Package ‘bigMap’

June 30, 2020

Type Package
Title Big Data Mapping
Version 2.3.1
Date 2020-06-01
Description Unsupervised clustering protocol for large scale structured data, based on a low dimensional representation of the data. Dimensionality reduction is performed using a parallelized implementation of the t-Stochastic Neighboring Embedding algorithm (Garriga J. and Bartumeus F. (2018), <arXiv:1812.09869>).
License GPL-3
Depends R (>= 3.5.0)
Imports Rcpp (>= 0.12.0), bigmemory (>= 4.5.0), parallel (>= 3.5.0), RColorBrewer, colorspace,
Suggests knitr, rmarkdown
LinkingTo Rcpp, RcppArmadillo, BH, bigmemory
LazyData FALSE
VignetteBuilder knitr
RoxygenNote 6.1.1
SystemRequirements GNU make
NeedsCompilation yes
Author Joan Garriga [aut, cre],
Frederic Bartumeus [aut]
Maintainer Joan Garriga <jgarriga@ceab.csic.es>
Repository CRAN
Date/Publication 2020-06-30 15:00:02 UTC

R topics documented:

bdm.boxp ......................................................... 2
bdm.cost .......................................................... 3
bdm.dMap .......................................................... 4
bdm.boxp(bdm, byVars = F, layer = 1)

Arguments

bdm A bdm instance as generated by bdm.init().
byVars A logical value. By default (byVars = FALSE) box-plots are grouped by cluster.
layer The number of a layer (1 by default).

Details

If the number of clusters is large, only the first 25 clusters will be plotted. Note that the WTT
algorithm numbers the clusters based on density value at the peak cell of the cluster. Thus, the num-
bering of the clusters is highly correlated with their relevance in terms of partial density. Therefore,
in case of more than 25 clusters, the most relevant should always be included in the plot.
bdm.cost

Value

None.

Examples

bdm.example()
bdm.boxp(exMap)
bdm.boxp(exMap, byVars = TRUE)

bdm.cost  ptSNE cost & size plot.

Description

ptSNE cost & size plot.

Usage

bdm.cost(bdm, offset = 0)

Arguments

bdm  A bdm instance as generated by bdm.init() or a list of them to make a comparative plot.

offset  X-axis offset in number of epochs (0 by default).

Value

None.

Examples

bdm.example()
bdm.cost(exMap)
bdm.dMap

Class density maps

Description

Compute the class density maps of a set of classes on the embedding grid. This function returns a fuzzy mapping of the set of classes on the grid cells. The classes can be whatever set of classes of interest and must be given as a vector of point-wise discrete labels (either numeric, string or factor).

Usage

bdm.dMap(bdm, threads = 2, type = "SOCK", data = NULL, layer = 1)

Arguments

bdm A bdm instance as generated by bdm.init().
threads The number of parallel threads (in principle only limited by hardware resources, i.e. number of cores and available memory)
type The type of cluster: 'SOCK' (default) for intra-node parallelization, 'MPI' for inter-node parallelization (message passing interface parallel environment).
data A vector of discret covariates or class labels. The covariate values can be of any factorizable type. By default (data=NULL) the function computes the density maps based on the clustering labels (i.e. equivalent to data=bdm.labels(bdm))
layer The number of the t-SNE layer (1 by default).

Details

bdm.dMap() computes the join distribution $P(V = v_i, C = c_j)$ where $V = v_1, \ldots, v_l$ is the discrete covariate and $C = c_1, \ldots, c_g$ are the grid cells of the paKDE raster. That is, this function recomputes the paKDE but keeping track of the covariate (or class) label of each data-point. This results in a fuzzy distribution of the covariate (class) at each cell.

Usually, figuring out the join distribution $P(V = v_i, C = c_j)$ entails an intensive computation. Thus bdm.dMap() performs the computation and stores the result in a dedicated element named $dMap$. Afterwards the class density maps can be visualized with the bdm.dMap.plot() function.

Value

A copy of the input bdm instance with element $dMap$, a matrix with a soft clustering of the grid cells.

Examples

# --- load example dataset
bdm.example()
## Not run:
```r
exMap <- bdm.dMap(exMap, threads = 4)
## End(Not run)
```

---

**bdm.dMap.plot**  
*Class density maps plot.*

**Description**  
Class density maps plot.

**Usage**  
```r
bdm.dMap.plot(bdm, classes = NULL, join = FALSE, class.pltt = NULL,  
pakde.pltt = NULL, pakde.lvls = 16, wtt.lwd = 1, plot.peaks = T,  
labels.cex = 1, layer = 1)
```

**Arguments**  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>bdm</code></td>
<td>A <em>bdm</em> instance as generated by <em>bdm.init()</em></td>
</tr>
<tr>
<td><code>classes</code></td>
<td>A vector with a subset of class names or covariate values. Default value is</td>
</tr>
<tr>
<td></td>
<td>classes=NULL. If no classes are specified (default value) all classes are</td>
</tr>
<tr>
<td></td>
<td>plotted.</td>
</tr>
<tr>
<td><code>join</code></td>
<td>Logical value. If FALSE (default value), class mapping is based on the</td>
</tr>
<tr>
<td></td>
<td>class conditional distributions. If TRUE, class mapping is based on the</td>
</tr>
<tr>
<td></td>
<td>overall classes join distribution.</td>
</tr>
<tr>
<td><code>class.pltt</code></td>
<td>A colour palette to show class labels in the hard mapping. By default</td>
</tr>
<tr>
<td></td>
<td>(class.pltt=NULL) the default palette is used.</td>
</tr>
<tr>
<td><code>pakde.pltt</code></td>
<td>A palette of colours to indicate the levels of the class density maps. The</td>
</tr>
<tr>
<td></td>
<td>length of the colour palette should be at least the number of levels</td>
</tr>
<tr>
<td></td>
<td>specified in <em>pakde.lvls</em>.</td>
</tr>
<tr>
<td><code>pakde.lvls</code></td>
<td>The number of levels of the heat-map when plotting class density maps (16</td>
</tr>
<tr>
<td></td>
<td>by default).</td>
</tr>
<tr>
<td><code>wtt.lwd</code></td>
<td>The width of the watertrack lines (as set in <em>par()</em>)</td>
</tr>
<tr>
<td><code>plot.peaks</code></td>
<td>Logical value (TRUE by default). If set to TRUE and the up-stream step</td>
</tr>
<tr>
<td></td>
<td><em>bdm$wtt()</em> is computed the peak of each cluster is depicted.</td>
</tr>
<tr>
<td><code>labels.cex</code></td>
<td>If <em>plot.peaks</em> is TRUE, the size of the labels of the clusters (as set in</td>
</tr>
<tr>
<td></td>
<td><em>par()</em>) By default labels.cex=0.0 and the labels of the clusters are not</td>
</tr>
<tr>
<td></td>
<td>depicted.</td>
</tr>
<tr>
<td><code>layer</code></td>
<td>The number of the layer from which the class density maps are computed (1</td>
</tr>
<tr>
<td></td>
<td>by default).</td>
</tr>
</tbody>
</table>
Details

`bdm.dMap.plot()` yields a multi-plot layout where the first plot shows the dominating value of the covariate (or dominating class) in each cell, and the rest of the plots show the density map of each covariate value (or class).

The join distribution $P(V = v_i, C = c_j)$ will be affected by the bias present in the marginal distribution of the covariate. Therefore, the join distribution $P(V = v_i, C = c_j)$ is transformed, by default, into a conditional distribution $P(c_j | V = v_i)$ (where the $c_j$ are the grid cells of the embedding and V is the covariate (or class)). Thus, the first plot shows a hard classification of grid-cells, (cells are coloured based on the dominating value of the covariate (or dominating class), i.e. the $v_i$ for which $P(c_j | V = v_i)$ is maximum), and the rest of the plots show the conditional distributions $P(C = c_j | V = v_i)$. This makes the plots of the different classes not directly comparable but the dominant areas of each class can be more easily identified.

However, the same plots can be depicted based on the join distribution by setting `join = TRUE`. This makes sense when the bias in the covariate values (or classes) is not significant. In this case the hard clustering shows the real dominance of each covariate value (or class) over the embedding area and the density maps are comparable one to each other (although, individually, they are not real density functions as they do not add up to one).

The multi-plot layout can be limited to a subset of the values of the covariate (or subset of classes) specified in parameter `classes`.

Value

None.

Examples

```r
# --- load example dataset
dbm.example()

## Not run:
exMap <- bdm.dMap(exMap, threads = 4)
bdm.dMap.plot(exMap)

## End(Not run)
```

Description

Loads an example of a mapping of a dataset.

Usage

`bdm.example()`
**Details**

A *bdm* instance is a list with elements: $\text{dSet}$ a name identifying the dataset (bdm.fName() use this name to generate a default file name); $\text{data}$ a matrix with raw data; $\text{lbls}$ a vector of datapoint labels (in case they are known); $\text{N}$ the dataset size; $\text{is.distance}$ a logical value that is set to TRUE when the raw data is a distance matrix. Downstream steps of the mapping protocol will add more elements to the list.

This example is based on a small synthetic dataset with $n = 5000$ observations drawn from a 4-variate Gaussian Mixture Model (GMM) with 16 Gaussian components.

**Value**

An example *bdm* instance named *exMap*.

**Examples**

```r
# --- load example dataset
bdm.example()
str(exMap)
```

<table>
<thead>
<tr>
<th>bdm.fName</th>
<th>Default bdm file name</th>
</tr>
</thead>
</table>

**Description**

Generates a default file name. The default file name is intended for functions *bdm.save()* and *bdm.scp()* to ease the task of working/organizing multiple runs on the same dataset.

**Usage**

```r
bdm.fName(bdm)
```

**Arguments**

*bdm* A *bdm* instance as generated by *bdm.init()*.

**Details**

The file name is generated based on *bdm$dSet* and main ptSNE parameters (threads, layers, rounds, boost and perplexity). In case that *bdm.wtt()* has been performed on any of the layers, the number of clusters in the first not null layer of *bdm$wtt* is also included.

**Value**

A *.RData* file name based on *bdm$\text{dSet}* and main *bdm* parameters.
bdm.init

Create a \textit{bdm} instance.

Usage

\begin{verbatim}
bdm.init(dSet.name, dSet.data, labels = NULL, is.distance = F,
        check.duplicates = T)
\end{verbatim}

Arguments

- \textbf{dSet.name} The name given to the input dataset. This name will be used to automatically generate a name to save the output as an \textit{.Rdata} file.
- \textbf{dSet.data} A \textit{data.frame} or \textit{matrix} with raw input-data. The dataset must not have duplicated rows.
- \textbf{labels} If available, labels can be included as a separate vector of length equal to \texttt{nrow(dSet.data)}. Label values are factorized as \texttt{as.numeric(as.factor(labels))}.
- \textbf{is.distance} A logical value (\texttt{FALSE} by default). \texttt{TRUE} indicates that the raw data is indeed a distance matrix.
- \textbf{check.duplicates} If set to \texttt{TRUE} (default value) the dataset is checked for duplicated rows. Checking for duplicates in big datasets can take some time. If the dataset is known to have no duplicates disabling this option will save time.

Value

A \textit{bdm} instance. A \textit{bdm} instance is initially a list with a few elements to which new elements are added at each step of the mapping protocol.

Examples

\begin{verbatim}
# --- get a matrix with raw-data
mydata <- matrix(rnorm(10000, mean = 0, sd = 3), ncol = 2)
mylabels <- apply(mydata, 1, function(row) round(sqrt(sum(row**2)), 0))
# --- create a \texttt{\textbackslash var{bdm}} instance with our raw-data matrix
mybdm <- bdm.init('mydataset', mydata, labels = mylabels)
str(mybdm)
\end{verbatim}
bdm.labels

Get data-point clustering labels.

Description
Given that clusters are computed at grid-cell level, this function returns the clustering label for each data-point.

Usage
bdm.labels(bdm, merged = T, layer = 1)

Arguments
- **bdm**: A `bdm` instance as generated by `bdm.init()`.
- **merged**: A logical value. If TRUE (default value) and the `bdm` has been merged, the data-point labelling indicate the number of the merged clusters. If `merged` is set to FALSE or the `bdm` has not been merged the data-point labels correspond to the top-level clustering.
- **layer**: The number of the t-SNE layer (1 by default).

Value
A vector of data-point clustering labels.

Examples
```r
bdm.example()
exMap.labels <- bdm.labels(exMap)
```

bdm.local

Set/get default local machine name or IP address

Description
Set/get default local machine name or IP address

Usage
bdm.local(dest = NULL)

Arguments
- **dest**: Name or IP address of the local machine.
bdm.merge.s2nr

Value

The current value of local

Examples

```r
# --- set default value of \var{local}
bdm.local('xxx.255.0.0')
bdm.local('mymachine.mydomain.cat')
```

bdm.merge.s2nr  Merging of clusters based on signal-to-noise-ratio.

Description

Performs a recursive merging of clusters based on minimum loss of signal-to-noise-ratio (S2NR) until reaching the desired number of clusters. The S2NR is the explained/unexplained variance ratio measured in the high dimensional space based on the given low dimensional clustering.

Usage

```r
bdm.merge.s2nr(bdm, k = 10, plot.merge = T, ret.merge = F, info = T, layer = 1, ...)
```

Arguments

- **bdm**: A `bdm` instance as generated by `bdm.init()`.
- **k**: The number of desired clusters. The clustering will be recursively merged until reaching this number of clusters (default value is `k = 10`). By setting `k < 0` we can specify the number of clusters that we are willing to merge.
- **plot.merge**: Logical value. If TRUE, the merged clustering is plotted (default value is `plot.merge = TRUE`).
- **ret.merge**: Logical value. If TRUE, the function returns a copy of the input `bdm` instance with the merged clustering attached as `bdm$merge` (default value is `ret.merge = FALSE`).
- **info**: Logical value. If TRUE, all merging steps are shown (default value is `info = FALSE`).
- **layer**: The `bdm$ptsne` layer to be used (default value is `layer = 1`).
- **...**: If `plot.merge` is TRUE, you can set the `bdm.wtt.plot()` parameters to control the plot.

Details

See details in `bdm.optk.s2nr()`.
Value

None if ret.merge = FALSE. Else, a copy of the input bdm instance with new element bdm$merge.

Examples

```r
bdm.example()
exMap.labels <- bdm.labels(exMap)
```

---

### bdm.mybdm

**Set/get default path for mybdm**

**Description**

Set/get default path for mybdm

**Usage**

```r
bdm.mybdm(path = NULL)
```

**Arguments**

- `path`: Path to mybdm.

**Value**

The current path value to mybdm

**Examples**

```r
# --- set default path for \var{mybdm}
bdm.mybdm("~/mybdm")
```

---

### bdm.optk.plot

**Plots the signal-to-nois-ratio as a function of the number of clusters.**

**Description**

The function `bdm.optk.sn2r()` computes the S2NR that results from recursively merging clusters and, by default, makes a plot of these values. For large datasets this computation can take a while, so we can save this result by setting `ret.optk = TRUE`. If this result is saved, we can plot it again at any time using this function.

**Usage**

```r
bdm.optk.plot(bdm)
```
Arguments

bdm
A \textit{bdm} instance as generated by \texttt{bdm.init()}.

Value

None.

Examples

\begin{verbatim}
bdm.example()
exMap <- bdm.optk.s2nr(exMap, ret.optk = TRUE)
bdm.optk.plot(exMap)
\end{verbatim}

\begin{verbatim}
bdm.optk.s2nr
\end{verbatim} \textit{Find optimal number of clusters based on signal-to-noise-ratio.}

Description

Performs a recursive merging of clusters based on minimum loss of signal-to-noise-ratio (S2NR). The S2NR is the explained/unexplained variance ratio measured in the high dimensional space based on the given low dimensional clustering. Merging is applied recursively until reaching a configuration of only 2 clusters and the S2NR is measured at each step.

Usage

\begin{verbatim}
bdm.optk.s2nr(bdm, info = T, plot.optk = T, ret.optk = F, layer = 1)
\end{verbatim}

Arguments

\begin{verbatim}
bdm
info
plot.optk
ret.optk
layer
\end{verbatim}

A clustered \textit{bdm} instance (\textit{i.e.} all up-stream steps performed: \texttt{bdm.ptse()}, \texttt{bdm.pakde()} and \texttt{bdm.wtt()}).

Logical value. If TRUE, all merging steps are shown (default value is \texttt{info = FALSE}).

Logical value. If TRUE, this function plots the heuristic measure versus the number of clusters (default value is \texttt{plot.optk = TRUE})

Logical value. For large datasets this computation can take a while and it might be interesting to save it. If TRUE, the function returns a copy of the \textit{bdm} instance with the values of S2NR attached as \texttt{bdm$optk} (default value is \texttt{ret.optk = FALSE})

The \texttt{bdm$ptsne} layer to be used (default value is \texttt{layer = 1}).
Details

The logic under this heuristic is that neighbouring clusters in the embedding correspond to close clusters in the high dimensional space, i.e. it is a merging heuristic based on the spatial distribution of clusters. For each cluster (child cluster) we choose the neighboring cluster with steepest gradient along their common border (father cluster). Thus, we get a set of pairs of clusters (child/father) as potential mergings. Given this set of candidates, the merging is performed recursively choosing, at each step, the pair of child/father clusters that results in a minimum loss of S2NR. A typical situation is that some clusters dominate over all of their neighboring clusters. This clusters have no father. Thus, once all candidate mergings have been performed we reach a blocked state where only the dominant clusters remain. This situation identifies a hierarchy level in the clustering. When this situation is reached, the algorithm starts a new merging round, identifying the child/father relations at that level of hierarchy. The process stops when only two clusters remain. Usually, the clustering hierarchy is clearly depicted by singular points in the S2NR function. This is a hint that the low dimensional clustering configuration is an image of a hierarchical spatial configuration in the high dimensional space. See bdm.optk.plot().

Value

None if ret.optk = FALSE. Else, a copy of the input bdm instance with new element bdm$optk (a matrix).

Examples

```r
# --- load mapped dataset
bdm.example()
# --- compute optimal number of clusters and attach the computation
bdm.optk.s2nr(exMap, plot.optk = TRUE, ret.optk = FALSE)
```

bdm.pakde

Perplexity-adaptive kernel density estimation

Description

Starts the paKDE algorithm (second step of the mapping protocol).

Usage

```r
bdm.pakde(bdm, layer = 1, threads = 2, type = "SOCK", ppx = 100,
itr = 100, tol = 1e-05, g = 200, g.exp = 3)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bdm</td>
<td>A bdm instance as generated by bdm.init().</td>
</tr>
<tr>
<td>layer</td>
<td>The number of the t-SNE layer (1 by default).</td>
</tr>
<tr>
<td>threads</td>
<td>The number of parallel threads (in principle only limited by hardware resources, i.e. number of cores and available memory)</td>
</tr>
</tbody>
</table>
type

The type of cluster: 'SOCK' (default) for intra-node parallelization, 'MPI' (message passing interface) for inter-node parallelization.

ppx

The value of perplexity to compute similarities in the low-dimensional embedding (100 by default).

itr

The number of iterations for computing input similarities (100 by default).

tol

The tolerance lower bound for computing input similarities (1e-05 by default).

g

The resolution of the density space grid \((g \times g)\) cells, 200 by default.

g.exp

A numeric factor to avoid border effects. The grid limits will be expanded so as to enclose the density of the kernel of the most extreme embedded datapoints up to \(g, \text{exp} \times \sigma\). By default, \((g, \text{exp} = 3)\) the grid limits are expanded so as to enclose the 0.9986 of the probability mass of the most extreme kernels.

Details

When computing the \(\text{paKDE}\) the embedding area is discretized as a grid of size \(g \times g\) cells. In order to avoid border effects, the limits of the grid are expanded by default so as to enclose at least the 0.9986 of the cumulative distribution function \((3\sigma)\) of the kernels of the most extreme mapped points in each direction.

The presence of outliers in the embedding can lead to undesired expansion of the grid limits. We can overcome this using lower values of \(g, \text{exp}\). By setting \(g, \text{exp} = 0\) the grid limits will be equal to the range of the embedding.

The values \(g, \text{exp} = c(1, 2, 3, 4, 5, 6)\) enclose cdf values of 0.8413, 0.9772, 0.9986, 0.99996, 0.99999, 1.0 respectively.

Value

A copy of the input \(bdm\) instance with new element \(bdm,\text{pakde}\) (paKDE output). \(bdm,\text{pakde}[[\text{layer}]]\)\$layer = 'NC' stands for not computed layers.

Examples

```r
# --- load mapped dataset
bdm.example()
# --- run paKDE
## Not run:
exMap <- bdm.pakde(exMap, threads = 4, ppx = 200, g = 200, g.exp = 3)
## End(Not run)
# --- plot paKDE output
bdm.pakde.plot(exMap)
```
bdm.pakde.plot

Plot paKDE (density landscape)

Description

Plot paKDE (density landscape)

Usage

bdm.pakde.plot(bdm, pakde.pltt = NULL, pakde.lvls = 16, layer = 1)

Arguments

bdm A bdm instance as generated by bdm.init() or a list of them to make a comparative plot.
pakde.pltt A colour palette to show levels in the paKDE plot. By default (pakde.pltt = NULL) the default palette is used.
pakde.lvls The number of levels of the density heat-map (16 by default).
layer The bdm$ptsne layer to be used (default value is layer = 1).

Value

None.

Examples

bdm.example()
exMap <- bdm.pakde.plot(exMap)

bdm.ptsne

Parallelized t-SNE

Description

Starts the ptSNE algorithm (first step of the mapping protocol).

Usage

bdm.ptsne(bdm, threads = 3, type = "SOCK", layers = 2, rounds = 1, boost = 2, whiten = 4, input.dim = NULL, ppx = 100, itr = 100, tol = 1e-05, alpha = 0.5, Y.init = NULL, info = 1)
Arguments

bdm
A bdm instance as generated by bdm.init().

threads
The number of parallel threads (in principle only limited by hardware resources, i.e. number of cores and available memory)

type
The type of cluster: 'SOCK' (default) for intra-node parallelization, 'MPI' (message passing interface) for inter-node parallelization.

layers
The number of layers (minimum 2, maximum the number of threads).

rounds
The number of rounds (2 by default).

boost
A running time accelerator factor. By default (boost == 1). See details.

whiten
Preprocessing of raw data. If whiten = 4 (default value) raw data is transformed to principal components (PCA) and whitened afterwards. If whiten = 3 only PCA is performed with NO whitening. If whiten = 2 raw data is only centered and scaled. If whiten = 1 raw data is only centered. If whiten = 0 no preprocessing is performed at all.

input.dim
If raw data is given as (or is transformed to) principal components, input.dim sets the number of principal components to be used as input dimensions. Otherwise all data columns are used as input dimensions. By default input.dim = ncol(bdm$data).

ppx
The value of perplexity to compute similarities (100 by default).

itr
The number of iterations for computing input similarities (100 by default).

tol
The tolerance lower bound for computing input similarities (1e-05 by default).

alpha
The momentum factor (0.5 by default).

Y.init
A nx2 matrix with initial mapping positions. By default (NULL) will use random initial positions)

info
Progress output information: 1 yields inter-round results for progressive analytics, 0 disables intermediate results. Default value is 1.

Details

By default the algorithm is structured in $\sqrt{n}$ epochs of $\sqrt{z}$ iterations each, where $n$ is the dataset size and $z$ is the thread-size ($z = n * layers/threads$). The running time of the algorithm is then determined by $epochs * iters * t_i + epochs * t_e$ where $t_i$ is the running time of a single iteration and $t_e$ is the inter-epoch running time.

The boost factor is meant to reduce the running time. With boost > 1 the algorithm is structured in $n/boost$ epochs with $z * boost$ iterations each. This structure performs the same total number of iterations but arranged into a lower number of epochs, thus decreasing the total running time to $epochs * iters * t_i + 1/boost * epochs * t_e$. When the number of threads is high, the inter-epoch time can be high, in particular when using 'MPI' parallelization, thus, reducing the number of epochs can result in a significant reduction of the total running time. The counterpart is that increasing the number of iterations per epoch might result in a lack of convergence, thus the boost factor must be used with caution. To the most of our knowledge using values up to boost = 2.5 is generally safe.

In case of extremely large datasets, we strongly recommend to initialize the bdm instance with already preprocessed data and use whiten = 0. Fast principal components approximations can be computed by means of e.g. flashpcaR or scater R packages.
A copy of the input `bdm` instance with new element `bdm$ptsne` (t-SNE output).

Examples

```r
# --- load example dataset
dbm.example()
# --- perform ptSNE
## Not run:
exMap <- bdm.ptsne(exMap, threads = 10, layers = 2, rounds = 2, ppx = 200)

## End(Not run)
# --- plot the Cost function
dbm.cost(exMap)
# --- plot ptSNE output
dbm.ptsne.plot(exMap)
```

Value

None.

Arguments

- `bdm`: A `bdm` instance as generated by `bdm.init()` or a list of them to make a comparative plot.
- `ptsne.cex`: The size of the mapped data-points in the ptSNE plot. Default value is `ptsne.cex = 0.5`.
- `ptsne.bg`: The background colour of the ptSNE plot. Default value is `ptsne.bg = "#FFFFFF"` (white).
- `class.pltt`: A colour palette to show class labels in the ptSNE plot. If `!is.null(bdm$wtt)` cluster labels are used by default, else if `!is.null(bdm$lbls)` are used by default. If `ptsne.pltt = NULL` (default value) the default palette is used.
- `layer`: The `bdm$ptsne` layer to be used (default value is `layer = 1`).

Description

Plot ptSNE (low-dimensional embedding)

Usage

```r
dbm.ptsne.plot(bdm, ptsne.cex = 0.5, ptsne.bg = "#FFFFFF", class.pltt = NULL, layer = 1)
```
Examples

```r
bdm.example()
exMap <- bdm ptsne.plot(exMap)
```

**bdm.qMap**  
*ptSNE quantile-maps*

**Description**

Shows the mapping of quantitative variables into the embedding space.

**Usage**

```r
bdm.qMap(bdm, data = NULL, labels = NULL, subset = NULL, qMap.levels = 8, qMap.cex = 0.3, qMap.bg = "#FFFFFF", class.pltt = NULL, layer = 1)
```

**Arguments**

- `bdm`: A `bdm` instance as generated by `bdm.init()`.
- `data`: A matrix/data.frame to be mapped. By default, the input data `bdm$data` is mapped.
- `labels`: A vector of class labels of length equal to `nrow(bdm$data)`. Label values are factorized as `as.numeric(as.factor(labels))`. If `!is.null(bdm$lbls)`, these labels are used by default.
- `subset`: A numeric vector with the indexes of a subset of data. Data-points in the subset are heat-mapped and the rest are shown in light grey. By default all data-points are heat-mapped.
- `qMap.levels`: The number of levels of the quantile-map (8 by default).
- `qMap.cex`: The size of the data-points (as in `par()`).
- `qMap.bg`: The background colour of the qMap plot. Default value is `ptsne.bg = #FFFFFF` (white).
- `class.pltt`: If `!is.null(labels)` or `!is.null(bdm$lbls)`, a colour palette to show class labels with the qMap plots. By default (`qMap.pltt = NULL`) the default palette is used.
- `layer`: The number of a layer (1 by default).

**Details**

This is not a heat-map but a quantile-map plot. This function splits the range of each variable into as many quantiles as specified by `levels` so that the color gradient will hardly ever correspond to a constant numeric gradient. Thus, the mapping will show more evenly distributed colors though at the expense of possibly exaggerating artifacts. For variables with very extrem distributions, it will be impossible to find as many quantiles as desired and the distribution of colors will not be so homogeneous.
bdm.save

Value

None.

Examples

bdm.example()
bdm.qMap(exMap)

# --- show only components (1, 2, 4, 8) of the GMM
bdm.qMap(exMap, subset = which(exMap$lbls %in% c(1, 4, 8, 16)))

bdm.save

Save bdm instance

Description

Saves a bdm instance with default path/file names, as given by bdm.mybdm() / bdm.fName(bdm). Default file name is generated based on bdm$dSet and ptSNE main parameters (threads, layers, boost, rounds, perplexity). The purpose of functions bdm.save() and bdm.scp() used with bdm.fName() is to ease the task of working/organizing multiple runs on the same dataset.

Usage

bdm.save(...)

Arguments

... A bdm instance as generated by bdm.init().

Value

None

Examples

# --- get a matrix with raw-data
mydata <- cbind(rnorm(10000, mean = 0, sd = 3), ncol = 2)
mylabels <- apply(mydata, 1, function(row) round(sqrt(sum(row**2)), 0))
# --- create a \var{bdm} instance with our raw-data matrix
mybdm <- bdm.init('mydataset', mydata, labels = mylabels)
str(mybdm)

# --- save it
## Not run:
bdm.save(mybdm)

## End(Not run)
**bdm.scp**

*Transfer bdm instance to a remote machine.*

**Description**

Transfers a `bdm` instance to a remote machine. By default a file name is generated based on `bdm$dSet` and t-SNE main parameters (threads, layers, rounds, perplexity). The purpose of functions `bdm.save()` and `bdm.scp()` used with `bdm.fName()` is to ease the task of working/organizing multiple runs on the same dataset.

**Usage**

```
bdm.scp(..., dest = NULL)
```

**Arguments**

- `...` A `bdm` instance as generated by `bdm.init()`.
- `dest` The name or IP address of a remote machine where to transfer the file of the `bdm` instance. By default is send to `bdm.local()` environment variable.

**Value**

None

**Examples**

```
## Not run:
# --- load example
bdm.example()
# --- scp to \var{bdm.local()} with default file name
bdm.scp(exMap)
# --- scp to IP address 'xxx.xxx.0.0' with default file name
bdm.scp(exMap, dest = 'xxx.xxx.0.0')
```

**bdm.wtt**

*Watertrack transform (WTT)*

**Description**

Starts the WTT algorithm (third setp of the mapping protocol).

**Usage**

```
bdm.wtt(bdm, layer = 1)
```
**Arguments**

- `bdm`: A `bdm` instance as generated by `bdm.init()`.
- `layer`: The number of the t-SNE layer (1 by default).

**Details**

This function requires the up-stream step `bdm.pakde()`.

**Value**

A copy of the input `bdm` instance with `bdm$wtt` (WTT output). `bdm$wtt[[layer]]$layer = 'NC'` stands for not computed layers.

**Examples**

```r
# --- load mapped dataset
bdm.example()
# --- perform WTT
exMap <- bdm.wtt(exMap)
# --- plot WTT output
bdm.wtt.plot(exMap)
```

---

**Description**

Plot WTT (clustering)

**Usage**

```r
bdm.wtt.plot(bdm, pakde.pltt = NULL, pakde.lvls = 16, wtt.lwd = 1,
            plot.peaks = T, labels.cex = 1, layer = 1)
```

**Arguments**

- `bdm`: A `bdm` instance as generated by `bdm.init()` or a list of them to make a comparative plot.
- `pakde.pltt`: A colour palette to show levels in the paKDE plot. By default (pakde.pltt = NULL) the default palette is used.
- `pakde.lvls`: The number of levels of the density heat-map (16 by default).
- `wtt.lwd`: The width of the watertrack lines (as set in `par()`).
- `plot.peaks`: Logical value (TRUE by default). If set to TRUE and the up-stream step `bdm$wtt()` is computed marks the peak of each cluster.
- `labels.cex`: If `plot.peaks` is TRUE, the size of the labels of the clusters (as set in `par()`). By default `labels.cex = 0.0` and the labels of the clusters are not depicted.
- `layer`: The `bdm$ptsne` layer to be used (default value is `layer = 1`).
Value

None.

Examples

```
bdm.example()
exMap <- bdm.wtt.plot(exMap)
```
Index

bdm.boxp, 2
bdm.cost, 3
bdm.dMap, 4
bdm.dMap.plot, 5
bdm.example, 6
bdm.fName, 7
bdm.init, 8
bdm.labels, 9
bdm.local, 9
bdm.merge.s2nr, 10
bdm.mybdm, 11
bdm.optk.plot, 11
bdm.optk.s2nr, 12
bdm.pakde, 13
bdm.pakde.plot, 15
bdm.ptsne, 15
bdm.ptsne.plot, 17
bdm.qMap, 18
bdm.save, 19
bdm.scp, 20
bdm.wtt, 20
bdm.wtt.plot, 21