

Implement a new Neural Network Data Point

The BRAPH 2 Developers

September 13, 2023

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

Contents

<i>Implementation of Connectivity Data for Regression (NNDataPoint_CON_REG)</i>	2
<i>Implementation of Multilayer Graphs</i>	8

Implementation of Connectivity Data for Regression (NNDataPoint_CON_REG)

We will start by implementing in detail `NNDataPoint_CON_REG`, which is a direct extension of `NNDataPoint`. 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`), where the input is the subject's connectivity data and the target is the subject's variables of interest.

Code 1: `NNDataPoint_CON_REG` element header. The header section of the generator code for `_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 ①
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.

```

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

Code 2: `NNDataPoint_CON_REG` element prop update. The `props_update` section of the generator code for `_NNDataPoint_CON_REG.gen.m` updates the properties of the `NNDataPoint_CON_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
   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')};
37
38 %%% iprop! ②
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 The property INPUT is the input value for this data point, which is obtained directly from the connectivity of Subject_CON by the code under icalculate!.

② The property TARGET is the target value for this data point, which is obtained directly from the variables of interest of VOI_DICT by the code under icalculate!.

Code 3: NNDataPoint_CON_REG element props. The props section of generator code for _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 mentioned properties INPUT and TARGET.

② The property TARGET_IDS defines the targets, where the targets 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 Create example files for regression (2)
9 %%% icode!
10 data_dir = [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
    REG CON XLS'];
11 if ~isdir(data_dir)
12     mkdir(data_dir);
13
14 % Brain Atlas (3)
15 im_ba = ImporterBrainAtlasXLS('FILE', 'desikan.atlas.xlsx');
16 ba = im_ba.get('BA');
17 ex_ba = ExporterBrainAtlasXLS( ...
18     'BA', ba, ...
19     'FILE', [data_dir filesep() 'atlas.xlsx'] ...
20 );
21 ex_ba.get('SAVE')
22 N = ba.get('BR_DICT').get('LENGTH');
23
24 % saves RNG
25 rng_settings_ = rng(); rng('default')
26
27 sex_options = {'Female' 'Male'};
28
29 % Group (4)
30 K = 2; % degree (mean node degree is 2K)
31 beta = 0.3; % Rewiring probability
32 gr_name = 'CON_Group_XLS';
33 gr_dir = [data_dir filesep() gr_name];
34 mkdir(gr_dir);
35 vois = [
36     {'Subject ID'} {'Age'} {'Sex'}
37     {} {} cell2str(sex_options)}
38 ];
39 for i = 1:100 % subject number
40     sub_id = ['SubjectCON_' num2str(i)];
41     % create WS graphs with random beta
42     beta(i) = rand(1); (5)
43     h = WattsStrogatz(N, K, beta(i)); % create WS graph (6)
44
45     A = full(adjacency(h)); A(1:length(A)+1:numel(A)) = 0; % extract the
    adjacency matrix
46     r = 0 + (0.5 - 0) * rand(size(A)); diffA = A - r; A(A ~= 0) = diffA(
    A ~= 0); % make the adjacency matrix weighted
47     A = max(A, transpose(A)); % make the adjacency matrix symmetric
48
49     writetable(array2table(A), [gr_dir filesep() sub_id '.xlsx'], '

```

(1) List of properties that are excluded from testing.

(2) creates the example connectivity data files for regression analysis.

(3) creates and exports the brain atlas file to the example directory.

(4) creates one group of subjects with specified degree and rewiring probability configurations.

(5) generates random rewiring probability settings for each subject.

(6) and (10) utilize the provided degree and rewiring probability settings to generate corresponding Watts-Strogatz model graphs.

```

    WriteVariableNames', false) ⑦
50
51     % variables of interest
52     age_upperBound = 80;
53     age_lowerBound = 50;
54     age = age_lowerBound + beta(i)*(age_upperBound - age_lowerBound);
    ⑧
55     vois = [vois; {sub_id, age, sex_options(randi(2)))}];
56 end
57 writetable(table(vois), [data_dir filesep() gr_name '.vois.xlsx'], '
    WriteVariableNames', false) ⑨
58
59 % reset RNG
60 rng(rng_settings_)
61 end
62 %%% itest_functions!
63 function h = WattsStrogatz(N, K, beta) ⑩
64 % H = WattsStrogatz(N,K,beta) returns a Watts-Strogatz model graph with N
65 % nodes, N*K edges, mean node degree 2*K, and rewiring probability beta.
66 %
67 % beta = 0 is a ring lattice, and beta = 1 is a random graph.
68
69 % Connect each node to its K next and previous neighbors. This constructs
70 % indices for a ring lattice.
71 s = repelem((1:N)', 1, K);
72 t = s + repmat(1:K, N, 1);
73 t = mod(t - 1, N) + 1;
74
75 % Rewire the target node of each edge with probability beta
76 for source = 1:N
77     switchEdge = rand(K, 1) < beta;
78
79     newTargets = rand(N, 1);
80     newTargets(source) = 0;
81     newTargets(s(t == source)) = 0;
82     newTargets(t(source, ~switchEdge)) = 0;
83
84     [~, ind] = sort(newTargets, 'descend');
85     t(source, switchEdge) = ind(1:nnz(switchEdge));
86 end
87
88 h = graph(s,t);
89 end
90
91 %%% itest!
92 %%% iname!
93 Create a NNDataSet containing NNDataPoint_CON_REG with simulated data
94 %%% icode!
95 % Load BrainAtlas
96 im_ba = ImporterBrainAtlasXLS( ...
97     'FILE', [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data
    NN REG CON XLS' filesep 'atlas.xlsx'], ...
98     'WAITBAR', true ...
99 );
100
101 ba = im_ba.get('BA');
102
103 % Load Group of SubjectCON
104 im_gr = ImporterGroupSubjectCON_XLS( ...
105     'DIRECTORY', [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example

```

⑦ exports the adjacency matrix of the graph to an Excel file.

⑧ associates the age value with each individual rewiring probability setting.

⑨ exports the variables of interest to an Excel file.

```

        data NN REG CON XLS' filesep 'CON.Group.XLS'], ...
106     'BA', ba, ...
107     'WAITBAR', true ...
108 );
109
110 gr = im_gr.get('GR');
111
112 % create a item list of NNDataPoint_CON_REG
113 it_list = cellfun(@(x) NNDataPoint_CON_REG( ...
114     'ID', x.get('ID'), ...
115     'SUB', x, ...
116     'TARGET_IDS', x.get('VOI_DICT').get('KEYS')), ...
117     gr.get('SUB_DICT').get('IT_LIST'), ...
118     'UniformOutput', false);
119
120 % create a NNDataPoint_CON_REG DICT
121 dp_list = IndexedDictionary(...
122     'IT_CLASS', 'NNDataPoint_CON_REG', ...
123     'IT_LIST', it_list ...
124 );
125
126 % create a NNDataset containing the NNDataPoint_CON_REG DICT
127 d = NNDataset( ...
128     'DP_CLASS', 'NNDataPoint_CON_REG', ...
129     'DP_DICT', dp_list ...
130 );
131
132 % Check whether the number of inputs matches
133 assert(length(d.get('INPUTS')) == gr.get('SUB_DICT').get('LENGTH'), ...
134     [BRAPH2.STR ':NNDataPoint_CON_REG:' BRAPH2.FAIL_TEST], ...
135     'NNDataPoint_CON_REG does not construct the dataset correctly. The
136     number of the inputs should be the same as the number of imported
137     subjects.' ...
138 );
139
140 % Check whether the number of targets matches
141 assert(length(d.get('TARGETS')) == gr.get('SUB_DICT').get('LENGTH'), ...
142     [BRAPH2.STR ':NNDataPoint_CON_REG:' BRAPH2.FAIL_TEST], ...
143     'NNDataPoint_CON_REG does not construct the dataset correctly. The
144     number of the targets should be the same as the number of imported
145     subjects.' ...
146 );
147
148 % Check whether the content of input for a single datapoint matches
149 for index = 1:1:gr.get('SUB_DICT').get('LENGTH')
150     individual_input = d.get('DP_DICT').get('IT', index).get('INPUT');
151     known_input = {gr.get('SUB_DICT').get('IT', index).get('CON')};
152
153     assert(isequal(individual_input, known_input), ...
154         [BRAPH2.STR ':NNDataPoint_CON_REG:' BRAPH2.FAIL_TEST], ...
155         'NNDataPoint_CON_REG does not construct the dataset correctly. The
156         input value is not derived correctly.' ...
157     );
158 end
159
160 %%% itest!
161 %%% iname!
162 %%% icode!
163 % ensure the example data is generated
164 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN

```

```

    REG CON XLS' filesep 'atlas.xlsx'])
161 test_NNDataPoint_CON_REG % create example files
162 end
163
164 example_NN_CON_REG
165
166 %%% itest!
167 %%% iname!
168 Example cross-validation regression
169 %%% icode!
170 % ensure the example data is generated
171 if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
    REG CON XLS' filesep 'atlas.xlsx'])
172 test_NNDataPoint_CON_REG % create example files
173 end
174
175 example_NNCV_CON_REG

```

Implementation of Multilayer Graphs