

Package ‘CGP’

July 21, 2025

Type Package

Title Composite Gaussian Process Models

Version 2.1-1

Date 2018-06-11

Author Shan Ba and V. Roshan Joseph

Maintainer Shan Ba <shanbatr@gmail.com>

Description Fit composite Gaussian process (CGP) models as described in Ba and Joseph (2012) ``Composite Gaussian Process Models for Emulating Expensive Functions'', Annals of Applied Statistics. The CGP model is capable of approximating complex surfaces that are not second-order stationary. Important functions in this package are CGP, print.CGP, summary.CGP, predict.CGP and plotCGP.

License LGPL-2.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-12 15:08:19 UTC

Contents

CGP-package	2
CGP	3
plotCGP	5
predict.CGP	6
print.CGP	8
summary.CGP	9

Index	11
--------------	-----------

CGP-package

*The composite Gaussian process model package***Description**

Build nonstationary surrogate models for emulating computationally-expensive computer simulations (computer models).

Details

Package: CGP
 Type: Package
 Version: 2.1-1
 Date: 2018-06-11
 License: LGPL-2.1

This package contains functions for fitting the composite Gaussian process (CGP) model, which consists of two Gaussian processes (GPs). The first GP captures the smooth global trend and the second one models local details. The model also incorporates a flexible variance model, which makes it more capable of approximating surfaces with varying volatility. It can be used as an emulator (surrogate model) for approximating computationally expensive functions that are not second-order stationary. When the underlying surface is stationary, the fitted CGP model should degenerate to a standard (stationary) GP model ($\hat{\lambda} \approx 0$).

The package implements maximum likelihood method to estimate model parameters and also provides predictions (with 5% and 95% prediction quantiles) for unobserved input locations. Leave-one-out cross validation diagnostic plot is also supported.

Gaussian correlation functions

$$g(\mathbf{h}) = \exp\left(-\sum_{j=1}^p \theta_j h_j^2\right), \quad l(\mathbf{h}) = \exp\left(-\sum_{j=1}^p \alpha_j h_j^2\right)$$

(with unknown parameters θ and α) are used to describe the correlations in the global and local processes respectively.

For a complete list of functions, please use `help(package="CGP")`. Important functions are [CGP](#), [print.CGP](#), [summary.CGP](#), [predict.CGP](#) and [plotCGP](#).

Author(s)

Shan Ba and V. Roshan Joseph
 Maintainer: Shan Ba <shanbatr@gmail.com>

References

Ba, S. and V. Roshan Joseph (2012). Composite Gaussian Process Models for Emulating Expensive Functions. *Annals of Applied Statistics*, 6, 1838-1860.

Description

Estimate parameters in the composite Gaussian process (CGP) model using maximum likelihood methods. Calculate the root mean squared (leave-one-out) cross validation error for diagnosis, and export intermediate values to facilitate predict.CGP function.

Usage

```
CGP(X, yobs, nugget_l = 0.001, num_starts = 5,
    theta_l = NULL, alpha_l = NULL, kappa_u = NULL)
```

Arguments

X	The design matrix
yobs	The vector of response values, corresponding to the rows of X
nugget_l	Optional, default is “0.001”. The lower bound for the nugget value (λ in the paper)
num_starts	Optional, default is “5”. Number of random starts for optimizing the likelihood function
theta_l	Optional, default is “0.0001”. The lower bound for all correlation parameters in the global GP (θ in the paper)
alpha_l	Optional. The lower bound for all correlation parameters in the local GP (α in the paper). It is also the upper bound for all correlation parameters in the global GP (the θ). Default is $\log(100) * \text{mean}(1/\text{dist}(\text{Stand_X})^2)$, where $\text{Stand_X} \leftarrow \text{apply}(X, 2, \text{function}(x) \ (x - \min(x))/\max(x - \min(x)))$. Please refer to Ba and Joseph (2012) for details
kappa_u	Optional. The upper bound for κ , where we define $\alpha_j = \theta_j + \kappa$ for $j = 1, \dots, p$. Default value is $\log(10^6) * \text{mean}(1/\text{dist}(\text{Stand_X})^2)$, \ where $\text{Stand_X} \leftarrow \text{apply}(X, 2, \text{function}(x) \ (x - \min(x))/\max(x - \min(x)))$

Details

This function fits a composite Gaussian process (CGP) model based on the given design matrix X and the observed responses yobs. The fitted model consists of a smooth GP to capture the global trend and a local GP which is augmented with a flexible variance model to capture the change of local volatilities. For p input variables, such two GPs involve $2p + 2$ unknown parameters in total. As demonstrated in Ba and Joseph (2012), by assuming $\alpha_j = \theta_j + \kappa$ for $j = 1, \dots, p$, fitting the CGP model only requires estimating $p + 3$ unknown parameters, which is comparable to fitting a stationary GP model (p unknown parameters).

Value

This function fits the CGP model and returns an object of class “CGP”. Function `predict.CGP` can be further used for making new predictions and function `summary.CGP` can be used to print a summary of the “CGP” object.

An object of class “CGP” is a list containing at least the following components:

<code>lambda</code>	Estimated nugget value (λ)
<code>theta</code>	Vector of estimated correlation parameters (θ) in the global GP
<code>alpha</code>	Vector of estimated correlation parameters (α) in the local GP
<code>bandwidth</code>	Estimated bandwidth parameter (b) in the variance model
<code>rmscv</code>	Root mean squared (leave-one-out) cross validation error
<code>Yp_jackknife</code>	Vector of Jackknife (leave-one-out) predicted values
<code>mu</code>	Estimated mean value (μ) for global trend
<code>tau2</code>	Estimated variance (τ^2) for global trend
<code>beststart</code>	Best starting value found for optimizing the log-likelihood
<code>objval</code>	Optimal objective value for the negative log-likelihood (up to a constant)
<code>var_names</code>	Vector of input variable names
<code>Sig_matrix</code>	Diagonal matrix containing standardized local variances at each of the design points
<code>sf</code>	Standardization constant for rescaling the local variance model
<code>res2</code>	Vector of squared residuals from the estimated global trend
<code>invQ</code>	Matrix of $(\mathbf{G} + \lambda \mathbf{\Sigma}^{1/2} \mathbf{L} \mathbf{\Sigma}^{1/2})^{-1}$
<code>temp_matrix</code>	Matrix of $(\mathbf{G} + \lambda \mathbf{\Sigma}^{1/2} \mathbf{L} \mathbf{\Sigma}^{1/2})^{-1}(\mathbf{y} - \hat{\mu} \mathbf{1})$

Author(s)

Shan Ba <shanbatr@gmail.com> and V. Roshan Joseph <roshan@isye.gatech.edu>

References

Ba, S. and V. Roshan Joseph (2012) “Composite Gaussian Process Models for Emulating Expensive Functions”. *Annals of Applied Statistics*, 6, 1838-1860.

See Also

[predict.CGP](#), [print.CGP](#), [summary.CGP](#)

Examples

```
x1<-c(0,.02,.075,.08,.14,.15,.155,.156,.18,.22,.29,.32,.36,
      .37,.42,.5,.57,.63,.72,.785,.8,.84,.925,1)
x2<-c(.29,.02,.12,.58,.38,.87,.01,.12,.22,.08,.34,.185,.64,
      .02,.93,.15,.42,.71,1,0,.21,.5,.785,.21)
X<-cbind(x1,x2)
yobs<-sin(1/((x1*0.7+0.3)*(x2*0.7+0.3)))
```

```
## Not run:
#Fit the CGP model
#Increase the lower bound for nugget to 0.01 (Optional)
mod<-CGP(X,yobs,nugget_l=0.01)
summary(mod)

mod$objval
#-27.4537
mod$lambda
#0.6210284
mod$theta
#6.065497 8.093402
mod$alpha
#143.1770 145.2049
mod$bandwidth
#1
mod$rmse
#0.5714969

## End(Not run)
```

plotCGP

Jackknife (leave-one-out) actual by predicted diagnostic plot

Description

Draw jackknife (leave-one-out) actual by predicted plot to measure goodness-of-fit.

Usage

```
plotCGP(object)
```

Arguments

object An object of class "CGP"

Details

Draw the actual observed values on the y-axis and the jackknife (leave-one-out) predicted values on the x-axis. The goodness-of-fit can be measured by how well the points lie along the 45 degree diagonal line.

Value

This function draws the jackknife (leave-one-out) actual by predicted plot.

Author(s)

Shan Ba <shanbatr@gmail.com> and V. Roshan Joseph <roshan@isye.gatech.edu>

References

Ba, S. and V. Roshan Joseph (2012) “Composite Gaussian Process Models for Emulating Expensive Functions”. *Annals of Applied Statistics*, 6, 1838-1860.

See Also

[CGP](#)

Examples

```
x1<-c(0,.02,.075,.08,.14,.15,.155,.156,.18,.22,.29,.32,.36,
.37,.42,.5,.57,.63,.72,.785,.8,.84,.925,1)
x2<-c(.29,.02,.12,.58,.38,.87,.01,.12,.22,.08,.34,.185,.64,
.02,.93,.15,.42,.71,1,0,.21,.5,.785,.21)
X<-cbind(x1,x2)
yobs<-x1^2+x2^2
## Not run:
#The CGP model
mod<-CGP(X,yobs,nugget_l=0.001)
plotCGP(mod)

## End(Not run)
```

predict.CGP	<i>Predict from the composite Gaussian process model</i>
-------------	--

Description

Compute predictions from the composite Gaussian process (CGP) model. 95% prediction intervals can also be calculated.

Usage

```
## S3 method for class 'CGP'
predict(object, newdata = NULL, PI = FALSE, ...)
```

Arguments

object	An object of class "CGP"
newdata	Optional. The matrix of predictive input locations, where each row of newdata corresponds to one predictive location
PI	If TRUE, 95% prediction intervals are also calculated. Default is FALSE
...	For compatibility with generic method predict

Details

Given an object of “CGP” class, this function predicts responses at unobserved newdata locations. If the PI is set to be TRUE, 95% predictions intervals are also computed.

If newdata is equal to the design matrix of the object, predictions from the CGP model will be identical to the yobs component of the object and the width of the prediction intervals will be shrunk to zero. This is due to the interpolating property of the predictor.

Value

The function returns a list containing the following components:

Yp	Vector of predictive values at newdata locations ($Y_p = g_p + l_p$)
gp	Vector of predictive values at newdata locations from the global process
lp	Vector of predictive values at newdata locations from the local process
v	Vector of predictive standardized local volatilities at newdata locations
Y_low	If PI=TRUE, vector of 5% predictive quantiles at newdata locations
Y_up	If PI=TRUE, vector of 95% predictive quantiles at newdata locations

Author(s)

Shan Ba <shanbatr@gmail.com> and V. Roshan Joseph <roshan@isye.gatech.edu>

References

Ba, S. and V. Roshan Joseph (2012) “Composite Gaussian Process Models for Emulating Expensive Functions”. *Annals of Applied Statistics*, 6, 1838-1860.

See Also

[CGP](#), [print.CGP](#), [summary.CGP](#)

Examples

```
### A simulated example from Gramacy and Lee (2012) ``Cases for the nugget
### in modeling computer experiments''. \emph{Statistics and Computing}, 22, 713-722.

#Training data
X<-c(0.775,0.83,0.85,1.05,1.272,1.335,1.365,1.45,1.639,1.675,
1.88,1.975,2.06,2.09,2.18,2.27,2.3,2.36,2.38,2.39)
yobs<-sin(10*pi*X)/(2*X)+(X-1)^4

#Testing data
UU<-seq(from=0.7,to=2.4,by=0.001)
y_true<-sin(10*pi*UU)/(2*UU)+(UU-1)^4

plot(UU,y_true,type="l",xlab="x",ylab="y")
points(X,yobs,col="red")
## Not run:
#Fit the CGP model
```

```

mod<-CGP(X,yobs)
summary(mod)

mod$objval
#-40.17315
mod$lambda
#0.01877432
mod$theta
#2.43932
mod$alpha
#578.0898
mod$bandwidth
#1
mod$rmse
#0.3035192

#Predict for the testing data 'UU'
modpred<-predict(mod,UU)

#Plot the fitted CGP model
#Red: final predictor; Blue: global trend
lines(UU,modpred$Yp,col="red",lty=3,lwd=2)
lines(UU,modpred$gp,col="blue",lty=5,lwd=1.8)

## End(Not run)

```

print.CGP

CGP model summary information

Description

Print a brief summary of a “CGP” object.

Usage

```

## S3 method for class 'CGP'
print(x, ...)

```

Arguments

x	An object of class "CGP"
...	For compatibility with generic method print

Details

This function prints a brief summary of a “CGP” object.

Value

This function prints the results of:

lambda	Estimated nugget value (λ)
theta	Estimated correlation parameters (θ) in the global GP
alpha	Estimated correlation parameters (α) in the local GP
bandwidth	Estimated bandwidth parameter (b) in the variance model

Author(s)

Shan Ba <shanbatr@gmail.com> and V. Roshan Joseph <roshan@isye.gatech.edu>

References

Ba, S. and V. Roshan Joseph (2012) “Composite Gaussian Process Models for Emulating Expensive Functions”. *Annals of Applied Statistics*, 6, 1838-1860.

See Also

[CGP](#), [summary.CGP](#), [predict.CGP](#)

Examples

```
x1<-c(0,.02,.075,.08,.14,.15,.155,.156,.18,.22,.29,.32,.36,
      .37,.42,.5,.57,.63,.72,.785,.8,.84,.925,1)
x2<-c(.29,.02,.12,.58,.38,.87,.01,.12,.22,.08,.34,.185,.64,
      .02,.93,.15,.42,.71,1,0,.21,.5,.785,.21)
X<-cbind(x1,x2)
yobs<-sin(1/((x1*0.7+0.3)*(x2*0.7+0.3)))
## Not run:
#Fit the CGP model
mod<-CGP(X,yobs)
print(mod)

## End(Not run)
```

summary.CGP

CGP model summary information

Description

Print a summary of a “CGP” object.

Usage

```
## S3 method for class 'CGP'
summary(object, ...)
```

Arguments

object	An object of class "CGP"
...	For compatibility with generic method summary

Details

This function prints a summary of a “CGP” object.

Value

This function prints the results of:

lambda	Estimated nugget value (λ)
theta	Estimated correlation parameters (θ) in the global GP
alpha	Estimated correlation parameters (α) in the local GP
bandwidth	Estimated bandwidth parameter (b) in the variance model
rmscv	Root mean squared (leave-one-out) cross validation error
mu	Estimated mean value (μ) for global trend
tau2	Estimated variance (τ^2) for global trend
beststart	Best starting value found for optimizing the log-likelihood
objval	Optimal objective value for the negative log-likelihood (up to a constant)

Author(s)

Shan Ba <shanbatr@gmail.com> and V. Roshan Joseph <roshan@isye.gatech.edu>

References

Ba, S. and V. Roshan Joseph (2012) “Composite Gaussian Process Models for Emulating Expensive Functions”. *Annals of Applied Statistics*, 6, 1838-1860.

See Also

[CGP](#), [print.CGP](#), [predict.CGP](#)

Examples

```
x1<-c(0,.02,.075,.08,.14,.15,.155,.156,.18,.22,.29,.32,.36,
      .37,.42,.5,.57,.63,.72,.785,.8,.84,.925,1)
x2<-c(.29,.02,.12,.58,.38,.87,.01,.12,.22,.08,.34,.185,.64,
      .02,.93,.15,.42,.71,1,0,.21,.5,.785,.21)
X<-cbind(x1,x2)
yobs<-sin(1/((x1*0.7+0.3)*(x2*0.7+0.3)))
## Not run:
#Fit the CGP model
mod<-CGP(X,yobs)
summary(mod)

## End(Not run)
```

Index

*** package**

CGP-package, [2](#)

CGP, [2](#), [3](#), [6](#), [7](#), [9](#), [10](#)

CGP-package, [2](#)

plotCGP, [2](#), [5](#)

predict.CGP, [2](#), [4](#), [6](#), [9](#), [10](#)

print.CGP, [2](#), [4](#), [7](#), [8](#), [10](#)

summary.CGP, [2](#), [4](#), [7](#), [9](#), [9](#)