Package ‘kernelTDA’

July 3, 2019

Type Package
Title Statistical Learning with Kernel for Persistence Diagrams
License GPL-3
Version 0.1.1
Date 2019-07-01
Maintainer Tullia Padellini <tullia.padellini@uniroma1.it>
Description Provides tools for exploiting topological information into standard statistical learning algorithms. To this aim, this package contains the most popular kernels defined on the space of persistence diagrams, and persistence images. Moreover, it provides a solver for kernel Support Vector Machines problems, whose kernels are not necessarily positive semidefinite, based on the C++ library 'LIBSVM' <https://www.csie.ntu.edu.tw/~cjlin/libsvm/> and on its R implementation 'e1071'. Additionally, it allows to compute Wasserstein distance between persistence diagrams with an arbitrary ground metric, building an R interface for the C++ library 'HERA' <https://bitbucket.org/grey_narn/hera/src/master/>.
Imports Rcpp (>= 1.0.1), mvtnorm, Rdpack, methods, stats
Suggests TDA, knitr, rmarkdown, SparseM, Matrix, kernlab, viridis
LinkingTo Rcpp, RcppEigen, BH
RoxygenNote 6.1.1
VignetteBuilder knitr
RdMacros Rdpack
NeedsCompilation yes
Author Tullia Padellini [aut, cre], Francesco Palini [aut], Pierpaolo Brutti [ctb], David Meyer [ctb, cph] (libsvm to R code (e1071 package)), Chih-Chung Chang [ctb, cph] (LIBSVM C++ code), Chih-Chen Lin [ctb, cph] (LIBSVM C++ code), Michael Kerber [ctb, cph] (HERA C++ code),
Dmitriy Morozov [ctb, cph] (HERA C++ code),
Arnur Nigmetov [ctb, cph] (HERA C++ code)

Repository CRAN
Date/Publication 2019-07-03 16:20:02 UTC

R topics documented:

gaus.kernel ................................................................. 2
kerneTDA ................................................................. 3
krein.svm ................................................................. 3
krein.svm.default .......................................................... 4
lapl.kernel ................................................................. 6
pers.image ................................................................. 7
pf.kernel ................................................................. 8
pss.kernel ................................................................. 9
sw.kernel ................................................................. 10
wass.kernel ............................................................... 11
wasserstein.distance ..................................................... 12

Index 13

<table>
<thead>
<tr>
<th>gaus.kernel</th>
<th>Geodesic Gaussian Kernel (GGK)</th>
</tr>
</thead>
</table>

Description
Computes the Geodesic Gaussian Kernel (GGK) between persistence diagrams.

Usage

```r
gaus.kernel(d1, d2 = NULL, h, dimension)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time) or a list of diagrams.</td>
</tr>
<tr>
<td>d2</td>
<td>A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).</td>
</tr>
<tr>
<td>h</td>
<td>bandwidth of the kernel.</td>
</tr>
<tr>
<td>dimension</td>
<td>The dimension of the topological feature (0 for connected components, 1 for cycles etc).</td>
</tr>
</tbody>
</table>

Value

If d1 is a list of Persistence Diagrams, this function returns a matrix whose (i,j) entry is the GGK computed in (d1[[i]], d2[[j]]), otherwise it returns the value for the GGK computed in (d1, d2).
The kernelTDA package provides implementation for the most common kernels defined in the space of Persistence Diagrams. These kernel can then be used in most kernel based methods.

### Description

Solves a kernelized Support Vector Machine in the case where the kernel used may not be positive semidefinite.

### Usage

```
krein.svm(kernelmat, ...)  
```

### Arguments

- `kernelmat` : the kernel matrix computed for all observations
- `...` : additional parameters, see `krein.svm.default` for more details on how to use this function

### Details

This function implements the Krein Support Vector Machine solver as defined by Loosli et al. (2015). The implementation of the solver is a modified version of the popular C++ library 'LIB-SVM', while the connection to 'R' heavily relies on the 'R'-package `e1701`.  

---

**Examples**

```
diag1 <- matrix(c(1,1,0,2,3,2,2,5,4), ncol = 3, byrow = FALSE)  
diag2 <- matrix(c(1,0,1,1,2), ncol = 3, byrow = FALSE)  
gaus.kernel(d1 = diag1, d2 = diag2, h = 1, dimension = 1)  
```
Value

An object of class \texttt{krein.svm.default} containing the fitted model, including:

- \texttt{SV} a matrix containing the Support Vectors
- \texttt{index} index of the resulting support vectors in the data matrix
- \texttt{coefs} a matrix containing corresponding coefficients times the training labels
- \texttt{rho} value of the (negative) intercept

Author(s)

Tullia Padellini, Francesco Palini, David Meyer. The included C++ library LIBSVM is authored by Chih-Chung Chang and Chih-Jen Lin

References


Examples

```r
library(TDA)
set.seed(123)
foo.data = list()
for(i in 1:20){
  foo = circleUnif(100)
  foo.data[[i]] = ripsDiag(foo, 1,1)$diagram
}
for(i in 21:40){
  foo = cbind(runif(100), runif(100))
  foo.data[[i]] = ripsDiag(foo, 1,1)$diagram
}
GSWkernel = gaus.kernel(foo.data, h =1, dimension = 1, q = 2)
GGKclass = krein.svm(kernelmat = GSWkernel, y = rep(c(1,2), c(20,20)))
```

---

\texttt{krein.svm.default} \hspace{1cm} Krein Support Vector Machine

Description

Solves a kernelized Support Vector Machine in the case where the kernel used may not be positive semidefinite.
## Default S3 method:

```r
krein.svm(kernelmat = NULL, y = NULL, cost = 1,
          class.weights = NULLL, cross = 0, probability = FALSE,
          fitted = TRUE, subset, ...)
```

### Arguments

- **kernelmat**: the kernel matrix computed for all observations
- **y**: a vector of labels
- **cost**: cost of violating the constraint
- **class.weights**: a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named. Specifying "inverse" will choose the weights inversely proportional to the class distribution.
- **cross**: number of fold in a k-fold cross validation
- **probability**: logical indicating whether the model should allow for probability predictions (default: FALSE).
- **fitted**: logical indicating whether the fitted values should be computed and included in the model or not (default: TRUE)
- **subset**: an index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
- **...**: additional parameters

### Details

This function implements the Krein Support Vector Machine solver as defined by Loosli et al. (2015). The implementation of the solver is a modified version of the popular C++ library 'LIBSVM', while the connection to 'R' heavily relies on the 'R'-package **e1701**.

### Value

An object of class **krein.svm** containing the fitted model, including:

- **SV**: a matrix containing the Support Vectors
- **index**: index of the resulting support vectors in the data matrix
- **coefs**: a matrix containing corresponding coefficients times the training labels
- **rho**: value of the (negative) intercept

### Author(s)

Tullia Padellini, Francesco Palini, David Meyer. The included C++ library LIBSVM is authored by Chih-Chung Chang and Chih-Jen Lin)
References


Examples

```r
# library(TDA)
# set.seed(123)
# foo.data = list()
# for(i in 1:20){
#   foo = circleUnif(100)
#   foo.data[[i]] = ripsDiag(foo, 1,1)$diagram
# } for(i in 21:40){
#   foo = cbind(runif(100), runif(100))
#   foo.data[[i]] = ripsDiag(foo, 1,1)$diagram
# }
# GSWkernel = gaus.kernel(foo.data, h =1, dimension = 1, q = 2)
# GGKclass = krein.svm(kernelmat = GSWkernel, y = rep(c(1,2), c(20,20)))
```

---

**lapl.kernel**

*Geodesic Laplacian Kernel (GLK)*

Description

Computes the Geodesic Laplacian Kernel (GLK) between persistence diagrams.

Usage

`lapl.kernel(d1, d2 = NULL, h, dimension)`

Arguments

- `d1`: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time) or a list of diagrams
- `d2`: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
- `h`: Bandwidth of the kernel.
- `dimension`: The dimension of the topological feature (0 for connected components, 1 for cycles etc)
pers.image

Value
If \( d_1 \) is a list of Persistence Diagrams, this function returns a matrix whose \((i,j)\) entry is the GLK computed in \((d_1[i], d_2[j])\), otherwise it returns the value for the GLK computed in \((d_1, d_2)\).

Author(s)
Tullia Padellini

References

Examples
```r
diag1 <- matrix(c(1,1,0,2,3,2,2.5,4), ncol = 3, byrow = FALSE)
diag2 <- matrix(c(1,1,0,1,1,2), ncol = 3, byrow = FALSE)
laplkernel(d1 = diag1, d2 = diag2, h = 1, dimension = 1)
```

pers.image Persistence Image

Description
Compute the Persistence Image for a given diagram, using piecewise linear weight functions and Gaussian baseline distribution.

Usage
```r
pers.image(d1, nbins, dimension, h)
```

Arguments
- **d1**: A persistence diagram, in the form of a matrix with 3 columns (first one is the dimension, second is the birth-time, last one is the death-time).
- **nbins**: Number of bins for the discretization of the Persistence Surface into the Persistence Image.
- **dimension**: Dimension of the topological features of interest (0 for connected components, 1 for cycles etc).
- **h**: Standard deviation of the Gaussian baseline used to compute the Persistence Surface.

Value
A \( nbins \times nbins \) matrix containing the Persistence Image.
**Author(s)**

Tullia Padellini

**References**


**Examples**

```r
diag1 <- matrix(c(1,1,1,0,2,3,2,2,5,4), ncol = 3, byrow = FALSE)
colnames(diag1) <- c("dimension", "birth", "death")
p1 <- pers.image(d1 = diag1, nbins = 20, dimension = 1, h = 1)
image(p1)
```

**pf.kernel**

*persistence Fisher Kernel (PFK)*

**Description**

Computes the Persistence Fisher Kernel (PFK) between persistence diagrams.

**Usage**

```r
pf.kernel(d1, d2 = NULL, h, dimension, sigma)
```

**Arguments**

- **d1**: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time) or a list of diagrams.
- **d2**: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
- **h**: bandwidth of the PFK.
- **dimension**: The dimension of the topological feature (0 for connected components, 1 for cycles etc)
- **sigma**: standard deviation of the base Gaussian Kernel.

**Value**

If d1 is a list of Persistence Diagrams, this function returns a matrix whose (i,j) entry is the PFK computed in (d1[[i]], d2[[j]]), otherwise it returns the value for the PFK computed in (d1, d2).

**Author(s)**

Tullia Padellini
References


Examples

d1 <- matrix(c(1,1,0,2,3,2,2,5,4), ncol = 3, byrow = FALSE)
d2 <- matrix(c(1,1,0,1,1,2), ncol = 3, byrow = FALSE)
pf.kernel(d1 = diag1, d2 = diag2, h = 1, dimension = 1, sigma = 1)

Description

Computes the Persistence Scale Space Kernel (PSSK) between persistence diagrams

Usage

pss.kernel(d1, d2 = NULL, h, dimension)

Arguments

d1 A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time) or a list of diagrams
d2 A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
h bandwidth of the kernel.
dimension The dimension of the topological feature (0 for connected components, 1 for cycles etc).

Value

If d1 is a list of Persistence Diagrams, this function returns a matrix whose (i,j) entry is the PSSK computed in (d1[[i]], d2[[j]]), otherwise it returns the value for the PSSK computed in (d1, d2).

Author(s)

Tullia Padellini

References

Examples

```r
diag1 <- matrix(c(1,1,0,2,3,2,2,5,4), ncol = 3, byrow = FALSE)
diag2 <- matrix(c(1,1,0,1,1,2), ncol = 3, byrow = FALSE)
pssNkernel(d1 = diag1, d2 = diag2, h = 1, dimension = 1)
```

---

**sw.kernel**

*Persistence Sliced Wasserstein Kernel (SWK)*

Description

Computes the Persistence Sliced Wasserstein Kernel (SWK) between persistence diagrams.

Usage

```r
sw.kernel(d1, d2 = NULL, h, dimension, M = 10)
```

Arguments

- **d1**: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time) or a list of diagrams
- **d2**: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
- **h**: bandwidth of the kernel
- **dimension**: The dimension of the topological feature (0 for connected components, 1 for cycles etc)
- **M**: number of directions on which to approximate the Sliced Wasserstein Distance

Value

If `d1` is a list of Persistence Diagrams, this function returns a matrix whose (i,j) entry is the SWK computed in (`d1[[i]], d2[[j]])`, otherwise it returns the value for the SWK computed in (`d1, d2`)

Author(s)

Tullia Padellini

References


Examples

```r
diag1 <- matrix(c(1,1,0,2,3,2,2,5,4), ncol = 3, byrow = FALSE)
diag2 <- matrix(c(1,1,0,1,1,2), ncol = 3, byrow = FALSE)
sw.kernel(d1 = diag1, d2 = diag2, h = 1, dimension = 1)
```
Description

Computes the $L_{\infty}$ q-Wasserstein Kernel (WK) between persistence diagrams.

Usage

\[
\text{wass.kernel}(d1, d2 = \text{NULL}, h, \text{dimension}, q)
\]

Arguments

- **d1**: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time) or a list of diagrams.
- **d2**: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
- **h**: bandwidth of the kernel.
- **dimension**: The dimension of the topological feature (0 for connected components, 1 for cycles etc).
- **q**: order of the q-Wasserstein distance.

Value

If \(d1\) is a list of Persistence Diagrams, this function returns a matrix whose (i,j) entry is the WK computed in \((d1[[i]], d2[[j]])\), otherwise it returns the value for the WK computed in \((d1, d2)\).

Author(s)

Tullia Padellini

Examples

\[
\text{diag1} \leftarrow \text{matrix}(c(1,1,1,0,2,3,2,2,5,4), \text{ncol} = 3, \text{byrow} = \text{FALSE}) \\
\text{diag2} \leftarrow \text{matrix}(c(1,1,0,1,1,2), \text{ncol} = 3, \text{byrow} = \text{FALSE}) \\
\text{wass.kernel}(d1 = \text{diag1}, d2 = \text{diag2}, h = 1, \text{dimension} = 1, q = 2)
\]
### Description

Compute the q-Wasserstein distance between persistence diagrams using an arbitrary L_p norm as ground metric.

### Usage

```r
wasserstein.distance(d1, d2, dimension, q, p = 2)
```

### Arguments

- `d1`: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
- `d2`: A persistence diagram (matrix with 3 col where the first one is the dimension, the second is the birth-time and the third is the death-time).
- `dimension`: Dimension of the topological features of interest (0 for connected components, 1 for cycles etc).
- `q`: Order of the q-Wasserstein distance.
- `p`: Order of the L_p norm to be used as a ground metric in the computation of the Wasserstein distance.

### Details

This function provides an R interface for the efficient C++ library ‘HERA’ by Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov ([https://bitbucket.org/grey_narn/hera/src/master/](https://bitbucket.org/grey_narn/hera/src/master/)).

### Value

The value for the L_p q-Wasserstein between `d1` and `d2`.

### Author(s)

Tullia Padellini, Francesco Palini. The included C++ library is authored by Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov

### References


### Examples

```r
diag1 <- matrix(c(1,1,1,0,2,3,2,5,4), ncol = 3, byrow = FALSE)
diag2 <- matrix(c(1,1,0,1,1,2), ncol = 3, byrow = FALSE)
wasserstein.distance(diag1, diag2, dimension = 1, q = 1, p = 2)
```
Index

gaus.kernel, 2

kernelTDA, 3
kernelTDA-package (kernelTDA), 3
krein.svm, 3
krein.svm.default, 3, 4

lapl.kernel, 6

pers.image, 7
pf.kernel, 8
pss.kernel, 9

sw.kernel, 10

wass.kernel, 11
wasserstein.distance, 12