School of Computer Science, University of Nottingham Malaysia Campus

G52AIM Artificial Intelligence Methods, 2017/2018

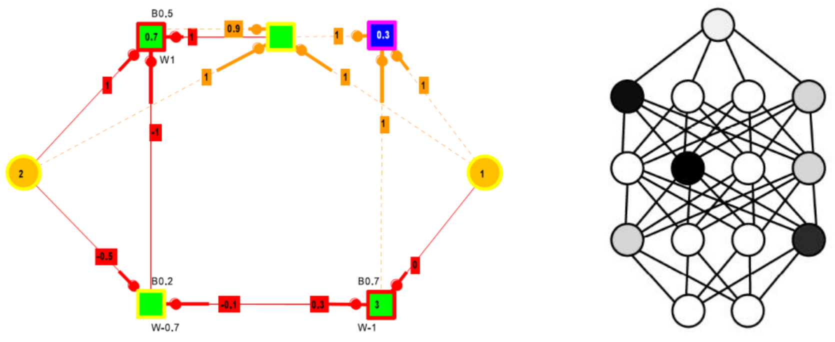
Tomas Maul, ZhiYuan Chen, Kasra Babaei, Mohammed Amer

Release and submission dates: refer to Moodle
Submit your solution by uploading to Moodle

Part I – General Introduction

This coursework is concerned with the problem of optimizing artificial neural networks (ANNs), through evolutionary-based methods. This is typically known as the field of neuroevolution. You will be given some base code to work with, which implements a specific type of neural network called a Neural Diversity Machine (NDM), and a basic set of evolutionary optimization heuristics. You should not modify the NDM code; rather your main goal will be to modify the optimization methods employed, in order to improve them in terms of speed and/or the resulting accuracy of the optimized networks.

Neural diversity machines are artificial neural networks that adopt a large number of diverse transfer functions. Refer to Figure 1(a) for an illustration of recurrent NDMs, that is, NDMs that have both forward and backward connections. Refer to Figure 1(b) for a diagram of feedforward NDMs, which do not contain any backward (or recurrent) connections. The base code that you have been provided with implements feedforward NDMs. Refer to Table 1 for a list of all of the transfer functions implemented. Note that, you are not required to fully understand the NDM, i.e.: you can treat it like a black-box. Your main concern should be about the cost (in this case “training error”) of a particular solution (i.e. a particular NDM), and the evolutionary methods that you will use to bring that cost lower as efficiently as possible.



(a) Recurrent NDM. (b) Feedforward NDM.

Figure 1. Left: illustration of a recurrent NDM. Right: diagram of a feedforward NDM.

Note that in order to design very good heuristics it is typically necessary to fully understand the system you are optimizing, however in the case of this coursework you are not expected to fully understand the nature of NDMs, and therefore you should adopt a more general/robust strategy. In other words you should strive to develop evolutionary heuristics and strategies that are robust to a broad set of problems.

Briefly put, your goal in this coursework is to apply what you have learnt in the lectures and earlier labs, in addition to your own reading and thoughts, to the problem of improving the optimization part of the given neuroevolution code.

Type	Functions	Equations
WF	Inner product	$a_j = b_j + \sum_{i=1}^{n} w_{ji} x_i$
WF	Euclidean distance	$a_j = \sqrt{\sum_{i=1}^{n} (w_{ji} - x_i)^2}$
WF	Product	$a_j = \prod_{i=1}^{n} c w_{ji} x_i$
WF	Higher-order product	$a_j = \prod_{i=1}^{n} \left\| x_i^{w_{ji}} \right\|$
WF	Standard deviation	$a_j = \left(\sum_{i=1}^{n} w_{ji}\right) \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2}$
WF	Minimum	$a_j = \min(w_{j1} x_1 \cdots w_{jn} x_n)$
WF	Maximum	$a_j = \max(w_{j1} x_1 \cdots w_{jn} x_n)$
NF	Identity	$o_j = a_j$
NF	Sigmoid	$o_j = \frac{1}{1 + e^{-c a_j}}$
NF	Gauss. w/ thresh.	$o_j = e^{-\frac{(a_j)^2}{c}}$, if $o_j = d$ then $o_j = 1$

Table 1. Transfer functions adopted in Neural Diversity Machines.

Getting Started

To first run the neuroevolution code, follow these steps:

- Download and unzip the neuroevolution code from Moodle.
- Open MatLab.
- In MatLab, go to the folder containing the Neuro_Evol code, right click the Neuro_Evol code folder, and click on *Add to Path > Select Folders and Subfolders*.
- Double click on the Data13img2D.mat file in the DataMy folder.
- In the MatLab interpreter, type and run the command `res = neuroevol(data6,'data6');`
- Explore the results provided in `res`. For example:
    - Use `res.neuroevol{1}.trainErrors` to see the training error curves.

If all goes well, you will be optimizing your first NDM and you should see a figure depicting the training error curve. This small set of steps will also show you the parent function, from which you will be able to uncover all the other relevant functions (i.e. `neuroevol` in the Neuroevolution folder).

Have a look at the code in the neuroevol function. In this function you will find several parameter initializations for NDMs (`limits`) and optimization (`memParams`). Notice the `memeticO1pureSEDeeNN1` function call. This is the optimization method you will be working with.

Take your time looking at all of the code starting from `neuroevol`, paying special attention to the following, and without forgetting other called functions:

- `neuroevol`
- `memeticO1pureSEDeeNN1`
- `doCrossOver`
- `doProbMingle`
- `doMutation`
- `doDifferential`

- `evaluateSolutionsCC`
- `evalKPH2PR1ccEuclideEDeeNN1`
- `trimSolutionsGen`

Notice where the main evolutionary heuristics are applied in `memeticO1pureSEDeeNN1`: see the code snippet below. This is where one of the main targets of your attention should lie for this coursework, although note that the optimization code should also be modified in other areas (e.g. population mechanics).

```
% Iterate through generations
while (totEval <= memParams.maxEval) && (dei <= memParams.maxIter) &&
(bestCost > memParams.minCost) && (~doStop)

    % *** Global expansions
    coSols = []; pmSols = []; mSols = []; deSols = [];
    % Cross-over
    if memParams.doCrossOver
        coSols = doCrossOver(solPostInit,memParams);
    end
    % Probabilistic mingling
    if memParams.doProbMingle
        pmSols = doProbMingle(solPostInit,memParams);
    end
    % Mutation
    if memParams.doMutation
        mSols = doMutation(solPostInit,memParams);
    end
    % Differential evolution
    if memParams.doDiffEvol
        deSols = doDifferential(solPostInit,memParams);
    end
(…)
```

Part II – Coursework Description

Expanding on what was stated earlier, your goal in this coursework is to design, implement, test, experiment with, and analyse new evolutionary heuristics and strategies in order to improve the optimization of neural diversity machines. More specifically your goal is to minimize the training error more efficiently (faster) and more effectively (to lower values). The training error we are concerned with is the one at the end of the optimization process (`res.neuroevol{1}.trainErrors(:,end)`), and your main focus will be on:

- modifying the given heuristics (`doCrossOver, doProbMingle, doMutation and doDifferential`) [at least one heuristic should be modified],
- adding completely new heuristics [at least one new heuristic should be added],
- removing heuristics [this is optional],
- and modifying population mechanics (e.g. modifying the `trimSolutionsGen` or related functions) [at least one change in population mechanics is required].

You should aim to not only be innovative, but also to perform your experiments systematically. In particular, you need to compare several conditions using the statistical tests that were explained to you in earlier labs.

Regarding the details of the different heuristics already implemented in the base code (e.g. `crossOverSolCC`), you are not necessarily required to understand all of the details (e.g. the selection of solution pairs via complementary classifications), since the implemented heuristics are above all placeholders for your code, and it is perfectly valid for you to re-implement them completely. It is important however, for you to understand the representation of solutions. Solutions are represented in matrices with r rows and c columns, where r consists of the number of solutions in a population, and c consists of 1 + p + d elements, where p consists of the number of patterns, and d consists of the number of parameters in a solution. In more detail, the first element of a row consists of the solution's cost, the next p elements consist of the pattern recognition errors committed by the solution on different patterns, and the next d elements consist of the actual solution parameters. Unless you are interested in the specific errors committed on specific patterns, you can focus only on the first element (i.e. cost), and the last d elements (i.e. solution parameters).

Note that you are not required to understand every last function in the base code. You just need to make your way down from `neuroevol,` and investigate the relevant called functions.

In order to facilitate comparisons between students, note that you should neither modify the parameters nor the code pertaining to neural diversity machines. In other words, all students should experiment with different optimization techniques but should work with the same NDM, which adopts the following parameters:

```
% Neural network parameters
limits.architecture = 1;
limits.numNodes = [2 6 6 1];
limits.P_minWB = -1;
limits.P_maxWB = 1;
limits.sig_minFP = 0.3;
limits.sig_maxFP = 3;
limits.gaus_minFP1 = 0.01;
limits.gaus_maxFP1 = 10;
limits.gaus_minFP2 = 0;
limits.gaus_maxFP2 = 2;
limits.weightFunctions = [1 2 3 4 5 6 7];
limits.nodeFunctions = [1 2 3];
```

The number of evolutionary generations should also be fixed at 50 (i.e. `memParams.maxIter = 50)` and the population size fixed at 24 `(i.e. memParams.ngSize = 24).`

As mentioned above statistical comparisons between approaches are critical. You could create different conditions, where each condition corresponds to typically one critical difference (e.g. a different heuristic, set of heuristics, or population mechanism). Armed with statistical tests, you should be able to compare the different conditions in terms of the efficiency and effectiveness of your optimization techniques.

Note that this coursework is to be done individually. Although you can and should have discussions with your colleagues, in order to accelerate your comprehension of the base code and lecture/lab material, and you can discuss some of your novel ideas (although make sure you always attribute the originator of the idea), however, the code, experiments, analyses, and report, should all be your own individual effort.

Part III – Deliverables and Marking Criteria

Main deliverables: (1) code, (2) report of 3 pages (font: Calibri size 11). More details on how to submit your code and report will become available in Moodle. The report should follow the format of a mini conference paper, with the following standard elements: title, abstract, keywords, introduction, methods, results, discussion, acknowledgements (optional), and references.

The table below summarizes the main marking criteria, which are all equally weighted.

Marking criteria	
Code	Example aspects
Quality of Source Code	Simplicity, clarity and efficiency.
Performance	How significant are the improvements?
Report	Example aspects
Writing style	Structure, clarity and grammar.
Introduction	Understanding of area; depth and relevance of references to the literature. Quality of problem statement.
Methods	Novelty, relevance and understanding.
Results	Clarity, completeness, relevance, and usage of statistics.
Discussion	Correctness of analysis; validity of interpretations; novelty and depth of insights.