Seminar 1:
Genetic Programming for Symbolic Regression

Intelligent Systems

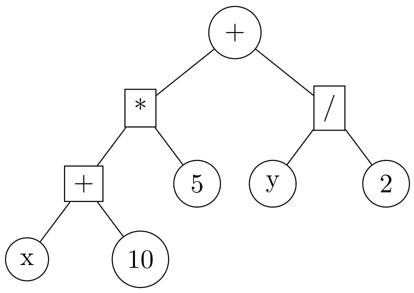
October 23, 2023

1 Introduction

In the first seminar assignment, the primary objective is to utilize genetic algorithms for evolving symbolic expressions (mathematical equations) to fit various sets of input-output pairs best. These equations are symbolically represented as trees, comprising operators such as $+$, $-$, $*$, $/$, and ˆ. Example of the equation

$$(x + 10) * 5 + \frac{y}{2}$$

represented as a tree:



Through genetic programming, the goal is to uncover the underlying mathematical formula that associates inputs to their respective outputs.

2 Dataset Examples

Given the following input-output pairs:

1. Example 1:

$$x : [1, 2, 3, 4, 5]$$
$$y : [2, 4, 6, 8, 10]$$

   Potential Equation: $y = 2x$

$$x : [1, 2, 3, 4, 5]$$
$$y : [1, 4, 9, 16, 25]$$

Potential Equation: $y = x^2$

2. Example 3:

$$x : [1, 2, 3, 4, 5]$$
$$y : [1, 0.5, 0.33, 0.25, 0.2]$$

Potential Equation: $y = \frac{1}{x}$

3 Task 1 - Representation (25%)

Develop a function to process the 2D representation of a dataset and yield the optimal symbolic expression, as derived by the genetic algorithm. The primary steps involve:

- Converting the dataset into an appropriate format. Hint: First, write a data-structure to parse the data into trees.
- Deciding on an effective solution representation.
- Designing a fitness function.
- Implementing and executing the genetic algorithm.

4 Task 2 - Crossover and Mutation (30%)

Standard crossover and mutation functions might not be ideal for evolving symbolic expressions, as they may produce invalid or nonsensical equations. Thus, it's pivotal to:

- Modify the crossover and mutation functions to ensure the generation of valid expressions.
- Consider the structure and constraints of symbolic expressions during mutation and crossover.
- Utilize existing genetic algorithm libraries, such as the pyGAD to suit this problem.

5 Task 3 - Complexity and Diversity (20%)

Introduce diversity in the types of equations and operators the genetic algorithm can evolve. This can include trigonometric functions, logarithms, and more. The goal is to allow the algorithm to explore a broader space of potential solutions, enabling it to tackle more complex input-output relationships. Additionally, when constructing the equations, try to retrieve the **shortest** and maximally **simplified** equations by **penalizing** long equations.

6 Task 4 - Evaluation and Report (25%)

Compile a comprehensive report detailing your approach, showcasing code highlights, and presenting the results. Prepare a single **notebook** for the results. It's crucial to:

- Compare the performance under different genetic algorithm configurations (e.g., different mutation rates, crossover methods, selection criteria).
- Assess the approach across varied datasets to understand its robustness and versatility.
- Visualize how the complexity of the dataset affects the algorithm's runtime using appropriate graphs.