Manual

Brain Network Construction and Classification Toolbox

BrainNetClass (Version 1.1)

(Updated date: 08-12-2019)

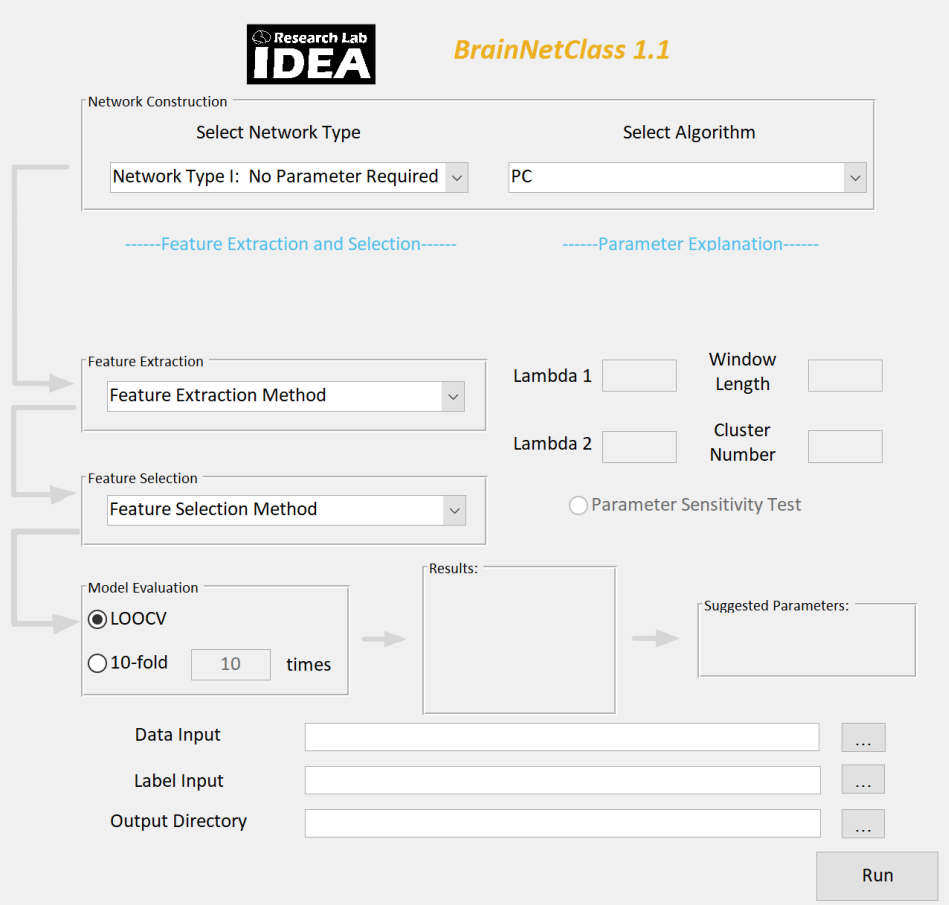


Table of Contents

1. Overview

Brain functional connectivity networks derived from resting-state functional MRI has become an important and popular technique to understand normative and altered brain functions. Machine learning on the brain functional networks for individualized classification, prediction, or diagnosis is booming in recent years. On the pressing demand by the neuroscientists and clinical researchers who would like to construct the brain functional networks for classification but with limited knowledge of machine learning or coding, we developed *BrainNetClass (v1.1)*.

The aim of *BrainNetClass* is to make it easier for neuroscientists, clinicians, and researchers from other fields conduct state-of-the-art brain network construction and rigorous machine learning-based classifications in a hassle-free, automatic, and interpretable way. It also helps to facilitate clinical applications of neuroimaging-based machine learning. It is hoped that this toolbox could be of help in standardization the methodology and boost reproducibility, generalizability, and interpretability of the network-based classification.

This toolbox is developed by Zhen Zhou, Xiaobo Chen, Yu Zhang, Han Zhang, and Dinggang Shen. The brain network construction algorithms were contributed by Lishan Qiao, Renping Yu, Xiaobo Chen, Yu Zhang, and Han Zhang. This work was supported in part by NIH grants EB022880, AG049371, AG042599, and AG041721.

For any issues and suggestions, please contact Zhen Zhou (zzstefan@email.unc.edu) and Han Zhang (hanzhang@med.unc.edu) at UNC-CH. If writing papers using our toolbox, please cite the following toolbox article. It is also recommended to cite the corresponding methodological paper(s).

Please always cite the toolbox article:
Zhou, Z., Chen, X., Zhang, Y., Qiao, L., Yu, R., Pan, G., Zhang, H., Shen, D., 2019. Brain network construction and classification toolbox (BrainNetClass). arXiv:1906.09908.

If using dHOFC, please also cite: **[1]** Chen, X., Zhang, H., Gao, Y., Wee, C.Y., Li, G., Shen, D., Alzheimer's Disease Neuroimaging, I., 2016. High-order resting-state functional connectivity network for MCI classification. Hum Brain Mapp 37, 3282-3296. **[2]** Zhang, H., Chen, X., Zhang, Y., Shen, D., 2017. Test-retest reliability of "high-order" functional connectivity in young healthy adults. Frontiers in Neuroscience, 11:439. **[3]** Chen, X., Zhang, H., Shen, D., 2017. Hierarchical High-Order Functional Connectivity Networks and Selective Feature Fusion for MCI Classification. Neuroinformatics, 15(3):271-284.

If using tHOFC, please also cite: **[1]** Zhang, H., Chen, X., Shi, F., Li, G., Kim, M., Giannakopoulos, P., Haller, S., Shen, D., 2016. Topographic Information based High-Order Functional Connectivity and its Application in Abnormality Detection for Mild Cognitive Impairment, Journal of Alzheimer's Disease, 54(3): 1095-1112. **[2]** Zhang, H., Chen, X., Zhang, Y., Shen, D., 2017. Test-retest reliability of "high-order" functional connectivity in young healthy adults. Frontiers in Neuroscience, 11:439.

If using aHOFC, please also cite: **[1]** Zhang, Y., Zhang, H., Chen, X., Lee, S.-W., Shen, D., 2017. Hybrid High-order Functional Connectivity Networks Using Resting-state Functional MRI for Mild Cognitive Impairment Diagnosis, Scientific Reports, 7: 6530. **[2]** Zhang, H., Chen, X., Zhang, Y., Shen, D., 2017. Test-retest reliability of "high-order" functional connectivity in young healthy adults. Frontiers in Neuroscience, 11:439.

If using LSR, please also cite: **[1]** Qiao, L., Zhang, H., Kim, M., Teng, S., Zhang, L., Shen, D., 2016. Estimating functional brain networks by incorporating a modularity prior. NeuroImage 141, 399-407.

If using SSGSR, please also cite: **[1]** Zhang, Y., Zhang, H., Chen, X., Liu, M., Zhu, X., Lee, S.-W., Shen, D., 2019. Strength and Similarity Guided Group-level Brain Functional Network Construction for MCI Diagnosis. Pattern Recognition, 88, 421-430.

If using SGR, WSR or WSGR, please also cite: **[1]** Yu, R., Zhang, H., An, L., Chen, X., Wei, Z., Shen, D., 2017. Connectivity strength-weighted sparse group representation-based brain network construction for MCI classification. Human Brain Mapping, 38(5): 2370-2383.

If using GSR, please also cite: **[1]** Wee, C.Y., Yap, P.T., Zhang, D., Wang, L., Shen, D., 2014. Group-constrained sparse fMRI connectivity modeling for mild cognitive impairment identification. Brain Struct Funct 219, 641-656.

2. Installation

First, download BrainNetClass from the Github site:
https://github.com/zzstefan/BrainNetClass
Together with the toolbox are the manual and exemplary data sets.

The recommended environment for running BrainNetClass is:
MATLAB version 2016b and higher, running on Windows 10 and Ubuntu 16.04 (the two platforms have been tested successfully but other platforms may also work well).

Warning: If running on a lower version MATLAB, there could be compatibility errors.

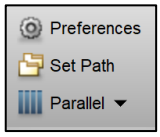
The installation of BrainNetClass is similar to setting up other MATLAB toolboxes. Simply download and unzip the package and add its main folder by using *addpath* to the MATLAB working path. There are two options to add path:

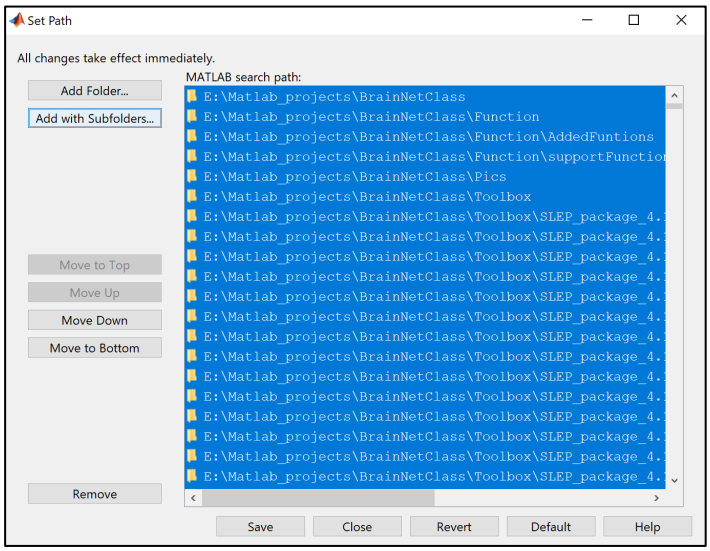
- Command line

Type the following command line in the MATLAB command window:
>> addpath(genpath('D:\BrainNetClass'));
Where the 'D: \BrainNetClass' is the exemplary path of BrainNetClass on your computer.

- Through Matlab Menu

Click 'Set Path' on the MATLAB panel, or type 'pathtool' in the MATLAB command window.
Click 'Add with Subfolders…' button, and select path, e.g., 'D:\BrainNetClass'.





Click 'Save' to save your change. If you do not have permission to save your changes on your computer (e.g., on the server), please save pathdef.m to another location where you will often launch MATLAB.

Warning: Make sure the BrainNetClass path DOES NOT include any space or special character.

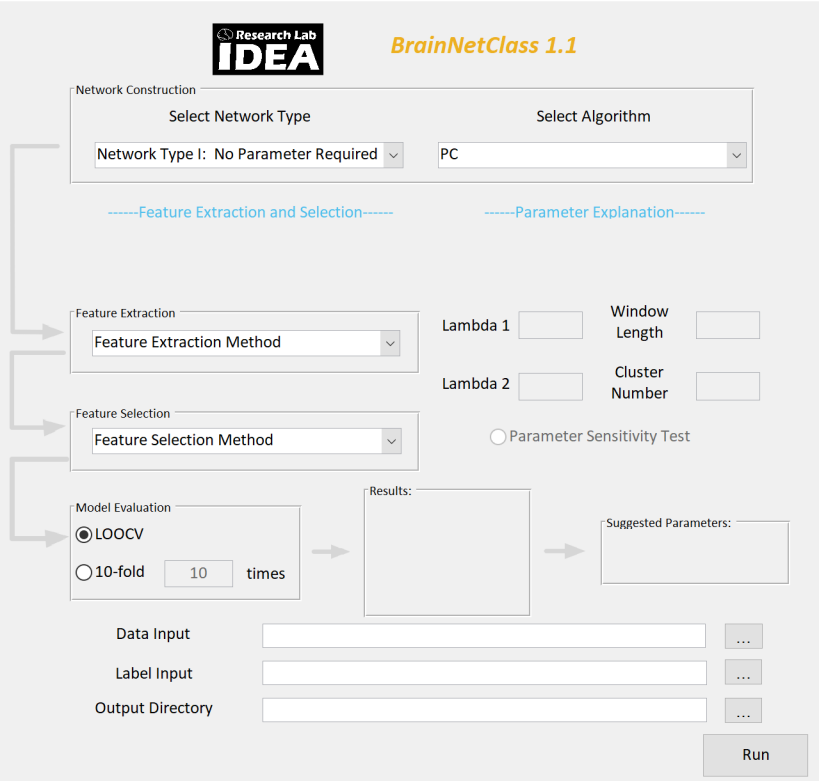
Note: BrainNetClass uses libsvm-3.23 and SLEP-4.1 toolboxes. To run the toolbox, one sometimes needs a compiled *libsvm* library and a compiled *SLEP* library. Although the compiled version of them are included in the toolbox, we highly recommend user make a compiled version by themselves if you meet any error regarding library compiling.

To compile libsvm, after adding path, please type the following in the command window:
>> cd 'D:\BrainNetClass';
>> cd 'toolbox/libsvm-3.23';
>> mex -setup

To compile SLEP:
>> cd 'D:\BrainNetClass';
>> cd 'toolbox/SLEP_package_4.1';
>> mex -setup

Note: Compiling of these two packages may require compilers, please refer to https://www.mathworks.com/support/requirements/supported-compilers.html.

After installation, to run the toolbox, type 'BrainNetClass' in the MATLAB command window. The following GUI window will pop-up.



3. Quick walk-through

1. Specify the RS-fMRI time series data of all subjects by selecting the folder containing all the text files (each subject's time series data is stored in each text file, the data is arranged as a matrix of Time × Node). Also, specify a text-formatted label file containing a column of labels for all subjects (e.g., -1 for patient and 1 for control) in the same order as what the Matlab takes when reading these time series data. The output directory should be also specified.

2. Choose the network construction method by first selecting the type of brain construction methods and then the specific method. When choosing a parameter required brain network construction method, the user needs to specify the parameter range(s) if they do not want to use the default settings. There are brief explanations of the meanings of the parameters on the panel above parameter settings for users to check.

3. Select or use predefined feature extraction and feature selection methods. There are also explanations on the panel above for users to check.

4. Choose model evaluation or cross-validation method. If choosing 10-fold cross-validation, users might also want to specify how many times the 10-fold cross validation will be repeated.

5. Click the 'Run' button and wait for all the processes complete, as an 'All Jobs Completed' window will pop out. The performance evaluation results will be printed out on the result panel and the suggested parameters will show up on the left (if applicable).

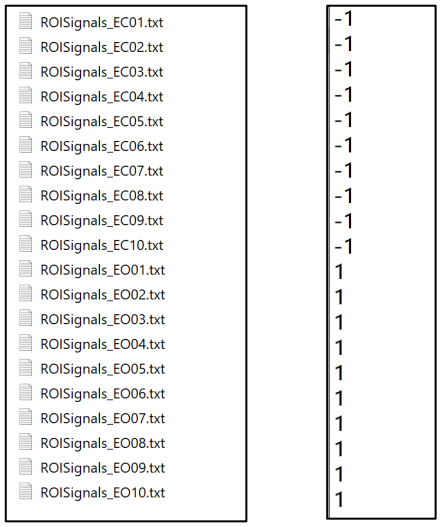
6. A full log of model configuration and classification results is also generated in the result folder.

7. The users may also want to test the performance of other baseline methods, such as PC and SR, to compare with the state-of-the-art method by repeating the above steps.

4. Input file preparation

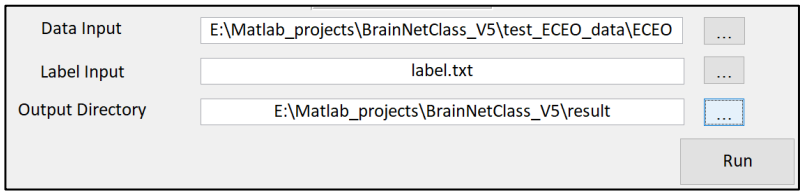
First, the user should specify the input folder that contains ALL of the input data files from ALL subjects. There are time series files and a label file, both of which should be in the *.txt file format.

The format of the regional time series file (e.g., 001.txt, 002.txt …) should be the same as that of the provided exemplary data (see the left panel of the figure below). For each subject, there should be one *.txt or *.csv file, which must be a matrix with rows denoting the time points and columns representing the ROIs. Each subject's data is in each file. The time series data can be obtained by the DPABI toolbox (http://rfmri.org/dpabi), REST toolbox (http://restfmri.net/forum/REST_V1.8), DPARSFA toolbox (http://rfmri.org/dpabi), Brant (https://sphinx-doc-brant.readthedocs.io/en/latest/) or other software. For DPABI/REST/DPARSFA, using 'extracting ROI time series' function to obtain region-averaged BOLD signals from the preprocessed fMRI data. For fMRI preprocessing, please refer to these toolboxes' manuals. The user should put all this extracted time series files into a single folder without any other files and set this folder as the input directory. The label text file should be prepared like the following figure (right panel), also in a *.txt or *.csv file, with each label located in one row for each subject in an order that is the same as the order of the time series data (please further check the order with MATLAB; usually it should follow alphabetic order). One can use -1 to represent patient and 1 to represent control.



Warning: Make sure the order of the two inputs is EXACTLY matched. If not sure, please check the 'Current Folder' window in MATLAB to get the idea of the ordering information for the time series files. To avoid confusion, it is recommended to use the following naming convention: ROISignals_sub0001.txt, ROISignals_sub0002.txt, …, because sometimes different operation systems will follow different order. The label should be in the same order and should be only -1 and 1.

The output directory should also be specified at the beginning. The user needs to create an empty folder and use it to store all the future generated results. Remember every time you run the toolbox, all the old result files in the result folder will be replaced by new result files. We suggest the user store the saved results elsewhere when a new running process will be start over. After all the inputs being specified, the bottom of the GUI should be filled like this:



5. Brain network construction

After specifying inputs, the user should choose a brain construction method to build brain functional networks for all subjects. The available methods can be categorized into two types: those with No Parameter Required (Network Type I), e.g., PC, aHOFC and tHOFC, and those with Parameter Required (Network Type II), e.g., SR, GSR, WSR, WSGR, SGR, SSGSR, SLR, and dHOFC. For explanation of each method, see below as a brief introduction and please refer to the mentioned original methodological papers for details. Feel free to cite relevant papers if they helped your research. Basically, Type I methods are simpler than those of Type II and could be more robust.

Tips: It is recommended that users select only one method that is assumed most appropriate for their own study. In addition to the main method, the PC and SR can be used as baseline methods. Please avoid blind selection of method and avoid testing all methods and only reporting the one with the best result, because it violates the rule of machine learning, as the optimized model is determined based on the testing data, which is considered double-dipping. If the sample size is large (e.g., > 200), better choose a Type I method, as parameter optimization could be time-consuming. If physical memory is low (e.g., < 16GB), better choose Type I method. If the data is noisy, Type II method is better, as they have abilities to reducing noise. For more method selection strategies, please see the toolbox paper and the following Tip Box.

Below is a brief explanation of all the available brain network construction methods. For more details, please see the toolbox paper and their respective original papers listed behind.

1. PC: Pearson's correlation. The most conventional method. It can be used as a baseline method. No parameter is required.
2. SR: Sparse representation with an L1-norm constraint. One parameter is required. When you expect the network is sparse or there is heavy noise in the data, you may use it. It can also be used as a baseline method.
3. GSR: Group sparse representation. It makes sure that all subjects have similar FC network pattern. One parameter is required. (Wee et al., 2014)
4. SSGSR: Generate within-group similar networks but retain necessary between-group differences. Two parameters are required. (Zhang et al., 2019)
5. SGR: Sparse group representation. Combining L1-norm and Lq,1-norm to preserve certain structured information in the adjacency matrix. Two parameters are required. (Yu et al., 2017)
6. WSR: FC-weighted SR. SR-based network construction but with strong FCs preserved, better preserving the original PC network structure while suppressing noise. One parameter is required. (Yu et al., 2017)
7. WSGR: Like SGR but with strong FC preserved by combining SGR and WSR. Two parameters are required. (Yu et al., 2017)
8. LSR: Low-rank constraint-based SR. Make sure that the network is both sparse and structured (having modular structures). Two parameters are required. (Qiao et al., 2016)
9. tHOFC (topographical similarity-based HOFC): A high-order FC metric and a good supplement of traditional PC-based network, with inter-regional functional relationship

estimated by the FC topological similarity rather than the BOLD signal similarity. No parameter is required. (Zhang et al., 2016)

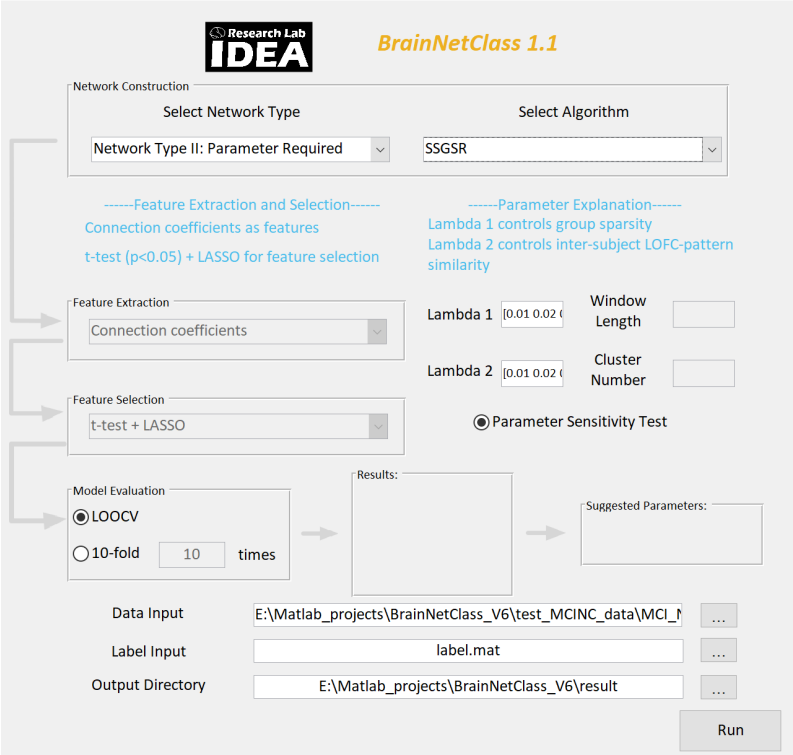
10. aHOFC (associated HOFC): A high-order FC metric and a good supplement of traditional PC-based network, which measures inter-level HOFC (the similarity between HOFC and conventional FC topological profiles. No parameter is required. (Zhang et al., 2016)
11. dHOFC: Dynamic FC-based HOFC that measures temporal synchronization of dynamic FC time series. Two parameters are required. (Chen et al., 2016)

Tips: How to choose network construction methods

- If users want to use a simple yet reliable network construction method, PC, tHOFC, and aHOFC are suggested. Compared to PC, tHOFC and aHOFC are more robust to noise yet interpretable, and most importantly, they provide supplementary information to PC.
- By using the dynamic FC, dHOFC could capture more high-level complex interaction among brain regions and perform better than the conventional low-order static FC. Therefore, for diseases (e.g., mental disorders) assumed to have little alterations in the static brain network, dHOFC is suggested.
- For data with potentially higher noise level, SR-based methods can be used. If the generated brain networks is too sparse, the user may choose WSR, WSGR, or SLR to make it less sparse, contain more strong connections, or have certain structures. If data is quite heterogeneous across subjects and the PC-based networks show large variability, users may choose group-wise sparse representation, such as GSR or SSGSR, to make the networks more similar across a group of subjects.

Note: It is not required to pre-specify parameters for the Network Type I, but when choose a method from the Type II, the user needs to specify the related parameter(s) by entering a range (or ranges) of it (them). Although the user could use the default parameter ranges, it is recommended to carefully set the range(s), as the parameters could significantly affect the constructed brain networks and the subsequent classification results. The toolbox will provide default parameter range(s) for users, but they are allowed to change the default parameters based on their own preference.

For example, as shown in the figure below, the default ranges for both parameters (λ_1 and λ_2) in the SSGSR method are from 0.01 to 0.1 with the increment of 0.01, i.e., [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1]. Also, it can be set as [2^-4 2^-3 2^-2 2^-1 2^0 2^1 2^2 2^3 2^4] (indicating [2^{-4} 2^{-3} 2^{-2} 2^{-1} 2^{0} 2^{1} 2^{2} 2^{3} 2^{4}]). The roles or functions of the λ_1 and λ_2 are shown in the panel below in blue for user's information.



For all the SR-based methods, we have λ_1 (and λ_2) to be specified, while for the dHOFC method, we have window length and number of clusters to be specified since it calculates and uses dynamic functional connectivity.

Tips: The memory and computation time required are proportional to the sample size, number of ROIs in the brain, brain network construction method, and number of parameters and their ranges (if applicable). We recommend use computer clusters or computer servers with a good amount of memory to facilitate the computing. The toolbox will also pop out a rough estimation of the memory use before the computing starts (see **8. Before hitting the 'Run' button**).

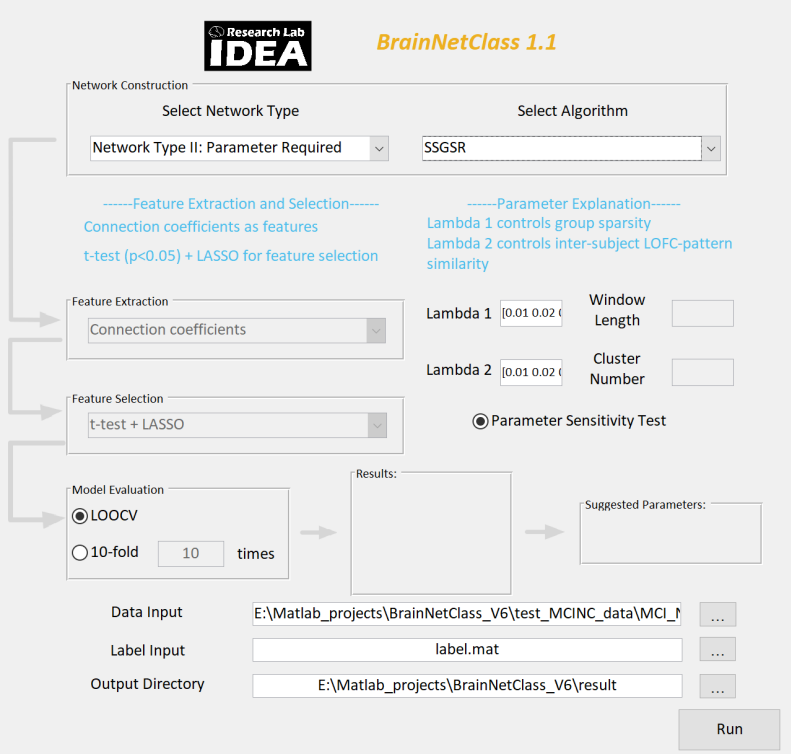
Please note that the toolbox provides user an opportunity to save out the constructed brain networks for future use. Please see "**11. Batch mode**".

6. Feature extraction and selection

After choosing a network construction method, the user needs to choose the methods for feature extraction and selection. We provide two types of feature extraction methods: connection coefficients (i.e., by directly using the connectivity strength of each link as a feature) and local clustering coefficient (i.e., one of the widely used nodal characteristic calculated based on a weighted graph with graph theoretical analysis on the brain network). For more details, please see (Chen et al., 2016; Zhang et al., 2019).

If the user chooses a network construction method requiring no parameters, he/she will then choose the method for feature extraction (i.e., connection coefficients and weighted local clustering coefficients) and selection (i.e., *t*-test, LASSO, and *t*-test + LASSO).

If the user chooses a network construction method requiring parameter(s), specific feature extraction/selection method will be automatically suggested to the user and it may be or may not be changed. For example, for the SR-based methods, users are restricted to use the connection-based coefficients as features combining with *t*-test+LASSO to perform feature selection (see the figure below). For the dHOFC method, users are restricted to use weighted-graph local clustering coefficients as features and LASSO to perform feature selection. All these are based on our best practice.



Note: The default threshold used for the *t*-test is $p < 0.05$. The parameter used for LASSO is 0.1. If the user wants to change these settings, please go to BrainNetClass.m file in the toolbox's main folder to change the settings. For example, one may change the value at the end of

[Line 119] 'handles.default.lasso_lambda=0.1'

to any value other than 0.1. By decreasing it, the selected parameters will be more, and vice versa. Similarly, the default *p* value can also be changed by going to the code of BrainNetClass.m. Only experienced users are recommended to make such changes as he/she must know the consequences and is able to control the whole process. For less experienced user, we highly recommend you use the default setting. Please see "**11. Batch mode**" for instructions on how to change the default settings.

7. Classification and model evaluation

We provided two types of cross-validation strategies (see figure below):leave-one-out cross-validation (LOOCV) and 10-fold cross-validation. If the sample size is large, we recommend 10-fold cross-validation to save time. With a limited sample size, we recommend LOOCV. Of note, 10-fold cross validation will be run for many times (default: 10; the user may specify more than 10 though, e.g., 100, but it will increase the processing time significantly).

As shown in the above figure, there is a 'Parameter Sensitivity Test' function provided to assess the effects of the parameters on classification accuracy using LOOCV. If the model is not sensitive to the parameters, the accuracy should not change abruptly as seen from the resulted bar plot. It is a good property of the model. See (Zhang et al., 2019) for the detailed rationale for doing so. It is suggested to do so and include the result in the paper. When the user chooses any parameter-required brain network construction method, the parameter sensitivity test will be chosen automatically. While it is free to be de-selected, we still encourage the user to do this test, because it will not take too long time to run and will provide a valuable result.

The toolbox also automatically records the selection occurrence (%) of each combination of the parameters. This information can be used to evaluate model robustness and is suggested to be included in the paper. For a robust model, there should be one value of the parameter (or one combination of the parameters) significantly selected more frequently than others. It is also a good property of a model.

In addition to the numeric classification performance evaluations and the ROC curve, the user may

also want to know which features contributed more (a.k.a., contributing features) or more important to the disease classification. We thus provide the averaged weight derived from the SVM across all the cross-validation runs for each feature (either a link or a node, for dHOFC, they are clusters of links) as the feature importance measure. Another quantitative measurement of the feature importance, i.e., selection occurrence of each feature being selected in the feature selection across all cross-validation runs, could also indicate feature importance. The more frequently a feature has been selected, the more important this feature could be. In summary, there are two metrics generated by our toolbox for user to better evaluate contributing features.

Specifically, for dHOFC method, the resultant contributing features will be a little bit different, which are some clusters including some nodes/edges and they will be saved in a different way. The saved result ('result_features.mat' in the result folder) will be a cell array with four columns, representing the selection occurrence of each feature (cluster), the indices of the involved nodes in each cluster (one can use these ROI indices to trace back to the contributing brain regions), a matrix indicating the connections in each cluster (nROI × nROI, if there is a connection between two nodes, then the value of this connection is set to 1), and the mean weight for each feature (cluster), respectively.

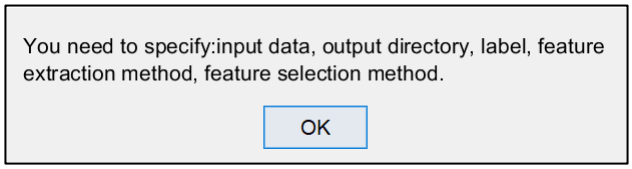
For other methods, if connection coefficients are used as features, the saved result will be a cell array with two columns. One is a matrix (nROI × nROI) with each element set to the averaged weight of that connection. The other is a matrix (nROI × nROI) with each element set to the normalized occurrence (%) of that connection. If local clustering coefficients are used as features, then the saved result will be a cell array with two columns. One is a vector (in a length of nROI) with each element representing an averaged weight of this node. The other is also a vector of the same length with each element representing the normalized occurrence (%) of this node.

Besides, after finishing the classification, we also show the group-averaged brain network for each of the two groups (saved as 'Mean_optimal_negativeLabel_network.mat' and 'Mean_optimal_positiveLabel_network.mat' in the result folder) in a form of two weighted adjacency matrices (if the user chooses the Type II network construction methods, they will be the networks constructed based on the optimal parameter(s)(as suggested by the toolbox based on the Parameter Sensitivity Test).

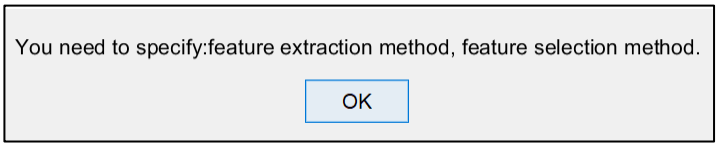
Another unique feature is that our toolbox saves the optimal classification model in the result folder ('saved_model.mat', see "**9. Result display and guidance of result interpretation**"). It provide an ability that when the new data is coming, the saved model can be directly applied on the new data to conduct classification based on it (i.e., to decide if this data is from a patient or a normal control). All the previously used and saved information, e.g., the network construction algorithm, the optimized parameters, the indices of the selected features, etc., will be automatically applied to this new data, but omitting the time consuming and resource demanding training and testing processes. This will highly improve the efficiency of the network-based classification.

8. Before hitting the 'Run' button

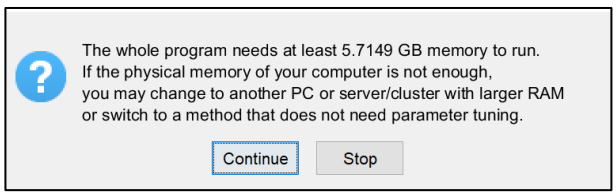
We have set up some reminders for the user to let them better use the toolbox by avoid wasting time on the failed computation due to some errors during the setting up. If the user forgets certain settings (i.e., feature extraction/selection method), a dialog box will pop out to remind the user, shown as below:



or

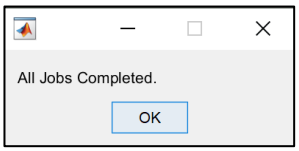


Once all the options are successfully set up, the user is ready to run the toolbox. There will be another reminder dialogue window on how much memory the toolbox could use. For example, when the user chooses dHOFC, a dialog window will pop up showing the estimated memory indicating the entire analysis will require at least 5.7 GB physical memory as shown below. Note that it is just a rough estimation. If the user still worries about not enough memory left, he/she may choose a network construction method without parameter optimization (i.e., Type I method), or narrow down the range of the parameters to be tuned (for example, by changing the parameter sampling strategy from [0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1] to [0.01 0.03 0.05 0.07 0.09]).

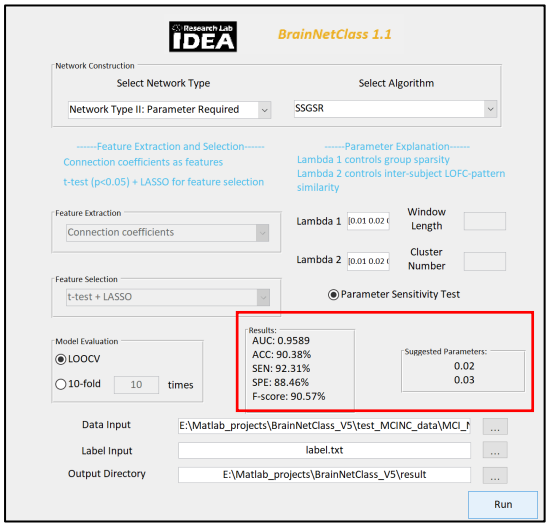


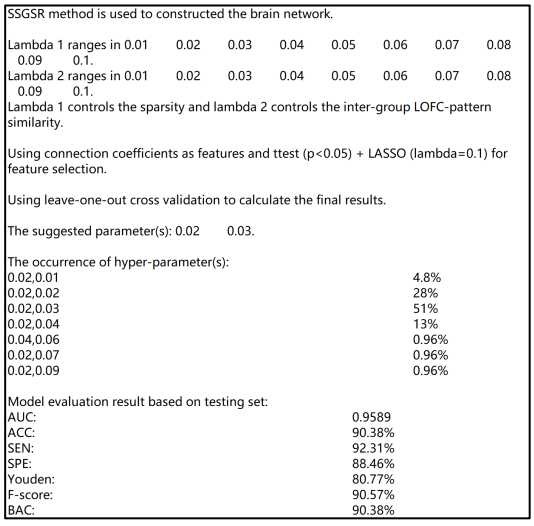
9. Result display and guidance of result interpretation

Once all the data analysis process has been finished, a dialog box indicating job finished will pop out, as shown below. If not, please wait for this dialog box patiently before doing anything further, as the toolbox is still running (cross-validation and parameter optimization may take longer-than-expect time).



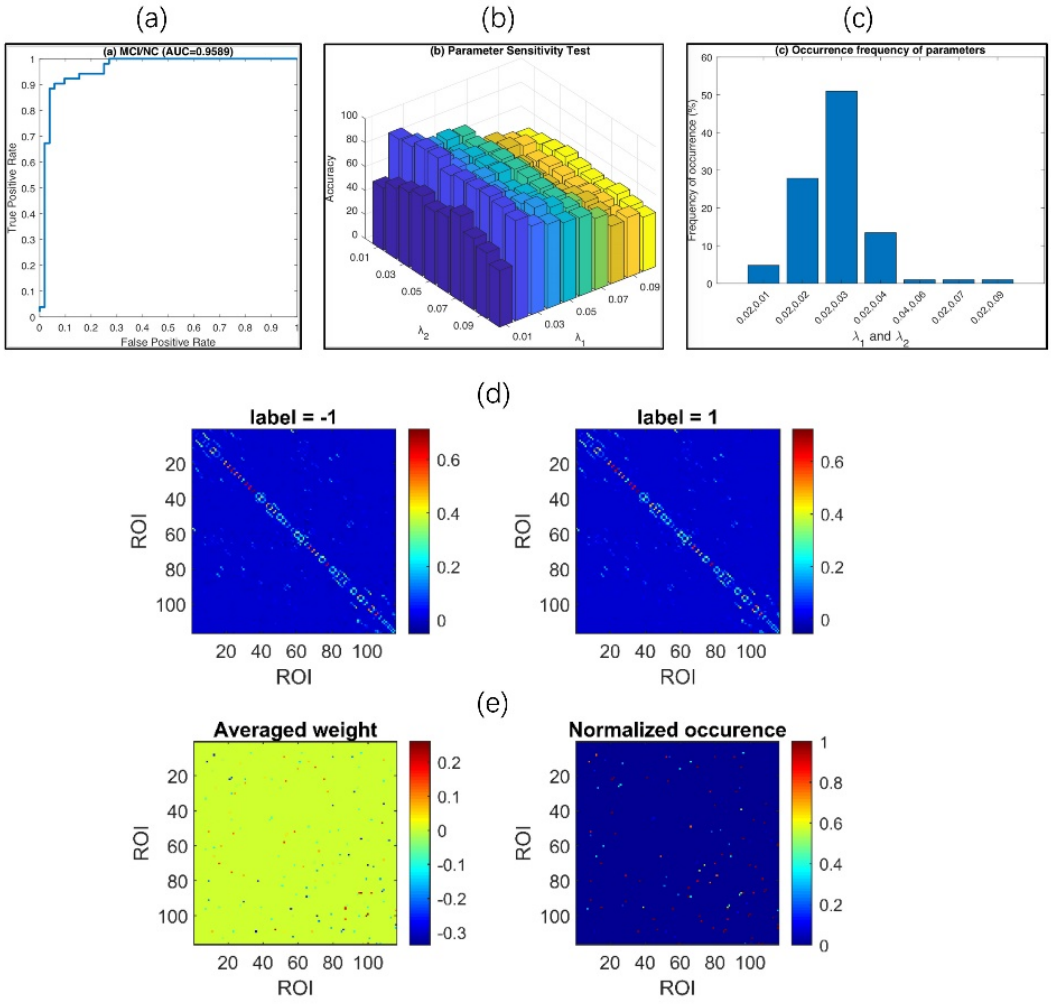
When all computational processes are finished, the classificaiton performance will be displayed on the GUI panel (see the highlighted part in the figure below). The results are also recorded and saved in a log file (in *.txt format). This log file records the method used to construct the brain network, the parameters user specified, feature extraction and selection methods, cross-validation methods, suggested parameter(s), occurrence of parameters, and performance of the classifier (see the right figure below).





Furthermore, if the user chooses parameter-required network construction methods and go through the whole process (including the parameter sensitivity test), the toolbox will generate five figures in the result folder (see the figures below) including::

a) Receiver Operating Characteristic (ROC) curve of the classification performance;
b) Classification accuracy bar plot based on the parameter sensitivity test;
c) Parameter selection occurrence for model robustness evaluation;
d) Averaged brain network for each group constructed based on the optimal parameters; and
e) Contributing features, shown as averaged weight and normalized occurrence.



The toolbox also generates four *.mat files in the result folder, including:

a) Mean_optimal_negativeLabel_network.mat
b) Mean_optimal_positiveLabel_network.mat (these (a) and (b) mat files correspond to the above figure (d)).
c) result_features.mat (for storing feature importance measurement, correpsonding to the above figure (e)).
d) saved_model.mat (for saving the whole model including all the settings; for its usage, see "**7. Classificaiton and model evaluation**" and "**11. Batch mode**").

For more interpretation of these generated results, please see the toolbox paper.

10.Exemplary data

We provide some exemplary data for the users to get familiar with toolbox usage. The data is from http://fcon_1000.projects.nitrc.org/indi/retro/BeijingEOEC.html (Beijing: Eyes Open Eyes Closed Study). The goal is to use resting-state fMRI time series from 116 brain regions to construct brain functional networks and predict whether a subject is in eyes closed or eyes open state. There is a full version (requires longer running time but generates a much better classification result) and a simplified version of the data (shorter running time, for quick walk-through). The labels (EC, EO) are provided in two versions as well (see label.txt and label_simple.txt). For more details and more experiments, please see the toolbox paper.

11.Batch mode

In addition to the GUI-based mode, BrainNetClass also offers a batch mode. The batch mode is for advanced users.

The related functions for batch processing or called during batch processing can be found in 'BatchExamples' folder in the main folder. They are summarized and explained below (for more information, please read the code):

1. 'param_select_demo.m'
This is the main batch-process function for all network-based classification that requires parameter optimization. It will be called by the provided demo functions listed in (3) and generate some results (e.g., ROC curve and other model evaluation metrics, as well as parameter sensitivity test result).

2. 'no_param_select_demo.m'
This is the main batch-process function for all network-based classification that does not require parameter optimization. It will be called by the provided demo functions listed in (3) and generate some results (e.g., ROC curve and other model evaluation metrics).

3. 'run_EC_EO_demo.m'
This is an exemplary demo function for users to conduct a two-class classification using dHOFC (the main method), SR (a baseline method), and PC (the other baseline method). It calls the main functions listed in (1) and (2). Users may modify it according to their needs and preference. For the details of the inputs and outputs and the options, please see 'param_select_demo.m' and 'no_param_select_demo.m'.

4. 'save_BrainNet.m'
Users may use this script to generate brain networks for each subject and save them for future use (e.g., to perform statistical comparison analysis). If running with it, the toolbox will skip the feature extraction/selection and the classification parts. This function is only in command-line mode.

5. 'input_BrainNet.m'
This script is useful when users have their own brain networks generated by other methods or toolboxes but also want to use this toolbox to perform network-based classification. It will skip network construction part and directly do feature extraction/selection and classification. Note: this is command-line mode only.

6. 'save_model.m'
With this script, users can save the trained classification model with the optimized parameter(s) and all the configurations (including the network construction algorithm used, feature extraction/selection methods used, and the cross validation method used) for future use. This function is useful if the users feel the model is ready to "use" and would like to directly apply it to the new data so that it will save time without having to go through the training and validation procedures repeatedly. For example, users may be able to save their classification model once it has

been trained with adequate samples and has reached satisfactory performance. They may not want to re-train the model every time when the new data is coming in. Therefore, they can direct use this script to save the trained model to perform a quick classification. Currently, it allows users to save the model, applying the saved model to perform classification for new data will be added in the future version.

For users to easily learn the script and the command-line mode, exemplar data is provided in the 'BatchExamples' folder as inputs, which contain 'ECEO_label.
mat' (containing the labels) and 'ECEO.mat' (containing the ROI time series). User can use 'run_EC_EO_demo.m' with these input files to run an exemplary analysis (see (3)).

References

Chen, X., Zhang, H., Gao, Y., Wee, C.Y., Li, G., Shen, D., Alzheimer's Disease Neuroimaging, I., 2016. High-order resting-state functional connectivity network for MCI classification. Hum Brain Mapp 37, 3282-3296.

Qiao, L., Zhang, H., Kim, M., Teng, S., Zhang, L., Shen, D., 2016. Estimating functional brain networks by incorporating a modularity prior. Neuroimage 141, 399-407.

Wee, C.Y., Yap, P.T., Zhang, D., Wang, L., Shen, D., 2014. Group-constrained sparse fMRI connectivity modeling for mild cognitive impairment identification. Brain Struct Funct 219, 641-656.

Yu, R., Zhang, H., An, L., Chen, X., Wei, Z., Shen, D., 2017. Connectivity strength-weighted sparse group representation-based brain network construction for MCI classification. Hum Brain Mapp 38, 2370-2383.

Zhang, H., Chen, X., Shi, F., Li, G., Kim, M., Giannakopoulos, P., Haller, S., Shen, D., 2016. Topographical Information-Based High-Order Functional Connectivity and Its Application in Abnormality Detection for Mild Cognitive Impairment. J Alzheimers Dis 54, 1095-1112.

Zhang, Y., Zhang, H., Chen, X., Liu, M., Zhu, X., Lee, S.-W., Shen, D., 2019. Strength and similarity guided group-level brain functional network construction for MCI diagnosis. Pattern Recognition 88, 421-430.