Package ‘hyperoverlap’

May 8, 2026

Title Overlap Detection in n-Dimensional Space

Version 1.1.4

Description

Uses support vector machines to identify a perfectly separating hyperplane (linear or curvilinear) between two entities in high-dimensional space. If this plane exists, the entities do not overlap. Applications include overlap detection in morphological, resource or environmental dimensions. More details can be found in: Brown et al. (2020) <doi:10.1111/2041-210X.13363> .

Depends R (>= 3.4.0)

Imports e1071, matlib, ggplot2, rgl, misc3d, methods, stats, MASS, graphics

License GPL-3

Encoding UTF-8

Suggests knitr, rmarkdown, plotrix, rlang, webshot2, markdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Matilda Brown [aut, cre] (ORCID: <`https://orcid.org/0000-0003-2536-8365`>), Greg Jordan [aut]

Maintainer Matilda Brown <`m.brown2@kew.org`>

Repository CRAN

Date/Publication 2026-01-21 12:50:02 UTC

Contents

`hyperoverlap-class` *Storage class for the description of hyperoverlaps*

Description

Storage class for the description of hyperoverlaps

Slots

`entity1` A length-one character vector

`entity2` A length-one character vector

`dimensions` A length n character vector containing the variables used to define the space

`occurrences` A matrix containing the labelled input data

`shape` shape of the decision boundary; either "linear" or "curvilinear"

`polynomial.order` a length-one numeric vector showing the polynomial order of the most accurate kernel function. "0" if linear kernel.

`result` a length-one character vector, either "overlap" or "non-overlap"

`accuracy` a 2x2 table with the true (y) and predicted (pred) labels

`number.of.points.misclassified` a length-one numeric vector

`model` svm model used to plot decision boundary

`hyperoverlap_detect` *Overlap detection in n-dimensional space using support vector machines (SVMs)*

Description

Given a matrix containing the ecological data (x) and labels (y) for two entities, a support vector machine is trained and the predicted label of each point is evaluated. If every point has been classified correctly, the entities can be separated and they do not overlap.

Usage

```
hyperoverlap_detect(x, y, kernel = "polynomial", kernel.degree = 3, cost = 500,
        stoppage.threshold = 0.4, verbose = TRUE, set = FALSE)
```

Arguments

`x`	A matrix or data.frame containing the variables of interest for both entities.
`y`	A vector of labels.
`kernel`	Character. Either "linear" or "polynomial" (default = "polynomial").
`kernel.degree`	Parameter needed for `kernel` = `polynomial` (default = 3).
`cost`	Specifies the SVM margin 'hardness'. Default value is 50, but can be increased for improved accuracy (although this increases runtimes and memory usage).
`stoppage.threshold`	Numeric. If the number of points misclassified using a linear hyperplane exceeds this proportion of the number of observations, non-linear separation is not attempted. Must be between 0 and 1 (default = 0.2).
`verbose`	Logical. If TRUE, prints diagnostic messages.
`set`	Logical. Is this function being called as part of `hyperoverlap_set()`? Should not need to be changed.

Details

Input data should be preprocessed so that all variables are comparable (e.g. same order of magnitude). Polynomial kernels allow curvilinear decision boundaries to be found between entities (see `https://www.cs.cmu.edu/~ggordon/SVMs/new-svms-and-kernels.pdf`). Smaller values of `kernel.degree` permit less complex decision boundaries; biological significance is likely to be lost at values > 5.

Value

A `hyperoverlap-class` object

Examples

```
data = iris[which(iris$Species!=("versicolor")),]
x = hyperoverlap_detect(data[,1:3],data$Species, kernel="linear")
```

`hyperoverlap_lda` *Hyperoverlap visualisation using linear discriminant analysis (LDA)*

Description

Hyperoverlap visualisation using linear discriminant analysis (LDA)

Usage

```
hyperoverlap_lda(x, return.plot=TRUE, visualise3d=FALSE, showlegend=TRUE)
```

Arguments

`x`	An `hyperoverlap-class` object.
`return.plot`	Logical. If TRUE, data are plotted using `plot()`.
`visualise3d`	Logical. If FALSE, data are projected onto two axes (LDA1, residualPCA1). If TRUE, data are projected onto three axes (LDA1, residualPCA1, residualPCA2)
`showlegend`	Logical. Used for 3D plots.

Details

This function provides a way to visualise overlap (or non-overlap) between classes of high dimensional data. For inspection, it is useful to use the base graphics package (implemented by return.plot=TRUE). The transformed coordinates of each point are also returned as a dataframe, which can be plotted with user-defined parameters.

Value

Returns a dataframe with columns "Entity", "LDA1", "residualPCA1", "residualPCA2" (if `visualise3d = TRUE`)

See Also

`hyperoverlap_detect`

Examples

```
#using iris dataset reduced to two species
data = iris[which(iris$Species!=("versicolor")),]
x = hyperoverlap_detect(data[1:4], data$Species)
hyperoverlap_lda(x)
```

`hyperoverlap_pairs_plot`

Overlap heatmap plotting for analysis of multiple entities

Description

This function plots a matrix of overlap.

Usage

```
hyperoverlap_pairs_plot(x, cols = pal)
```

Arguments

`x`	A matrix of the form produced by produced by `hyperoverlap_set()` (see Details).
`cols`	A vector of colours (default: `c("red","blue")`).

Details

Input matrix must contain columns named "entity1", "entity2" and "result"

Value

A `ggplot` object

Examples

```
hyperoverlap.iris.set = hyperoverlap_set(iris[1:3],iris$Species, kernel="linear")
hyperoverlap_pairs_plot(hyperoverlap.iris.set)
```

`hyperoverlap_plot` *Overlap plotting for low-dimensional spaces*

Description

Plot the optimal separating hyperplane found by hyperoverlap_detect() in 3D .

Usage

```
hyperoverlap_plot(x)
```

Arguments

`x`	An `hyperoverlap-class` object.

See Also

`hyperoverlap_detect` , `hyperoverlap_lda`

Examples

```
data = iris[which(iris$Species!=("versicolor")),]
x = hyperoverlap_detect(data[,1:3],data$Species, kernel="linear")
hyperoverlap_plot(x)
```

`hyperoverlap_set` *Pairwise overlap detection in n-dimensional space of multiple entities using support vector machines (SVMs)*

Description

This function is a wrapper for `hyperoverlap_detect` for pairwise overlap detection between multiple entities.

Usage

```
hyperoverlap_set(x, y, kernel = "polynomial",kernel.degree = 3, cost = 1000,
stoppage.threshold = 0.2, write.to.file = FALSE,
path = NULL,
sample.dimensionality.omit = "FALSE")
```

Arguments

`x`	A matrix or data.frame containing the variables of interest for both entities.
`y`	A vector of labels.
`kernel`	Character. Either "linear" or "polynomial" (default = "polynomial").
`kernel.degree`	Parameter needed for `kernel` = `polynomial` (default = 3).
`cost`	Specifies the SVM margin 'hardness'. Default value is 1000, but can be increased for improved accuracy (although this increases runtimes and memory usage).
`stoppage.threshold`	Numeric. If the number of points misclassified using a linear hyperplane exceeds this proportion of the number of observations, non-linear separation is not attempted. Must be between 0 and 1 (default = 0.2).
`write.to.file`	Logical. If TRUE, each `hyperoverlap-class` object is saved as a .rds file.
`path`	Character. Path to write .rds files to. Ignored if `write.to.file=FALSE`
`sample.dimensionality.omit`	Logical. If TRUE, omits any entity pairs with a combined sample size less than n+1, where n is the number of dimensions (see details).

Details

In n dimensions, any set of points up to n+1 points can be separated using a linear hyperplane. This may produce an artefactual non-overlap result. The `sample.dimensionality.omit` parameter gives two options for dealing with these pairs when they form part of a larger analysis. If `sample.dimensionality.omit = "TRUE"`, this pair is removed from the analysis (result = NA). If `sample.dimensionality.omit = "FALSE"`, the pair is included, but a warning is printed.

Value

A long-form matrix with the following columns: entity1, entity2, shape, polynomial.order (if `kernel="polynomial"`), result, number.of.points.misclassified.

If specified, individual `Hyperoverlap-class` objects are written to file.

Examples

```
data(iris)
hyperoverlap.iris.set = hyperoverlap_set(iris[1:3],iris$Species, kernel="linear")
```

Index