Package 'ProjectionBasedClustering'

March 29, 2020

Type Package

Title Projection Based Clustering

Version 1.1.1

Date 2020-03-19

Description A clustering approach applicable to every projection method is proposed here. The two-dimensional scatter plot of any projection method can construct a topographic map which displays unapparent data structures by using distance and density information of the data. The generalized U*-matrix renders this visualization in the form of a topographic map, which can be used to automatically define the clusters of high-dimensional data. The whole system is the generalization of on an idea from the book ``Projection-Based Clustering through Self-Organization and Swarm Intelligence" <DOI:10.1007/978-3-658-20540-9>. Selecting the correct projection method will result in a visualization in which mountains surround each cluster. The number of clusters can be determined by counting valleys on the topographic map. Most projection methods are wrappers for already available methods in R. By contrast, the neighbor retrieval visualizer (NeRV) is based on C++ source code of the 'dredviz' software package, and the Curvilinear Component Analysis (CCA) is translated from 'MATLAB' ('SOM Toolbox' 2.0) to R.

License GPL-3

Imports Rcpp, ggplot2, stats, graphics, vegan, deldir, geometry, GeneralizedUmatrix, shiny, shinyjs, shinythemes, plotly, grDevices

Suggests DataVisualizations, fastICA, tsne, FastKNN, MASS, pcaPP, spdep, pracma, grid, mgcv, fields, png, reshape2, Rtsne, methods

LinkingTo Rcpp

LazyData TRUE

Encoding UTF-8

SystemRequirements C++11

Depends R (>= 3.0)

NeedsCompilation yes

URL http://www.deepbionics.org

LazyLoad yes

BugReports https://github.com/Mthrun/ProjectionBasedClustering/issues

Author Michael Thrun [aut, cre, cph],
Florian Lerch [aut],
Felix Pape [aut],
Tim Schreier [aut],
Luis Winckelmann [aut],
Kristian Nybo [cph],
Jarkko Venna [cph]

Maintainer Michael Thrun <m.thrun@gmx.net>

Repository CRAN

Date/Publication 2020-03-29 10:50:03 UTC

R topics documented:

ProjectionBasedClustering-package
Projection Based Clustering

Description

The package is based on a conference talk [Thrun/Ultsch, 2017], see <DOI:10.13140/RG.2.2.13124.53124>. The abstract follows:

Many data mining methods rely on some concept of the dissimilarity between pieces of information encoded in the data of interest. These methods can be used for cluster analysis. However, no generally accepted definition of clusters exists in the literature [Hennig et al., 2015]. Here, consistent with Bouveyron et al., it is assumed that a cluster is a group of similar objects [Bouveyron et al., 2012]. The clusters are called natural because they do not require a dissection; instead, they are clearly separated in the data [Duda et al., 2001, Theodoridis/Koutroumbas, 2009, pp. 579, 600]. These clusters can be identified by distance or density based high-dimensional structures. Dimensionality reduction techniques are able to reduce the dimensions of the input space to facilitate the exploration of structures in high-dimensional data. If they are used for visualization, they are called projection methods. The generalized U*-matrix technique is applicable for these and can be used to visualize both distance- and density-based structures [Thrun 2018; Ultsch/Thrun, 2017]. The idea that the abstract U*-matrix (AU-matrix) can be used for clustering [Ultsch et al., 2016]. The distances required for hierarchical clustering are defined by the AU-matrix [Lötsch/Ultsch, 2014]. Using this distance we propose a clustering approach for every projection method based on the U*-matrix visualization of a topographic map [Thrun 2018; Thrun/Ultsch, 2017]. The number of clusters and the cluster structure can be estimated by counting the valleys in a topographic map [Thrun et al., 2016]. If the number of clusters and the clustering method are chosen correctly, then the clusters will be well separated by mountains in the visualization. Outliers are represented as volcanoes and can be interactively marked in the visualization after the automated clustering process.

Details

A comparison of the Pswarm projection method [Thrun, 2018] using projection based clustering to 26 common clustering algorithms is provided in http://www.deepbionics.org/Projects/ClusteringAlgorithms.html.

Note

If you want to verifiy your clustering result externally, you can use `Heatmap` or `SilhouettePlot` of the CRAN package `DataVisualizations`.

Additionally you can use the standard `ShepardScatterPlot` or the better approach through the `ShepardDensityPlot` of the CRAN package `DataVisualizations`.

Author(s)

Michael Thrun, Felix Pape, Florian Lerch, Tim Schreier, Luis Winckelmann

References

[Thrun/Ultsch, 2017] Thrun, M.C., Ultsch, A.: Projection based Clustering, Conf. Int. Federation of Classification Societies (IFCS),DOI:10.13140/RG.2.2.13124.53124, Tokyo, 2017.

[Bouveyron et al., 2012] Bouveyron, C., Hammer, B., & Villmann, T.: Recent developments in clustering algorithms, Proc. ESANN, Citeseer, 2012.

[Duda et al., 2001] Duda, R. O., Hart, P. E., & Stork, D. G.: Pattern classification, (Second Edition ed.), Ney York, USA, John Wiley & Sons, ISBN: 0-471-05669-3, 2001.

[Hennig et al., 2015] Hennig, C., Meila, M., Murtagh, F., & Rocci, R.: Handbook of cluster analysis, New York, USA, CRC Press, ISBN: 9781466551893, 2015.

[Lötsch/Ultsch, 2014] Lötsch, J., & Ultsch, A.: Exploiting the Structures of the U-Matrix, in Villmann, T., Schleif, F.-M., Kaden, M. & Lange, M. (eds.), Proc. Advances in Self-Organizing Maps and Learning Vector Quantization, pp. 249-257, Springer International Publishing, Mittweida, Germany, 2014.

[Theodoridis/Koutroumbas, 2009] Theodoridis, S., & Koutroumbas, K.: Pattern Recognition, (Fourth Edition ed.), Canada, Elsevier, ISBN: 978-1-59749-272-0, 2009.

[Thrun et al., 2016] Thrun, M. C., Lerch, F., Lötsch, J., & Ultsch, A.: Visualization and 3D Printing of Multivariate Data of Biomarkers, in Skala, V. (Ed.), International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Vol. 24, Plzen, http://wscg.zcu.cz/wscg2016/short/A43-full.pdf, 2016.

[Ultsch et al., 2016] Ultsch, A., Behnisch, M., & Lötsch, J.: ESOM Visualizations for Quality Assessment in Clustering, In Merényi, E., Mendenhall, J. M. & O'Driscoll, P. (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization: Proceedings of the 11th International Workshop WSOM 2016, Houston, Texas, USA, January 6-8, 2016, (10.1007/978-3-319-28518-4_3pp. 39-48), Cham, Springer International Publishing, 2016.

[Ultsch/Thrun, 2017] Ultsch, A., & Thrun, M. C.: Credible Visualizations for Planar Projections, in Cottrell, M. (Ed.), 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM), IEEE Xplore, France, 2017.

[Thrun, 2018] Thrun, M. C.: Projection Based Clustering through Self-Organization and Swarm Intelligence, doctoral dissertation 2017, Springer, Heidelberg, ISBN: 978-3-658-20539-3, https://doi.org/10.1007/978-3-658-20540-9, 2018.

Examples

```
data('Hepta')
#2d projection
# Visualizuation of GeneralizedUmatrix

projectionpoints=NeRV(Hepta$Data)
#Computation of Generalized Umatrix
library(GeneralizedUmatrix)
visualization=GeneralizedUmatrix(Data = Hepta$Data,projectionpoints)
TopviewTopographicMap(visualization$Umatrix,visualization$Bestmatches)
#or in 3D if rgl package exists
#plotTopographicMap(visualization$Umatrix,visualization$Bestmatches)

##Interactive Island Generation
```

```
## from a tiled Umatrix (toroidal assumption)
## Not run:
Imx = ProjectionBasedClustering::interactiveGeneralizedUmatrixIsland(visualization$Umatrix,
visualization$Bestmatches)
#plotTopographicMap(visualization$Umatrix,visualization$Bestmatches, Imx = Imx)

## End(Not run)

# Automatic Clustering
LC=c(visualization$Lines,visualization$Columns)
# number of Cluster from dendrogram or visualization (PlotIt=TRUE)
Cls=ProjectionBasedClustering(k=7, Hepta$Data,

visualization$Bestmatches, LC,PlotIt=TRUE)
# Verification
library(GeneralizedUmatrix)
TopviewTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)
#or in 3D if rgl package exists
#plotTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)

## Sometimes you can improve a Clustering interactivly or mark additional Outliers manually
## Not run:
Cls2 = interactiveClustering(visualization$Umatrix, visualization$Bestmatches, Cls)

## End(Not run)
```

CCA *Curvilinear Component Analysis*

Description

CCA Projects data vectors using Curvilinear Component Analysis.

Unknown values (NaN's) in the data: projections of vectors with unknown components tend to drift towards the center of the projection distribution. Projections of totally unknown vectors are set to unknown (NaN).

Usage

```
CCA(DataOrDists,Epochs,OutputDimension=2,method='euclidean',

alpha0 = 0.5, lambda0,PlotIt=FALSE,Cls)
```

Arguments

DataOrDists	array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric epochs (scalar) training length
Epochs	(scalar) training length

`OutputDimension`	Number of dimensions in the Outputspace, default=2
`method`	method specified by distance string. One of: 'euclidean','cityblock=manhatten','cosine','chebychev','jacc
`alpha0`	(scalar) initial step size, 0.5 by default
`lambda0`	(scalar) initial radius of influence, 3*max(std(D)) by default
`PlotIt`	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown
`Cls`	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

A n by OutputDimension matrix containing coordinates of the projected points.

Note

Only Transfered from matlab to R. Matlabversion: Contributed to SOM Toolbox 2.0, February 2nd, 2000 by Juha Vesanto http://www.cis.hut.fi/projects/somtoolbox/

You can use the standard `ShepardScatterPlot` or the better approach through the `ShepardDensityPlot` of the CRAN package `DataVisualizations`.

Author(s)

Florian Lerch

References

Demartines, P., Herault, J., "Curvilinear Component Analysis: a Self-Organizing Neural Network for Nonlinear Mapping of Data Sets", IEEE Transactions on Neural Networks, vol 8, no 1, 1997, pp. 148-154.

`DefaultColorSequence` *Default color sequence for plots*

Description

Defines the default color sequence for plots made within the Projections package.

Usage

```
data("DefaultColorSequence")
```

Format

A vector with 562 different strings describing colors for plots.

`Delaunay4Points` *Adjacency matrix of the delaunay graph for BestMatches of Points*

Description

Calculates the adjacency matrix of the delaunay graph for BestMatches (BMs) in tiled form if BestMatches are located on a toroid grid

Usage

```
Delaunay4Points(Points, IsToroid = TRUE,Grid=NULL,PlotIt=FALSE)
```

Arguments

`Points`	[1:n,1:3] matrix containing the BMKey, X and Y coordinates of the n, BestMatches NEED NOT BE UNIQUE, however, there is an edge in the Deaunay between duplicate points!
`IsToroid`	OPTIONAL, logical, indicating if BM's are on a toroid grid. Default is True
`Grid`	OPTIONAL, A vector of length 2, containing the number of lines and columns of the Grid
`PlotIt`	OPTIONAL, bool, Plots the graph

Details

as

Value

Delaunay[1:n,1:n] adjacency matrix of the Delaunay-Graph

Author(s)

Michael Thrun

References

[Thrun, 2018] Thrun, M. C.: Projection Based Clustering through Self-Organization and Swarm Intelligence, doctoral dissertation 2017, Springer, ISBN: 978-3-658-20539-3, Heidelberg, 2018.

`DijkstraSSSP` *Dijkstra SSSP*

Description

Dijkstra's SSSP (Single source shortest path) algorithm:

gets the shortest path (geodesic distance) from source vertice(point) to all other vertices(points) defined by the edges of the adjasency matrix

Usage

```
DijkstraSSSP(Adj, Costs, source)
```

Arguments

`Adj`	[1:n,1:n] 0/1 adjascency matrix, e.g. from delaunay graph or gabriel graph
`Costs`	[1:n,1:n] matrix, distances between n points (normally euclidean)
`source`	int, vertice(point) from which to calculate the geodesic distance to all other points

Details

Preallocating space for DataStructures accordingly to the maximum possible number of vertices which is fixed set at the number 10001.

Value

ShortestPaths[1:n] vector, shortest paths (geodesic) to all other vertices including the source vertice itself

Note

runs in O(E*Log(V))

Author(s)

Michael Thrun

References

uses a changed code which is inspired by Shreyans Sheth 28.05.2015, see http://ideone.com/qkmt31

Hepta *Hepta from FCPS*

Description

clearly defined clusters, different variances

Usage

```
data("Hepta")
```

Details

Size 212, Dimensions 3, stored in `Hepta$Data`

Classes 7, stored in `Hepta$Cls`

References

Ultsch, A.: U* C: Self-organized Clustering with Emergent Feature Maps, Lernen, Wissensentdeckung und Adaptivitaet (LWA), pp. 240-244, Saarbruecken, Germany, 2005.

Examples

```
data(Hepta)
str(Hepta)
```

ICA *Independent Component Analysis)*

Description

Independent Component Analysis:

Negentropie: difference of entropy to a corresponding normally-distributed random variable J(y)=|E(G(y)-E(G(v)))|^2

Usage

```
ICA(Data,OutputDimension=2,Contrastfunction="logcosh",

Alpha=1,Iterations=200,PlotIt=FALSE,Cls)
```

Arguments

Data	array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
OutputDimension	Number of dimensions in the Outputspace, default=2
Contrastfunction	Maximierung der Negentropie ueber geeignete geeignete Kontrastfunktion Default: 'logcosh' G(u)=1/a*log cosh(a*u) 'exp': G(u)=-exp(u^2/2)
Alpha	onstant with 1<=alpha<=2 used in approximation to neg-entropy when fun == "logcosh"
Iterations	maximum number of iterations to perform.
PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown
Cls	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

ProjectedPoints	[1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projectio
Mixing	[1:OutputDimension,1:d] Mischungsmatrix s.d gilt Data=MixingMatrix*ProjectedPoints
Unmixing	Entmischungsmatrix with Data*Unmixing=ProjectedPoints
PCMatrix	pre-whitening matrix that projects data onto the first n.comp principal components.

Note

A wrapper for fastICA

You can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Author(s)

Michael Thrun

`interactiveClustering` *GUI for interactive cluster analysis*

Description

This tool is an interactive shiny tool that visualizes a given generalized Umatrix and allows the user to select areas and mark them as clusters to improve a projection based clustering.

Arguments

`Umatrix`	[1:Lines,1:Columns] Matrix of Umatrix Heights
`Bestmaches`	[1:n,1:2]Array with positions of Bestmatches
`Cls`	[1:n]Classification of the Bestmatches
`Imx`	[1:4*Lines,1:4*Columns]Matrix of an island that will be cut out of the umatrix, use package Umatrix for generation.
`Toroid`	Are Bestmatches placed on a toroid grid? TRUE by default.

Details

Clicking on "Quit" returns the Cls vector to the workspace.

Value

Cls[1:n]: A vector containing the selected class ids. The order is corresponding to the given Bestmatches

Note

If you want to verifiy your clustering result externally, you can use `Heatmap` or `SilhouettePlot` of the CRAN package `DataVisualizations`.

Author(s)

Florian Lerch, Michael Thrun

References

[Thrun/Ultsch, 2017] Thrun, M.C., Ultsch, A.: Projection based Clustering, Conf. Int. Federation of Classification Societies (IFCS),DOI:10.13140/RG.2.2.13124.53124, Tokyo, 2017.

[Thrun, 2018] Thrun, M. C.: Projection Based Clustering through Self-Organization and Swarm Intelligence, doctoral dissertation 2017, Springer, Heidelberg, ISBN: 978-3-658-20539-3, https://doi.org/10.1007/978-3-658-20540-9, 2018.

Examples

```
data('Hepta')
#2d projection
# Visualizuation of GeneralizedUmatrix

projectionpoints=NeRV(Hepta$Data)
#Computation of Generalized Umatrix
library(GeneralizedUmatrix)
visualization=GeneralizedUmatrix(Data = Hepta$Data,projectionpoints)

## Semi-Automatic Clustering done interactivly in a  shiny gui
## Not run:
Cls = interactiveClustering(visualization$Umatrix, visualization$Bestmatches)
##Plotting
plotTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)

## End(Not run)
```

---

interactiveGeneralizedUmatrixIsland

GUI for cutting out an Island.

Description

The toroid Umatrix is usually drawn 4 times, so that connected areas on borders can be seen as a whole. An island is a manual cutout of such a tiled visualization, that is selected such that all connected areas stay intact. This shiny tool allows the user to do this manually.

Usage

```
interactiveGeneralizedUmatrixIsland(Umatrix, Bestmatches=NULL, Cls=NULL)
```

Arguments

`Umatrix`	[1:Lines,1:Columns] Matrix of Umatrix Heights
`Bestmatches`	Array with positions of Bestmatches
`Cls`	Classification of the Bestmatches

Details

Clicking on "Quit" returns the Imx matrix to the workspace. Details can bee read in [Thrun et al, 2016, Thrun/Ultsch, 2017].

Value

Boolean Matrix that represents the island within the tiled Umatrix.

Note

This function is a depricated version of a function from the Umatrix packages created by Florian Lerch and Michael Thrun

Author(s)

Michael Thrun, Florian Lerch

References

[Thrun, et al.,2016] Thrun, M. C., Lerch, F., Loetsch, J., Ultsch, A.: Visualization and 3D Printing of Multivariate Data of Biomarkers, in Skala, V. (Ed.), International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision,Plzen, 2016.

[Thrun/Ultsch, 2017] Thrun, M.C., Ultsch, A.: Projection based Clustering, Conf. Int. Federation of Classification Societies (IFCS),<DOI:10.13140/RG.2.2.13124.53124>, Tokyo, 2017.

Examples

```
data("Hepta")
Data=Hepta$Data
Cls=Hepta$Cls
InputDistances=as.matrix(dist(Data))
res=cmdscale(d=InputDistances, k = 2, eig = TRUE, add = FALSE, x.ret = FALSE)
ProjectedPoints=as.matrix(res$points)
#see also ProjectionBasedClustering package for other common projection methods

library(GeneralizedUmatrix)
resUmatrix=GeneralizedUmatrix(Data,ProjectedPoints)
TopviewTopographicMap(resUmatrix$Umatrix,resUmatrix$Bestmatches,Cls)
#or in 3D if rgl package exists
#plotTopographicMap(resUmatrix$Umatrix,resUmatrix$Bestmatches,Cls)

##Interactive Island Generation
## from a tiled Umatrix (toroidal assumption)

## Not run:
Imx = interactiveGeneralizedUmatrixIsland(resUmatrix$Umatrix,

resUmatrix$Bestmatches)
plotTopographicMap(resUmatrix$Umatrix,

resUmatrix$Bestmatches, Imx = Imx)

## End(Not run)
```

`interactiveProjectionBasedClustering` *Interactive Projection-Based Clustering of [Thrun et al., 2020] based on [Thrun/Ultsch, 2017].*

Description

An interactive clustering tool using the topographic map visualizations of the generalized U-matrix and a variety of different projection methods. This function receives a dataset and starts a shiny interface where one is able to choose a projection method and generate a plot.ly visualization of the topograhpic mao [Thrun et al., 2016] of the generalized U-matrix [Ultsch/Thrun, 2017] and the projected points. It includes capabilities for interactive clustering within the interface.

Usage

```
interactiveProjectionBasedClustering(Data, Cls=NULL )
```

Arguments

`Data`	The dataset [1:n,1:d] of n cases and d vriables with which the U-matrix and the projection will be calculated.
`Cls`	Optional: Prior Classification of the data for the [1:n] cases of k classes.

Details

Cls is a numerical vector of the length n (number of cases) with the integer elements of numbers from 1 to k if k is the number of groups in the data. Each element of Cls as an unambigous mapping to a case of Data indicating by the rownames of Data. If Data has no rownames a vector from 1:n is generated and then Cls is named by it.

Value

Returns a List of:

`Cls`	[1:n] numerical vector of the clustering of the dataset for then cases of k clusters
`Plot`	The final plot generated by plot.ly when closing the tool
`Umatrix`	[1:Lines,1:Columns] Umatrix to be plotted, numerical matrix storing the U-heights, see [Thrun, 2018] for definition.
`Bestmatches`	[1:n,2] GridConverted Projected Points information converted by convertProjectionProjectedPoints() to predefined Grid by Lines and Columns

Author(s)

Tim Schreier,Felix Pape, Luis Winckelmann, Michael Thrun

References

[Thrun, 2018] Thrun, M. C.: Projection Based Clustering through Self-Organization and Swarm Intelligence, doctoral dissertation 2017, Springer, Heidelberg, ISBN: 978-3-658-20539-3, https://doi.org/10.1007/978-3-658-20540-9, 2018.

[Ultsch/Thrun, 2017] Ultsch, A., & Thrun, M. C.: Credible Visualizations for Planar Projections, in Cottrell, M. (Ed.), 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM), IEEE Xplore, France, 2017.

[Thrun/Ultsch, 2017] Thrun, M. C., & Ultsch, A. : Projection based Clustering, Proc. International Federation of Classification Societies (IFCS), pp. 250-251, Japanese Classification Society (JCS), Tokyo, Japan, 2017.

[Thrun et al., 2020] Thrun, M. C., Pape, F., & Alfred, U.: An Interactive Machine Learning Tool For Clustering in Visual Analytics, technical report beeing submitted, 2020.

`Isomap` *Isomap projection method*

Description

Isomap procetion as introduced in 2000 by Tenenbaum, de Silva and Langford

Even with a manifold structure, the sampling must be even and dense so that dissimilarities along a manifold are shorter than across the folds. If data do not have such a manifold structure, the results are very sensitive to parameter values.

Usage

```
Isomap(Inputdistances,k,OutputDimension=2,PlotIt=FALSE,Cls)
```

Arguments

`Inputdistances`	Matrix containing the distances of the data
`k`	number of k nearest neighbors, if the data is fragmented choose an higher k
`OutputDimension`	Number of dimensions in the output space, default = 2
`PlotIt`	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. If OutputDimension > 2 only the first two dimensions will be shown.
`Cls`	Optional and only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

ProjectedPoints[1:n,OutputDimension] n by OutputDimension matrix containing coordinates of the Projection: A matrix of the fitted configuration..

Note

A wrapper for isomap of the package vegan

if Data fragmented choose an higher k

You can use the standard `ShepardScatterPlot` or the better approach through the `ShepardDensityPlot` of the CRAN package `DataVisualizations`.

Author(s)

Michael Thrun

`KruskalStress` *Kruskal stress calculation*

Description

Calculates the stress as defined by Kruskal for 2 distance matrices

Usage

```
KruskalStress(InputDistances, OutputDistances)
```

Arguments

`InputDistances`	Distance matrix of the original Data
`OutputDistances`	Distance matrix of the projected Data

Details

An short overview of different types of quality measures can be found in [Thrun, 2018, p.68, Fig. 6.3] (https://doi.org/10.1007/978-3-658-20540-9).

Value

A single numerical representing the Kruskal stress of the distance matrices.

Author(s)

Felix Pape

MDS *Classical multidimensional scaling (MDS)*

Description

Classical multidimensional scaling of a data matrix. Also known as principal coordinates analysis

Usage

```
MDS(DataOrDists,method='euclidean',OutputDimension=2,PlotIt=FALSE,Cls)
```

Arguments

DataOrDists	array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
method	method specified by distance string: 'euclidean','cityblock=manhatten','cosine','chebychev','jaccard','m
OutputDimension	Number of dimensions in the Outputspace, default=2
PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization.
Cls	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

ProjectedPoints	[1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projectio
Eigenvalues	the eigenvalues of MDSvalues*MDSvalues'
Stress	Shephard-Kruskal Stress

Note

A wrapper for cmdscale

You can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Author(s)

Michael Thrun

NeRV *NeRV projection*

Description

Projection is done by the neighbor retrieval visualizer (NeRV)

Usage

```
NeRV(Data, lambda = 0.1, neighbors = 20, iterations = 10,

cg_steps = 2, cg_steps_final = 40, randominit = T, OutputDimension = 2,

PlotIt = FALSE, Cls)
```

Arguments

`Data`	Matrix of the Data to be projected
`lambda`	Optional: Controls the trustworthiness-continuity tradeoff. Default = 0.1
`neighbors`	Optional: Set the number of nearest neighbours that each point should have. Should be positive. Default = 20
`iterations`	Optional: The number of iterations to perform. Default = 10
`cg_steps`	Optional: The number of conjugate gradient steps to perform per iteration in NeRV's optimization scheme. Default = 2
`cg_steps_final`	Optional: The number of conjugate gradient steps to perform on the final iteration in NeRV's optimization scheme. Default = 40
`randominit`	Optional: TRUE: Random Initialization (default), FALSE: PCA initializiation
`OutputDimension`	Optional: Number of dimensions in the Outputspace, default=2
`PlotIt`	Optional: Should the projected points be plotted? Default: FALSE. Note: this is only usefull if OutputDimension = 2.
`Cls`	Optional: Vector containing the number of the class for each row in Data. This is only used to color the points according to their classes if PlotIt = T

Details

Uses the NeRV projection with matrix Data and lambda. Lambda controls the trustworthiness-continuity tradeoff.

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

OutputDimension-dimensional matrix of projected points

Note

PCA initialization changes form the original C++ Sourcecode of http://research.cs.aalto.fi/pml/software/dredviz/ to the R version of the projections package. Other changes are made only regarding data types of Rcpp in comparison to the original C++ Source code.

You can use the standard `ShepardScatterPlot` or the better approach through the `ShepardDensityPlot` of the CRAN package `DataVisualizations`.

Author(s)

Michael Thrun, Felix Pape

References

Jarkko Venna, Jaakko Peltonen, Kristian Nybo, Helena Aidos, and Samuel Kaski. Information Retrieval Perspective to Nonlinear Dimensionality Reduction for Data Visualization. Journal of Machine Learning Research, 11:451-490, 2010.

Jarkko Venna and Samuel Kaski. Nonlinear Dimensionality Reduction as Information Retrieval. In Marina Meila and Xiaotong Shen, editors, Proceedings of AISTATS 2007, the 11th International Conference on Artificial Intelligence and Statistics. Omnipress, 2007. JMLR Workshop and Conference Proceedings, Volume 2: AISTATS 2007.

PCA *Principal component analysis*

Description

Performs a principal components analysis on the given data matrix projection=SammonsMapping(Data)

Usage

```
PCA(Data,OutputDimension=2,Scale=FALSE,Center=FALSE,PlotIt=FALSE,Cls)
```

Arguments

`Data`	array of data: n cases in rows, d variables in columns
`OutputDimension`	Number of dimensions in the Outputspace, default=2
`Scale`	a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE for consistency with S, but in general scaling is advisable. Alternatively, a vector of length equal the number of columns of x can be supplied. The value is passed to scale.
`Center`	a logical value indicating whether the variables should be shifted to be zero centered. Alternately, a vector of length equal the number of columns of x can be supplied. The value is passed to scale
`PlotIt`	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown

Cls	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

ProjectedPoints	[1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projectio
Rotation	the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors)
sDev	the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix)
TransformedData	matrix with PCA transformed Data
Center	the centering used, or FALSE
Scale	the scaling used, or FALSE

Note

A wrapper for prcomp

You can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Author(s)

Michael Thrun

PlotProjectedPoints *Plot Projected Points*

Description

plots XY data colored by Cls with ggplot2

Usage

```
PlotProjectedPoints(Points,Cls,BMUorProjected=F,PlotLegend=FALSE,

xlab='X',ylab='Y',main="Projected Points",PointSize=2.5)
```

Arguments

Points	[1:n,1:2] xy cartesian coordinates of a projection
Cls	numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.
BMUorProjected	Default ==FALSE, If TRUE assuming BestMatches of ESOM instead of Projected Points
PlotLegend	...
xlab	Optional: Label of the x axis
ylab	Optional: Label of the y axis
main	Optional: title
PointSize	Optional: size of points

Value

ggobject of ggplot2

Author(s)

Michael Thrun

ProjectionBasedClustering

automated Clustering approach of the Databonic swarm with abstact U distances

---

Description

automated Clustering approach of the Databonic swarm with abstact U distances, which are the geodesic distances based on high-dimensional distances combined with low dimensional graph paths by using ShortestGraphPathsC.

Usage

```
ProjectionBasedClustering(k, Data, BestMatches, LC, StructureType = TRUE, PlotIt = FALSE,
                  method = "euclidean")
```

Arguments

k	number of clusters, how many to you see in the 3d landscape?
Data	[1:n,1:d] Matrix of Data (n cases, d dimensions) that will be used. One DataPoint per row
BestMatches	[1:n,1:2] Matrix with positions of Bestmatches=ProjectedPoints, one matrix line per data point
LC	grid size c(Lines,Columns)

StructureType	Optional, bool; =TRUE: compact structure of clusters assumed, =FALSE: connected structure of clusters assumed. For the two options vor Clusters, see [Thrun, 2017] or Handl et al. 2006
PlotIt	Optional, bool, Plots Dendrogramm
method	Optional, distance method, do not change

Details

ProjectionBasedClustering is a flexible and robust clustering framework based on a chose projection method and projection method a parameter-free high-dimensional data visualization technique, which generates projected points on a topographic map with hypsometric colors, see package GeneralizedUmatrix function GeneralizedUmatrix, called the generalized U-matrix. The clustering method with no sensitive parameters is done by this function. The clustering can be verified by the visualization and vice versa.

If you want to verifiy your clustering result externally, you can use Heatmap or SilhouettePlot of the CRAN package DataVisualizations.

Additionally you can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Value

Cls [1:n] vector with selected classes of the bestmatches. You can use plotTopographicMap(Umatrix,Bestmatches,Cls) for verification.

Note

If you used pswarm with distance matrix instead of a data matrix you can mds transform your distances into data. The correct dimension can be found through the Sheppard diagram or kruskals stress.

Often it is better to mark the outliers manually after the prozess of clustering; use in this case the visualization plotTopographicMap of the package GeneralizedUmatrix. If you would like to mark the outliers interactivly in the visualization use the interactiveClustering function.

Author(s)

Michael Thrun

References

[Thrun/Ultsch, 2017] Thrun, M.C., Ultsch, A.: Projection based Clustering, Conf. Int. Federation of Classification Societies (IFCS),DOI:10.13140/RG.2.2.13124.53124, Tokyo, 2017.

[Thrun, 2018] Thrun, M. C.: Projection Based Clustering through Self-Organization and Swarm Intelligence, doctoral dissertation 2017, Springer, Heidelberg, ISBN: 978-3-658-20539-3, https://doi.org/10.1007/978-3-658-20540-9, 2018.

Examples

```
data('Hepta')
#2d projection
projectionpoints=NeRV(Hepta$Data)
#Computation of Generalized Umatrix
library(GeneralizedUmatrix)
visualization=GeneralizedUmatrix(Data = Hepta$Data,projectionpoints)
# Visualizuation of GeneralizedUmatrix
library(GeneralizedUmatrix)
TopviewTopographicMap(visualization$Umatrix,visualization$Bestmatches)
#or in 3D if rgl package exists
#plotTopographicMap(visualization$Umatrix,visualization$Bestmatches)
# Automatic Clustering
LC=c(visualization$Lines,visualization$Columns)
# number of Cluster from dendrogram or visualization (PlotIt=T)
Cls=ProjectionBasedClustering(k=7, Hepta$Data,

visualization$Bestmatches, LC,PlotIt=TRUE)
# Verification
TopviewTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)
#or in 3D if rgl package exists
#plotTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)
```

`ProjectionPursuit` *Projection Pursuit*

Description

In the absence of a generative model for the data the algorithm can be used to find the projection pursuit directions. Projection pursuit is a technique for finding 'interesting' directions in multidimensional datasets

Usage

```
ProjectionPursuit(Data,OutputDimension=2,Indexfunction="logcosh",

Alpha=1,Iterations=200,PlotIt=FALSE,Cls)
```

Arguments

`Data`	array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
`OutputDimension`	Number of dimensions in the Outputspace, default=2
`Indexfunction`	Criterium for Minimization: default: 'logcosh' G(u)=1/a*log cosh(a*u) (ICA) 'exp': G(u)=-exp(u^2/2) 'kernel' 1/(1* pi)*exp(r/2)

Alpha	constant with 1<=alpha<=2 used in approximation to neg-entropy when fun == "logcosh"
Iterations	maximum number of iterations to perform.
PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown
Cls	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

ProjectedPoints
: [1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projectio

Note

You can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Author(s)

Michael Thrun

SammonsMapping *Sammons Mapping*

Description

Improved MDS thorugh a normalization of the Input space

Usage

```
SammonsMapping(DataOrDists,method='euclidean',OutputDimension=2,PlotIt=FALSE,Cls)
```

Arguments

DataOrDists	array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
method	method specified by distance string: 'euclidean','cityblock=manhatten','cosine','chebychev','jaccard','m
OutputDimension	Number of dimensions in the Outputspace, default=2

PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization.
Cls	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

ProjectedPoints	[1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projectio
Stress	Shephard-Kruskal Stress

Note

A wrapper for sammon

You can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Author(s)

Michael Thrun

ShortestGraphPathsC *Shortest GraphPaths = geodesic distances*

Description

Dijkstra's SSSP (Single source shortest path) algorithm, from all points to all points

Usage

```
ShortestGraphPathsC(Adj, Cost)
```

Arguments

Adj	[1:n,1:n] 0/1 adjascency matrix, e.g. from delaunay graph or gabriel graph
Cost	[1:n,1:n] matrix, distances between n points (normally euclidean)

Details

Vertices are the points, edges have the costs defined by weights (normally a distance). The algorithm runs in runs in O(n*E*Log(V)), see also [Jungnickel, 2013, p. 87]. Further details can be foubd in [Jungnickel, 2013, p. 83-87].

Value

ShortestPaths[1:n,1:n] vector, shortest paths (geodesic) to all other vertices including the source vertice itself from al vertices to all vertices, stored as a matrix

Note

require C++11 standard (set flag in Compiler, if not set automatically)

Author(s)

Michael Thrun

References

[Dijkstra, 1959] Dijkstra, E. W.: A note on two problems in connexion with graphs, Numerische mathematik, Vol. 1(1), pp. 269-271. 1959.

[Jungnickel, 2013] Jungnickel, D.: Graphs, networks and algorithms, (4th ed ed. Vol. 5), Berlin, Heidelberg, Germany, Springer, ISBN: 978-3-642-32278-5, 2013.

[Thrun/Ultsch, 2017] Thrun, M.C., Ultsch, A.: Projection based Clustering, Conf. Int. Federation of Classification Societies (IFCS),DOI:10.13140/RG.2.2.13124.53124, Tokyo, 2017.

See Also

DijkstraSSSP

---

tSNE *T-distributed Stochastic Neighbor Embedding*

Description

T-distributed Stochastic Neighbor Embedding res = tSNE(Data, KNN=30,OutputDimension=2)

Usage

```
tSNE(DataOrDists,k,OutputDimension=2,Algorithm='tsne_cpp',

method="euclidean",Whitening=FALSE, Iterations=1000,PlotIt=FALSE,Cls,...)
```

Arguments

DataOrDists	array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
k	number of k nearest neighbors=number of effective nearest neighbors("perplexity") Important parameter, if not given Settings of package t-SNE will be used
OutputDimension	Number of dimensions in the Outputspace, default=2

Algorithm	tsne_cpp': T-Distributed Stochastic Neighbor Embedding using a Barnes-HutImplementation in C++ of **Rtsne** 'tsne_r': pure R implementation of the t-SNE algorithm of of **tsne**
method	method specified by distance string: 'euclidean','cityblock=manhatten','cosine','chebychev','jaccard','m
Whitening	A boolean value indicating whether the matrix data should be whitened (tsne_r) or if pca should be used priorly (tsne_cpp)
Iterations	maximum number of iterations to perform.
PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown
Cls	[1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classification in numbers: every element is the cluster number of a certain corresponding element of data.
...	Further arguments passed on to either 'Rtsne' or 'tsne'

Details

An short overview of different types of projection methods can be found in [Thrun, 2018, p.42, Fig. 4.1] (https://doi.org/10.1007/978-3-658-20540-9).

Value

List of

ProjectedPoints	[1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projection
ModelObject	NULL for tsne_r, further information if tsne_cpp is selected

Note

A wrapper for Rtsne

or for tsne

You can use the standard ShepardScatterPlot or the better approach through the ShepardDensityPlot of the CRAN package DataVisualizations.

Author(s)

Michael Thrun

Examples

```
data('Hepta')
projection=tSNE(Hepta$Data,PlotIt=TRUE,Cls=Hepta$Cls)
```

Index