Find Arithmetic Formula by GP

JANUARY 20

Saeed77t@gmail.com / student number: 40160957
Authored by: Reza Tahmasebi

Define problem

This homework aims to get any formula from input and try to reach the formula with genetic programming.

About Genetic programming

Genetic programming (GP) is an evolutionary algorithm (EA), a subset of machine learning. EA is used to solve problems that humans cannot directly solve. Free from human prejudices and prejudices, EA's adaptability allows it to produce solutions that match and often outperform human best efforts.

Inspired by biological evolution and its underlying mechanisms, the GP software system implements algorithms that solve user-defined tasks using random mutations, crossovers, fitness functions, and multigenerational evolution. To do. GP can be used to discover functional relationships between features in data (symbolic regression), group data into categories (classification), and aid in designing electrical circuits, antennas, and quantum algorithms. GP is applied to software development through code synthesis, gene enhancement, automated debugging, game strategy development, and more.

Implementation

In this section, the implementation of the algorithm is discussed. First, the terminal and function sets were defined; the terminal set contained 'x.' and digits one to nine, and the function set is four leading math operators (+ - * /).

Initial first seed

To create the first population, the *InitalEq* function is used. This function gets terminal and function sets as arguments and makes a random equation with them; the process has a parameter to control the length and depth of the equation.

Because GP uses the tree as the chromosomes, we used parentheses to show the tree and subtrees. This function will be called in a *for* loop to create the initial population, like the picture below.

```
[['(x+6)', 0],
 ['((4-3)/(3/4)*(x*3)/(x/x))', 0],
 ['((9+x)+(x+x))', 0],
 ['((x+x)-(6-x)-(x-(x-x)))', 0],
 ['(x*5)', 0],
 ['((x/x)*((7/x)*x))', 0],
 ['x', 0],
 ['3', 0],
 ['((x*3)+(x*x)*(4*(x-7)))', 0],
 ['((x/x)/((4*x)/x)-((x-x)-9))', 0],
 ['((7/1)-(9-2)+(x+x)*((x*x)*x))', 0],
 ['(3/4)', 0],
 ['5', 0],
 ['((x-8)+(x+x))', 0],
 ['(x-x)', 0],
 ['8', 0],
 ['((x+x)/(4/x)/(1/x)*((x+x)*9))', 0],
 ['((6*x)*(6*7)/(3/x)-(2-x))', 0],
 ['((x+5)/(1/x))', 0],
 ['((2/x)*(x*9))', 0],
```

Example of the first seed

Fitness function

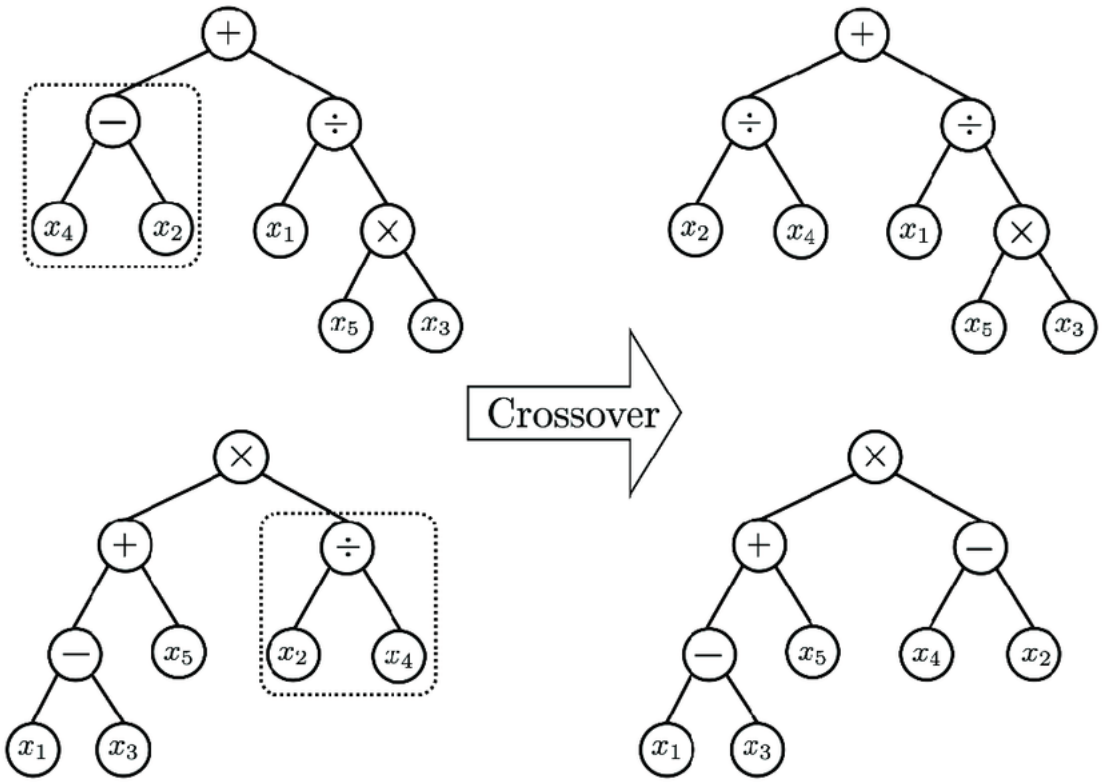
The straightforward fitness function is used to calculate each chromosome's fitness. In this method, a set of numbers will be considered and get some answers with our input equation, and then we use the same set of numbers to get answers with the chromosome, and then the number of equal solutions will be the fitness.
***CalcFitness* function is used to calculate the fitness of the function; this method gets population, numbers, and the input equation as the arguments and adds the fitness to the end of the chromosome. Also, this function will apply a penalty for chromosomes with very high lengths to prevent bloat.**

Parent selection

The *ParentSelection function* divides the population into two groups based on their fitness with a specific percentage and ensures that 80% of the parents will select out of group one.

Recombination

This algorithm uses crossover as the recombination operator, such that a part of the parent's trees (subtree) will be cut, and the subtree of parent one will append to parent two. The sub-tree of parent two appends to parent one. *Xover function* is used to do such a thing in this algorithm, in a way that parenthesis in the chromosome representation is one of the subtrees of the chromosome, and the function calls *CutBox* function to get the location of the chromosome to cut. Two control operators are used in the *Xover method to prevent the singularity of the* parenthesis and one to control the number of parentheses in the chromosome.



Schema of the cross-over in the GP algorithm

Mutation

Two types of mutation are used in this algorithm, *mutation* function that randomly changes one of the terminals or operators of the chromosome, and *mutation2, which* randomly changes a sub-tree of the chromosome or appends a random subtree at the end of the chromosome.

Survival selection

***The survivalSelection* function uses a steady-state method, with one percent elitism which guarantees that a good chromosome of the previous generation stays in the next generation.**

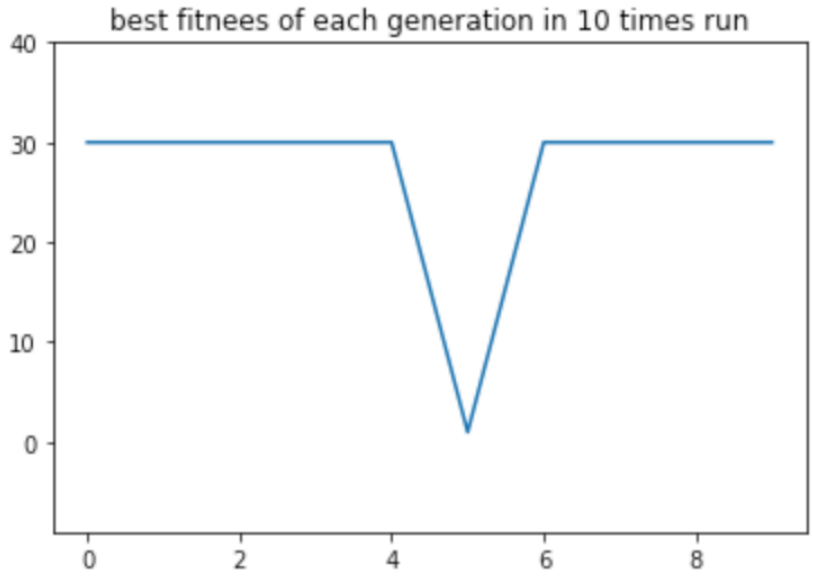
The results

The algorithm ran ten times for input equation 'x*x+x+x+1-4', and the results are as follows:

in run 1 solution found in iteration : 0 ['((x*x)+(x+x)-3)', 30]
Run : 2 ,Iteration : 1 ,rank1 = ['((3+2)-(x-x)-(8-x)*(9*5))', 1]
Run : 2 ,Iteration : 51 ,rank1 = ['((((4*9)+(x*x)+1)))', 1]
in run 2 solution found in iteration : 68 ['(((4+x)+(x*x)+(x-7)))', 30]
Run : 3 ,Iteration : 1 ,rank1 = ['((x*x)+((5*9)+6))', 1]
in run 3 solution found in iteration : 43 ['((((x-3)+(x*x)+x)))', 30]
in run 4 solution found in iteration : 0 ['((7/7)*(x*x)+(x+5)-(8-x))', 30]
Run : 5 , iteration : 1 ,rank1 = ['(((x+x)+(x+9)+((8-x)+7))-(x-(x*x)))', 1]
Run : 5 , iteration : 51 ,rank1 = ['(((x/x)+(1-x)*((x/6)*(3-x))))', 0]

in run 5 solution found in iteration : 59 ['((((x-3)+(x/x)*(x*x)+x)))', 30]
Run : 6 , iteration : 1 ,rank1 = ['(4*x)', 0]
Run : 6 , iteration : 51 ,rank1 = ['((((x-6)*(8+x)+x)))', 1]
Run : 6 , iteration : 101 ,rank1 = ['((((x-6)*(8+x)+x)))', 1]
Run : 6 , iteration : 151 ,rank1 = ['((((x-6)*(8+x)+x)))', 1]
run 6 solution is : ['((((x-6)*(8+x)+x)))', 1]
Run : 7 , iteration : 1 ,rank1 = ['((4*x)/(4/5)*((4-x)/x))', 0]
Run : 7 , iteration : 51 ,rank1 = ['((((4-4)-((6/x)*(4-1)/(1+x)*x))-(x/x)))', 0]
in run 7 solution found in iteration : 62 ['(((x+(x*x)+(x-3))))', 30]
Run : 8 , iteration : 1 ,rank1 = ['((8*9)*((x+6)+x))', 1]
in run 8 solution found in iteration : 45 ['(((3+x)*(x-1)))', 30]
in run 9 solution found in iteration : 0 ['((x*x)+(2+x)-(5-x))', 30]
Run : 10 , iteration : 1 ,rank1 = ['((2+8)*((6-2)*(1/x)))', 0]
in run 10 solution found in iteration : 35 ['((((x+x)+(x*x)-3)))', 30]

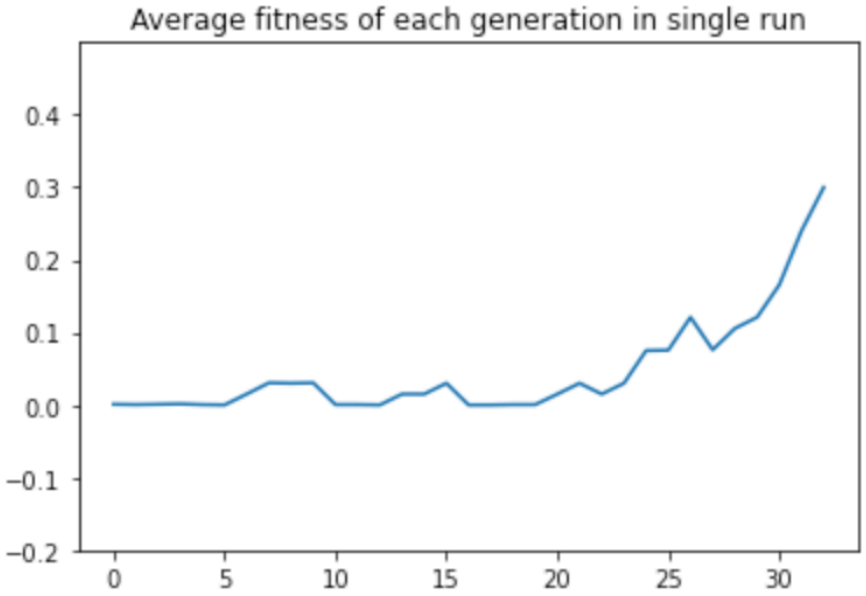
In two runes solution was in the initial population.
In one run, premature convergence happened, and the algorithm got stuck in local optima.
Seven times algorithm converged to global optima.



The plot of the best fitness of each time run

The results in a single run are as follows:

```
Itration :  1  ,rank1  =  ['((5*8)*((9*2)+x)+5)', 1]
AVG Fitt =  0.001
Itration :  11  ,rank1  =  ['(((x*(x/x)*x)+(x/4)+(3+2)))', 0]
AVG Fitt =  0.0005
Itration :  21  ,rank1  =  ['(((6-x)*(x/3)/(2+x)-(6/x)))', 0]
AVG Fitt =  0.015
Itration :  31  ,rank1  =  ['((((x-x))*(x-1)))', 0]
AVG Fitt =  0.1655
soloution found in itration  :  33 ['(((x+3)*(x-1)))', 30]
```



The average fitness of each population in a single run