Package 'MUCM'

April 6, 2016

Type Package

Title MUCM

Version 0.2-2

Date 2016-03-23

Author Sajni Malde, Jeremy Oakley, David Wyncoll

Maintainer Jeremy Oakley <j.oakley@sheffield.ac.uk>

Description Gaussian process emulator

License file LICENSE

Depends R (>= 2.10)

Imports mvtnorm, Hmisc, Formula, ggplot2, fields

LazyData yes

ByteCompile yes

NeedsCompilation no

RoxygenNote 5.0.1

R topics documented:

MUCM-package *MUCM*

Description

This package allows the user to estimate the output of a simulator at any point, without actually running it, given a few hundred simulator runs (referred to as the training runs). The emulator essentially determines using Bayesian Analysis the posterior mean and variance of a Gaussian Process for a given input data set, conditioned on the training runs and a user-specified prior mean function. The emphasis is on complex codes that take weeks or months to run, and that have a large number of input parameters; many metrological prediction models fall into this class. A working example is given here for the main functions of this package, which should be the first point of reference.

Author(s)

Sajni Malde, Jeremy Oakley (<j.oakley@sheffield.ac.uk>) and David Wyncoll.

References

J. Oakley 1999. 'Bayesian uncertainty analysis for complex computer codes', PhD thesis, University of Sheffield.
Bastos, L. S. and O'Hagan, A. (2009). Diagnostics for gaussian process emulators, Technometrics, 51 (4): 425-438.

Examples

```
# Plot the training data to look for trends
plot(surfebm[1:25, 1:3], lower.panel = NULL)

# Fit the emulator using a linear prior mean (default) and
# a Gaussian correlation function
fit <- fitEmulator(inputs = surfebm[1:25, 1:2],
                   outputs = surfebm[1:25, 3, drop = FALSE],
                   cor.function = corGaussian)

# Use fitted emulator to predict posterior means and variances at the new points
predictions <- predict(fit, surfebm[26:35, 1:2], sd = FALSE, var.cov = TRUE)

# Compare predictions with true values for the new inputs
# Can also compare accuracy of prediction based on posterior variance
validateEmulator(fit, surfebm[26:35, 3], predictions, plot = TRUE)
```

backPredict *Transform output from Prcomp / Princomp*

Description

These functions reverse the outcome of the `predict` function when the class of the first argument is either `prcomp` or `princomp`. It is used to transform data where the variables are principal components of a dataset of original variables, given the `object`.

Usage

```
backPredict(object, newdata)

backPredictVar(object, new.vardata, only.var = TRUE)
```

Arguments

object	Object of class inheriting from princomp or prcomp
newdata	A matrix of principal components which is to be converted back to the original output variables. An important feature here is that newdata can have a reduced number of dimensions with which to predict, however it has to be a matrix.
new.vardata	A matrix of variances where each column represents the variances of one principal component's prediction for many observations, and each row represents the variances of multiple principal components of one observation.
only.var	This parameter lets the user choose how the output of the function backPredictVar should appear. The default option is set to TRUE. In this case the output of the function only contains the variances of each observation with the output variable. If FALSE this function returns an array of 3 dimensions representing the cross covariances of output variables and observations. The 1st dimension is the number of rows in new.vardata and the 2nd and 3rd dimensions represent a variance covariance matrix between output j and output k for row i in new.vardata.

Details

backPredict defines a function to reverse the outcome of the predict.prcomp or predict.princomp function. It is used to transform data (typically values predicted using regressions) where the variables are principal components of a dataset of original variables, given in the object.

backPredictVar defines a similar function to reverse the outcome of the predict.prcomp or predict.princomp function. It is used to transform data that represents variances of principal components of an original dataset given in the object. It uses the linear variance transformation property.

Value

backPredict returns a matrix of data in terms of the original output variables.

backPredictVar returns a matrix of variances (and covariances if only.var = FALSE) of each observation with the original output variable

Author(s)

Sajni Malde

See Also

princomp, prcomp, predict.prcomp

Examples

```
library(mvtnorm)

#defining data
Sigma <- matrix(c(10.3, 3.6,3.6,2.4),2,2)
out <- rmvnorm(n=100000, c(1.2,2.4), Sigma); colnames(out) <- c('Y1', 'Y2')
cov(out)
new.out <- rmvnorm(n=100000, c(1.2,2.4), Sigma); colnames(new.out) <- c('Y1', 'Y2')

# using the 'princomp' function to convert the data
out.pca <- princomp(out) # converting the data to principal components
# predicting principal components for a new data set
new.out.pca<- predict(out.pca, new.out)

# applying backPredict and comparing values
backpredict <- backPredict(out.pca, new.out.pca)
head(backpredict) # should be quite similar to head(new.out)
# reducing the dimensions # only 1 components used to 'backPredict'
backpredict2 <- backPredict(out.pca, new.out.pca[, 1, drop = FALSE])

# applying backPredictVar and comparing values
# check to see if assuming off diagonals are equal to zero is a reasonable assumption
cov(new.out.pca)
(backpredict.var <- backPredictVar(out.pca, new.vardata = matrix(diag(cov(new.out.pca)),
                                    ncol = 2), only.var = FALSE)[1,,])
# backpredict.var should be very close to cov(out)
```

`buildCovMat` *Building a Covariance Matrix for the LMC method*

Description

This function is used to build a covariance matrix for multivariate outputs.

Usage

```
buildCovMat(phi, sigma, inputs, inputs2, cor.function, n.outputs, ...)
```

Arguments

`phi`	Maximum likelihood estimate of the correlation length parameter.
`sigma`	Maximum likelihood estimate of the between outputs variance covariance matrix.
`inputs`	A data frame, matrix or vector containing the input values of the data.
`inputs2`	A data frame, matrix or vector containing additional input values. (Used when predicting the model at new points)
`cor.function`	Specifies a correlation function used as part of the prior information for the emulator.
`n.outputs`	Number of outputs.
`...`	additional arguments to be passed on to `cor.function`.

Value

The function returns a covariance matrix

Author(s)

Sajni Malde

`corGaussian` *Correlation functions*

Description

Functions to calculate the correlation between two points. The correlation functions give the correlation between the simulator output at any given vector of input values and the output at any other given input vector. Within this package, when these two vectors are the same then the correlation function `C` at input vector `x` is `C(x,x) = 1` In principal, the correlation function can take a very wide variety of forms. Only a few are included in this package, however, a user is able to define his own function.

Usage

```
corGaussian(inputs, inputs2, phi)

corMatern2.5(inputs, inputs2, phi)

corGaussianPeriodic(inputs, inputs2, phi, period)

corGaussianPeriodic2(inputs, inputs2, phi, period)

corCombined(inputs, inputs2, phi, cor.funcs, ...)
```

Arguments

`inputs`	A data frame, matrix or vector containing the input values of the data where each row gives the observations.
`inputs2`	An additional data frame, matrix or vector containing the input values of the data where each row gives the observations.
`phi`	Maximum likelihood estimate of the correlation length parameter (a vector).
`period`	A scalar or a vector indicating the period for each input parameter. If scalar, period for all input parameters will be set to the same scalar value as specified. Set period such that $cor(x_i, x_i + period) = 1$
`cor.funcs`	An ordered vector of length equal to number of inputs containing characters 'g', 'm2.5', 'gp', or 'gp2' as an indicator for whether a particular input should be calculated using `corGaussian`, `corMatern2.5`, `corGaussianPeriodic`, or `corGaussianPeriodic2` respectively.
`...`	additional arguments to be passed on to correlation functions.

Details

`corGaussian` uses the Gaussian correlation function defined as

$$\prod_{i=1}^{n} \exp[-\frac{(x_i - x_i')}{\delta_i}^2]$$

`corMatern2.5` uses the Matern correlation function with $\nu = 2.5$ defined as

$$(1 + 5^{0.5}r + \frac{5}{3}r^2)\exp(-5^{0.5}r)$$

where r is distance between inputs x_i and x_j, scaled by delta:

$$[\frac{(x_{i1} - x_{j1})^2}{\delta_1^2} + ... + \frac{(x_{id} - x_{jd})^2}{\delta_d^2}]^{0.5}$$

`corGaussianPeriodic` uses the Guassian periodic correlation function defined as

$$\exp(-\delta^2 sin(\frac{\pi}{period}|x_i - x_j|)^2)$$

`corGaussianPeriodic2` uses a different way of accounting for the periodicity in the data. Both `corGaussianPeriodic` and `corGaussianPeriodic2` give the same results.

`corCombined` is a function that allows the user to specify a different correlation function for each output (from the list of predefined functions).

Value

Returns the correlation matrix A calculated by the specific function.

Note

These functions use `rdist` from package **fields** for speed.

Author(s)

Originally written by Jeremy Oakley. Modified by Sajni Malde

See Also

See `optim` or `MCMCMetropolisGibbs` where the function is used.

`crossVal` *Cross Validation*

Description

Leave-one-out cross validation

Usage

```
crossVal(fit, lmcompare = FALSE, lmformula = fit$formula)
```

Arguments

fit	Object of class emulatorFit generated by fitEmulator or fitEmulatorData
lmcompare	Compare with a linear model fitted with OLS.
lmformula	The formula for the linear model to be used in the comparison, if lmcompare is TRUE.

Author(s)

Jeremy Oakley.

---

emPlot1D *emPlot1D*

Description

Plot emulator distribution across a single input

Usage

```
emPlot1D(fit, input = 1, lower = NULL, upper = NULL, alpha = 0.05)
```

Arguments

fit	Object of class emulatorFit generated by fitEmulator or fitEmulatorData
input	The default is set to 1
lower	The default is set to Null
upper	The default is set to Null
alpha	The default is set to 0.05

Author(s)

Jeremy Oakley.

---

fitEmulator *Fit Emulator*

Description

A function that fits an emulator to the data given resulting in an object of class emulatorFit.

Usage

```
fitEmulator(inputs, outputs, prior.mean = "linear",
  cor.function = corGaussian, method = c("separable", "LMC"), ...)

fitEmulatorLMC(inputs, outputs, prior.mean = "linear",
  cor.function = corGaussian, phi.opt, sigmasq.opt, N.iterations = 50,
  phi.init, sigmasq.init, MC.plot = FALSE, param = "matrix-log", ...)

fitEmulatorSEP(inputs, outputs, prior.mean = "linear",
  cor.function = corGaussian, phi.opt, sigmasq.opt, N.iterations = 50,
  phi.init, sigmasq.init, MC.plot = FALSE, nugget = NULL, ...)
```

Arguments

inputs	A data frame, matrix or vector containing the input values of the training data. For matrices and data frames, use one column per input variable.
outputs	A data frame, matrix or vector containing the output values of the training data. For multivariate output, use one column per output variable.
prior.mean	Either an object of class 'formula', that specifies the regressors in the prior mean function, or one of the two strings 'linear' or 'constant', that specify a linear prior mean in each input and a constant prior mean respectively. See details If method is LMC and you want to specify a formula.
cor.function	Specifies a correlation function used as part of the prior information for the emulator. This package has options of: corGaussian, corMatern2.5, corGaussianPeriodic, corGaussianPeriodic2 and corCombined. One can also specify a user defined function.
method	Can take string 'sep' or 'LMC'
...	additional arguments to be used for fitEmulator and passed on to correlation functions.
phi.opt	An optional vector that indicates the optimal phi values used to fit the emulator. It avoids running the optimiser to generate optimal phi values (in this case, the phi.init values are ignored). If not given phi.init values are optimised.
sigmasq.opt	An optional scalar that indicates the optimal sigmasq value used to fit the emulator. It avoids running the optimiser to generate optimal sigmasq value (in this case, the sigmasq.init values are ignored). If not given sigmasq.init values are optimised.
N.iterations	An optional value that indicates how many iterations to run the MCMC for. A non-positive value indicates not to apply the MCMC to estimate the starting values for the optimisation. In this case it will use the values given by phi.init as the starting values for the optimisation. Note: MCMC is run in order to get better starting values for the optim function that evaluates an optimum value for the hyperparameters,
phi.init	A optional vector used as starting values for either the optimisation or the MCMC output depending on N.iterations given. If provided, it should be a vector of length the same as the number of inputs. The default value is a vector of zeros, however when method is LMC, the default value is derived by fitting individual optimized estimates from the univariate emulator.
sigmasq.init	A optional scalar used as a starting value for either the optimisation or the MCMC output depending on N.iterations given. If provided, it should be a vector of length the same as the number of inputs. The default value is a vector of zeros, however when method is LMC, the default value is derived by fitting a multivariate separate emulator and obtaining the estimates of the optimised sigmasq.hat values
MC.plot	If TRUE, produces a trace plot of the MCMC output of log likelihood against the number of iterations. (default = TRUE)
param	the type parametrization chosen to appopriately parametrize the covariance matrix. param can take one of the following values: "original", "cholesky", "log-cholesky", "spherical", "matrix-log" (defult).
nugget	For noisy data, a scalar or a vector giving the observation variance for each training data point. If scalar, nugget for each training data point will be set to the same scalar value as specified. Currently only works with univariate output.
n.outputs	Number of outputs

Details

This function implements emulator fitting based on the procedures described in http://mucm.aston.ac.uk/MUCM/MUCMToolkit. For the prior.mean arguement, if method is LMC and a formula is specified, it should have the following format. (The actual equations can vary). Note: the vertical lines separate prior mean formula for each output, for instance in the example below (x1 + x2) correspond to y1. Given 3 output variables (y1, y2 and y3) and 3 input variables (x1, x2 and x3): y1 | y2 | y3 ~ x1 + x2 | x1 + x3 | x2 + x3. This format allows the user to specify different prior mean formulas for each output.

Value

fitEmulator returns an object of class 'emulatorFit'. An object of class 'emulatorFit' is a list containing at least the following components:

betahat	Posterior mean of β.
sigmasq.hat	Posterior mean of σ^2.
out.min.Hbeta	$(y - H\beta)$.
Ainv.H	$A^{-1}H$.
phi.hat	Optimum maximum likelihood estimate of the correlation length parameter
training.inputs	Inputs used to train the emulator.
training.outputs	Outputs used to train the emulator.
H.training	Part of the prior.mean.
tL	The cholesky decomposition of matrix A.
K	The cholesky decomposition of the inverse of tL multiplied with H.
Ainv.e	$A^{-1}(y - H\beta)$.
n.train	Number of training runs (rows of D).
n.regressors	Number of columns of H.
n.outputs	Number of outputs.
cor.function	The correlation function chosen for the prior variance.
cor.functions.args	Extra arguments used for cor.functions and passed on for further evaluation.
formula	The regressor terms in the prior mean function data.
opt.convergence	Only given if the optimisation was run. It is An integer code where 0 indicates successful comple
log.lik	Only given if the optimisation was run. It gives the log-likelihood value.
opt.message	Only given if the optimisation was run. A character string giving any additional information retu

See Also

If all data is in single matrix, see fitEmulatorData for a more conveniant method.

Examples

```
fit1 <- fitEmulator(inputs = surfebm[1:25,1:2], outputs = surfebm[1:25,3])

fit2 <- fitEmulator(inputs = surfebm[1:25,1:2], outputs = surfebm[1:25,3],
                    prior.mean = 'constant', cor.function = corMatern2.5,
                    N.iterations = 1000)
```

---

fitEmulatorData *Fit Emulator (Data)*

Description

A function equivalent to `fitEmulator`. In `fitEmulatorData` the user can provide the inputs and outputs in one matrix/dataframe. To provide them seperately, use `fitEmulator`

Usage

```
fitEmulatorData(formula, data, input.names = NULL, ...)
```

Arguments

`formula`	Formula to include the RHS and LHS. The LHS indicates the outputs and the RHS defines the prior mean function defined by the column names of `data`. A multivariate formula can also be given (see `as.Formula`) .
`data`	A data frame or matrix containing at least the inputs and outputs of the model.
`input.names`	an optional vector of character containing the names of variables in data to be considered as input variables. By default, all variables in the data matrix, except those on the left hand side of the formula, are treated as inputs.
`...`	additional arguments to be passed to the more specific `fitEmulator` function. Arguments such as `N.iterations`, `phi.opt`, `MC.plot`, `x.scale`, etc.

Value

`fitEmulatorData` returns an object of class 'emulatorFit'. See `fitEmulator` for details.

See Also

If input and output data is in different objects, see `fitEmulator` as a more conveniant method.

Examples

```
# specify the prior mean function as a object of class 'formula'
# ensure colnames of datasets include at least these variables.
emulator.prior.mean.formula <- Y ~ X.1 + X.2

# fit the emulator using a prior mean specified above and a Gaussian correlation function.
fit <- fitEmulatorData(formula = emulator.prior.mean.formula, data = surfebm[1:25, ],
                       cor.function = corGaussian)

# Use fitted emulator to predict posterior means and variances at the new inputs
predictions <- predict(fit, surfebm[26:35, 1:2], sd = FALSE, var.cov = TRUE)

# view results
print(predictions$posterior.mean)
```

GPSample *GP Sample*

Description

Function to sample from posterior Gaussian process

Usage

```
GPSample(n, mean, var)
```

Arguments

n	the number of samples required
mean	a vector giving the means of the variables
var	a positive-definite symmetric matrix specifying the covariance matrix of the variables

Author(s)

Jeremy Oakley.

makeEmulatorConstants *Make Emulator Constants*

Description

Function for collating and calculating constants needed for fitting the emulator

Usage

```
makeEmulatorConstants(phi, inputs, H, outputs, cor.function, nugget = NULL,
  sigmasq.hat = NULL, ...)

makeEmulatorConstantsLMC(phi, sigma, inputs, H, outputs, cor.function,
  n.outputs, ...)
```

Arguments

phi	Optimum Maximum likelihood estimate of the correlation length parameter
inputs	A data frame, matrix or vector containing the input values of the training data.
H	A matrix of prior mean regressors for the training data
outputs	A data frame, matrix or vector containing the output values of the training data.In makeEmulatorConstantLMC the outputs should be stacked (either a vector or a matrix with 1 column).
cor.function	Specifies a correlation function used as part of the prior information for the emulator. This package has options of: corGaussian, corMatern2.5, corGaussianPeriodic, corGaussianPeriodic2 and corCombined. One can also specify a user defined function.

nugget	For noisy data, a vector giving the observation variance for each training data point.
sigmasq.hat	Optimum Maximum likelihood estimate of the unknown scale parameter.
...	additional arguments to be passed on to correlation functions (see corGaussian)
sigma	Optimum Maximum likelihood estimate of the between outputs variance covariance matrix.

Value

The function returns a list containting some components as the emulatorFit class. See fitEmulator for details

Author(s)

Originally written by Jeremy Oakley. Modified by Sajni Malde

References

J. Oakley 1999. 'Bayesian uncertainty analysis for complex computer codes', PhD thesis, University of Sheffield.

MCMCMetropolisGibbs *MCMC using Metroplis Hastings within Gibbs*

Description

Generate a Markov Chain of the parameters in the correlation function Using Metropolis-Hastings within Gibbs

Usage

```
MCMCMetropolisGibbs(inputs, outputs, fn, H, N.iterations, starting.values,
  proposal.sd = 0.1, cor.function, MC.plot = TRUE, ...)
```

Arguments

inputs	A data frame, matrix or vector containing the input values of the training data.
outputs	A data frame, matrix or vector containing the output values of the training data.
fn	A function used to maximise the negetive log likelihood
H	A matrix of prior mean regressors from the training data
N.iterations	The number of iterations that MCMC should be run for
starting.values	the starting values for which the MCMC can start running
proposal.sd	is the standard deviation of the random walk proposal (default 0.1)
cor.function	Specifies a correlation function used as part of the prior information for the emulator. This package has options of: corGaussian, corMatern2.5, corGaussianPeriodic, corGaussianPeriodic2 and corCombined. One can also specify a user defined function.
MC.plot	If TRUE, produces a trace plot of the MCMC output of log likelihood against the number of iterations. (default=TRUE)
...	additional arguments to be passed on to correlation functions (see corGaussian)

Value

The function returns a list containting the following components:

density.sample	The negetive log likelihood of the MCMC output
theta.sample	A matrix of the theta sample per iteration

Author(s)

Originally written by Jeremy Oakley. Modified by Sajni Malde

---

negLogLik *Negetive Log Likelihood*

Description

These functions are minimised to find the optimal value of theta. negLogLikNugget is used in the univariate case for uncertain input parameters. negLogLikLMCOptim is used when applying the Non-Separable Emulator for Multivariate models

Usage

```
negLogLik(theta, nugget = NULL, inputs, H, outputs, cor.function, ...)

negLogLikNugget(theta, nugget, inputs, H, outputs, cor.function, ...)

negLogLikLMCOptim(theta, inputs, H, outputs, cor.function, n.outputs, param,
  ...)
```

Arguments

theta	Initial values for the parameters to be optimized over. (includes values for all the parameters that need to be estimated.)
nugget	For noisy data, a vector giving the observation variance for each training data point.
inputs	A data frame, matrix or vector containing the input values of the training data.
H	A matrix of prior mean regressors for the training data.
outputs	A data frame, matrix or vector containing the output values of the training data. In negLogLikLMCOptim, the outputs should be stacked (either a vector or a matrix with 1 column).
cor.function	Specifies a correlation function used as part of the prior information for the emulator
...	additional arguments to be passed on to correlation functions (see corGaussian)
n.outputs	Number of outputs
param	the type parametrization chosen to appopriately parametrize the covariance matrix.

Value

The function returns the negetive log-likelihood of `theta`

Author(s)

Sajni Malde, Jeremy Oakley

See Also

`corGaussian`

`predict.emulatorFit` *Predicts value and confidence interval at new inputs using Gaussian Process Emulation*

Description

Calculates the posterior distribution of the new inputs using methods described by Oakley. This function should be preceded by the `fitEmulator` function.

Usage

```
## S3 method for class 'emulatorFit'
predict(object, newdata, var.cov = FALSE, sd = TRUE,
  ...)
```

Arguments

`object`	of class inheriting from 'emulatorFit'
`newdata`	A data matrix of input(s) at which emulation is desired (new inputs). Must contain atleast all parameteres given in object$training.inputs.
`var.cov`	Optionally calculates posterior variance covariance matrix. Default is set to FALSE.
`sd`	Optionally calculates only the posterior standard deviation. Default is set to TRUE.
`...`	Further arguments not used - Error thrown

Details

Note that when using the LMC method, calculating the posterior variance is quite time-consuming.

Value

The function returns a list containting the following components:

`posterior.mean`	Approximation of the outputs for the given inputs in `newdata`
`posterior.variance`	Variance covariance matrix around this approximation
`standard.deviation`	Standard Deviation of the approximation. It equals the sqare root of the diagonal of the `posteri`

When the number of outputs to emulate is more than 1, and object is of class `"emulatorFit"`

(method = SEP), two extra values are returned from this function. These are

correlation.Matrix	A spatial correlation matrix
sigmahat	A between outputs covariance matrix

Author(s)

Originally written by Jeremy Oakley. Modified by Sajni Malde

References

Oakley, J. (1999). Bayesian uncertainty analysis for complex computer codes, Ph.D. thesis, University of Sheffield.

---

surfebm *surfebm*

Description

This is an example dataset used in package.

Usage

```
surfebm
```

Format

data.frame of 35 observations for 3 variables.

Author(s)

Jeremy Oakley

---

updateFit *Update fit*

Description

Updates old "emulatorFit" objects to work with latest code to allow for improvements to be made within the code.

Usage

```
updateFit(object)
```

Arguments

object	An "emulatorFit" object to update.

Value

Returns the updated "emulatorFit" object.

Note

A warning is issued if the object is updated.

---

validateEmulator *Validate the Emulator*

Description

Uses diagnostics described in Bastos and O'Hagan (2009) to validate the emulator.

Usage

```
validateEmulator(emulator, new.outputs, emulator.predictions,
  which.output = 1, plot = TRUE)
```

Arguments

emulator	Object of class emulatorFit generated by fitEmulator or fitEmulatorData
new.outputs	A matrix outputs used to validate the emulator predictions. This Matrix must have columns equal to the number of outputs.
emulator.predictions	Output generated from the predict.emulatorFit function.
which.output	Defaults to 1. Is an indicative integer that validates the output stated based on column number from the left.
plot	Optionally plot validation plots. Default is set to true.

Details

validateEmulator returns a grid of 4 plots and prints out a value. These are described below:

Top Left plot	Plots validation output against the posterior mean generated using the emulator. The error bars are the
Top Right plot	Shows the QQ plot of cholesky residuals. If the points lie close to the 45 degree line then the normalit
Bottom Left plot	Plot of Pivoted cholesky prediciton errors against the pivoting index. The pivoting index gives the ord
Bottom right Plot	The red line on the plot is the value of the calculated Mahalanobis distance taking into account correla

Value

The class of 'validateEmulator' is a list containting atleast the following components:

coverage	Proportion of validation outputs in 95 percent confidence intervals of the emulator predicted output
RMSE	The root mean square error of the predicted values
normRMSE	The normalised value of RMSE over the range of the data

Author(s)

Originally written by Jeremy Oakley. Modified by Sajni Malde

References

Bastos, L. S. and O'Hagan, A. (2009). Diagnostics for gaussian process emulators, Technometrics, 51 (4): 425-438.

See Also

`predict.emulatorFit` and `fitEmulator`

Index