Package 'GeneralizedUmatrix'

May 30, 2023

Type Package

Title Credible Visualization for Two-Dimensional Projections of Data

Version 1.2.6

Date 2023-05-30

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

Description Projections are common dimensionality reduction methods, which represent high-dimensional data in a two-dimensional space. However, when restricting the output space to two dimensions, which results in a two dimensional scatter plot (projection) of the data, low dimensional similarities do not represent high dimensional distances coercively [Thrun, 2018] <DOI:10.1007/978-3-658-20540-9>. This could lead to a misleading interpretation of the underlying structures [Thrun, 2018]. By means of the 3D topographic map the generalized Umatrix is able to depict errors of these two-dimensional scatter plots. The package is derived from the book of Thrun, M.C.: ``Projection Based Clustering through Self-Organization and Swarm Intelligence'' (2018) <DOI:10.1007/978-3-658-20540-9> and the main algorithm called simplified self-organizing map for dimensionality reduction methods is published in <DOI:10.1016/j.mex.2020.101093>.

License GPL-3

Imports Rcpp (>= 1.0.8), RcppParallel (>= 5.1.4), ggplot2

Suggests DataVisualizations, rgl, grid, mgcv, png, reshape2, fields, ABCanalysis, plotly, deldir, methods, knitr (>= 1.12), rmarkdown (>= 0.9)

LinkingTo Rcpp, RcppArmadillo, RcppParallel

Depends R (>= 3.0)

NeedsCompilation yes

SystemRequirements GNU make, pandoc (>=1.12.3, needed for vignettes)

LazyLoad yes

LazyData TRUE

URL https://www.deepbionics.org

Encoding UTF-8

VignetteBuilder knitr

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

Author Michael Thrun [aut, cre, cph] (<https://orcid.org/0000-0001-9542-5543>),
Felix Pape [ctb, ctr],
Tim Schreier [ctb, ctr],
Luis Winckelman [ctb, ctr],
Quirin Stier [ctb, ctr],
Alfred Ultsch [ths]

Repository CRAN

Date/Publication 2023-05-30 10:20:09 UTC

R topics documented:

`GeneralizedUmatrix-package`

Credible Visualization for Two-Dimensional Projections of Data

---

Description

Projections are common dimensionality reduction methods, which represent high-dimensional data in a two-dimensional space. However, when restricting the output space to two dimensions, which results in a two dimensional scatter plot (projection) of the data, low dimensional similarities do not represent high dimensional distances coercively [Thrun, 2018] <DOI: 10.1007/978-3-658-20540-9>. This could lead to a misleading interpretation of the underlying structures [Thrun, 2018]. By means of the 3D topographic map the generalized Umatrix is able to depict errors of these two-dimensional scatter plots. The package is derived from the book of Thrun, M.C.: "Projection Based Clustering through Self-Organization and Swarm Intelligence" (2018) <DOI:10.1007/978-3-658-20540-9> and the main algorithm called simplified self-organizing map for dimensionality reduction methods is published in <DOI: 10.1016/j.mex.2020.101093>.

Details

For a brief introduction to **GeneralizedUmatrix** please see the vignette Introduction of the Generalized Umatrix Package.

For further details regarding the generalized Umatrix see [Thrun, 2018], chapter 4-5, or [Thrun/Ultsch, 2020].

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

Index of help topics:

```
CalcUstarmatrix         Calculate the U*matrix for a given Umatrix and
                        Pmatrix.
Chainlink               Chainlink is part of the Fundamental Clustering
                        Problem Suit (FCPS) [Thrun/Ultsch, 2020].
DefaultColorSequence    Default color sequence for plots
Delta3DWeightsC         intern function
EsomNeuronsAsList       Converts wts data (EsomNeurons) into the list
                        form
ExtendToroidalUmatrix   Extend Toroidal Umatrix
GeneralizedUmatrix      Generalized U-Matrix for Projection Methods
                        published in [Thrun/Ultsch, 2020]
GeneralizedUmatrix-package
                        Credible Visualization for Two-Dimensional
                        Projections of Data
GeneratePmatrix         Generates the P-matrix
ListAsEsomNeurons       Converts List to WTS
LowLand                 LowLand
NormalizeUmatrix        Normalize Umatrix
ReduceToLowLand         ReduceToLowLand
TopviewTopographicMap   Top view of the topographic map in 2D
Uheights4Data           Uheights4Data
UmatrixColormap         U-Matrix colors
UniqueBestMatchingUnits
                        UniqueBestMatchingUnits
XYcoords2LinesColumns   XYcoords2LinesColumns(X,Y) Converts points
```

	`given as x(i),y(i) coordinates to integer coordinates Columns(i),Lines(i)`
`addRowWiseC`	`intern function`
`plotTopographicMap`	`Visualizes the generalized U-matrix in 3D`
`sESOM4BMUs`	`simplified ESOM`
`setdiffMatrix`	`setdiffMatrix shortens Matrix2Curt by those rows that are in both matrices.`
`trainstepC`	`internal function for s-esom`
`upscaleUmatrix`	`Upscale a Umatrix grid`

Author(s)

Michal Thrun

Maintainer: Michael Thrun <mthrun@informatik.uni-marburg.de>

References

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A.: Uncovering High-Dimensional Structures of Projections from Dimensionality Reduction Methods, MethodsX, Vol. 7, pp. 101093, DOI doi:10.1016/j.mex.2020.101093, 2020.

[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, doi:10.1007/9783658205409, 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.

Examples

```
data("Chainlink")
Data=Chainlink$Data
Cls=Chainlink$Cls
InputDistances=as.matrix(dist(Data))
res=cmdscale(d=InputDistances, k = 2, eig = TRUE, add = FALSE, x.ret = FALSE)
ProjectedPoints=as.matrix(res$points)
#see also ProjectionBasedClustering package for other common projection methods
#see DatabionicSwarm for projection method without parameters or objective function
# ProjectedPoints=DatabionicSwarm::Pswarm(Data)$ProjectedPoints

resUmatrix=GeneralizedUmatrix(Data,ProjectedPoints)
plotTopographicMap(resUmatrix$Umatrix,resUmatrix$Bestmatches,Cls)

##Interactive Island Generation
## from a tiled Umatrix (toroidal assumption)
## Not run:
Imx = ProjectionBasedClustering::interactiveGeneralizedUmatrixIsland(resUmatrix$Umatrix,
resUmatrix$Bestmatches)
plotTopographicMap(resUmatrix$Umatrix,

resUmatrix$Bestmatches, Imx = Imx)
```

```
## End(Not run)
#External Verification
## Not run:

 DataVisualizations::Heatmap(Data,Cls)
 #if spherical cluster strcuture
 DataVisualizations::SilhouettePlot(Data,Cls)

## End(Not run)
```

addRowWiseC *intern function*

Description

Adds the Vector DataPoint to every row of the matrix WeightVectors

Usage

```
addRowWiseC(WeightVectors,DataPoint)
```

Arguments

`WeightVectors`	WeightVectors. n weights with m components each
`DataPoint`	Vector with m components

Value

`WeightVectors`	[1:m,1:n]

CalcUstarmatrix *Calculate the U*matrix for a given Umatrix and Pmatrix.*

Description

Calculate the U*matrix for a given Umatrix and Pmatrix.

Arguments

`Umatrix`	[1:Lines,1:Column] Local averages of distances at each point of the trainedGridWts[1:Lines,1:Column,1:variables] of ESOM or other SOM of same format
`Pmatrix`	[1:Lines,1:Column] Local densities at each point of the trainedGridWts[1:Lines,1:Column,1:variables] of ESOM or other SOM of same format.

Value

`UStarMatrix`	[1:Lines,1:Column]

Author(s)

Michael Thrun

References

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

`Chainlink` *Chainlink is part of the Fundamental Clustering Problem Suit (FCPS) [Thrun/Ultsch, 2020].*

Description

linear not separable dataset of two interwined chains.

Usage

```
data("Chainlink")
```

Details

Size 1000, Dimensions 3, stored in `Chainlink$Data`

Teo clusters, stored in `Chainlink$Cls`

Published in [Ultsch et al.,1994] in German and [Ultsch 1995] in English.

References

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A.: Clustering Benchmark Datasets Exploiting the Fundamental Clustering Problems, Data in Brief,Vol. 30(C), pp. 105501, DOI 10.1016/j.dib.2020.105501 , 2020.

[Ultsch 1995] Ultsch, A.: Self organizing neural networks perform different from statistical k-means clustering, Proc. Society for Information and Classification (GFKL), Vol. 1995, Basel 8th-10th March, 1995.

[Ultsch et al.,1994] Ultsch, A., Guimaraes, G., Korus, D., & Li, H.: Knowledge extraction from artificial neural networks and applications, Parallele Datenverarbeitung mit dem Transputer, pp. 148-16Chainlink, Springer, 1994.

Examples

```
data(Chainlink)
str(Chainlink)

## Not run:
require(DataVisualizations)
DataVisualizations::Plot3D(Chainlink$Data,Chainlink$Cls)

## End(Not run)
```

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.

Delta3DWeightsC *intern function*

Description

Thr implementation of the main formula of SOM, ESOM, sESOM algorithms.

Usage

```
Delta3DWeightsC(vx,Datasample)
```

Arguments

`vx`	Numeric array of weights [1:Lines,1:Columns,1:Weights]
`Datasample`	Numeric vector of one datapoint[1:n]

Details

intern function in case of `ComputeInR==FALSE` in `GeneralizedUmatrix`

Value

modified array of weights [1:Lines,1:Columns,1:Weights]

Author(s)

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, doi:10.1007/9783658205409, 2018.

`EsomNeuronsAsList` *Converts wts data (EsomNeurons) into the list form*

Description

Converts wts data into the list form

Arguments

`EsomNeurons`	[1:Lines, 1:Columns, 1:Variables] high dimensional array with grid positions in the first two dimensions.

Details

One could describe this function as a transformation or a special case of wide to long format, see also `ListAsEsomNeurons`

Value

`TrainedNeurons`	[1:(Lines*Columns),1:Variables] List of Weights as a matrix (not `list` like in R) as matrix or two dimensional array

Author(s)

Michael Thrun, Florian Lerch

References

Ultsch, A. Maps for the visualization of high-dimensional data spaces. in Proc. Workshop on Self organizing Maps. 2003.

`ExtendToroidalUmatrix` *Extend Toroidal Umatrix*

Description

Extends Umatrix by toroidal continuation of the given Umatrix defined by `ExtendBorders` in all four directions.

Usage

```
ExtendToroidalUmatrix(Umatrix, Bestmatches, ExtendBorders)
```

Arguments

`Umatrix`	[1:Lines,1:Columns] Matrix of Umatrix Heights
`Bestmatches`	[1:n, 1:2] Matrix with positions of Bestmatches for n datapoints, first columns is the position in `Lines` and second column in `Columns`
`ExtendBorders`	number of lines and columns the umatrix should be extended with

Details

Function assumes that U-matrix is not planaer (has no borders), i.e. is toroidal, and not tiled. Bestmatches are moved to new positions accordingly. Example is shown in conference talk of [Thrun et al., 2020].

Value

`Umatrix`	[1:Lines+2*ExtendBorders,1:Columns+2*ExtendBorders] Matrix of U-Heights
`Bestmatches`	Array with positions of Bestmatches

Note

Currently can be only used if untiled U-Matrix (the default) is presented, but 4-tiled U-matrix does not work.

Author(s)

Michael Thrun

References

[Thrun et al., 2020] Thrun, M. C., Pape, F., & Ultsch, A.: Interactive Machine Learning Tool for Clustering in Visual Analytics, 7th IEEE International Conference on Data Science and Advanced Analytics (DSAA 2020), Vol. accepted, pp. 1-9, IEEE, Sydney, Australia, 2020.

Examples

```
#ToDO
```

`GeneralizedUmatrix` *Generalized U-Matrix for Projection Methods published in [Thrun/Ultsch, 2020]*

Description

Generalized U-Matrix visualizes high-dimensional distance and density based structurs in two-dimensional scatter plots of projectios methods like CCA, MDS, PCA or NeRV [Ultsch/Thrun, 2017] with the help of a topographic map with hypsometrioc tints [Thrun et al. 2016] using a simplified emergent SOM published in [Thrun/Ultsch, 2020].

Usage

```
GeneralizedUmatrix(Data,ProjectedPoints,

PlotIt=FALSE,Cls=NULL,Toroid=TRUE, Tiled=FALSE,

ComputeInR=FALSE,Parallel=TRUE,DataPerEpoch=1,...)
```

Arguments

`Data` [1:n,1:d] array of data: n cases in rows, d variables in columns

`ProjectedPoints` [1:n,2] matrix containing coordinates of the Projection: A matrix of the fitted configuration.

`PlotIt` Optional,bool, defaut=FALSE, if =TRUE: U-Marix of every current Position of Databots will be shown However, the amount of details shown will be less than in `plotTopographicMap`.

`Cls` Optional, For plotting, see `plotUmatrix` in package Umatrix

`Toroid` Optional, Default=TRUE,
==FALSE planar computation with borders defined by projection method
==TRUE: toroid borderless (toroidal) computation, the four borders defined by projection method are ignored.

`Tiled` Optional,For plotting see `plotUmatrix` in package Umatrix

`ComputeInR` Optional, =T: Rcode, =F Cpp Code

`Parallel` Optional, =TRUE: compute parallel Cpp Code, =FALSE do not compute parallel Cpp Code

`DataPerEpoch` Optional, scalar, value above zero and below 1 starts sampling and defines percentage of data points sampled in each epoch during the learning phase. Beware: Experimental!

`...` Further parameters.

Details

Introduced first in the PhD thesis in [Thrun, 2018, p.46]. Furthermore the two parts of the work were peer-reviewed and published in [Ultsch/Thrun, 2017, Thrun/Ultsch, 2020].

Value

List with

`Umatrix` [1:Lines,1:Columns] Umatrix to be plotted, numerical matrix storing the U-heights, see [Thrun, 2018] for definition.

`EsomNeurons` [1:Lines,1:Columns,1:weights] 3-dimensional numeric array (wide format), not wts (long format).

`Bestmatches` [1:n,1:2] Positions of GridConverted Projected Points on the Umatrix to the predefined Grid by Lines and Columns, First Columns has the content of the Line No and second Column of the Column number.

`sESOMparamaters` internals for debugging

`Lines` Number of Lines

`Columns` Number of Columns

`gplotres` output of ggplot2

Author(s)

Michael Thrun

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 (WSCG), Vol. 24, Plzen, http://wscg.zcu.cz/wscg2016/short/A43-full.pdf, 2016.

[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, doi:10.1007/9783658205409, 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, 2020] Thrun, M. C., & Ultsch, A.: Uncovering High-Dimensional Structures of Projections from Dimensionality Reduction Methods, MethodsX, Vol. 7, pp. 101093, DOI doi:10.1016/j.mex.2020.101093, 2020.

Examples

```
data("Chainlink")
Data=Chainlink$Data
Cls=Chainlink$Cls
InputDistances=as.matrix(dist(Data))
res=cmdscale(d=InputDistances, k = 2, eig = TRUE, add = FALSE, x.ret = FALSE)
ProjectedPoints=as.matrix(res$points)
## Not run:
Stress = ProjectionBasedClustering::KruskalStress(InputDistances,
as.matrix(dist(ProjectedPoints)))

## End(Not run)


resUmatrix=GeneralizedUmatrix(Data,ProjectedPoints)
plotTopographicMap(resUmatrix$Umatrix,resUmatrix$Bestmatches,Cls)
```

`GeneratePmatrix` *Generates the P-matrix*

Description

Generates a P-matrix too visualize only density based structures of high-dimensional data.

Arguments

`Data`	[1:n,1:d], A `[n,d]` matrix containing the data
`EsomNeurons`	[1:Lines,Columns,1:Weights] 3D array of weights given by ESOM or sESOM algorithm.
`Radius`	The radius for measuring the density within the hypersphere.
`PlotIt`	If set the Pmatrix will also be plotted
`...`	If set the Pmatrix will also be plotted

Details

To set the Radius the ABCanalysis of high-dimensional distances can be used [Ultsch/Lötsch, 2015]. For a deteailed definition and equation of automated density estimation (Radius) see Thrun et al. 2016.

Value

PMatrix [1:Lines,1:Columns]

Author(s)

Michael Thrun

References

Ultsch, A.: Maps for the visualization of high-dimensional data spaces, Proc. Workshop on Self organizing Maps (WSOM), pp. 225-230, Kyushu, Japan, 2003.

Ultsch, A., Loetsch, J.: Computed ABC Analysis for Rational Selection of Most Informative Variables in Multivariate Data, PloS one, Vol. 10(6), pp. e0129767. doi 10.1371/journal.pone.0129767, 2015.

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.

ListAsEsomNeurons *Converts List to WTS*

Description

Converts wts data in list form into a 3 dimensional array

Arguments

`wts_list`	[1:(Lines*Columns),1:Variables] Matrix with weights in the 2nd dimension(not list() like in R)
`Lines`	Lines/Height of the desired grid
`Columns`	Columns/Width of the desired grid

Details

One could describe this function as a transformation or a special case of long to wide format, see also `EsomNeuronsAsList`

Value

`EsomNeurons`	[1:Lines, 1:Columns, 1:Variables] 3 dimensional array containing the weights of the neural grid. For a more general explanation see reference

Author(s)

Michael Thrun, Florian Lerch

References

Ultsch, A.: Maps for the visualization of high-dimensional data spaces, Proc. Workshop on Self organizing Maps (WSOM), pp. 225-230, Kyushu, Japan, 2003.

LowLand *LowLand*

Description

LowLand

Usage

```
LowLand(BestMatchingUnits, GeneralizedUmatrix, Data, Cls, Key, LowLimit)
```

Arguments

`BestMatchingUnits`	[1:n,1:n,1:n] BestMatchingUnits =[BMkey, BMLineCoords, BMColCoords]
`GeneralizedUmatrix`	[1:l,1:c] U-Matrix heights in Matrix form
`Data`	[1:n,1:d] data cases in lines, variables in Columns or [] or 0
`Cls`	[1:n] a possible classification of the data or [] or 0
`Key`	[1:n] the keys of the data or [] or 0
`LowLimit`	GeneralizedUmatrix heights up to this are considered to lie in the low lands default: LowLimit = prctile(Uheights,80) nur die 80# tiefsten

Value

`LowLandBM`	the unique BestMatchingUnits in the low lands of an u-Matrix
`LowLandInd`	index such that UniqueBM = BestMatchingUnits(UniqueInd,]
`LowLandData`	Data reduced to LowLand: LowLandData = Data(LowLandInd,]
`LowLandCls`	Cls reduced to LowLand: LowLandCls = Cls(LowLandInd)
`LowLandKey`	Key reduced to LowLand: LowLandKey = Key(LowLandInd)

Author(s)

ALU 2021 in matlab, MCT reimplemented in R

NormalizeUmatrix *Normalize Umatrix*

Description

Normalizing the U-matrix using the abstact U-Matrix concept [Loetsch/Ultsch, 2014].

Usage

```
NormalizeUmatrix(Data, Umatrix, BestMatches)
```

Arguments

`Data`	[1:n,1:d] numerical matrix of data with n cases and d variables
`Umatrix`	[1:lines,1:Columns] matrix of U-heights
`BestMatches`	[1:n,1:2] Bestmatching units.

Details

see publication [Loetsch/Ultsch, 2014]..

Value

Normalized Umatrix[1:lines,1:Columns] using the abstact U-Matrix concept.

Author(s)

Felix Pape, Michael Thrun

References

Loetsch, 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.

Examples

```
data("Chainlink")
Data=Chainlink$Data
Cls=Chainlink$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


  resUmatrix=GeneralizedUmatrix(Data,ProjectedPoints)
  ## Normalization
```

```
normalizedUmatrix=NormalizeUmatrix(Data,resUmatrix$Umatrix,resUmatrix$Bestmatches)
## visualization
TopviewTopographicMap(GeneralizedUmatrix = normalizedUmatrix,resUmatrix$Bestmatches)
```

plotTopographicMap *Visualizes the generalized U-matrix in 3D*

Description

The generalized U-matrix is visualized as the topographic map with hypsometric tints. The topographic map represents high-dimensional distance and density-based structurs in form of a 3D landscape.

Usage

```
plotTopographicMap(GeneralizedUmatrix, BestMatchingUnits,

Cls=NULL,ClsColors=NULL,Imx=NULL,Names=NULL,

BmSize=0.5,RenderingContourLines=TRUE,...)
```

Arguments

`GeneralizedUmatrix`
: [1:Lines,1:Columns] U-matrix to be plotted, numerical matrix storing the U-heights, see [Thrun, 2018] for definition.

`BestMatchingUnits`
: [1:n,1:2], Positions of bestmatches to be plotted as spheres onto the topographic map

`Cls`
: [1:n], numerical vector of classification of `k` clusters, one label for each best-match at that given point

`ClsColors`
: Vector of colors that will be used to colorize the different clusters, default is GeneralizedUmatrix::DefaultColorSequence

`Imx`
: a mask (Imx) that will be used to cut out the U-matrix

`Names`
: If set: [1:k] character vector naming the k clusters for the legend. In this case, further parameters with the possibility to adjust are: `NamesCex`: (size); `NamesPosition`: Legend position; `NamesTitle`: title of legend; `NamesColors`: colors if ClsColors are not default (NULL), etc.

`BmSize`
: size(diameter) of the points in the visualizations. The points represent the Best-MatchingUnits

`RenderingContourLines`
: FALSE: disables plotting of contour lines resulting in a much faster plot.

...	Besides the legend/names parameter the list of further parameters, use only of you know what you are doing:

- **Tiled** Should the U-matrix be drawn 4times?
- **ShowAxis** shall the axis be shown?
- **NoLevels** number of contour lines
- **ExtendBorders** scalar, extends U-matrix by toroidal continuation of the given U-matrix
- **Colormap** in the case of density p matrix...
- **title** same as main
- **main** same as title
- **sub** same as in `plot`
- **xlab** same as in `plot`
- **ylab** same as in `plot`
- **zlab** same as in `plot`
- **NamesPosition** same as in `bgplot3d`
- **NamesColors** same as col in `bgplot3d`
- **NamesCex** same as cex in `bgplot3d`
- **NamesTitle** same as title in `bgplot3d`
- **NamesPch** same as pch in `bgplot3d`

Details

The visualization of this function is a topographic map with hypsometric tints (Thrun, Lerch, L?tsch, & Ultsch, 2016). "Hypsometric tints are surface colors that represent ranges of elevation (Patterson and Kelso 2004). Here, contour lines are combined with a specific color scale. The color scale is chosen to display various valleys, ridges, and basins: blue colors indicate small distances (sea level), green and brown colors indicate middle distances (low hills), and white colors indicate vast distances (high mountains covered with snow and ice). Valleys and basins represent clusters, and the watersheds of hills and mountains represent the borders between clusters. In this 3D landscape, the borders of the visualization are cyclically connected with a periodicity (L,C). The number of clusters can be estimated by the number of valleys of the visualization. The clustering is valid if mountains do not partition clusters indicated by colored points of the same color and colored regions of points (see examples in section 4.1 and 4.2)."[Thrun/Ultsch, 2020].

A central problem in clustering is the correct estimation of the number of clusters. This is addressed by the topographic map which allows assessing the number of clusters as the number of valleys (Thrun et al., 2016). Please see chapter 5 of [Thrun, 2018] for further details.

Value

An object of class "htmlwidget" in mode invisible, please `rglwidget` for details.

Note

First version of algorithm was partly based on the U-matrix package.

Author(s)

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, doi:10.1007/9783658205409, 2018.

[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 (WSCG), Vol. 24, Plzen, http://wscg.zcu.cz/wscg2016/short/A43-full.pdf, 2016.

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A. : Using Projection based Clustering to Find Distance and Density based Clusters in High-Dimensional Data, Journal of Classification, DOI 10.1007/s00357-020-09373-2, in press, Springer, 2020.

See Also

GeneralizedUmatrix

Examples

```
data("Chainlink")
Data=Chainlink$Data
Cls=Chainlink$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

resUmatrix=GeneralizedUmatrix(Data,ProjectedPoints)
## visualization
plotTopographicMap(GeneralizedUmatrix = resUmatrix$Umatrix,resUmatrix$Bestmatches)


## Open window in specific resolution
#relevant if Names given

library(rgl)
r3dDefaults$windowRect = c(0,0,1200,1200)
plotTopographicMap(GeneralizedUmatrix = resUmatrix$Umatrix,resUmatrix$Bestmatches)

## Not run:
## To save as STL for 3D printing
 rgl::writeSTL("GenerelizedUmatrix_3d_model.stl")

## Save the visualization as a picture with
library(rgl)
rgl.snapshot('test.png')
```

```
## End(Not run)

## Save interactive html file
## Not run:
widgets=plotTopographicMap(GeneralizedUmatrix = resUmatrix$Umatrix,resUmatrix$Bestmatches)
if(requireNamespace("htmlwidgets"))
  htmlwidgets::saveWidget(widgets,file = "interactiveTopographicMap.html")

## End(Not run)
```

`ReduceToLowLand` *ReduceToLowLand*

Description

ReduceToLowLand

Usage

```
ReduceToLowLand(BestMatchingUnits, GeneralizedUmatrix, Data = NULL, Cls = NULL,
Key = NULL, LowLimit,Force=FALSE)
```

Arguments

`BestMatchingUnits`
[1:n,1:n,1:n] BestMatchingUnits =[BMkey, BMLineCoords, BMColCoords]

`GeneralizedUmatrix`
[1:l,1:c] U-Matrix heights in Matrix form

`Data` [1:n,1:d] data cases in lines, variables in Columns or [] or 0

`Cls` [1:n] a possible classif(ication of the data or [] or 0

`Key` [1:n] the keys of the data or [] or 0

`LowLimit` GeneralizedUmatrix heights up to this are considered to lie in the low lands default: LowLimit = prctile(Uheights,80) nur die 80# tiefsten

`Force` ==TRUE: Always perform reduction

Value

`LowLandBM` the unique BestMatchingUnits in the low lands of an u-Matrix

`LowLandInd` index such that UniqueBM = BestMatchingUnits(UniqueInd,]

`LowLandData` Data reduced to LowLand: LowLandData = Data(LowLandInd,]

`LowLandCls` Cls reduced to LowLand: LowLandCls = Cls(LowLandInd)

`LowLandKey` Key reduced to LowLand: LowLandKey = Key(LowLandInd)

Author(s)

ALU 2021 in matlab, MCT reimplemented in R

sESOM4BMUs *simplified ESOM*

Description

internfunction for the simplified ESOM Algorithmus [Thrun/Ultsch, 2020] for fixed BestMatchingUnits

Usage

```
sESOM4BMUs(BMUs,Data, esom, toroid,

CurrentRadius,ComputeInR=FALSE,Parallel=TRUE)
```

Arguments

`BMUs`	[1:Lines,1:Columns], BestMAtchingUnits generated by ProjectedPoints2Grid()
`Data`	[1:n,1:d] array of data: n cases in rows, d variables in columns
`esom`	[1:Lines,1:Columns,1:weights] array of NeuronWeights, see ListAsEsomNeurons()
`toroid`	TRUE/FALSE - topology of points
`CurrentRadius`	number betweeen 1 to x
`ComputeInR`	=T: Rcode, =F Cpp Code
`Parallel`	=T: Rcode, =F Cpp Code

Details

Algorithm is described in [Thrun, 2018, p. 48, Listing 5.1].

Value

`esom`	array [1:Lines,1:Columns,1:d], d is the dimension of the weights, the same as in the ESOM algorithm. modified esomneuros regarding a predefined neighborhood defined by a radius

Note

Usually not for seperated usage!

Author(s)

Michael Thrun

References

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A.: Uncovering High-Dimensional Structures of Projections from Dimensionality Reduction Methods, MethodsX, Vol. in press, pp. 101093. doi 10.1016/j.mex.2020.101093, 2020.

See Also

GeneralizedUmatrix

setdiffMatrix *setdiffMatrix shortens Matrix2Curt by those rows that are in both matrices.*

Description

setdiffMatrix shortens Matrix2Curt by those rows that are in both matrices.

Arguments

Matrix2Curt	[n,k] matrix, which will be shortened by x rows
Matrix2compare	[m,k] matrix whose rows will be compared to those of Matrix2Curt x rows in Matrix2compare equal rows of Matrix2Curt (order of rows is irrelevant). Has the same number of columns as Matrix2Curt.

Value

V$CurtedMatrix	[n-x,k] Shortened Matrix2Curt

Author(s)

Michael Thrun with the help of Catharina Lippmann

TopviewTopographicMap *Top view of the topographic map in 2D*

Description

Fast visualization of the generalized U-matrix in 2D which visualizes high-dimensional distance and density based structurs of the combination two-dimensional scatter plots (projections) with high-dimensional data.

Usage

```
TopviewTopographicMap(GeneralizedUmatrix, BestMatchingUnits,

Cls, ClsColors = NULL, Imx = NULL,

ClsNames = NULL, BmSize = 6, DotLineWidth = 2,

alpha = 1, ...)
```

Arguments

`GeneralizedUmatrix`
: [1:Lines,1:Columns] U-matrix to be plotted, numerical matrix storing the U-heights, see [Thrun, 2018] for definition.

`BestMatchingUnits`
: [1:n,1:2], Positions of bestmatches to be plotted onto the U-matrix

`Cls`
: [1:n], numerical vector of classification of `k` classes for the bestmatch at the given point

`ClsColors`
: Vector of colors that will be used to colorize the different classes

`Imx`
: a mask (Imx) that will be used to cut out the U-matrix

`ClsNames`
: If set: [1:k] character vector naming the k classes for the legend. In this case, further parameters with the possibility to adjust are: `LegendCex`: (size); `NamesOrientation`: Legend position "v" or "h"; `NamesTitle`: title of legend.

`BmSize`
: size(diameter) of the points in the visualizations. The points represent the BestMatchingUnits

`DotLineWidth`
: ...

`alpha`
: ...

`...`
: **Tiled** Should the U-matrix be drawn 4times?

  main set specific title in plot

  ExtendBorders scalar, extends U-matrix by toroidal continuation of the given U-matrix

  MainCex scalar, magnification to be used for legend

  LegendCex scalar, magnification to be used for main titles

  _ Further Arguments relevant for interactive shiny application

Details

Please see `plotTopographicMap`. This function is currently still experimental because not all functionallity is fully tested yet.

Value

plotly handler

Note

Names are currently under development, Imx in testing phase.

Author(s)

Tim Schreier, 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, doi:10.1007/9783658205409, 2018.

[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 (WSCG), Vol. 24, Plzen, http://wscg.zcu.cz/wscg2016/short/A43-full.pdf, 2016.

See Also

plotTopographicMap

Examples

```
data("Chainlink")
Data=Chainlink$Data
Cls=Chainlink$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

resUmatrix=GeneralizedUmatrix(Data,ProjectedPoints)
## visualization
TopviewTopographicMap(GeneralizedUmatrix = resUmatrix$Umatrix,resUmatrix$Bestmatches)
```

trainstepC *internal function for s-esom*

Description

Does the training for fixed bestmatches in one epoch of the sESOM.

Usage

```
trainstepC(vx,vy, DataSampled,BMUsampled,Lines,Columns, Radius, toroid)
```

Arguments

`vx` array [1:Lines,1:Columns,1:Weights], WeightVectors that will be trained, internally transformed von NumericVector to cube

`vy` array [1:Lines,1:Columns,1:2], meshgrid for output distance computation

`DataSampled` NumericMatrix, n cases shuffled Dataset[1:n,1:d] by `sample`

`BMUsampled` NumericMatrix, n cases shuffled BestMatches[1:n,1:2] by `sample` in the same way as `DataSampled`

`Lines` double, Height of the grid

`Columns` double, Width of the grid

`Radius` double, The current Radius that should be used to define neighbours to the bm

`toroid` bool, Should the grid be considered with cyclically connected borders?

Details

Algorithm is described in [Thrun, 2018, p. 48, Listing 5.1].

Value

WeightVectors, array[1:Lines,1:Columns,1:weights] with the adjusted Weights

Note

Usually not for seperated usage!

Author(s)

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, doi:10.1007/9783658205409, 2018.

`Uheights4Data` *Uheights4Data*

Description

Uheights4Data

Usage

```
Uheights4Data(BestMatchingUnits, GeneralizedUmatrix)
```

Arguments

`BestMatchingUnits`	[1:n,1:d] BMKey = BestMatchingUnits[,1)
`GeneralizedUmatrix`	[1:Lines,1:Columns] a GeneralizedUmatrix

Value

`Uheights`	Uheights
`BMLineCoords`	BMLineCoords
`BMColCoords`	BMColCoords

Author(s)

ALU 2021 in matlab, MCT reimplemented in

`UmatrixColormap` *U-Matrix colors*

Description

Defines the default color sequence for plots made for Umatrix

Usage

```
data("UmatrixColormap")
```

Format

Returns the vectors for a (heat) colormap.

`UniqueBestMatchingUnits` *UniqueBestMatchingUnits*

Description

UniqueBestMatchingUnits

Usage

```
UniqueBestMatchingUnits(NonUniqueBestMatchingUnits)
```

Arguments

`NonUniqueBestMatchingUnits`
[1:n,1:n,1:n] UniqueBestMatchingUnits =[BMkey, BMLineCoords, BMColCoords]

Value

`UniqueBM` [1:u,1:u,1:u] UniqueBM =[UBMkey, UBMLineCoords, UBMColCoords]

`UniqueInd` Index such that UniqueBM = UniqueBestMatchingUnits(UniqeInd,:)

`Uniq2AllInd` Index such that UniqueBestMatchingUnits = UniqueBM(Uniq2AllInd,:)

Author(s)

ALU 2021 in matlab, MCT reimplemented in R

`upscaleUmatrix` *Upscale a Umatrix grid*

Description

Use linear interpolation to increase the size of a umatrix. This can be used to produce nicer ggplot plots in `plotTopographicMap` and is going to be used for further normalization of the umatrix.

Usage

```
upscaleUmatrix(Umatrix, Factor = 2,BestMatches, Imx)
```

Arguments

`Umatrix` The umatrix which should be upscaled

`BestMatches` The BestMatches which should be upscaled

`Factor` Optional: The factor by which the axes will be scaled. Be aware that the size of the matrix will grow by Factor squared. Default: 2

`Imx` Optional: Island cutout of the umatrix. Should also be scaled to the new size of the umatrix.

Value

A List consisting of:

`Umatrix` A matrix representing the upscaled umatrix.

`BestMatches` If BestMatches was given as parameter: The rescaled BestMatches for an island cutout. Otherwise: `NULL`

`Imx` If Imx was given as parameter: The rescaled matrix for an island cutout. Otherwise: `NULL`

Author(s)

Felix Pape

`XYcoords2LinesColumns` *XYcoords2LinesColumns(X,Y) Converts points given as x(i),y(i) coordinates to integer coordinates Columns(i),Lines(i)*

Description

XYcoords2LinesColumns(X,Y) Converts points given as x(i),y(i) coordinates to integer coordinates Columns(i),Lines(i)

Arguments

`X`	[1:n] first coordinate: x(i), y(i) is the i-th point on a plane
`Y`	[1:n] second coordinate: x(i), y(i) is the i-th point on a plane
`minNeurons`	minimal size of the corresponding grid i.e max(Lines)*max(Columns)>=MinGridSize , default MinGridSize = 4096 defined by the numer of neurons
`MaxDifferentPoints`	TRUE: the discretization error is minimal FALSE: number of Lines and Columns is minimal
`PlotIt`	Plots the result
`na.rm`	if non finite values should be disregarded in the computation then set to TRUE

Details

Non finite values are not filtered out even if na.rm=TRUE, only ignored. Details are written down in [Thrun, 2018, p. 47].

Value

GridConvertedPoints[1:Columns,1:Lines,2] IntegerPositions on a grid corresponding to x,y

Author(s)

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, doi:10.1007/9783658205409, 2018.

Examples

```
data("Chainlink")
Data=Chainlink$Data
InputDistances=as.matrix(dist(Data))
res=cmdscale(d=InputDistances, k = 2, eig = TRUE, add = FALSE, x.ret = FALSE)
ProjectedPoints=as.matrix(res$points)
GridConvertedPoints=XYcoords2LinesColumns(ProjectedPoints[,1],ProjectedPoints[,2],PlotIt=FALSE)
```

Index