

GCRFs TOOL USER MANUAL

CONTENTS

1.	Introduction	2
2.	Overview of the GCRF method.....	4
3.	GCRF extensions	5
4.	Unstructured predictors	6
5.	Important definitions	7
6.	Installation	8
7.	Configuration.....	9
8.	Train and test	11
8.1	Train and test on networks	11
8.2	Train and test on temporal networks.....	14
8.3	Train and test on random networks.....	18
8.3.1	Train on random networks	18
8.3.2	Train on random networks	19
9.	Datasets	21
9.1	Add dataset.....	21
9.2	Manage datasets	24
10.	Case studies.....	25

1. Introduction

Structured regression models are designed to use relationships between objects for predicting output variables. In other words, structured regression models consider the attributes of objects and dependencies between the objects to make predictions.

General objective of regression is to predict the output variable Y , as accurately as possible, given an input vector of attributes X . Traditional supervised learning models, like neural networks, use only information contained in X to predict Y , while structured regression models use dependencies among outputs to improve final predictions. These problems can be seen as a graphs, where the nodes correspond to objects with attributes X and outputs Y , while the links of this graph contain weights that are given by the similarity measures. We usually have some prior knowledge about relationships among the outputs Y . Mostly, those relationships are application-specific where the dependencies are defined in advance, either by domain knowledge or by assumptions, and represented by statistical models. For example, relationships between hospitals can be based on similarity of their specialization, relationships between pairs of scientific papers can be presented as the similarity of sequences of citation, relationships between documents can be quantified based on similarity of their contents, etc.

The Gaussian Conditional Random Fields (GCRF) model is one type of structured regression models that incorporate the outputs of unstructured predictors (based on the given attributes values) and the correlation between output variables in order to achieve a higher prediction accuracy. This model was first applied in computer vision, but since then it has been used in different applications, and extended for various purposes.

GCRF GUI TOOL integrates various GCRF methods and supports training and testing those methods on real-world data from different domains. Common procedure for solving GCRF-based structure regression problems includes the following steps:

- (1) prepare training and test data
- (2) select and train an unstructured predictor
- (3) train a selected GCRF model
- (4) test selected GCRF model
- (5) obtain predicted values
- (6) calculate accuracy

Users provide a dataset for their specific problem at step 1 and GCRF GUI TOOL will finish all remaining structure regression steps. Figure 1 presents an overview of the system architecture of GCRF GUI TOOL.

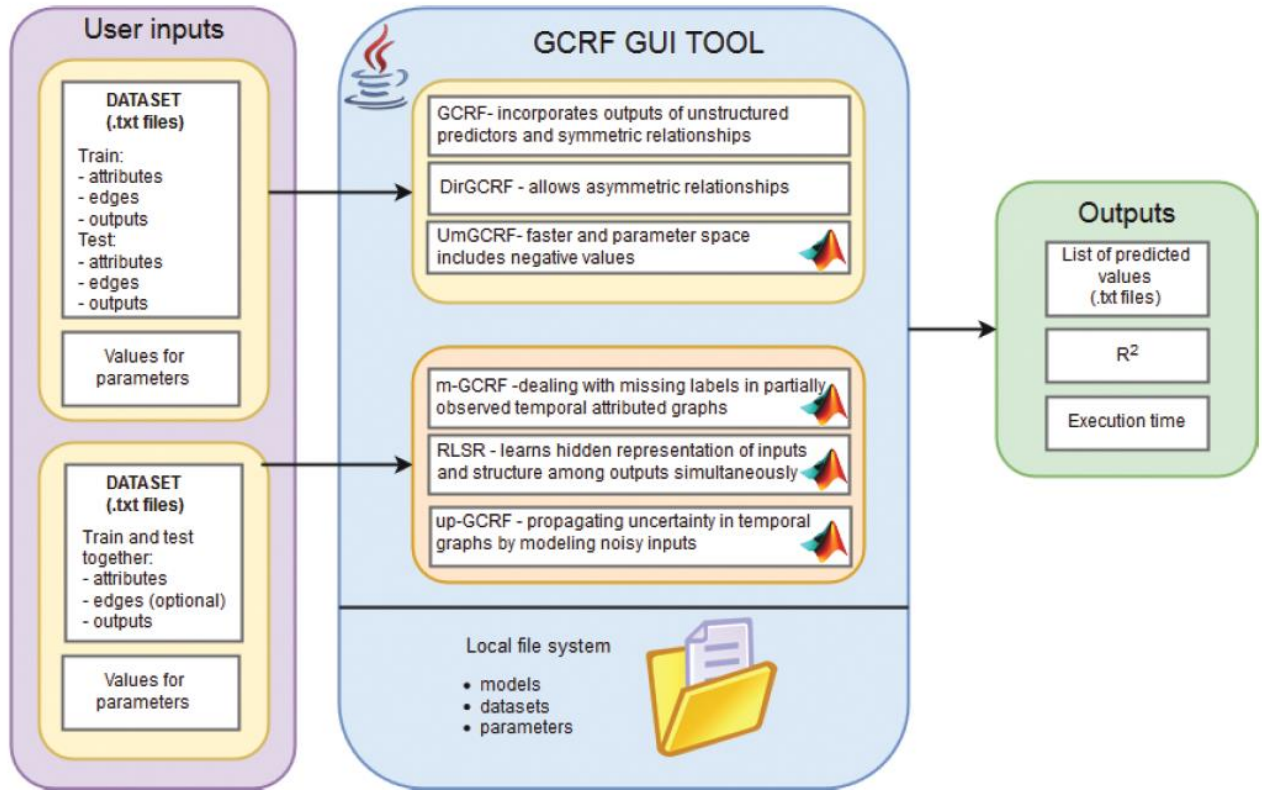


Figure 1. System architecture

To calculate the regression accuracy of all methods, we used R^2 coefficient of determination that measures how closely the output of the model matches the actual value of the data. A score of 0 indicates a very poor matching, while a score of 1 indicates a perfect match. R^2 coefficient of determination is calculated using following equation:

$$R^2 = 1 - \sum_i \frac{(y_i - \hat{y}_i)^2}{(y_i - y_{avg})^2}$$

where \hat{y}_i is the predicted value, y_i is the true value, and y_{avg} is the average of y values

2. Overview of the GCRF method

The **Gaussian Conditional Random Fields (GCRF)** model is a structured regression model that incorporates the outputs of unstructured predictors and the correlation between output variables in order to achieve a higher prediction accuracy. In a Conditional Random Field (CRF) model the observable attributes X interact with each of the targets Y directly and independently of one another. The CRF probability function can be represented by an equation of the form:

$$P(y|x) = \frac{1}{Z(x, \alpha, \beta)} \exp \left(\sum_{i=1}^N A(\alpha, y_i, x) + \sum_{j \sim i} I(\beta, y_i, y_j) \right)$$

where A and I are real valued functions that are known in CRF literature as association and interaction potential. The larger the value of A , the more y_i is related to attributes x . The larger the value of I , the more y_i is related to y_j . The potentials can be written as:

$$A(\alpha, y_i, x) = - \sum_{k=1}^K \alpha_k (y_i - R_{i,k}(x))^2$$

$$I(\beta, y_i, y_j) = - \sum_{l=1}^L \beta_l S_{ij}^l (y_i - y_j)^2$$

where K unstructured predictors are used to predict a single output y_i and L similarity functions are used to represent different types of dependencies among the nodes. The GCRF model is a CRF model with both quadratic feature and quadratic interaction functions that can be transposed directly onto a Gaussian multivariate probability distribution:

$$P(y|x) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} \exp \left(-\frac{1}{2} (y - \mu)^T Q (y - \mu) \right)$$

When setting these two conditional probability models equal to one another, we get a precision matrix (Q) defined in terms of the confidence of input predictors and the pairwise interaction structure, measured by α and β respectively. The learning task is to choose the parameters α and β to maximize the conditional log-likelihood of the set of training examples. Precision matrix is calculated by the following equation:

$$Q = \sum_k \alpha_k I + \sum_j \beta_j L_j$$

where L_j is the Laplacian matrix of j th S matrix. Representing input predictions as a matrix R , the formula for the final prediction can be concisely written as:

$$\mu = Q^{-1} R \alpha$$

3. GCRF extensions

GCRF has been used on a broad set of applications: climate, energy forecasting, healthcare, speech recognition, computer vision etc. GCRF is extended for various purposes, and following extensions of GCRF are integrated in GCRFs tool:

- **Directed GCRF (DirGCRF)** method extends the GCRF method by considering asymmetric similarity. In many real-world networks objects are asymmetrically linked, and standard GCRF could not be directly applied on these problems since this method requires a symmetric matrix. For example, DirGCRF is applied on the Teenagers [12] dataset that consists of three temporal observations of 50 teenagers. Friendship network (up to 12 best friends) and teenager's alcohol consumption (ranging from 1 to 5) are provided for each time point. The goal was to predict alcohol consumption at the observation time point 3, based on two previous observations. Results showed that the DirGCRF model has 17% larger accuracy than the standard GCRF model, and 4% larger accuracy than the Neural Network.
- **Unimodal GCRF (UmGCRF)** method extends the GCRF parameter space to facilitate joint modeling of positive and negative influences and yields a huge speed up. GCRF is restricted to positive weights and UmGCRF expands the parameter search space to allow for negative links and negative influence of the unstructured predictors while maintaining the positive semi-definiteness of the precision matrix and improve computational efficiency. UmGCRF was evaluated on the problem of predicting monthly hospital admissions for 189 classes of diseases in California. For each of more than 35 million inpatient discharge records collected over 9 years in the California HCUP database there are up to 253 diagnosis codes in CSS coding schema. Authors constructed monthly disease graphs such that each node represents one disease, and those graphs had 189 nodes because of incomplete information over the time period analyzed. Relationships between diseases are based their similarity (on a zero to one scale). The UmGCRF had 17% and 12% improvement in test accuracy over input baselines. Also, new mathematical formulation caused a dramatic speed up, it far exceeds the speed and scalability of standard GCRF, and it is nearly as fast as approximation techniques.
- **Marginalized GCRF (m-GCRF)** is structured regression method for dealing with missing labels in partially observed temporal attributed graphs. A common problem in a real-life applications is that the large fraction of observations is often missing. This method extends GCRF to naturally handle missing labels, rather than expecting the missing data to be treated in a preprocessing stage. The benefits of the m-GCRF are demonstrated on a challenging application for predicting precipitation based on partial observations of climate variables in a temporal graph that spans the entire continental US. Each temporal graph has 1218 nodes (meteorological stations). The spatial information is used for calculating similarities (correlations) between stations, and for each station participation and 6 more attributes are provided. There are no missing values in input variables, but about 5% of the dependent variables (precipitations) are missing. Experiments on this data provided an evidence that m-GCRF brings accuracy improvement (5% versus Neural Network).

- **Uncertainty Propagation GCRF (up-GCRF)** is a method for propagating uncertainty in temporal graphs by modeling noisy inputs. It is aimed to support structured regression for long-term decision making, which has been of interest in many high impact applications. Up-GCRF method takes into account uncertainty that comes from the data when estimating uncertainty of the model predictions. up-GCRF was evaluated on the California HCUP data. Problem considered in this study is long-term prediction of admission and mortality rate based on inpatient discharge data. In all experiments the up-GCRF model outperformed baselines in terms of both accuracy and uncertainty propagation.
- **Representation Learning based Structured Regression (RLSR)** method is able to simultaneously learn hidden representation of objects and relationships among outputs. The objective of the method is to improve the representational power of a general class of GCRFs by introducing hidden variables that are nonlinear functions of input variables. Such a method can learn more informative representations and structural dependencies simultaneously. One of RLSR applications is prediction of the daily solar energy income at 98 Oklahoma Mesonet sites. On this dataset the RLSR model outperformed all baselines by at least 50%.

4. Unstructured predictors

Different unstructured predictors can be incorporated in the GCRF methods and this tool integrates following:

- **Neural networks (NN)** - Neurons in an artificial neural network are grouped in three layers: input, output, and hidden layer. The number of neurons in the input layer is same as the number of attributes in the chosen dataset, while the number of neurons in the output layer is 1 since we have only one predicted variable. User is asked to insert the number of neurons in the hidden layer.
- **Linear Regression (LR) or Multivariate Linear Regression (MLR)** - Linear regression is an approach for modeling the relationship between a dependent variable y and one or more explanatory variables x . This relationship is modeled using linear predictor functions whose parameters are estimated from the data. LR or MLR is used depending on the number of attributes in the dataset.

5. Important definitions

- Unstructured predictor – traditional machine learning techniques (e.g. neural network)
- X – attributes
- Y – outputs
- R – values provided by the unstructured predictor
- S – similarity matrix that quantifies the connections among the nodes of the graph (structure)
- α – parameter that defines confidence of the unstructured predictor, a greater value of α means that the model is putting more emphasis on values that are provided by the unstructured predictor (R)
- β – parameter that defines confidence of the structure, a greater value of β means that the model is putting more emphasis on structure (S)
- Learning rate – tuning parameter in an optimization algorithm that determines the step size at each iteration
- Gradient descent – learning algorithm used to find the optimal values for parameters α and β

6. Installation

Download latest zip file from following [link](#).

Extract zip to the desired location.

The structure of extracted folder *GCRFsTOOL* is presented at Figure 2. To start the tool run the executable file *gui.jar* that is located in the *GCRFsTOOL*.

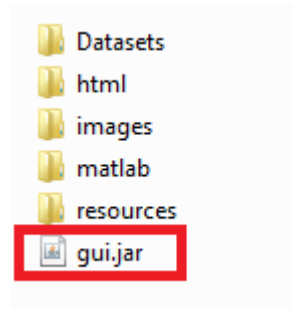


Figure 2. GCRFsTOOL folder structure

The *matlab* folder contains Matlab source code for all methods that are implemented in MATLAB.

The *html* folder contains files for Help.

The *Datasets* folder contains dataset samples that can be used to test specific methods. Samples are provided in .txt files, in the format that is required by this tool. In these samples files are denoted as follows:

- readme.txt – main information about dataset
- x.txt - attributes
- y. txt - desired output
- s.txt - similarity matrix that quantifies graph that presents relationships between objects

Each time when new dataset is added it will be stored in this folder.

7. Configuration

When you run *gui.jar* at the first time Configuration panel will be displayed and main menu will be disabled (Figure 3).

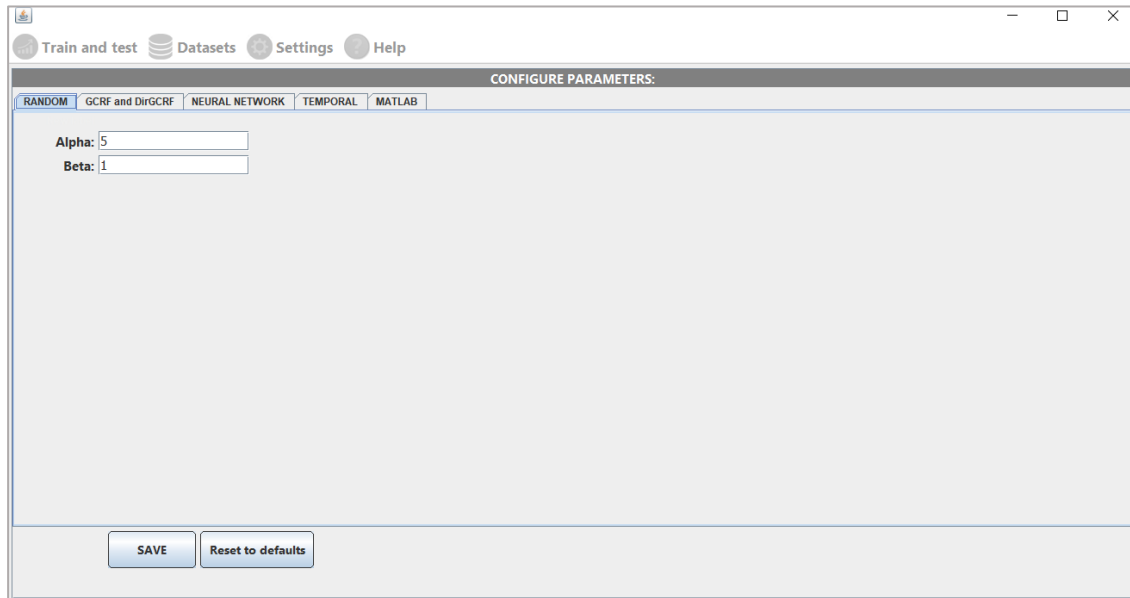


Figure 3. Configuration panel

Most of the parameters have default values and those values can be changed. Configuration panel includes five tabs:

1. Random - default values for α and β parameters that are used to calculation Y array for random generated networks.
2. GCRF and DirGCRF - default values of initial α and β parameters, learning rate and the maximal number of iterations for gradient descent algorithm.
3. Neural Network - default number of iterations and number of hidden neurons for NN training.
4. Temporal - default number of iterations and parameters for m-GCRF (default values for regularization α and β parameters) and RLSR (default _ set, number of iterations for NN and maximal number of iterations for structure learning part-SSE and for backtracking line search in structure learning-SSE LS).
5. Matlab - path to matlab.exe file and value for Matlab proxy timeout. In order to successfully save configuration, user should insert path to *matlab.exe* file (Figure 4). If user does not want to use Matlab or if Matlab is not installed on their computer they can uncheck the “Use methods implemented in MATLAB” check box. In that case GUI will show only the methods implemented in Java.

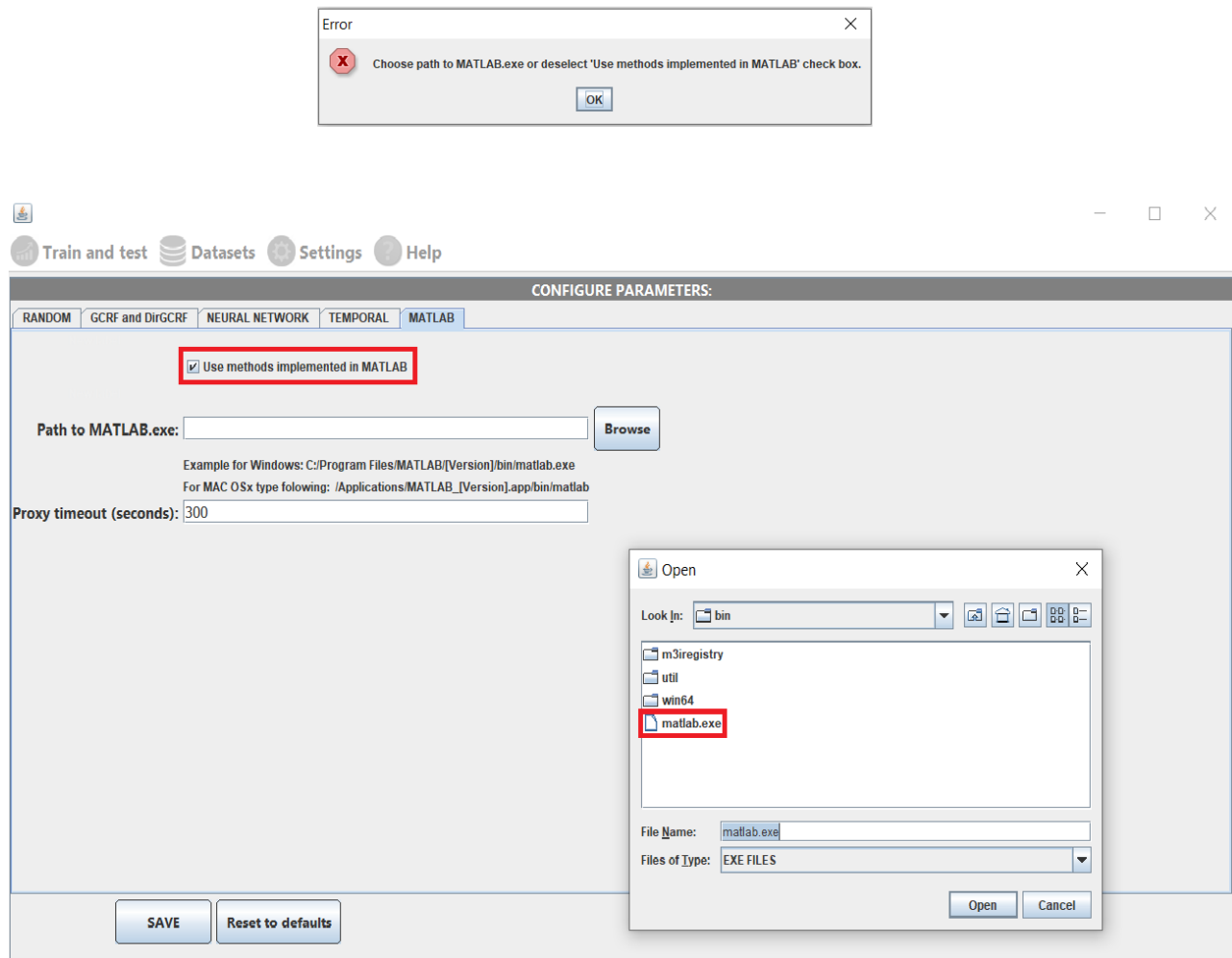


Figure 4. Setting up a connection with Matlab

When user clicks *Save* button Configuration panel will disappear and main menu will be enabled. If user wants to change default values of the parameters the *Configuration* panel can be opened from main menu: *Settings->Configuration*.

8. Train and test

Train on networks menu item has three subitems (Figure 5):

- Train and test on networks
- Train and test on temporal networks
- Train and test on random networks

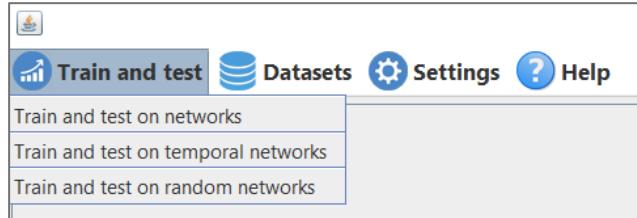


Figure 5. Train and test menu item

8.1 Train and test on networks

Train and test on networks menu item (Figure 6) is used to train following GCRF methods:

- Standard GCRF
- Directed GCRF (DirGCRF)
- Unimodal GCRF (UmGCRF)

Note: UmGCRF is visible only if you checked use MATLAB in configuration panel.

The screenshot shows the 'Train and test' software window. At the top, there is a navigation bar with icons and labels for 'Train and test', 'Datasets', 'Settings', and 'Help'. The main area is divided into three sections: 'DATA:', 'UNSTRUCTURED PREDICTOR:', and 'METHOD:'. In the 'DATA:' section, there is a 'Dataset:' dropdown menu with 'choose dataset' selected and a 'Model name:' text input field with the placeholder 'Insert the name of your model'. In the 'UNSTRUCTURED PREDICTOR:' section, there is an 'Unstructured predictor:' dropdown menu with 'choose predictor' selected and a 'Test predictor' button. In the 'METHOD:' section, there is a 'Method:' dropdown menu with 'choose method' selected and a 'TRAIN and TEST' button.

Figure 6. Train and test on networks

First step is to choose the dataset and name your model (Figure 7). This software provides 4 dataset samples for this functionality. Also, users can add their own datasets using *Add dataset* option in *Datasets* menu item.

This screenshot shows a closer view of the 'DATA:' section of the software. The 'Dataset:' dropdown menu is now set to 'Geostep Symmetric'. The 'Model name:' text input field contains the text 'test'.

Figure 7. Train and test on networks -> Data

Second step is to choose the unstructured predictor. Two unstructured predictors can be selected: neural networks and linear regression. If neural network is selected user should insert the number of hidden neurons and the number of iterations (Figure 8). Data for neural network will be normalized automatically. If linear regression is selected, standard linear regression or multivariate linear regression will be applied, depending on number of attributes in the dataset.

UNSTRUCTURED PREDICTOR:

Unstructured predictor: neural network ? Test predictor

No. of hidden neurons:

No. of iterations:

Figure 8. Train and test on networks ->Unstructured predictors – “neural network” option

Final step is to select the desired method from *Method* combo box. Fields for that method’s parameters will automatically show bellow the combo box and the fields will be filled with the default values that are set in the *Configuration* panel (Figure 9). All parameters values can be changed.

METHOD:

Method: GCRF ?

First alpha:

First beta:

Learning rate:

Max. iterations:

TRAIN and TEST

Figure 9. Train and test on networks ->Method – “GCRF” option

When user clicks the *TRAIN and TEST* button, the training and testing process will start. Model accuracy will be automatically calculated and R^2 value will be shown. Example of the results is presented at Figure 10.

DATA:

Dataset: Geostep Asymmetric ?

Model name:

UNSTRUCTURED PREDICTOR:

Unstructured predictor: neural network ? Test predictor

No. of hidden neurons:

No. of iterations:

METHOD:

Method: DirGCRF ?

First alpha:

First beta:

Learning rate:

Max. iterations:

Apply standard GCRF: ☐ ?

TRAIN and TEST

Results

Testing with same data:
 * R^2 value for DirGCRF is: 0.788
 Time in seconds:
 * DirGCRF: 0.33

Testing with test data:
 * R^2 DirGCRF: 0.5274
 Predicted values for test data successfully exported.
 File location: ...

OK

Figure 10. Train and test on networks - example with the results

Each time when user trains a new model, folder for that model will be created in *GCRFsTOOL* folder. Models will be grouped by methods. After training process, folder for each model will contain following folders (example is presented at Figure 11):

1. data – where .txt files with data are copied
2. nn, lr or mlr – contains files with parameters for unstructured predictor
3. parameters – contains files with parameters for selected method
4. test – .txt file with predicted values

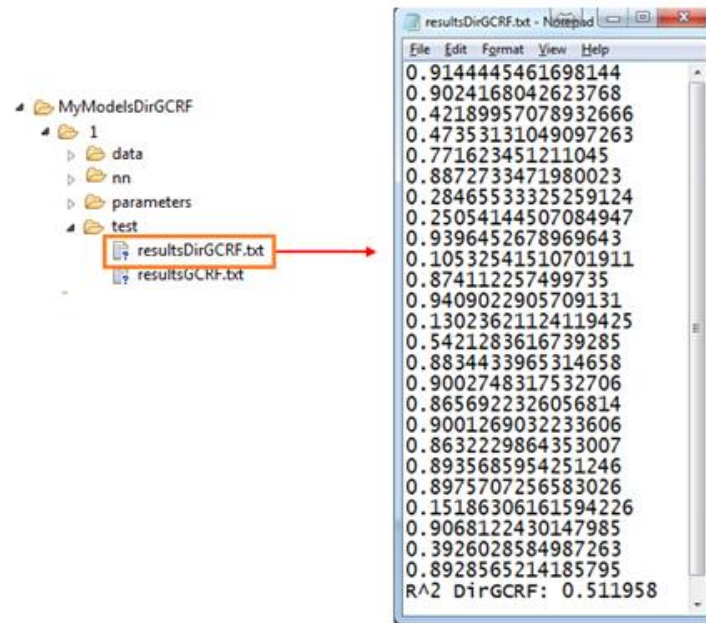


Figure 11. Train and test on networks – Example of the file with predicted values for model “1”

8.2 Train and test on temporal networks

Train and test on temporal networks menu item (Figure 12) is used to train following GCRF methods:

- Representation Learning based Structured Regression (RLSR)
- Uncertainty propagation GCRF (up-GCRF)
- Marginalized Gaussian Conditional Random Fields (m-GCRF)

Note: All methods are visible only if you checked use MATLAB in configuration panel (Figure 13).

Train and test Datasets Settings Help

DATA
Provide train and test data together

Dataset: choose dataset ?

☐ Learn similarity

No. of time points:

No. of time points for train:

Model name:

METHOD:

Method: choose method

TRAIN and TEST

Figure 12. Train and test on temporal networks

METHOD:

Method: no available methods without MATLAB

Figure 13. Train and test on temporal networks - Methods combo box if “use MATLAB” is not checked

First step is to choose the dataset and name your model (Figure 14). For those methods data for training and data for testing are provided together, so user is asked to insert the number of time points that will be used to train the method. This software provides 3 dataset samples for this functionality (one for each methods). Also, users can add their own datasets using *Add dataset* option in *Datasets* menu item. RLSR and up-GCRF methods do not require similarity information, as they can learn similarity.

DATA
Provide train and test data together

Dataset: Energy RLSR ?

☒ Learn similarity

No. of time points: 1600

No. of time points for train: 1000

Model name: 1

Figure 14. Train and test on temporal networks -> Data

Second step is to select the desired method from *Method* combo box. Fields for that method's parameters will automatically show below the combo box and the fields will be filled with the default values that are set in the *Configuration* panel (Figure 15). All parameters values can be changed..

Figure 15. Train and test on temporal networks -> Method – “RLSR” option

When user clicks the *TRAIN and TEST* button, the training and testing process will start. Since all methods are implemented in MATLAB they may require more time to provide results and it is not possible to show the progress (Figure 16). GCRFs TOOL launches and controls a Matlab session without any user intervention. When the function is completed, the session is automatically closed and the results will be shown (Figure 17).

Figure 16. Train and test on temporal networks – Example

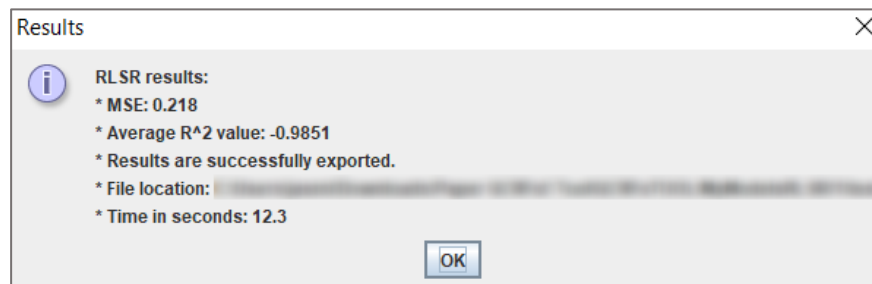


Figure 17. Train and test on temporal networks – Example of the results

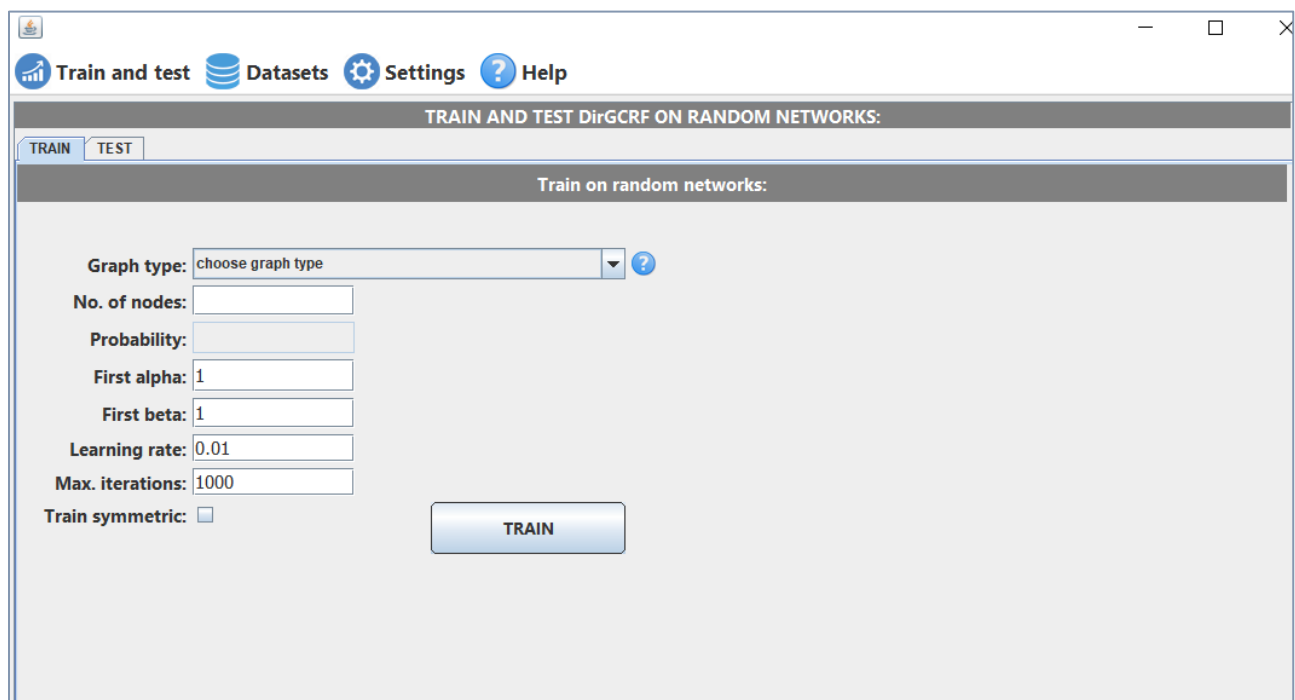
8.3 Train and test on random networks

The purpose of *Train and test on random networks* option is to test the accuracy of DirGCRF method under controlled conditions on different types of directed graphs and to compare it with the accuracy of standard GCRF.

For this functionality training and tested process are separated, since one model can be tested with different setups.

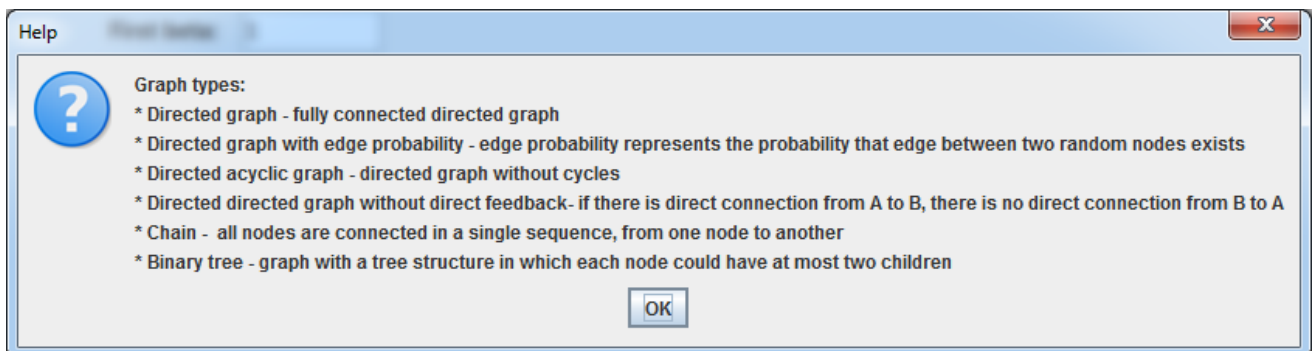
8.3.1 Train on random networks

First tab, *Train* tab (Figure 18), is used to train DirGCRF method (and GCRF if *Train symmetric* check box is checked) on randomly generated graph. User should select graph type (Figure 19) and insert number of nodes. Other parameters have default values that can be changed. For all models with randomly generated data new folder will be created in GCRFsTOOL folder and model is named based on graph type and number of nodes.



The screenshot shows a software window titled "Train and test" with a menu bar containing "Train and test", "Datasets", "Settings", and "Help". Below the menu bar is a tabbed interface with "TRAIN" and "TEST" tabs. The "TRAIN" tab is active, showing a section titled "Train on random networks:". This section contains several input fields: "Graph type:" with a dropdown menu showing "choose graph type" and a help icon; "No. of nodes:"; "Probability:"; "First alpha:" with a value of 1; "First beta:" with a value of 1; "Learning rate:" with a value of 0.01; "Max. iterations:" with a value of 1000; and a "Train symmetric:" checkbox which is currently unchecked. A "TRAIN" button is located at the bottom right of the input fields.

Figure 18. Train on random networks



The screenshot shows a "Help" window with a question mark icon. It lists "Graph types:" followed by six bullet points: "* Directed graph - fully connected directed graph", "* Directed graph with edge probability - edge probability represents the probability that edge between two random nodes exists", "* Directed acyclic graph - directed graph without cycles", "* Directed directed graph without direct feedback- if there is direct connection from A to B, there is no direct connection from B to A", "* Chain - all nodes are connected in a single sequence, from one node to another", and "* Binary tree - graph with a tree structure in which each node could have at most two children". An "OK" button is at the bottom.

Figure 19. Different types of directed graphs

When user clicks the *TRAIN* button, the training process will start. When the process is completed the results will be shown in the table below the *TRAIN* button (Figure 20).

TRAIN AND TEST DirGCRF ON RANDOM NETWORKS:

TRAIN TEST

Train on random networks:

Graph type: directed graph

No. of nodes: 100

Probability:

First alpha: 1

First beta: 1

Learning rate: 0.01

Max. iterations: 1000

Train symmetric: ☒

TRAIN

Alg.	Alpha	Beta	R ² (with same data)
DirGCRF	29.066022903809433	6.223692662043896	0.9699720409545994
GCRF	28.33856366059981	4.815144659500325	0.8446324297566534

Time in seconds: DirGCRF: 6.28 GCRF: 5.62

Figure 20. Train on random networks - Example

8.3.2 Train on random networks

Second tab, *Test* tab (Figure 21), is used to test trained models on randomly generated graphs. User should select model (Figure 22) and insert number of graphs for testing process.

TRAIN AND TEST DirGCRF ON RANDOM NETWORKS:

TRAIN TEST

Test on random networks:

Model: choose model

No. of nodes:

Probability:

No. of graphs:

TEST

Figure 21. Test on random networks

Model: choose model
 No. of nodes: 100
 Probability:
 No. of graphs:
 TEST

Figure 22. Test on random networks – Example of models

When user clicks the *TEST* button, the testing process will start. When the process is completed the results will be shown in the table below the *TEST* button (Figure 23).

TRAIN AND TEST DirGCRF ON RANDOM NETWORKS:

Test on random networks:

Model: DirectedGraph - 100nodes
 No. of nodes: 100
 Probability:
 No. of graphs: 5
 TEST

No.	R ² DirGCRF	R ² GCRF
1	0.9641230790157287	0.8829203261512062
2	0.9440267876498273	0.8380475441226853
3	0.9562781008841896	0.7596553877747935
4	0.9648877499877694	0.7164315215478119
5	0.9469200230738087	0.7805601082741356
Average	0.9552471481222649	0.7955229775741266
Standard deviation	0.008579320694	0.058672762285

Figure 23. Test on random networks – example

Example of the folder structure for Directed graph with 100 nodes and file with test results for 5 randomly generated graphs is presented at Figure 24.

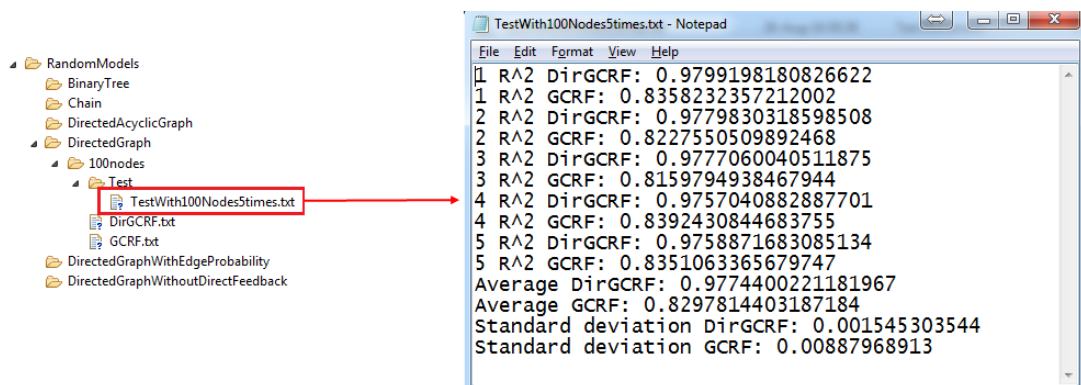


Figure 24. Example of the folder structure and file with test results

9. Datasets

Datasets menu item has two subitems:

- Add dataset
- Manage datasets

9.1 Add dataset

In *Datasets* -> *Add dataset* menu item new dataset can be added (Figure 25). For each dataset, the user should provide .txt files with edges, attributes and outputs. The formats of these files are described in Table 1. Explanations are also included in the tool and user can see them by clicking question mark icon next to the file pickers (Figure 26).

There are two types of datasets that can be added:

1. Dataset in which training and test data are provided separately (networks) - this dataset can be used with models that are listed in the Train on networks menu item. Example is presented at Figure 27.
2. Dataset in which training and test data are provided together (temporal networks) - this dataset can be used with methods that are listed in the Train on temporal networks menu item. In that panel the user will be asked to insert how to split the data (how many time points should be used for training, testing and validation). For these datasets a file with edges is not mandatory – use can choose the “Learn similarity” option. Example is presented at Figure 28.

The screenshot shows a software window titled "Add dataset" with a menu bar containing "Train", "Test", "Datasets", "Settings", and "Help". The window is divided into two main sections: "TRAIN DATA:" and "TEST DATA:". The "TRAIN DATA:" section contains a "Dataset name:" text field, and three file selection fields: "File with edges:", "File with attributes:", and "File with outputs:". Each file field has a "Browse" button and a question mark icon. There is also a checkbox labeled "Learn similarity". The "TEST DATA:" section contains three file selection fields: "File with edges:", "File with attributes:", and "File with outputs:". Each file field has a "Browse" button and a question mark icon. At the bottom of the "TEST DATA:" section, there is a checkbox labeled "train and test data are provided together". A "SAVE" button is located at the bottom center of the window.

Figure 25. Add dataset

Table 1. Formats of files in datasets

File	Description	Format
Edges (s.txt)	Connections between nodes (edges)	<ul style="list-style-type: none"> - Format: from node, to node, weight (for undirected graphs both directions should be included). - Each edge should be in a separate line. - Nodes are represented by ordinal numbers (numbers, from 1 to the number of nodes).
Attributes (x.txt)	Value of each attribute for each node	<ul style="list-style-type: none"> - Attributes for each node should be in a separate line. - Attributes should be comma separated. - All attributes should be numbers. - Order should be consistent with ordinal numbers of nodes in the file with edges. - For more than one time stamp attributes for one node should be in one line, comma separated.
Outputs (y.txt)	Actual output for each node	<ul style="list-style-type: none"> - Each output should be in a separate line. - Output should be number. - Order of outputs should be consistent with ordinal numbers of nodes in the file with edges. - For more than one time stamp outputs for one node should be in one line, comma separated.

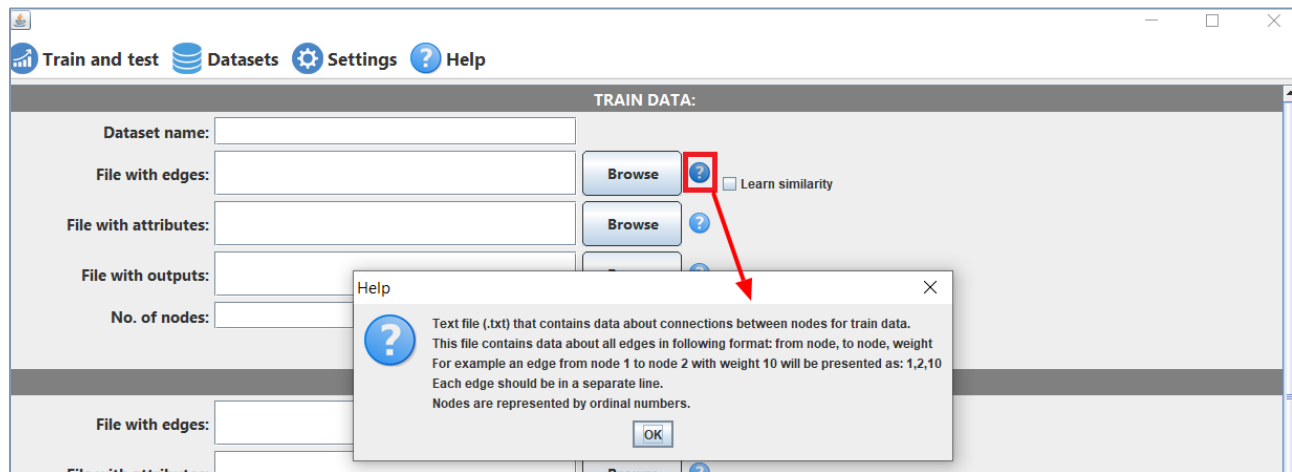


Figure 26. Example of explanation for file structure

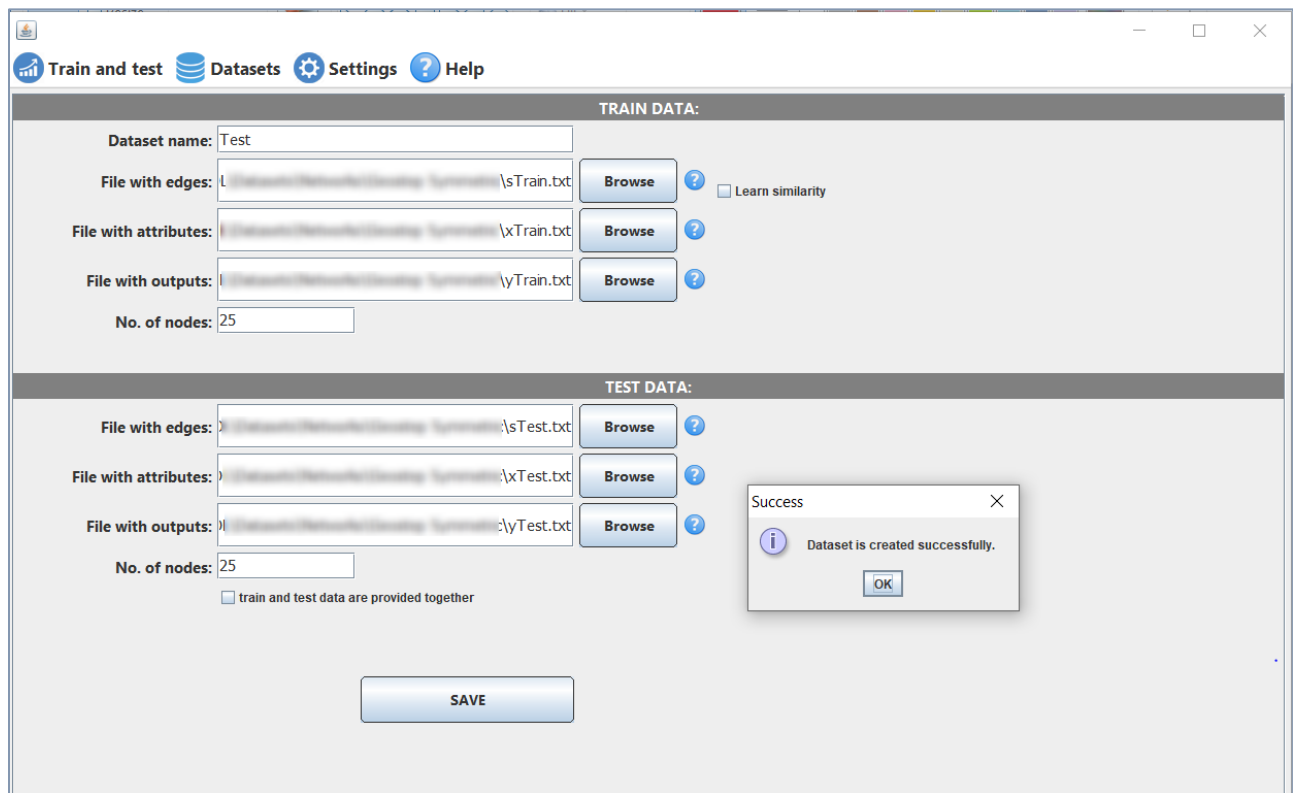


Figure 27. Example of adding a dataset in which training and test data are provided separately

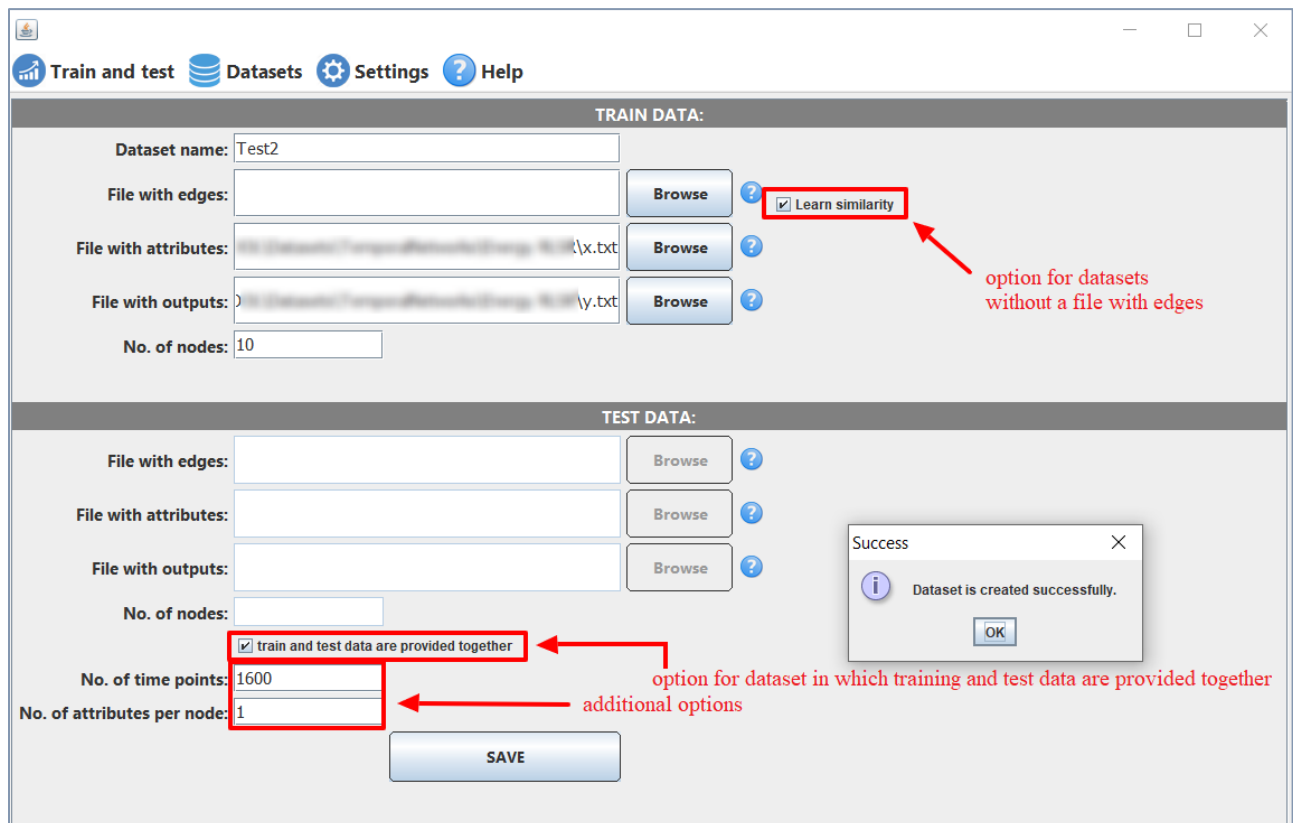
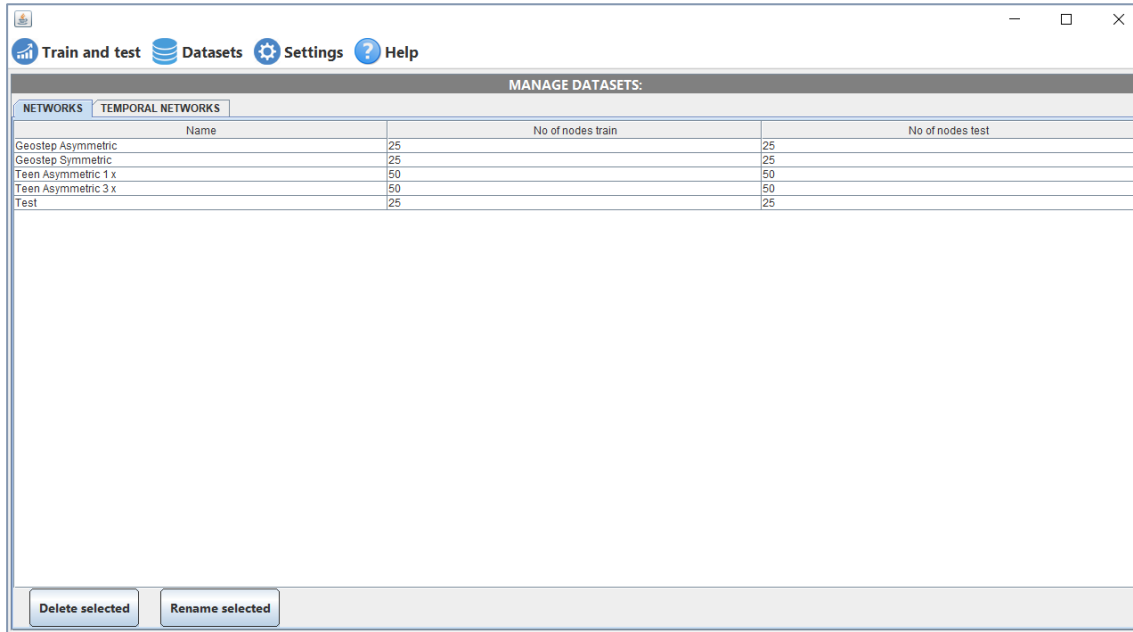


Figure 28. Example of adding a dataset in which training and test data are provided together

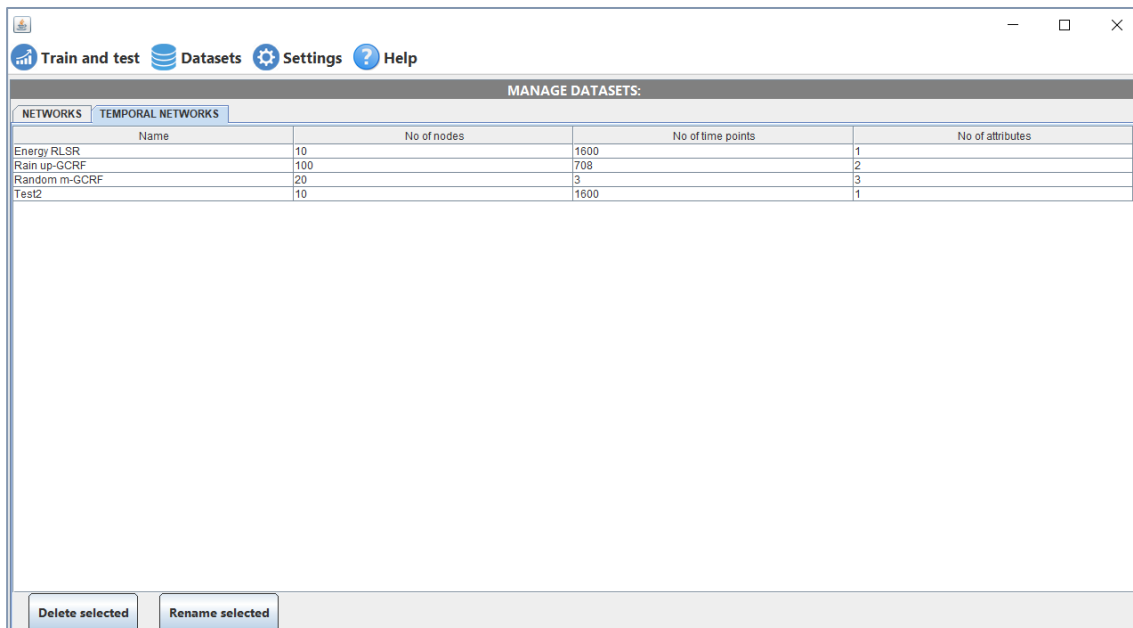
9.2 Manage datasets

All datasets are listed in *Datasets* -> *Manage datasets* menu item. Datasets are divided in two tabs: *NETWORKS* for the datasets in which training and test data are provided separately (Figure 29) and *TEMPORAL NETWORKS* for the dataset in which training and test data are provided together (Figure 30). Each dataset can be deleted or renamed.



Name	No of nodes train	No of nodes test
Geostep Asymmetric	25	25
Geostep Symmetric	25	25
Teen Asymmetric 1 x	50	50
Teen Asymmetric 3 x	50	50
Test	25	25

Figure 29. Manage datasets – Networks



Name	No of nodes	No of time points	No of attributes
Energy RLSR	10	1600	1
Rain up-GCRF	100	708	2
Random m-GCRF	20	3	3
Test2	10	1600	1

Figure 30. Manage datasets – Temporal networks

10. Case studies

Main software functionalities can be presented throughout two case studies:

- *Case study 1:* The Geostep dataset consists of 50 treasure hunt games. Each game can have maximum 10 clues and each clue belongs to one of 4 categories (business, social, travel, and irrelevant). Each game has six attributes: the number of clues in each category, game privacy scope, and game duration. Training data contains 25 games that are randomly chosen, whereas the rest of games are in the test data. The file with edges contains similarity between games. Two versions of this datasets are provided, “Geostep Asymmetric” with asymmetric similarities, and “Geostep Symmetric” with symmetric similarities. The goal is to predict probability that the game can be used for touristic purposes. In this case study the “*Train and test on network*” function is used to train the DirGCRF method on the “Geostep Asymmetric” dataset. Since this dataset has an asymmetric similarity matrix DirGCRF is the only method that can be applied. A neural network is used as the unstructured predictor. Also, an option “Apply standard GCRF” is selected, which means that the similarity matrix will be automatically converted from asymmetric to symmetric in order to apply standard GCRF and get its accuracy for comparison. After the “TRAIN and TEST” button is clicked the training and testing process will start and execution time and R^2 on the training and testing data will be displayed. An example of this setup and results is presented in Figure 31. This model is named as “1”, and that name cannot be used again for the same method. The calculated values of parameters for the chosen method and unstructured predictor, the values predicted by unstructured predictor and the predicted values will be stored in the local file system.

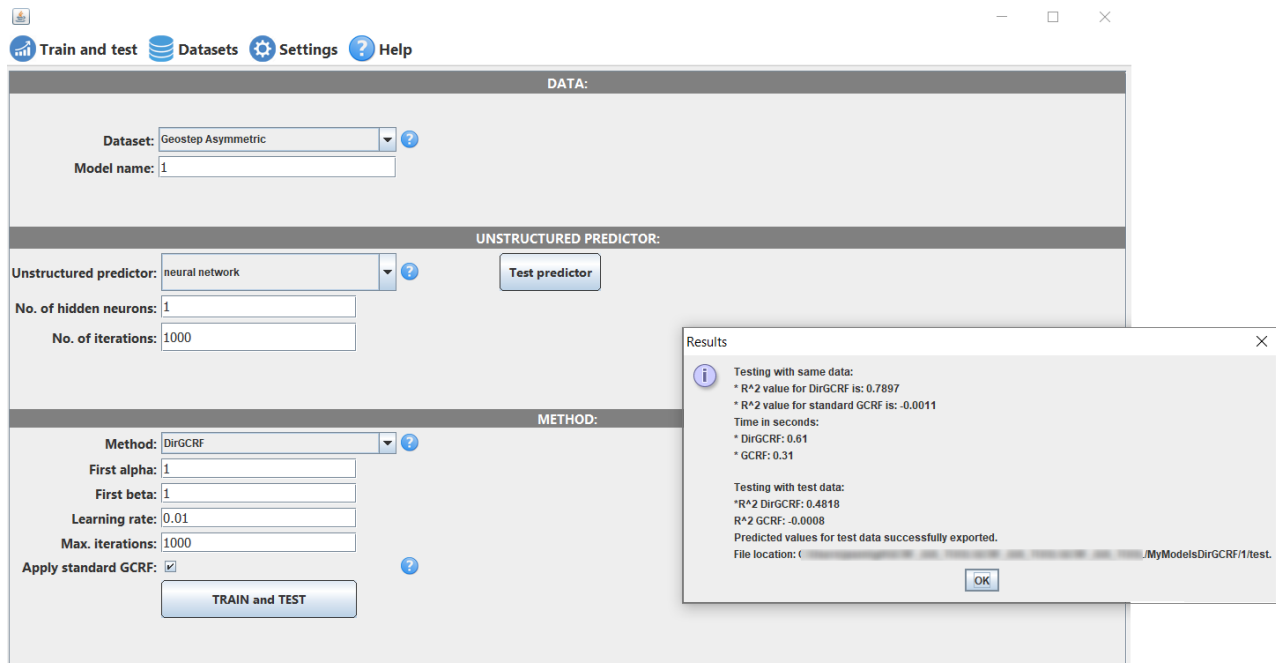


Figure 31. Setup and results for Case study 1

- Case study 2:** The Energy dataset consists of solar energy forecasting in Oklahoma Mesonet sites. The original dataset contains 15 attributes for 98 sites measured in 1600 time points. For the dataset that is included in the GCRFsTOOL we have randomly selected 10 sites; for each site we have only one attribute for all 1600 time points. The file with edges is not provided, which means that the method should learn similarities between sites. The goal is to predict the daily solar energy income at those sites. In this case study the “Train and test on temporal networks” function is used to train and test the RLSR on the “Energy” dataset. The main difference from the previous function is that in this one the test will be completed automatically and that the user should choose how to split the training and test data. We are using 1000 time points for the training process, 300 for validation, and 300 for testing. For all other parameters default values are used. Since this dataset does not contain similarity information, the “Learn similarity” option is automatically checked. This model is named as “2”, and that name cannot be used again for the same method. When the *TRAIN and TEST* button is clicked, Matlab will be called. When Matlab completes all calculations, the results from training and testing phase will be provided together. An example of this setup is presented in Figure 32.

The screenshot shows the GCRFsTOOL software interface. The top menu bar includes 'Train and test', 'Datasets', 'Settings', and 'Help'. The main window is divided into two sections: 'DATA' and 'METHOD'.

DATA Section:

- Dataset: Energy RLSR
- ☒ Learn similarity
- No. of time points: 1600
- No. of time points for train: 1000
- Model name: 2

METHOD Section:

- Method: RLSR
- No. of time points for validation: 300
- Lambda set: 0.01
- No. of time points for test: 300
- No. of hidden neurons for NN: 20
- Max. iterations: 50
- No. of iterations for NN: 200
- LF size: 5
- SSE max. iterations: 1000
- SSE LS max. iterations: 1000

A 'TRAIN and TEST' button is located at the bottom of the METHOD section.

Results Dialog Box:

- RLSR results:
- * MSE: 0.218
- * Average R^2 value: -0.9851
- * Results are successfully exported.
- * File location: MyModelsRLSR/2/test
- * Time in seconds: 21.79

An 'OK' button is located at the bottom of the Results dialog.

Figure 32. Setup and results for Case study 1