Package 'WassersteinPCA'

November 6, 2018

Type Package

Title Compute Principal Components in the 2-Wasserstein-Space

Version 1.0

Author Florian Heinemann

Maintainer Florian Heinemann <florian.heinemann@stud.uni-goettingen.de>

Description This package computes Principal Geodesics and Barycenters in the 2-Wasserstein-Space to perform PCA on images.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.4)

Imports transport, Barycenter, Matrix, imager, grDevices, graphics, stats

Suggests animation, ks

RoxygenNote 6.1.0

R topics documented:

WassersteinPCA-package

Package for the computation and visual analysis of Principal Geodesics in the 2-Wasserstein-Space.

Description

Visual Principal Component Analysis in the 2-Wasserstein-Space.

Details

This package is designed to easily enable principal component analysis on image datasets. To this end the images are supposed to be point measures in the 2-Wasserstein-Space and then principal geodesics in this space are computed. For computations there are four methods available at the moment. The default methods relies on linearizing the problem and then essentially performing standard PCA. This has the advantage, that it is fast to compute, hence for everyday application this is the preferred method. Additionally, there are multiple iterative methods available. Given sufficient iterations, this methods will outperform the default method, but this is extremely costly and can easily have run times surpassing the 24-hours mark. It is therefore only recommended in the cases where optimal results are pivotal.

In addition to the computational tools, this package is equipped with two visualization tools. It has the ability to visualize the computed principal geodesic in a way, that allows for easy visual analysis. The geodesics are evaluated on a specified time grid and then kernel density estimators are used to obtain values on a pixel grid. The resulting series of images can either be turned into a series of png-images, that are generated in the current working directory, or ImageMagick is called to transform this images into a gif. The second option requires the installation of the animation-package as well as the installation of the tool ImageMagick. For details on the installation of ImageMagick refer to the manual of the animation package. (Hint: On some operation system it is necessary to check the box "install legacy utilities" during the installation). The function that generates this visuals is visual_geodesic.
The easiest way to use the package is by using the image_pca function. This is a wrapper function for all the methods availible in the package. It also allows to create the visual outputs, namely images and animations.

The only case where it is really necessary to call the other functions directly is the case, that one does not consider image data, but general two-dimensional weighted point measures. In this case refer to the documentations of the individual functions pca_log, pca_grid, pca_ws_tp and pca_ws.

This package also offers the option to compute the Wasserstein-Barycenters of general weighted point measures with a support that is not necessarily a subset of a regular grid. This is necessary to perform PCA, but this feature can also be used individually by calling the wsbary_cont function.

References

Seguy, Vivien and Cuturi, Marco;Principal Geodesic Analysis for Probability Measures under the Optimal Transport Metric; Advances in Neural Information Processing Systems 28;2015.

Cazelles, Elsa & Seguy, Vivien & Bigot, Jérémie & Cuturi, Marco & Papadakis, Nicolas; Log-PCA versus Geodesic PCA of histograms in the Wasserstein space; SIAM Journal on Scientific Computing. 40; 2017.
A Linear Optimal Transportation Framework for Quantifying and Visualizing Variations in Sets of Images; International Journal of Computer Vision; Wang, Wei & Slepcev, Dejan & Basu, Saurav & Ozolek, John A. & Rohde, Gustavo K.;2013.
Marco Cuturi and Arnaud Doucet; Fast computation of wasserstein barycenters; In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14); 2014.

`dir_image_pca` *Perform Principal Component Analysis on a directory of images.*

Description

Perform Principal Component Analysis on a directory of images.

Usage

```
dir_image_pca(dir, imagetype = "png", imagesize, imagecount = NULL,
  barycenter = NULL, pca.count = 3, method = "log", K = 20,
  steps = 10^-4, lambda = 1, maxiter = 600, error = 10^-7,
  submaxiter = 1000, suberror = 10^-12, substeps = 10^-7,
  sample.points = 100, visual = TRUE, gif = FALSE, H = (1/1000) *
  matrix(c(1, 0, 0, 1), 2, 2), grid.size = 100, file = "RPCA")
```

Arguments

`dir`	A string specifying a directory location that contains images.
`imagetype`	A string specifying the type of images in the directory. The file ending has to be given here.
`imagesize`	A positive integer specifying the size at which the images should be imported. Note that sizes over 100 might requiere significant amounts of run time and RAM. Even for moderate imagesizes, the computation time might exceed a single day.
`imagecount`	A positive integer specifying the amount of images from the directory which are to be imported. If this value is not specified, then all files are imported. Otherwise imagecount images are sampled from the directory uniformly.
`barycenter`	An N x M Matrix representing the barycenter of the images. If non is specified, then the one is approximated iteratively.
`pca.count`	An integer specifying the number of principal components to be computed. Has to be lower than the amount of data images used for the pca.
`method`	A string specifying the method used to compute the pca. The availible options are:

"log" The default method. The image measures are mapped to the tangent space at the barycenter and standard euclidean is performed on the linearized data. Afterwards the computed principal components are mapped back to the Wasserstein Space.

"sh": This method computes the principal components via a penalized, projected gradient descent method. It uses entropy regulazied Sinkhorn distances implemented in the Barycenter package to approximate the Wasserstein Distances.

"tp" This method computes the principal components via a penalized, projected gradient descent method. The Wasserstein distance is computed exactly using the implementation in the transport package.

"grid" This method computes the principal components via a projected gradient descent method. It assumes the data to be supported on the full pixel grid. If the data has indeed mass on nearly all pixels, than this method is significantly faster, than the "sh" and "tp" method. If the data is rather sparse it is strongly advised not to use this method.

WARNING: The iterative methods are volatile to the choice of step sizes and maximum iterations. If one obtains unreasonable results, it is advised to rerun the computation with a smaller step size and higher number of iterations. Additionally the run time of the iterative methods might exceed a day even for moderate image sizes. However the performance can be improved greatly by running it on clusters for parallelization. The "log" method computes results quickly, but the iterative methods yield better results given sufficient iterations. Hence it is recommended to use "log" for everyday operations and deploy the other methods when it is important to obtain the best possible results with limited regard to run time.

`K`	Integer used for "sh" and "tp". It specifies the size of the time grid the geodesic is computed on in every iteration. Choosing this too small negatively impacts convergence, but one should be cautios, since computation time scales linearly in K.
`steps`	The step size for the gradient descent used in the iterative methods.
`lambda`	The penalty parameter for the "tp" and "sh" methods. If set to zero generalized geodesics are computed instead of true ones. Otherwise the penalization forces the two velocity fields in the generalized geodesic to be collinear.
`maxiter`	The maximum number of iterations for the iterative algorithms.
`error`	The threshold in the change in the Wasserstein Distance between two steps at which to stop the iterative algorithms.
`submaxiter`	The maximum number of iterations for the sub algorithm in the "grid" method.
`suberror`	The threshold of change in each step at which to stop the sub algorithm in the "grid" method.
`substeps`	The step size in the sub algorithm used for the "grid" method.
`sample.points`	The number of time points the geodesic is evaluated and returned.

`visual`	A boolean indicating whether a visual representation of the components should be generated. If set to TRUE, then a series of png images will be created.
`gif`	A boolean specifying whether the png-images should be used to create a gif, animating the principal components. This requieres the animation package and imagemagik to be installed and be availible to R.
`H`	The 2x2 bandwidth matrix for the computation of the kernel density estimator.
`grid.size`	The size of the two dimensional grid on which the kde is evaluated.
`file`	String determining the name of the exported files.

Details

This is a wrapper function for the different algorithms to perform principal component analysis on images contained in this package. This function takes a directory, imports the files within the directory and then calls image_pca with the imported data. Refer to documentation of image_pca for more information.

Value

A list of the computed principal components sampled at the specified time points is returned. Each of these list contains a list of the position matrices of the geodesics at the numbered time points. If the visual parameters are set to TRUE, then in addition the chosen visualisations will be generated.

References

Seguy, Vivien and Cuturi, Marco;Principal Geodesic Analysis for Probability Measures under the Optimal Transport Metric; Advances in Neural Information Processing Systems 28;2015.
Cazelles, Elsa & Seguy, Vivien & Bigot, Jérémie & Cuturi, Marco & Papadakis, Nicolas; Log-PCA versus Geodesic PCA of histograms in the Wasserstein space; SIAM Journal on Scientific Computing. 40; 2017.
A Linear Optimal Transportation Framework for Quantifying and Visualizing Variations in Sets of Images; International Journal of Computer Vision; Wang, Wei & Slepcev, Dejan & Basu, Saurav & Ozolek, John A. & Rohde, Gustavo K.;2013.
Marco Cuturi and Arnaud Doucet; Fast computation of wasserstein barycenters; In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14); 2014.

`image_pca` *Perform Principal Component Analysis on Image Data.*

Description

Perform Principal Component Analysis on Image Data.

Usage

```
image_pca(imagelist, barycenter = NULL, pca.count = 3,
  method = "log", K = 20, steps = 10^-4, lambda = 1,
  maxiter = 600, error = 10^-7, submaxiter = 1000,
  suberror = 10^-12, substeps = 10^-7, sample.points = 100,
  visual = TRUE, gif = FALSE, H = (1/1000) * matrix(c(1, 0, 0, 1), 2,
  2), grid.size = 100, file = "RPCA")
```

Arguments

`imagelist`	A list of N x M Matrices representing the images that PCA is to be performed on.
`barycenter`	An N x M Matrix representing the barycenter of the images. If non is specified, then the one is approximated iteratively.
`pca.count`	An integer specifying the number of principal components to be computed. Has to be lower than the amount of data images used for the pca.
`method`	A string specifying the method used to compute the pca. The availible options are: "log" The default method. The image measures are mapped to the tangent space at the barycenter and standard euclidean PCA is performed on the linearized data. Afterwards the computed principal components are mapped back to the Wasserstein Space. "sh": This method computes the principal components via a penalized, projected gradient descent method. It uses entropy regularized Sinkhorn distances implemented in the Barycenter package to approximate the Wasserstein Distances. "tp" This method computes the principal components via a penalized, projected gradient descent method. The Wasserstein distance is computed exactly using the implementation in the transport package. "grid" This method computes the principal components via a projected gradient descent method. It assumes the data to be supported on the full pixel grid. If the data has indeed mass on nearly all pixels, than this method is significantly faster, than the "sh" and "tp" method. If the data is rather sparse it is strongly advised not to use this method. WARNING: The iterative methods are volatile to the choice of step sizes and maximum iterations. If one obtains unreasonable results, it is advised to rerun the computation with a smaller step size and higher number of iterations. Additionally the run time of the iterative methods might exceed a day even for moderate image sizes. However, the performance can be improved greatly by running it on clusters for parallelization. The "log" method computes results quickly, but the iterative methods yield better results given sufficient iterations. Hence, it is recommended to use "log" for everyday operations and deploy the other methods when it is important to obtain the best possible results with limited regard to run time.

`K`	Integer used for "sh" and "tp". It specifies the size of the time grid the geodesic is computed on in every iteration. Choosing this too small negatively impacts convergence, but one should be cautious, since computation time scales linearly in K.
`steps`	The step size for the gradient descent used in the iterative methods.
`lambda`	The penalty parameter for the "tp" and "sh" methods. If set to zero generalized geodesics are computed instead of true ones. Otherwise the penalization forces the two velocity fields in the generalized geodesic to be collinear.
`maxiter`	The maximum number of iterations for the iterative algorithms.
`error`	The threshold in the change in the Wasserstein Distance between two steps at which to stop the iterative algorithms.
`submaxiter`	The maximum number of iterations for the sub algorithm in the "grid" method.
`suberror`	The threshold of change in each step at which to stop the sub algorithm in the "grid" method.
`substeps`	The step size in the sub algorithm used for the "grid" method.
`sample.points`	The number of time points the geodesic is evaluated and returned.
`visual`	A boolean indicating whether a visual representation of the components should be generated. If set to TRUE, then a series of png images will be created.
`gif`	A boolean specifying whether the png-images should be used to create a gif, animating the principal components. This requieres the animation package and imagemagik to be installed and be available to R.
`H`	The 2x2 bandwidth matrix for the computation of the kernel density estimator.
`grid.size`	The size of the two dimensional grid on which the kde is evaluated.
`file`	String determining the name of the exported files.

Details

This is a wrapper function for the different algorithms to perform principal component analysis on images contained in this package. This function takes a list of input images with their corresponding barycenter and computes a specified numbers of principal components in the 2-Wasserstein space. It also provides visual representations of those components using series of png images and gifs of the geodesic sampled at discrete time points.

Value

A list of the computed principal components sampled at the specified time points is returned. Each of these lists contains a list of the position matrices of the geodesics at the numbered time points. If the visual parameters are set to TRUE, then in addition the chosen visualisations will be generated.

References

Seguy, Vivien and Cuturi, Marco;Principal Geodesic Analysis for Probability Measures under the Optimal Transport Metric; Advances in Neural Information Processing Systems 28;2015.

Cazelles, Elsa & Seguy, Vivien & Bigot, Jérémie & Cuturi, Marco & Papadakis, Nicolas; Log-PCA versus Geodesic PCA of histograms in the Wasserstein space; SIAM Journal on Scientific

Computing. 40; 2017.
A Linear Optimal Transportation Framework for Quantifying and Visualizing Variations in Sets of Images; International Journal of Computer Vision; Wang, Wei & Slepcev, Dejan & Basu, Saurav & Ozolek, John A. & Rohde, Gustavo K.;2013.
Marco Cuturi and Arnaud Doucet; Fast computation of wasserstein barycenters; In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14); 2014.

Examples

```
res<-image_pca(mnist_zero_mat,mnist_zero_bary_mat)
```

`mnist_zero` *This data is used for the examples.*

Description

This is a small subset of the MNIST(Modified National Institute of Standards and Technology database) data set consisting of over 60000 images of handwritten numbers. This set only contains 10 samples from the number zero. The images have been reduced to be the size of 14x14.

Format

A list of two. The first entries holds the positions of the positions of the weighted point pattern. The second entries hold the weights.

Source

`http://yann.lecun.com/exdb/mnist/`

`mnist_zero_bary` *This data is used for the examples.*

Description

This is the barycenter of the mnist_zero data approximated on a support of the size 500.

Format

A list of two. The first entry corresponds to the positions of the barycenter and the second to its weights.

Source

`http://yann.lecun.com/exdb/mnist/`

`mnist_zero_bary_mat` *This data is used for the examples.*

Description

This is the barycenter of the mnist_zero_mat data approximated on a 14x14 grid.

Format

A 14x14 matrix containing the weights of the barycenter.

Source

`http://yann.lecun.com/exdb/mnist/`

`mnist_zero_mat` *This data is used for the examples.*

Description

This is a small subset of the MNIST(Modified National Institute of Standards and Technology database) data set consisting of over 60000 images of handwritten numbers. This set only contains 10 samples from the number zero. The images have been reduced to be the size of 14x14.

Format

A list containing 14x14 matrices representing the images.

Source

`http://yann.lecun.com/exdb/mnist/`

pca_grid — *Computes principal components of a data set of weighted point measures in R^d.*

Description

Takes a set of weighted point measures in R^d and their barycenter and computes their principal geodesics based on this.

Usage

```
pca_grid(data.weight, bary.weight, support, sup_size, t0.grid = 0,
  pca.count = 1, maxiter = 500, submaxiter = 100, tau.t = 10^-8,
  tau.v = 10^-6)
```

Arguments

`data.weight`	A list of NxM matrices containing the weights of the data measures.
`bary.weight`	A NxM matrix containing the weights of the barycenter of the data.
`support`	A (NxM)x2 matrix containing the positions of the grid points. Make sure to align this properly with the matrix positions of the weights.
`sup_size`	A two-dimensional vector containing the value N and M.
`t0.grid`	A tuning parameter shifting the center of the geodesic. Can remain at default. Might be set to an vector containing a one dimensional grid centered around zero to obtain a slightly better approximation, at the cost of a linear scaling in the number of grid points.
`pca.count`	Integer specifying the number of principal geodesic to be computed.
`maxiter`	Integer specifying the maximum number of iterations in the gradient descent.
`submaxiter`	Integer specifying the maximum number of iterationns in the sub-algorithm.
`tau.t`	Real value, specifying the step size in the sub algorithm. This should be reduced, if one encounters unreasonble results from the gradient descent.
`tau.v`	Real value, specifying the step size in the gradient descent. This should be reduced, if one encounters unreasonble results from the gradient descent.

Details

This functions computes principal geodesics in the 2-Wasserstein space. It takes a set of discretized or discrete input measures and computes iteratively principal directions using a gradient descent method. It then returns the velocity fields describing the principal geodesics in the tangent space at the barycenter of the data. This method requires the data to be supported on a rectangular grid in [0,1]. If one wants to consider data with a more sparse or less regular support, it should be refered to the pca_ws or image_pca function.

Value

This function returns a list with two entries. The first entry is opt_times. This is a list containing the optimal time points for each image for each principal geodesic respectively. This is for each pair of a data measure and a geodesic the t in [0,1], for which the distance between the geodesic and the measure is minimal.
The second entry is opt_velocity. This is a list containing a velocity field for each computed principal geodesic. They describe the movement of the mass of the barycenter along each geodesic, respectively. principal geodesic respectively. The positions of the k-th geodesic at time t in [-1,1] is given by support+t * opt_velocity[[k]]. The weights are given by the weights of the barycenter at the corresponding support points.

References

#' Cazelles, Elsa & Seguy, Vivien & Bigot, Jérémie & Cuturi, Marco & Papadakis, Nicolas; Log-PCA versus Geodesic PCA of histograms in the Wasserstein space; SIAM Journal on Scientific Computing. 40; 2017.

Examples

```
grid<-WassersteinPCA:::build.grid(14,14)
res<-pca_grid(mnist_zero_mat,mnist_zero_bary_mat,grid,c(14,14))
```

---

pca_log — *Computes principal components of a data set of weighted point measures in R^2.*

Description

Computes principal components in the 2-Wasserstein Space

Usage

```
pca_log(image.weights, image.pos, bary.weights, bary.pos, pca.count = 3)
```

Arguments

`image.weights`	A list of vectors containing the weights of the individial point measures/image pixels. All vector entries are assumed to be greater 0.
`image.pos`	A list of Nx2 matrices containing the positions of the data measures/pixels in [0,1]^2 in their respective rows.
`bary.weights`	A vector specifying the weights of the barycenter. All vector entries are assumed to be greater 0.
`bary.pos`	A Nx2 matrix specifying the positions of the barycenter.
`pca.count`	An integer specifying the number of principal components to be computed.

Details

This function computes the principal components of a data set consisting of weighted point measures in R^2. To do this it first maps the data to the tangent space at the barycenter and then performs standard euclidean PCA on this space. Afterwards the resulting components are mapped back to the 2-Wasserstein Space.

Value

The algorithm returns a list containing the entries components and evalues. components contains a list of velocity fields describing the respective principal components. evalues containts the eigenvalues corresponding to each principal component. This is also the variance of the data in this specific direction. The velocity fields paramatrizing the c-th principal component is then given by the product of the c-th component and the square root of the c-th eigenvalue.

References

Cazelles, Elsa & Seguy, Vivien & Bigot, Jérémie & Cuturi, Marco & Papadakis, Nicolas; Log-PCA versus Geodesic PCA of histograms in the Wasserstein space; SIAM Journal on Scientific Computing. 40; 2017.
A Linear Optimal Transportation Framework for Quantifying and Visualizing Variations in Sets of Images; International Journal of Computer Vision; Wang, Wei & Slepcev, Dejan & Basu, Saurav & Ozolek, John A. & Rohde, Gustavo K.;2013.

Examples

```
components<-pca_log(mnist_zero[[2]],mnist_zero[[1]],mnist_zero_bary[[2]],mnist_zero_bary[[1]])
for (c in 1:3){
 V<-components$components[[c]]*sqrt(components$evalues[c])
 name<-paste("RtestPCA","_pcanr_",c,sep="")
 visual_geodesic(mnist_zero_bary[[1]],mnist_zero_bary[[2]],V,V,timesteps = 100,
 grid.size=28*4,create.gif = FALSE,file.name=name)
}
```

pca_ws *Computes principal components of a data set of weighted point measures in R^d.*

Description

Takes a set of weighted point measures in R^d and their barycenter and computes their principal geodesics based on this.

Usage

```
pca_ws(data.pos, data.weights, bary.pos, bary.weights, stepsize, lambda,
  pca.count, K = 20, max_iter, error)
```

Arguments

`data.pos`	A list of Nxd matrices of data positions.
`data.weights`	A list of N-dim vectors of the data weights.
`bary.pos`	A Nxd matrix of the positions of the barycenter
`bary.weights`	A N-dim vector of the weights of the barycenter
`stepsize`	The step size of the gradient descent algorithm. Note choosing this too high might lead to divergence, but choosing it too small might lead to long runtimes.
`lambda`	This is a regularisation parameter which is greater or equal to zero. Choosing this greater than zero will lead to a regularization step, where the velocity fields are forced to be collinear.
`pca.count`	Integer determining the amount of principal components to be computed.
`K`	Integer determining the size of the time grid in the computation. The defauls usually works fine, but in case of bad convergence properties it might be helpful to increase this paramter. Note that the run time is linear in K.
`max_iter`	Integer specifying the maximal amount of iterations in the gradient descent method.
`error`	The value of change between iterations at which the gradient descent should terminate.

Details

Let Y be the positions of the barycenter in R^d, then each principal geodesic can be parametrized as G(t)=Y-V1+t(V1+V2) for t in [0,1] and two velocity fields V1 and V2. This corresponds to the definition of a generalized geodesic. If one wishes to obtain a true geodesic, one can use the regularization parameter lambda to force the fields to be collinear, but for visual analysis this is often not neccesary. This function takes data sets consisting of sums of weighted point measures and computes V1 and V2 via projected gradient descent. For each pca that is to be computed the corresponding V1 and V2 are returned. Since we are restricted to point measures, the velocity fields are simple Nxd matrices here, where each row corresponds to a vector in R^d along which the mass is transported on the geodesic.

Value

The function returns an array of the size pca.countx2xNxd, where result[k,1,,] contains the first velocity field of the k-th pca and result[k,2,,] contains the second velocity field of the k-th pca.

References

Seguy, Vivien and Cuturi, Marco; Principal Geodesic Analysis for Probability Measures under the Optimal Transport Metric; Advances in Neural Information Processing Systems 28; 2015.

Examples

```
res<-pca_ws(mnist_zero[[1]],mnist_zero[[2]],mnist_zero_bary[[1]],mnist_zero_bary[[2]],
10^(-4),0,1,20,400,10^(-6))
visual_geodesic(mnist_zero_bary[[1]],mnist_zero_bary[[2]],res[1,1,,],res[1,2,,])
```

`pca_ws_tp` *Computes principal components of a data set of weighted point measures in R^d.*

Description

Takes a set of weighted point measures in R^d and their barycenter and computes their principal geodesics based on this.

Usage

```
pca_ws_tp(data.pos, data.weights, bary.pos, bary.weights, stepsize, lambda,
  pca.count, K = 20, max_iter, error)
```

Arguments

`data.pos`	A list of Nxd matrices of data positions.
`data.weights`	A list of N-dim vectors of the data weights.
`bary.pos`	A Nxd matrix of the positions of the barycenter
`bary.weights`	A N-dim vector of the weights of the barycenter
`stepsize`	The step size of the gradient descent algorithm. Note choosing this too high might lead to divergence, but choosing it too small might lead to long runtimes.
`lambda`	This is a regularisation parameter which is greater or equal to zero. Choosing this greater than zero will lead to a regularization step, where the velocity fields are forced to be collinear.
`pca.count`	Integer determining the amount of principal components to be computed.
`K`	Integer determining the size of the time grid in the computation. The defauls usually works fine, but in case of bad convergence properties it might be helpful to increase this paramter. Note that the run time is linear in K.
`max_iter`	Integer specifying the maximal amount of iterations in the gradient descent method.
`error`	The value of change between iterations at which the gradient descent should terminate.

Details

Let Y be the positions of the barycenter in R^d, then each principal geodesic can be parametrized as G(t)=Y-V1+t(V1+V2) for t in [0,1] and two velocity fields V1 and V2. This corresponds to the definition of a generalized geodesic. If one wishes to obtain a true geodesic, one can use the regularization parameter lambda to force the fields to be collinear, but for visual analysis this is often not neccesary. This function takes data sets consisting of sums of weighted point measures and computes V1 and V2 via projected gradient descent. For each pca that is to be computed the corresponding V1 and V2 are returned. Since we are restricted to point measures, the velocity fields are simple Nxd matrices here, where each row corresponds to a vector in R^d along which the mass

is transported on the geodesic.
This function is mostly identical to the pca_ws function. The only difference is, that the former uses an entropy regularization to approximate the Wasserstein-Distances, while this function explicitly solves the corresponding to optimal transport problem by calling the transport-package. The pca_ws_tp function is therefore slightly slower, but also a little more precise.

Value

The function returns an array of the size pca.countx2xNxd, where result[k,1,,] contains the first velocity field of the k-th pca and result[k,2,,] contains the second velocity field of the k-th pca.

References

Seguy, Vivien and Cuturi, Marco; Principal Geodesic Analysis for Probability Measures under the Optimal Transport Metric; Advances in Neural Information Processing Systems 28; 2015.

Examples

```
res<-pca_ws_tp(mnist_zero[[1]],mnist_zero[[2]],mnist_zero_bary[[1]],mnist_zero_bary[[2]],
10^(-4),0,1,20,400,10^(-6))
visual_geodesic(mnist_zero_bary[[1]],mnist_zero_bary[[2]],res[1,1,,],res[1,2,,])
```

`visual_geodesic` *Creates a visualization of a generalized geodesic in the 2-Wasserstein-Space.*

Description

Creates a visualization of a general geodesic in the 2-Wasserstein-Space. It is parametrized as Y-V1+t(V1+V2), where Y is the matrix of positions of the barycenter and V1 and V2 are velocity fields. It creates either a series of images or a gif to visualize the geodesic. The creation of a gif requieres the installation of Imagemagik and the animation package.

Usage

```
visual_geodesic(bary.pos, bary.weights, V1, V2, timesteps = 100,
  H = (1/1000) * matrix(c(1, 0, 0, 1), 2, 2), grid.size = 100,
  create.gif = TRUE, create.images = TRUE, file.name = "RGeodesic",
  delay.time = 20)
```

Arguments

`bary.pos`	Nx2 Matrix of Positions of the barycenter in two dimensional space.
`bary.weights`	A vector of the weights of each position of the barycenter
`V1`	A Nx2 matrix specifying the first velocity field
`V2`	A Nx2 matrix specifying the second velocity field

`timesteps`	The amount of time steps between t=0 and t=1 at which the Geodesic is computed.
`H`	The 2x2 bandwidth matrix for the computation of the kernel density estimator.
`grid.size`	The size of the two dimensional grid on which the kde is evaluated.
`create.gif`	Boolean determining whether the results should be exported as a gif. Note that this requieres the installation of the animation package and imagemagik.
`create.images`	Boolean determining whether the results should be exported as a series of png-images.
`file.name`	String determining the name of the exported files.
`delay.time`	The time between two frames in the generated gif.

Value

The function returns an array of the size gridsize x gridsize x (2+timesteps) containing the evaluation of the kde on the grid at each timepoint. If create.image=TRUE, then the function exports the results as a series of png images at the corresponding time steps. If create.gif=TRUE, then it creates a gif from this images to visualize the transport corresponding to the geodesic. Note that create.gif and create.images operate independent from each other, i.e. you need to set both values to TRUE if you want both outputs.

Examples

```
components<-pca_log(mnist_zero[[2]],mnist_zero[[1]],mnist_zero_bary[[2]],mnist_zero_bary[[1]])
for (c in 1:3){
 V<-components$components[[c]]*sqrt(components$evalues[c])
 name<-paste("RtestPCA","_pcanr_",c,sep="")
 visual_geodesic(mnist_zero_bary[[1]],mnist_zero_bary[[2]],V,V,timesteps = 100,grid.size=28*4,
 create.gif = FALSE,file.name=name)
}
```

`wsbary_cont` *Computes an approximation of the 2-Wasserstein-Barycenter of discrete input measures.*

Description

This function computes iteratively the best approximation of the data's Wasserstein-Barycenter supported on a support of a specified size.

Usage

```
wsbary_cont(data.weights, data.pos, sup_size, maxiter = 50,
  sub_iter = 5, theta = 0.5)
```

Arguments

`data.weights`	A list of vectors specifying the weights of the data measures. Each vector is assumed to sum to one.
`data.pos`	A list of Mxd matrices, specifying the positions of the data measures. Different measures do not have to have the same support size and do not need to be supported on some grid.
`sup_size`	A positive integer specifying the size of the support of the approximated barycenter.
`maxiter`	A positive integer specifying the maximum number of iterations in the algorithm.
`sub_iter`	A positive integer specifying the number of iterations in the weight-updates.
`theta`	A real value in [0,1] used as a tuning parameter for the gradient descent.

Value

The function returns a list with two entries. The first entry contains the positions of the computed barycenter and the second entry contains the corresponding weights.

References

Marco Cuturi and Arnaud Doucet; Fast computation of wasserstein barycenters; In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14); 2014.

Examples

```
res<-wsbary_cont(mnist_zero[[2]],mnist_zero[[1]],100,30,2)
```

Index