COMP 307 — *Introduction to AI*

Assignment 2: Neural and Evolutionary Learning (***Draft***)

12% of Final Mark — Due 23:59 Monday 8 May 2017

1 Objectives

The goal of this assignment is to help you understand the basic concepts and algorithms of neural and evolutionary learning, use these algorithms to perform regression and classification tasks, and analyse the results to draw some conclusions. In particular, the following topics should be reviewed:

- Multilayer feed forward neural network architectures and applications,
- Back (error) propagation algorithm and its variations,
- Tackling a problem with an existing neural network package,
- Evolutionary computing and learning paradigms,
- Genetic programming for solving real world applications particularly for regression and binary classification problems, and
- Tackling a problem with an existing genetic programming package.

These topics are (to be) covered in lectures 08–12. The online materials can also be checked.

In this assignment, neural networks refer to the standard multilayer feed forward neural networks trained by the back propagation algorithm. Genetic programming refers to the standard genetic programming approach with the tree-like structure for the evolved programs.

2 Question Description

Part 1: Neural networks for Classification [30 marks]

In this part, you are required to use an existing neural network package to perform classification on the *iris* data set described below. Please note that you already used this data set in the previous assignment.

Problem Description

The *iris* data set is taken from the UCI Machine Learning Repository (`http://mlearn.ics.uci.edu/MLRepository.html`). The original data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. There are four numeric attributes: *sepal length in cm, sepal width in cm, petal length in cm*, and *petal width in cm*. More detailed information can be seen from the file `iris.names`.

Requirements

You should define a neural network architecture for this problem, determine related network learning parameters, use the training set `iris-training.txt` to learn/train your neural network, then apply the learned/trained neural network on the test set `iris-test.txt`.

While it is good to write your own program packages to implement the back (error) propagation algorithm for training the multilayer feed forward networks, it might take you quite a while to do so. So we recommend two neural network simulators (packages) for doing this assignment, one in ANSI C (called BPNN), one in Java (called JOONE, having some minor problems), and one with a GUI interface – a very powerful simulator used in many universities in the world (called SNNS). You can choose any of these existing packages/simulators to perform the task. You can also choose any other package from the website, but please describe your choice in your report.

In this case, you will not need to write any programs for this task. Instead, you should submit the following files electronically and also a report in hard copy.

- `new training set/test set file(s) with a correct format` for the neural network package you have chosen, and
- `A report` in any of the PDF, text or DOC formats. The report should include:
    1. Determine and report the network architecture, including the number of input nodes, the number of output nodes, the number of hidden nodes (assume only one hidden layer is used here). Describe the rationale of your choice.
    2. Determine the learning parameters, including the learning rate, momentum, initial weight ranges, and any other parameters you used. Describe the rationale of your choice.
    3. Determine your network training termination criteria. Describe the rationale of your decision.
    4. Report your results (average results of 10 independent experiment runs) on both the training set and the test set. Analyse your results and make your conclusions.
    5. (optional/bonus) Compare the performance of this method (neural networks) and the nearest neighbour methods.

Part 2: Genetic Programming for Symbolic Regression [30 marks]

In this part, you are required to use genetic programming to evolve a mathematical function (which is an individual program in the population) for a simple symbolic regression task. In real world applications, a test set of data is needed in many situations but not needed in other scenarios, depending on the nature of the problems to be solved. In this assignment, to make the question simple, it is not required to have a test set — you are just required to evolve a mathematical function to reveal the relationship between the output variable and the input variable(s) from a (training) set of instances.

Problem Description

The task involves mapping a single input variable x to the (single) output variable y. In a 2D (two-dimensional) space (x-y coordinators), there are 20 points (x-y pairs). The task is to find a mathematical function to describe the relationship between the output variable y and input variable x. The 20 points are as follows. They are also saved in the file `regression.txt` in plain text format.

x	-2.0	-1.75	-1.50	-1.25	-1.00	-0.75	-0.50	-0.25	0.00	0.25
y	37.0000	24.1602	15.0625	8.9102	5.0000	2.7227	1.5625	1.0977	1.0000	1.0352
x	0.50	0.75	1.00	1.25	1.50	1.75	2.00	2.25	2.50	2.75
y	1.0625	1.0352	1.0000	1.0977	1.5625	2.7227	5.0000	8.9102	15.0625	24.1602

Requirements

It is very time consuming to write your own GP package from the beginning. As many GP packages are available, this is actually not necessary. In this assignment, we receommend three GP packages for the GP questions. They are JGAP (a Java GP library), RMITGP (written in C++, fully documented), GPC++ (written in C++, fully documented). Again, you can choose other GP packages such as VGP (Developed by VUW, in C/C++), ECJ (in Java). Please describe your choice in your report.

Your job is to use any of the genetic programming packages with necessary minor changes of the terminal set, function set, fitness function, parameters and termination criteria to solve the simple task described above.

You should submit the following files electronically and also a report in hard copy.

- `Program code` written by yourself (both the source code and the executable program running on ECS School machines),
- `readme.txt` describing how to run your program, and
- `A report` in PDF, text or DOC format. The report should include:
    1. Determine a good terminal set for this task.
    2. Determine a good function set for this task.
    3. Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).

4. Describe the relevant parameter values and the stopping criteria you used.
5. List three different best programs evolved by GP and the fitness value of them (you need to run your GP system several times and report the best programs of the runs).
6. (optional, bonus) Analyse one of the best programs to reveal why it can solve the problem in the task.

Part 3: Genetic Programming for Classification [40 marks]

In this part, you are required to use GP to classify the Wisconsin data set instances into two classes. This data set was obtained from the *UCI machine learning repository* (`http://mlearn.ics.uci.edu/MLRepository.html`).

Problem Description

This data set consists of 699 instances of breast cancer diagnosis data, obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. The data instances are defined by 9 numerically valued attributes (excluding the sample id number) and result in either a classification of *benign* or *malignant*. This dataset contains 458 "benign" instances (65.5%) and 241 "malignant" instances (34.5%). The file `breast-cancer-wisconsin.data` is the data set consisting of these 699 instances.

All of the 9 attributes are integer valued between 1 and 10 and are listed as follows: *Clump Thickness (CT), Uniformity of Cell Size (USz), Uniformity of Cell Shape (UShp), Marginal Adhesion (MA), Single Epithelial Cell Size (SESz), Bare Nuclei (BN), Bland Chromatin (BC), Normal Nucleoli (NN), Mitoses (M).* Sixteen of the instances also contain "missing attributes" (denoted by "?"). For these attribute values, one suggestion is to replace these question marks ("?") with -1 for classification using GP. All the 699 instances including the 16 instances with missing attributes should be used in the experiments. The file `breast-cancer-wisconsin.names` gives detailed description of the data.

Requirements

Your job is to use any of the genetic programming packages with necessary minor changes of the terminal set, function set, fitness function, parameters and termination criteria to solve the simple task described above.

You should submit the following files electronically and also a report in hard copy.

- `Program code` written by yourself (both the source code and the executable program running on MSCS School machines),
- `training.txt` and `test.txt` with appropriate formats,
- `readme.txt` describing how to run your program, and
- `A report` in PDF, text or DOC format. The report should include:
  1. Determine a good terminal set for this task.
  2. Determine a good function set for this task.
  3. Construct a good fitness function and describe it using plain language (and mathematical formula, or other formats you think appropriate).
  4. Describe the relevant parameter values and the stopping criteria you used.
  5. Split the original data set into a training set `training.txt` and a test set `test.txt` with appropriate format for the GP package you have chosen. Describe your main considerations of the splitting.
  6. Report the classification accuracy (average accuracy over 10 independent experiment runs) on both the training set and the test set.
  7. List three best programs evolved by GP and the fitness value of them.
  8. (optional, bonus) Analyse one of best programs to reveal why it can solve the problem in the task.

3 Relevant Data Files and Program Files

The relevant data files, information files about the data sets, and some utility programs files can be found from the following directory:

`/vol/comp307/assignment2/`

Under this directory, there are three subdirectories `part1, part2`, and `part3` corresponding to the three parts of the assignment, respectively.

4 Notes

As an assignment in a 300 level course, you can make your own assumptions if necessary.

During the time between the assignment handout and submission, the tutor(s) will run a number of helpdesks to provide assistance.

5 Submission Guidelines

5.1 Submission Requirements

1. Program codes you wrote and training (and test) set files that are used for all individual parts. To avoid confusion, all the individual parts should use directories `part1/, part2/, ...` and all pieces of programs should be stored in their corresponding directories. Within each directory, please provide a `readme file` that specifies how to run your program. If posible, a script file called `sampleoutput.txt` should also be provided to show how your program run properly. If you programs cannot run properly, you can provide a `buglist` file. You need to submit these codes and files in *soft copy* only.

2. A document consisting of the report of all the individual parts. The document should mark each part clearly. The document can be written in PDF, text or the DOC format. You need to submit this document in both *soft copy* and *hard copy*.

5.2 Submission Method

The programs and the PDF/Text/DOC version of the document should be submitted through web submission system from the COMP307 course web site **by the due time**.

The hard copy of the document is required to submit to the COMP 307 handin box in the 2nd floor corridor of the Cotton building by the due time.

5.3 Late Penalties

All assignments must be submitted on time unless you have made a prior arrangement with the lecturer or have a valid medical excuse (for minor illnesses it is sufficient to discuss this with the lecturer.) Assignments that are handed in late without prior arrangement will be marked if time permits, but the mark might not contribute to your final grade.