Package: metaSVR (via r-universe)

May 18, 2026

Type Package

Title Support Vector Regression with Metaheuristic Algorithms Optimization

Version 0.1.0

Description Provides a hybrid modeling framework combining Support Vector Regression (SVR) with metaheuristic optimization algorithms, including the Archimedes Optimization Algorithm (AO) (Hashim et al. (2021) <doi:10.1007/s10489-020-01893-z>), Coot Bird Optimization (CBO) (Naruei & Keynia (2021) <doi:10.1016/j.eswa.2021.115352>), and their hybrid (AOCBO), as well as several others such as Harris Hawks Optimization (HHO) (Heidari et al. (2019) <doi:10.1016/j.future.2019.02.028>), Gray Wolf Optimizer (GWO) (Mirjalili et al. (2014) <doi:10.1016/j.advengsoft.2013.12.007>), Ant Lion Optimization (ALO) (Mirjalili (2015) <doi:10.1016/j.advengsoft.2015.01.010>), and Enhanced Harris Hawk Optimization with Coot Bird Optimization (EHHOCBO) (Cui et al. (2023) <doi:10.32604/cmes.2023.026019>). The package enables automatic tuning of SVR hyperparameters (cost, gamma, and epsilon) to enhance prediction performance. Suitable for regression tasks in domains such as renewable energy forecasting and hourly data prediction. For more details about implementation and parameter bounds see: Setiawan et al. (2021) <doi:10.1016/j.procs.2020.12.003> and Liu et al. (2018) <doi:10.1155/2018/6076475>.

Imports e1071, stats, hms

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/rechtianaputri/metaSVR

BugReports https://github.com/rechtianaputri/metaSVR/issues

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5.0)

Repository https://rechtianaputri.r-universe.dev

Date/Publication 2025-07-29 05:00:55 UTC

RemoteUrl https://github.com/rechtianaputri/metasvr

RemoteRef HEAD

RemoteSha 89116804d84b4d513bd6e1b1f9664f123aec4d76

Contents

`ALO` *Ant Lion Optimizer*

Description

An algorithm built by Mirjalili (2015) inspired by the hunting behaviour of antlion whose making pit trap for ant prey in order to optimized real-valued objective function in continuous search space in a population-based manner.

Usage

```
ALO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.
fobj	An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

The algorithm mimics the ALO hunting behaviour by simulating a stochastic search where ants move around randomly under the influence of selected antlions and an elite antlion.

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors 'lb' and 'ub' must have the same length as the number of dimensions 'dim'.

This optimization function used inside svrHybrid function.

References

Mirjalili, S. (2015). The Ant Lion Optimizer. Advances in Engineering Software, 83, 80-98. https://doi.org/10.1016/j.advengsoft.2015.01.010.

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# ALO optimization
set.seed(123)
result <- ALO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)

# View best fitness and position found
result$best_fitness
result$best_position
}
```

AO *Archimedes Optimization*

Description

An algorithm built by Hashim et al. (2021) use buoyancy law and fluid dynamics behavior in Archimedes principle to optimized real-valued objective function in continuous search space in a population-based manner.

Usage

```
AO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.
fobj	An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

This algorithm uses population-based search to conduct physical law such as volume, density difference, and acceleration in every iteration. It balancing the exploration and exploitation phase by using Transfer Function (TF) as a shifting indicates.

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors 'lb' and 'ub' must have the same length as the number of dimensions 'dim'.

This optimization function used inside svrHybrid function.

Constant of C3 = 1 and C4 = 2 used in basic standard optimization function.

References

Hashim, F. A., Hussain, K., Houssein, E. H., Mabrouk, M. S., & Al-Atabany, W. (2021). Archimedes Optimization Algorithm: A New Metaheuristic Algorithm for Solving Optimization Problems. Applied Intelligence, 51(3), 1531–1551. https://doi.org/10.1007/s10489-020-01893-z

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# AO optimization
set.seed(123)
result <- AO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)

# View best fitness and position found
result$best_fitness
result$best_position
}
```

AOCBO *Combined Archimedes Optimization with Coot Bird Optimization*

Description

A hybrid metaheuristic algorithm that combines Archimedes Optimization (AO) with Coot Bird Optimization (CBO) to optimized real-valued objective function in continuous search space.

Usage

```
AOCBO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.
fobj	An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

This metaheuristic implement combination of all step of Archimedes Optimization with first step used after initialization is Coot Leader selection stage in CBO as early exploration step. The hybrid design enhances convergence and stability in optimization step so it can maximize the best parameter.

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors ‘lb‘ and ‘ub‘ must have the same length as the number of dimensions ‘dim‘.

This optimization function used inside svrHybrid function.

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# AOCBO optimization
set.seed(123)
result <- AOCBO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)
```

```
# View best fitness and position found
result$best_fitness
result$best_position
}
```

CBO *Coot Bird Optimization*

Description

An algorithm built by Naruei & Keynia (2021) that mimics the regular-irregular movement behaviour of Coot birds. Its population divided by two groups as leaders to guide the process and coots to follow leaders and randomly explore search space. This movement use to optimized real-valued objective function in continuous search space.

Usage

```
CBO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.
fobj	An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

This algorithms used movement such as: random movement, chain movement, adjusting the position based on the group leaders, and leader movement to emphasize the exploration and exploitation phase to get the best fitness.

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors 'lb' and 'ub' must have the same length as the number of dimensions 'dim'.

This optimization function used inside svrHybrid function.

References

Naruei, I., & Keynia, F. (2021). A New Optimization Method Based on COOT Bird Natural Life Model. Expert Systems with Applications, 183. https://doi.org/10.1016/j.eswa.2021.115352

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# CBO optimization
set.seed(123)
result <- CBO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)

# View best fitness and position found
result$best_fitness
result$best_position
}
```

`denormalize` *Denormalize*

Description

Convert normalized data back to original scale using given min and max.

Usage

```
denormalize(x, min, max)
```

Arguments

x	Numeric vector that has been normalized (values is between 0 and 1).
min	The minimum value of the original data.
max	The maximum value of the original data.

Value

A numeric vector already converted to original scale.

Examples

```
# Example of the original data
original_data <- c(10, 20, 30, 40, 50)

# Data being normalized
normalized_data <- (original_data - min(original_data)) /
                   (max(original_data) - min(original_data))

# Denormalization function use to change value to the original
denormalize(normalized_data, min(original_data), max(original_data))
```

EHHOCBO *Enhanced Harris Hawks Optimization with Coot Bird Optimization*

Description

This function implements a hybrid metaheuristic optimization algorithm that combines Harris Hawks Optimization with leader selection of Coot Bird Optimization to optimized real-valued objective function in continuous search space in a population-based manner built by Cui et al. (2023).

Usage

```
EHHOCBO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.
fobj	An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

This algorithm start by adding leadership mechanism of CBO into HHO process so it can make better foundation for the global search. Ensemble Mutation Strategy (EMS) to improve the exploration trend and population diversity also Refracted Opposition-Based Learning (ROBL) to update current optimal solution in the swarm added to enhanced the combination of HHO and CBO.

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors ‘lb‘ and ‘ub‘ must have the same length as the number of dimensions ‘dim‘.

This optimization function used inside svrHybrid function.

References

Cui, H., Guo, Y., Xiao, Y., Wang, Y., Li, J., Zhang, Y., & Zhang, H. (2023). Enhanced Harris Hawks Optimization Integrated with Coot Bird Optimization for Solving Continuous Numerical Optimization Problems. CMES - Computer Modeling in Engineering and Sciences, 137(2), 1635–1675. https://doi.org/10.32604/cmes.2023.026019

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# EHHOCBO optimization
set.seed(123)
result <- EHHOCBO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)

# View best fitness and position found
result$best_fitness
result$best_position
}
```

`get_default_bounds` *Default Bounds Initialization for SVR Optimization*

Description

This function return the default value of lower and upper bounds also the dimension for SVR optimization. The three dimensions represent as the parameter that need to be optimized in SVR with exact range of bound. Three dimension and the range represent as: Cost (C): 2^0 to 2^10; Gamma: 2^(-8) to 2^0; Epsilon: 2^(-8) to 2^0.

Usage

```
get_default_bounds()
```

Value

A list containing:

lb A numeric vector of lower bounds.

ub A numeric vector of upper bounds.

dim An integer representing the number of dimensions, 3.

Note

The bounds for parameters search space is based on previous research with range: Cost=[2^0,2^10], Gamma=[2^(-8),2^0], dan Epsilon=[2^(-8),2^0]

References

Liu, H.-H., Chang, L.-C., Li, C.-W., & Yang, C.-H. (2018). Particle Swarm Optimization-Based Support Vector Regression for Tourist Arrivals Forecasting. Computational Intelligence and Neuroscience, 2018, 1–13. https://doi.org/10.1155/2018/6076475.

Examples

```
bounds <- get_default_bounds()
bounds$lb  # Lower bounds
bounds$ub  # Upper bounds
bounds$dim # Number of parameters
```

GWO *Grey Wolf Optimizer*

Description

An algorithm built by Mirjalili et al. (2014) inspired by leadership hierarchy and hunting mechanism of grey wolves in nature to optimized real-valued objective function in continuous search space in a population-based manner.

Usage

```
GWO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.
fobj	An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

This algorithm proposed social hierarchy on GWO to obtain the best fitness and get the best proposed hunting method to locate probable position of the pray. Adaptive values on alpha and A make it possible smooth transition between exploration and exploitation phase.

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors 'lb' and 'ub' must have the same length as the number of dimensions 'dim'.

This optimization function used inside svrHybrid function.

References

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. Advances in engineering software, 69, 46-61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# GWO optimization
set.seed(123)
result <- GWO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)

# View best fitness and position found
result$best_fitness
result$best_position
}
```

HHO *Harris Hawks Optimization*

Description

An algorithm built by Heidari et al. (2019) that inspired by the movement of Harris Hawks on cooperative hunting behaviour to optimized real-valued objective function in continous search space in a population-based manner.

Usage

```
HHO(N, Max_iter, lb, ub, dim, fobj)
```

Arguments

N	An integer indicate population size.
Max_iter	An integer indicate maximum number of iterations.
lb	A numeric vector that show lower bounds of the search space. One value per dimension.
ub	A numeric vector that show upper bounds of the search space. One value per dimension.
dim	An integer show the number of dimension (parameters) of the problem to optimize. It indicate the number of parameters to be optimized.

`fobj` An objective function used to be minimized. It is return single numeric value that show evaluation matrix result in every iteration. It used to calculate the best fitness in every iteration.

Details

There are two phase of Harris Hawks hunting, namely exploration and exploitation that will be modelized to find optimization result. The movement used in this algorithms such as: exploration phase; transition between exploitation and exploration phase; and exploitation phase that has 4 different strategies based on E and r (soft besiege, hard besiege, soft besiege with progressive rapid, and hard besiege with progressive rapid).

The algorithm performs until maximum iteration reached or convergence condition when the difference in objective values for ten consecutive times is less than 10^-5.

Value

A list containing:

best_fitness The best (minimum) fitness value found.

best_position The parameter vector (position) corresponding to the best fitness.

jml_iter The number of iterations executed.

param Matrix of best parameters found across every iterations (dim × iter).

param_list Vector of best fitness values at each iteration.

Note

The input vectors ‘lb‘ and ‘ub‘ must have the same length as the number of dimensions ‘dim‘.

This optimization function used inside svrHybrid function.

References

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. Future generation computer systems, 97, 849-872. https://doi.org/10.1016/j.future.2019.02.028

Examples

```
{
sphere_fn <- function(x) sum(x^2) # simple function for objective function

# HHO optimization
set.seed(123)
result <- HHO(N = 20, Max_iter = 50, lb = c(-5,-5,-5), ub = c(5,5,5), dim = 3, fobj = sphere_fn)

# View best fitness and position found
result$best_fitness
result$best_position
}
```

loss_calculate *Calculate Loss Based on Selected Objective Function*

Description

Compute the loss between predictive and actual values using a selected objective function. Supported objecive functions used in this functions are: "SMAPE', "MAPE", "RMSE", and "MAE".

Usage

```
loss_calculate(preds, actuals, objective)
```

Arguments

preds	A numeric vector of predicted values.
actuals	A numeric vector of actual (true) values.
objective	A string character that indicates the loss function type: "SMAPE", "MAPE", "RMSE", or "MAE".

Value

A numeric value that represent the computed loss.

Examples

```
preds <- c(80, 120, 180)
actuals <- c(95, 115, 177)
loss_calculate(preds, actuals, "RMSE")
```

mae *Mean Absolute Error*

Description

Calculate the RMSE value between predicted and actual values.

Usage

```
mae(preds, actuals)
```

Arguments

preds	A numeric vector of predicted values.
actuals	A numeric vector of actual (true) values.

Value

MAE value.

Examples

```
preds <- c(80, 120, 180)
actuals <- c(95, 115, 177)
mae(preds, actuals)
```

`mape` *Mean Absolute Percentage Error*

Description

Calculate the MAPE value between predicted and actual values. Can't be used if the actual values contain 0 value.

Usage

```
mape(preds, actuals)
```

Arguments

`preds`	A numeric vector of predicted values.
`actuals`	A numeric vector of actual (true) values.

Value

MAPE value (percentage).

Examples

```
preds <- c(80, 120, 180)
actuals <- c(95, 115, 177)
mape(preds, actuals)
```

normalize *Normalize*

Description

Normalize data using min-max scale.

Usage

```
normalize(x)
```

Arguments

x	is a predictor variable that is a numeric vector to be normalized.

Value

A numeric vector scaled between 0 and 1.

Examples

```
# Normalize example use:
data <- c(10, 20, 30, 40, 50)
normalize(data)
```

rmse *Root Mean Squared Error*

Description

Calculate the RMSE value between predicted and actual values.

Usage

```
rmse(preds, actuals)
```

Arguments

preds	A numeric vector of predicted values.
actuals	A numeric vector of actual (true) values.

Value

RMSE value.

Examples

```
preds <- c(80, 120, 180)
actuals <- c(95, 115, 177)
rmse(preds, actuals)
```

`smape` *Symmetric Mean Absolute Percentage Error*

Description

Calculate the SMAPE value between predicted and actual values.

Usage

```
smape(preds, actuals)
```

Arguments

`preds`	A numeric vector of predicted values.
`actuals`	A numeric vector of actual (true) values.

Value

SMAPE value (percentage).

Examples

```
preds <- c(80, 120, 180)
actuals <- c(95, 115, 177)
smape(preds, actuals)
```

`svrHybrid` *SVR with Metaheuristic Algorithms Optimization*

Description

Trains a Support vector Regression Model by optimizing its parameter (Cost, Gamma, and Epsilon) using Metaheuristic Algorithms such as: Archimedes Optimization (AO), Coot Bird Optimization (CBO), Combined Archimedes Optimization with Coot Bird Optimization (AOCBO), Harris Hawks Optimization (HHO), Grey Wolf Optimizer (GWO), Ant Lion Optimization (ALO), and Enhanced Harris Hawks Optimization with Coot Bird Optimization (EHHOCBO).

Usage

```
svrHybrid(
  x_train,
  y_train,
  x_test,
  y_test,
  kernel = "radial",
  optimizer = "AO",
  objective = "RMSE",
  is.y.normalize = FALSE,
  min.y = min.y,
  max.y = max.y,
  max_iter = 100,
  N = 30,
  seed = 123,
  degree = 3,
  coef0 = 0,
  nu = 0.5,
  class.weights = NULL,
  cachesize = 40,
  tolerance = 0.001,
  scale = TRUE,
  shrinking = TRUE,
  cross = 0,
  probability = FALSE,
  fitted = TRUE,
  subset,
  na.action = na.omit
)
```

Arguments

x_train	A matrix or data frame contain predictors variable for training the model.
y_train	A numeric vector of target values for training model.
x_test	A matrix or data frame contain predictors variable for testing the model. It can be replaced by data validation to get the parameter if you separated the data as three categories and need more reliable model.
y_test	A numeric vector of target values for training model. It can be replaced by data validation to get the parameter if you separated the data as three categories and need more reliable model.
kernel	SVR kernel type used for modelling. Options: "linear", "radial", "polynomial", or "sigmoid". Default is radial.
optimizer	Metaheuristic Algorithms selection, options: "AO", "CBO", "AOCBO", "HHO", "GWO", "ALO", or "EHHOCBO". Default is AO.
objective	Objective function used for optimization as prediction quality measures. Options: "SMAPE", "MAPE", "RMSE", or "MAE". Default is RMSE.

is.y.normalize	Logical; use when prediction of target variable 'y' is on min-max scalling normalization. Default is FALSE. Note: It is only use when the data normalize by normalize() function in this package.
min.y	Minimum value of target (used for denormalization). No need to fill this parameter if y is not normalize.
max.y	Maximum value of target (used for denormalization). No need to fill this parameter if y is not normalize.
max_iter	Maximum number of iterations for the optimizer. Default is 100.
N	Population size for the optimizer. Default is 30.
seed	Random seed for reproducibility algorithm. Default is 123.
degree	Degree parameter for polynomial kernel.Default is 3.
coef0	Coefficient parameter used in polynomial/sigmoid kernels.
nu	Parameter for 'nu-regression' to controlling max proportion of error training and minimum proportion of support vectors. Default is 0.5, range: 0.1-0.9. Only use if the type of regression choosen is 'nu-regression'.
class.weights	A named list of class weights.
cachesize	Size of kernel cache (in MB). Default is 40.
tolerance	Tolerance of termination criterion.
scale	Logical; whether to scale inputs. Default is TRUE.
shrinking	Logical; whether to use shrinking heuristics. Default is TRUE.
cross	Number of folds for cross-validation. Default is 0, no cross validation.
probability	Logical; whether to enable probability model. Default is FALSE.
fitted	Logical; whether to keep fitted values. Default is TRUE.
subset	Optional vector specifying subset of observations to be used in the training fit.
na.action	Function which indicates what should happen when the data contain NAs.

Value

A list containing:

best_params A list with the best values for 'cost', 'gamma', and 'epsilon'.

total_iter Total number of iterations run by the optimizer.

model The final trained SVR model (using 'e1071::svm').

time Total training time in HMS format.

Examples

```
{
set.seed(1)
x <- matrix(rnorm(100), ncol = 2)
y <- x[,1] * 3 + rnorm(50)
model <- svrHybrid(x_train = x[1:40,], y_train = y[1:40],
                   x_test = x[41:50,], y_test = y[41:50],
```

```
                    kernel = "radial", optimizer = "AO",
                    objective = "RMSE", is.y.normalize = FALSE)
# To
model$best_params
}
```

Index