Package 'boostPM'

July 26, 2026

Type Package

Title Unsupervised Tree Boosting for Learning Probability Distributions

Version 0.1.0

Description Implements the unsupervised tree boosting method for learning probability distributions introduced by Awaya and Ma (2024). Provides model fitting, density evaluation, simulation, and diagnostic methods.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/nawaya040/boostPM-cran

BugReports https://github.com/nawaya040/boostPM-cran/issues

Imports graphics, Rcpp (>= 1.0.7), stats

LinkingTo Rcpp, RcppArmadillo

Suggests testthat (>= 3.0.0), knitr (>= 1.43), rmarkdown (>= 2.20)

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.2

NeedsCompilation yes

Author Naoki Awaya [aut, cre]

Maintainer Naoki Awaya <nawaya@waseda.jp>

Repository CRAN

Date/Publication 2026-07-26 10:30:02 UTC

Contents

`boostPM-package` *boostPM: Unsupervised Tree Boosting for Learning Probability Distributions*

Description

Implements the unsupervised tree boosting method of Awaya and Ma (2024) for fitting tree-ensemble probability distributions, evaluating fitted densities, and generating samples.

Author(s)

Maintainer: Naoki Awaya <nawaya@waseda.jp>

References

Awaya, N. and Ma, L. (2024). Unsupervised Tree Boosting for Learning Probability Distributions. *Journal of Machine Learning Research*, 25, 1–52.

See Also

`fit_boostpm()`, `predict.boostPM_fit()`, `simulate.boostPM_fit()`, and `plot.boostPM_fit()`.

`fit_boostpm` *Fit a boostPM Distribution*

Description

Fits the unsupervised tree boosting procedure of Awaya and Ma (2024). Marginal trees are fitted first, followed by dependence trees. The fitted ensemble represents a probability distribution on a rectangular support.

Usage

```
fit_boostpm(
  data,
  Omega = NULL,
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 8,
  max_split_depth = 15,
```

```
  min_node_observations = 5,
  c0 = 0.1,
  gamma = 0.1,
  early_stop = NULL,
  prior_split_prob = 0.9,
  add_noise = TRUE,
  progress = c("none", "stage")
)
```

Arguments

`data`	A finite numeric matrix with observations in rows and variables in columns. Missing, infinite, and constant columns are not supported.
`Omega`	Either `NULL`, or a finite numeric matrix with one row per variable and two columns giving lower and upper support limits. Training observations must lie strictly inside supplied limits. If `NULL`, the processed range of each variable is expanded by 10 percent on both sides.
`max_marginal_trees`	Non-negative integer giving the maximum number of trees fitted for each marginal distribution. Set to zero to skip marginal fitting.
`max_dependence_trees`	Non-negative integer giving the maximum number of trees fitted to the dependence structure after marginal fitting. Set to zero to skip this stage.
`n_bins`	Integer of at least two. Candidate split fractions are `1 / n_bins, ..., (n_bins - 1) / n_bins`, so the number of interior candidates is `n_bins - 1`. Larger values provide a finer search at greater computational cost.
`max_split_depth`	Non-negative integer giving the deepest node depth that remains eligible for splitting. Consequently, terminal leaves may occur at depth `max_split_depth + 1`.
`min_node_observations`	Positive integer. A node containing fewer than `min_node_observations` fitting observations is not split; a node containing exactly `min_node_observations` observations remains eligible.
`c0`	Global learning rate, strictly between zero and one. Larger values move each fitted node mass more strongly toward the empirical residual distribution and therefore apply less shrinkage.
`gamma`	Non-negative scale-specific learning-rate exponent. Zero gives the constant learning rate `c0`; larger values reduce the learning rate more strongly in small-volume nodes.
`early_stop`	Either `NULL`, or `c(threshold, window)`, where `threshold` is finite and `window` is an integer of at least two. See **Adaptive stopping** for the held-out evaluation rule.
`prior_split_prob`	Numeric value in `[0, 1]` giving the prior probability of splitting a node before candidate marginal likelihoods are considered. It is constant across depths and is not the posterior or realized split frequency.

`add_noise` A single logical value. If `TRUE`, tied observations are perturbed before fitting using R's random-number generator and spacings between adjacent distinct values.

`progress` Character string controlling progress messages. `"none"` suppresses progress output and `"stage"` reports each marginal fitting stage and the dependence stage.

Value

An object of class `boostPM_fit`, represented as a list containing:

trees Serialized fitted trees used by package methods. This is an internal representation and is not a stable interface for direct editing.

residual_coordinates Final residual coordinates as a variables-by-observations numeric matrix. This advanced diagnostic is not intended for direct modification.

tree_diagnostics A data frame with one row per accepted tree and columns `tree_index`, `stage`, `variable`, `node_count`, and `max_depth`. `stage` is `"marginal"` or `"dependence"`; `variable` identifies the marginal variable and is `NA` for dependence trees.

variable_importance Unnormalized accumulated splitting improvement attributed to each variable. Larger values indicate greater contribution within the fitted ensemble. The vector is named when `data` has column names.

heldout_diagnostics A data frame with one row per held-out candidate and columns `candidate_index`, `stage`, `variable`, `mean_log_density_improvement`, and `accepted`, or `NULL` when `early_stop = NULL`. The candidate that triggers stopping has `accepted = FALSE`.

support A variables-by-two matrix containing named `lower` and `upper` original-scale support limits. Row names follow the training variable names when available.

elapsed_time Elapsed fitting time as a `difftime` object.

call The matched call used to fit the model.

control A named list of the validated controls actually used: `max_marginal_trees`, `max_dependence_trees`, `n_bins`, `max_split_depth`, `min_node_observations`, `c0`, `gamma`, `early_stop`, `prior_split_prob`, `add_noise`, and `progress`.

Fitting stages

The procedure first fits up to `max_marginal_trees` univariate trees for each variable. It then fits up to `max_dependence_trees` multivariate trees to the remaining dependence structure. These arguments are upper bounds; adaptive stopping can end a stage earlier.

Scale-specific learning rate

For a node A in the normalized unit cube, the learning rate is

$$c(A) = c_0(1 - \log_2[\mathrm{vol}(A)])^{-\gamma}.$$

Thus `c0` determines the global update size. When `gamma = 0`, every node uses `c0`. When `gamma > 0`, smaller-volume nodes receive smaller updates, which imposes stronger shrinkage on local structure.

Adaptive stopping

If `early_stop = NULL`, all observations fit each candidate tree and no adaptive stopping is applied. With `early_stop = c(threshold, window)`, each candidate tree is fitted to a random 90 percent subset of the current residuals and evaluated by its average log density on the held-out 10 percent. A stage ends when the mean of the most recent `window` held-out scores is strictly below `threshold`. The candidate that triggers stopping is recorded in `heldout_diagnostics` with `accepted = FALSE` but is not added to the ensemble. `c(0, 50)` is the paper-oriented setting; the implementation uses a strict comparison with zero, whereas the paper describes non-positive improvement.

Support and tied observations

Data are transformed from `Omega` to the unit cube before fitting. If `Omega = NULL`, the range after optional tie perturbation is expanded by 10 percent on each side. Set `add_noise = TRUE` when a continuous distribution is assumed and ties are technical artifacts such as rounding. Set it to `FALSE` when perturbing ties would be inappropriate.

Split prior

The default `prior_split_prob = 0.9` follows the archived implementation and public experiment code. Setting `prior_split_prob = 0.5` corresponds to the constant split prior implied by the 0.5 stopping probability reported in Appendix C of Awaya and Ma (2024). Larger values favor splitting before the data-dependent candidate likelihoods are incorporated.

Reproducibility and progress

Tree construction is stochastic even when `add_noise = FALSE`, and adaptive stopping also draws random fitting subsets. Call `set.seed()` before every fit that must be reproducible. Fitting is silent by default; set `progress = "stage"` for stage-level messages. Elapsed time is stored in the result and can be inspected with `print()` or `summary()`. Exact equality across operating systems has not been established.

References

Awaya, N. and Ma, L. (2024). Unsupervised Tree Boosting for Learning Probability Distributions. *Journal of Machine Learning Research*, 25, 1–52.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x2 <- stats::rbeta(n, shape1 = 2 + 6 * x1, shape2 = 4)
x <- cbind(x1 = x1, x2 = x2)
support <- cbind(lower = c(0, 0), upper = c(1, 1))

set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = support,
  max_marginal_trees = 100,
```

```
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
  add_noise = FALSE
)
print(fit)

grid <- seq(0.01, 0.99, length.out = 50L)
evaluation_grid <- as.matrix(expand.grid(x1 = grid, x2 = grid))
fitted_density <- matrix(
  predict(fit, newdata = evaluation_grid, type = "density"),
  nrow = length(grid)
)
graphics::image(grid, grid, fitted_density, xlab = "x1", ylab = "x2")
graphics::contour(
  grid, grid, fitted_density, add = TRUE, drawlabels = FALSE
)

generated <- simulate(fit, nsim = 500, seed = 321)
graphics::plot(
  generated, xlab = "x1", ylab = "x2", main = "Generated observations"
)
```

`plot.boostPM_fit` *Plot Diagnostics for a Fitted boostPM Distribution*

Description

Displays a bar plot of variable importance, the number of nodes per tree, or maximum tree depths.

Usage

```
## S3 method for class 'boostPM_fit'
plot(
  x,
  type = c("variable_importance", "tree_node_counts", "tree_depths"),
  ...
)
```

Arguments

`x`	A fitted object returned by `fit_boostpm()`.
`type`	Character string selecting "variable_importance", "tree_node_counts", or "tree_depths".
`...`	Additional arguments passed to `graphics::barplot()`. User values for `names.arg` and `ylab` override the method defaults. `height` cannot be supplied because it is determined by the selected diagnostic.

Value

The bar midpoints returned by `graphics::barplot()`, invisibly.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x <- cbind(x1 = x1, x2 = stats::rbeta(n, 2 + 6 * x1, 4))
set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = cbind(lower = c(0, 0), upper = c(1, 1)),
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
  add_noise = FALSE
)
plot(fit, type = "variable_importance")
plot(fit, type = "tree_node_counts")
plot(fit, type = "tree_depths")
```

---

predict.boostPM_fit *Evaluate a Fitted boostPM Density*

Description

Evaluates the probability density represented by a fitted tree ensemble at supplied points.

Usage

```
## S3 method for class 'boostPM_fit'
predict(object, newdata, type = c("log_density", "density", "details"), ...)
```

Arguments

`object`	A fitted object returned by `fit_boostpm()`.
`newdata`	A finite numeric matrix with evaluation points in rows and one column per fitted variable.
`type`	Character string selecting the return value: `"log_density"` returns log densities, `"density"` returns densities, and `"details"` returns a diagnostic list containing log densities and the cumulative mean log-density path across the fitted trees.
`...`	Unused. Additional arguments are an error.

Details

Points outside the fitted rectangular support receive log density `-Inf` and density zero. At a split point, evaluation follows the left-child convention in the method specification. Because `mean_log_density_path` averages over every row of `newdata`, the entire path is `-Inf` when any row lies outside the support.

Value

For `type = "log_density"` or `type = "density"`, a numeric vector with one entry per row of `newdata`, in the original row order. For `type = "details"`, a list containing:

log_density The same vector returned by `type = "log_density"`.

mean_log_density_path A numeric vector with one entry per fitted tree. Entry `k` is the mean log density over all rows of `newdata` after the first `k` trees have been applied.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x <- cbind(x1 = x1, x2 = stats::rbeta(n, 2 + 6 * x1, 4))
set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = cbind(lower = c(0, 0), upper = c(1, 1)),
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
  add_noise = FALSE
)

grid <- seq(0.01, 0.99, length.out = 50L)
evaluation_grid <- as.matrix(expand.grid(x1 = grid, x2 = grid))
fitted_density <- matrix(
  predict(fit, evaluation_grid, type = "density"),
  nrow = length(grid)
)
graphics::image(grid, grid, fitted_density, xlab = "x1", ylab = "x2")
graphics::contour(
  grid, grid, fitted_density, add = TRUE, drawlabels = FALSE
)
```

`print.boostPM_fit` *Print a Fitted boostPM Distribution*

Description

Prints a compact description of a boostPM_fit object.

Usage

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

Arguments

`x`	A fitted object returned by `fit_boostpm()`.
`...`	Unused.

Value

The input object, invisibly.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x <- cbind(x1 = x1, x2 = stats::rbeta(n, 2 + 6 * x1, 4))
set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = cbind(lower = c(0, 0), upper = c(1, 1)),
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
  add_noise = FALSE
)
print(fit)
```

`print.summary.boostPM_fit` *Print a boostPM Fit Summary*

Description

Prints the fitted call, data dimensions, support, structural tree diagnostics, variable importance, and elapsed fitting time.

Usage

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

Arguments

`x`	An object returned by `summary.boostPM_fit()`.
`...`	Unused. Additional arguments are an error.

Value

The input summary object, invisibly.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x <- cbind(x1 = x1, x2 = stats::rbeta(n, 2 + 6 * x1, 4))
set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = cbind(lower = c(0, 0), upper = c(1, 1)),
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
  add_noise = FALSE
)
print(summary(fit))
```

`simulate.boostPM_fit` *Simulate from a Fitted boostPM Distribution*

Description

Draws samples from the probability distribution represented by a fitted tree ensemble.

Usage

```
## S3 method for class 'boostPM_fit'
simulate(object, nsim = 1L, seed = NULL, ...)
```

Arguments

`object`	A fitted object returned by `fit_boostpm()`.
`nsim`	Non-negative integer. Number of observations to simulate.
`seed`	Either `NULL`, or a non-negative integer used to set R's random number generator immediately before simulation.
`...`	Unused. Additional arguments are an error.

Details

With `seed = NULL`, randomness is controlled by calling `set.seed()` before `simulate()`. Supplying `seed` sets R's random-number generator and changes its subsequent state.

Value

A numeric matrix with `nsim` rows and one column per fitted variable, in the same order as the training data. Training variable names are preserved as column names when available. Every draw lies within the fitted rectangular support.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x <- cbind(x1 = x1, x2 = stats::rbeta(n, 2 + 6 * x1, 4))
set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = cbind(lower = c(0, 0), upper = c(1, 1)),
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
```

```
  add_noise = FALSE
)
generated <- simulate(fit, nsim = 500, seed = 321)

old_par <- graphics::par(mfrow = c(1, 2))
graphics::plot(x, xlab = "x1", ylab = "x2", main = "Training data")
graphics::plot(
  generated,
  xlab = "x1",
  ylab = "x2",
  main = "Generated data"
)
graphics::par(old_par)
```

`summary.boostPM_fit` *Summarize a Fitted boostPM Distribution*

Description

Extracts compact structural diagnostics from a boostPM_fit object.

Usage

```
## S3 method for class 'boostPM_fit'
summary(object, ...)
```

Arguments

`object`	A fitted object returned by `fit_boostpm()`.
`...`	Unused.

Value

An object of class `summary.boostPM_fit`, represented as a list containing:

call The matched call used to fit the model.

n_observations Number of training observations, or `NA` when it cannot be recovered from the fitted object.

n_variables Number of fitted variables.

n_trees Number of accepted trees.

support The original-scale lower and upper support limits.

tree_diagnostics The per-tree diagnostic data frame returned by `fit_boostpm()`.

variable_importance Accumulated improvement attributed to each variable.

elapsed_time Elapsed fitting time as a `difftime` object.

Examples

```
set.seed(42)
n <- 400L
x1 <- stats::rbeta(n, shape1 = 2, shape2 = 5)
x <- cbind(x1 = x1, x2 = stats::rbeta(n, 2 + 6 * x1, 4))
set.seed(123)
fit <- fit_boostpm(
  x,
  Omega = cbind(lower = c(0, 0), upper = c(1, 1)),
  max_marginal_trees = 100,
  max_dependence_trees = 1000,
  n_bins = 100,
  max_split_depth = 15,
  min_node_observations = 10,
  c0 = 0.1,
  gamma = 0.5,
  add_noise = FALSE
)
summary(fit)
```

Index