Package 'HNPclassifier'

July 14, 2026

Title Hierarchical Neyman-Pearson Classification for Ordered Classes

Version 0.2.1

Description The Hierarchical Neyman-Pearson (H-NP) classification framework extends the Neyman-Pearson classification paradigm to multi-class settings where classes have a natural priority ordering. This is particularly useful for classification in unbalanced dataset, for example, disease severity classification, where under-classification errors (misclassifying patients into less severe categories) are more consequential than other misclassifications. The package implements H-NP umbrella algorithms that controls under-classification errors under user specified control levels with high probability. It supports the creation of H-NP classifiers using scoring functions based on built-in classification methods (including logistic regression, support vector machines, and random forests), as well as user-trained scoring functions. The package exports `base_function()` to train these built-in base learners directly for use in the H-NP pipeline.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, e1071, MASS, nnet, randomForest

NeedsCompilation no

Author Che Shen [aut, cre] (Implementation and maintenance),
Lujia Yang [aut] (Testing and debugging),
Lijia Wang [aut] (Original theory and supervision),
Shunan Yao [aut] (Supervision and debugging)

Maintainer Che Shen <chshen3-c@my.cityu.edu.hk>

Repository CRAN

Date/Publication 2026-07-14 07:00:02 UTC

Contents

`base_function` *Built-in Base Learner for H-NP Classification*

Description

Train a multi-class scoring classifier using one of the built-in methods (random forest, support vector machine, or logistic regression). The returned fitted model can be used directly or passed to `hnp_umbrella()` via the `pretrained_model` argument.

Usage

```
base_function(x, y, method = "randomforest")
```

Arguments

`x`	A data.frame or matrix of predictors/features.
`y`	A vector or factor of class labels of length `nrow(x)`, with at least two classes. When used inside the H-NP pipeline, labels should already be internal levels `"1"`, ..., `"T"`.
`method`	Character string: one of `"randomforest"`, `"svm"`, or `"logistic"`.

Value

A fitted model object (`randomForest::randomForest`, `e1071::svm`, or `nnet::multinom`, depending on `method`).

See Also

`hnp_umbrella()`, `train_nn_and_get_scores()`

Examples

```
set.seed(123)
X <- data.frame(x1 = rnorm(60), x2 = rnorm(60))
Y <- factor(sample(c("1", "2", "3"), 60, replace = TRUE))
fit_rf <- base_function(X, Y, method = "randomforest")
fit_svm <- base_function(X, Y, method = "svm")
fit_logistic <- base_function(X, Y, method = "logistic")
```

generate_ball_data *Generate Synthetic Three-class Ball Data*

Description

Generate a synthetic three-class dataset in which each class is sampled uniformly from a 3-dimensional ball with a given center and radius. Useful for demonstrating and testing the HNP Umbrella pipeline.

Usage

```
generate_ball_data(n, centers, radii)
```

Arguments

n	Integer. Number of samples to draw for each class.
centers	A list of three numeric vectors of length 3, giving the center coordinates of the balls for classes "A", "B", and "C".
radii	Numeric vector of length 3 giving the radius of each class ball.

Value

A data.frame with feature columns x1, x2, x3 and a factor column y with levels c("A","B","C"), containing 3 * n rows.

Examples

```
set.seed(123)
data <- generate_ball_data(
  n = 100,
  centers = list(c(0, 0, 0), c(2, 2, 2), c(4, 0, 0)),
  radii = c(1, 1, 1)
)
```

hnp_boxplot *HNP Under-classification Error Box Plot*

Description

Summarize and visualize the under-classification errors and the overall misclassification error across repeated runs for a T-class problem. Given a list of confusion matrices (one per run), it draws grouped boxplots of the class-wise under-classification errors and the overall error. When a second list of confusion matrices is supplied, the two methods (e.g. a classical classifier vs. H-NP) are compared side by side. Optional control levels and tolerances add reference lines and (1 - delta) quantile markers.

Usage

```
hnp_boxplot(
  conf_1,
  conf_2 = NULL,
  levels = NULL,
  tolerances = NULL,
  name_1 = "Classical",
  name_2 = "H-NP"
)
```

Arguments

conf_1	A list of T x T confusion matrices (one per run) for the first method.
conf_2	Optional list of T x T confusion matrices (one per run) for the second method, with the same number of runs as conf_1. If NULL, only conf_1 is plotted (single mode).
levels	Optional numeric vector of length T - 1. Control levels (alpha) per class; drawn as dashed reference lines. Each must lie in (0, 1).
tolerances	Optional numeric vector of length T - 1. Tolerances (delta) per class; used to mark the (1 - delta) quantile as red points. Each must lie in (0, 1).
name_1	Character. Legend label for the first method.
name_2	Character. Legend label for the second method (used only when conf_2 is provided).

Value

Invisibly draws the boxplot and returns a list with two components: classwise (a data.frame of per-class control level, tolerance, under-classification error mean/sd, and violation rate) and overall (a data.frame of the overall misclassification error mean and sd).

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
make_cm <- function() {
  m <- matrix(sample(5:20, 9, replace = TRUE), 3, 3)
  dimnames(m) <- list(c("1", "2", "3"), c("1", "2", "3"))
  m
}
conf_classical <- replicate(20, make_cm(), simplify = FALSE)
conf_hnp <- replicate(20, make_cm(), simplify = FALSE)
tab <- hnp_boxplot(
  conf_1 = conf_classical,
```

```
  conf_2 = conf_hnp,
  levels = c(0.1, 0.1),
  tolerances = c(0.1, 0.1)
)
```

`hnp_map_classes` *Classes Mapping function for HNP Algorithm Map ordered class labels to internal levels "1", ..., "T"*

Description

Validate the class column and re-label the provided class names to internal factor levels "1", "2", ..., "T". The input order of the labels in `...` is treated as the importance order: the first label maps to level "1" (highest priority), and the last label maps to the level "T" (lowest priority). Supports any number of classes `T >= 2`.

Usage

```
hnp_map_classes(data, class_col, ...)
```

Arguments

`data`	A data.frame or data.table containing the dataset.
`class_col`	Character scalar. Name of the class/label column in `data`.
`...`	Two or more original class labels given in importance order. The first label maps to "1", the second to "2", and so on. Labels must be non-empty, non-NA, and unique.

Value

A data.frame with the input `data` added with `class_col` converted to a factor whose levels are the internal labels c("1","2",...,"T").

Examples

```
df <- data.frame(y = c("low", "mid", "high", "mid"), x1 = rnorm(4))
df2 <- hnp_map_classes(df, class_col = "y", "high", "mid", "low")
```

hnp_summary *HNP Classifier's Performance Summary*

Description

Evaluate a T-class classifier produced by the HNP pipeline (or any compatible model/function). Given features and true labels, it computes the confusion matrix, class-wise false positive/negative rates, under- and over-classification errors, overall accuracy, and a row-normalized error table. The class priority order is resolved from `importance_order`, from attributes attached to the `classifier`, or inferred from the labels.

Usage

```
hnp_summary(classifier, X, Y, importance_order = NULL)
```

Arguments

`classifier`	Either a function `function(X) ...` returning class labels or an `n x T` score/probability matrix, or a fitted model object (e.g. randomForest, support vector machine, multinomial logistic regression) from which predictions can be derived.
`X`	A data.frame of features (or a score matrix) to predict on.
`Y`	A vector of true class labels of length `nrow(X)`.
`importance_order`	Optional character vector giving the class priority order (most severe first). If `NULL`, the order is taken from the classifier's attributes or inferred from `Y`.

Value

A list with components of the HNP experiment results including `confusion_matrix`, `false_positive_rate`, `false_negative_rate`, `overall_accuracy`, `predictions`, `predictions_sample`, `under_classification_error`, `over_classification_error`, `total_under_classification_error`, `total_over_classification_error`, `error_table`, `class_levels`, `internal_levels`, `label_mapping`, `importance_order`, and `order_source`.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
n <- 300
X <- data.frame(x1 = rnorm(n), x2 = rnorm(n))
Y <- sample(c("A", "B", "C"), n, replace = TRUE)
clf <- hnp_umbrella(
  X = X, Y = Y,
```

```
  levels = c(0.1, 0.1),
  tolerances = c(0.1, 0.1),
  importance_order = c("A", "B", "C"),
  method = "logistic"
)
result <- hnp_summary(clf, X = X, Y = Y, importance_order = c("A", "B", "C"))
```

---

hnp_umbrella *HNP Umbrella Algorithm*

Description

Implementation of the Hierarchical Neyman-Pearson (HNP) Umbrella algorithm for general T-class classification (T >= 2). Classes are ordered by importance_order and re-labelled internally to the classes "1", ..., "T". The algorithm trains (or reuses) a scoring-type classifier (e.g. random forest, support vector machine, or logistic regression) and selects thresholds that control the under-classification error of the more severe classes at the requested levels with the given tolerances, either by treating upper bounds as thresholds or by a grid search that minimizes the weighted remaining error.

Usage

```
hnp_umbrella(
  X,
  Y,
  levels,
  tolerances,
  importance_order,
  method = "logistic",
  pretrained_model = NULL,
  input_is_score = FALSE,
  grid_search = TRUE,
  grid_set = NULL,
  max_grid = 15,
  max_combinations = 2000,
  hnp_split = NULL,
  verbose = FALSE
)
```

Arguments

X	A data.frame of features, or an n x T score matrix when input_is_score = TRUE.
Y	A vector of class labels of length nrow(X). All labels must be covered by importance_order.
levels	Numeric vector of length T - 1. Control levels (alpha) for the under-classification error of classes 1, ..., T-1. Each must lie in (0, 1).

`tolerances` Numeric vector of length `T - 1`. Tolerance (delta) values for the confidence bounds. Each must lie in (0, 1).

`importance_order` Character vector giving the class priority order, most severe first. Defines the mapping to internal levels "1", ..., "T".

`method` Character string: one of 'randomforest', 'svm', or 'logistic'. The base method used when training a scoring-type classifier internally.

`pretrained_model` Optional. A user-supplied scoring model: a fitted model object, a function returning an `n x T` score/probability matrix, or a list containing `score_functions`. When provided, no internal model is trained.

`input_is_score` Logical. If `TRUE`, `X` is treated as a precomputed `n x T` score matrix and `pretrained_model` is ignored.

`grid_search` Logical. If `TRUE`, search candidate thresholds to minimize the weighted remaining error; if `FALSE`, use the upper-bound as thresholds directly.

`grid_set` Optional list of length `T - 1` of candidate threshold vectors per class. If `NULL`, candidates are derived from the threshold set.

`max_grid` Integer. Maximum number of candidate thresholds considered per class in the grid search.

`max_combinations` Integer. Maximum number of threshold combinations explored in the grid search.

`hnp_split` Optional list of length `T` specifying the class-wise data split ratios (train / threshold / error). If `NULL`, sensible defaults are used based on `grid_search` and whether scores are already available.

`verbose` Logical. If `TRUE`, print progress information.

Value

A HNP Umbrella classifier function `function(new_data, output_internal = FALSE)` that returns predicted class labels (original labels by default, internal labels if `output_internal = TRUE`), with attributes recording the selected thresholds, objective value, `importance_order`, label mapping, and other metadata. Returns `NULL` if no feasible classifier is found.

References

Lijia Wang, Y. X. Rachel Wang, Jingyi Jessica Li, and Xin Tong (2024). "Hierarchical Neyman-Pearson Classification for Prioritizing Severe Disease Categories in COVID-19 Patient Data." *Journal of the American Statistical Association*, 119(545), 39-51. doi:10.1080/01621459.2023.2270657

Examples

```
set.seed(123)
n <- 500
X <- data.frame(x1 = rnorm(n), x2 = rnorm(n))
Y <- sample(c("A", "B", "C"), n, replace = TRUE, prob = c(0.2, 0.3, 0.5))
clf <- hnp_umbrella(
  X = X, Y = Y,
```

```
  levels = c(0.1, 0.1),
  tolerances = c(0.1, 0.1),
  importance_order = c("A", "B", "C"),
  method = "logistic"
)
preds <- clf(X)
```

`train_nn_and_get_scores` *Neural Network Classifier Helper Function*

Description

Fit a single-hidden-layer softmax neural network (`nnet::nnet`) for multi-class classification and return the trained model together with the metadata needed to produce class scores. This can be used to supply a pretrained scoring model to the HNP Umbrella pipeline.

Usage

```
train_nn_and_get_scores(X, Y)
```

Arguments

`X`	A data.frame or matrix of predictors/features.
`Y`	A vector or factor of class labels of length `nrow(X)`, with at least two classes.

Value

A list with components: `model` (the fitted `nnet` object), `class_levels` (the ordered class labels), and `feature_names` (the column names of `X`).

Examples

```
set.seed(123)
X <- data.frame(x1 = rnorm(60), x2 = rnorm(60))
Y <- factor(sample(c("1", "2", "3"), 60, replace = TRUE))
fit <- train_nn_and_get_scores(X, Y)
```

Index