Package ‘BORT’

July 7, 2026

Type Package

Title Beyond Pareto: Bi-Objective and Multi-Objective Regression
Trees’

Version 0.1.0

Author Erick G.G. de Paz [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7878-8238>),
Arturo Hernández-Aguirre [aut] (ORCID:
<https://orcid.org/0000-0002-3744-9827>),
Iván Cruz-Aceves [aut] (ORCID: <https://orcid.org/0000-0002-5197-2059>)

Maintainer Erick G.G. de Paz <erick.giles@cimat.mx>

Description Implements the Bi-objective Regression Tree (BORT) for efficiently
learning vector-valued functions. Unlike traditional methods that rely on
constructing multiple models or static scalarisation, BORT integrates the
exploration of the Pareto front directly into a single tree's growth process.
It provides high-efficiency, single-model approaches that can Pareto-dominate
entire Pareto-consistent families of trees, supported by a C backend for
fast computation. For more details see
Paz (2026) <doi:10.1007/978-3-032-28393-1_2> and
Paz (2025) <doi:10.1007/978-3-031-78401-9_2>.

License GPL-2

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date 2026-07-01

Depends R (>= 2.10.0)

Date/Publication 2026-07-07 09:50:08 UTC

Contents

bort *BORT: Multi-objective Regression Trees*

Description

Constructs a multiobjective regression tree or a Pareto-consistent family of trees based on a top-down generalisation. The partitioning process selects the hyper-rectangle with the maximum Lebesgue measure. The split thresholds are chosen to minimise the sum of the Weighted Mean Squared Error across dimensions.

Usage

```
bort(X, Y, k = 1, type = c("PARETO", "BORT"), minSample = NULL)
```

Arguments

`X`	A numeric matrix of size $n \times p$ containing the input variables (features) for n samples.
`Y`	A numeric matrix of size $n \times q$ containing the target variables (objectives) to be predicted.
`k`	An integer scalar specifying the number of trees to generate. Default is 1.
`type`	A character string indicating the modelling strategy. `"PARETO"` generates a Pareto-consistent family of trees weighting the error via a single Dirichlet-sampled vector per tree. `"BORT"` updates the weight vector dynamically at each split iteration to adaptively explore the Pareto front.
`minSample`	An integer indicating the minimum number of samples a node must contain to be eligible for further splitting. If `NULL`, it defaults to 5% of n.

Details

This implementation maps continuous multiobjective functions $f : \mathbb{R}^p \rightarrow \mathbb{R}^q$. It partitions the Cartesian space D bounded by $[min(X[, i]) - \delta, max(X[, i]) + \delta]$, where $\delta = 0.1 * (max(X[, i]) - min(X[, i]))$. For `type = "PARETO"`, scalarisation of errors uses a constant weight vector across the tree depth. For `type = "BORT"`, a novel weighting approach selects random weights at every partition, achieving efficient single-model dominance over entirely consistent families. Both approaches are explained in de Paz (2025).

Value

A list of length `k` containing R functions. Each function accepts a numeric vector `x` of length p and returns a predicted numeric vector `y` of length q.

References

de Paz, E.G.G., Hernández-Aguirre, A., Cruz-Aceves, I. (2026). Beyond Pareto: A High-Efficiency Approach to Bi-objective Regression Trees. In *Pattern Recognition*. Springer Nature Switzerland. doi:10.1007/9783032283931_2

de Paz, E.G.G., Vaquera Huerta, H., Albores Velasco, F.J., Bauer Mengelberg, J.R., Romero Padilla, J.M. (2025). A Splitting Criterion for CART Models Based on Bayesian Optimisation. In *Statistics, Society and Environment*. Springer Nature Switzerland. doi:10.1007/9783031784019_2

Examples

```
# Ensure the C shared library is loaded before running
# dyn.load(paste0("multiobjective_tree", .Platform$dynlib.ext))

# Prepare the iris dataset
data(iris)
X <- as.matrix(iris[, 1:2]) # Sepal.Length, Sepal.Width
Y <- as.matrix(iris[, 3:4]) # Petal.Length, Petal.Width

# Generate a single BORT model
bort_models <- bort(X, Y, k = 1, type = "BORT")

# Predict for the first instance
test_point <- X[1, ]
prediction <- bort_models[[1]](test_point)
print(prediction)

# Generate a Pareto-consistent family of 5 trees
pareto_models <- bort(X, Y, k = 5, type = "PARETO")
pred_pareto <- pareto_models[[1]](test_point)
print(pred_pareto)
```

Index