

Implement Cross Validation for Neural Network

The BRAPH 2 Developers

October 16, 2024

This is the developer tutorial for implementing a new neural network cross validation. In this tutorial, you will learn 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, you will use as examples `NNRegressorMLP_CrossValidation` (a cross validation for multi-layer perceptron regressors) and `NNClassifierMLP_CrossValidation` (a cross validation for multi-layer perceptron classifiers).

Contents

<i>Cross validation for regressors (NNRegressorMLP_CrossValidation)</i>	<i>2</i>
<i>Cross validation for classifiers (NNClassifierMLP_CrossValidation)</i>	<i>8</i>

Cross validation for regressors (NNRegressorMLP_CrossValidation)

You will start by implementing in detail NNRegressorMLP_CrossValidation, which is a direct extension of NNCrossValidation. NNRegressorMLP_CrossValidation performs 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')
10
11 %%% ibuild!
12 1

```

(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 cross validation.

```

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 neural 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 evaluator 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.get('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'), ...

```

① defines the neural network template to be NNRegressorMLP, which will be used in ⑥ to set the relevant parameters.

② defines the neural network evaluator template to be NNRegressorMLP_Evaluator, which will be used in ⑦ to set the relevant parameters.

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

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

```

68         'PLOT_TRAINING', nncv.get('PLOT_TRAINING'));
69     end
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
78     folds.
79 %%% icalculate!
80 %%% iprop!
81 EVALUATOR_LIST (result, itemlist) contains the evaluators corresponding to k
82     folds.
83 %%% icalculate!
84 d_list = nncv.get('D_LIST');
85 nn_list = nncv.get('NN_LIST');
86
87 if ~isa(nncv.get('NNEVALUATOR_TEMPLATE'), 'NoValue')
88     nne_template = nncv.get('NNEVALUATOR_TEMPLATE'); ⑦
89 else
90     nne_template = NNRegressorMLP_Evaluator( ... ⑧
91         'P', nncv.get('P'));
92 end
93
94 value = cellfun(@(d, nn) NNRegressorMLP_Evaluator('TEMPLATE', nne_template,
95     'D', d, 'NN', nn), ... ⑨
96     d_list, nn_list, 'UniformOutput', false);

```

⑥ initializes all of the NNRegressorMLPs.

⑦ and ⑧ sets the parameters to all the NNRegressorMLP_Evaluators 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  AV_CORR (result, rvector) provides the metric of the correlation of
5      coefficients. ①
6  e_list = nncv.get('EVALUATOR_LIST');
7
8  value = cellfun(@(e) e.get('CORR'), ...
9      e_list, 'UniformOutput', false);
10
11 if isempty(value)
12     value = [];
13 else
14     value = mean(cell2mat(value), 1);
15 end
16
17 %%% iprop!
18 AV_DET (result, rvector) provides the coefficient of determination, a
19     measure showing how well the predictions are replicated by the model.
20     ②
21 %%% icalculate!
22 e_list = nncv.get('EVALUATOR_LIST');

```

①—④ calculate 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.

```

21
22 value = cellfun(@(e) e.get('DET'), ...
23     e_list, 'UniformOutput', false);
24
25 if isempty(value)
26     value = [];
27 else
28     value = mean(cell2mat(value), 1);
29 end
30
31 %%% iprop!
32 AV_MAE (result, rvector) provides the metric of the mean absolute error. ③
33 %%% icalculate!
34 e_list = nncv.get('EVALUATOR_LIST');
35
36 value = cellfun(@(e) e.get('MAE'), ...
37     e_list, 'UniformOutput', false);
38
39 if isempty(value)
40     value = [];
41 else
42     value = mean(cell2mat(value), 1);
43 end
44
45 %%% iprop!
46 AV_MSE (result, rvector) provides the metric of the mean squared error. ④
47 %%% icalculate!
48 e_list = nncv.get('EVALUATOR_LIST');
49
50 value = cellfun(@(e) e.get('MSE'), ...
51     e_list, 'UniformOutput', false);
52
53 if isempty(value)
54     value = [];
55 else
56     value = mean(cell2mat(value), 1);
57 end
58
59 %%% iprop!
60 AV_RMSE (result, rvector) provides the metric of the root mean squared error
61 .
62 %%% icalculate!
63 e_list = nncv.get('EVALUATOR_LIST');
64
65 value = cellfun(@(e) e.get('RMSE'), ...
66     e_list, 'UniformOutput', false);
67
68 if isempty(value)
69     value = [];
70 else
71     value = mean(cell2mat(value), 1);
72 end

```

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.

```

1 %% itests!
2
3 %% itest!
4 %%% iname!
5 evaluate a regressor cross-validation with the example data
6 %%% icode!
7 % ensure the example data is generated
8 if ~isfile([fileparts(which('NNDatapoint_CON_REG')) filesep 'Example data NN
   REG CON XLS' filesep 'atlas.xlsx'])
9     create_data_NN_REG_CON_XLS() % create example files
10 end
11
12 % Load BrainAtlas
13 im_ba = ImporterBrainAtlasXLS( ...
14     'FILE', [fileparts(which('NNDatapoint_CON_REG')) filesep 'Example data
   NN REG CON XLS' filesep 'atlas.xlsx'], ...
15     'WAITBAR', true ...
16 );
17
18 ba = im_ba.get('BA');
19
20 % Load Groups of SubjectCON
21 im_gr = ImporterGroupSubjectCON_XLS( ...
22     'DIRECTORY', [fileparts(which('NNDatapoint_CON_REG')) filesep 'Example
   data NN REG CON XLS' filesep 'CON_Group_XLS'], ...
23     'BA', ba, ...
24     'WAITBAR', true ...
25 );
26
27 gr = im_gr.get('GR');
28
29 % create a item list of NNDatapoint_CON_REG
30 it_list = cellfun(@(x) NNDatapoint_CON_REG( ...
31     'ID', x.get('ID'), ...
32     'SUB', x, ...
33     'TARGET_IDS', x.get('VOI_DICT').get('KEYS')), ...
34     gr.get('SUB_DICT').get('IT_LIST'), ...
35     'UniformOutput', false);
36
37 % create a NNDatapoint_CON_REG DICT
38 dp_list = IndexedDictionary(...
39     'IT_CLASS', 'NNDatapoint_CON_REG', ...
40     'IT_LIST', it_list ...
41 );
42
43 % create a NNdata containing the NNDatapoint_CON_REG DICT
44 d = NNdataset( ...
45     'DP_CLASS', 'NNDatapoint_CON_REG', ...
46     'DP_DICT', dp_list ...
47 );
48
49 kfold = 3;
50 nncv = NNRegressorMLP_CrossValidation('KFOLDS', kfold, 'D', d);
51
52 nn_list = nncv.get('NN_LIST');
53 assert(length(nn_list) == kfold, ... ①
54     [BRAPH2.STR ':NNRegressorMLP_CrossValidation:' BRAPH2.FAIL_TEST], ...
55     'NNRegressorMLP_CrossValidation does not calculate the neural network
   list correctly.' ...
56 )

```

① and ② check whether the data, regressors, and evaluators are initialized according to the user-specified number of folds.

```
57 e_list = nncv.get('EVALUATOR_LIST');
58 assert(length(e_list) == kfold, ... (2)
59     [BRAPH2.STR ':NNRegressorMLP_CrossValidation:' BRAPH2.FAIL_TEST], ...
60     'NNRegressorMLP_CrossValidation does not calculate the evaluator list
61     correctly.' ...
62 )
```

Cross validation for classifiers (NNClassifierMLP_CrossValidation)

We can now use NNRegressorMLP_CrossValidation as the basis to implement the NNClassifierMLP_CrossValidation. The parts of the code that are modified are highlighted. A cross validation for multi-layer perceptron classifier (NNClassifierMLP_CrossValidation) is a procedure designed for evaluating multi-layer perceptron classifiers using cross-validation.

Code 5: NNClassifierMLP_CrossValidation element

header. The header section of the generator code for _NNClassifierMLP_CrossValidation.gen.m provides the general information about the NNClassifierMLP_CrossValidation element.

```

1 %% iheader!
2 NNClassifierMLP_CrossValidation < NNCrossValidation (nncv, neural network
   cross-validation) is a process for evaluating multi-layer perceptron
   classifiers using cross-validation.
3
4 %%% idescription!
5 A cross validation for multi-layer perceptron classifiers (
   NNClassifierMLP_CrossValidation) is a process that facilitates the
   evaluation of multi-layer perceptron classifiers 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')
10
11 %%% ibuild!
12 1

```

Code 6: NNClassifierMLP_CrossValidation element prop

update. The props_update section of the generator code for _NNClassifierMLP_CrossValidation.gen.m updates the properties of the NNClassifierMLP_CrossValidation element. This defines the core properties of the cross validation.

```

1 %% iprops_update!
2
3 %%% iprop!
4 NAME (constant, string) is the name of the cross-validation.
5 %%% idefault!
6 'NNClassifierMLP_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 classifiers (
   NNClassifierMLP_CrossValidation) is a process that facilitates the
   evaluation of multi-layer perceptron classifiers 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 cross-validation.
15 %%%% isettings!
16 'NNClassifierMLP_CrossValidation'
17
18 %% iprop!
19 ID (data, string) is a few-letter code for the cross-validation.
20 %%%% idefault!
21 'NNClassifierMLP_CrossValidation ID'
22
23 %% iprop!
24 LABEL (metadata, string) is an extended label of the cross-validation.
25 %%%% idefault!
26 'NNClassifierMLP_CrossValidation label'
27
28 %% iprop!
29 NOTES (metadata, string) are some specific notes about the cross-validation.
30 %%%% idefault!
31 'NNClassifierMLP_CrossValidation notes'
32
33 %% iprop!
34 NN_TEMPLATE (parameter, item) is the neural network template to set all
    neural network parameters.
35 %%%% isettings!
36 'NNClassifierMLP' ①
37
38 %% iprop!
39 NNEVALUATOR_TEMPLATE (parameter, item) is the neural network evaluator
    template to set all evaluator parameters.
40 %%%% isettings!
41 'NNClassifierMLP_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)
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.get('NN_TEMPLATE'), 'NoValue')
60         nn_template = nncv.get('NN_TEMPLATE');
61     else
62         nn_template = NNClassifierMLP( ...
63             'EPOCHS', nncv.get('EPOCHS'), ...
64             'BATCH', nncv.get('BATCH'), ...
65             'SHUFFLE', nncv.get('SHUFFLE'), ...

```

① and ② define the NN_TEMPLATE as NNClassifierMLP and the NNEVALUATOR_TEMPLATE as NNClassifierMLP_Evaluator.

```

66         'SOLVER', nncv.get('SOLVER'), ...
67         'VERBOSE', nncv.get('VERBOSE'), ...
68         'PLOT_TRAINING', nncv.get('PLOT_TRAINING'));
69     end
70
71     value = cellfun(@(d) NNClassifierMLP( ...
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 d_list = nncv.get('D_LIST');
80 nn_list = nncv.get('NN_LIST');
81
82 if ~isa(nncv.get('NNEVALUATOR_TEMPLATE'), 'NoValue')
83     nne_template = nncv.get('NNEVALUATOR_TEMPLATE');
84 else
85     nne_template = NNClassifierMLP_Evaluator( ...
86         'P', nncv.get('P'));
87 end
88
89 value = cellfun(@(d, nn) NNClassifierMLP_Evaluator('TEMPLATE', nne_template,
    'D', d, 'NN', nn), ...
90     d_list, nn_list, 'UniformOutput', false);

```

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

```

1 %%% iprops!
2 %%% iprop! ①
3 AV_AUC (result, rvector) provides the average value of the area under the
    receiver operating characteristic curve across k folds.
4 %%% icalculate!
5 e_list = nncv.get('EVALUATOR_LIST');
6
7 aucs = cellfun(@(e) e.get('AUC'), ...
8     e_list, 'UniformOutput', false);
9
10 if isempty(aucs)
11     value = [];
12 else
13     value = mean(cell2mat(aucs), 1);
14 end
15
16 %%% iprop! ②
17 AV_MACRO_AUC (result, scalar) provides the metric of the average macro AUC
    value across k folds.
18 %%% icalculate!
19 e_list = nncv.get('EVALUATOR_LIST');
20
21 macro_auc = cellfun(@(e) e.get('MACRO_AUC'), ...
22     e_list, 'UniformOutput', false);
23
24 if isempty(macro_auc)

```

① and ② calculate the average value of the area under the receiver operating characteristic curve across k folds.

```

25     value = 0;
26 else
27     value = mean(cell2mat(macro_aucs), 1);
28 end
29
30 %% iprop! ③
31 C_MATRIX (result, matrix) provides the confusion matrix across k folds.
32 %%% icalculate!
33 e_list = nncv.get('EVALUATOR_LIST');
34
35 c_matrices = cellfun(@(e) e.get('C_MATRIX'), ...
36     e_list, 'UniformOutput', false);
37
38 combined_c_matrix = cellfun(@(x) double(x), c_matrices, 'UniformOutput',
39     false);
40 value = sum(cat(3, combined_c_matrix{:}), 3);
41
42 %% iprop!
43 AV_FEATURE_IMPORTANCE (result, cell) averages the feature importances across
44     k folds.
45 %%% icalculate!
46 e_list = nncv.get('EVALUATOR_LIST');
47
48 all_fi = cellfun(@(e) cell2mat(e.get('FEATURE_IMPORTANCE')), ...
49     e_list, 'UniformOutput', false);
50
51 if isempty(cell2mat(all_fi))
52     value = {};
53 else
54     average_fi = zeros(size(all_fi{1}));
55     for i = 1:numel(all_fi)
56         % Add the current cell contents to the averageCell
57         average_fi = average_fi + all_fi{i};
58     end
59     average_fi = average_fi / numel(all_fi);
60     value = {average_fi};
61 end

```

③ aggregates the confusion matrix across k folds.

Code 8: NNClassifierMLP_CrossValidation element

tests. The tests section from the element generator `_NNClassifierMLP_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.

```

1 %% itests!
2
3 %% itest!
4 %%% iname!
5 evaluate a classifier cross-validation with the example data
6 %%% icode!
7
8 % ensure the example data is generated
9 if ~isfile([fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data NN
   CLA CON XLS' filesep 'atlas.xlsx'])
   create_data_NN_CLA_CON_XLS() % create example files
11 end
12
13 % Load BrainAtlas
14 im_ba = ImporterBrainAtlasXLS( ...
15     'FILE', [fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example data
   NN CLA CON XLS' filesep 'atlas.xlsx'], ...
16     'WAITBAR', true ...
17 );
18
19 ba = im_ba.get('BA');
20
21 % Load Groups of SubjectCON
22 im_gr1 = ImporterGroupSubjectCON_XLS( ...
23     'DIRECTORY', [fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example
   data NN CLA CON XLS' filesep 'CON_Group_1_XLS'], ...
24     'BA', ba, ...
25     'WAITBAR', true ...
26 );
27
28 gr1 = im_gr1.get('GR');
29
30 im_gr2 = ImporterGroupSubjectCON_XLS( ...
31     'DIRECTORY', [fileparts(which('NNDataPoint_CON_CLA')) filesep 'Example
   data NN CLA CON XLS' filesep 'CON_Group_2_XLS'], ...
32     'BA', ba, ...
33     'WAITBAR', true ...
34 );
35
36 gr2 = im_gr2.get('GR');
37
38 % create item lists of NNDataPoint_CON_CLA
39 [~, group_folder_name] = fileparts(im_gr1.get('DIRECTORY'));
40 it_list1 = cellfun(@(x) NNDataPoint_CON_CLA( ...
41     'ID', x.get('ID'), ...
42     'SUB', x, ...
43     'TARGET_IDS', {group_folder_name}), ...
44     gr1.get('SUB_DICT').get('IT_LIST'), ...
45     'UniformOutput', false);
46
47 [~, group_folder_name] = fileparts(im_gr2.get('DIRECTORY'));
48 it_list2 = cellfun(@(x) NNDataPoint_CON_CLA( ...
49     'ID', x.get('ID'), ...

```

```

50     'SUB', x, ...
51     'TARGET_IDS', {group_folder_name}), ...
52     gr2.get('SUB_DICT').get('IT_LIST'), ...
53     'UniformOutput', false);
54
55 % create NNDataPoint_CON_CLA DICT items
56 dp_list1 = IndexedDictionary(...
57     'IT_CLASS', 'NNDataPoint_CON_CLA', ...
58     'IT_LIST', it_list1 ...
59 );
60
61 dp_list2 = IndexedDictionary(...
62     'IT_CLASS', 'NNDataPoint_CON_CLA', ...
63     'IT_LIST', it_list2 ...
64 );
65
66 % create a NNDataset containing the NNDataPoint_CON_CLA DICT
67 d1 = NNDataset( ...
68     'DP_CLASS', 'NNDataPoint_CON_CLA', ...
69     'DP_DICT', dp_list1 ...
70 );
71
72 d2 = NNDataset( ...
73     'DP_CLASS', 'NNDataPoint_CON_CLA', ...
74     'DP_DICT', dp_list2 ...
75 );
76
77 % combine the two datasets
78 d = NNDatasetCombine('D_LIST', {d1, d2}).get('D');
79
80 kfolgs = 7;
81 nncv = NNClassifierMLP_CrossValidation('KFOLDS', kfolgs, 'D', d);
82
83 nn_list = nncv.get('NN_LIST');
84 assert(length(nn_list) == kfolgs, ... ①
85     [BRAPH2.STR 'NNClassifierMLP_CrossValidation:' BRAPH2.FAIL_TEST], ...
86     'NNClassifierMLP_CrossValidation does not calculate the neural network
87     list correctly.' ...
88 );
89 e_list = nncv.get('EVALUATOR_LIST');
90 assert(length(e_list) == kfolgs, ... ②
91     [BRAPH2.STR 'NNClassifierMLP_CrossValidation:' BRAPH2.FAIL_TEST], ...
92     'NNClassifierMLP_CrossValidation does not calculate the evaluator list
93     correctly.' ...
94 );

```

① and ② check whether the data, classifiers, and evaluators are initialized according to the user-specified number of folds.