Implement a new Neural Network Cross Validation

The BRAPH 2 Developers

September 19, 2023

This is the developer tutorial for implementing a new neural network cross validation. In this Tutorial, we will explain how to create the generator file `*.gen.m` for a new neural network cross validation, which can then be compiled by `braph2genesis`. All kinds of neural network cross validation are (direct or indirect) extensions of the base element `NNCrossValidation`. Here, we will use as examples the cross validation for multi-layer perceptron regressors `NNRegressorMLP_CrossValidation` and the cross validation for multi-layer perceptron classifiers `NNClassifierMLP_CrossValidation`.

Contents

Implementation of the Cross Validation for Regressors

We will start by implementing in detail NNRegressorMLP_CrossValidation, which is a direct extension of NNCrossValidation. A cross validation for multi-layer perceptron regressors (NNRegressorMLP_CrossValidation) is a procedure designed for evaluating multi-layer perceptron regressors using cross-validation.

Code 1: **NNRegressorMLP_CrossValidation element header.** The header section of the generator code for _NNRegressorMLP_CrossValidation.gen.m provides the general information about the NNRegressorMLP_CrossValidation element.

```
1  %% iheader!
2  NNRegressorMLP_CrossValidation < NNCrossValidation (nncv, neural network
       cross-validation) is a process for evaluating multi-layer perceptron
       regressors using cross-validation.  (1)
3
4  %%% idescription!
5  A cross validation for multi-layer perceptron regressors (
       NNRegressorMLP_CrossValidation) is a process that facilitates the
       evaluation of multi-layer perceptron regressors using cross-validation.
6   It involves splitting a dataset into multiple subsets (folds), training the
        model on some folds while validating on others, and then repeating the
        process for all combinations of folds.
7   This helps in assessing the generalization performance of the model and
       detecting overfitting.
8
9  To train all the neural networks for all folds, use: nncv.get('TRAIN')
```

(1) defines NNRegressorMLP_CrossValidation.gen as a subclass of NNCrossValidation. The moniker will be nncv.

Code 2: **NNRegressorMLP_CrossValidation element prop update.** The props_update section of the generator code for _NNRegressorMLP_CrossValidation.gen.m updates the properties of the NNRegressorMLP_CrossValidation element. This defines the core properties of the data point.

```
1   %% iprops_update!
2
3   %%% iprop!
4   NAME (constant, string) is the name of the cross-validation.
5   %%%% idefault!
6   'NNRegressorMLP_CrossValidation'
7
8   %%% iprop!
9   DESCRIPTION (constant, string) is the description of the cross-validation.
10  %%%% idefault!
11  'A cross validation for multi-layer perceptron regressors (
       NNRegressorMLP_CrossValidation) is a process that facilitates the
       evaluation of multi-layer perceptron regressors using cross-validation.
        It involves splitting a dataset into multiple subsets (folds),
       training the model on some folds while validating on others, and then
       repeating the process for all combinations of folds. This helps in
       assessing the generalization performance of the model and detecting
       overfitting.'
12
13  %%% iprop!
```

```
14  TEMPLATE (parameter, item) is the template of the nerual cross-validation.
15  %%%% isettings!
16  'NNRegressorMLP_CrossValidation'
17
18  %%% iprop!
19  ID (data, string) is a few-letter code for the cross-validation.
20  %%%% idefault!
21  'NNRegressorMLP_CrossValidation ID'
22
23  %%% iprop!
24  LABEL (metadata, string) is an extended label of the cross-validation.
25  %%%% idefault!
26  'NNRegressorMLP_CrossValidation label'
27
28  %%% iprop!
29  NOTES (metadata, string) are some specific notes about the cross-validation.
30  %%%% idefault!
31  'NNRegressorMLP_CrossValidation notes'
32
33  %%% iprop!
34  NN_TEMPLATE (parameter, item) is the neural network template to set all
         neural network parameters.
35  %%%% isettings!
36  'NNRegressorMLP'  (1)
37
38  %%% iprop!
39  NNEVALUATOR_TEMPLATE (parameter, item) is the neural network evaluator
         template to set all evalutor parameters.
40  %%%% isettings!
41  'NNRegressorMLP_Evaluator'  (2)
42
43  %%% iprop!
44  NN_LIST (result, itemlist) contains the neural network models corresponding
         to k folds.
45  %%%% icalculate!
46  d_list = nncv.get('D_LIST');
47
48  if isempty(d_list)
49      value = {};
50  else
51      for i = 1:length(d_list)  (3)
52          d_training_set{i} = d_list;
53          d_training_set{i}(i) = [];  % Exclude the i-th element
54          d_training_set{i} = NNDatasetCombine('D_LIST', d_training_set{i}).
         get('D');
55      end
56
57      d_training_set = d_training_set';
58
59      if ~isa(nncv.getr('NN_TEMPLATE'), 'NoValue')
60          nn_template = nncv.get('NN_TEMPLATE');  (4)
61      else
62          nn_template = NNRegressorMLP( ...  (5)
63              'EPOCHS', nncv.get('EPOCHS'), ...
64              'BATCH', nncv.get('BATCH'), ...
65              'SHUFFLE', nncv.get('SHUFFLE'), ...
66              'SOLVER', nncv.get('SOLVER'), ...
67              'VERBOSE', nncv.get('VERBOSE'), ...
68              'PLOT_TRAINING', nncv.get('PLOT_TRAINING'));
69      end
```

(1) defines the neural network tempate to be NNRegressorMLP, which will be used to set the parameters for all the neural network regressors for the cross validation.

(2) defines the neural network evaluator tempate to be NNRegressorMLP_Evaluator, which will be used to set the parameters for all the neural network evaluator for the cross validation.

(3) constructs the training sets iteratively by concatenating all of the remaining folds after exlcuding the one as the validation set.

(4) and (5) set the parameters to all the NNRegressorMLP based on either the NN_TEMPLATE or the parameters from this cross validation.

```
70
71      value = cellfun(@(d) NNRegressorMLP( ...  (6)
72          'TEMPLATE', nn_template, 'D', d), ...
73          d_training_set, 'UniformOutput', false);
74  end
75
76  %%% iprop!
77  EVALUATOR_LIST (result, itemlist) contains the evaluators corresponding to k
          folds.
78  %%%% icalculate!
79  %%% iprop!
80  EVALUATOR_LIST (result, itemlist) contains the evaluators corresponding to k
          folds.
81  %%%% icalculate!
82  d_list = nncv.get('D_LIST');
83  nn_list = nncv.get('NN_LIST');
84
85  if ~isa(nncv.getr('NNEVALUATOR_TEMPLATE'), 'NoValue')
86      nne_template = nncv.get('NNEVALUATOR_TEMPLATE');(7)
87  else
88      nne_template = NNRegressorMLP_Evaluator( ...  (8)
89          'P', nncv.get('P'));
90  end
91
92  value = cellfun(@(d, nn) NNRegressorMLP_Evaluator('TEMPLATE', nne_template,
          'D', d, 'NN', nn), ...  (9)
93      d_list, nn_list, 'UniformOutput', false);
```

(6) initializes all of the NNRegressorMLPs.

(7) and (8) set the parameters to all the NNRegressorMLP_Evaluator based on either the NNEVALUATOR_TEMPLATE or the parameters from this cross validation.

(9) initializes all of the NNRegressorMLP_Evaluators.

Code 3: **NNRegressorMLP_CrossValidation element props.** The props section of generator code for _NNRegressorMLP_CrossValidation.gen.m defines the properties to be used in NNRegressorMLP_CrossValidation.

```
1  %% iprops!
2
3  %%% iprop!
4  P (parameter, scalar) is the permutation number.(1)
5  %%%% idefault!
6  1e+2
7  %%%% icheck_prop!
8  check = value > 0 && value == round(value);
9
10 %%% iprop!
11 AV_CORR (result, rvector) provides the metric of the correlation of
          coefficients.  (2)
12 %%%% icalculate!
13 e_list = nncv.get('EVALUATOR_LIST');
14
15 value = cellfun(@(e) e.get('CORR'), ...
16     e_list, 'UniformOutput', false);
17
18 if isempty(value)
19     value = [];
20 else
21     value = mean(cell2mat(value), 1);
22 end
23
24 %%% iprop!
```

(1) defines the number for permuation feature importance.

(2), (3), (4), and (5) calculates the average metrics, including the correlation of coefficients, the coefficient of determination, the mean absolute error, the mean squared error, and the root mean squared error.

```matlab
25   AV_DET (result, rvector) provides the coefficient of determination, a
         measure showing how well the predictions are replicated by the model.
                 ③
26   %%%% ¡calculate!
27   e_list = nncv.get('EVALUATOR_LIST');
28
29   value = cellfun(@(e) e.get('DET'), ...
30       e_list, 'UniformOutput', false);
31
32   if isempty(value)
33       value = [];
34   else
35       value = mean(cell2mat(value), 1);
36   end
37
38   %%% ¡prop!
39   AV_MAE (result, rvector) provides the metric of the mean absolute error.  ④
40   %%%% ¡calculate!
41   e_list = nncv.get('EVALUATOR_LIST');
42
43   value = cellfun(@(e) e.get('MAE'), ...
44       e_list, 'UniformOutput', false);
45
46   if isempty(value)
47       value = [];
48   else
49       value = mean(cell2mat(value), 1);
50   end
51
52   %%% ¡prop!
53   AV_MSE (result, rvector) provides the metric of the mean squared error.  ⑤
54   %%%% ¡calculate!
55   e_list = nncv.get('EVALUATOR_LIST');
56
57   value = cellfun(@(e) e.get('MSE'), ...
58       e_list, 'UniformOutput', false);
59
60   if isempty(value)
61       value = [];
62   else
63       value = mean(cell2mat(value), 1);
64   end
65
66   %%% ¡prop!
67   AV_RMSE (result, rvector) provides the metric of the root mean squared error
         .
68   %%%% ¡calculate!
69   e_list = nncv.get('EVALUATOR_LIST');
70
71   value = cellfun(@(e) e.get('RMSE'), ...
72       e_list, 'UniformOutput', false);
73
74   if isempty(value)
75       value = [];
76   else
77       value = mean(cell2mat(value), 1);
78   end
79
80   %%% ¡prop!
81   AV_FEATURE_IMPORTANCE (result, cell) averages the feature importances across
```

```matlab
        k folds.  ⑥
82  %%%% ¡calculate!
83  e_list = nncv.get('EVALUATOR_LIST');
84
85  all_fi = cellfun(@(e) cell2mat(e.get('FEATURE_IMPORTANCE')), ...
86      e_list, 'UniformOutput', false);
87
88  if isempty(cell2mat(all_fi))
89      value = {};
90  else
91      average_fi = zeros(size(all_fi{1}));
92      for i = 1:numel(all_fi)
93          % Add the current cell contents to the averageCell
94          average_fi = average_fi + all_fi{i};
95      end
96      average_fi = average_fi / numel(all_fi);
97      value = {average_fi};
98  end
```

⑥ calculate the average feature importance.

Code 4: **NNRegressorMLP_CrossValidation element tests.** The `tests` section from the element generator `_NNRegressorMLP_CrossValidation.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.

```matlab
%% !tests!

%%% !test!
%%%% !name!
evaluate a regressor cross-validation with the example data
%%%% !code!
% ensure the example data is generated
if ~isfile([fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data NN
        REG CON XLS' filesep 'atlas.xlsx'])
    test_NNDataPoint_CON_REG % create example files
end

% Load BrainAtlas
im_ba = ImporterBrainAtlasXLS( ...
    'FILE', [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example data
     NN REG CON XLS' filesep 'atlas.xlsx'], ...
    'WAITBAR', true ...
    );

ba = im_ba.get('BA');

% Load Groups of SubjectCON
im_gr = ImporterGroupSubjectCON_XLS( ...
    'DIRECTORY', [fileparts(which('NNDataPoint_CON_REG')) filesep 'Example
     data NN REG CON XLS' filesep 'CON_Group_XLS'], ...
    'BA', ba, ...
    'WAITBAR', true ...
    );

gr = im_gr.get('GR');

% create a item list of NNDataPoint_CON_REG
it_list = cellfun(@(x) NNDataPoint_CON_REG( ...
    'ID', x.get('ID'), ...
    'SUB', x, ...
    'TARGET_IDS', x.get('VOI_DICT').get('KEYS')), ...
    gr.get('SUB_DICT').get('IT_LIST'), ...
    'UniformOutput', false);

% create a NNDataPoint_CON_REG DICT
dp_list = IndexedDictionary(...
        'IT_CLASS', 'NNDataPoint_CON_REG', ...
        'IT_LIST', it_list ...
        );

% create a NNData containing the NNDataPoint_CON_REG DICT
d = NNDataset( ...
    'DP_CLASS', 'NNDataPoint_CON_REG', ...
    'DP_DICT', dp_list ...
    );

kfolds = 3;
nncv = NNRegressorMLP_CrossValidation('KFOLDS', kfolds, 'D', d);
```

```
51
52 nn_list = nncv.get('NN_LIST');
53 assert(length(nn_list) == kfolds, ... (1)
54     [BRAPH2.STR ':NNRegressorMLP_CrossValidation:' BRAPH2.FAIL_TEST], ...
55     'NNRegressorMLP_CrossValidation does not calculate the neural network
         list correctly.' ...
56     )
57 e_list = nncv.get('EVALUATOR_LIST');
58 assert(length(e_list) == kfolds, ...
59     [BRAPH2.STR ':NNRegressorMLP_CrossValidation:' BRAPH2.FAIL_TEST], ...
60     'NNRegressorMLP_CrossValidation does not calculate the evaluator list
         correctly.' ...
61     )
```

(1) checks whether the data, regressors, and evaluators are initialized according to the user-specified number of folds.