SCPLearn: Software Manual

Harini Eavani

Harini.Eavani@uphs.upenn.edu

cbica-software@uphs.upenn.edu

March 28, 2016

1 Introduction

This software is used to calculate Sparse Connectivity Patterns (SCPs) from resting state fMRI data. SCPs consist of those regions whose connectivity co-varies across subjects. This algorithm was developed as a complementary approach to existing network identification methods. SCPLearn has the following advantages:

1. Does not require thresholding of correlation matrices

2. Allows for both positive and negative correlations

3. Does not constrain the SCPs to have spatial/temporal orthogonality/independence

4. Provides group-common SCPs and subject-specific measures of average correlation within each SCP

2 Method

The objective of our method is to find SCPs consisting of functionally synchronous regions, and are smaller than the whole-brain network. The information content within any one of these SCPs is also relatively low, since all the nodes within an SCP are correlated, and express the same information. Hence, if a correlation matrix were constructed for each of these SCPs, it

would show two properties - (1) large number of edges with zero weights, or sparsity and (2) low information content - or rank deficiency.

The input to our method is size $P \times P$ correlation matrices $\mathbf{\Sigma}_n \succeq 0$, one for each subject n, $n = 1, 2, \ldots, N$. We would like to find SCPs common to all the subjects, such that a non-negative combination of these SCPs generates the full-correlation matrix $\mathbf{\Sigma}_n$, for each subject n. Each of these K SCPs can be represented by a vector of node-weights $\mathbf{b}_k$, where $-1 \preceq \mathbf{b}_k \preceq 1$, $\mathbf{b}_k \in \mathbf{R}^P$. Each vector $\mathbf{b}_k$ reflects the membership of the nodes to the sub-network k. If $|\mathbf{b}_k(i)| > 0$, node i belongs to the sub-network k, and if $\mathbf{b}_k(i) = 0$ it does not. If two nodes in $\mathbf{b}_k$ have the same sign, then they are positively correlated and opposing sign reflects anti-correlation. Thus, the rank-one matrix $\mathbf{b}_k \mathbf{b}_k^T$ reflects the correlation behavior of SCP k. In addition, we constrain these SCPs to be much smaller than the whole-brain network by restricting the l_1-norm of $\mathbf{b}_k$ to not exceed a constant value λ.

We would like to approximate each matrix $\{\mathbf{\Sigma}_n\}_{n=1}^N$ by a non-negative combination of SCPs $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_K]$. Thus, we want

$$\mathbf{\Sigma}_n \approx \sum_{k=1}^K c_n(k) \mathbf{b}_k \mathbf{b}_k^T + \mathrm{diag}(\mathbf{e}_n) = \mathbf{B} \, \mathrm{diag}(\mathbf{c}_n) \, \mathbf{B}^T + \mathrm{diag}(\mathbf{e}_n) \triangleq \hat{\mathbf{\Sigma}}_n$$

$$||\mathbf{b}_k||_1 \leq \lambda, \quad -1 \leq \mathbf{b}_k(i) \leq 1, \quad \mathbf{c}_n \geq 0, \quad \mathbf{e}_n \geq 0 \qquad (1)$$

where $\mathrm{diag}(\mathbf{c}_n)$ denotes a diagonal matrix with values $\mathbf{c}_n \in \mathbf{R}_+^K$ along the diagonal. The diagonal matrix $\mathrm{diag}(\mathbf{e}_n)$ is added to each approximation to make it full-rank. Thus, each subject n is associated with a vector of K subject-specific measures $\mathbf{c}_n$ which are non-negative and reflect the relative contribution of each SCP to the whole-brain functional network in the n-th subject.

We quantify the approximation between $\mathbf{\Sigma}_n$ and $\hat{\mathbf{\Sigma}}_n$ using the frobenius norm. Note that there is an ambiguity in amplitude between the two factors - if $\mathbf{b}_k$ and $c_n(k)$ is a solution, $\alpha \mathbf{b}_k$ and $c_n(k)/\alpha^2$ is also a solution for any positive scalar α. To prevent this, we fix the maximum value in each SCP to unity; i.e., $\max_i |\mathbf{b}_k(i)| = 1$.

Bringing the objective and the constraints together, we have the following optimization problem w.r.t the unknowns $\mathbf{B}$, $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n]$ and $\mathbf{E} =$

$[\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n]$:

$$\underset{\mathbf{B},\mathbf{C}}{\text{minimize}} \ \sum_{n=1}^{N} \left|\left| \boldsymbol{\Sigma}_n - \hat{\boldsymbol{\Sigma}}_n \right|\right|_F^2$$

subject to

$$\begin{aligned}
||\mathbf{b}_k||_1 &\leq \lambda, \quad k = 1, \ldots, K, \\
-1 \leq \mathbf{b}_k(i) &\leq 1, \quad \max_i |\mathbf{b}_k(i)| = 1, \quad i = 1, \ldots, P \\
\mathbf{c}_n &\geq 0, \qquad n = 1, \ldots, N
\end{aligned}$$

(2)

The objective function in the proposed model is non-convex w.r.t both unknown variables $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{E}$. We use the method of alternating minimization to solve for the three unknowns. At each iteration a local minimum is obtained using projected gradient descent. Such a procedure converges to a local minimum. The variable $\mathbf{B}$ is initialized using k-means run on the time-points; $\mathbf{C}$ and $\mathbf{E}$ are initialized to randomly chosen values.

Hierarchical Framework This method can be applied to the data in a hierarchical fashion; first, "primary SCPs" are computed by running the software on the correlation matrices $\boldsymbol{\Sigma}_n$. Then, for each primary SCP, multiple smaller "secondary SCPs" can be found by re-applying the method only for those regions that belong to the primary SCP.

Sample weights This software also allows subject-specific weights as input - e.g., in cases when subjects have different numbers of time-points. In this case, the optimization objective is modified to be $\sum_{n=1}^{N} \alpha_n \left|\left| \boldsymbol{\Sigma}_n - \hat{\boldsymbol{\Sigma}}_n \right|\right|_F^2$, where α_n is the subject-specific weight.

3 Software

3.1 Within main directory

1. Detailed documentation is contained in the latex file UserManual.tex and the pdf UserManual.pdf produced from it.

2. Author information is in AUTHORS.txt

3. All information needed to get started with the software is in README.md

4. The file m2cpp.pl is a helper perl script for doxygen. It is needed to pick up comments from MATLAB script files.

5. The file INSTALL.txt has installation and usage information.

6. Doxyfile is the configuration file for generating doxygen documentation.

7. ignore.txt contains a list of files that doxygen ignores from parsing.

8. The folder docs/ will contain all the documentation once doxygen is run using Doxyfile.

9. The folder src/ contains source code for SCPLearn.

10. The folder licenses/ contains license information related to this software.

3.2 Within `src/` directory

This software is implemeted primarily in MATLAB. For command-line argument parsing, and nifti I/O operations, python code is used.

For ease of use, the software takes as input either rsfMRI voxel time-series data in NIFTI format, or ROI time-series data in mat format. The list of nifti files is provided as a text file.

It also takes as input the node definitions as an atlas in NIFTI space. This atlas must have the ROI regions numbered $1, 2, 3, \ldots$ and must be in the same space as the subject data.

1. Main functions are scplearn and scptest. These functions parses the input arguments and calls "ComputeROIAverages.py".

2. ComputeROIAverages.py reads the rsfMRI data and the node definitions. It extracts average time-series of each ROI for each subject and saves it in ".mat" files.

3. These mat files are loaded into MATLAB using the function SCPLearn-FromMatFiles.m . Then, based on the input parameters, either (1) SCPLearn.m is called, which computes only primary SCPs, or (2) SCPLearn_two_level.m is called which computes both primary and secondary level SCPs or (3) SCPTest.m is called, which computes only SCP coefficients.

4. Both the above functions call BlockSolverLowRankFrob.m, which implements the alternate minimization strategy discussed in the previous section. This function calls (1) BSolver_lowrank_frob.m (2) CSolver_lowrank_frob.m (3) ESolver_lowrank_frob.m until termination criteria are met.

5. BSolver_lowrank_frob.m calls two functions (1) spg_04262015.m which implements projected gradient descent. (2) ProjectionOnUnitBoxSimplex.m implements the projection function for the sparse basis.

6. After the MATLAB code has finished running, scplearn calls replaceLabels_nib.py. This python code replaces the labels in the atlas with the values of the nodes in each SCP. In this manner, for each SCP, a nifti image is written to disk.

7. The set of python functions in SCPUtils.py are used for error checking, handling exceptions, file check, etc

8. The MATLAB functions SCPLearn_SplitSample.m, CompareSCPs.m,Hungarian.m implement split-sample SCP reproducibility and error calculation, as described and used in the main paper.

9. The python script make calls mcc to build SCPLearnFromMatFiles.m as an mcc-executible. It then copies all python files to the install directory.

4   Installation

4.1   Dependencies

This software has been primarily implemented for Linux operating systems.

- MATLAB Compiler mcc version 5.2 (R2014B)

- MATLAB R2014B

- Python 2.7.9

- Python library numpy 1.7.2

- Python library scipy 0.15.1

- Python library pandas 0.16.2

- Python library nibabel 2.0.1

Make sure all dependencies are met before proceeding with install.

4.2   Generating standalone executables

1. Within the `src/` directory, run `./make`. This runs mcc, and generates the executible for `SCPLearnFromMatFiles.m`

2. Within the `src/` directory, run `./make test`. This runs scplearn on synthetic test data, and checks if the results match with ground truth. This command should return three SCPs, saved as nifti files in `src/test/test_2d_SCP_1,2,3.` along with other output. By design, three patterns were used as 'ground-truth' SCPs to generate the synthetic data; a cross, a circle and a square. Was SCPLearn able to clearly separate these three patterns in the output? You can compare your result to the results in `src/test/Test_2d_all.png`

   If testing succeeded, you will see the message:

   `Match between results and ground truth xx.xxxx>70% - test passed!",`
   `where xx.xx is the percentage match between result and ground-truth.`
   Proceed to install only if testing is completed.

3. Within the `src/` directory, run `./make install` with the location of the install directory as the argument:

   `./make install <installDir>`

4. Add the install directory to your path by running the following command.

   Replace `$installDir` with the location of the install directory from step 1 above.

   `export PATH=$PATH:$installDir`

4.3   Generating documentation

The above make install command creates doxygen documentation in "docs/*".
Please open "docs/index.html" in you browser.

5   Usage

This software provides two executibles for users - scplearn and scptest.

scplearn takes a set of data as input(either NIFTI data or mat files), and generates both SCPs as well as SCP coeffcients for that data. SCPs are common to the whole group, SCP coefficients are subject specific.

```
scplearn--
    Generates Sparse Connectivity Patterns from resting state fMRI data

    Usage: scplearn [OPTIONS]

    Required Options:
    [-d --data]    Specify the text file with list of nifti files (required)
                    *** OR ***
                   Specify the text file with list of mat files (required)
                   Each mat file must contain a variable named 'ts'
                   'ts' must be a matrix of time-series,
                   size (# of ROIs X # of timepoints)

    [-m --mask]    Specify the nifti ROI/parcel/atlas file (required)
    [-p --prefix]  Specify the prefix of the output file  (required)

    Options:
    [-t --type]     Specify the data type of input with either "matlab" or "nii".
    [-s --sparsity]  Specify the sparsity constraint as positive value.
      Default = nROIs/10.
    [-n --numberOfSCPs]   Specify number of primary SCPs.
     Default = 10.
    [-r --pruning]   Specify the pruning threshold as a value between [0,1].
     Default = 0.7.
    [-l --levels]   Specify number of secondary SCPs.
     Default = 50.

    [-o --outputDir]      The output directory to write the results.
     Defaults to the location of the input file
    [-w --workingDir]      Specify a working directory.
     By default a tmp dir is created and used
```

```
[-u --usage | -h --help]    Display this message
[-v --verbose]              Verbose output
[-V --Version]              Display version information

Examples:
scplearn -d list_of_nifti_files.txt -t nii -m Grasp_level5.nii \
 -p SCP_results -n 10 -s 50 -o /sbia/sbiaprj/BLSA -v
scplearn -d list_of_mat_files.txt -m Grasp_level5.nii \
 -p SCP_results -n 10 -r 0.5 -o /sbia/sbiaprj/BLSA -v
scplearn -d list_of_mat_files.txt -t matlab -m Grasp_level5.nii \
 -p SCP_results -l -n 10 -o /sbia/sbiaprj/BLSA -v

Example list_of_nifti_files.txt:
ProjName_subj_165464.nii.gz
ProjName_subj_26464.nii.gz
ProjName_subj_1054.nii.gz......
```

scplearn takes as input a text file with the list of nifti/mat files. If input
is nifti, each nifti file must be pre-processed rsfMRI 4D scan in a common
template/standard space. If input is mat files, each mat file must contain a
variable named 'ts'. 'ts' must be a matrix of time-series, size (# of ROIs X #
of timepoints) It also requires the "node" definitions (e.g. atlas, parcellation,
ROI definition, etc) to be input as a nifti file. The node definitions need to
be in the same template space as the other nifti files.

Mandatory arguments are:

1. List of nifti/mat files

2. NIFTI mask/atlas/ROI file

3. Prefix for all output files

Optional arguments are:

1. Number of SCPs at the primary level. Default = 10.

2. Sparsity level of SCPs. Default = Number of ROIs / 10.

3. Pruning threshold. Default = 0.7. SCPs with inner-product overlap
   greater than this threshold are discarded

4. Number of SCPs at the secondary level in the hierarchical decomposition. Default=50.

 In the output directory, the software returns:

1. One NIFTI file for each SCP that is generated, with file name ¡prefix¿_SCP_#.nii.gz

2. A <prefix>_SCP_Coeffs.csv file with the SCP coefficients for all the subjects, indexed by the NIFTI/mat filename that was input

3. A <prefix>_SCP_basis.csv file with the SCP basis, in csv format

4. A <prefix>_SCPs.mat file with all the outputs in MATLAB '.mat' format

5. A <prefix>_ts.mat file with the average time-series from all the subjects in MATLAB '.mat' format

6. A <subject>_<atlas>.mat file with the average time-series for each subject

 scptest takes a set of data AND SCPs as input, and generates on SCP coefficients for the data, corresponding to the SCPs that were input. For example, scplearn may be used when generating SCPs for the first time within a study. For subsequent participants that are acquired and processed, there is no need to re-compute SCPs; one can use existing SCPs as the reference and generate only SCP coefficients for the subsequent participants.

```
scptest--
Computes SCP coefficients for input data given input SCP basis

Usage: scptest [OPTIONS]

Required Options:
[-d --data]    Specify the text file with list of nifti files (required)
                *** OR ***
               Specify the text file with list of mat files (required)
               Each mat file must contain a variable named 'ts'
               'ts' must be a matrix of time-series,
                size (# of ROIs X # of timepoints)
```

```
[-m --mask]     Specify the nifti ROI/parcel/atlas file (required)
[-b --basis]     Specify the input SCP basis,
 saved as variable 'B' in mat file (required)
[-p --prefix]      Specify the prefix of the output file  (required)

Options:
[-o --outputDir]       The output directory to write the results.
 Defaults to the location of the input file
[-w --workingDir]      Specify a working directory.
 By default a tmp dir is created and used
[-u --usage | -h --help]    Display this message
[-v --verbose]              Verbose output
[-V --Version]              Display version information

Examples:
scptest -d list_of_nifti_files.txt -b BLSA_SCP_Basis.mat -m Grasp_level5.nii \
 -p SCP_test_coeffs -o /sbia/sbiaprj/BLSA -v
scptest -d list_of_mat_files.txt -b BLSA_SCP_Basis.mat -m Grasp_level5.nii \
 -p SCP_test_coeffs -o /sbia/sbiaprj/BLSA -v

Example list_of_nifti_files.txt:
ProjName_subj_165464.nii.gz
ProjName_subj_26464.nii.gz
ProjName_subj_1054.nii.gz......
```

scptest takes as input a text file with the list of nifti/mat files, NIFTI "node"
file as well as the SCPs stored in "mat" file format.

Mandatory arguments are:

1. List of nifti/mat files

2. NIFTI mask/atlas/ROI file

3. SCPs saved in mat file format (generated by a prior run of scplearn)

4. Prefix for all output files

In the output directory, the software returns:

1. A <prefix>_SCP_Coeffs.csv file with the SCP coefficients for all the
   subjects, indexed by the NIFTI/mat filename that was input

2. A <prefix>_SCPs.mat file with all the outputs in MATLAB '.mat' format

3. A <prefix>_ts.mat file with the average time-series from all the subjects in MATLAB '.mat' format

4. A <subject>_<atlas>.mat file with the average time-series for each subject

6   Citation

If you find this code useful, please cite:
```
Eavani, H., Satterthwaite, T. D., Filipovych, R., Gur, R. E., Gur,
R. C., Davatzikos, C. (2015).  Identifying Sparse Connectivity Patterns
in the brain using resting-state fMRI. Neuroimage, 105, 286-299.
```