Scola Documentation

Release 0.0.0

Sadamori Kojaku

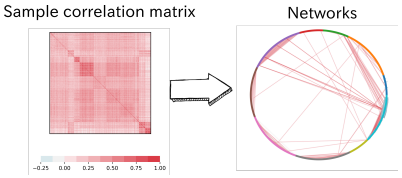
Dec 08, 2019

Table of contents:

The Scola is an algorithm that takes a correlation matrix or a precision matrix as input and outputs a network. In the generated network, edges between nodes indicate correlations that are not accounted for by some expected properties (e.g., noise independent for different variables or a global trend).

Please cite the paper if you use this package:

Sadamori Kojaku and Naoki Masuda. Proceedings of the Royal Society A, 475, 2231 (2019). [Preprint]



CHAPTER 1

Installing the Scola

`scola` supports both Python 2.x and 3.x and can be installed on Ubuntu, CentOS, macOS and Windows. We recommend using **pip** for installation.

1.1 Install with pip

```
pip install scola
```

1.2 Install from source

Download the source file from GitHub. Then, under /scola directory, type

```
python setup.py install
```

1.2.1 Dependencies

`scola` has dependencies on the following packages:

- NumPy
- SciPy
- tqdm

CHAPTER 2

Getting Started

We demonstrate how to generate a network with `scola`.

An example data, sample.txt, is available on GitHub, which is composed of L=300 rows and N=25 columns (space separated), where L is the number of samples and N is the number of variables (i.e., nodes). The following code loads the data.

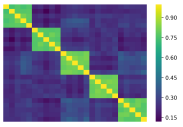
```
import numpy as np
import pandas as pd

X = pd.read_csv("https://raw.githubusercontent.com/skojaku/scola/master/data/sample.
↪txt", header=None, sep=" ").values
L = X.shape[0] # Number of samples
N = X.shape[1] # Number of nodes
```

Then, compute the sample correlation matrix by

```
C_samp = np.corrcoef(X.T) # NxN correlation matrix
```

`C_samp` looks like

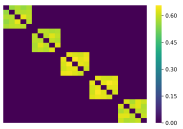


Finally, provide `C_samp` and `L` to estimate the network and associated null model:

```
import scola
W, C_null, selected_null_model, EBIC, construct_from, all_networks = scola.corr2net.
↪transform(C_samp, L)
```

`W` is the weighted adjacency matrix of the generated network, where W[i,j] indicates the weight of the edge between nodes i and j.

The `W` looks like

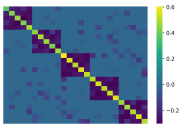


See the *scola package* for other return values.

Scola can construct a network from precision matrices, which is often different from that constructed from correlation matrices. To do this, give an extra parameter `construct_from='pres'`:

```
import scola
W, C_null, selected_null_model, EBIC, construct_from, all_networks = scola.corr2net.
↪transform(C_samp, L, construct_from="pres")
```

which produces a different network:



If one sets `construct_from='auto'`, the Scola constructs networks from correlation matrices and precision matrices. Then, it chooses the one that best represents the given data in terms of the extended BIC. The selected type of the matrix is indicated by `construct_from` in the return variables.

CHAPTER 3

Examples

CHAPTER 4

scola package

Scola package consists of two modules: `corr2net` module and `null_models` module.

The `corr2net` module contains functions to generate networks from correlation matrices. The `null_model` module contains functions to generate null correlation matrices.

4.1 Submodules

4.2 scola.corr2net module

transform(*C_samp*, *L*, *null_model='all'*, *disp=True*, *construct_from='corr'*, *beta=0.5*)

Generate a network from a correlation matrix using the Scola algorithm.

Parameters

- **C_samp** (*2D numpy.ndarray, shape (N, N)*) – Sample correlation matrix. *N* is the number of nodes.
- **L** (*int*) – Number of samples.
- **null_model** (*str or list of str or list of functions, default 'all'*) – Null model to be used for constructing the network. The following three null models are available:
  - White noise model (null_model='white-noise'),
  - the Hirschberger-Qi-Steuer model (null_model='hqs')
  - the configuration model (null_model='config').

  One can set multiple null models by a list, e.g., null_model = ["white-noise", "hqs", "config"] or equivalently null_model = "all". If multiple null models are given, the best one among the three null models in terms of the extended Bayesian information criterion (BIC) is selected.

To use other null models, one can set null_model = func or null_model = [func1, func2,...], where func is a function taking the sample correlation matrix as the input and outputs the null correlation matrix (2D numpy.ndarray, shape(N,N)), the number of parameters for the null model (int), and the name of the null model (str).

- **disp** (*bool, default True*) – Set disp=True to display the progress of computation. Otherwise, set disp=False.
- **construct_from** (*str, default 'corr'*) – Type of matrix to construct a network. Setting "corr" constructs based on the correlation matrix. Setting "pres" constructs based on the precision matrix. If construct_from='auto', the Scola constructs networks from the correlation matrix and precision matrix. Then, it chooses the best one in terms of the extended BIC.
- **beta** (*float, default 0.5*) – Hyperparameter for the extended BIC. When beta = 0, the EBIC is equivalent to the BIC. The higher value yields a sparser network. Range [0,1].

Returns

- **W** (*2D numpy.ndarray, shape (N, N)*) – Weighted adjacency matrix of the generated network.
- **C_null** (*2D numpy.ndarray, shape (N, N)*) – Estimated null correlation matrix used for constructing the network.
- **selected_null_model** (*str*) – The null model selected by the Scola.
- **EBIC** (*float*) – The extended BIC value for the generated network.
- **construct_from** (*str*) – construct_from='corr' or construct_from='pres' indicates that the network is constructed from the correlation matrix or the precision matrix, respectively.
- **all_networks** (*list of dict*) – Results of all generated networks. Each dict object in the list consists of 'W', 'C_null', 'null_model', 'EBIC_min', 'construct_from' and 'W_list'. 'W_list' is a list of dict objects, in which each dict consists of a network (i.e., 'W') and its EBIC value (i.e., 'EBIC') found by the golden section search algorithm.

Example:: import scola W, C_null, selected_null_model, EBIC, construct_from, all_networks = scola.corr2net.transform(C_samp, L)

4.3 scola.null_models module

white_noise_model(*C_samp*)

Compute the white noise model for correlation matrices.

Parameters **C_samp** (*2D numpy.ndarray, shape (N, N)*) – Sample correlation matrix.

Returns

- **C_null** (*2D numpy.ndarray, shape (N, N)*) – The correlation matrix under the white-noise model.
- **K_null** (*int*) – Number of parameters to generate the null correlation matrix
- **name** (*str*) – Name of the null model ("white-noise")

hqs_model(*C_samp*)

Compute the HQS model for correlation matrices.

Parameters **C_samp** (*2D numpy.ndarray, shape (N, N)*) – Sample correlation matrix.

Returns

- **C_null** (*2D numpy.ndarray, shape (N, N)*) – The correlation matrix under the HQS model.
- **K_null** (*int*) – Number of parameters to generate the null correlation matrix
- **name** (*str*) – Name of the null model ("hqs")

configuration_model(*C_samp*, *tolerance=0.005*)

Compute the configuration model for correlation matrices using the gradient descent algorithm.

Parameters

- **C_samp** (*2D numpy.ndarray, shape (N, N)*) – Sample correlation matrix.
- **tolerance** (*float*) – Tolerance in relative error.

Returns

- **C_null** (*2D numpy.ndarray, shape (N, N)*) – The correlation matrix under the config model.
- **K_null** (*int*) – Number of parameters to generate the null correlation matrix
- **name** (*str*) – Name of the null model ("config")

CHAPTER 5

Licence

GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007

Python Module Index

S

Index