Spectralclust Package

01 January 2021

Description: Spectral clustering for SAS

Authors: Michal Pastuszka <michal.pastuszka98@gmail.com>

Wojciech Bogucki <>

Version: 0.5

Modules:

Overview:

The `spectralclust` package provides an implementation of the spectral clustering algorithm written natively in SAS. Spectral clustering is an algorithm utilizing spectral graph theory to improve clustering results.

This algorithm is based on a set of preprocessing steps applied to the data, which then can be clustered using an ordinary clustering method, such as k-means. For that reason we recommend using the `spccSpectralize` function which provides a wrapper for the preprocessing steps represented by other functions in the package. You can then save the resulting matrix to a dataset, and apply a clustering algorithm of your choice, such as the one provided by the FastClus procedure provided by SAS.

Before you use the `spectralclust` package, you must install it by using the PACKAGE INSTALL statement. For example, if the ZIP file is located in the directory C:\Packages, then the following statement installs the package:

```
proc iml;
package install "C:\Packages\spectralclust.zip";
quit;
```

Data Sets

The `spectralclust` package contains a toy dataset.

The **jain** [1] data set represents two crescent shaped groups of points, that present a simple case, where an algorithm such as k-means fails to recognize them correctly, while spectral clustering allows for precise separation of the groups :

- VAR1 and VAR2 variables contain the x and y coordinates of points in 2-D space
- VAR3 represents, to which crescent the point belongs.

spccSpectralize Function

Syntax

spccSpectralize(m, nclus, laplacian='normalizedRW', neighborhood_fun='gaussian', sigma=1, neighborhood_type='knn', k=10);

Parameters

m input matrix storing observations in rows or a similarity matrix. Must be numeric. When using a similarity matrix specify neighborhood_fun as 'none'. Otherwise choose another option.

nclus target number of clusters. It coresponds to the number of eigenvectors used.

laplacian type of laplacian used in the algorithm. Default = 'normalizedRW'. Possible options are:

'normalizedRW' - recommended in most cases - default

'normalizedSym'

'unnormalized' - usually not recommended

neighborhood_fun neighborhood function used to weight the similarity graph. Default = 'gaussian'. Currently possible options are:

'none' - don't compute a similarity matrix. Use when providing a similarity matrix as input

'gaussian' - a gaussian similarity function

'neg_euclid' - negative euclidean distances.

sigma sigma parameter for 'gaussian' neighborhood_fun. Affects the width of neighborhood. Default = 1

neighborhood_type type of neighborhood used. Default = 'knn'. Possible values:

'complete' - all vertices are connected

'knn' - vertices are connected only if one of them belongs to the k nearest neighbors of the other

'mutual_knn' - vertices are connected only if both of them belong to the k nearest neighbors of the other

k k parameter used by 'knn' and 'mutual_knn' neighborhood_types. Default = 10

Description

The function processes the data according to the spectral clustering algorithm, providing a wrapper for the rest of the package. It create eigenvectors of a graph laplacian matrix given a dataset for use in clustering. This eigenvectors can then be used as an input to a clustering procedure.

Example

```
m = {0 1, 0 2, 0 3, 0 4};

out = spccSpectralize(m, 2, 'normalizedRW', 'gaussian', 0.5,
'knn', 2);
print out;
```

spccNegEuclidNeigh Function

Syntax

spccNegEuclidNeigh(m);

Parameters

m a matrix of observations stored in rows. Must be numeric

Description

Computes the pairwise neighborhood values between observations using negative euclidean pairwise distances. Creates a 2-D matrix containing the values for each pair of observations, and zeros on the main diagonal

Example

```
m = {0 0,  0 1, 0 2};

out = spccNegEuclidNeigh(m);

print out;
```

spccGaussNeigh Function

Syntax

spccNegEuclidNeigh(m, sigma=1);

Parameters

m a matrix of observations stored in rows. Must be numeric

sigma *a parameter dictating the width of the neighborhood*

Description

Computes the pairwise neighborhood values between observations using the gaussian neighborhood function. Creates a 2-D matrix containing the values for each pair of observations, and zeros on the main diagonal

Example

```
m = {0 0,  0 1, 0 2};

out = spccGaussNeigh(m, 0.5);

print out;
```

spccKNNNeigh Function

Syntax

spccKNNNeigh(neigh, k=10);

Parameters

neigh a neighborhood matrix

k *number of nearest neghbors to leave connected. default = 10*

Description

Transforms a complete neighborhood matrix to a knn neighborhood matrix connecting only vertices where at least one belongs to the k nearest neighbors of the other

Example

```
m = {0 3 2 1, 3 0 3 2, 2 3 0 3, 1 2 3 0};

out = spccKNNNeigh(m, 1);

print out;
```

spccMutKNNNeigh Function

Syntax

spccMutKNNNeigh(neigh, k=10);

Parameters

neigh a neighborhood matrix

k *number of nearest neghbors to leave connected. default = 10*

Description

Transforms a complete neighborhood matrix to a mutual knn neighborhood matrix connecting only vertices that both belong to the k nearest neighbors of the other. May lead to creating vertices with no corresponding edges, which causes the algorithm to fail. In that case use a higher k value or the `spccKNNNeigh` function instead.

Example

```
m = {0 3 2 1, 3 0 3 2, 2 3 0 3, 1 2 3 0};

out = spccMutKNNNeigh(m, 1);

print out;
```

spccEigenLSym Function

Syntax

spccEigenLSym(m, nvecs);

Parameters

m a neighborhood matrix

nvecs *number of eigenvectors to return. It is recommended to use number of vectors equal to number of clusters searched.*

Description

Compute eigenvectors of the normalized LSym laplacian as described in [2], based on a neighborhood matrix. These eigenvectors can then be used to cluster the data

Example

```
m = {0 3 2 0, 3 0 3 2, 2 3 0 3, 0 2 3 0};

out = spccEigenLSym(m, 3);

print out;
```

spccEigenLRW Function

Syntax

spccEigenLRW(m, nvecs);

Parameters

m a neighborhood matrix

nvecs *number of eigenvectors to return. It is recommended to use number of vectors equal to number of clusters searched.*

Description

Compute eigenvectors of the normalized LRW laplacian as described in [2], based on a neighborhood matrix. These eigenvectors can then be used to cluster the data

Example

```
m = {0 3 2 0, 3 0 3 2, 2 3 0 3, 0 2 3 0};

out = spccEigenLRW(m, 3);

print out;
```

spccEigenL Function

Syntax

spccEigenL(m, nvecs);

Parameters

m	a neighborhood matrix
nvecs	*number of eigenvectors to return. It is recommended to use number of vectors equal to number of clusters searched.*

Description

Compute eigenvectors of the unnormalized laplacian as described in [2], based on a neighborhood matrix. These eigenvectors can then be used to cluster the data

Example

```
m = {0 3 2 0, 3 0 3 2, 2 3 0 3, 0 2 3 0};

out = spccEigenL(m, 3);

print out;
```

spccFastClus Function

Syntax

spccFastClus(m, nclus, laplacian='normalizedRW', neighborhood_fun='gaussian', sigma=1, neighborhood_type='knn', k=10);

Parameters

m input matrix storing observations in rows or a similarity matrix. Must be numeric. When using a similarity matrix specify neighborhood_fun as 'none'. Otherwise choose another option.

nclus target number of clusters. It coresponds to the number of eigenvectors used.

laplacian type of laplacian used in the algorithm. Default = 'normalizedRW'. Possible options are:

'normalizedRW' - recommended in most cases - default

'normalizedSym'

'unnormalized' - usually not recommended

neighborhood_fun neighborhood function used to weight the similarity graph. Default = 'gaussian'. Currently possible options are:

'none' - don't compute a similarity matrix. Use when providing a similarity matrix as input

'gaussian' - a gaussian similarity function

'neg_euclid' - negative euclidean distances.

sigma sigma parameter for 'gaussian' neighborhood_fun. Affects the width of neighborhood. Default = 1

neighborhood_type type of neighborhood used. Default = 'knn'. Possible values:

'complete' - all vertices are connected

'knn' - vertices are connected only if one of them belongs to the k nearest neighbors of the other

'mutual_knn' - vertices are connected only if both of them belong to the k nearest neighbors of the other

k k parameter used by 'knn' and 'mutual_knn' neighborhood_types. Default = 10

Description

The function wraps the spccSpectralize function with a call to the FastClus SAS procedure, and serves as a builtin example uf the functionality.

Example

```
m = {0 1, 0 2, 0 3, 0 4};

out = spccFastClus(m, 2, 'normalizedRW', 'gaussian', 0.5,
'knn', 2);
print out;
```

References:

[1] A. Jain and M. Law, Data clustering: A user's dilemma. *Lecture Notes in Computer Science*, 2005. 3776: p. 1-10.

[2] Von Luxburg, U. (2007). “A Tutorial on Spectral Clustering.” *Statistics and Computing*, *17*(4).