# Package: MRFA (via r-universe)

October 14, 2024

**Type** Package

**Title** Fitting and Predicting Large-Scale Nonlinear Regression Problems using Multi-Resolution Functional ANOVA (MRFA) Approach

**Version** 0.6

**Date** 2023-11-01

**Author** Chih-Li Sung

**Maintainer** Chih-Li Sung <sungchih@msu.edu>

**Description** Performs the MRFA approach proposed by Sung et al. (2020) <doi:10.1080/01621459.2019.1595630> to fit and predict nonlinear regression problems, particularly for large-scale and high-dimensional problems. The application includes deterministic or stochastic computer experiments, spatial datasets, and so on.

**License** GPL-2 | GPL-3

**RoxygenNote** 7.2.3

**Depends** R (>= 2.14.1)

**Imports** fields, glmnet, grplasso, methods, plyr, randtoolbox, foreach, stats, graphics, utils

**NeedsCompilation** no

**Date/Publication** 2023-11-10 19:43:20 UTC

**Repository** https://chihli.r-universe.dev

**RemoteUrl** https://github.com/cran/MRFA

**RemoteRef** HEAD

**RemoteSha** 1d414715d5dee945b3493e0e7d406871a309760d

## Contents

---

`aic.MRFA` *Extract AIC from a Fitted Multiresolution Functional ANOVA (MRFA) Model*

---

## Description

The function extracts Akaike information criterion (AIC) from a fitted MRFA model.

## Usage

```
aic.MRFA(fit)
```

## Arguments

| | |
|---|---|
| `fit` | a class MRFA object estimated by `MRFA_fit`. |

## Value

a vector with length `length(lambda)` returing AICs.

## Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

## See Also

`predict.MRFA` for prediction of the MRFA model.

## Examples

```
## Not run:

#####             Testing function: GRAMACY & LEE (2009) function             #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
grlee09 <- function(xx)
{
  x1 <- xx[1]
  x2 <- xx[2]
  x3 <- xx[3]
  x4 <- xx[4]
  x5 <- xx[5]
  x6 <- xx[6]

  term1 <- exp(sin((0.9*(x1+0.48))^10))
```

```
  term2 <- x2 * x3
  term3 <- x4

  y <- term1 + term2 + term3
  return(y)
}

library(MRFA)
#####   Training data and testing data   #####
set.seed(2)
n <- 100; n_rep <- 3; n_new <- 50; d <- 6
X.train <- matrix(runif(d*n), ncol = d)
X.train <- matrix(rep(X.train, each = n_rep), ncol = d)
Y.train <- apply(X.train, 1, grlee09)
Y.train <- Y.train + rnorm(n*n_rep, 0, 0.05)
X.test <- matrix(runif(d*n_new), ncol = d)
Y.test <- apply(X.test, 1, grlee09)

#####   Fitting    #####
MRFA_model <- MRFA_fit(X.train, Y.train)
print(aic.MRFA(MRFA_model))
print(bic.MRFA(MRFA_model))

#####   Prediction : AIC and BIC  ######
lambda.aic <- MRFA_model$lambda[which.min(aic.MRFA(MRFA_model))]
Y.pred <- predict(MRFA_model, X.test, lambda = lambda.aic)$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

lambda.bic <- MRFA_model$lambda[which.min(bic.MRFA(MRFA_model))]
Y.pred <- predict(MRFA_model, X.test, lambda = lambda.bic)$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

## End(Not run)
```

---

`bic.MRFA` *Extract BIC from a Multiresolution Functional ANOVA (MRFA) Model*

---

## Description

The function extracts Bayesian information criterion (BIC) from a fitted MRFA model.

## Usage

```
bic.MRFA(fit)
```

## Arguments

| | |
|---|---|
| `fit` | a class MRFA object estimated by `MRFA_fit`. |

### Value

a vector with length `length(lambda)` returing BICs.

### Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

### See Also

`predict.MRFA` for prediction of the MRFA model.

### Examples

```
## Not run:

#####             Testing function: GRAMACY & LEE (2009) function             #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
grlee09 <- function(xx)
{
  x1 <- xx[1]
  x2 <- xx[2]
  x3 <- xx[3]
  x4 <- xx[4]
  x5 <- xx[5]
  x6 <- xx[6]

  term1 <- exp(sin((0.9*(x1+0.48))^10))
  term2 <- x2 * x3
  term3 <- x4

  y <- term1 + term2 + term3
  return(y)
}

library(MRFA)
#####   Training data and testing data   #####
set.seed(2)
n <- 100; n_rep <- 3; n_new <- 50; d <- 6
X.train <- matrix(runif(d*n), ncol = d)
X.train <- matrix(rep(X.train, each = n_rep), ncol = d)
Y.train <- apply(X.train, 1, grlee09)
Y.train <- Y.train + rnorm(n*n_rep, 0, 0.05)
X.test <- matrix(runif(d*n_new), ncol = d)
Y.test <- apply(X.test, 1, grlee09)

#####   Fitting    #####
MRFA_model <- MRFA_fit(X.train, Y.train)
print(aic.MRFA(MRFA_model))
print(bic.MRFA(MRFA_model))

#####   Prediction : AIC and BIC   ######
lambda.aic <- MRFA_model$lambda[which.min(aic.MRFA(MRFA_model))]
```

```
Y.pred <- predict(MRFA_model, X.test, lambda = lambda.aic)$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

lambda.bic <- MRFA_model$lambda[which.min(bic.MRFA(MRFA_model))]
Y.pred <- predict(MRFA_model, X.test, lambda = lambda.bic)$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

## End(Not run)
```

---

`confidence.MRFA` *Confidence Interval for Multiresolution Functional ANOVA (MRFA) Model*

---

## Description

The function computes the confidence intervals of predicted responses (only works for linear regression model).

## Usage

```
confidence.MRFA(
  object,
  xnew,
  X,
  lambda = object$lambda,
  conf.level = 0.95,
  var.estimation = c("rss", "cv", "posthoc")[1],
  w.estimation = c("cv", "nugget")[1],
  K = 5,
  nugget = 1e-06,
  parallel = FALSE,
  verbose = FALSE
)
```

## Arguments

| | |
|---|---|
| `object` | a class MRFA object estimated by `MRFA_fit`. |
| `xnew` | a testing matrix with dimension `n_new` by `d` in which each row corresponds to a predictive location. |
| `X` | input for `MRFA_fit`. |
| `lambda` | a value. The default is `min(object$lambda)`. |
| `conf.level` | a value specifying confidence level of the confidence interval. The default is 0.95. |
| `var.estimation` | a character string specifying the estimation method for variance. "rss" specifies residual sum of squares, "cv" specifies a cross-validation method with `K` fold, and "posthoc" specifies a post-hoc estimation method. The default is "rss". |

| | |
|---|---|
| w.estimation | a character string specifying the estimation method for weights w. "cv" specifies a cross-validation method with K fold, and "nugget" specifies a least square error method with nugget=nugget. The default is "cv". |
| K | a positive integer specifying the number of folds. |
| nugget | a value specifying the nugget value for w.estimation. The default is 1e-6. It only works when w.estimation="nugget". |
| parallel | logical. If TRUE, apply function in parallel using parallel backend provided by foreach. |
| verbose | logical. If TRUE, additional diagnostics are printed. |

## Details

When The details about var.estimation and w.estimation can be seen in Sung et al. (2017+).

## Value

| | |
|---|---|
| lower bound | a vector with length n_new displaying lower bound of predicted responses at locations xnew. |
| upper bound | a vector with length n_new displaying upper bound of predicted responses at locations xnew. |
| conf.level | as above. |

## Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

## See Also

MRFA_fit for fitting of a multi-resolution functional ANOVA model; predict.MRFA for prediction of a multi-resolution functional ANOVA model.

## Examples

```
## Not run:

#####              Testing function: OTL circuit function                      #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
otlcircuit <- function(xx)
{
  Rb1  <- 50   + xx[1] * 100
  Rb2  <- 25   + xx[2] * 45
  Rf   <- 0.5  + xx[3] * 2.5
  Rc1  <- 1.2  + xx[4] * 1.3
  Rc2  <- 0.25 + xx[5] * 0.95
  beta <- 50   + xx[6] * 250

  Vb1 <- 12*Rb2 / (Rb1+Rb2)
  term1a <- (Vb1+0.74) * beta * (Rc2+9)
  term1b <- beta*(Rc2+9) + Rf
```

```
  term1 <- term1a / term1b

  term2a <- 11.35 * Rf
  term2b <- beta*(Rc2+9) + Rf
  term2 <- term2a / term2b

  term3a <- 0.74 * Rf * beta * (Rc2+9)
  term3b <- (beta*(Rc2+9)+Rf) * Rc1
  term3 <- term3a / term3b

  Vm <- term1 + term2 + term3
  return(Vm)
}



library(MRFA)
#####   training data and testing data   #############
set.seed(2)
n <- 100; n_new <- 10; d <- 6
X.train <- matrix(runif(d*n), ncol = d)
Y.train <- apply(X.train, 1, otlcircuit)
X.test <- matrix(runif(d*n_new), ncol = d)
Y.test <- apply(X.test, 1, otlcircuit)

#####   Fitting    #####
MRFA_model <- MRFA_fit(X.train, Y.train)

#####   Prediction   ######
Y.pred <- predict(MRFA_model, X.test, lambda = min(MRFA_model$lambda))$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

### confidence interval ###
conf.interval <- confidence.MRFA(MRFA_model, X.test, X.train, lambda = min(MRFA_model$lambda))
print(conf.interval)

## End(Not run)
```

---

`cv.MRFA` *Compute K-fold cross-validated error for Multi-Resolution Functional ANOVA (MRFA) Model*

---

## Description

Computes the K-fold cross validated mean squared prediction error for multiresolution functional ANOVA model.

## Usage

```
cv.MRFA(
```

```
  X,
  Y,
  order = 10,
  level = 10,
  lambda = exp(seq(log(500), log(0.001), by = -0.01)),
  K = 10,
  plot.it = TRUE,
  parallel = FALSE,
  verbose = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| X | input for MRFA_fit. |
| Y | input for MRFA_fit. |
| order | input for MRFA_fit. |
| level | input for MRFA_fit. |
| lambda | lambda values at which CV curve should be computed. |
| K | a positive integer specifying the number of folds. |
| plot.it | logical. If TRUE, a CV curve will be shown. The default is TRUE. |
| parallel | logical. If TRUE, apply cross-validation function in parallel using parallel backend provided by foreach. The default is FALSE. |
| verbose | logical. If TRUE, additional diagnostics are printed. The default is FALSE. |
| ... | additional arguments to MRFA_fit. |

## Value

| | |
|---|---|
| lambda | lambda values at which CV curve is computed. |
| cv | the CV curve at each value of lambda. |
| cv.error | the standard error of the CV curve |

## Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

## See Also

MRFA_fit for fitting a multiresolution functional ANOVA model.

## Examples

```
## Not run:

#####             Testing function: GRAMACY & LEE (2009) function             #####
#####   Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
grlee09 <- function(xx)
```

```
{
  x1 <- xx[1]
  x2 <- xx[2]
  x3 <- xx[3]
  x4 <- xx[4]
  x5 <- xx[5]
  x6 <- xx[6]

  term1 <- exp(sin((0.9*(x1+0.48))^10))
  term2 <- x2 * x3
  term3 <- x4

  y <- term1 + term2 + term3
  return(y)
}

library(MRFA)
#####   Training data and testing data   #####
set.seed(2)
n <- 100; n_rep <- 3; n_new <- 50; d <- 6
X.train <- matrix(runif(d*n), ncol = d)
X.train <- matrix(rep(X.train, each = n_rep), ncol = d)
Y.train <- apply(X.train, 1, grlee09)
Y.train <- Y.train + rnorm(n*n_rep, 0, 0.05)
X.test <- matrix(runif(d*n_new), ncol = d)
Y.test <- apply(X.test, 1, grlee09)

#####   Fitting    #####
MRFA_model <- MRFA_fit(X.train, Y.train)

#####   Computes the K-fold cross validated   #####
cv.out <- cv.MRFA(X.train, Y.train, K = 5, lambda = seq(0.01,3,0.1))

#####   Prediction : CV  ######
lambda_cv <- cv.out$lambda[which.min(cv.out$cv)]
Y.pred <- predict(MRFA_model, X.test, lambda = lambda_cv)$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

## End(Not run)
```

---

`MRFA_fit` *Fit a Multi-Resolution Functional ANOVA (MRFA) Model*

---

## Description

The function performs the multi-resolution functional ANOVA (MRFA) approach.

## Usage

```
MRFA_fit(
```

```
    X,
    Y,
    weights = rep(1, length(Y)),
    order = 10,
    level = 10,
    lambda.min = 1e-05,
    converge.tol = 1e-10,
    nvar.max = min(3 * length(Y), 3000),
    k = 2,
    pen.norm = c("2", "N")[1],
    model = LinReg(),
    standardize.d = TRUE,
    center = TRUE,
    standardize = TRUE,
    parallel = FALSE,
    verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| X | a design matrix with dimension n by d. |
| Y | a response vector of size n. |
| weights | a vector of observation weights. |
| order | a positive integer specifying the highest order of interactions that can be entertained in the model. The default is 10. |
| level | a positive integer specifying the highest resolution level that can be entertained in the model. The default is 10. |
| lambda.min | a positive value specifying the minimum penalty value to be performed before the convergence criterion is met. |
| converge.tol | convergence tolerance. It converges when relative difference with respect to function value (penalized likelihood) is smaller than the tolerance. The default is 1e-10. |
| nvar.max | maximum number of non-zero variables. |
| k | a positive integer specifying the order of Wendland covariance function. The default is 2. |
| pen.norm | a character string specifying the type of penalty norm for group lasso to be computed. "2" or 2 specifies 2-norm, and "N" specifies native norm. The default is "2". |
| model | an object of class specifying other models. LinReg() (default) fits a linear regression, LogReg() fits a logistic regression, and PoissReg() fits a Poisson regression. |
| standardize.d | logical. If TRUE, the columns of the design matrix will be standardized into [0,1]. |
| center | logical. If TRUE, the columns of the model matrix will be centered (except a possible intercept column). |
| standardize | logical. If TRUE, the model matrix will be blockwise orthonormalized. |

| | |
|---|---|
| parallel | logical. If TRUE, apply function in parallel in ldply using parallel backend provided by foreach. |
| verbose | logical. If TRUE, additional diagnostics are printed. |

## Details

A multi-resolution functional ANOVA (MRFA) model targets a low resolution representation of a low order functional ANOVA, with respect to strong effect heredity, to form an accurate emulator in a large-scale and high dimensional problem. This function fits an MRFA model using a modified group lasso algrithm. One can consider the loss function

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\sum_{|u|=1}^{D_{\max}}\sum_{r=1}^{R_{\max}}\sum_{k=1}^{n_u(r)}\beta_u^{rk}\varphi_u^{rk}(x_{iu})\right)^2+\lambda\sum_{|u|=1}^{D_{\max}}\sum_{r=1}^{R_{\max}}\sqrt{N_u(r)\sum_{v\subseteq u}\sum_{s\leq r}\sum_{k=1}^{n_v(s)}(\beta_v^{sk})^2},$$

where $\varphi_u^{rk}(x_{iu})$ is the basis function with resolution level $r$ and with dimension $u\subset\{1,2,\ldots,d\}$, and $D_{\max}$ and $R_{\max}$ respectively are the maximal orders of functional ANOVA and multi-resolution level, which are indicated by `order` and `level`.

The group lasso path along the penalty parameter $\lambda$ is given by the function, where the $\lambda_{\max}$ is automatically given and $\lambda_{\min}$ is given by users, which is indicated by `lambda.min`. The group lasso algrithm is implemented via the modifications to the source code of the `grplasso` package (Meier, 2015).

`lambda.min`, `converge.tol` and `nvar.max` are the options for stopping the fitting process. Smaller `lambda.min`, or smaller `converge.tol`, or larger `nvar.max` yields more accurate results, paricularly for deterministic computer experiments. `pen.norm` specifies the type of penalty norm in the loss function. `model` specifies the response type, which can be non-continuous response, in the case the loss function is replaced by negative log-likelihood function. More details can be seen in Sung et al. (2017+).

## Value

An MRFA object is returned, for which `aic.MRFA`, `bic.MRFA` and `predict` methods exist.

## Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

## See Also

`predict.MRFA` for prediction of the MRFA model.

## Examples

```
## Not run:

#####             Testing function: OTL circuit function                  #####
#####  Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
otlcircuit <- function(xx)
{
```

```
  Rb1  <- 50   + xx[1] * 100
  Rb2  <- 25   + xx[2] * 45
  Rf   <- 0.5  + xx[3] * 2.5
  Rc1  <- 1.2  + xx[4] * 1.3
  Rc2  <- 0.25 + xx[5] * 0.95
  beta <- 50   + xx[6] * 250

  Vb1 <- 12*Rb2 / (Rb1+Rb2)
  term1a <- (Vb1+0.74) * beta * (Rc2+9)
  term1b <- beta*(Rc2+9) + Rf
  term1 <- term1a / term1b

  term2a <- 11.35 * Rf
  term2b <- beta*(Rc2+9) + Rf
  term2 <- term2a / term2b

  term3a <- 0.74 * Rf * beta * (Rc2+9)
  term3b <- (beta*(Rc2+9)+Rf) * Rc1
  term3 <- term3a / term3b

  Vm <- term1 + term2 + term3
  return(Vm)
}

library(MRFA)
#####   Training data and testing data   #####
set.seed(2)
n <- 1000; n_new <- 100; d <- 6
X.train <- matrix(runif(d*n), ncol = d)
Y.train <- apply(X.train, 1, otlcircuit)
X.test <- matrix(runif(d*n_new), ncol = d)
Y.test <- apply(X.test, 1, otlcircuit)

#####   Fitting    #####
MRFA_model <- MRFA_fit(X.train, Y.train, verbose = TRUE)

#####   Prediction   ######
Y.pred <- predict(MRFA_model, X.test, lambda = min(MRFA_model$lambda))$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

## End(Not run)
```

---

`predict.MRFA` *Prediction of Multi-Resolution Functional ANOVA (MRFA) Model*

---

## Description

The function computes the predicted responses.

## Usage

```
## S3 method for class 'MRFA'
predict(object, xnew, lambda = object$lambda, parallel = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | a class MRFA object estimated by MRFA_fit. |
| xnew | a testing matrix with dimension n_new by d in which each row corresponds to a predictive location. |
| lambda | a value, or vector of values, indexing the path. The default is object$lambda. |
| parallel | logical. If TRUE, apply function in parallel in ldply using parallel backend provided by foreach. |
| ... | for compatibility with generic method predict. |

## Value

| | |
|---|---|
| lambda | as above. |
| coefficients | coefficients with respect to the basis function value. |
| y_hat | a matrix with dimension n_new by length(lambda) displaying predicted responses at locations xnew. |

## Author(s)

Chih-Li Sung <iamdfchile@gmail.com>

## See Also

MRFA_fit for fitting a multiresolution functional ANOVA model.

## Examples

```
## Not run:

#####             Testing function: OTL circuit function                    #####
#####  Thanks to Sonja Surjanovic and Derek Bingham, Simon Fraser University #####
otlcircuit <- function(xx)
{
  Rb1  <- 50   + xx[1] * 100
  Rb2  <- 25   + xx[2] * 45
  Rf   <- 0.5  + xx[3] * 2.5
  Rc1  <- 1.2  + xx[4] * 1.3
  Rc2  <- 0.25 + xx[5] * 0.95
  beta <- 50   + xx[6] * 250

  Vb1 <- 12*Rb2 / (Rb1+Rb2)
  term1a <- (Vb1+0.74) * beta * (Rc2+9)
  term1b <- beta*(Rc2+9) + Rf
  term1 <- term1a / term1b
```

```
  term2a <- 11.35 * Rf
  term2b <- beta*(Rc2+9) + Rf
  term2 <- term2a / term2b

  term3a <- 0.74 * Rf * beta * (Rc2+9)
  term3b <- (beta*(Rc2+9)+Rf) * Rc1
  term3 <- term3a / term3b

  Vm <- term1 + term2 + term3
  return(Vm)
}

library(MRFA)
#####   Training data and testing data   #####
set.seed(2)
n <- 1000; n_new <- 100; d <- 6
X.train <- matrix(runif(d*n), ncol = d)
Y.train <- apply(X.train, 1, otlcircuit)
X.test <- matrix(runif(d*n_new), ncol = d)
Y.test <- apply(X.test, 1, otlcircuit)

#####   Fitting    #####
MRFA_model <- MRFA_fit(X.train, Y.train, verbose = TRUE)

#####   Prediction   ######
Y.pred <- predict(MRFA_model, X.test, lambda = min(MRFA_model$lambda))$y_hat
print(sqrt(mean((Y.test - Y.pred)^2)))

## End(Not run)
```

# Index