Package 'rkriging'

July 19, 2024

Type Package

Title Kriging Modeling

Version 1.0.1

Description
An 'Eigen'-based computationally efficient 'C++' implementation for fitting various kriging models to data. This research is supported by U.S. National Science Foundation grant DMS-2310637.

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports Rcpp, nloptr (>= 1.2.0), methods, stats

LinkingTo Rcpp, RcppEigen (>= 0.3.3.5.0), BH (>= 1.75.0.0), nloptr (>= 1.2.0)

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation yes

Author Chaofan Huang [aut, cre],
V. Roshan Joseph [aut]

Maintainer Chaofan Huang <10billhuang01@gmail.com>

Repository CRAN

Date/Publication 2024-07-19 10:10:03 UTC

Contents

`Evaluate.Kernel` *Evaluate Kernel*

Description

This function computes the kernel (correlation) matrix. Given the kernel class object and the input data X of size n, this function computes the corresponding $n \times n$ kernel (correlation) matrix.

Usage

```
Evaluate.Kernel(kernel, X)
```

Arguments

`kernel`	a kernel class object
`X`	input data

Value

The kernel (correlation) matrix of X evaluated by the kernel.

Author(s)

Chaofan Huang and V. Roshan Joseph

See Also

Get.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

kernel <- Gaussian.Kernel(lengthscale)
Evaluate.Kernel(kernel, X)
```

`Fit.Kriging` *Fit or Create Kriging Models*

Description

This function provides a common interface to fit various kriging models from the data. OK (Ordinary.Kriging), UK (Universal.Kriging), LK (Limit.Kriging), RK (Rational.Kriging), and GRK (Generalized.Rational.Kriging) are supported in this package.

Usage

```
Fit.Kriging(
  X,
  y,
  interpolation = TRUE,
  fit = TRUE,
  model = "OK",
  model.parameters = list(),
  kernel = NULL,
  kernel.parameters = list(),
  nlopt.parameters = list()
)
```

Arguments

`X`	a matrix for input (feature)
`y`	a vector for output (target), only one-dimensional output is supported
`interpolation`	whether to interpolate, for noisy data please set `interpolate=FALSE`
`fit`	whether to fit the length scale parameters from data
`model`	choice of kriging model: OK, UK, LK, RK, GRK
`model.parameters`	a list of parameters required for the specific kriging model, e.g. basis functions for universal kriging
`kernel`	a kernel class object
`kernel.parameters`	a list of parameters required for the kernel, if no kernel class object is provided

`nlopt.parameters`
: a list of parameters required for NLopt, including choice of optimization algorithm and maximum number of evaluation

Details

Kriging gives the best linear unbiased predictor given the data. It can also be given a Bayesian interpretation by assuming a Gaussian process (GP) prior for the underlying function. Please see Santner et al. (2003), Rasmussen and Williams (2006), and Joseph (2024) for details.

For data from deterministic computer experiments, use `interpolation=TRUE` and will give an interpolator. For noisy data, use `interpolation=FALSE`, which will give an approximator of the underlying function. Currently, approximation is supported for OK (Ordinary.Kriging) and UK (Universal.Kriging).

The kernel choices are required and can be specified by (i) providing the kernel class object to `kernel` or (ii) specifying the kernel type and other parameters in `kernel.parameters`. Please see examples section for detail usages.

When the lengthscale / correlation parameters are known, simply provide them in (i) the kernel class object to `kernel` or (ii) in `kernel.parameters`, and set `fit=FALSE`. The kriging model will be fitted with the user provided parameters.

When the lengthscale / correlation parameters are unknown, they can be estimated via Maximum Likelihood method by setting `fit=TRUE`. The initial / lower bound / upper bound of the lengthscale parameters can be provided in `kernel.parameters`, otherwise a good initial and range would be estimated from the data. The optimization is performed via NLopt, a open-source library for nonlinear optimization. All gradient-free optimization methods in NLopt are supported and can be specified in `nlopt.parameters`. See `nloptr::nloptr.print.options()` for the list of available derivative-free algorithms (prefix with NLOPT_GN or NLOPT_LN). The maximum number of optimization steps can also be defined in `nlopt.parameters`. Please see examples section for detail usages.

Value

A Kriging Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. (2006). *Limit kriging*. Technometrics, 48(4), 458-466.

Joseph, V. R. (2024). Rational Kriging. *Journal of the American Statistical Association*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Santner, T. J., Williams, B. J., Notz, W. I., & Williams, B. J. (2003). *The design and analysis of computer experiments (Vol. 1)*. New York: Springer.

See Also

Predict.Kriging, Get.Kriging.Parameters, Get.Kernel, Ordinary.Kriging, Universal.Kriging, Limit.Kriging, Rational.Kriging, Generalized.Rational.Kriging.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

##############################################################################
############### Minimal Example for Fitting a Kriging Model ################
##############################################################################
# Ordinary Kriging
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# Universal Kriging
basis.function <- function(x) {c(1,x[1],x[1]^2)}
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="UK",
                       model.parameters=list(basis.function=basis.function),
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# Limit Kriging
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="LK",
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
```

```
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# Rational Kriging
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="RK",
                       kernel.parameters=list(type="RQ",alpha=1))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# Generalized Rational Kriging
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="GRK",
                       kernel.parameters=list(type="RQ",alpha=1))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

##############################################################################
################ Fitting a Kriging Model with Kernel Object #################
##############################################################################
kernel <- Gaussian.Kernel(rep(1,p))
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK", kernel=kernel)
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

##############################################################################
################ Creating a Kriging Model with Kernel Object #################
##############################################################################
# set fit = FALSE to create Kriging model with user provided kernel
# no optimization for the length scale parameters
kernel <- Gaussian.Kernel(rep(1e-1,p))
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=FALSE, model="OK", kernel=kernel)
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
```

```
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

################################################################################
############ Fitting a Kriging Model with Range for Lengthscale #############
################################################################################
kernel.parameters <- list(
    type = 'Gaussian',
    lengthscale = rep(1,p), # initial value
    lengthscale.lower.bound = rep(1e-3,p), # lower bound
    lengthscale.upper.bound = rep(1e3,p) # upper bound
) # if not provided, a good estimate would be computed from data
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=kernel.parameters)
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

################################################################################
########### Fitting a Kriging Model with Different Optimization #############
################################################################################
nlopt.parameters <- list(
    algorithm = 'NLOPT_GN_DIRECT', # optimization method
    maxeval = 250 # maximum number of evaluation
)
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"),
                       nlopt.parameters=nlopt.parameters)
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# Global-Local Optimization
nlopt.parameters <- list(
    algorithm = 'NLOPT_GN_MLSL_LDS', # optimization method
    local.algorithm = 'NLOPT_LN_SBPLX', # local algorithm
    maxeval = 250 # maximum number of evaluation
)
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"),
                       nlopt.parameters=nlopt.parameters)
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
```

```
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

#############################################################################
################ Fitting a Kriging Model from Noisy Data ####################
#############################################################################
y <- y + 0.1 * rnorm(length(y))
kriging <- Fit.Kriging(X, y, interpolation=FALSE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)
```

`Gaussian.Kernel` *Gaussian Kernel*

Description

This function specifies the Gaussian / Squared Exponential (SE) / Radial Basis Function (RBF) kernel.

Usage

```
Gaussian.Kernel(lengthscale)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters

Details

The Gaussian kernel is given by

$$k(r) = \exp(-r^2/2),$$

where

$$r(x, x') = \sqrt{\sum_{i=1}^{p} \left(\frac{x_i - x_i'}{l_i} \right)^2}$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's.

Value

A Gaussian Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- Gaussian.Kernel(lengthscale)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="Gaussian")
Evaluate.Kernel(kernel, X)
```

`Generalized.Rational.Kriging`
Generalized Rational Kriging

Description

This functions fits the generalized rational kriging model to the data.

Usage

```
Generalized.Rational.Kriging(
  X,
  y,
  fit = TRUE,
  kernel = NULL,
  kernel.parameters = list(),
  nlopt.parameters = list()
)
```

Arguments

`X`	a matrix for input (feature)
`y`	a vector for output (target), only one-dimensional output is supported
`fit`	whether to fit the length scale parameters from data
`kernel`	a kernel class object
`kernel.parameters`	a list of parameters required for the kernel, if no kernel class object is provided
`nlopt.parameters`	a list of parameters required for NLopt, including choice of optimization algorithm and maximum number of evaluation

Details

Ordinary kriging and rational kriging can be obtained as special cases of generalized rational kriging. Please see Joseph (2024) for details. The Spectra library is used for fast computation of the first eigenvalues/vectors. Only interpolation is available. Noisy output is not supported for generalized rational kriging.

The kernel choices are required and can be specified by (i) providing the kernel class object to `kernel` or (ii) specifying the kernel type and other parameters in `kernel.parameters`. Please see examples section of Fit.Kriging for detail usages.

When the lengthscale / correlation parameters are unknown, all parameters including the constant mean can be estimated via Maximum Likelihood method by setting `fit=TRUE`. The initial / lower bound / upper bound of the lengthscale parameters can be provided in `kernel.parameters`, otherwise a good initial and range would be estimated from the data. The optimization is performed via NLopt library, a open-source library for nonlinear optimization. All gradient-free optimization methods in NLopt are supported and can be specified in `nlopt.parameters`. See `nloptr::nloptr.print.options()` for the list of available derivative-free algorithms (prefix with NLOPT_GN or NLOPT_LN). The maximum number of optimization steps can also be defined in `nlopt.parameters`. Please see examples section of Fit.Kriging for detail usages.

Value

A Generalized Rational Kriging Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. (2024). Rational Kriging. *Journal of the American Statistical Association*.

Qiu, Y., Guennebaud, G., & Niesen, J. (2015). Spectra: C++ library for large scale eigenvalue problems.

See Also

Fit.Kriging, Predict.Kriging, Get.Kriging.Parameters.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

# approach 1
kriging <- Generalized.Rational.Kriging(X, y, fit=TRUE,
                                        kernel.parameters=list(type="RQ",alpha=1))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# approach 2
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="GRK",
                       kernel.parameters=list(type="RQ",alpha=1))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)
```

`Get.Kernel` *Kernel*

Description

This function provides a common interface to specify various kernels. See arguments section for the available kernels in this package.

Usage

```
Get.Kernel(lengthscale, type, parameters = list())
```

Arguments

lengthscale	a vector for the positive length scale parameters
type	kernel type: Gaussian, RQ, Matern12, Matern32, Matern52, Matern, UDF, MultiplicativeRQ, MultiplicativeMatern, MultiplicativeUDF
parameters	a list of parameters required for the specific kernel

Value

A Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Evaluate.Kernel, Gaussian.Kernel, RQ.Kernel, Matern12.Kernel, Matern32.Kernel, Matern52.Kernel Matern.Kernel, UDF.Kernel, MultiplicativeRQ.Kernel, MultiplicativeMatern.Kernel, MultiplicativeUDF.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# Gaussian
kernel <- Get.Kernel(lengthscale, type="Gaussian")
Evaluate.Kernel(kernel, X)

# Rational Quadratic (RQ)
kernel <- Get.Kernel(lengthscale, type="RQ", parameters=list(alpha=1))
Evaluate.Kernel(kernel, X)

# Matern(1/2)
kernel <- Get.Kernel(lengthscale, type="Matern12")
Evaluate.Kernel(kernel, X)

# Matern(3/2)
kernel <- Get.Kernel(lengthscale, type="Matern32")
Evaluate.Kernel(kernel, X)

# Matern(5/2)
```

```
kernel <- Get.Kernel(lengthscale, type="Matern52")
Evaluate.Kernel(kernel, X)

# Generalized Matern
kernel <- Get.Kernel(lengthscale, type="Matern", parameters=list(nu=2.01))
Evaluate.Kernel(kernel, X)

# User Defined Function (UDF) Kernel
kernel.function <- function(sqdist) {return (exp(-sqrt(sqdist)))}
kernel <- Get.Kernel(lengthscale, type="UDF",
                     parameters=list(kernel.function=kernel.function))
Evaluate.Kernel(kernel, X)

# Multiplicative Rational Quadratic (RQ)
kernel <- Get.Kernel(lengthscale, type="MultiplicativeRQ", parameters=list(alpha=1))
Evaluate.Kernel(kernel, X)

# Multiplicative Generalized Matern
kernel <- Get.Kernel(lengthscale, type="MultiplicativeMatern", parameters=list(nu=2.01))
Evaluate.Kernel(kernel, X)

# Multiplicative User Defined Function (UDF)
kernel.function <- function(sqdist) {return (exp(-sqrt(sqdist)))}
kernel <- Get.Kernel(lengthscale, type="MultiplicativeUDF",
                     parameters=list(kernel.function=kernel.function))
Evaluate.Kernel(kernel, X)
```

`Get.Kriging.Parameters` *Get Kriging Parameters*

Description

This function can be used for extracting the estimates of the kriging parameters.

Usage

```
Get.Kriging.Parameters(kriging)
```

Arguments

`kriging`	a kriging class object

Value

`nllh`	negative log-likelihood of the kriging model
`mu`	mean of the kriging model
`nu2`	variance of the kriging model

`sigma2`	variance of the random noise when `interpolation=FALSE`
`beta`	coefficients of the basis functions for universal kriging
`c`	c for the rational / generalized rational kriging, see Joseph (2024)
`c0`	c0 for the generalized rational kriging, see Joseph (2024)

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. (2006). *Limit kriging*. Technometrics, 48(4), 458-466.

Joseph, V. R. (2024). Rational Kriging. *Journal of the American Statistical Association*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Santner, T. J., Williams, B. J., Notz, W. I., & Williams, B. J. (2003). *The design and analysis of computer experiments (Vol. 1)*. New York: Springer.

See Also

Fit.Kriging.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"))
Get.Kriging.Parameters(kriging)
```

`Limit.Kriging` *Limit Kriging*

Description

This functions fits the limit kriging model to the data.

Usage

```
Limit.Kriging(
  X,
  y,
  fit = TRUE,
  kernel = NULL,
  kernel.parameters = list(),
  nlopt.parameters = list()
)
```

Arguments

`X`	a matrix for input (feature)
`y`	a vector for output (target), only one-dimensional output is supported
`fit`	whether to fit the length scale parameters from data
`kernel`	a kernel class object
`kernel.parameters`	a list of parameters required for the kernel, if no kernel class object is provided
`nlopt.parameters`	a list of parameters required for NLopt, including choice of optimization algorithm and maximum number of evaluation

Details

Limit kriging avoids the mean reversion issue of ordinary kriging. Please see Joseph (2006) for details. Only interpolation is available. Noisy output is not supported for limit kriging.

The kernel choices are required and can be specified by (i) providing the kernel class object to `kernel` or (ii) specifying the kernel type and other parameters in `kernel.parameters`. Please see examples section of Fit.Kriging for detail usages.

When the lengthscale / correlation parameters are unknown, all parameters including the constant mean can be estimated via Maximum Likelihood method by setting `fit=TRUE`. The initial / lower bound / upper bound of the lengthscale parameters can be provided in `kernel.parameters`, otherwise a good initial and range would be estimated from the data. The optimization is performed via NLopt, a open-source library for nonlinear optimization. All gradient-free optimization methods in NLopt are supported and can be specified in `nlopt.parameters`. See `nloptr::nloptr.print.options()` for the list of available derivative-free algorithms (prefix with NLOPT_GN or NLOPT_LN). The maximum number of optimization steps can also be defined in `nlopt.parameters`. Please see examples section of Fit.Kriging for detail usages.

Value

A Limit Kriging Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. (2006). *Limit kriging*. Technometrics, 48(4), 458-466.

See Also

Fit.Kriging, Predict.Kriging, Get.Kriging.Parameters.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

# approach 1
kriging <- Limit.Kriging(X, y, fit=TRUE, kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# approach 2
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="LK",
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)
```

`Matern.Kernel` *Generalized Matern Kernel*

Description

This function specifies the (Generalized) Matern kernel with any smoothness parameter ν.

Usage

```
Matern.Kernel(lengthscale, nu = 2.01)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters
`nu`	a positive scalar parameter that controls the smoothness

Details

The Generalized Matern kernel is given by

$$k(r;\nu) = \frac{2^{1-\nu}}{\Gamma(\nu)}(\sqrt{2\nu}r)^{\nu}K_{\nu}(\sqrt{2\nu}r),$$

where ν is the smoothness parameter, K_{ν} is the modified Bessel function, Γ is the gamma function, and

$$r(x,x') = \sqrt{\sum_{i=1}^{p}\left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's. As $\nu \to \infty$, it converges to the Gaussian.Kernel.

Value

A Generalized Matern Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Matern12.Kernel, Matern32.Kernel, Matern52.Kernel, MultiplicativeMatern.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- Matern.Kernel(lengthscale, nu=2.01)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="Matern", parameters=list(nu=2.01))
Evaluate.Kernel(kernel, X)
```

Matern12.Kernel *Matern(1/2) Kernel*

Description

This function specifies the Matern kernel with smoothness parameter ν=1/2. It is also known as the Exponential kernel.

Usage

```
Matern12.Kernel(lengthscale)
```

Arguments

lengthscale	a vector for the positive length scale parameters

Details

The Matern(1/2) kernel is given by

$$k(r) = \exp(-r),$$

where

$$r(x, x') = \sqrt{\sum_{i=1}^{p} \left(\frac{x_i - x_i'}{l_i} \right)^2}$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's.

Value

A Matern(1/2) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Matern.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- Matern12.Kernel(lengthscale)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="Matern12")
Evaluate.Kernel(kernel, X)
```

Matern32.Kernel *Matern(3/2) Kernel*

Description

This function specifies the Matern kernel with smoothness parameter ν=3/2.

Usage

```
Matern32.Kernel(lengthscale)
```

Arguments

lengthscale	a vector for the positive length scale parameters

Details

The Matern(3/2) kernel is given by

$$k(r) = (1 + \sqrt{3}r)\exp(-\sqrt{3}r),$$

where

$$r(x, x') = \sqrt{\sum_{i=1}^{p} \left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's.

Value

A Matern(3/2) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Matern.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- Matern32.Kernel(lengthscale)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="Matern32")
Evaluate.Kernel(kernel, X)
```

`Matern52.Kernel` *Matern(5/2) Kernel*

Description

This function specifies the Matern kernel with smoothness parameter ν=5/2.

Usage

```
Matern52.Kernel(lengthscale)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters

Details

The Matern(5/2) kernel is given by

$$k(r) = (1 + \sqrt{5}r + 5r^2/3)\exp(-\sqrt{5}r),$$

where

$$r(x, x') = \sqrt{\sum_{i=1}^{p}\left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's.

Value

A Matern(5/2) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Matern.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- Matern52.Kernel(lengthscale)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="Matern52")
Evaluate.Kernel(kernel, X)
```

`MultiplicativeMatern.Kernel`
Multiplicative Generalized Matern Kernel

Description

This function specifies the Multiplicative Generalized Matern kernel.

Usage

```
MultiplicativeMatern.Kernel(lengthscale, nu = 2.01)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters
`nu`	a positive scalar parameter that controls the smoothness

Details

The Multiplicative Generalized Matern kernel is given by

$$k(r;\nu) = \prod_{i=1}^{p} \frac{2^{1-\nu}}{\Gamma(\nu)} (\sqrt{2\nu} r_i)^{\nu} K_{\nu}(\sqrt{2\nu} r_i),$$

where ν is the smoothness parameter, K_ν is the modified Bessel function, Γ is the gamma function, and

$$r_i(x, x') = \sqrt{\left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the dimension-wise euclidean distances between x and x' weighted by the length scale parameters l_i's.

Value

A Multiplicative Generalized Matern Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

Matern.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- MultiplicativeMatern.Kernel(lengthscale, nu=2.01)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="MultiplicativeMatern", parameters=list(nu=2.01))
Evaluate.Kernel(kernel, X)
```

MultiplicativeRQ.Kernel

Multiplicative Rational Quadratic (RQ) Kernel

Description

This function specifies the Multiplicative Rational Quadratic (RQ) kernel.

Usage

```
MultiplicativeRQ.Kernel(lengthscale, alpha = 1)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters
`alpha`	a positive scalar for the scale mixture parameter that controls the relative weighting of large-scale and small-scale variations

Details

The Multiplicative Rational Quadratic (RQ) kernel is given by

$$k(r;\alpha) = \prod_{i=1}^{p} \left(1 + \frac{r_i^2}{2\alpha}\right)^{-\alpha},$$

where α is the scale mixture parameter and

$$r_i(x, x') = \sqrt{\left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the dimension-wise euclidean distances between x and x' weighted by the length scale parameters l_i's.

Value

A Multiplicative Rational Quadratic (RQ) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

RQ.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- MultiplicativeRQ.Kernel(lengthscale, alpha=1)
Evaluate.Kernel(kernel, X)
```

```
# approach 2
kernel <- Get.Kernel(lengthscale, type="MultiplicativeRQ", parameters=list(alpha=1))
Evaluate.Kernel(kernel, X)
```

`MultiplicativeUDF.Kernel`
Multiplicative User Defined Function (UDF) Kernel

Description

This function specifies the Multiplicative kernel with the user defined R function.

Usage

```
MultiplicativeUDF.Kernel(lengthscale, kernel.function)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters
`kernel.function`	user defined kernel function

Details

The Multiplicative User Defined Function (UDF) kernel is given by

$$k(r) = \prod_{i=1}^{p} f(r_i),$$

where f is the user defined kernel function that takes r_i^2 as input, where

$$r_i(x, x') = \sqrt{\left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the dimension-wise euclidean distances between x and x' weighted by the length scale parameters l_i's.

Value

A Multiplicative User Defined Function (UDF) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

UDF.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

kernel.function <- function(sqdist) {return (exp(-sqrt(sqdist)))}

# approach 1
kernel <- MultiplicativeUDF.Kernel(lengthscale, kernel.function=kernel.function)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="MultiplicativeUDF",
                     parameters=list(kernel.function=kernel.function))
Evaluate.Kernel(kernel, X)
```

Ordinary.Kriging *Ordinary Kriging*

Description

This functions fits the ordinary kriging model to the data.

Usage

```
Ordinary.Kriging(
  X,
  y,
  interpolation = TRUE,
  fit = TRUE,
  kernel = NULL,
  kernel.parameters = list(),
  nlopt.parameters = list()
)
```

Arguments

X	a matrix for input (feature)
y	a vector for output (target), only one-dimensional output is supported
interpolation	interpolation whether to interpolate, for noisy data please set interpolate=FALSE
fit	whether to fit the length scale parameters from data
kernel	a kernel class object
kernel.parameters	a list of parameters required for the kernel, if no kernel class object is provided
nlopt.parameters	a list of parameters required for NLopt, including choice of optimization algorithm and maximum number of evaluation

Details

Ordinary kriging assumes a constant mean. Please see Santner et al. (2003) for details.

For data from deterministic computer experiments, use interpolation=TRUE and will give an interpolator. For noisy data, use interpolation=FALSE, which will give an approximator of the underlying function.

The kernel choices are required and can be specified by (i) providing the kernel class object to kernel or (ii) specifying the kernel type and other parameters in kernel.parameters. Please see examples section of Fit.Kriging for detail usages.

When the lengthscale / correlation parameters are unknown, all parameters including the constant mean can be estimated via Maximum Likelihood method by setting fit=TRUE. The initial / lower bound / upper bound of the lengthscale parameters can be provided in kernel.parameters, otherwise a good initial and range would be estimated from the data. The optimization is performed via NLopt, a open-source library for nonlinear optimization. All gradient-free optimization methods in NLopt are supported and can be specified in nlopt.parameters. See nloptr::nloptr.print.options() for the list of available derivative-free algorithms (prefix with NLOPT_GN or NLOPT_LN). The maximum number of optimization steps can also be defined in nlopt.parameters. Please see examples section of Fit.Kriging for detail usages.

Value

A Ordinary Kriging Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Santner, T. J., Williams, B. J., Notz, W. I., & Williams, B. J. (2003). *The design and analysis of computer experiments (Vol. 1)*. New York: Springer.

See Also

Fit.Kriging, Predict.Kriging, Get.Kriging.Parameters.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

# approach 1
kriging <- Ordinary.Kriging(X, y, interpolation=TRUE, fit=TRUE,
                            kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# approach 2
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)
```

Predict.Kriging *Kriging Prediction*

Description

This function gives prediction and uncertainty quantification of the kriging model on a new input.

Usage

```
Predict.Kriging(kriging, X)
```

Arguments

kriging	a kriging class object
X	a matrix for the new input (features) to perform predictions

Value

mean	kriging mean computed at the new input
sd	kriging standard computed at the new input

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. (2006). *Limit kriging*. Technometrics, 48(4), 458-466.

Joseph, V. R. (2024). Rational Kriging. *Journal of the American Statistical Association*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Santner, T. J., Williams, B. J., Notz, W. I., & Williams, B. J. (2003). *The design and analysis of computer experiments (Vol. 1)*. New York: Springer.

See Also

Fit.Kriging.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="OK",
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
```

```
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
```

`Rational.Kriging` *Rational Kriging*

Description

This functions fits the rational kriging model to the data.

Usage

```
Rational.Kriging(
  X,
  y,
  fit = TRUE,
  kernel = NULL,
  kernel.parameters = list(),
  nlopt.parameters = list()
)
```

Arguments

`X`	a matrix for input (feature)
`y`	a vector for output (target), only one-dimensional output is supported
`fit`	whether to fit the length scale parameters from data
`kernel`	a kernel class object
`kernel.parameters`	a list of parameters required for the kernel, if no kernel class object is provided
`nlopt.parameters`	a list of parameters required for NLopt, including choice of optimization algorithm and maximum number of evaluation

Details

Rational kriging gives a rational predictor in terms of the inputs, which gives a more stable estimate of the mean. Please see Joseph (2024) for details. Only interpolation is available. Noisy output is not supported for rational kriging.

The kernel choices are required and can be specified by (i) providing the kernel class object to `kernel` or (ii) specifying the kernel type and other parameters in `kernel.parameters`. Please see examples section of Fit.Kriging for detail usages.

When the lengthscale / correlation parameters are unknown, all parameters including the constant mean can be estimated via Maximum Likelihood method by setting `fit=TRUE`. The initial / lower bound / upper bound of the lengthscale parameters can be provided in `kernel.parameters`, otherwise a good initial and range would be estimated from the data. The optimization is performed via

NLopt, a open-source library for nonlinear optimization. All gradient-free optimization methods in NLopt are supported and can be specified in `nlopt.parameters`. See `nloptr::nloptr.print.options()` for the list of available derivative-free algorithms (prefix with NLOPT_GN or NLOPT_LN). The maximum number of optimization steps can also be defined in `nlopt.parameters`. Please see examples section of Fit.Kriging for detail usages.

Value

A Rational Kriging Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. (2024). Rational Kriging. *Journal of the American Statistical Association*.

See Also

Fit.Kriging, Predict.Kriging, Get.Kriging.Parameters.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

# approach 1
kriging <- Rational.Kriging(X, y, fit=TRUE, kernel.parameters=list(type="RQ",alpha=1))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# approach 2
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="RK",
                       kernel.parameters=list(type="RQ",alpha=1))
```

```
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)
```

`RQ.Kernel` *Rational Quadratic (RQ) Kernel*

Description

This function specifies the Rational Quadratic (RQ) kernel.

Usage

```
RQ.Kernel(lengthscale, alpha = 1)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters
`alpha`	a positive scalar for the scale mixture parameter that controls the relative weighting of large-scale and small-scale variations

Details

The Rational Quadratic (RQ) kernel is given by

$$k(r;\alpha) = \left(1 + \frac{r^2}{2\alpha}\right)^{-\alpha},$$

where α is the scale mixture parameter and

$$r(x,x') = \sqrt{\sum_{i=1}^{p}\left(\frac{x_i - x_i'}{l_i}\right)^2}$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's. As $\alpha \to \infty$, it converges to the Gaussian.Kernel.

Value

A Rational Quadratic (RQ) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

MultiplicativeRQ.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

# approach 1
kernel <- RQ.Kernel(lengthscale, alpha=1)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="RQ", parameters=list(alpha=1))
Evaluate.Kernel(kernel, X)
```

UDF.Kernel *User Defined Function (UDF) Kernel*

Description

This function specifies a kernel with the user defined R function.

Usage

```
UDF.Kernel(lengthscale, kernel.function)
```

Arguments

`lengthscale`	a vector for the positive length scale parameters
`kernel.function`	user defined kernel function

Details

The User Defined Function (UDF) kernel is given by

$$k(r) = f(r)$$

where f is the user defined kernel function that takes r^2 as input, where

$$r(x, x') = \sqrt{\sum_{i=1}^{p} \left(\frac{x_i - x_i'}{l_i} \right)^2},$$

is the euclidean distance between x and x' weighted by the length scale parameters l_i's.

Value

A User Defined Function (UDF) Kernel Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Duvenaud, D. (2014). *The kernel cookbook: Advice on covariance functions*.

Rasmussen, C. E. & Williams, C. K. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

See Also

MultiplicativeUDF.Kernel, Get.Kernel, Evaluate.Kernel.

Examples

```
n <- 5
p <- 3
X <- matrix(rnorm(n*p), ncol=p)
lengthscale <- c(1:p)

kernel.function <- function(sqdist) {return (exp(-sqrt(sqdist)))}

# approach 1
kernel <- UDF.Kernel(lengthscale, kernel.function=kernel.function)
Evaluate.Kernel(kernel, X)

# approach 2
kernel <- Get.Kernel(lengthscale, type="UDF",
                     parameters=list(kernel.function=kernel.function))
Evaluate.Kernel(kernel, X)
```

`Universal.Kriging` *Universal Kriging*

Description

This functions fits the universal kriging model to the data.

Usage

```
Universal.Kriging(
  X,
  y,
  basis.function,
  interpolation = TRUE,
  fit = TRUE,
  kernel = NULL,
  kernel.parameters = list(),
  nlopt.parameters = list()
)
```

Arguments

`X`	a matrix for input (feature)
`y`	a vector for output (target), only one-dimensional output is supported
`basis.function`	the basis functions for specifying the prior mean
`interpolation`	interpolation whether to interpolate, for noisy data please set `interpolate=FALSE`
`fit`	whether to fit the length scale parameters from data
`kernel`	a kernel class object
`kernel.parameters`	a list of parameters required for the kernel, if no kernel class object is provided
`nlopt.parameters`	a list of parameters required for NLopt, including choice of optimization algorithm and maximum number of evaluation

Details

Universal kriging permits a more general function of mean, which can be specified using `basis.function`. Please see Santner et al. (2003) for details.

For data from deterministic computer experiments, use `interpolation=TRUE` and will give an interpolator. For noisy data, use `interpolation=FALSE`, which will give an approximator of the underlying function.

The kernel choices are required and can be specified by (i) providing the kernel class object to `kernel` or (ii) specifying the kernel type and other parameters in `kernel.parameters`. Please see examples section of Fit.Kriging for detail usages.

When the lengthscale / correlation parameters are unknown, all parameters including the constant mean can be estimated via Maximum Likelihood method by setting `fit=TRUE`. The initial / lower bound / upper bound of the lengthscale parameters can be provided in `kernel.parameters`, otherwise a good initial and range would be estimated from the data. The optimization is performed via NLopt, a open-source library for nonlinear optimization. All gradient-free optimization methods in NLopt are supported and can be specified in `nlopt.parameters`. See `nloptr::nloptr.print.options()` for the list of available derivative-free algorithms (prefix with NLOPT_GN or NLOPT_LN). The maximum number of optimization steps can also be defined in `nlopt.parameters`. Please see examples section of Fit.Kriging for detail usages.

Value

A Universal Kriging Class Object.

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Santner, T. J., Williams, B. J., Notz, W. I., & Williams, B. J. (2003). *The design and analysis of computer experiments (Vol. 1)*. New York: Springer.

See Also

Fit.Kriging, Predict.Kriging, Get.Kriging.Parameters.

Examples

```
# one dimensional example
f <- function(x) {
  x <- 0.5 + 2*x
  y <- sin(10*pi*x)/(2*x) + (x-1)^4
  return (y)
}

set.seed(1234)
# train set
n <- 30
p <- 1
X <- matrix(runif(n),ncol=p)
y <- apply(X, 1, f)
newX <- matrix(seq(0,1,length=1001), ncol=p)

basis.function <- function(x) {c(1,x[1],x[1]^2)}

# approach 1
kriging <- Universal.Kriging(X, y, basis.function=basis.function,
                             interpolation=TRUE, fit=TRUE,
                             kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
```

```
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)

# approach 2
kriging <- Fit.Kriging(X, y, interpolation=TRUE, fit=TRUE, model="UK",
                       model.parameters=list(basis.function=basis.function),
                       kernel.parameters=list(type="Gaussian"))
pred <- Predict.Kriging(kriging, newX)
plot(newX, f(newX), "l")
points(X, y, pch=16, col="blue")
lines(newX, pred$mean, col="red", lty=2)
lines(newX, pred$mean-2*pred$sd, col="red", lty=3)
lines(newX, pred$mean+2*pred$sd, col="red", lty=3)
Get.Kriging.Parameters(kriging)
```

Index