Plotting trees that the maptree library is installed, which itself requires that the combinat library also be installed.

See vignette("tgp") and the examples sections of the functions under “See Also”, below

Author(s)

Robert B. Gramacy (rbgramacy@ams.ucsc.edu)

References

http://www.ams.ucsc.edu/~rbgramacy/tgp.html

See Also

bgpllm,btlm,blm,bgp,btgpllm,plot.tgp,mapT,vignette("tgp")
Index

*Topic datasets
  exp2d, 9
  exp2d.rand, 10
  friedman.1.data, 11
*Topic design
  dopt.gp, 7
  lhs, 13
  tgp.design, 26
*Topic hplot
  mapT, 15
  plot.tgp, 17
  tgp-package, 26
  tgp.trees, 29
*Topic internal
  tgp-internal, 19
*Topic loess
  interp.loess, 12
*Topic models
  btgp, 1
  tgp, 20
  tgp-package, 20
  tgp.default.params, 25
*Topic nonlinear
  btgp, 1
  tgp, 20
*Topic nonparametric
  btgp, 1
  tgp, 20
  tgp-package, 20
  tgp.default.params, 25
*Topic smooth
  btgp, 1
  interp.loess, 12
  tgp, 20
  tgp-package, 20
  tgp.default.params, 25
*Topic spatial
  btgp, 1
  dopt.gp, 7
  lhs, 13
  tgp, 20
  tgp-package, 20
  tgp.default.params, 25
  tgp.design, 26
*Topic tree
  btgp, 1
  mapT, 15
  partition, 16
  plot.tgp, 17
  tgp, 20
  tgp-package, 20
  tgp.default.params, 25
  tgp.design, 26
  tgp.get.partitions, 28
  tgp.trees, 29
bgp, 9, 11, 12, 17, 19, 20, 23, 27, 30
bgp (btgp), 1
bgpllm, 9, 11, 12, 17, 19, 20, 23, 27, 30
bgpllm (btgp), 1
blm, 9, 11, 12, 17, 19, 20, 23, 27, 30
blm (btgp), 1
btgp, 1, 16, 17, 27, 28
btgpllm, 9, 11, 12, 15–17, 19, 20, 23, 27, 28, 30
btgpllm (btgp), 1
btlm, 9, 11, 12, 16, 17, 19, 20, 23, 27, 28, 30
btlm (btgp), 1
check.matrix (tgp-internal), 19
check.proj (tgp-internal), 19
check.slice (tgp-internal), 19
contour, 12, 13
dopt.gp, 7, 14, 27
draw.tree, 30
exp2d, 9, 10, 11
exp2d.rand, 9, 10
expand.grid, 12
franify.X (tgp-internal), 19
friedman.1.data, 11
getlocs (tgp-internal), 19
image, 12, 13
interp, 12, 13, 19
interp.loess, 12, 18
interp.old, 18
layout.torture(tgp-internal), 19
lhs, 8, 13, 27
loess, 12, 13, 18, 19
mapT, 15, 19, 30
mean0.range1(tgp-internal), 19
partition, 16, 27
persp, 12, 13
plot, 19
plot.tgp, 4, 5, 16, 17, 20, 21, 23, 27, 30
predict.loess, 12
print.tgp(tgp-internal), 19
print.tgptraces(tgp-internal), 19
slice.contour(tgp-internal), 19
slice.image(tgp-internal), 19
slice.interp(tgp-internal), 19
slice.persp(tgp-internal), 19
tgp, 1, 3–5, 9, 11, 12, 16, 17, 19, 20, 25–28
tgp-internal, 19
tgp-package, 20
tgp.check.params(tgp-internal), 19
tgp.choose.as(tgp-internal), 19
tgp.cleanup(tgp-internal), 19
tgp.default.params, 4, 21–23, 25
tgp.design, 8, 14–16, 26, 29
tgp.get.partitions, 28
tgp.get.trees(tgp-internal), 19
tgp.locations(tgp-internal), 19
tgp.partition(tgp-internal), 19
tgp.plot.parts.1d(tgp-internal), 19
tgp.plot.parts.2d(tgp-internal), 19
tgp.plot.proj(tgp-internal), 19
tgp.plot.slice(tgp-internal), 19
tgp.plot.tree(tgp-internal), 19
tgp.read.traces(tgp-internal), 19
tgp.trees, 4, 5, 16, 19–21, 23, 29, 29
undo.mean0.range1(tgp-internal), 19