

Implement a New Neural Network Data Point

The BRAPH 2 Developers

October 15, 2024

This is the developer tutorial for implementing a new neural network data point. In this Tutorial, you will learn how to create the generator file *.gen.m for a new neural network data point, which can then be compiled by braph2genesis. All kinds of neural network data point are (direct or indirect) extensions of the base element NNDataPoint. Here, you will use as examples the neural network data point NNDataPoint_CON_REG (connectivity data for regression), NNDataPoint_CON_CLA (connectivity data for classification), NNDataPoint_Graph_REG (adjacency matrix for regression), NNDataPoint_Graph_CLA (adjacency matrix for classification), NNDataPoint_Measure_REG (graph measure for regression), and NNDataPoint_Measure_CLA (graph measure for classification).

Contents

<i>Implementation of a data point with connectivity data</i>	<i>2</i>
<i>Connectivity data point for regression (NNDataPoint_CON_REG)</i>	<i>2</i>
<i>Connectivity data point for classification (NNDataPoint_CON_CLA)</i>	<i>6</i>
<i>Implementation of a data point with a graph</i>	<i>11</i>
<i>Graph data point for regression (NNDataPoint_Graph_REG)</i>	<i>11</i>
<i>Graph data point for classification (NNDataPoint_Graph_CLA)</i>	<i>17</i>
<i>Implementation of a data point with graph measures</i>	<i>23</i>
<i>Graph measure data point for regression (NNDataPoint_Measure_REG)</i>	<i>23</i>
<i>Graph measure data point for classification (NNDataPoint_Measure_CLA)</i>	<i>28</i>

Implementation of a data point with connectivity data

Connectivity data point for regression (NNDataPoint_CON_REG)

You will start by implementing in detail NNDataPoint_CON_REG, which is a direct extension of NNDataPoint. A NNDataPoint_CON_REG contains the input and target for neural network analysis with a subject with connectivity data (SubjectCON), where the input is the subject's connectivity data and the target are the subject's variables of interest.

Code 1: NNDataPoint_CON_REG element header. The header section of the generator code in _NNDataPoint_CON_REG.gen.m provides the general information about the NNDataPoint_CON_REG element.

```

1 %% iheader!
2 NNDataPoint_CON_REG < NNDataPoint (dp, connectivity regression data point)
   is a data point for regression with connectivity data.
3
4
5 %% idescription!
6 A data point for regression with connectivity data (NNDataPoint_CON_REG)
7 contains the input and target for neural network analysis with a subject
   with connectivity data (SubjectCON).
8 The input is the connectivity data of the subject.
9 The target is obtained from the variables of interest of the subject.
10
11 %% ibuild!
12 1

```

① defines NNDataPoint_CON_REG as a subclass of NNDataPoint. The moniker will be dp.

Code 2: NNDataPoint_CON_REG element props update. The props_update section of the generator code in _NNDataPoint_CON_REG.gen.m updates the properties of the NNDataPoint element. This defines the core properties of the data point.

```

1 %% iprops_update!
2
3 %% iprop!
4 NAME (constant, string) is the name of a data point for regression with
   connectivity data.
5 %%% idefault!
6 'NNDataPoint_CON_REG'
7
8 %% iprop!
9 DESCRIPTION (constant, string) is the description of a data point for
   regression with connectivity data.
10 %%% idefault!
11 'A data point for regression with connectivity data (NNDataPoint_CON_REG)
   contains the input and target for neural network analysis with a
   subject with connectivity data (SubjectCON). The input is the
   connectivity data of the subject. The target is obtained from the
   variables of interest of the subject.'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of a data point for regression
   with connectivity data.

```

```

15 %%%% isettings!
16 'NNDataPoint_CON_REG'
17
18 %%%% iprop!
19 ID (data, string) is a few-letter code for a data point for regression with
    connectivity data.
20 %%%% idefault!
21 'NNDataPoint_CON_REG ID'
22
23 %%%% iprop!
24 LABEL (metadata, string) is an extended label of a data point for regression
    with connectivity data.
25 %%%% idefault!
26 'NNDataPoint_CON_REG label'
27
28 %%%% iprop!
29 NOTES (metadata, string) are some specific notes about a data point for
    regression with connectivity data.
30 %%%% idefault!
31 'NNDataPoint_CON_REG notes'
32
33 %%%% iprop! ①
34 INPUT (result, cell) is the input value for this data point.
35 %%%% icalculate!
36 value = {dp.get('SUB').get('CON')}; ②
37
38 %%%% iprop! ③
39 TARGET (result, cell) is the target value for this data point.
40 %%%% icalculate!
41 value = cellfun(@(x) dp.get('SUB').get('VOI_DICT').get('IT', x).get('V'), dp
    .get('TARGET_IDS'), 'UniformOutput', false); ④

```

① The property INPUT is the input value for this data point, which is obtained directly from the connectivity data of Subject_CON ②.

③ The property TARGET is the target value for this data point, which is obtained directly from the variables of interest ④.

Code 3: **NNDataPoint_CON_REG element props.** The props section of generator code in `_NNDataPoint_CON_REG.gen.m` defines the properties to be used in `NNDataPoint_CON_REG`.

```

1 %%%% iprops!
2
3 %%%% iprop! ①
4 SUB (data, item) is a subject with connectivity data.
5 %%%% isettings!
6 'SubjectCON'
7
8 %%%% iprop! ②
9 TARGET_IDS (parameter, stringlist) is a list of variable-of-interest IDs to
    be used as regression targets.

```

① The property SUB is a subject with connectivity data (Subject_CON), which is used to calculate the properties INPUT and TARGET.

② The property TARGET_IDS defines the IDs of target, where the target IDs should be from the subject's variable-of-interest IDs.

Code 4: **NNDataPoint_CON_REG element tests**. The tests section from the element generator `_NNDataPoint_CON_REG.gen.m`. A test for creating example files should be prepared to test the properties of the data point. Furthermore, additional test should be prepared for validating the value of input and target for the data point.

```

1 %% itests!
2
3 %% iexcluded_props! ①
4 [NNDataPoint_CON_REG.SUB]
5
6 %% itest!
7 %%% iname!
8 Example ②
9 %%% icode!
10 create_data_NN_REG_CON_TXT() % only creates files if the example folder
    doesn't already exist
11 create_data_NN_REG_CON_XLS() % only creates files if the example folder
    doesn't already exist
12
13 %% itest!
14 %%% iname! ③
15 Create a NNDataset containing NNDataPoint_CON_REG with simulated data
16 %%% icode!
17 % Load BrainAtlas
18 im_ba = ImporterBrainAtlasXLS( ...
19     'FILE', [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data
    NN REG CON XLS' filesep 'atlas.xlsx'], ...
20     'WAITBAR', true ...
21 );
22
23 ba = im_ba.get('BA');
24
25 % Load Group of SubjectCON
26 im_gr = ImporterGroupSubjectCON_XLS( ...
27     'DIRECTORY', [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example
    data NN REG CON XLS' filesep 'CON_Group_XLS'], ...
28     'BA', ba, ...
29     'WAITBAR', true ...
30 );
31
32 gr = im_gr.get('GR');
33
34 % create an item list of NNDataPoint_CON_REG ④
35 it_list = cellfun(@(x) NNDataPoint_CON_REG(...
36     'ID', x.get('ID'), ...
37     'SUB', x, ...
38     'TARGET_IDS', x.get('VOI_DICT').get('KEYS')), ...
39     gr.get('SUB_DICT').get('IT_LIST'), ...
40     'UniformOutput', false);
41
42 % create a NNDataPoint_CON_REG DICT ⑤
43 dp_list = IndexedDictionary(...
44     'IT_CLASS', 'NNDataPoint_CON_REG', ...
45     'IT_LIST', it_list ...
46 );
47
48 % create a NNDataset containing the NNDataPoint_CON_REG DICT ⑥
49 d = NNDataset( ...

```

① List of properties that are excluded from testing.

② creates the example connectivity data files for regression analysis.

③ validates the data point by using assertions to confirm that the input and target calculated values match the connectivity data and the variables of interest in the example files.

④—⑥ create an item list for the data points, subsequently generates the data point dictionary using the list, and then constructs the neural network dataset containing these data points.

```

50     'DP_CLASS', 'NNDataPoint_CON_REG', ...
51     'DP_DICT', dp_list ...
52 );
53
54 % Check whether the number of inputs matches ⑦
55 assert(length(d.get('INPUTS')) == gr.get('SUB_DICT').get('LENGTH'), ...
56 [BRAPH2.STR ':NNDataPoint_CON_REG:' BRAPH2.FAIL_TEST], ...
57 'NNDataPoint_CON_REG does not construct the dataset correctly. The
   number of the inputs should be the same as the number of imported
   subjects.' ...
58 )
59
60 % Check whether the number of targets matches ⑧
61 assert(length(d.get('TARGETS')) == gr.get('SUB_DICT').get('LENGTH'), ...
62 [BRAPH2.STR ':NNDataPoint_CON_REG:' BRAPH2.FAIL_TEST], ...
63 'NNDataPoint_CON_REG does not construct the dataset correctly. The
   number of the targets should be the same as the number of imported
   subjects.' ...
64 )
65
66 % Check whether the content of input for a single data point matches ⑨
67 for index = 1:1:gr.get('SUB_DICT').get('LENGTH')
68     individual_input = d.get('DP_DICT').get('IT', index).get('INPUT');
69     known_input = {gr.get('SUB_DICT').get('IT', index).get('CON')};
70
71     assert(isequal(individual_input, known_input), ...
72 [BRAPH2.STR ':NNDataPoint_CON_REG:' BRAPH2.FAIL_TEST], ...
73 'NNDataPoint_CON_REG does not construct the dataset correctly. The
   input value is not derived correctly.' ...
74 )
75 end
76
77 %%% itest!
78 %%% iname! ⑩
79 Example training-test regression
80 %%% icode!
81 % ensure the example data is generated
82 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
   REG CON XLS' filesep 'atlas.xlsx'])
83     create_data_NN_REG_CON_XLS() % create example files
84 end
85
86 example_NN_CON_REG

```

⑦ tests the number of inputs from the dataset matches the number of subjects in the group.

⑧ tests the number of targets from the dataset matches the number of subjects in the group.

⑨ tests the value of each input from the data point matches the subject's connectivity data.

⑩ executes the corresponding example scripts to ensure the functionalities.

Connectivity data point for classification (NNDataPoint_CON_CLA)

You can now use NNDataPoint_CON_REG as the basis to implement the NNDataPoint_CON_CLA, which can be used for classification tasks. The parts of the code that are modified are highlighted.

Code 5: NNDataPoint_CON_CLA element header. The header section of the generator code in _NNDataPoint_CON_CLA.gen.m provides the general information about the NNDataPoint_CON_CLA element. ← [Code 1](#)

```

1 %% iheader!
2 NNDataPoint_CON_CLA < NNDataPoint (dp, connectivity classification data
    point) is a data point for classification with connectivity data.
3
4 %% idescription!
5 A data point for classification with connectivity data (NNDataPoint_CON_CLA)
6 contains the input and target for neural network analysis with a subject
    with connectivity data (SubjectCON).
7 The input is the connectivity data of the subject.
8 The target is obtained from the variables of interest of the subject.
9
10 %% ibuild!
11 1

```

Code 6: NNDataPoint_CON_CLA element props update. The props_update section of the generator code in _NNDataPoint_CON_CLA.gen.m updates the properties of the NNDataPoint_CON_CLA element. This defines the core properties of the data point. ← [Code 2](#)

```

1 %% iprops_update!
2
3 %% iprop!
4 NAME (constant, string) is the name of a data point for classification with
    connectivity data.
5 %%% idefault!
6 'NNDataPoint_CON_CLA'
7
8 %% iprop!
9 DESCRIPTION (constant, string) is the description of a data point for
    classification with connectivity data.
10 %%% idefault!
11 'A data point for classification with connectivity data (NNDataPoint_CON_CLA
    ) contains the input and target for neural network analysis with a
    subject with connectivity data (SubjectCON). The input is the
    connectivity data of the subject. The target is obtained from the
    variables of interest of the subject.'
12
13 %% iprop!
14 TEMPLATE (parameter, item) is the template of a data point for
    classification with connectivity data.
15 %%% isettings!
16 'NNDataPoint_CON_CLA'
17
18 %% iprop!
19 ID (data, string) is a few-letter code for a data point for classification
    with connectivity data.

```

```

20 %%% idefault!
21 'NNDataPoint_CON_CLA ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of a data point for
    classification with connectivity data.
25 %%% idefault!
26 'NNDataPoint_CON_CLA label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about a data point for
    classification with connectivity data.
30 %%% idefault!
31 'NNDataPoint_CON_CLA notes'
32
33 %%% iprop!
34 INPUT (result, cell) is the input value for this data point.
35 %%% icalculate!
36 value = {dp.get('SUB').get('CON')};
37
38 %%% iprop! ①
39 TARGET (result, cell) is the target values for this data point.
40 %%% icalculate!
41 value = cellfun(@(c) sum(double(c)), dp.get('TARGET_CLASS'), 'UniformOutput'
    , false);

```

① defines the target value using the data point's label in the form of a string, e.g., 'Group1', contained in a cell array and representing the class of the data point.

Code 7: NNDataPoint_CON_CLA element props. The props section of generator code in `_NNDataPoint_CON_CLA.gen.m` defines the properties to be used in `NNDataPoint_CON_CLA`.

```

1 %%% iprops!
2
3 %%% iprop!
4 SUB (data, item) is a subject with connectivity data.
5 %%% isettings!
6 'SubjectCON'
7
8 %%% iprop!
9 TARGET_CLASS (parameter, stringlist) is a list of variable-of-interest IDs
    to be used as the class targets.

```

Code 8: NNDataPoint_CON_CLA element tests. The tests section from the element generator `_NNDataPoint_CON_CLA.gen.m`. A test for creating example files should be prepared to test the properties of the data point. Furthermore, additional test should be prepared for validating the value of input and target for the data point.

```

1 %%% itests!
2
3 %%% iexcluded_props!
4 [NNDataPoint_CON_CLA.SUB]
5
6 %%% itest!
7 %%% iname!
8 Example
9 %%% icode!
10 create_data_NN_CLA_CON_XLS() % only creates files if the example folder
    doesn't already exist

```

```

11 create_data_NN_CLA_CON_TXT() % only creates files if the example folder
    doesn't already exist
12
13
14 %% itest!
15 %%% iname!
16 Create a NNDataset containing NNDataPoint_CON_CLA with simulated data
17 %%% icode!
18 % Load BrainAtlas
19 im_ba = ImporterBrainAtlasXLS( ...
20     'FILE', [fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data
    NN CLA CON XLS' filesep 'atlas.xlsx'], ...
21     'WAITBAR', true ...
22 );
23
24 ba = im_ba.get('BA');
25
26 % Load Groups of SubjectCON ⑥
27 im_gr1 = ImporterGroupSubjectCON_XLS( ...
28     'DIRECTORY', [fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example
    data NN CLA CON XLS' filesep 'CON_Group_1_XLS'], ...
29     'BA', ba, ...
30     'WAITBAR', true ...
31 );
32
33 gr1 = im_gr1.get('GR');
34
35 im_gr2 = ImporterGroupSubjectCON_XLS( ...
36     'DIRECTORY', [fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example
    data NN CLA CON XLS' filesep 'CON_Group_2_XLS'], ...
37     'BA', ba, ...
38     'WAITBAR', true ...
39 );
40
41 gr2 = im_gr2.get('GR');
42
43 % create item lists of NNDataPoint_CON_CLA ⑦
44 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
45 it_list1 = cellfun(@(x) NNDataPoint_CON_CLA( ...
46     'ID', x.get('ID'), ...
47     'SUB', x, ...
48     'TARGET_CLASS', {group_folder_name}), ...
49     gr1.get('SUB_DICT').get('IT_LIST'), ...
50     'UniformOutput', false);
51
52 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
53 it_list2 = cellfun(@(x) NNDataPoint_CON_CLA( ...
54     'ID', x.get('ID'), ...
55     'SUB', x, ...
56     'TARGET_CLASS', {group_folder_name}), ...
57     gr2.get('SUB_DICT').get('IT_LIST'), ...
58     'UniformOutput', false);
59
60 % create NNDataPoint_CON_CLA DICT items
61 dp_list1 = IndexedDictionary(...
62     'IT_CLASS', 'NNDataPoint_CON_CLA', ...
63     'IT_LIST', it_list1 ...
64 );
65
66 dp_list2 = IndexedDictionary(...
67     'IT_CLASS', 'NNDataPoint_CON_CLA', ...

```

⑥ imports two groups of simulated data.

⑦ creates two datasets for the two groups.


```

68         'IT_LIST', it_list2 ...
69     );
70
71 % create a NNDataSet containing the NNDataPoint_CON_CLA DICT
72 d1 = NNDataSet( ...
73     'DP_CLASS', 'NNDataPoint_CON_CLA', ...
74     'DP_DICT', dp_list1 ...
75 );
76
77 d2 = NNDataSet( ...
78     'DP_CLASS', 'NNDataPoint_CON_CLA', ...
79     'DP_DICT', dp_list2 ...
80 );
81
82 % Check whether the number of inputs matches ⑧
83 assert(length(d1.get('INPUTS')) == gr1.get('SUB_DICT').get('LENGTH'), ...
84     [BRAPH2.STR ':NNDataPoint_CON_CLA:' BRAPH2.FAIL_TEST], ...
85     'NNDataPoint_CON_CLA does not construct the dataset correctly. The
86     number of the inputs should be the same as the number of imported
87     subjects of group 1.' ...
88 );
89
90 assert(length(d2.get('INPUTS')) == gr2.get('SUB_DICT').get('LENGTH'), ...
91     [BRAPH2.STR ':NNDataPoint_CON_CLA:' BRAPH2.FAIL_TEST], ...
92     'NNDataPoint_CON_CLA does not construct the dataset correctly. The
93     number of the inputs should be the same as the number of imported
94     subjects of group 2.' ...
95 );
96
97 % Check whether the number of targets matches ⑨
98 assert(length(d1.get('TARGETS')) == gr1.get('SUB_DICT').get('LENGTH'), ...
99     [BRAPH2.STR ':NNDataPoint_CON_CLA:' BRAPH2.FAIL_TEST], ...
100     'NNDataPoint_CON_CLA does not construct the dataset correctly. The
101     number of the targets should be the same as the number of imported
102     subjects of group 1.' ...
103 );
104
105 % Check whether the content of input for a single data point matches ⑩
106 for index = 1:1:gr1.get('SUB_DICT').get('LENGTH')
107     individual_input = d1.get('DP_DICT').get('IT', index).get('INPUT');
108     known_input = {gr1.get('SUB_DICT').get('IT', index).get('CON')};
109
110     assert(isequal(individual_input, known_input), ...
111         [BRAPH2.STR ':NNDataPoint_CON_CLA:' BRAPH2.FAIL_TEST], ...
112         'NNDataPoint_CON_CLA does not construct the dataset correctly. The
113         input value is not derived correctly.' ...
114     );
115 end
116
117 for index = 1:1:gr2.get('SUB_DICT').get('LENGTH')
118     individual_input = d2.get('DP_DICT').get('IT', index).get('INPUT');
119     known_input = {gr2.get('SUB_DICT').get('IT', index).get('CON')};

```

⑧ tests the number of inputs from the dataset matches the number of subjects in the group.

⑨ tests the number of targets from the dataset matches the number of subjects in the group.

⑩ tests the value of each input from the data point matches the subject's connectivity data.

```

119     assert(isequal(individual_input, known_input), ...
120           [BRAPH2.STR ':NNDataPoint_CON_CLA:' BRAPH2.FAIL_TEST], ...
121           'NNDataPoint_CON_CLA does not construct the dataset correctly. The
            input value is not derived correctly.' ...
122           )
123 end
124
125 %%% itest!
126 %%% iname!
127 Example training-test classification (11)
128 %%% icode!
129 % ensure the example data is generated
130 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
            CLA CON XLS' filesep 'atlas.xlsx'])
131     create_data_NN_CLA_CON_XLS() % create example files
132 end
133
134 example_NN_CON_CLA

```

(11) executes the corresponding example scripts to ensure the functionalities.

Implementation of a data point with a graph

Graph data point for regression (NNDataPoint_Graph_REG)

Now you will implement `NNDataPoint_Graph_REG` based on previous codes `NNDataPoint_CON_REG`. This neural network data point with graphs utilizes the adjacency matrix extracted from the derived graph of the subject. The modified parts of the code are highlighted.

Code 9: NNDataPoint_Graph_REG element header. The header section of the generator code in `_NNDataPoint_Graph_REG.gen.m` provides the general information about the `NNDataPoint_Graph_REG` element.

```

1 %% iheader!
2 NNDataPoint_Graph_REG < NNDataPoint (dp, measure regression) is
  a data point for regression with a graph.
3
4 %% idescription!
5 A data point for regression with a graph (NNDataPoint_Graph_REG)
6 contains both input and target for neural network analysis.
7 The input is the value of the adjacency matrix extracted from the derived
  graph of the subject.
8 The target is obtained from the variables of interest of the subject.
9
10 %% ibuild!
11 1

```

Code 10: NNDataPoint_Graph_REG element props update. The `props_update` section of the generator code in `_NNDataPoint_Graph_REG.gen.m` updates the properties of the `NNDataPoint_Graph_REG` element. This defines the core properties of the data point.

```

1 %% iprops_update!
2
3 %% iprop!
4 NAME (constant, string) is the name of a data point for regression with a
  graph.
5 %%% idefault!
6 'NNDataPoint_Graph_REG'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of a data point for
  regression with a graph.
10 %%% idefault!
11 'A data point for regression with a graph (NNDataPoint_Graph_REG) contains
  both input and target for neural network analysis. The input is the
  value of the adjacency matrix extracted from the derived graph of the
  subject. The target is obtained from the variables of interest of the
  subject.'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of a data point for regression
  with a graph.
15 %%% isettings!

```

```

16 'NNDataPoint_Graph_REG'
17
18 %% iprop!
19 ID (data, string) is a few-letter code for a data point for regression with
    a graph.
20 %%%% idefault!
21 'NNDataPoint_Graph_REG ID'
22
23 %% iprop!
24 LABEL (metadata, string) is an extended label of a data point for regression
    with a graph.
25 %%%% idefault!
26 'NNDataPoint_Graph_REG label'
27
28 %% iprop!
29 NOTES (metadata, string) are some specific notes about a data point for
    regression with a graph.
30 %%%% idefault!
31 'NNDataPoint_Graph_REG notes'
32
33 %% iprop!
34 INPUT (result, cell) is the input value for this data point.
35 %%%% icalculate!
36 value = dp.get('G').get('A'); ①
37
38 %% iprop!
39 TARGET (result, cell) is the target value for this data point.
40 %%%% icalculate!
41 value = cellfun(@(x) dp.get('SUB').get('VOI_DICT').get('IT', x).get('V'), dp
    .get('TARGET_IDS'), 'UniformOutput', false);

```

① extracts the adjacency matrix from a Graph element as the input for this data point. Note that a Graph can be any kind of Graph, including GraphWU, MultigraphBUD, and MultiplexBUT, among others.

Code 11: NNDataPoint_Graph_REG element props. The props section of generator code in `_NNDataPoint_Graph_REG.gen.m` defines the properties to be used in `NNDataPoint_Graph_REG`.

```

1 %% iprops!
2
3 %% iprop! ①
4 G (data, item) is a graph.
5 %%%% isettings!
6 'Graph'
7
8 %% iprop!
9 SUB (data, item) is a subject.
10 %%%% isettings!
11 'Subject'
12
13 %% iprop!
14 TARGET_IDS (parameter, stringlist) is a list of variable-of-interest IDs to
    be used as the class targets.

```

① defines the Graph element which contains its corresponding adjacency matrix.

Code 12: **NNDataPoint_Graph_REG element tests**. The tests section from the element generator `_NNDataPoint_Graph_REG.gen.m`. A test for creating example files should be prepared to test the properties of the data point. Furthermore, additional test should be prepared for validating the value of input and target for the data point.

```

1 %% itests!
2
3 %%% iexcluded_props!
4 [NNDataPoint_Graph_REG.G NNDataPoint_Graph_REG.SUB]
5
6 %%% itest!
7 %%% iname! ①
8 Construct the data point with the adjacency matrix derived from its weighted
   undirected graph (GraphWU)
9 %%% icode!
10 % ensure the example data is generated
11 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
   REG CON XLS' filesep 'atlas.xlsx'])
12     create_data_NN_REG_CON_XLS() % create example files
13 end
14
15 % Load BrainAtlas
16
17 ...
18
19 % Analysis CON WU ②
20 a_WU = AnalyzeEnsemble_CON_WU( ...
21     'GR', gr ...
22 );
23
24 a_WU.memorize('G_DICT'); ③
25
26 % create item lists of NNDataPoint_Graph_REG
27 it_list = cellfun(@(g, sub) ④ NNDataPoint_Graph_REG( ...
28     'ID', sub.get('ID'), ...
29     'G', g, ...
30     'SUB', sub, ...
31     'TARGET_IDS', sub.get('VOI_DICT').get('KEYS')), ...
32     ⑤ a_WU.get('G_DICT').get('IT_LIST'), gr.get('SUB_DICT').get('IT_LIST')
   , ...
33     'UniformOutput', false);
34
35 % create NNDataPoint_Graph_REG DICT items
36 dp_list = IndexedDictionary(...
37     'IT_CLASS', 'NNDataPoint_Graph_REG', ...
38     'IT_LIST', it_list ...
39 );
40
41 % create a NNDataset containing the NNDataPoint_Graph_REG DICT
42 d = NNDataset( ...
43     'DP_CLASS', 'NNDataPoint_Graph_REG', ...
44     'DP_DICT', dp_list ...
45 );
46
47 % Check whether the content of input for a single data point matches ⑥
48 for index = 1:1:gr.get('SUB_DICT').get('LENGTH')
49     individual_input = d.get('DP_DICT').get('IT', index).get('INPUT');
50     known_input = a_WU.get('G_DICT').get('IT', index).get('A');

```

① tests with the GraphWU element which contains weighted undirected adjacency matrix.

② and ③ create the `AnalyzeEnsemble_CON_WU` element and then memorize its graph dictionary `G_DICT`.

④ and ⑤ creates the `NNDataPoint_Graph_REG` element and use the Graph from `G_DICT`.

⑥ tests whether the value of each input from the data point matches the graph's adjacency matrix.

```

51
52     assert(isequal(individual_input, known_input), ...
53         [BRAPH2.STR 'NNDataPoint_Graph_REG:' BRAPH2.FAIL_TEST], ...
54         'NNDataPoint_Graph_REG does not construct the dataset correctly. The
           input value is not derived correctly.' ...
55     )
56 end
57
58 %%% itest!
59 %%% iname! ⑦
60 Construct the data point with the adjacency matrix derived from its binary
    undirected multigraph with fixed densities (MultigraphBUD)
61 %%% icode!
62 % ensure the example data is generated
63 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
    REG CON XLS' filesep 'atlas.xlsx'])
64     create_data_NN_REG_CON_XLS() % create example files
65 end
66
67 % Load BrainAtlas
68
69 ...
70
71 % Analysis CON WU
72 densities = 0:25:100;
73
74 a_BUD = ⑧ AnalyzeEnsemble_CON_BUD( ...
75     'DENSITIES', densities, ...
76     'GR', gr ...
77 );
78
79 a_BUD.memorize('G_DICT');
80
81 % create item lists of NNDataPoint_Graph_REG
82 it_list = cellfun(@(g, sub) ⑨ NNDataPoint_Graph_REG( ...
83     'ID', sub.get('ID'), ...
84     'G', g, ...
85     'SUB', sub, ...
86     'TARGET_IDS', sub.get('VOI_DICT').get('KEYS')), ...
87     ⑩ a_BUD.get('G_DICT').get('IT_LIST'), gr.get('SUB_DICT').get('IT_LIST'
88     ), ...
89     'UniformOutput', false);
89
90 % create NNDataPoint_Graph_REG DICT items
91 dp_list = IndexedDictionary(...
92     'IT_CLASS', 'NNDataPoint_Graph_REG', ...
93     'IT_LIST', it_list ...
94 );
95
96 ...
97
98 %%% itest!
99 %%% iname! ⑪
100 Construct the data point with the adjacency matrix derived from its
    multiplex weighted undirected graph (MultiplexWU)
101 %%% icode!
102 % ensure the example data is generated
103 if ~isfile([fileparts(which('SubjectCON_FUN_MP')) filesep 'Example data
    CON_FUN_MP XLS' filesep 'atlas.xlsx'])

```

⑦ tests with the MultigraphBUD element which contains the adjacency matrix of binary undirected graph at fixed densities.

⑧—⑩ create the NNDataPoint_Graph_REG element and use the Graph from AnalyzeEnsemble_CON_BUD.

⑪ tests with the MultiplexWU element which contains the adjacency matrix of weighted undirected multiplex.

```

104     create_data_NN_REG_CON_FUN_MP_XLS() % create example files
105 end
106
107 % Load BrainAtlas
108
109 ...
110
111 % Analysis CON FUN MP WU
112 a_WU = (12)AnalyzeEnsemble_CON_FUN_MP_WU( ...
113     'GR', gr ...
114 );
115
116 a_WU.memorize('G_DICT');
117
118 % create item lists of NNDataPoint_Graph_REG
119 it_list = cellfun(@(g, sub) (13)NNDataPoint_Graph_REG( ...
120     'ID', sub.get('ID'), ...
121     'G', g, ...
122     'SUB', sub, ...
123     'TARGET_IDS', sub.get('VOI_DICT').get('KEYS')), ...
124     (14)a_WU.get('G_DICT').get('IT_LIST'), gr.get('SUB_DICT').get('IT_LIST')
125     , ...
126     'UniformOutput', false);
127
128 % create NNDataPoint_Graph_REG DICT items
129 dp_list = IndexedDictionary(...
130     'IT_CLASS', 'NNDataPoint_Graph_REG', ...
131     'IT_LIST', it_list ...
132 );
133 ...
134
135 %% itest!
136 %% iname! (15)
137 Example script for binary undirected graph (MultigraphBUT) using
138     connectivity data
139
140 %% icode!
141 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
142     REG CON XLS' filesep 'atlas.xlsx'])
143     create_data_NN_REG_CON_XLS() % create example files
144 end
145 example_NNCV_CON_BUT_REG
146
147 %% itest!
148 %% iname! (16)
149 Example script for binary undirected multiplex at fixed densities (
150     MultiplexBUD) using connectivity data and functional data
151
152 %% icode!
153 if ~isfile([fileparts(which('NNDataPoint_CON_FUN_MP_REG')) filesep 'Example
154     data NN REG CON_FUN_MP XLS' filesep 'atlas.xlsx'])
155     create_data_NN_REG_CON_FUN_MP_XLS() % create example files
156 end
157 example_NNCV_CON_FUN_MP_BUD_REG
158
159 %% itest!
160 %% iname! (17)
161 Example script for binary undirected multiplex at fixed thresholds (
162     MultiplexBUT) using connectivity data and functional data

```

(12)–(14) creates the
NNDataPoint_Graph_REG
element and use the Graph from
AnalyzeEnsemble_CON_BUD.

(15) tests with the MultigraphBUT
element with the simulated connectivity
data.

(16) tests with the MultiplexBUD
element with the simulated connectivity
and functional data.

(17) tests with the MultiplexBUT
element with the simulated connectivity
and functional data.

```
156 %%%% icode!
157 if ~isfile([fileparts(which('NNDataPoint_CON_FUN_MP_REG')) filesep 'Example
    data NN REG CON_FUN_MP XLS' filesep 'atlas.xlsx'])
158     create_data_NN_REG_CON_FUN_MP_XLS % create example files
159 end
160 example_NNCV_CON_FUN_MP_BUT_REG
```

Graph data point for classification (NNDataPoint_Graph_CLA)

Now we implement NNDataPoint_Graph_CLA based on previous codes NNDataPoint_CON_CLA. This neural network data point with graphs utilizes the adjacency matrix extracted from the derived graph of the subject. The modified parts of the code are highlighted.

Code 13: NNDataPoint_Graph_CLA element header. The header section of the generator code in _NNDataPoint_Graph_CLA.gen.m provides the general information about the NNDataPoint_Graph_CLA element.

```

1 %% iheader!
2 NNDataPoint_Graph_CLA < NNDataPoint (dp, graph classification data point) is
    a data point for classification with a graph.
3
4 %%% idescription!
5 A data point for classification with a graph (NNDataPoint_Graph_CLA)
6 contains both input and target for neural network analysis.
7 The input is the value of the adjacency matrix extracted from the derived
    graph of the subject.
8 The target is obtained from the variables of interest of the subject.

```

Code 14: NNDataPoint_Graph_CLA element props update. The props_update section of the generator code in _NNDataPoint_Graph_CLA.gen.m updates the properties of the NNDataPoint_Graph_CLA element. This defines the core properties of the data point.

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of a data point for classification with
    a graph.
5 %%% idefault!
6 'NNDataPoint_Graph_CLA'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of a data point for
    classification with a graph.
10 %%% idefault!
11 'A data point for classification with a graph (NNDataPoint_Graph_CLA)
    contains both input and target for neural network analysis. The input
    is the value of the adjacency matrix extracted from the derived graph
    of the subject. The target is obtained from the variables of interest
    of the subject.'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of a data point for
    classification with a graph.
15 %%% isettings!
16 'NNDataPoint_Graph_CLA'
17
18 %%% iprop!
19 ID (data, string) is a few-letter code for a data point for classification
    with a graph.
20 %%% idefault!

```

```

21 'NNDataPoint_Graph_CLA ID'
22
23 %% iprop!
24 LABEL (metadata, string) is an extended label of a data point for
    classification with a graph.
25 %%%% idefault!
26 'NNDataPoint_Graph_CLA label'
27
28 %% iprop!
29 NOTES (metadata, string) are some specific notes about a data point for
    classification with a graph.
30 %%%% idefault!
31 'NNDataPoint_Graph_CLA notes'
32
33 %% iprop!
34 INPUT (result, cell) is the input value for this data point.
35 %%%% icalculate!
36 value = dp.get('G').get('A'); ①
37
38 %% iprop!
39 TARGET (result, cell) is the target value for this data point.
40 %%%% icalculate!
41 value = cellfun(@(c) sum(double(c)), dp.get('TARGET_CLASS'), 'UniformOutput
    ', false);

```

① extracts the adjacency matrix from a Graph element as the input for this data point. Note that a Graph can be any kind of Graph, including GraphWU, MultigraphBUD, and MultiplexBUT, among others.

Code 15: NNDataPoint_Graph_CLA element props. The props section of generator code in `_NNDataPoint_Graph_CLA.gen.m` defines the properties to be used in `NNDataPoint_Graph_CLA`.

```

1 %% iprops!
2
3 %% iprop! ①
4 G (data, item) is a graph.
5 %%%% isettings!
6 'Graph'
7
8 %% iprop!
9 TARGET_CLASS (parameter, stringlist) is a list of variable-of-interest IDs
    to be used as the class targets.

```

① defines the Graph element which contains its corresponding adjacency matrix.

Code 16: **NNDataPoint_Graph_CLA element tests**. The tests section from the element generator `_NNDataPoint_Graph_CLA.gen.m`. A test for creating example files should be prepared to test the properties of the data point. Furthermore, additional test should be prepared for validating the value of input and target for the data point.

```

1 %% itests!
2
3 %%% iexcluded_props!
4 [NNDataPoint_Graph_CLA.G]
5
6 %%% itest!
7 %%% iname! ①
8 Construct the data point with the adjacency matrix derived from its weighted
   undirected graph (GraphWU)
9 %%% icode!
10 % ensure the example data is generated
11 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
   CLA CON XLS' filesep 'atlas.xlsx'])
12     create_data_NN_CLA_CON_XLS() % create example files
13 end
14
15 % Load BrainAtlas
16
17 ...
18
19 % Analysis CON WU
20 a_WU1 = ②AnalyzeEnsemble_CON_WU( ...
21     'GR', gr1 ...
22 );
23
24 a_WU2 = AnalyzeEnsemble_CON_WU( ...
25     'TEMPLATE', a_WU1, ...
26     'GR', gr2 ...
27 );
28
29 a_WU1.memorize('G_DICT');
30 a_WU2.memorize('G_DICT');
31
32 % create item lists of NNDataPoint_Graph_CLA
33 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
34 it_list1 = cellfun(@(x) ③NNDataPoint_Graph_CLA( ...
35     'ID', x.get('ID'), ...
36     'G', x, ...
37     'TARGET_CLASS', {group_folder_name}), ...
38     ④a_WU1.get('G_DICT').get('IT_LIST'), ...
39     'UniformOutput', false);
40
41 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
42 it_list2 = cellfun(@(x) NNDataPoint_Graph_CLA( ...
43     'ID', x.get('ID'), ...
44     'G', x, ...
45     'TARGET_CLASS', {group_folder_name}), ...
46     a_WU2.get('G_DICT').get('IT_LIST'), ...
47     'UniformOutput', false);
48
49 % create NNDataPoint_Graph_CLA DICT items
50
51 ...

```

① tests with the GraphWU element which contains weighted undirected adjacency matrix.

②—④ create Analyzensemble_CON_WU and use its G_DICT to initialize NNDataPoint_Graph_CLA.

```

52
53 %%% itest!
54 %%% iname! ⑤
55 Construct the data point with the adjacency matrix derived from its binary
    undirected multigraph with fixed densities (MultigraphBUD)
56 %%% icode!
57 % ensure the example data is generated
58 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
    CLA CON XLS' filesep 'atlas.xlsx'])
59     create_data_NN_CLA_CON_XLS() % create example files
60 end
61
62 % Load BrainAtlas
63
64 ...
65
66 % Analysis CON WU
67 densities = 0:25:100;
68
69 a_BUD1 = ⑥AnalyzeEnsemble_CON_BUD( ...
70     'DENSITIES', densities, ...
71     'GR', gr1 ...
72 );
73
74 a_BUD2 = AnalyzeEnsemble_CON_BUD( ...
75     'TEMPLATE', a_BUD1, ...
76     'GR', gr2 ...
77 );
78
79 a_BUD1.memorize('G_DICT');
80 a_BUD2.memorize('G_DICT');
81
82 % create item lists of NNDataPoint_Graph_CLA
83 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
84 it_list1 = cellfun(@(x) ⑦NNDataPoint_Graph_CLA( ...
85     'ID', x.get('ID'), ...
86     'G', x, ...
87     'TARGET_CLASS', {group_folder_name}), ...
88     ⑧a_BUD1.get('G_DICT').get('IT_LIST'), ...
89     'UniformOutput', false);
90
91 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
92 it_list2 = cellfun(@(x) NNDataPoint_Graph_CLA( ...
93     'ID', x.get('ID'), ...
94     'G', x, ...
95     'TARGET_CLASS', {group_folder_name}), ...
96     a_BUD2.get('G_DICT').get('IT_LIST'), ...
97     'UniformOutput', false);
98
99 % create NNDataPoint_Graph_CLA DICT items
100 dp_list1 = IndexedDictionary(...
101     'IT_CLASS', 'NNDataPoint_Graph_CLA', ...
102     'IT_LIST', it_list1 ...
103 );
104
105 dp_list2 = IndexedDictionary(...
106     'IT_CLASS', 'NNDataPoint_Graph_CLA', ...
107     'IT_LIST', it_list2 ...
108 );
109

```

⑤ tests with the MultigraphBUD element which contains binary undirected adjacency matrix at fixed densities.

⑥—⑧ create Analyzensemble_CON_BUD and use its G_DICT to initialize NNDataPoint_Graph_CLA.

```

110 % create a NNdataset containing the NNDataPoint_Graph_CLA DICT
111
112 ...
113
114 %% itest!
115 %%% iname! ⑨
116 Construct the data point with the adjacency matrix derived from its
      multiplex weighted undirected graph (MultiplexWU)
117 %%% icode!
118 % ensure the example data is generated
119 if ~isfile([fileparts(which('SubjectCON_FUN_MP')) filesep 'Example data
      CON_FUN_MP_XLS' filesep 'atlas.xlsx'])
120     create_data_NN_CLA_CON_FUN_MP_XLS() % create example files
121 end
122
123 % Load BrainAtlas
124
125 ...
126
127 % Analysis CON FUN MP WU
128 a_WU1 = ⑩AnalyzeEnsemble_CON_FUN_MP_WU( ...
129     'GR', gr1 ...
130 );
131
132 a_WU2 = AnalyzeEnsemble_CON_FUN_MP_WU( ...
133     'TEMPLATE', a_WU1, ...
134     'GR', gr2 ...
135 );
136
137 a_WU1.memorize('G_DICT');
138 a_WU2.memorize('G_DICT');
139
140 % create item lists of NNDataPoint_Graph_CLA
141 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
142 it_list1 = cellfun(@(x) ⑪NNDataPoint_Graph_CLA( ...
143     'ID', x.get('ID'), ...
144     'G', x, ...
145     'TARGET_CLASS', {group_folder_name}), ...
146     ⑫a_WU1.get('G_DICT').get('IT_LIST'), ...
147     'UniformOutput', false);
148
149 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
150 it_list2 = cellfun(@(x) NNDataPoint_Graph_CLA( ...
151     'ID', x.get('ID'), ...
152     'G', x, ...
153     'TARGET_CLASS', {group_folder_name}), ...
154     a_WU2.get('G_DICT').get('IT_LIST'), ...
155     'UniformOutput', false);
156
157 % create NNDataPoint_Graph_CLA DICT items
158
159 ...
160
161 %% itest!
162 %%% iname! ⑬
163 Example script for weighted undirected graph (GraphWU) using connectivity
      data
164 %%% icode!
165 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN

```

⑨ tests with the MultiplexWU element which contains weighted undirected adjacency matrix from multiplex graph.

⑩—⑫ create Analyzensemble_CON_FUN_MP_WU and use its G_DICT to initialize NNDataPoint_Graph_CLA.

⑬ tests with the GraphWU element with simulated data.

```

        CLA CON XLS' filesep 'atlas.xlsx'])
166     create_data_NN_CLA_CON_XLS() % create example files
167 end
168 example_NNCV_CON_WU_CLA
169
170 %%% itest!
171 %%% iname! 14
172 Example script for binary undirected graph at fixed densities (MultigraphBUD
    ) using connectivity data
173 %%% icode!
174 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
    CLA CON XLS' filesep 'atlas.xlsx'])
175     create_data_NN_CLA_CON_XLS() % create example files
176 end
177 example_NNCV_CON_BUD_CLA
178
179 %%% itest!
180 %%% iname! 15
181 Example script for weighted undirected multiplex (MultiplexWU) using
    connectivity data and functional data
182 %%% icode!
183 if ~isfile([fileparts(which('NNDataPoint_CON_FUN_MP_CLA')) filesep 'Example
    data NN CLA CON_FUN_MP XLS' filesep 'atlas.xlsx'])
184     create_data_NN_CLA_CON_FUN_MP_XLS() % create example files
185 end
186 example_NNCV_CON_FUN_MP_WU_CLA

```

14 tests with the MultigraphBUD element with simulated data.

15 tests with the MultiplexWU element with simulated data.

Implementation of a data point with graph measures

Graph measure data point for regression (NNDataPoint_Measure_REG)

Now we implement NNDataPoint_Measure_REG based on previous codes NNDataPoint_Graph_REG. This neural network data point utilizes graph measures obtained from the adjacency matrix from the derived graph of the subject. The modified parts of the code are highlighted.

Code 17: NNDataPoint_Measure_REG element

header. The header section of the generator code in `_NNDataPoint_Measure_REG.gen.m` provides the general information about the NNDataPoint_Measure_REG element.

```

1 %% iheader!
2 NNDataPoint_Measure_REG < NNDataPoint (dp, measure regression data point) is
   a data point for regression with graph measures.
3
4 %%% idescription!
5 A data point for regression with graph measures (NNDataPoint_Measure_REG)
6 contains both input and target for neural network analysis.
7 The input is the value of the graph measures (e.g. Degree, DegreeAv, and
   Distance),
8 calculated from the derived graph of the subject.
9 The target is obtained from the variables of interest of the subject.

```

Code 18: NNDataPoint_Measure_REG element props up-

date. The props_update section of the generator code in `_NNDataPoint_Measure_REG.gen.m` updates the properties of the NNDataPoint_Measure_REG element. This defines the core properties of the data point.

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of a data point for regression with
   graph measures.
5 %%%% idefault!
6 'NNDataPoint_Measure_REG'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of a data point for
   regression with graph measures.
10 %%%% idefault!
11 'A data point for regression with graph measures (NNDataPoint_Measure_REG)
   contains both input and target for neural network analysis. The input
   is the value of the graph measures (e.g. Degree, DegreeAv, and Distance
   ), calculated from the derived graph of the subject. The target is
   obtained from the variables of interest of the subject.'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of a data point for regression
   with graph measures.
15 %%%% isettings!
16 'NNDataPoint_Measure_REG'

```

```

17
18 %% iprop!
19 ID (data, string) is a few-letter code for a data point for regression with
    graph measures.
20 %%% idefault!
21 'NNDataPoint_Measure_REG ID'
22
23 %% iprop!
24 LABEL (metadata, string) is an extended label of a data point for regression
    with graph measures.
25 %%% idefault!
26 'NNDataPoint_Measure_REG label'
27
28 %% iprop!
29 NOTES (metadata, string) are some specific notes about a data point for
    regression with graph measures.
30 %%% idefault!
31 'NNDataPoint_Measure_REG notes'
32
33 %% iprop!
34 INPUT (result, cell) is the input value for this data point.
35 %%% icalculate!
36 value = cellfun(@(m_class) dp.get('G').get('MEASURE', m_class).get('M'), dp.
    get('M_LIST'), 'UniformOutput', false); ①
37
38 %% iprop!
39 TARGET (result, cell) is the target value for this data point.
40 %%% icalculate!
41 value = cellfun(@(x) dp.get('SUB').get('VOI_DICT').get('IT', x).get('V'), dp
    .get('TARGET_IDS'), 'UniformOutput', false);

```

① calculates the graph measures, specified with M_LIST, from a Graph element for this data point. Note that a Graph can be any kind of Graph, including GraphWU, MultigraphBUD, and MultiplexBUT, among others.

Code 19: NNDataPoint_Measure_REG element props. The props section of generator code in `_NNDataPoint_Measure_REG.gen.m` defines the properties to be used in `NNDataPoint_Measure_REG`.

```

1 %% iprops!
2
3 %% iprop!
4 G (data, item) is a graph.
5 %%% isettings!
6 'Graph'
7
8 %% iprop! ①
9 M_LIST (parameter, classlist) is a list of graph measure to be used as the
    input
10
11 %% iprop!
12 SUB (data, item) is a subject.
13 %%% isettings!
14 'Subject'
15
16 %% iprop!
17 TARGET_IDS (parameter, stringlist) is a list of variable-of-interest IDs to
    be used as the class targets.

```

① defines the graph measure list which will be obtained as INPUT for this data point.

Code 20: **NNDataPoint_Measure_REG element tests.** The tests section from the element generator `_NNDataPoint_Measure_REG.gen.m`. A test for creating example files should be prepared to test the properties of the data point. Furthermore, additional test should be prepared for validating the value of input and target for the data point.

```

1 %% itests!
2
3 %% iexcluded_props!
4 [NNDataPoint_Measure_REG.G NNDataPoint_Measure_REG.SUB]
5
6 %%% itest!
7 %%% iname!
8 Construct the data point with the adjacency matrix derived from its weighted
   undirected graph (GraphWU)
9 %%% icode!
10 % ensure the example data is generated
11 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
   REG CON XLS' filesep 'atlas.xlsx'])
12     create_data_NN_REG_CON_XLS() % create example files
13 end
14
15 % Load BrainAtlas
16
17 ...
18
19 % Analysis CON WU
20 a_WU = (1)AnalyzeEnsemble_CON_WU( ...
21     'GR', gr ...
22 );
23
24 a_WU.get('MEASUREENSEMBLE', 'Degree').get('M'); (2)
25 a_WU.get('MEASUREENSEMBLE', 'DegreeAv').get('M'); (3)
26 a_WU.get('MEASUREENSEMBLE', 'Distance').get('M'); (4)
27
28 % create item lists of NNDataPoint_Measure_REG
29 it_list = cellfun(@(g, sub) (5)NNDataPoint_Measure_REG( ...
30     'ID', sub.get('ID'), ...
31     'G', g, ...
32     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
33     'SUB', sub, ...
34     'TARGET_IDS', sub.get('VOI_DICT').get('KEYS')), ...
35     (6)a_WU.get('G_DICT').get('IT_LIST'), gr.get('SUB_DICT').get('IT_LIST')
36     , ...
37     'UniformOutput', false);
38
39 % create NNDataPoint_Measure_REG DICT items
40 dp_list = IndexedDictionary(...
41     'IT_CLASS', 'NNDataPoint_Measure_REG', ...
42     'IT_LIST', it_list ...
43 );
44 ...
45
46 %%% itest!
47 %%% iname! (7)
48 Construct the data point with the adjacency matrix derived from its binary
   undirected multigraph with fixed densities (MultigraphBUD)

```

(1)–(4) create a `AnalyzeEnsemble_CON_WU` and add various kinds of graph measure with the `GraphWU` element which contains weighted undirected adjacency matrix.

(5) and (6) use `AnalyzeEnsemble_CON_WU`'s `G_DICT` to set up a `NNDataPoint_Measure_REG`.

(7)–(10) tests various kinds of graph measure with the `MultigraphBUD`.

```

49 %%%% icode!
50 % ensure the example data is generated
51 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
    REG CON XLS' filesep 'atlas.xlsx'])
52     create_data_NN_REG_CON_XLS() % create example files
53 end
54
55 % Load BrainAtlas
56
57 ...
58
59 % Analysis CON WU
60 densities = 0:25:100;
61
62 a_BUD = (8)AnalyzeEnsemble_CON_BUD( ...
63     'DENSITIES', densities, ...
64     'GR', gr ...
65 );
66
67 a_BUD.get('MEASUREENSEMBLE', 'Degree').get('M');
68 a_BUD.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
69 a_BUD.get('MEASUREENSEMBLE', 'Distance').get('M');
70
71 % create item lists of NNDataPoint_Measure_REG
72 it_list = cellfun(@(g, sub) (9)NNDataPoint_Measure_REG( ...
73     'ID', sub.get('ID'), ...
74     'G', g, ...
75     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
76     'SUB', sub, ...
77     'TARGET_IDS', sub.get('VOI_DICT').get('KEYS')), ...
78     (10)a_BUD.get('G_DICT').get('IT_LIST'), gr.get('SUB_DICT').get('IT_LIST'
79     ),...
80     'UniformOutput', false);
81
82 % create NNDataPoint_CON_CLA_DICT items
83 dp_list = IndexedDictionary(...
84     'IT_CLASS', 'NNDataPoint_Measure_REG', ...
85     'IT_LIST', it_list ...
86 );
87 ...
88
89 %%% itest!
90 %%% iname! (11)
91 Construct the data point with the adjacency matrix derived from its
    multiplex weighted undirected graph (MultiplexWU)
92 %%%% icode!
93 % ensure the example data is generated
94 if ~isfile([fileparts(which('SubjectCON_FUN_MP')) filesep 'Example data
    CON_FUN_MP XLS' filesep 'atlas.xlsx'])
95     create_data_NN_REG_CON_FUN_MP_XLS() % create example files
96 end
97
98 % Load BrainAtlas
99
100 ...
101
102 % Analysis CON FUN MP WU
103 a_WU = (12)AnalyzeEnsemble_CON_FUN_MP_WU( ...

```

(11) — (14) tests various kinds of graph measure with the MultiplexWU.

```

104     'GR', gr ...
105 );
106
107 % To be added the multiplex measures
108 a_WU.get('MEASUREENSEMBLE', 'Degree').get('M');
109 a_WU.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
110 a_WU.get('MEASUREENSEMBLE', 'Distance').get('M');
111
112 % create item lists of NNDataPoint_Measure_REG
113 it_list = cellfun(@(g, sub) (13) NNDataPoint_Measure_REG( ...
114     'ID', sub.get('ID'), ...
115     'G', g, ...
116     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
117     'SUB', sub, ...
118     'TARGET_IDS', sub.get('VOI_DICT').get('KEYS')), ...
119     (14) a_WU.get('G_DICT').get('IT_LIST'), gr.get('SUB_DICT').get('IT_LIST')
120     , ...
121     'UniformOutput', false);
122 % create NNDataPoint_Measure_REG DICT items
123
124 ...
125
126 %%% itest!
127 %%% iname! (15)
128 Example script for weighted undirected graph (GraphWU) using connectivity
    data
129 %%% icode!
130 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
    REG CON XLS' filesep 'atlas.xlsx'])
131     create_data_NN_REG_CON_XLS() % create example files
132 end
133 example_NNCV_CON_WU_M_REG
134
135 %%% itest!
136 %%% iname! (16)
137 Example script for weighted undirected multiplex (MultiplexWU) using
    connectivity data and functional data
138 %%% icode!
139 if ~isfile([fileparts(which('NNDataPoint_CON_FUN_MP_REG')) filesep 'Example
    data NN REG CON_FUN_MP XLS' filesep 'atlas.xlsx'])
140     create_data_NN_REG_CON_FUN_MP_XLS() % create example files
141 end
142 example_NNCV_CON_FUN_MP_WU_M_REG

```

(15) tests various kinds of graph measure with the GraphWU using example data.

(16) tests various kinds of graph measure with the MultiplexWU using example data.

Graph measure data point for classification (NNDataPoint_Measure_CLA)

Now we implement NNDataPoint_Measure_CLA based on previous codes NNDataPoint_Graph_CLA. This neural network data point utilizes graph measures obtained from the adjacency matrix from the derived graph of the subject. The modified parts of the code are highlighted.

Code 21: NNDataPoint_Measure_CLA element

header. The header section of the generator code in `_NNDataPoint_Measure_CLA.gen.m` provides the general information about the NNDataPoint_Measure_CLA element.

```

1 %% iheader!
2 NNDataPoint_Measure_CLA < NNDataPoint (dp, measure classification data point
   ) is a data point for classification with graph measures.
3
4 %%% idescription!
5 A data point for classification with graph measures (NNDataPoint_Measure_CLA
   )
6 contains both input and target for neural network analysis.
7 The input is the value of the graph measures (e.g. Degree, DegreeAv, and
   Distance),
8 calculated from the derived graph of the subject.
9 The target is obtained from the variables of interest of the subject.

```

Code 22: NNDataPoint_Measure_CLA element props up-

date. The props_update section of the generator code in `_NNDataPoint_Measure_CLA.gen.m` updates the properties of the NNDataPoint_Measure_CLA element. This defines the core properties of the data point.

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of a data point for classification with
   graph measures.
5 %%% idefault!
6 'NNDataPoint_Measure_CLA'
7
8 %%% iprop!
9 DESCRIPTION (constant, string) is the description of a data point for
   classification with graph measures.
10 %%% idefault!
11 'A data point for classification with graph measures (
   NNDataPoint_Measure_CLA) contains both input and target for neural
   network analysis. The input is the value of the graph measures (e.g.
   Degree, DegreeAv, and Distance), calculated from the derived graph of
   the subject. The target is obtained from the variables of interest of
   the subject.'
12
13 %%% iprop!
14 TEMPLATE (parameter, item) is the template of a data point for
   classification with graph measures.
15 %%% isettings!
16 'NNDataPoint_Measure_CLA'
17

```

```

18 %%% iprop!
19 ID (data, string) is a few-letter code for a data point for classification
    with graph measures.
20 %%% idefault!
21 'NNDataPoint_Measure_CLA ID'
22
23 %%% iprop!
24 LABEL (metadata, string) is an extended label of a data point for
    classification with graph measures.
25 %%% idefault!
26 'NNDataPoint_Measure_CLA label'
27
28 %%% iprop!
29 NOTES (metadata, string) are some specific notes about a data point for
    classification with graph measures.
30 %%% idefault!
31 'NNDataPoint_Measure_CLA notes'
32
33 %%% iprop!
34 INPUT (result, cell) is the input value for this data point.
35 %%% icalculate!
36 value = cellfun(@(m_class) dp.get('G').get('MEASURE', m_class).get('M'), dp.
    get('M_LIST'), 'UniformOutput', false); ①
37
38 %%% iprop!
39 TARGET (result, cell) is the target value for this data point.
40 %%% icalculate!
41 value = cellfun(@(c) sum(double(c)), dp.get('TARGET_CLASS'), 'UniformOutput
    ', false);

```

① calculates or extract the graph measures, which are specified with M_LIST from a Graph element for this data point. Note that a Graph can be any kind of Graph, including GraphWU, MultigraphBUD, and MultiplexBUT, among others.

Code 23: NNDataPoint_Measure_CLA element props. The props section of generator code in `_NNDataPoint_Measure_CLA.gen.m` defines the properties to be used in `NNDataPoint_Measure_CLA`.

```

1 %%% iprops!
2
3 %%% iprop!
4 G (data, item) is a graph.
5 %%% isettings!
6 'Graph'
7
8 %%% iprop! ①
9 M_LIST (parameter, classlist) is a list of graph measure to be used as the
    input
10
11 %%% iprop!
12 TARGET_CLASS (parameter, stringlist) is a list of variable-of-interest IDs
    to be used as the class targets.

```

① defines the graph measure list which will be obtained as INPUT for this data point.

Code 24: **NNDataPoint_Measure_CLA element tests.** The tests section from the element generator `_NNDataPoint_Measure_CLA.gen.m`. A test for creating example files should be prepared to test the properties of the data point. Furthermore, additional test should be prepared for validating the value of input and target for the data point.

```

1 %% itests!
2
3 %%% iexcluded_props!
4 [NNDataPoint_Measure_CLA.G]
5
6 %%% itest!
7 %%% iname!
8 Construct the data point with the graph measures derived from its weighted
   undirected graph (GraphWU)
9 %%% icode!
10 % ensure the example data is generated
11 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
   CLA CON XLS' filesep 'atlas.xlsx'])
12     create_data_NN_CLA_CON_XLS() % create example files
13 end
14
15 % Load BrainAtlas
16
17 ...
18
19 % Analysis CON WU
20 a_WU1 = (1)AnalyzeEnsemble_CON_WU( ...
21     'GR', gr1 ...
22 );
23
24 a_WU2 = AnalyzeEnsemble_CON_WU( ...
25     'TEMPLATE', a_WU1, ...
26     'GR', gr2 ...
27 );
28
29 a_WU1.get('MEASUREENSEMBLE', 'Degree').get('M'); (2)
30 a_WU1.get('MEASUREENSEMBLE', 'DegreeAv').get('M'); (3)
31 a_WU1.get('MEASUREENSEMBLE', 'Distance').get('M'); (4)
32
33 a_WU2.get('MEASUREENSEMBLE', 'Degree').get('M');
34 a_WU2.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
35 a_WU2.get('MEASUREENSEMBLE', 'Distance').get('M');
36
37 % create item lists of NNDataPoint_Measure_CLA
38 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
39 it_list1 = cellfun(@(x) (5)NNDataPoint_Measure_CLA( ...
40     'ID', x.get('ID'), ...
41     'G', x, ...
42     'MLIST', {'Degree' 'DegreeAv' 'Distance'}, ...
43     'TARGET_CLASS', {group_folder_name}), ...
44     (6)a_WU1.get('G_DICT').get('IT_LIST'), ...
45     'UniformOutput', false);
46
47 % create NNDataPoint_Measure_CLA DICT items
48
49 ...
50
51 %%% itest!

```

(1)–(4) create a `AnalyzeEnsemble_CON_WU` and add various kinds of graph measure with the `GraphWU` element which contains weighted undirected adjacency matrix. (5) and (6) use `AnalyzeEnsemble_CON_WU`'s `G_DICT` to set up a `NNDataPoint_Measure_CLA`.

```

52 %%%% iname! ⑦
53 Construct the data point with the graph measures derived from its binary
    undirected multigraph with fixed densities (MultigraphBUD)
54 %%%% icode!
55 % ensure the example data is generated
56 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
    CLA CON XLS' filesep 'atlas.xlsx'])
57     create_data_NN_CLA_CON_XLS() % create example files
58 end
59
60 % Load BrainAtlas
61
62 ...
63
64 % Analysis CON WU
65 densities = 0:25:100;
66
67 a_BUD1 = ⑧ AnalyzeEnsemble_CON_BUD( ...
68     'DENSITIES', densities, ...
69     'GR', gr1 ...
70 );
71
72 a_BUD2 = AnalyzeEnsemble_CON_BUD( ...
73     'TEMPLATE', a_BUD1, ...
74     'GR', gr2 ...
75 );
76
77 a_BUD1.get('MEASUREENSEMBLE', 'Degree').get('M'); ⑨
78 a_BUD1.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
79 a_BUD1.get('MEASUREENSEMBLE', 'Distance').get('M');
80
81 a_BUD2.get('MEASUREENSEMBLE', 'Degree').get('M');
82 a_BUD2.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
83 a_BUD2.get('MEASUREENSEMBLE', 'Distance').get('M');
84
85 % create item lists of NNDataPoint_Measure_CLA
86 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
87 it_list1 = cellfun(@(x) ⑩ NNDataPoint_Measure_CLA( ...
88     'ID', x.get('ID'), ...
89     'G', x, ...
90     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
91     'TARGET_CLASS', {group_folder_name}), ...
92     a_BUD1.get('G_DICT').get('IT_LIST'), ...
93     'UniformOutput', false);
94
95 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
96 it_list2 = cellfun(@(x) NNDataPoint_Measure_CLA( ...
97     'ID', x.get('ID'), ...
98     'G', x, ...
99     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
100     'TARGET_CLASS', {group_folder_name}), ...
101     a_BUD2.get('G_DICT').get('IT_LIST'), ...
102     'UniformOutput', false);
103
104 % create NNDataPoint_Measure_CLA items
105 dp_list1 = IndexedDictionary(...
106     'IT_CLASS', 'NNDataPoint_Measure_CLA', ...
107     'IT_LIST', it_list1 ...
108 );
109

```

⑦—⑩ tests various kinds of graph measure with the MultigraphBUD.

```

110 ...
111
112 %%% itest!
113 %%% iname! ⑪
114 Construct the data point with the graph measures derived from its multiplex
    weighted undirected graph (MultiplexWU)
115 %%% icode!
116 % ensure the example data is generated
117 if ~isfile([fileparts(which('SubjectCON_FUN_MP')) filesep 'Example data
    CON_FUN_MP XLS' filesep 'atlas.xlsx'])
118     create_data_NN_CLA_CON_FUN_MP_XLS() % create example files
119 end
120
121 % Load BrainAtlas
122
123 ...
124
125 % Analysis CON FUN MP WU
126 a_WU1 = ⑫ AnalyzeEnsemble_CON_FUN_MP_WU( ...
127     'GR', gr1 ...
128 );
129
130 a_WU2 = AnalyzeEnsemble_CON_FUN_MP_WU( ...
131     'TEMPLATE', a_WU1, ...
132     'GR', gr2 ...
133 );
134
135 % To be added the multiplex measures
136 a_WU1.get('MEASUREENSEMBLE', 'Degree').get('M'); ⑬
137 a_WU1.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
138 a_WU1.get('MEASUREENSEMBLE', 'Distance').get('M');
139
140 a_WU2.get('MEASUREENSEMBLE', 'Degree').get('M');
141 a_WU2.get('MEASUREENSEMBLE', 'DegreeAv').get('M');
142 a_WU2.get('MEASUREENSEMBLE', 'Distance').get('M');
143
144 % create item lists of NNDataPoint_Graph_CLA
145 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
146 it_list1 = cellfun(@(x) ⑭ NNDataPoint_Measure_CLA( ...
147     'ID', x.get('ID'), ...
148     'G', x, ...
149     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
150     'TARGET_CLASS', {group_folder_name}), ...
151     a_WU1.get('G_DICT').get('IT_LIST'), ...
152     'UniformOutput', false);
153
154 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
155 it_list2 = cellfun(@(x) NNDataPoint_Measure_CLA( ...
156     'ID', x.get('ID'), ...
157     'G', x, ...
158     'M_LIST', {'Degree' 'DegreeAv' 'Distance'}, ...
159     'TARGET_CLASS', {group_folder_name}), ...
160     a_WU2.get('G_DICT').get('IT_LIST'), ...
161     'UniformOutput', false);
162
163 % create NNDataPoint_Measure_CLA_DICT items
164 dp_list1 = IndexedDictionary(...
165     'IT_CLASS', 'NNDataPoint_Measure_CLA', ...
166     'IT_LIST', it_list1 ...

```

⑪—⑭ tests various kinds of graph measure with the MultigraphBUD.


```

167     );
168
169     ...
170
171     %% itest!
172     %%% iname! (15)
173     Example script for binary undirected graph at fixed densities (GraphBUD)
        using connectivity data
174     %%% icode!
175     if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
        CLA CON XLS' filesep 'atlas.xlsx'])
176         create_data_NN_CLA_CON_XLS() % create example files
177     end
178     example_NNCV_CON_BUD_M_CLA
179
180     %% itest!
181     %%% iname! (16)
182     Example script for binary undirected graph at fixed thresholds (
        MultigraphBUT) using connectivity data
183     %%% icode!
184     if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
        CLA CON XLS' filesep 'atlas.xlsx'])
185         create_data_NN_CLA_CON_XLS() % create example files
186     end
187     example_NNCV_CON_BUT_M_CLA
188
189     %% itest!
190     %%% iname! (17)
191     Example script for binary undirected graph at fixed densities (MultigraphBUD
        ) using connectivity data
192     %%% icode!
193     if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
        CLA CON XLS' filesep 'atlas.xlsx'])
194         create_data_NN_CLA_CON_XLS() % create example files
195     end
196     example_NNCV_CON_BUD_M_CLA
197
198     %% itest!
199     %%% iname! (18)
200     Example script for binary undirected multiplex at fixed densities (
        MultiplexBUD) using connectivity data and functional data
201     %%% icode!
202     if ~isfile([fileparts(which('NNDataPoint_CON_FUN_MP_CLA')) filesep 'Example
        data NN CLA CON_FUN_MP XLS' filesep 'atlas.xlsx'])
203         create_data_NN_CLA_CON_FUN_MP_XLS() % create example files
204     end
205     example_NNCV_CON_FUN_MP_BUD_M_CLA
206
207     %% itest!
208     %%% iname! (19)
209     Example script for binary undirected multiplex at fixed thresholds (
        MultiplexBUT) using connectivity data and functional data
210     %%% icode!
211     if ~isfile([fileparts(which('NNDataPoint_CON_FUN_MP_CLA')) filesep 'Example
        data NN CLA CON_FUN_MP XLS' filesep 'atlas.xlsx'])
212         create_data_NN_CLA_CON_FUN_MP_XLS % create example files
213     end
214     example_NNCV_CON_FUN_MP_BUT_M_CLA

```

(15) tests various kinds of graph measure with the MultigraphBUD using example connectivity data.

(16) tests various kinds of graph measure with the MultigraphBUT using example data.

(17) tests various kinds of graph measure with the MultigraphBUD using example connectivity data.

(18) tests various kinds of graph measure with the MultiplexBUD using example functional data.

(19) tests various kinds of graph measure with the MultiplexBUT using example functional data.