Package 'SBOAtools'

May 7, 2026

Type Package

Title Secretary Bird Optimization for Continuous Optimization and Neural Network Training

Version 0.1.1

Description Provides an implementation of Secretary Bird Optimization for general-purpose continuous optimization, benchmark optimization, and training single-hidden-layer feed-forward neural network models. The implemented optimizer is based on the Secretary Bird Optimization Algorithm proposed by Fu et al. (2024) <doi:10.1007/s10462-024-10729-y>. The neural network training functionality is based on Dilber and Özdemir (2026) <doi:10.1007/s00521-026-11874-x>.

License MIT + file LICENSE

URL https://github.com/burakdilber/SBOAtools

BugReports https://github.com/burakdilber/SBOAtools/issues

Encoding UTF-8

RoxygenNote 7.3.3

Imports stats, graphics

NeedsCompilation no

Author Burak Dilber [aut, cre, cph], A. Fırat Özdemir [aut, cph]

Maintainer Burak Dilber <burakdilber91@gmail.com>

Repository CRAN

Date/Publication 2026-05-03 06:30:17 UTC

Contents

`apply_minmax` *Apply min-max normalization using existing bounds*

Description

Apply min-max normalization using existing bounds

Usage

```
apply_minmax(x, mins, maxs)
```

Arguments

`x`	Numeric vector or matrix.
`mins`	Column minima.
`maxs`	Column maxima.

Value

Scaled matrix.

clip_bounds *Clip values to lower and upper bounds*

Description

Restricts each element of a vector to the given bounds.

Usage

```
clip_bounds(x, lower, upper)
```

Arguments

x	Numeric vector.
lower	Lower bound vector.
upper	Upper bound vector.

Value

A bounded numeric vector.

denormalize_minmax *Reverse min-max normalization*

Description

Reverse min-max normalization

Usage

```
denormalize_minmax(x_scaled, mins, maxs)
```

Arguments

x_scaled	Scaled numeric vector or matrix.
mins	Column minima.
maxs	Column maxima.

Value

Values on the original scale.

forward_mlp *Forward pass for a single-hidden-layer MLP*

Description

Computes hidden-layer activations and output predictions.

Usage

```
forward_mlp(params, X, input_dim, hidden_dim, output_dim)
```

Arguments

params	Numeric parameter vector.
X	Input matrix.
input_dim	Number of input variables.
hidden_dim	Number of hidden neurons.
output_dim	Number of output neurons.

Value

A list containing hidden activations and predictions.

get_benchmark *Get a built-in benchmark definition*

Description

Returns a built-in benchmark definition by name.

Usage

```
get_benchmark(name)
```

Arguments

name	Benchmark name such as "F1" or "F14".

Value

A list containing the benchmark function and its metadata.

Examples

```
b <- get_benchmark("F1")
b$fn(rep(0, 5))
```

levy_flight *Lévy flight step generator*

Description

Generates Lévy-distributed random steps.

Usage

```
levy_flight(d, beta = 1.5)
```

Arguments

d	Dimension of the step vector.
beta	Stability parameter.

Value

A numeric vector of length d.

list_benchmarks *List built-in benchmark functions*

Description

Returns a summary table of the built-in F1-F23 benchmark functions.

Usage

```
list_benchmarks()
```

Value

A data frame.

Examples

```
list_benchmarks()
```

mae_vec *Mean absolute error*

Description

Computes the mean absolute error between actual and predicted values.

Usage

```
mae_vec(actual, predicted)
```

Arguments

actual	Numeric vector or matrix of observed values.
predicted	Numeric vector or matrix of predicted values.

Value

A numeric scalar.

mape_vec *Mean absolute percentage error*

Description

Computes the mean absolute percentage error between actual and predicted values.

Usage

```
mape_vec(actual, predicted, eps = 1e-08)
```

Arguments

actual	Numeric vector or matrix of observed values.
predicted	Numeric vector or matrix of predicted values.
eps	Small constant to avoid division by zero.

Value

A numeric scalar.

mlp_mse_fitness *MSE fitness function for MLP training*

Description

Computes mean squared error for a single-hidden-layer MLP.

Usage

```
mlp_mse_fitness(params, X, Y, input_dim, hidden_dim, output_dim)
```

Arguments

params	Numeric parameter vector.
X	Input matrix.
Y	Output matrix.
input_dim	Number of input variables.
hidden_dim	Number of hidden neurons.
output_dim	Number of output neurons.

Value

Mean squared error.

normalize_minmax *Min-max normalization*

Description

Scales each column of a matrix to the interval from 0 to 1.

Usage

```
normalize_minmax(x)
```

Arguments

x	Numeric vector or matrix.

Value

A list containing scaled data and min-max values.

plot.sboa *Plot method for SBOA objects*

Description

Plots the convergence curve of an SBOA optimization result.

Usage

```
## S3 method for class 'sboa'
plot(x, ...)
```

Arguments

x	An object of class "sboa".
...	Additional graphical arguments passed to graphics::plot().

Value

No return value. Called for its side effect.

plot.sboa_mlp *Plot method for SBOA-MLP objects*

Description

Plots the convergence curve of a trained SBOA-MLP model.

Usage

```
## S3 method for class 'sboa_mlp'
plot(x, ...)
```

Arguments

x	An object of class "sboa_mlp".
...	Additional graphical arguments passed to graphics::plot().

Value

No return value. Called for its side effect.

predict.sboa_mlp *Predict method for SBOA-MLP objects*

Description

Generates predictions from a trained SBOA-MLP model.

Usage

```
## S3 method for class 'sboa_mlp'
predict(object, newdata, ...)
```

Arguments

object	An object of class "sboa_mlp".
newdata	New predictor data.
...	Additional arguments, ignored.

Value

A matrix of predicted values.

print.sboa *Print method for SBOA objects*

Description

Prints a summary of an SBOA optimization result.

Usage

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

Arguments

x	An object of class "sboa".
...	Additional arguments, ignored.

Value

The input object, invisibly.

print.sboa_mlp *Print method for SBOA-MLP objects*

Description

Prints a summary of a trained SBOA-MLP model.

Usage

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

Arguments

x	An object of class "sboa_mlp".
...	Additional arguments, ignored.

Value

The input object, invisibly.

rmse_vec *Root mean squared error*

Description

Computes the root mean squared error between actual and predicted values.

Usage

```
rmse_vec(actual, predicted)
```

Arguments

actual	Numeric vector or matrix of observed values.
predicted	Numeric vector or matrix of predicted values.

Value

A numeric scalar.

rsq_vec *R-squared*

Description

Computes the coefficient of determination.

Usage

```
rsq_vec(actual, predicted)
```

Arguments

actual	Numeric vector or matrix of observed values.
predicted	Numeric vector or matrix of predicted values.

Value

A numeric scalar.

sboa *Secretary Bird Optimization Algorithm*

Description

General-purpose continuous optimization using the Secretary Bird Optimization Algorithm (SBOA).

Usage

```
sboa(
  fn,
  lower,
  upper,
  n_agents = 30,
  max_iter = 500,
  ...,
  verbose = TRUE,
  seed = NULL
)
```

Arguments

fn	Objective function to be minimized, or a character string naming a built-in benchmark function such as "F1".
lower	Lower bounds for decision variables.
upper	Upper bounds for decision variables.
n_agents	Number of search agents.
max_iter	Maximum number of iterations.
...	Additional arguments passed to fn.
verbose	Logical; if TRUE, progress is printed.
seed	Optional random seed.

Value

An object of class "sboa".

References

Fu, W., Wang, K., Liu, J., et al. (2024). Secretary Bird Optimization Algorithm. Artificial Intelligence Review. DOI: 10.1007/s10462-024-10729-y

Examples

```
sphere <- function(x) sum(x^2)

res <- sboa(
  fn = sphere,
  lower = rep(-10, 5),
  upper = rep(10, 5),
  n_agents = 10,
  max_iter = 20,
  seed = 123,
  verbose = FALSE
)

res2 <- sboa(
  fn = "F1",
  lower = rep(-100, 5),
  upper = rep(100, 5),
  n_agents = 10,
  max_iter = 20,
  seed = 123,
  verbose = FALSE
)

print(res)
print(res2)
list_benchmarks()
get_benchmark("F9")
```

`sboa_mlp` *Train a single-hidden-layer MLP using SBOA*

Description

Trains a single-hidden-layer multilayer perceptron with the Secretary Bird Optimization Algorithm.

Usage

```
sboa_mlp(
  X_train,
  y_train,
  hidden_dim = 10,
  n_agents = 30,
  max_iter = 500,
  lower = -1,
  upper = 1,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`X_train`	Training input data.
`y_train`	Training output data.
`hidden_dim`	Number of hidden neurons.
`n_agents`	Number of search agents.
`max_iter`	Maximum number of iterations.
`lower`	Lower bound for parameter search.
`upper`	Upper bound for parameter search.
`seed`	Optional random seed.
`verbose`	Logical; if `TRUE`, progress is printed.

Value

An object of class `"sboa_mlp"`.

References

Dilber, B., and Ozdemir, A. F. (2026). A novel approach to training feed-forward multi-layer perceptrons with recently proposed secretary bird optimization algorithm. Neural Computing and Applications. DOI: 10.1007/s00521-026-11874-x

Examples

```
set.seed(123)

X_train <- matrix(runif(40), nrow = 10, ncol = 4)
y_train <- matrix(runif(10), nrow = 10, ncol = 1)

fit <- sboa_mlp(
  X_train = X_train,
  y_train = y_train,
  hidden_dim = 3,
  n_agents = 10,
  max_iter = 20,
  lower = -1,
  upper = 1,
  seed = 123,
  verbose = FALSE
)

print(fit)
pred <- predict(fit, X_train)
head(pred)
```

---

sigmoid *Sigmoid activation function*

Description

Computes the sigmoid transformation.

Usage

```
sigmoid(x)
```

Arguments

x	A numeric vector, matrix, or scalar.

Value

Transformed values in the interval (0, 1).

unpack_mlp_params *Unpack MLP parameter vector*

Description

Converts a flat parameter vector into weight matrices and bias vectors for a single-hidden-layer multilayer perceptron.

Usage

```
unpack_mlp_params(params, input_dim, hidden_dim, output_dim)
```

Arguments

params	Numeric parameter vector.
input_dim	Number of input variables.
hidden_dim	Number of hidden neurons.
output_dim	Number of output neurons.

Value

A list containing W1, b1, W2, and b2.

Index