Project 2:

Genetic Programming for Symbolic Regression

Assigned: Tuesday, Oct. 2

Multiple Due Dates (all submissions must be submitted via BlackBoard):
Undergrads: Generally complete, compilable code: due Mon., Oct. 15, 23:59:59 (i.e., midnight)
Completed software and paper: due Monday, Oct. 22, 23:59:59 (i.e., midnight)

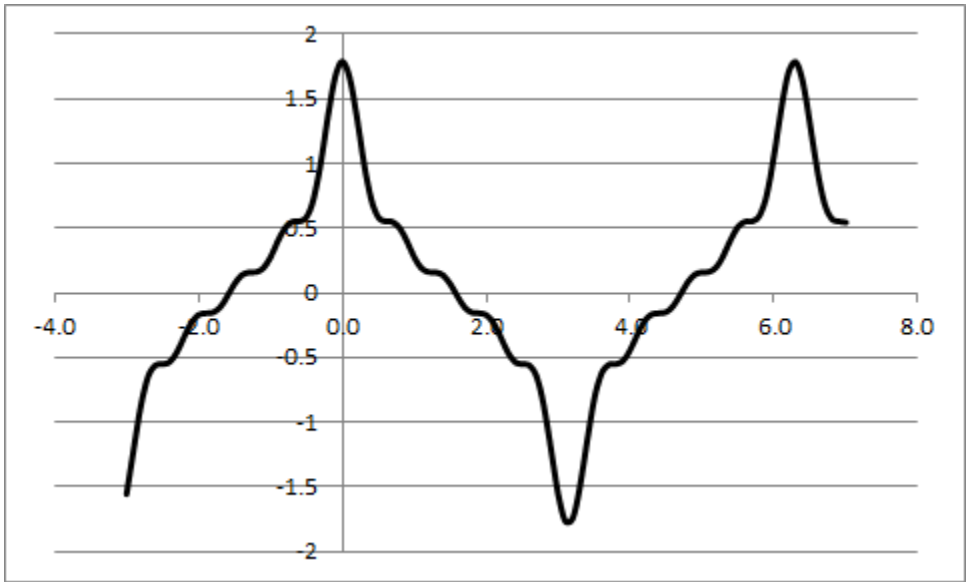
Grads: Completed software: due Friday, Oct. 19, 23:59:59 (i.e., midnight)
Completed paper: due Monday, Oct. 22, 23:59:59 (i.e., midnight)

Introduction

In one of our class handouts from the Koza text (section 7.3), an example of symbolic regression is discussed. In this project, you will implement genetic programming to solve a specific symbolic regression problem. Recall that *symbolic regression* is the problem of fitting a function to a set of data, and expressing the resulting function in symbolic mathematical form (in contrast to sets of matrices, which is all that can be derived from a neural network). The objective is to find the symbolic function that matches the data as closely as possible. Your genetic programming approach will be very similar to that described in the Koza text, although you may want to define some details differently.

Sample data

The data you will be fitting is available on the course website at http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/Projects/proj2-data.dat, and consists of 101 pairs of x and $f(x)$, with x values ranging from -3 to 7, in increments of 0.1. To help you visualize the data, I have included a graph of this data below.



[As a hint, I will say that the objective function, $f(x)$, that I used to create the data for this problem made use of "ordinary" mathematical operations.] Your objective in this project is to implement a genetic

program that can discover the original function that created this data, with the smallest error possible for these data points. For this project, the error that you measure is sum of squared error (i.e., take the difference between each data point and the value your function generates, square it, and add it to all the other squared differences). For this data, you should expect to achieve a sum of squared error of 0.25 or less.

About your approach

To generate your original population, you must use the "ramped half-and-half" approach we discussed in class (which is from the Koza handout, pg. 93). In this approach, you must eliminate duplicates from the original population. The selection of the maximum depth to use for this approach is up to you. You must use sum of squared error as your fitness function. You may also select all the other parameters of the genetic program as you see fit, including the function for fitness selection (e.g., fitness proportionate, tournament selection, rank selection, etc.).

Data to Gather and Results to Report

Your results should include:

- A graph that plots (on a single graph) (1) the average fitness of the population, and (2) the fitness of the best individual in the population, as a function of the generation number.
- A graph that plots the percentage variety of the population, as a function of the generation number. This just shows what percentage of the population is unique (and should start at 100%, since duplicates are removed when the population is initially formed).
- A single graph that plots the data points you are provided (in proj2-data.dat), along with the data values generated by the best function your genetic program learns. Be sure it is clear in your graph which data set each point belongs to.
- The sum of squared error for the most fit individual learned by your GP.

Your reported results should include the function that represents the best individual found by your learning system. To the extent possible, you should also express this result in "ordinary" mathematical terms, simplifying as needed, so that it is clear what the function is that your program has discovered. (For example, your solution might be (sin (+ (* x x) (+ x x))). You should also convert and simplify this to: $\sin(x^2 + 2x)$). You should make it clear how many generations it took to derive this solution, and any other interesting observations you make about your solution.

What your submitted code should do

The final code that you submit should have 2 options – one option is the genetic program in learning mode, and the second option is to test the best learned program on a specific input value of *x* between -3 and 7. Your code should allow the user to specify whether the code should be run in the training mode or the testing mode. This means that your project submission should include the best genetic program you found during your own training, so that you can use those during testing mode.

It is up to you to design the user interface to allow the user to specify which mode the code should be in. Your user interface should also allow the user to input the *x* value to use for testing your code (again, between -3 and 7).

Be sure to provide documentation in your "README" file on how the user should specify the inputs. Or, make your user interface so easy to use that the user just has to follow your program's online instructions.

CS425 – Undergraduate Instructions

Undergraduate Student Paper Guidelines

As part of your project, you must prepare a 1-2 page document (using Latex, in 2-column IEEE pdf format) that documents the genetic program design you used and the values of any parameters (such as crossover rate) that you used in your system. Note that this paper must be created using Latex in the IEEE style, as described in the graduate student instructions for Project #1. Name your paper *yourlastname*-Project2-Paper.pdf.

Your paper must include your formulation of the problem; specifically your definition of:

- Terminal set [Give the names and types]
- Primitive function set [Give the list of functions, the number of operands, the types of the operands, and a brief description of the mathematical function represented by each primitive, if it isn't obvious. Also, provide details of any "protected" function characteristics, if applicable (e.g., preventing divide by 0).]
- Fitness function [Give the function]
- Method of probabilistic selection [Give the function]
- Other GP Parameters: population size, fraction of population to be replaced by crossover, maximum depth of members in initial population, maximum depth for trees created by crossover, probability of mutation, and perhaps other design-specific parameters.

This document must also include results of your genetic program, as described above in the "Data to Gather" section. Remember to include the sum of squared error of the most fit solution your GP discovers. If you want to present results from several different runs of the genetic program, or different parameter settings, that would be terrific. Just be sure to label everything so that it is clear which results correspond to which parameter settings, etc. To the extent you desire, you are welcome to add additional descriptions along the lines of the graduate student paper. However, this is not required. Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates.

Undergraduate Grading

Note that, as with Project #1, you first must turn in a "generally complete" version of your software, prior to the final version of your software. "Generally complete" means that your code:

- Compiles
- Implements the genetic programming approach, including selection, crossover, and mutation
- Runs without crashing

This code (for the first deadline) does not have to demonstrate good learning performance, however. It just needs to be at the stage that you can tweak the design of the GP and the parameters of the system to get it to learn properly. However, it does need to have all the implementation completed for the main learning procedures. For the final submission, you will turn in working software and the paper, as described earlier.

Your grade will be based on:

- 15%: "Generally complete" software (i.e., software with the above characteristics). This software does not have to learn well. But, it has to have all the pieces in place for you to begin tweaking the parameters to achieve good learning performance.
- 65%: The quality of your final working code implementation, and software documentation. You should have a "working" software implementation, meaning that the learning algorithm is implemented in software, it runs without crashing, performs learning iterations as appropriate for a genetic program, and learns reasonably well. Your final code should have the training mode and testing mode, as outlined earlier in this document.
- 20%: Your paper, generated in Latex using IEEE styling, and submitted in pdf format -- Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates. The content should include all the results and discussion mentioned previously in this document.

Undergraduates: Turning in your project

You should submit your project via Blackboard by the deadlines. ***Note that you will submit to Blackboard twice, since you have two (2) deadlines for this project.***

Submission #1:

1. This submission is of the "generally complete code". It should include all the programs, include files, makefiles etc., needed to compile and run your project on the CS linux machines (cetus or hydra), including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project2-prelim.cc (or whatever extension is appropriate for the programming language you're using). You DO NOT need to submit the data file we provided you (i.e., the x, f(x) data set).

Submission #2:

This submission should be in 2 parts:

1. Paper (in pdf, 2-column IEEE format, generated using Latex, named *yourlastname*-Project2-Paper.pdf)
2. All the programs, data files, include files, makefiles etc., needed to compile and run your project on the CS linux machines, including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project2.cc (or whatever extension is appropriate for the programming language you're using). Combine all files together into a single tarball (with extension .tar or .tar.gz or .tar.zip). You DO NOT need to submit the data file we provided you (i.e., the x, f(x) data set).

When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work ☹, and reserve the right to count off points for egregious cases. So please test your instructions before you send your files to us.

CS528 – Graduate Instructions

Graduate Student Paper Guidelines

As always, as part of your completed project, you must prepare a paper (3-6 pages) describing your project. Your paper must be formatted using Latex and the standard 2-column IEEE conference paper format, the same as in the previous project.

The paper you turn in must be in pdf format. Name your paper *yourlastname*-Project2-Paper.pdf. Your paper must include the following:

- An abstract of 200 to 300 words summarizing your findings.
- An introduction describing the learning task and your formulation of the problem, including the definition of all the following genetic programming design items:
    - Terminal set [Give the names and types]
    - Primitive function set [Give the list of functions, the number of operands, the types of the operands, and a brief description of the mathematical function represented by each primitive, if it isn't obvious. Also, provide details of any "protected" function characteristics, if applicable (e.g., preventing divide by 0).]
    - Fitness function [Give the function]
    - Method of probabilistic selection [Give the function]
    - Other GP Parameters: population size, fraction of population to be replaced by crossover, maximum depth of members in initial population, maximum depth for trees created by crossover, probability of mutation, and perhaps other design-specific parameters.
- A detailed description of your experiments, with enough information that would enable someone to recreate your experiments.
- An explanation and discussion of the results, including the best function learned by your learning system (and its symbolic equivalent). Include the graphs mentioned earlier in "Data to Gather and Results to Report". Use additional figures, graphs, and tables where appropriate.
- Additional explorations of your GP approach, as outlined in the grading section below.

Graduate Grading

Graduate students will be graded more strictly on quality of the research and paper presentation. I expect a more thorough analysis of your results. Your analysis should include a discussion (and perhaps results) on some of the following points (in addition to the points previously noted above):

- (If relevant) other function and/or terminal sets that you experimented with before you got a working system
- Effect of system parameters on results (such as population size, percentage for selection, method for selection, percentage for crossover, etc.)
- Effect of different techniques for selection, or for generating the initial population

- Analysis of population variation, as compared to fitness, and whether it sheds any insight on the learning solution quality or convergence
- Future work that you believe would improve the learning
- Any other insightful observations you'd like to make

You should not address these points in a bullet-type fashion, but instead work the answers into your paper in a discussion-style format. The paper should have the "look and feel" of a technical conference paper, with logical flow, good grammar, sound arguments, illustrative figures, etc. As always, graduate students are expected to format their paper in standard IEEE conference format, as described in the Project #1 assignment.

Please do not exceed 6 pages for your paper.

Your grade will be based on:

- 65%: Your documented code, which achieves highly accurate results. Your final code should have the training mode and testing mode, as outlined earlier in this document.
- 35%: Your paper, which includes an analysis of your genetic programming project regarding the points mentioned previously

Graduates: Turning in your project

You should submit your project via Blackboard by the deadlines. ***Note that you will submit to Blackboard twice, since you have two (2) deadlines for this project.***

Submission #1:

This submission is of your final code, and should include all the programs, include files, makefiles etc., needed to compile and run your project on the CS linux machines (cetus or hydra), including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project2.cc (or whatever extension is appropriate for the programming language you're using). Combine all files together into a single tarball (with extension .tar or .tar.gz or .tar.zip). You DO NOT need to submit the data file we provided you (i.e., the $x, f(x)$ data set).

When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work ☹. So please test your instructions before you send your files to us.

Submission #2:

This submission is of your paper (in pdf format, named *yourlastname*-Project1-Paper.pdf), formatted in the IEEE conference style format mentioned earlier.