Package ‘ProjectionBasedClustering’

August 21, 2017

Type  Package
Title  Projection Based Clustering
Version 1.0.4
Date 2017-08-21
Description A clustering approach for every projection method based on the generalized U*-matrix visualization of a topographic map is made available here [Thrun/Ultsch,2017] <DOI:10.13140/RG.2.2.13124.53124>. The number of clusters and the cluster structure can be estimated by counting the valleys in a topographic map. If the number of clusters and the clustering method are chosen correctly, then the clusters will be well separated by mountains in the visualization. Most projection methods are wrappers for already available methods in R. However, 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
Suggests fastICA, tsne, FastKNN, MASS, pcaPP, spdep, methods, pracma, grid, mgcv, fields, png, reshape2
LinkingTo Rcpp
LazyData TRUE
Encoding UTF-8
SystemRequirements C++11
Depends R (>= 3.0)
NeedsCompilation yes
URL https://www.uni-marburg.de/fb12/datenbionik/software-en
LazyLoad yes
Author Michael Thrun [aut, cre],
     Florian Lerch [aut],
     Felix Pape [aut],
     Kristian Nybo [cph],
     Jarkko Venna [cph]
The package is based on a conference talk [Thrun/Ultsch, 2017], see <DOI:10.13140/RG.2.2.13124.53124>. The abstract follows:

Many data mining methods rely on some concept of the dissimilarity between pieces of information encoded in the data of interest. These methods can be used for cluster analysis. However, no generally accepted definition of clusters exists in the literature [Hennig et al., 2015]. Here, consistent with Bouveyron et al., it is assumed that a cluster is a group of similar objects [Bouveyron et al., 2012]. The clusters are called natural because they do not require a dissection; instead, they are clearly separated in the data [Duda et al., 2001, Theodoridis/Koutroumbas, 2009, pp. 579, 600]. These clusters can be identified by distance or density based high-dimensional structures. Dimensionality reduction techniques are able to reduce the dimensions of the input space to facilitate the exploration of structures in high-dimensional data. If they are used for visualization, they are called projection
methods. The generalized U*-matrix technique is applicable for these and can be used to visualize both distance- and density-based structures [Thrun 2017; 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 2017; 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.

Note

For interactive Island Generation of an generalized Umatrix see interactiveGeneralizedUmatrixIsland function in the package GeneralizedUmatrix.

Author(s)

Michael Thrun, Felix Pape, Florian Lerch

References


Examples

data('Hepta')
# 2d projection
# Visualization of Generalized Umatrix
projectionpoints=NeRV(Hepta$Data)
# Computation of Generalized Umatrix
library(GeneralizedUmatrix)
visualization=GeneralizedUmatrix(Data = Hepta$Data, projectionpoints)
plotTopographicMap(visualization$Umatrix, visualization$Bestmatches)

# Automatic Clustering
LC=c(visualization$Lines, visualization$Columns)
# number of cluster from dendrogram or visualization (PlotIt=T)
Cls=ProjectionBasedClustering(k=7, Hepta$Data,
visualization$Bestmatches, LC, PlotIt=T)
# Verification
plotTopographicMap(visualization$Umatrix, visualization$Bestmatches,Cls)
## Sometimes you can improve a Clustering interactively or mark additional Outliers manually
Cls2 = interactiveClustering(visualization$Umatrix, visualization$Bestmatches,Cls)

---

CCA

**Curvilinear Component Analysis**

**Description**

CCA Projects data vectors using Curvilinear Component Analysis.

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

**Usage**

CCA(DataOrDists,Epochs,OutputDimension=2,method='euclidean',
alpha0 = 0.5, lambda0,PlotIt=FALSE,Cls)
**DefaultColorSequence**

**Arguments**

- **DataOrDists**
  - array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
- **Epochs**
  - (scalar) training length
- **OutputDimension**
  - Number of dimensions in the Outputspace, default=2
- **method**
  - method specified by distance string. One of: 'euclidean', 'cityblock=manhattan', 'cosine', 'chebychev', 'jaccard', 'minkowski', 'manhattan', 'binary'
- **alpha0**
  - (scalar) initial step size, 0.5 by default
- **lambda0**
  - (scalar) initial radius of influence, $3 \times \text{max}(\text{std}(D))$ 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.

**Value**

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

**Note**


**Author(s)**

Florian Lerch

**References**


---

**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, Best-Matches 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

Dijkstra’s SSSP (Single source shortest path) algorithm:
gets the shortest path (geodesic distance) from source vertex(point) to all other vertices(points)
defined by the edges of the adjacency matrix

Usage
DijkstraSSSP(Adj, Costs, source)

Arguments
Adj [1:n,1:n] 0/1 adjacency matrix, e.g. from delaunay graph or gabriel graph
Costs [1:n,1:n] matrix, distances between n points (normally euclidean)
source int, vertex(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 vertex
itself

Note
runs in O(E*Log(V))

Author(s)
Michael Thrun

References
uses a changed code which is inspired by Shreys Sheth 28.05.2015, see http://ideone.com/qkmt31
**ICA**

**Independent Component Analysis**

**Description**

Independent Component Analysis:

Negentropy: difference of entropy to a corresponding normally-distributed random variable

\[
J(y) = |E(G(y)) - E(G(v))| \leq 2
\]

**Usage**

```
ICA(Data, OutputDimension=2, ContrastFunction="logcosh",
    Alpha=1, Iterations=200, PlotIt=FALSE, Cls)
```
**Arguments**

**Data**
array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric

**OutputDimension**
Number of dimensions in the Outputspace, default=2

**Contrastfunction**
Maximierung der Negentropie ueber geeignete geeignete Kontrastfunktion Default: 'logcosh' G(u)=1/a*log cosh(a*u) 'exp': G(u)=-exp(u^2/2)

**Alpha**
onstant with 1<=alpha<=2 used in approximation to neg-entropy when fun == "logcosh"

**Iterations**
maximum number of iterations to perform.

**PlotIt**
Default: FALSE, If TRUE: Plots the projection as a 2d visualization. OutputDimension>2: only the first two dimensions will be shown

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

**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**
Entermischungsmatrix with Data*Unmixing=ProjectedPoints

**PCMatrix**
pre-whitening matrix that projects data onto the first n.comp principal components.

**Note**
A wrapper for fastICA

**Author(s)**
Michael Thrun

---

**interactiveClustering**  
* GUI for interactive cluster analysis

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

**Arguments**

- **Umatrix**  

- **Bestmatches**  
  [1:n,1:2] Array with positions of Bestmatches

- **Cls**  
  [1:n] Classification of the Bestmatches

- **Imx**  
  [1:4*Lines,1:4*Columns] Matrix of an island that will be cut out of the umatrix, use package Umatrix for generation.

- **Toroid**  
  Are Bestmatches placed on a toroid grid? TRUE by default.

**Details**

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

**Value**

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

**Author(s)**

Florian Lerch, Michael Thrun

**References**


**Examples**

```r
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 interactively in a shiny gui
Cls = interactiveClustering(visualization$Umatrix, visualization$Bestmatches)  
# Plotting
plotTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)
```
**Description**

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

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

**Usage**

isomap(inputdistances,k,OutputDimension=2,PlotIt=FALSE,Cl=)

**Arguments**

- **inputdistances**: Matrix containing the distances of the data
- **k**: number of k nearest neighbors, if the data is fragmented choose an higher k
- **OutputDimension**: Number of dimensions in the output space, default = 2
- **PlotIt**: Default: FALSE, If TRUE: Plots the projection as a 2d visualization. If Output-Dimension > 2 only the first two dimensions will be shown.
- **Cl**: 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.

**Value**

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

**Note**

A wrapper for isomap of the package vegan

if Data fragmented choose an higher k

**Author(s)**

Michael Thrun
### Kruskal Stress

**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

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

**Author(s)**
Felix Pape

### Classical multidimensional scaling (MDS)

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

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

**Arguments**
- **DataOrDists**: array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
- **method**: method specified by distance string: 'euclidean','cityblock=manhattan','cosine','chebychev','jaccard','minkowski','manhattan','binary'
- **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.
Value

ProjectedPoints

\[1:n,\text{OutputDimension}\], \text{n by OutputDimension matrix containing coordinates of the Projectio}

Eigenvalues

the eigenvalues of \text{MDSvalues}^*\text{MDSvalues}'

Stress

Shephard-Kruskal Stress

Note

A wrapper for \text{cmdscale}

Author(s)

Michael Thrun


Description

Projection is done by the neighbor retrieval visualizer (NeRV)

Usage

\text{NeRV(Data, lambda = 0.1, neighbors = 20, iterations = 10,}

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

PlotIt = FALSE, CIs)

Arguments

Data

Matrix of the Data to be projected

lambda

Optional: Controls the trustworthiness-continuity tradeoff. Default = 0.1

neighbors

Optional: Set the number of nearest neighbours that each point should have. Should be positive. Default = 20

iterations

Optional: The number of iterations to perform. Default = 10

cg_steps

Optional: The number of conjugate gradient steps to perform per iteration in NeRV’s optimization scheme. Default = 2

cg_steps_final

Optional: The number of conjugate gradient steps to perform on the final iteration in NeRV’s optimization scheme. Default = 40

randominit

Optional: TRUE: Random Initialization (default), FALSE: PCA initialization

OutputDimension

Optional: Number of dimensions in the Outputspace, default=2
PCA

PlotIt  Optional: Should the projected points be plotted? Default: FALSE. Note: this is only useful 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.

Value

OutputDimension-dimensional matrix of projected points

Note

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

Author(s)

Michael Thrun, Felix Pape

References


---

PCA  Principal component analysis

Description

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

Usage

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

PC
PCA

Arguments

Data       array of data: n cases in rows, d variables in columns
OutputDimension  Number of dimensions in the Outputspace, default=2
Scale      a logical value indicating whether the variables should be scaled to have unit
            variance before the analysis takes place. The default is FALSE for consistency
            with S, but in general scaling is advisable. Alternatively, a vector of length equal
            the number of columns of x can be supplied. The value is passed to scale.
Center     a logical value indicating whether the variables should be shifted to be zero
            centered. Alternately, a vector of length equal the number of columns of x can
            be supplied. The value is passed to scale
PlotIt     Default: FALSE. If TRUE: Plots the projection as a 2d visualization. OutputDi-
            mension>2: only the first two dimensions will be shown
Cls        [1:n,1] Optional,: only relevant if PlotIt=TRUE. Numeric vector, given Classifi-
            cation in numbers: every element is the cluster number of a certain correspond-
            ing element of data.

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 eigen-
                  vectors)
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

Author(s)

Michael Thrun
**PlotProjectedPoints**  

**Description**
plots XY data colored by Cls with ggplot2

**Usage**

```r
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 ==F, If TRUE assuming BestMatches of ESOM instead of Projected Points
- **PlotLegend** ...
- **xlab** Optional: Label of the x axis
- **ylab** Optional: Label of the y axis
- **main** Optional: title
- **PointSize** Optional: size of points

**Value**
ggobject of ggplot2

**Author(s)**
Michael Thrun
ProjectionBasedClustering

automated Clustering approach of the Databonic swarm with abstact U distances

Description

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

Usage

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

Arguments

- **k**: number of clusters, how many to you see in the 3d landscape?
- **Data**: [1:n,1:d] Matrix of Data (n cases, d dimensions) that will be used. One Data-Point per row
- **BestMatches**: [1:n,1:2] Matrix with positions of Bestmatches=ProjectedPoints, one matrix line per data point
- **LC**: grid size c(Lines,Columns)
- **StructureType**: Optional, bool; =TRUE: compact structure of clusters assumed, =FALSE: connected structure of clusters assumed. For the two options vor Clusters, see [Thrun, 2017] or Handl et al. 2006
- **PlotIt**: Optional, bool, Plots Dendrogramm
- **method**: Optional, distance method, do not change

Details

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

Value

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

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

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

Author(s)

Michael Thrun

References


Examples

data('Hepta')
# 2d projection
projectionpoints=NeRV(Hepta$Data)
# Computation of Generalized Umatrix
visualization=GeneralizedUmatrix(Data = Hepta$Data,projectionpoints)
# Visualization of GeneralizedUmatrix
library(GeneralizedUmatrix)
plotTopographicMap(visualization$Umatrix,visualization$Bestmatches)
# Automatic Clustering
LC=c(visualization$Lines,visualization$Columns)
# number of cluster from dendrogram or visualization (PlotIt=T)
Cls=ProjectionBasedClustering(k=7, Hepta$Data,
  visualization$Bestmatches, LC,PlotIt=T)
# Verification
plotTopographicMap(visualization$Umatrix,visualization$Bestmatches,Cls)
SammonsMapping

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' \( G(u) = \frac{1}{a} \log \cosh(a \cdot u) \) (ICA) 'exp': \( G(u) = -\exp(u^2/2) \) 'kernel' \( \frac{1}{(1/\pi)^d \exp(r/2)} \)

Alpha constant with \( 1 \leq \alpha \leq 2 \) 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.

Value

ProjectedPoints

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

Author(s)

Michael Thrun

SammonsMapping

Sammons Mapping

Description

Improved MDS thorugh a normalization of the Input space

Usage

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

DataOrDists array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric
method method specified by distance string: 'euclidean', 'cityblock=manhatten', 'cosine', 'chebychev', 'jaccard', 'minkowski', 'manhattan', 'binary'
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.

Value

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

Note

A wrapper for sammon

Author(s)

Michael Thrun

Description

This function plots a Shepard diagram of InputDist and OutputDist

Usage

ShepardDiagram(InputDist, OutputDist, xlabel = "Input Distances", ylabel = "Output Distances", fancy = F, label = "ProjectionMethod", gPlot = ggplot())

Arguments

InputDist Matrix containing the distances of the inputspace.
OutputDist Matrix containing the distances of the outputspace.
xlabel Label of the x axis in the resulting Plot.
ylabel Label of the y axis in the resulting Plot.
fancy Set FALSE for PC and TRUE for publication
label Title of the Shepard diagram
gPlot Ggplot2 object to plot upon.
ShortestGraphPathsC

Value

ggplot2 object containing the plot.

Author(s)

Michael Thrun

ShortestGraphPathsC  Shortest GraphPaths = geodesic distances

Description

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

Usage

ShortestGraphPathsC(Adj, Cost)

Arguments

Adj  [1:n,1:n] 0/1 adjacency 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 O(n*E*Log(V)), see also [Jungnickel, 2013, p. 87]. Further details can be found in [Jungnickel, 2013, p. 83-87].

Value

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

References


See Also

DijkstraSSSP

---

tSNE

T-distributed Stochastic Neighbor Embedding

Description

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

Usage

tSNE(DataOrDist,k,OutputDimension=2,method="euclidean",Whitening=TRUE,InitialDimensions=NULL, Iterations=1000,PlotIt=FALSE,Cls)

Arguments

DataOrDist array of data: n cases in rows, d variables in columns, matrix is not symmetric or distance matrix, in this case matrix has to be symmetric

k number of k nearest neighbors=number of effective nearest neighbors("perplexity") Important parameter, if not given Settings of package t-SNE will be used

OutputDimension Number of dimensions in the Outputspace, default=2

method method specified by distance string: 'euclidean','cityblock=manhattan','cosine','chebychev','jaccard','minkowski','manhattan','binary'

Whitening A boolean value indicating whether the matrix data should be whitened

InitialDimensions The number of dimensions to use in reduction method.

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.
**Value**

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

**Note**

A wrapper for [tsne](https://github.com/jvandew/tsne)

**Author(s)**

Michael Thrun
Index

*Topic **Databonic swarm**
  ProjectionBasedClustering, 17

*Topic **Delaunay**
  DelaunayPoints, 6

*Topic **Dijkstra's SSSP**
  DijkstraSSSP, 7

*Topic **Dijkstra**
  DijkstraSSSP, 7

*Topic **Isomap**
  Isomap, 11

*Topic **Points**
  DelaunayPoints, 6

*Topic **SSSP**
  DijkstraSSSP, 7

*Topic **ShortestGraphPaths**
  ShortestGraphPathsC, 21

*Topic **ShortestPaths**
  ShortestGraphPathsC, 21

*Topic **Single source shortest path**
  DijkstraSSSP, 7

*Topic **cluster analysis**
  ProjectionBasedClustering, 17

*Topic **clustering**
  ProjectionBasedClustering, 17

*Topic **cluster**
  ProjectionBasedClustering, 17

*Topic **datasets,Hepta,FCPS**
  Hepta, 8

*Topic **isomap**
  Isomap, 11

*Topic **swarm**
  ProjectionBasedClustering, 17

Hepta, 8

ICA, 8

interactiveClustering, 9

Isomap, 11

KruskalStress, 12

MDS, 12

NeRV, 13

PCA, 14

PlotProjectedPoints, 16

prcomp, 15

ProjectionBasedClustering, 17

ProjectionBasedClustering-package, 2

ProjectionPursuit, 18

sammon, 20

SammonMapping, 19

ShepardDiagram, 20

ShortestGraphPathsC, 21

tSNE, 22
tsn, 23

CCA, 4

DefaultColorSequence, 5

DelaunayPoints, 6

DijkstraSSSP, 7, 22

fastICA, 9