2017/07/21

Command line tools usage

D1 Introduction

This document explains how to use the Windows command line tools: SDNN_Training.exe and SDNN_Testing.exe. Both tools are located in the LIBSDNN/command line tools folder.

D2 Installing the tools

Move both of the above .exe files into the Windows path folder and install Microsoft Visual C++ Redistributable for Visual Studio 2017 (Microsoft 2017). The project files and sources for both can be found in the LIBSDNN/command line tools/projects and sources folder.

D3 Using the tools

1. Save the parameter file, selective desensitization file, and correlation-matrix file to the current folder (see also **How to describe a parameter file**).
2. To produce an SDNN-model file, train the SDNN using SDNN_Training.exe as follows:

```
C:\Users\**> SDNN_Training.exe “sdnn_parameter_file” “training_sample_file” “comple
tion_condition” “SDNN_model_file”
```

where the following command line arguments are used:

- sdnn_parameter_file
  The name of the SDNN parameter file.
- training_sample_file
  The name of the training sample file. In this file, each training sample is written on a separate line in the form target_value, input1, input2, ... (Figure. D1). In handling the pattern recognition issue, target_value must be described using class number (consecutive integers starting from 0).
- completion_condition
  The training-completion condition. It can be set using one of the following strings:
  - iteration(*n*)
    This tells the SDNN to repeat the training procedure *n* times.
  - rmse(*p*,*m*)
    This tells the SDNN to repeat the training process until the root-mean-square error becomes less than *p*. If the SDNN cannot satisfy this condition after *m* iterations, the training process terminates. This completion condition can only be applied to function approximation issues.
  - accuracy(*a*,*m*)
    This tells the SDNN to repeat the training process until the classification accuracy becomes greater than *a*. If the SDNN cannot satisfy this condition after *m* iterations, the training process terminates. This completion condition can only be applied to pattern recognition issues.
- SDNN_model_file
  The name of the SDNN-model file containing the parameters and synaptic weights of the SDNN; its extension should be .bin.

3. To obtain recognition/estimation results, use SDNN_Testing.exe to recognize/estimate the samples as follows:

```
C:\Users\**> SDNN_Testing.exe “SDNN_model_file” “testing_sample_file” “testing_res
ult_file”
```

where the following command line arguments are used:

- SDNN_model_file
  The name of the SDNN-model file. SDNN-model files can be created using SDNN_Training.exe or imported from other applications.
- testing_sample_file
  The name of the testing sample file. In this file, each testing sample is written on a separate line in the form comment, input1, input2, ... Anything (for example, the sample id) can be written in first column of the testing sample file.
- testing_result_file
  The name of the testing-result file; its extension should be .csv. In the testing result file, the recognition/estimation result for each sample is written on a separate line in the form the comment written in the testing sample file, result.

Examples of parameter, training/testing files, and batch files for Windows can be found in the LIBSDNN/command line tools/example folder. These samples can be used to reproduce, for example, the experimental results of Nonaka et al. (2011) (Figure D1).

```
0.00947575,   0.01,   0.27
0.089394,     0.47,   1
0.000611692,  0.05,   0.09
0.44937,      0.09,   0.87
0.0663242,    0.81,   0.87
0.586195,     0.72,   0.92
…
```

Figure D1. Example training file from Nonaka's study (two-variable function approximation) found in LIBSDNN/command line tools/example.

D4 Notes regarding SDNN inputs and outputs

Both numerical and symbolic inputs can be used in LIBSDNN. Please note that numerical inputs must be normalized within the range of [0, 1] and symbolic inputs must be converted to integers starting from 0.

When applying pattern recognition issues, the recognition results are represented as integers starting from 0.

References

Microsoft, "Visual Studio Downloads," https://www.visualstudio.com/downloads/ (accessed online 2018/5/8)

K. Nonaka, F. Tanaka, and M. Morita. Empirical comparison of feedforward neural network on two-variable function approximation. *IEICE TRANSACTIONS on Information and Systems* (in Japanese), J94(12): 2114-2125, 2011.