Using Genetic Programming to Implement Symbolic Regression

Caroline Thompson and **Rich Korzelius**
{cathompson, rikorzelius}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

In this experiment, we use genetic programming to implement symbolic regression. Genetic programming is a method that finds the most optimal solution among a population of many possible solutions by pruning the population at each stage and combining traits of individual solutions to produce more optimal offspring. Specifically, we use genetic programming to implement symbolic regression. Our goal is to use symbolic regression to determine a mystery function f. We do this by creating an initial population of various mathematical expressions and use genetic programming to find the best fit to the mystery function. We are still awaiting results to find the functions that best fit our data points.

1 Introduction

Genetic programming is a useful artificial intelligence problem because it enables computers to automatically find the most optimal solution to a problem using the evolutionary idea of natural selection and survival of the fittest. In this problem, we create a computer program to determine a mystery function by using symbolic regression. To do this, we use genetic programming to search for the best function that matches the mystery function. We assign each individual a fitness value and use that to determine which individuals survive for the next generation. To weed out weak individual functions and improve the optimality of each subsequent generation, we probabilistically select fit individuals to participate in three methods to produce the next generation: reproduction, crossover, and mutation.

We use methods such as reproduction, crossover and mutation to improve the fittest functions, as described by John Koza (**?**). Although we use LEAST SQUARES?? to determine each individual's fitness value, other literature suggests that combining this method with linear scaling can produce better results (**?**). In future work, we would include this method in our experiments ad see how the output compares to our original results. Another issue that could hinder our results is premature convergence, in which the computer may cease searching for a solution before the optimal solution is determined. James Edward Baker suggests the use of adaptive selection methods such as a fixed selection algorithm or hybrid systems to prevent this problem (**?**).

We are still working on our results, which will be briefly summarized in this section.

In the following sections, we discuss the background of this problem, our experiment, and our results.

2 Background

We later describe how we execute the reproduction, mutation, and crossover methods to produce generations, but we use a method from Brad Miller and David Goldberg (**?**) called tournament selection to select individuals. In this method, we choose a sample of the population and select the winner to be the individual with the best fitness value. We insert this invidual into a "mating pool" of other tournament winners. Since the average fitness value of the mating pool is higher than that of the general population of individuals, we can use this difference to calculate the selection pressure, i.e. the amount that fitter individuals are preferred. This ensures that subsequent generations have higher average fitness values.

3 Experiments

To find a mystery function using genetic programming, we first generate an initial population of possible functions. To do this, we randomly create functions expressed by the arithmetic operators +, -, *, /, positive integer powers of x, and integer constants. We choose an initial population size of 300 individuals and a maximum depth of 10 for each individual tree. We choose these parameters because they are large enough for us to get a large variety of possible functions, but small enough to run quickly and prevent memory issues. We only use integers from -5 to 5 for the same reasons. To prevent the possibility of functions calculating negative exponents, we take the absolute value of each integer power of x. When creating the initial population, each operator has an equal chance at being chosen for each equation. Thus, we have a wide variety of different functions.

Second, we give each function a fitness value to determine how closely its value matches the mystery function's value. For each individual, we evaluate the function value for a given x, and subtract this from the mystery function value of x. Then, we take the absolute value of this function and assign it as the individual's fitness value. We use the fitness value to determine the fittest individuals and choose individuals for crossover, mutation, and reproduction using tournament selection. IS THIS RIGHT –¿ Our program runs until an individual's fitness is as close to zero as possible, which indicated that we have found the mystery function.

Next, we select individuals for reproduction, mutation, and crossover. We decide on a 1 percent reproduction rate, 10 percent mutation rate, and 89 percent crossover rate. HOW DID WE DECIDE ON THESE? Rather than simply selecting the fittest individuals for these operations, we select individuals proportionately to their fitness values using tournament selection, as described previously. This prevents possible diversity issues from arising because although fitter individuals are more likely to be chosen, less-fitter individuals also have a chance at being chosen. The first operation to be executed is reproduction. In this method, we select an individual and simply copy it into the next generation. The next operation is mutation. We select an individual and then randomly choose a node in the tree to mutate. Once we select a node, we delete the subtree after that point and reconstruct another tree using the same method we use to generate the initial population. Finally, we execute the crossover operation. In this method, we select two individuals and randomly choose either the right or left subtrees in each individual. Then, we swap the trees at these points in the two trees we selected.

After executing these operations a number of times, the trees get fitter and the fitness values get closer to zero, so we get closer to finding the mystery function with each generation.

We are also tasked with performing symbolic regression on a file of 100,000 data points with measured values of three x-values. Although we are unable to produce cohesive results, we explain how we would like to execute this regression. We continue to use the same parameters as in the first regression. However, to prevent overfitting, we separate the data file into a test file with 20,000 data points and a training file with 80,000 points. Next, we calculate the fitness values of the individuals in the training set and determine the 10 fittest individuals. We then use these individuals with the data from the test set to find the optimal solution.

4 Results

We are working on our results, which will be presented and analyzed in this section.

5 Conclusions

In this paper, we use genetic programming to implement symbolic regression and determine a mystery function. We use reproduction, mutation, and crossover to improve each generation's fitness and find the individual that best represents the mystery function. We are working on the results, which will be briefly summarized in this section. In further studies, we would expand our function span to include trigonometric functions, non-integer powers of x, and non-integer constants. We would also like to experiment with different selection methods, such as hybrid systems to prevent premature convergence, as mentioned previously in this paper.