**Analysis of different type of function models using symbolic regression**

Sirisha Narayansa

Department of Computer science

UMSL

St. Louis MO 63121

snzfy@umsystem.edu

**Abstract:** This paper examines the study of different forms of goal functions and analyze how some functions are difficult is to find or discover mathematical expressions of functions that fit the given data set in genetic programming. Symbolic regression is a difficult task in general due to the richness of the space of mathematical expressions. For this experiment I have used in cartesian genetic programming. In this paper I have used some benchmark function to see how CGP evolves problem without prior knowledge and analyze with some artificial benchmarks with recommendations for good parameter settings are provided. Through comprehensive experiments, results show that some model performs strongly compared to other benchmark competing models with respect to the tree size, fitness evaluation.

**Keywords**: Genetic Programming, Benchmark functions, artificial benchmark functions, genetic operators.

**Introduction:** Regression analysis is set of statistical process for estimating the relationships between dependent variable (output-x) and one or more independent variable(x)[1]. The main goal or purpose of regression analysis is used to draw conclusions between expected output function and objective function.Symbolic regression is a machine learning method that enables to find the optimal mathematical expression to explain a relationship. That anticipate new data, it initializes a population of naïve random formulas to indicate a link between known independent variables and their dependent variable goals. Each generation of programs is then evolved from the one before it by picking the fittest individuals from the population for genetic operations. Symbolic Regression (SR) is a type of regression analysis that explores a set of mathematical expressions for the optimal solution to match a given dataset in terms of accuracy and simplicity. As a starting point for the algorithm, no specific model is offered.

Symbolic Regression (SR) is the task of determining the form of a function that explains hidden relationships in data without prior knowledge of its form.Symbolic regression avoids imposing prior assumptions, and instead infers the model from the data.

**Background:** Devising a good benchmark suite has been recognized as an important issue for the next ten years of GP [2]. While much of the GP application literature focuses on nontrivial, domain-specific problems, the fundamental comparison and analysis literature frequently utilizes relatively simple benchmark problems. Such benchmarks, we believe, do little to advance the discipline and, in fact, may hold it back by rewarding strategies that are effective at solving minor issues quickly rather than doing as well as possible on difficult ones. These two purposes may be at odds, as argued in [3]. The introduction of measures that promote such rewards aggravates the situation. In this paper, we examine related work and explain what makes a meaningful benchmark, the state of benchmarks in general practice, and the statistical challenges that arise when evaluating performance on benchmarks.

**Genetic Programming:**

(Reichenberg 1965) created evolution strategies that started with a population of computer programs, altered or “mutated” individual programs in the population and tested whether the mutated program was better than its parent by using a fitness measure.

Genetic programming is a way of optimizing algorithms in machine learning that is inspired by the natural selection of genes conveying beneficial characteristics through evolution over generations. The general concept is to create a population of programs that are gradually enhanced through 'natural' selection. Because Genetic Programming may be defined in a very generic way by putting together its components, we'll save mathematical representation and statistical models until later. For the time being, we will define the purpose and components in simple terms and draw out a concept of how the program would behave in general.

Genetic programming is a domain-independent way of solving problems by genetically breeding a population of computer programs. Genetic programming iteratively evolves a population of computer programs into a new generation of programs by using analogs of naturally occurring genetic operations.

Benchmark functions: Benchmark function is an optimization function which can. randomly divide the objective variables into several groups, each of which contains several variables.

Benchmarks is an important part of tests and validation is to use benchmark functions to test how the new algorithm may perform in comparison with other algorithms [4]. However here I used some benchmark functions as goal functions to analyze empirically how CGP can find in the search space to find best suitable mathematical expression.

Symbolic Regression is performed by methods which minimize various error metrics while searching the space of mathematical expressions to estimate the accurate and simple model that best fits the observed dataset [5].

**Populations and generations:**

The population is made up of fixed-size programs. When we replace the programs in our population with new ones, we say we've progressed from the current generation to the next.

**Fitness and selection:**

The fitness function is a metric for determining how effectively a population's programs execute. The fitness function used is critical, as a bad choice might slow down the rate of development between generations. The technique of selection is also a significant consideration, and the best option is determined by the problem.

**Crossover and mutation:**

Even though genetic programming run without mutation and with only crossover or with only mutation without crossover it will not give best solutions or offspring to evolve. That’s why I have used standard crossover and mutation rate for this experiment in later sections we will examine the different standard input parameters.

**Representing model formula:** The model formulae must be mentioned when creating a new population of models. Inverted trees are a useful approach to describe model formulas in Symbolic Regression. Because the computer can evaluate expressions recursively, they are frequently represented in software as trees or lists. Each node in the tree represents a binary operator (+,-,\*,/), an analytic function (sin, cos, exp, log,...), or a constant or variable. The model space can be explored by including analytic functions and division. The examples below demonstrate how we may use trees to describe both simple and sophisticated model formulas.

It's worth noting that each formula has a single 'root' node, with all of its edges pointing away from it. The inputs are represented by edges going away from a node, therefore a binary operator (+,-,\*,/) will have two edges connected to its inputs, an analytic function (sin, cos, exp, log) will have one edge connecting to its input, and constants, parameters, and variables will have none (i.e. they will be leaves of the tree). This allows us to quickly determine whether a formula is valid.

The genetic operations that drive the learning process in Genetic Programming, crossover and mutation, are likewise simple to express and depict using tree representation.

For example: y = a + bx

**Experiment setup:**

**Input parameters:** Population size of 500 with max generations 100. Now for the initial generating of populationI have used init. method half and half method and init. depth of 2- 8 with maximum nodes of 500 as the population size was 500. Maximum depth of tree size is 15.

Breeding is the term used in lilgp for creation of new population of each generation [6]. Three operators are crossover (0.85), reproduction (0.05), mutation (0.1) with tournament size of 7. Best member of the population is returned. Boundaries are kept [-1 ,1] for all the functions default. Here my experiment didn’t include the dataset to evaluate.

Not all functions are suitable to be used as benchmarks, however. With these parameters with different forms of objective functions I observed empirically how difficult is to find the desired targeted expressions.

Software I have used is cgp 2.1 with type constraints. Different forms of functions to carry experiment is some modified benchmark functions and some polynomial functions and with trigonometry functions.

|  |  |  |
| --- | --- | --- |
| 1. Trigonometry function | 1 dimension | Y = x\*x + sin(x+2) -x |
|  |  | Y= x\*x + cos(x+2) -x |
|  |  | Y= a\*sin(x) + b\*cos(x) |
|  |  |  |
| 2. Polynomial function | 1dimension | Ax + b(x\*x) + c \* (\*x\*x\*x) |

Table 1.1: Some of objective functions or goal functions used to analyse empirically how program evolve in the search space.

In Table 1.1 Trigonometry functions with same form: Y= a\*sin(x) + b\*cos(x)

Function 3: Y = 0.1 \* sin(x) + 0.7 \* cos(x)

Function 4: Y= 0.3 \* sin(x) + 0.9 \* cos(x)

Polynomial function –

Function 5: 0.3 \*x + 0.9 \*x \*x + 0.7 \*x \*x \*x

|  |  |  |
| --- | --- | --- |
| Matyas function | 2 dimension x1 and x2 | 0.26((x1\* x1) +(x2 \* x2)) – 0.48 x1 \* x2 |
|  |  |  |
| New function | 2 dimensions x1 and x2 | x1\* x2 + sin(x1+2) -x2 |
| Barteles con function | 2dimensions with mod function | |(x1\*x1) + (x2\*x2) + x1\* x2 +|(sin(x1))| + |(cos(x2))| |

Table 1.2 : Benchmark functions from SS Jay algorithm[7]

**Criteria for computation:**

So how can we differentiate which functions are difficult to evolve and which functions easy to aims to identify an underlying mathematical expression that best describes?

1.We can compute by analyzing empirically the no of trees generated per each function throughout the generations so as no of trees size increases it shows it will take more time to compute each generated function .

2.As no of tree size increases it will become more expensive as well.

3. We can also take fitness values in consideration while considering which function is difficult to solve and find expression in search space by GP.

In order to induce different functions in regression application particular changes have been made.

Step 1: First need to check induced functions have how many dimensions.

Step 2: If there is only one variable then no need to change much in other files.

Step 3: Objective function can be induced in app.c file .

Step 4: Make and after evolve program for 10 runs 100 generations.

If there are 2 or more dimensions in the objective function:

Step 1: Structural changes need to be done in app.c and function.c

Step 2: Increase the set size as X changes from X1 and new variable X2.

Step 3: With change in independent variable there is need to change return values in function.c.

Step 4: App.h need to be updates as newly independent variable is introduced.

Step 5: Function definitions need to be updated variables of type ‘double’.

Step 6: app\_initialize() function plays a major role in evaluating the fitness value.

Step 7: Due to structural changes the function app\_eval\_fitness() have the following changes in the function definition.

**Results:**

Chart

Description automatically generated

Figure 1

Chart, line chart

Description automatically generated

Figure 2

In figure 1 we can see that first five functions which is trigonometry and polynomial equation of third degree is giving best fitness values which it means easy to evolve and find suitable mathematical expressions and from figure 2 we can see that function 4 and function 5 has less tree size which in return it is easy to compute and evolve in search space for the optimal mathematical expression.

In figure 1 we can see that last two functions which is new function and Matyas and Bartels Con n function is producing very less fitness values and giving yield to larger tree size[figure 2] which is again difficult to evolve in a given program.

**Experiment with different constraints (functional and terminal set ) for polynomial function:** Firstly I checked with baseline performance with not giving any constraints then I have studied how these constraints helps CGP to find search space for the expected mathematical expression. So, I came with some cases of constraints.

My induced function is: **Y = Ax + B(x\*x) + C(x\*x\*x)**

Case 1: Giving only ( \* + X R)

Case 2: (+ \* X R exp Rlog)

Case 3: ( + \* exp X R)

Case 4: (Unconstraints)A picture containing graphical user interface

Description automatically generated

**Observations and results:** When we give the constraints which is having similar expressions in induced function it gives the best fitness values and it also concludes that it will easy for cgp to search in search space to build achieve objective function.

Text

Description automatically generated

**Conclusion:** In my work I observed that within different types of goal function with different variables yielding difficult to finding underlying function with the increase in search space and as tree size increase symbolic regression is become expensive to evaluate as the generation increases and tree size increases. Best can say that Polynomial function of third degree easy to evaluate and find underlying function with that searches the space of mathematical expressions to find the model that best fits a given data.

**References:**

**[1]** Wikipedia contributors. (2021, November 28). Symbolic regression. In *Wikipedia, The Free Encyclopedia*. Retrieved 23:14, December 12, 2021, from <https://en.wikipedia.org/w/index.php?title=Symbolic_regression&oldid=1057642689>

[2] M. O’Neill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open Issues in Genetic Programming. GPEM, 11(3/4):339–363, 2010.

[3] S. Luke and L. Panait. Is The Perfect The Enemy Of The Good? In Proc. GECCO. 2002.

[4] Ying Tan, in [Gpu-Based Parallel Implementation of Swarm Intelligence Algorithms](https://www.sciencedirect.com/book/9780128093627/gpu-based-parallel-implementation-of-swarm-intelligence-algorithms), 2016

[5] Koza, J.R., 1994. Genetic programming as a means for programming computers by natural selection. Statistics and computing, 4(2), pp.87- 112.

[6]Douglas Zongker lilgp1.01User’s manual. Michigan state university 27 March 1996

[7] Rao, R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int. J. Ind. Eng. Comput. 2016, 7, 19–34.