Package: hetNSC (via r-universe)

November 7, 2026

Type Package

Title Heteroscedastic Nearest Shrunken Centroid Classification

Version 0.1.0

Description Nearest shrunken centroid (NSC) classifiers for high-dimensional, low-sample-size data such as gene expression. Implements the original homoscedastic NSC with soft thresholding (Prediction Analysis of Microarrays; Tibshirani et al. 2002 <doi:10.1073/pnas.082099299>), its hard- and order-thresholding versions (Sahtout, Wang and Ghimire 2024 <doi:10.1080/03610918.2022.2047201>), and the heteroscedastic extension of Sahtout and Wang that uses class-specific variances in both the test statistic and the discriminant function (methods STh3, HTh3 and OTh3). The thresholding parameter is selected by a refined (``deep'') cross-validation search. Also provides the simulation design, per-class performance metrics and the relative-heterogeneity diagnostic used in the paper.

License GPL-2

Encoding UTF-8

Depends R (>= 3.5.0)

Imports MASS, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, pamr

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.1

NeedsCompilation no

Author Haiyan Wang [aut, cre], Mohammad Omar Sahtout [aut], Trevor Hastie [ctb] (Author of the 'pamr' code this package is derived from), Robert Tibshirani [ctb] (Author of the 'pamr' code this package is derived from), Balasubramanian Narasimhan [ctb] (Author of the 'pamr' code this package is derived from), Gilbert Chu [ctb] (Author of the 'pamr' code this package is derived from)

Maintainer Haiyan Wang <hwang@ksu.edu>

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

Date/Publication 2026-11-07 21:00:37 UTC

RemoteUrl https://github.com/cran/hetNSC

RemoteRef HEAD

RemoteSha 4f40de6124a32cad636fff3c87cdf40067e7d0a8

Contents

balanced_folds . . . 2
compare_nsc . . . 3
compute_rho2 . . . 4
hnsc . . . 5
hnsc_cv . . . 6
hnsc_methods . . . 7
hnsc_select . . . 8
hnsc_train . . . 9
make_cm . . . 10
predict.hnsc . . . 11
simulate_hetero_data . . . 12
summary.hnsc_comparison . . . 14

Index **15**

`balanced_folds` *Class-balanced cross-validation folds*

Description

Splits sample indices into folds so that every class is spread as evenly as possible across folds. This is the fold generator of the **pamr** package, which is also used inside [hnsc_cv()].

Usage

```
balanced_folds(y, nfolds = min(min(table(y)), 10))
```

Arguments

`y`	Vector of class labels.
`nfolds`	Number of folds. Defaults to the size of the smallest class, capped at 10 (and never less than 2).

Value

A list of integer vectors of sample indices, one per fold.

Examples

```
set.seed(1)
balanced_folds(rep(c("A", "B"), c(6, 9)))
```

`compare_nsc` *Compare homoscedastic and heteroscedastic NSC classifiers on one dataset*

Description

Reproduces the evaluation protocol of the paper: for each thresholding rule, the homoscedastic classifier (STh/HTh/OTh) and its heteroscedastic counterpart (STh3/HTh3/OTh3) are fitted ‘n.runs‘ times, each time with the threshold chosen by [hnsc_select()] after a random reordering of the training samples (which changes the cross-validation folds). Each fit is evaluated on the test set.

Usage

```
compare_nsc(
  x,
  y,
  xtest,
  ytest,
  n.runs = 100,
  thresholding = c("hard", "soft", "order"),
  seed_fun = function(i) 1000 + 10 * i,
  file = NULL,
  verbose = TRUE
)
```

Arguments

`x, y`	Training data (genes x samples) and labels.
`xtest, ytest`	Test data (genes x samples) and labels.
`n.runs`	Number of repetitions (100 in the paper).
`thresholding`	Which thresholding rules to run, in this order.
`seed_fun`	Function of the run index returning the seed set before that run. The default, ‘function(i) 1000 + 10 * i‘, is the one used in the paper. Use ‘NULL‘ to not set seeds.
`file`	Optional path; if given the result is also saved with [saveRDS()].
`verbose`	Print progress.

Details

The training samples are re-shuffled cumulatively across runs and thresholding rules, exactly as in the script used for the paper, so that ‘compare_nsc()‘ reproduces those numbers.

Value

A list of class ‘"hnsc_comparison"‘ with * ‘error‘: mean test error rate of each method, * ‘ave_num_genes‘: mean number of genes used, * ‘errors‘, ‘n_genes‘: the per-run values (runs x methods matrices), * ‘cm_list‘: per method, a list of per-run confusion matrices, * ‘classes‘, ‘n‘ (training sample size) and ‘K‘.

Use [summary()] on the result for a table that includes balanced accuracy, and [recall_table()] for per-class recall.

Examples

```
set.seed(8900)
dat <- simulate_hetero_data(ntr1 = 15, ntr2 = 20, nte1 = 50, nte2 = 50,
                            p = 300, a = 15)
res <- compare_nsc(dat$xtr, dat$ytr, dat$xte, dat$yte, n.runs = 2,
                   verbose = FALSE)
summary(res)
```

`compute_rho2` *Empirical relative heterogeneity rho_k^2 and the crossover condition*

Description

Computes, for each class k, the empirical relative heterogeneity rho_k^2 = (sigma_k^2 / sigma_bar^2 - 1)^2 of Section 2.4 of the paper, where sigma_bar^2 = sum_k n_k sigma_k^2 / n, summarised across genes, and compares it with the simple crossover bound 2 / (n_k - 1) (obtained with w_k = 1 in eq. 2.11). When ‘crossover_met‘ is ‘TRUE‘ the class-specific variance estimate is expected to have lower mean squared error than the pooled one, i.e. the heteroscedastic classifiers are expected to help.

Usage

```
compute_rho2(
  X_train,
  y_train,
  X_test = X_train,
  y_test = y_train,
  summary_fn = stats::median
)
```

Arguments

`X_train`	Genes x samples matrix of training data.
`y_train`	Training class labels (length ‘ncol(X_train)‘).
`X_test, y_test`	Optional separate data used only to estimate the class variances (useful when training classes are very small). The weights n_k always come from the training labels.
`summary_fn`	Function used to summarise across genes (default [stats::median()]).

Value

A data frame with one row per class: ‘class‘, ‘n_k‘, ‘sigma2_hat‘, ‘sigma_bar2‘, ‘rho2_hat‘, ‘threshold‘ (= 2/(n_k - 1)) and ‘crossover_met‘.

Examples

```
set.seed(1)
dat <- simulate_hetero_data(ntr1 = 15, ntr2 = 20, nte1 = 5, nte2 = 5,
                            p = 500, a = 12)
compute_rho2(dat$xtr, dat$ytr)
```

---

hnsc *Fit a nearest shrunken centroid classifier with the threshold chosen by cross-validation*

Description

The main user-facing function. It selects the threshold with the refined cross-validation search of [hnsc_select()] (unless ‘threshold‘ is given) and then fits the classifier on all training data at that threshold. Use [predict.hnsc()] to classify new samples.

Usage

```
hnsc(
  x,
  y,
  method = c("hard", "soft", "order"),
  variance = c("hetero", "homo"),
  threshold = NULL,
  offset.percent = 50,
  prior = NULL,
  max.iter = 100
)
```

Arguments

x	Numeric matrix of expression values with **genes in rows and samples in columns** (the same layout as **pamr**).
y	Vector or factor of class labels, one per column of ‘x‘.
method	Thresholding rule: ‘"hard"‘, ‘"soft"‘ or ‘"order"‘. A paper label (‘"STh"‘, ‘"HTh"‘, ‘"OTh"‘, ‘"STh3"‘, ‘"HTh3"‘, ‘"OTh3"‘) is also accepted, in which case ‘variance‘ is ignored. See [hnsc_methods()].
variance	‘"hetero"‘ (class-specific variances, the proposed method) or ‘"homo"‘ (pooled variance). With ‘"homo"‘, ‘method = "soft"‘ is the original PAM classifier; ‘"hard"‘ and ‘"order"‘ are the homoscedastic classifiers of Sahtout, Wang and Ghimire (2024).

`threshold`	Optional single threshold. If ‘NULL‘ (default) it is chosen by [hnsc_select()].
`offset.percent`	Percentile of the standard deviations (homoscedastic) or of the class-specific variances (heteroscedastic) added as the stabilising constant s_0 (s_0k). Default 50, the median.
`prior`	Class prior probabilities. Defaults to the class proportions in the training data.
`max.iter`	Passed to [hnsc_select()].

Value

An object of class ‘"hnsc"‘ (see [hnsc_train()]) fitted at the selected threshold, with an extra component ‘selection‘ holding the [hnsc_select()] result (or ‘NULL‘ if ‘threshold‘ was supplied) and ‘selected.genes‘, the row indices of the genes used by the classifier.

References

Sahtout, M. O. and Wang, H. Heteroscedastic nearest shrunken centroid classification: methods and applications to high-dimensional data. *Statistical Applications in Genetics and Molecular Biology* (accepted).

Sahtout, M. O., Wang, H. and Ghimire, S. (2024). Different thresholding methods on nearest shrunken centroid algorithm. *Communications in Statistics - Simulation and Computation*, 53(3), 1444-1460. doi:10.1080/03610918.2022.2047201

Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G. (2002). Diagnosis of multiple cancer types by shrunken centroids of gene expression. *PNAS*, 99(10), 6567-6572.

Examples

```
set.seed(1)
dat <- simulate_hetero_data(ntr1 = 15, ntr2 = 20, nte1 = 50, nte2 = 50,
                            p = 500, a = 15)
fit <- hnsc(dat$xtr, dat$ytr, method = "HTh3")
fit
pred <- predict(fit, dat$xte)
mean(pred != dat$yte)            # test error
head(predict(fit, dat$xte, type = "posterior"))
```

---

hnsc_cv *Cross-validation for a fitted NSC classifier*

Description

Estimates the misclassification error at each threshold of a fitted [hnsc_train()] object by class-balanced K-fold cross-validation (the procedure of **pamr**’s ‘pamr.cv‘). The number of folds is the size of the smallest class, capped at 10.

Usage

```
hnsc_cv(fit, x, y, folds = NULL)
```

Arguments

`fit`	An object returned by [hnsc_train()].
`x, y`	The training data used to create ‘fit‘.
`folds`	Optional list of sample-index vectors defining the folds (for example from [balanced_folds()]).

Value

A list of class ‘"hnsc_cv"‘ with components ‘threshold‘, ‘size‘ (number of genes kept by the full-data fit), ‘error‘ (cross-validated error rate), ‘yhat‘ (matrix of cross-validated predictions) and ‘folds‘.

Examples

```
set.seed(1)
dat <- simulate_hetero_data(ntr1 = 10, ntr2 = 10, nte1 = 5, nte2 = 5,
                            p = 200, a = 15)
fit <- hnsc_train(dat$xtr, dat$ytr, method = "HTh3")
cv <- hnsc_cv(fit, dat$xtr, dat$ytr)
cv
```

`hnsc_methods` *The six nearest shrunken centroid classifiers*

Description

Lists the six classifiers implemented in the package, with the labels used in the paper and the corresponding ‘method‘ / ‘variance‘ arguments.

Usage

```
hnsc_methods()
```

Details

Only STh (soft thresholding with pooled variance) is the original PAM classifier; HTh and OTh are the homoscedastic classifiers of Sahtout, Wang and Ghimire (2024).

Value

A data frame with columns ‘label‘, ‘method‘, ‘variance‘, ‘description‘ and ‘reference‘.

Examples

```
hnsc_methods()
```

`hnsc_select` *Choose the threshold by refined ("deep") cross-validation*

Description

Implements the threshold search of Sahtout, Wang and Ghimire (2024) used in the paper: a cross-validation over the default threshold grid, followed by repeated refinement of the grid between the neighbours of the current best threshold until the number of selected genes can no longer change. Among thresholds with equal cross-validated error, the one examined last (for soft and hard thresholding typically the larger, sparser one) is chosen.

Usage

```
hnsc_select(
  x,
  y,
  method = c("hard", "soft", "order"),
  variance = c("hetero", "homo"),
  offset.percent = 50,
  max.iter = 100
)
```

Arguments

`x`	Numeric matrix of expression values with **genes in rows and samples in columns** (the same layout as **pamr**).
`y`	Vector or factor of class labels, one per column of ‘x‘.
`method`	Thresholding rule: ‘"hard"‘, ‘"soft"‘ or ‘"order"‘. A paper label (‘"STh"‘, ‘"HTh"‘, ‘"OTh"‘, ‘"STh3"‘, ‘"HTh3"‘, ‘"OTh3"‘) is also accepted, in which case ‘variance‘ is ignored. See [hnsc_methods()].
`variance`	‘"hetero"‘ (class-specific variances, the proposed method) or ‘"homo"‘ (pooled variance). With ‘"homo"‘, ‘method = "soft"‘ is the original PAM classifier; ‘"hard"‘ and ‘"order"‘ are the homoscedastic classifiers of Sahtout, Wang and Ghimire (2024).
`offset.percent`	Percentile of the standard deviations (homoscedastic) or of the class-specific variances (heteroscedastic) added as the stabilising constant s_0 (s_0k). Default 50, the median.
`max.iter`	Safety cap on the number of refinement rounds.

Value

A list of class ‘"hnsc_select"‘ with ‘threshold‘ (selected value), ‘size‘ (genes kept), ‘error‘ (its CV error), ‘path‘ (matrix of every threshold / size / CV error evaluated), ‘method‘, ‘variance‘ and ‘label‘.

Examples

```
set.seed(1)
dat <- simulate_hetero_data(ntr1 = 10, ntr2 = 10, nte1 = 5, nte2 = 5,
                            p = 200, a = 15)
sel <- hnsc_select(dat$xtr, dat$ytr, method = "OTh3")
sel$threshold
```

`hnsc_train` *Train a nearest shrunken centroid classifier over a grid of thresholds*

Description

Fits one of the six nearest shrunken centroid (NSC) classifiers for a sequence of threshold values. ‘variance = "homo"‘ uses the PAM test statistic and discriminant function with a pooled within-class standard deviation; combined with ‘method = "soft"‘ this is the original PAM classifier (STh, Tibshirani et al. 2002), while ‘method = "hard"‘ or ‘"order"‘ gives the homoscedastic classifiers of Sahtout, Wang and Ghimire (2024) (HTh, OTh). ‘variance = "hetero"‘ gives the heteroscedastic statistic of Sahtout and Wang with class-specific variances in both the test statistic (eq. 2.7) and the discriminant function (eq. 2.9) (STh3, HTh3, OTh3).

Usage

```
hnsc_train(
  x,
  y,
  method = c("hard", "soft", "order"),
  variance = c("hetero", "homo"),
  threshold = NULL,
  n.threshold = 30,
  offset.percent = 50,
  prior = NULL,
  sign.contrast = c("both", "positive", "negative"),
  remove.zeros = TRUE
)
```

Arguments

`x`	Numeric matrix of expression values with **genes in rows and samples in columns** (the same layout as **pamr**).
`y`	Vector or factor of class labels, one per column of ‘x‘.
`method`	Thresholding rule: ‘"hard"‘, ‘"soft"‘ or ‘"order"‘. A paper label (‘"STh"‘, ‘"HTh"‘, ‘"OTh"‘, ‘"STh3"‘, ‘"HTh3"‘, ‘"OTh3"‘) is also accepted, in which case ‘variance‘ is ignored. See [hnsc_methods()].
`variance`	‘"hetero"‘ (class-specific variances, the proposed method) or ‘"homo"‘ (pooled variance). With ‘"homo"‘, ‘method = "soft"‘ is the original PAM classifier; ‘"hard"‘ and ‘"order"‘ are the homoscedastic classifiers of Sahtout, Wang and Ghimire (2024).

threshold	Optional vector of thresholds. For soft and hard thresholding these are cut-offs on |d_ik|; for order thresholding they are the number of largest |d_ik| values (over all genes and classes) that are kept. If ‘NULL‘, a default grid of ‘n.threshold‘ values is used.
n.threshold	Number of thresholds in the default grid.
offset.percent	Percentile of the standard deviations (homoscedastic) or of the class-specific variances (heteroscedastic) added as the stabilising constant s_0 (s_0k). Default 50, the median.
prior	Class prior probabilities. Defaults to the class proportions in the training data.
sign.contrast	‘"both"‘ (default), ‘"positive"‘ or ‘"negative"‘: restrict to centroid differences of one sign.
remove.zeros	Drop thresholds beyond the first one that keeps no genes.

Details

Most users will want [hnsc()], which also chooses the threshold by cross-validation.

Value

An object of class ‘"hnsc"‘: a list containing, among others, ‘threshold‘, ‘nonzero‘ (number of genes kept at each threshold), ‘errors‘ (training resubstitution errors), ‘centroids‘, ‘centroid.overall‘, ‘sd‘ (denominator of d_ik), ‘sik2‘ (s_ik^2 + s_0k^2, heteroscedastic only), ‘delta‘ (the unshrunken test statistics d_ik), ‘prior‘, ‘method‘, ‘variance‘ and ‘label‘.

See Also

[hnsc()], [hnsc_cv()], [predict.hnsc()]

Examples

```
set.seed(1)
dat <- simulate_hetero_data(ntr1 = 10, ntr2 = 10, nte1 = 5, nte2 = 5,
                            p = 200, a = 15)
fit <- hnsc_train(dat$xtr, dat$ytr, method = "hard", variance = "hetero")
fit
```

---

make_cm *Confusion matrix, per-class recall and balanced accuracy*

Description

‘make_cm()‘ builds a K x K confusion matrix (rows = true class, columns = predicted class). ‘per_class_recall()‘ returns the diagonal divided by the row totals (sensitivity of each class) and ‘balanced_acc()‘ their mean (macro-averaged recall).

Usage

```
make_cm(
  true_labels,
  pred_labels,
  classes = sort(unique(as.character(true_labels)))
)

per_class_recall(cm)

balanced_acc(cm)
```

Arguments

`true_labels, pred_labels`	Vectors of true and predicted labels.
`classes`	Class labels defining the rows/columns. Labels not in ‘classes‘ are ignored.
`cm`	A confusion matrix as returned by ‘make_cm()‘.

Value

‘make_cm()‘: an integer matrix; ‘per_class_recall()‘: a named numeric vector; ‘balanced_acc()‘: a number.

Examples

```
cm <- make_cm(c("A", "A", "B", "B"), c("A", "B", "B", "B"), c("A", "B"))
cm
per_class_recall(cm)
balanced_acc(cm)
```

---

predict.hnsc *Predict with a nearest shrunken centroid classifier*

Description

Predict with a nearest shrunken centroid classifier

Usage

```
## S3 method for class 'hnsc'
predict(
  object,
  newx = NULL,
  threshold = NULL,
  type = c("class", "posterior", "centroid", "nonzero"),
  ...
)
```

Arguments

object	An object returned by [hnsc()] or [hnsc_train()].
newx	Matrix of new samples, genes in rows (same genes and order as the training data). Not needed for ‘type = "centroid"‘ or ‘"nonzero"‘.
threshold	Threshold to use. Defaults to the threshold selected by [hnsc()]; must be supplied for objects from [hnsc_train()].
type	‘"class"‘ (predicted labels), ‘"posterior"‘ (class posterior probabilities), ‘"centroid"‘ (shrunken class centroids, genes x classes) or ‘"nonzero"‘ (row indices of the genes used).
...	Not used.

Value

A factor, matrix or integer vector depending on ‘type‘.

Examples

```
set.seed(1)
dat <- simulate_hetero_data(ntr1 = 10, ntr2 = 10, nte1 = 5, nte2 = 5,
                            p = 200, a = 15)
fit <- hnsc_train(dat$xtr, dat$ytr, method = "STh3")
predict(fit, dat$xte, threshold = fit$threshold[5])
```

---

simulate_hetero_data *Simulate two-class data with correlated informative genes*

Description

Generates the two-class designs with correlated informative genes used in the paper's simulation study. Class A has mean zero for every gene; class B has mean ‘a/10 * i‘ for the first ‘ngene‘ (informative) genes and zero elsewhere. All non-informative genes are independent with standard deviation ‘s[i]‘ in both classes.

Usage

```
simulate_hetero_data(
  ntr1,
  ntr2,
  nte1,
  nte2,
  ngene = 20,
  p = 10000,
  s = rep_len(1:20, p),
  a = 10,
  rho = 0.95,
  correlated = c("B", "both")
)
```

Arguments

ntr1, ntr2	Training sample sizes of classes A and B.
nte1, nte2	Test sample sizes of classes A and B.
ngene	Number of informative genes.
p	Total number of genes.
s	Vector of length ‘p‘ with the gene standard deviations. The default cycles through 1, 2, ..., 20 as in the paper.
a	Signal strength; class-B means are ‘a/10 * (1:ngene)‘.
rho	Correlation parameter (0.95 in the paper).
correlated	Which classes have correlated informative genes: * ‘"B"‘ (default): only class B; class A is independent with standard deviation ‘s[i]‘ for every gene. This is the heteroscedastic correlated design (script ‘simu_heter_correlated.R‘). * ‘"both"‘: both classes share the covariance above, so the design is homoscedastic (script ‘simu_homo_correlated.R‘).

Details

The informative genes are drawn jointly from a multivariate normal distribution with covariance

$$\Sigma = vv^\top, \qquad v_i = \sqrt{\rho^{i-1}}\, s_i,$$

i.e. $\Sigma_{ij} = \rho^{(i+j-2)/2} s_i s_j$. This is the matrix used to produce the results in the paper. It has rank one, so the informative genes are perfectly correlated with one another, and gene ‘i‘ has variance $\rho^{i-1} s_i^2$.

Random numbers are drawn in the same order as in the original scripts, so with the same seed the data are identical (after transposition) to those used for the paper.

Value

A list with ‘xtr‘, ‘xte‘ (genes x samples matrices), ‘ytr‘, ‘yte‘ (character label vectors with values ‘"A"‘ and ‘"B"‘).

Examples

```
set.seed(8900)
dat <- simulate_hetero_data(ntr1 = 15, ntr2 = 20, nte1 = 50, nte2 = 50,
                            p = 1000, a = 12)
dim(dat$xtr)
table(dat$ytr)
```

summary.hnsc_comparison *Summaries of a classifier comparison*

Description

'summary()' gives, for each method, the mean and standard deviation of the test error over runs, the balanced accuracy and the mean number of genes used. 'recall_table()' gives the recall (sensitivity) of every class. Both are computed from the confusion matrix averaged over runs, as in the paper.

Usage

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

recall_table(object)
```

Arguments

object	A result of [compare_nsc()].
...	Not used.

Value

A data frame. 'summary()': one row per method. 'recall_table()': one row per class and one 'recall_<method>' column per method.

Examples

```
set.seed(8900)
dat <- simulate_hetero_data(ntr1 = 15, ntr2 = 20, nte1 = 50, nte2 = 50,
                            p = 300, a = 15)
res <- compare_nsc(dat$xtr, dat$ytr, dat$xte, dat$yte, n.runs = 2,
                   thresholding = "hard", verbose = FALSE)
summary(res)
recall_table(res)
```

Index

balanced_acc *(*make_cm*)*, 10
balanced_folds, 2

compare_nsc, 3
compute_rho2, 4

hnsc, 5
hnsc_cv, 6
hnsc_methods, 7
hnsc_select, 8
hnsc_train, 9

make_cm, 10

per_class_recall *(*make_cm*)*, 10
predict.hnsc, 11

recall_table *(*summary.hnsc_comparison*)*, 14

simulate_hetero_data, 12
summary.hnsc_comparison, 14