motifcluster

Release 0.0.1

William George Underwood

Apr 26, 2020

CONTENTS:

Motif-Based Spectral Clustering of Weighted Directed Networks

CHAPTER

ONE

CLUSTERING METHODS

Functions for spectral clustering are in *motifcluster.clustering*.

cluster_spectrum(*spectrum*, *num_clusts*)

Get cluster assignments from spectrum using k-means++.

Get a list of cluster assignments from a spectrum, using k-means++ and *num_clusts* clusters.

Parameters

- **spectrum** (*dict*) – A dictionary containing *"vects"*: the matrix of eigenvectors to pass to k-means++.
- **num_clusts** (*int*) – The number of clusters to find.

Returns **cluster_assigns** – A list of integers from *1* to *num_clusts*, representing cluster assignments.

Return type list of int

run_motif_clustering(*adj_mat*, *motif_name*, *motif_type='struc'*, *mam_weight_type='unweighted'*, *mam_method='sparse'*, *num_eigs=2*, *type_lap='comb'*, *num_clusts=2*)

Run motif-based clustering.

Run motif-based clustering on the adjacency matrix of a (weighted directed) network, using a specified motif, motif type, weighting scheme, embedding dimension, number of clusters and Laplacian type.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix to be embedded.
- **motif_name** (*str*) – Motif used for the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to use. One of *"func"* or *"struc"*.
- **mam_weight_type** (*str*) – Weighting scheme for the motif adjacency matrix. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – The method to use for building the motif adjacency matrix. One of *"sparse"* or *"dense"*.
- **num_eigs** (*int*) – Number of eigenvalues and eigenvectors for the embedding.
- **type_lap** (*str*) – Type of Laplacian for the embedding. One of *"comb"* or *"rw"*.
- **num_clusts** (*int*) – The number of clusters to find.

Returns

- **adj_mat** (*matrix*) – The original adjacency matrix.
- **motif_adj_mat** (*matrix*) – The motif adjacency matrix.
- **comps** (*list*) – The indices of the largest connected component of the motif adjacency matrix.

- **adj_mat_comps** (*matrix*) – The original adjacency matrix restricted to the largest connected component of the motif adjacency matrix.
- **motif_adj_mat_comps** (*matrix*) – The motif adjacency matrix restricted to its largest connected component.
- **vals** (*list*) – A length-*num_eigs* list containing the eigenvalues associated with the Laplace embedding of the restricted motif adjacency matrix.
- **vects** (*matrix*) – A matrix containing the eigenvectors associated with the Laplace embedding of the restricted motif adjacency matrix.
- *clusts* – A vector containing integers representing the cluster assignment of each vertex.

Examples

```
>>> adj_mat = np.array(range(1, 10)).reshape((3, 3))
>>> run_motif_clustering(adj_mat, "M1", "func",
...   "mean", "sparse", 2, "rw", 2)
```

CHAPTER

TWO

ADJACENCY AND INDICATOR MATRICES

Functions for building adjacency and indicator matrices are in *motifcluster.indicators*.

`_build_G`(*adj_mat*)

Build sparse adjacency matrix.

Build the sparse adjacency matrix *G* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **G** – The adjacency matrix in sparse form.

Return type sparse matrix

`_build_Gd`(*adj_mat*)

Build double-edge adjacency matrix.

Build the sparse double-edge adjacency matrix *Gd* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Gd** – A double-edge adjacency matrix in sparse form.

Return type sparse matrix

`_build_Gp`(*adj_mat*)

Build product matrix.

Build the sparse product matrix *Gp* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Gp** – A product matrix in sparse form.

Return type sparse matrix

`_build_Gs`(*adj_mat*)

Build single-edge indicator matrix.

Build the sparse single-edge adjacency matrix *Gs* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Gs** – A single-edge adjacency matrix in sparse form.

Return type sparse matrix

`_build_Id`(*adj_mat*)

Build identity matrix.

Build the sparse identity matrix *Id* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Id** – An identity matrix in sparse form.

Return type sparse matrix

`_build_J`(*adj_mat*)

Build directed indicator matrix.

Build the sparse directed indicator matrix *J* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **J** – A directed indicator matrix in sparse form.

Return type sparse matrix

`_build_J0`(*adj_mat*)

Build missing-edge indicator matrix.

Build the missing-edge indicator matrix *J0* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **J0** – A missing-edge indicator matrix.

Return type sparse matrix

`_build_Jd`(*adj_mat*)

Build double-edge indicator matrix.

Build the sparse double-edge indicator matrix *Jd* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Jd** – A double-edge indicator matrix in sparse form.

Return type sparse matrix

`_build_Je`(*adj_mat*)

Build edge-and-diagonal matrix.

Build the sparse edge-and-diagonal matrix *Ie* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Ie** – An edge-and-diagonal matrix in sparse form.

Return type sparse matrix

`_build_Jn`(*adj_mat*)

Build vertex-distinct indicator matrix.

Build the vertex-distinct indicator matrix *Jn* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Jn** – A vertex-distinct indicator matrix.

Return type sparse matrix

`_build_Js`(*adj_mat*)

Build single-edge indicator matrix.

Build the sparse single-edge indicator matrix *Js* from a graph adjacency matrix.

Parameters **adj_mat** (*matrix*) – The original adjacency matrix.

Returns **Js** – A single-edge indicator matrix in sparse form.

Return type sparse matrix

CHAPTER

THREE

MOTIF ADJACENCY MATRICES

Functions for building motif adjacency matrices are in *motifcluster.motifadjacency*.

`build_motif_adjacency_matrix`(*adj_mat*, *motif_name*, *motif_type='struc'*, *mam_weight_type='unweighted'*, *mam_method='sparse'*)

Build a motif adjacency matrix.

Build a motif adjacency matrix from an adjacency matrix. Entry (*i, j*) of a motif adjacency matrix is the sum of the weights of all motifs containing both nodes *i* and *j*.

- The motif is specified by name and the type of motif instance can be one of:
  - Functional: motifs should appear as subgraphs.
  - Structural: motifs should appear as induced subgraphs.
- The weighting scheme can be one of:
  - Unweighted: the weight of any motif instance is one.
  - Mean: the weight of any motif instance is the mean of its edge weights.
  - Product: the weight of any motif instance is the product of its edge weights.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_name** (*str*) – Motif used for the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build. One of *"func"* or *"struc"*.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*. The sparse formulation avoids generating large dense matrices so tends to be faster for large sparse graphs.

Returns A motif adjacency matrix.

Return type sparse matrix

Examples

```
>>> adj_mat = np.array(range(1, 10)).reshape((3, 3))
>>> build_motif_adjacency_matrix(adj_mat, "M1", "func", "mean")
```

mam_M1 (*adj_mat*, *motif_type*, *mam_weight_type*)
Perform the motif adjacency matrix calculations for motif M1.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

mam_M10 (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif M10.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

mam_M11 (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif M11.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

mam_M12 (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif M12.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.

- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M13` (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)

Perform the motif adjacency matrix calculations for motif M13.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M2` (*adj_mat*, *motif_type*, *mam_weight_type*)

Perform the motif adjacency matrix calculations for motif M2.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M3` (*adj_mat*, *motif_type*, *mam_weight_type*)

Perform the motif adjacency matrix calculations for motif M3.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M4` (*adj_mat*, *mam_weight_type*)

Perform the motif adjacency matrix calculations for motif M4.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M5` (*adj_mat*, *motif_type*, *mam_weight_type*)
Perform the motif adjacency matrix calculations for motif M5.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M6` (*adj_mat*, *motif_type*, *mam_weight_type*)
Perform the motif adjacency matrix calculations for motif M6.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M7` (*adj_mat*, *motif_type*, *mam_weight_type*)
Perform the motif adjacency matrix calculations for motif M7.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M8` (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif M8.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_M9` (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif M9.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_Mcoll` (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif Mcoll.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_Md` (*adj_mat*, *mam_weight_type*)
Perform the motif adjacency matrix calculations for motif Md.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_Mexpa` (*adj_mat*, *motif_type*, *mam_weight_type*, *mam_method*)
Perform the motif adjacency matrix calculations for motif Mexpa.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.
- **mam_method** (*str*) – Which formulation to use. One of *"dense"* or *"sparse"*.

Returns A motif adjacency matrix.

Return type sparse matrix

`mam_Ms` (*adj_mat*, *motif_type*, *mam_weight_type*)

Perform the motif adjacency matrix calculations for motif Ms.

Parameters

- **adj_mat** (*matrix*) – Adjacency matrix from which to build the motif adjacency matrix.
- **motif_type** (*str*) – Type of motif adjacency matrix to build.
- **mam_weight_type** (*str*) – The weighting scheme to use. One of *"unweighted"*, *"mean"* or *"product"*.

Returns A motif adjacency matrix.

Return type sparse matrix

CHAPTER

FOUR

NETWORK SAMPLING

Functions for random sampling of weighted directed networks are in *motifcluster.sampling*.

demonstration_graph()

Generate a small graph for demonstrations.

Generate the sparse and dense adjacency matrices of a small weighted directed graph, for demonstrating methods and running tests.

Returns

- **adj_mat_dense** (*matrix*) – the adjacency matrix in dense form.
- **adj_mat_sparse** (*sparse matrix*) – the adjacency matrix in sparse form.

sample_bsbm(*source_block_sizes*, *dest_block_sizes*, *bipartite_connection_matrix*, *bipartite_weight_matrix=None*, *sample_weight_type='unweighted'*)

Sample a bipartite stochastic block model (BSBM).

Sample the (weighted) adjacency matrix of a (weighted) bipartite stochastic block model (BSBM) with specified parameters.

Parameters

- **source_block_sizes** (*list of int*) – A list containing the size of each block of source vertices.
- **dest_block_sizes** (*list of int*) – A list containing the size of each block of destination vertices.
- **bipartite_connection_matrix** (*matrix*) – A matrix containing the source block to destination block connection probabilities.
- **sample_weight_type** (*str*) – The type of weighting scheme. One of *"unweighted"*, *"constant"* or *"poisson"*.
- **weight_matrix** (*matrix*) – A matrix containing the source block to destination block weight parameters. Unused for *sample_weight_type = "constant"*. Defaults to *None*.

Returns **adj_mat** – A randomly sampled (weighted) adjacency matrix of a BSBM.

Return type sparse matrix

Examples

```
>>> source_block_sizes = [10, 10]
>>> dest_block_sizes = [10, 10, 10]
>>> bipartite_connection_matrix = np.array([0.8, 0.5, 0.1, 0.1, 0.5, 0.8]).
↪reshape((2, 3))
>>> bipartite_weight_matrix = np.array([20, 10, 2, 2, 10, 20]).reshape((2, 3))
>>> sample_bsbm(block_sizes, bipartite_connection_matrix,
...   bipartite_weight_matrix, "poisson")
```

sample_dsbm(*block_sizes*, *connection_matrix*, *weight_matrix=None*, *sample_weight_type='unweighted'*)
Sample a directed stochastic block model (DSBM).

Sample the (weighted) adjacency matrix of a (weighted) directed stochastic block model (DSBM) with specified parameters.

Parameters

- **block_sizes** (*list of int*) – A list containing the size of each block of vertices.
- **connection_matrix** (*matrix*) – A matrix containing the block-to-block connection probabilities.
- **sample_weight_type** (*str*) – The type of weighting scheme. One of *"unweighted"*, *"constant"* or *"poisson"*.
- **weight_matrix** (*matrix*) – A matrix containing the block-to-block weight parameters. Unused for *sample_weight_type* = *"constant"*. Defaults to *None*.

Returns **adj_mat** – A randomly sampled (weighted) adjacency matrix of a DSBM.

Return type sparse matrix

Examples

```
>>> block_sizes = [10, 10]
>>> connection_matrix = np.array([0.8, 0.1, 0.1, 0.8]).reshape((2, 2))
>>> weight_matrix = np.array([10, 3, 3, 10]).reshape((2, 2))
>>> sample_dsbm(block_sizes, connection_matrix, weight_matrix, "poisson")
```

CHAPTER

FIVE

UTILITY FUNCTIONS

Assorted utility functions for the motifcluster module are in *motifcluster.utils*.

`_a_b_one` (*a_mat*, *b_mat*)

Compute a right-multiplication with the ones matrix.

Compute *a * (b @ one_mat)* where *a*, *b*, *ones_mat* are square matrices of the same size, and *ones_mat* contains all entries equal to one. The product * is an entry-wise (Hadamard) product, while @ represents matrix multiplication. This method is more efficient than the naive approach when *a* or *b* are sparse.

Parameters **a, b** (*matrix*) – Square matrices of the same size.

Returns The sparse square matrix *a * (b @ one_mat)*.

Return type sparse matrix

`_a_one_b` (*a_mat*, *b_mat*)

Compute a left-multiplication with the ones matrix.

Compute *a * (one_mat @ b)* where *a*, *b*, *ones_mat* are square matrices of the same size, and *ones_mat* contains all entries equal to one. The product * is an entry-wise (Hadamard) product, while @ represents matrix multiplication. This method is more efficient than the naive approach when *a* or *b* are sparse.

Parameters **a, b** (*matrix*) – Square matrices of the same size.

Returns The sparse square matrix *a * (one_mat @ b)*.

Return type sparse matrix

`_drop0_killdiag` (*some_mat*)

Set diagonal entries to zero and sparsify.

Set the diagonal entries of a matrix to zero and convert it to sparse form.

Parameters **some_mat** (*matrix*) – A square matrix.

Returns **sparse_mat** – A sparse-form copy of *some_mat* with its diagonal entries set to zero.

Return type sparse matrix

`_random_sparse_matrix` (*m*, *n*, *p*, *sample_weight_type='constant'*, *w=1*)

Build a random sparse matrix.

Build a sparse matrix of size *m * n* with non-zero probability *p*. Edge weights can be unweighted, constant-weighted or Poisson-weighted.

Parameters

- **m, n** (*int*) – Dimension of matrix to build is *(m, n)*.
- **p** (*float*) – Probability that each entry is non-zero (before weighting).

- **sample_weight_type** (*str*) – Type of weighting scheme.
- **w** (*float*) – Weight parameter.

Returns A random sparse matrix.

Return type sparse matrix

get_largest_component(*adj_mat*)

Get largest connected component.

Get the indices of the vertices in the largest connected component of a graph from its adjacency matrix.

Parameters **adj_mat** (*matrix*) – An adjacency matrix of a graph.

Returns **verts_to_keep** – A list of indices corresponding to the vertices in the largest connected component.

Return type list

Examples

```
>>> adj_mat = np.array([0, 1, 0, 0, 0, 0, 0, 0, 0]).reshape((3, 3))
>>> get_largest_component(adj_mat)
```

get_motif_names()

Get common motif names.

Get the names of some common motifs as strings.

Returns **motif_names** – A list of names (strings) of common motifs.

Return type list

CHAPTER

SIX

INDEX

- genindex

PYTHON MODULE INDEX

INDEX