Symbolic Regression (SR)

Wikipedia defines Symbolic Regression (SR) as a type of regression analysis that searches the space of mathematical expressions to find the model that best fits a given dataset, both in terms of accuracy and simplicity. No particular model is provided as a starting point to the algorithm. Instead, initial expressions are formed by randomly combining mathematical building blocks such as mathematical operators, analytic functions, constants, and state variables. Usually, a subset of these primitives will be specified by the person operating it, but that's not a requirement of the technique. Symbolic regression is one of the best known problems in Genetic Programming. Aside from being commonly used as a tuning problem for new algorithms, it is also widely used with real-life distributions, where other regression methods may not work. All symbolic regression problems use an arbitrary data distribution, and try to fit the data with the most accurate symbolic formula available. Usually, measures like the RMSE (Root Mean Square Error) or MSE (Mean Squared Error) are used to measure an individual's fitness.

In this project, I will be using a classical distribution, the quartic polynomial ($5x^2 - 3x + 8$), a one-dimension distribution. *20* equidistant points are generated in the range [-10 11 2], and are used to evaluate the fitness.

Symbolic Regression of A Quadratic Polynomial

Genetic Programming is often used in the problem of symbolic regression where the goal is to automatically create a computer program whose output is equal to the values of the quadratic polynomial $5x^2 - 3x + 8$ in the range from -10 11 2. Essentially there are 4 preparatory steps to the problem and this leads onto the executional steps of the actual running of Genetic Programming.

Preparatory Steps

The purpose of the first two preparatory steps is to specify the ingredients of the to-be-evolved

program.

1. The terminal set (inputs to the to-be-evolved program) includes the independent variable, x. The terminal set also includes numerical constants. That is, the terminal set, (x x x 1 2 3 4 5 6 7 8 9), with constant numerical terminals in some reasonable test-domain range of (-10 11 2).
2. The function set consists of the four ordinary arithmetic functions of addition, subtraction, multiplication, and the division function (%) which returns a value of 1 when division by 0 is attempted (including 0 divided by 0), but otherwise returns the quotient of its two arguments.. This choice is reasonable because mathematical expressions typically include these functions. Thus, the function set, F, for this problem is F = '(+ - *).

The third preparatory step involves constructing the fitness measure.

3. The purpose of the fitness measure is to specify what the human wants. The high-level goal of this problem is to find a program whose output is equal to the values of the quadratic polynomial $5x^2 - 3x + 8$. Therefore, the fitness assigned to a particular individual in the population for this problem must reflect how closely the output of an individual program comes to the target polynomial $5x^2 - 3x + 8$. A smaller value of fitness (error) is better. A fitness (error) of zero would indicate a perfect score.

For most problems of symbolic regression or system identification, it is not practical or possible to analytically compute the value of the integral of the absolute error. Thus, in practice, the integral is numerically approximated using dozens or hundreds of different values of the independent variable x in the range between − 1.0 and +1.0. In actual practice, the population size for a run of genetic programming consists of thousands or millions of individuals. In actual practice, the crossover operation is commonly performed

on about 90% of the individuals in the population; the reproduction operation is performed on about 8% of the population; the mutation operation is performed on about 1% of the population; and the architecture-altering operations are performed on perhaps 1% of the population.

Lastly the user has to specify the termination criterion.

4. A reasonable termination criterion for this problem is that the run will continue from generation to generation until the fitness of some individual gets below 0.01.

Executional Steps

Once the human user has performed the preparatory steps, the executional steps shown in the flowchart of genetic programming are performed. Genetic programming starts by randomly creating a population of the specified individual computer programs. After the fitness of each individual in the population is ascertained, genetic programming then probabilistically selects relatively more fit programs from the population. The genetic operations (reproduction, mutation and crossover) are now applied to the selected individuals to create offspring programs.

In summary, genetic programming I have implemented has automatically created a computer program whose output is equal to the values of the quadratic polynomial **$5x^2 - 3x + 8$** in the range from **[-10 11 2].** The coding is self explanatory.

Main Implementation Guideline:

==========

This is an Implementation of Genetic Programming to do Symbolic Regression.

I set this up for regression of polynomials, which is used in those situations where the relationship between study and explanatory variables is **curvilinear**. Sometimes a nonlinear relationship in a small range of explanatory variables can also be modelled by polynomials.

Note this Instructions

Modify `test-function` or set `test-domain` and `test-values` to something you want to evaluate fitness against.

You can equally modify the function and terminal sets. Functions currently must be of type `(-> Number Number Number)`

To run this program, just type the below at the prompt:

> (run-gp)

To Compile:

.DS_Store

***.rkt~**

compiled/