Package 'ProjectionBasedClustering'

June 14, 2024

Type Package

Title Projection Based Clustering

Version 1.2.2

Date 2024-06-14

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 based on Thrun and Ultsch, ``Using Projection based Clustering to Find Distance and Density based Clusters in High-Dimensional Data" <DOI:10.1007/s00357-020-09373-2>. 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, dendextend, umap, uwot, DatabionicSwarm, parallelDist, parallel

LinkingTo Rcpp

LazyData TRUE

Encoding UTF-8

SystemRequirements C++17

Depends R (>= 3.0)

NeedsCompilation yes

URL https://www.deepbionics.org

LazyLoad yes

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

Author Michael Thrun [aut, cre, cph],
Quirin Stier [ctb, rev] (<https://orcid.org/0000-0002-7896-4737>),
Brinkmann Luca [ctb],
Florian Lerch [aut],
Felix Pape [aut],
Tim Schreier [aut],
Luis Winckelmann [aut],
Kristian Nybo [cph],
Jarkko Venna [cph],
van der Maaten Laurens [cph]

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

Repository CRAN

Date/Publication 2024-06-14 17:50:01 UTC

Contents

`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>. and [Thrun/Ultsch, 2020]. The abstract of the conference talk is as 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.

Furthermore, [Thrun et al., 2021] presents an interactive parameter-free approach, that incorporates a human-in-the-loop, for projection-based clustering.

Details

A comparison to 32 common clustering algorithms is provided in [Thrun/Ultsch, 2020].

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/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, Vol. 38(2), pp. 280-312, Springer, DOI: 10.1007/s00357-020-09373-2, 2020.

[Thrun et al., 2021] Thrun, M. C., Pape, F. & Ultsch, A.: Conventional displays of structures in data compared with interactive projection-based clustering (IPBC), International Journal of Data Science and Analyitics, Vol. 12(3), pp. 249-271, Springer, DOI: 10.1007/s41060-021-00264-2, 2021

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 (CCA)*

Description

CCA Projects data vectors using Curvilinear Component Analysis [Demartines/Herault, 1995],[Demartines/Herault, 1997].

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(DataOrDistances,Epochs,OutputDimension=2,method='euclidean',

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

Arguments

DataOrDistances	Numerical matrix defined as either Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features, or Distances, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., as.matrix(dist(Data,method))
Epochs	Number of eppochs (scalar), i.e, 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] (doi:10.1007/9783658205409).

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.

You can use the standard Sheparddiagram or the better approach through the ShepardDensityScatter of the CRAN package DataVisualizations.

Author(s)

Florian Lerch

References

[Demartines/Herault, 1997] 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(1), pp. 148-154. 1997.

[Demartines/Herault, 1995] Demartines, P., & Herault, J.: CCA:" Curvilinear component analysis", Proc. 15 Colloque sur le traitement du signal et des images, Vol. 199, GRETSI, Groupe d'Etudes du Traitement du Signal et des Images, France 18-21 September, 1995.

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=CCA(Data,Epochs=20)

## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

`ContTrustMeasure` *continuity and trustworthiness*

Description

Computes trustworthiness and continuity for projected data (see [Kaski2003]).

Usage

```
ContTrustMeasure(datamat, projmat, lastNeighbor)
```

Arguments

`datamat`	numerical matrix of data: n cases in rows, d variables in columns
`projmat`	numerical matrix of projected data: n cases in rows, k variables in columns, where k is the projection output dimension
`lastNeighbor`	scalar, maximal size of neighborhood to be considered

Details

This is a wrapper that is used in the **DRquality** to investigate varius quality measurements [Thrun et al, 2023]. The paper indicates, that the Gabriel classification error seems to be a good alternative. [Thrun et al, 2023].

Value

numerical [k,7] matrix, where k is the lastNeighbor value. The matrix contains the columns:

Neighborhood size; worst-case trustworthiness; average trustworthiness; best-case trustworthiness; worst-case continuity; average continuity; best-case continuity

where neighborhood size is the size of the neighberhood considered, which ranges from 1:lastNeighbor

Note

C++ source code comes from https://research.cs.aalto.fi/pml/software/dredviz/

Author(s)

Luca Brinkmann, Felix Pape

References

[Kaski2003]: Samuel Kaski, Janne Nikkilä, Merja Oja, Jarkko Venna, Petri Törönen, and Eero Castren. Trustworthiness and metrics in visualizing similarity of gene expression. BMC Bioinformatics, 4:48, 2003.

See Also

For plotting see `plotMeasureTundD` in the package **DRquality**. An alternative measure is the KLMeasure, see also GabrielClassificationError

Examples

```
data('Hepta')
Data=Hepta$Data
res=MDS(Data)
Proj = res$ProjectedPoints
ContTrustMeasure(Hepta$Data, Proj, 10)
```

`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 https://ideone.com/qkmt31

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

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

[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.

Examples

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

ICA — *Independent Component Analysis (ICA)*

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	numerical matrix of n cases in rows, d variables in columns, matrix is not 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] (doi:10.1007/9783658205409).

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

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=ICA(Data)

## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

`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

Details

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

Value

List of

`EnlargedUmatrix`
: [1:Lines,1:Columns] Matrix of Umatrix Heights taken four times and arranged in a square 2x2.

`EnlargedBestmaches`
: [1:n,1:2] Array with positions of Bestmatches according to the enlarged umatrix.

`EnlargedCls`
: [1:n] Classification of the Bestmatches according to the enlarged umatrix.

`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

`TopView_TopographicMap`
: Plot of a topographic map.

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, doi:10.1007/9783658205409, 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, Plotter="plotly",NoLevels=NULL)
```

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
Cls	Classification of the Bestmatches
Plotter	Choose between plotting frameworks: "plotly" and "ggplot2"
NoLevels	number of contour lines in topographic map that will be done. NULL: uses default values, for plotly the defuult is 15, for ggplot2 the default is round(maxU2/max(minU2,0.05),0)

Details

The Imx is a matrix that overlays the 4-tiled (generalized) Umatrix to define an island within the four tiles. The Umatrix is computed first 4 times (i.e. within 4 tiles) to account for border effects. Then zeros mark which part of the Umatrix shall be shown to the user as a topogrpahic map and ones change the Umatrix values to zeros which will be visualized as an ocean. The result is an borderless island of high-dimensional structures. Usually the goal is to cut out the island in a way that mountain ranges define the borders of the island.

NoLevels also influences the number of colors used in the topographic map. In general, a lower number will result in faster plotting and therefore improve interactivity but lower the number of details that are visible.

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

Value

[1:2*Lines,1:2*Columns] Boolean Matrix that represents the island within the tiled Umatrix. Zeros mark the inside and ones the outside of the island.

Author(s)

Michael Thrun, Quirin Stier

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 (IPBC)

Description

An interactive clustering tool published in [Thrun et al., 2020] that uses 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 map [Thrun et al., 2016] of the generalized U-matrix [Ultsch/Thrun, 2017] combined with projected points. It includes capabilities for interactive clustering within the interface as well as automatic projection-based clustering based on [Thrun/Ultsch, 2020].

Usage

```
interactiveProjectionBasedClustering(Data, Cls=NULL)

IPBC(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. Please see also the note below.
Cls	Optional: Prior Classification of the data for the [1:n] cases of k classes.

Details

To cluster data interactively, i.e., select specific data points and create a cluster), first generate the visualization. Thereafter, switch in the menu to clustering, hold the left mouse button and then frame a valley. Simple mouse clicks will not start the lasso functionality of plotly.

The resulting clustering is stored in Cls which 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
Umatrix	[1:Lines,1:Columns] generalized Umatrix to be plotted, numerical matrix storing the U-heights, see [Thrun, 2018] for definition.
Bestmatches	[1:n,2] Matrix of GridConverted Projected Points [1:n, 1:2] called Bestmatches that defines positions for n datapoints, first columns is the position in Lines and second column in Columns
LastProjectionMethodUsed	name of last projection method that was used as a string
TopView_TopographicMap	The final plot generated by plot.ly when closing the tool

Note

Some dimensionality reduction methods will assume data without missing values, some other DR methods assume unique data points, i.e., no distance=0 for any two cases(rows) of data. In these cases the IPBC method will crash.

Author(s)

Tim Schreier, Felix Pape, Luis Winckelmann, Michael Thrun

References

[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/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, Springer, DOI: 10.1007/s00357-020-09373-2, 2020.

[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), pp. 672-680, DOI 10.1109/DSAA49011.2020.00062, IEEE, Sydney, Australia, 2020.

Examples

```
if(interactive()){
  data('Hepta')
  Data=Hepta$Data

  V=interactiveProjectionBasedClustering(Data)

  # with prior classification
  Cls=Hepta$Cls
  V=IPBC(Data,Cls)
}
```

`Isomap` *Isomap*

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(Distances,k,OutputDimension=2,PlotIt=FALSE,Cls)
```

Arguments

`Distances`	Symmetric [1:n,1:n] distance matrix, e.g. `as.matrix(dist(Data,method))`
`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] (doi:10.1007/9783658205409).

Value

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

Note

A wrapper enabling a planar projection of the manifold learning method based on the 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

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=Isomap(as.matrix(dist(Data)),k=7)

## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

KLMeasure *Smoothed Precision and Recall*

Description

Computes the quality measurement of rank-based smoothed precision an recall, with cost function based on Kullback-Leibler-divergence (see [Venna2010]) used to evaluated dimensionality reduction methods.

Usage

```
KLMeasure(Data, pData, NeighborhoodSize = 20L)
```

Arguments

`Data`	numerical matrix of data: n cases in rows, d variables in columns
`pData`	numerical matrix of projected data: n cases in rows, k variables in columns, where k is the projection output dimension
`NeighborhoodSize`	Number of points in neighborhood to be considered. Default is 20

Details

This is a wrapper that is used in the **DRquality** to investigate varius quality measurements [Thrun et al, 2023]. The paper indicates, that the Gabriel classification error seems to be a good alternative. [Thrun et al, 2023].

Value

`SmoothedPrecision`	Scalar, smoothed precision value
`SmoothedRecall`	Scalar, smoothed recall value

Note

C++ source code comes from https://research.cs.aalto.fi/pml/software/dredviz/

Author(s)

Luca Brinkmann, Felix Pape

References

[Venna2010]: 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.

[Thrun et al, 2023] Thrun, M.C, Märte, J., Stier, Q.: Analyzing Quality Measurements for Dimensionality Reduction, Machine Learning and Knowledge Extraction (MAKE), Vol 5., accepted, 2023.

See Also

An alternative measure is the ContTrustMeasure, see also GabrielClassificationError

Examples

```
data('Hepta')
Data=Hepta$Data
res=MDS(Data)
Proj = res$ProjectedPoints

kl_m = KLMeasure(Hepta$Data, Proj)
# Smoothed precision
```

```
print(kl_m[[1]])
# Smoothed recall
print(kl_m[[2]])
```

`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] (doi:10.1007/9783658205409).

Value

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

Author(s)

Felix Pape

`MDS` *Multidimensional Scaling (MDS)*

Description

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

Usage

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

Arguments

DataOrDistances	Numerical matrix defined as either Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features, or Distances, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., as.matrix(dist(Data,method))
method	method specified by distance string: 'euclidean','cityblock=manhatten','cosine','chebychev','jaccard','mi
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] (doi:10.1007/9783658205409).

Value

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

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=MDS(Data)

## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

NeRV *Neighbor Retrieval Visualizer (NeRV)*

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`	Numerical matrix of the Data to be projected, [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features
`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] (doi:10.1007/9783658205409).

Value

OutputDimension-dimensional matrix of projected points

Note

PCA initialization changes form the original C++ Sourcecode of https://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.

Examples

```
data('Hepta')
Data=Hepta$Data
## Not run:
Proj=NeRV(Data)
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

PCA *Principal Component Analysis (PCA)*

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	numerical matrix 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] (doi:10.1007/9783658205409).

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

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=PCA(Data)

## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

`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

PolarSwarm *Polar Swarm (Pswarm)*

Description

Swarm-based Projection method using game theory published in [Thrun/Ultsch, 2020].

Usage

```
PolarSwarm(DataOrDistances, method = "euclidean", PlotIt = FALSE, Cls)
```

Arguments

DataOrDistances	Numerical matrix defined as either Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features, or Distances, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., as.matrix(parallelDist::parallel
method	If Data is given the method to computing the distances can be specified here. Please see the documentation of package **parallelDist** for the types that are possible.
PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown
Cls	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

By exploting swarm intelligence and game theory no parameter have to be set.

Value

List of

ProjectedPoints	[1:n,2], n by 2 matrix containing coordinates of the Projection
ModelObject	output of Pswarm

Author(s)

Michael Thrun

References

[Thrun/Ultsch, 2020] Thrun, M. C., & Ultsch, A.: Swarm Intelligence for Self-Organized Clustering, Artificial intelligence, Vol. 290, pp. 103237, doi 10.1016/j.artint.2020.103237, 2020.

See Also

Pswarm

Examples

```
data('Hepta')
Data=Hepta$Data

Distances=as.matrix(dist(Data))
Proj=PolarSwarm(Data)
## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

Projection2Bestmatches

Projection to Bestmatches

Description

Transformation of projected points to bestmatches defined by generalized Umatrix

Usage

```
Projection2Bestmatches(ProjectedPoints)
```

Arguments

ProjectedPoints	[1:n,1:2] n projected points in two-dimensional space.

Details

It is assumed that an unambiguous assignment between projected points and data points is given.

Value

Bestmatches	[1:n,1:2] Positions of GridConverted Projected Points, which can be used for the generalized 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.
LC	[1:2] vector if Line No. and ColumnNo. which defines the size of the grid of the generalized Umatrix

Note

Details of the equations used are written down in [Thrun, 2018, p. 47].

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.

See Also

XYcoords2LinesColumns

Examples

```
data('Hepta')
ProjList=MDS(Hepta$Data)
trafo=Projection2Bestmatches(ProjList$ProjectedPoints)
```

ProjectionBasedClustering

Automatic Projection-based Clustering (PBC) [Thrun/Ultsch, 2020]

Description

Three steps are necessary for PBC. First, a projection method has to be chosen to generate projected points of high-dimensional data points. Second, the generalized U*-matrix has to be applied to the projected points by using a simplified emergent self-organizing map (ESOM) algorithm which is an unsupervised neural network [Thrun, 2018]. The resulting generalized U-matrix can be visualized by the topographic map [Thrun et al., 2016]. Third, the clustering itself is built on top of the generalized U-matrix using the concept of the abstract U-Matrix and shortest graph paths using ShortestGraphPathsC.

Usage

```
ProjectionBasedClustering(k, DataOrDistances, BestMatches, LC,

StructureType = TRUE, PlotIt = FALSE, method = "euclidean")
```

Arguments

k	number of clusters, how many to you see in the 3d landscape?
DataOrDistances	Numerical matrix that will be used for clustering with one DataPoint per row, defined as either as Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features,

	or `Distances`, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., `as.matrix(dist(Data,method))`
`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 used in **parallelDist** if `Data` given.

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. The visualization combines projected points with a topographic map with hypsometric colors, defined by the generalized U-matrix (see package GeneralizedUmatrix function GeneralizedUmatrix).

The clustering method with no sensitive parameters is done in this function and the algorithm is introduced in detail in [Thrun/Ultsch, 2020]. 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`.

If **parallelDist** is not installed, function automatically falls back to `dist`.

Value

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

Note

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 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.

[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/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, Vol. 38(2), pp. 280-312, Springer, DOI: 10.1007/s00357-020-09373-2, 2020.

Examples

```
data('Hepta')
#Step I: 2d projection
projectionpoints=NeRV(Hepta$Data)

#Step II (Optional): 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)

# Step III: Automatic Clustering
trafo=Projection2Bestmatches(projectionpoints)
# number of Cluster from dendrogram  (PlotIt=T) or visualization above
Cls=ProjectionBasedClustering(k=7, Hepta$Data,

trafo$Bestmatches, trafo$LC,PlotIt=TRUE)

# Verification of Clustering
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] (doi:10.1007/9783658205409).

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

Arguments

DataOrDistances	Numerical matrix defined as either Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features, or Distances, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., as.matrix(dist(Data,method))
method	method specified by distance string: 'euclidean','cityblock=manhatten','cosine','chebychev','jaccard','mi
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] (doi:10.1007/9783658205409).

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

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=SammonsMapping(Data)
```

```
## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

`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 (t-SNE)*

Description

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

Usage

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

method="euclidean",Whitening=FALSE, Iterations=1000,PlotIt=FALSE,Cls,num_threads=1,...)
```

Arguments

DataOrDistances	Numerical matrix defined as either Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features, or Distances, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., as.matrix(dist(Data,method))
k	number of k nearest neighbors=number of effective nearest neighbors("perplexity"); Important parameter. If not given, settings of packages of t-SNE will be used depending Algorithm
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**. Requires Version >= 0.15 of **Rtsne** for multicore parallelisation. 'tsne_opt_cpp': T-Distributed Stochastic Neighbor Embedding with automated optimized parameters using a Barnes-HutImplementation in C++ of [Ulyanov, 2016]. '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.
num_threads	Number of threads for parallel computation, only usable for Algorithm='tsne_cpp' or 'tsne_opt_cpp'
...	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], doi:10.1007/9783658205409.

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 (Algorithm='tsne_cpp'),

Multicore-opt-tSNE (Algorithm='tsne_opt_cpp'),

or for tsne (Algorithm='tsne_r')

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

Author(s)

Michael Thrun, Luca Brinkmann

References

Anna C. Belkina, Christopher O. Ciccolella, Rina Anno, Josef Spidlen, Richard Halpert, Jennifer Snyder-Cappione: Automated optimal parameters for T-distributed stochastic neighbor embedding improve visualization and allow analysis of large datasets, bioRxiv 451690, doi: https://doi.org/10.1101/451690, 2018.

L.J.P van der Maaten: Accelerating t-SNE using tree-based algorithms, Journal of Machine Learning Research 15.1:3221-3245, 2014.

Ulyanov, Dmitry: Multicore-TSNE, GitHub repository URL https://github.com/DmitryUlyanov/Multicore-TSNE, 2016.

Examples

```
data('Hepta')
Data=Hepta$Data

## Not run:
Proj=tSNE(Data,k=7)

PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

UniformManifoldApproximationProjection

Uniform Manifold Approximation and Projection

Description

Uniform manifold approximation and projection is a technique for dimension reduction. The algorithm was described by [McInnes et al., 2018].

Usage

```
UniformManifoldApproximationProjection(DataOrDistances, k,

Epochs,OutputDimension=2,Algorithm='umap_pkg',PlotIt=FALSE,Cls,...)
```

Arguments

DataOrDistances	Numerical matrix defined as either Data, i.e., [1:n,1:d], nonsymmetric, and consists of n cases of d-dimensional data points with every case having d attributes, variables or features, or Distances, i.e.,[1:n,1:n], symmetric and consists of n cases, e.g., as.matrix(dist(Data,method))
k	number of k nearest neighbors, Important parameter, if not given, settings of package **umap** will be used, default of package **umap** is currently 15
Epochs	Number of eppochs (scalar), i.e, training length, default of package **umap** is currently 200
OutputDimension	Number of dimensions in the Outputspace, default=2
Algorithm	"umap_pkg": provides an interface for two implementations. One is written from scratch other one requires python **umap** "uwot_pkg": complete re-implementation in R (and C++, via the 'Rcpp' package) of **uwot**

PlotIt	Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown
Cls	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.
...	one of the other 21 parameters that can be specified, please see umap.defaults of package **umap** for details or parameters to be set in package **uwot** depending on the choice of Algorithm.

Details

To the knowledge of the author of this function no peer-reviewed publication of the method exists. Use with greate care.

Value

List of

ProjectedPoints	[1:n,OutputDimension], n by OutputDimension matrix containing coordinates of the Projection
ModelObject	output of umap or of package **uwot** depending on Algorithm
Setting	specific settings used in UniformManifoldApproximationProjection

Note

Uniform Manifold Approximation and Projection and U-matrix [Ultsch/Siemon, 1990] are both sometimes abbreviated with Umap. Hence the abbreveviation is omitted here.

Author(s)

Michael Thrun

References

[McInnes et al., 2018] McInnes, L., Healy, J., & Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction, arXiv preprint arXiv:1802.03426, 2018.

[Ultsch/Siemon, 1990] Ultsch, A., & Siemon, H. P.: Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis, International Neural Network Conference, pp. 305-308, Kluwer Academic Press, Paris, France, 1990.

See Also

umap of **umap**

umap of **uwot**

Examples

```
data('Hepta')
Data=Hepta$Data

Proj=UniformManifoldApproximationProjection(Data)

## Not run:
PlotProjectedPoints(Proj$ProjectedPoints,Hepta$Cls)

## End(Not run)
```

Index