Genetic Programming

Last Week: Genetic Algorithms

- Population based solution
- Used concepts from natural selection to evolve individuals
- Used properties of DNA to exchange and change information between individuals
- Individuals represented as lists
  - [$Gene_0$, $Gene_1$, $Gene_2$, …, $Gene_{N-1}$]

This Week: Genetic Programming

- Instead of taking an individual and having a function evaluator to obtain objective scores…

Individual

[0, 1, 0, 1, 0, 1] → Evaluator → 3

Objective Score

- The individual is the function itself

Input Data

[0, 1, 2, 3, 4, 5] → Individual → [0, 1, 4, 9, 16, 25] → Evaluator

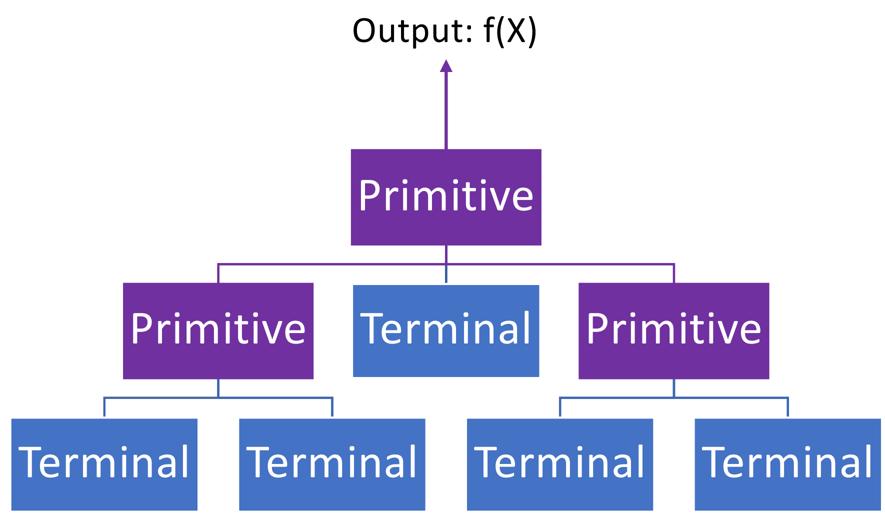
Output Data

What's the function?

$Y = X^2$

Tree Representation

- We can represent a program as a tree structure
  - Nodes are called primitives and represent functions
  - Leaves are called terminals and represent parameters
    - The input can be thought of as a particular type of terminal.
    - The output is produced at the root of the tree.

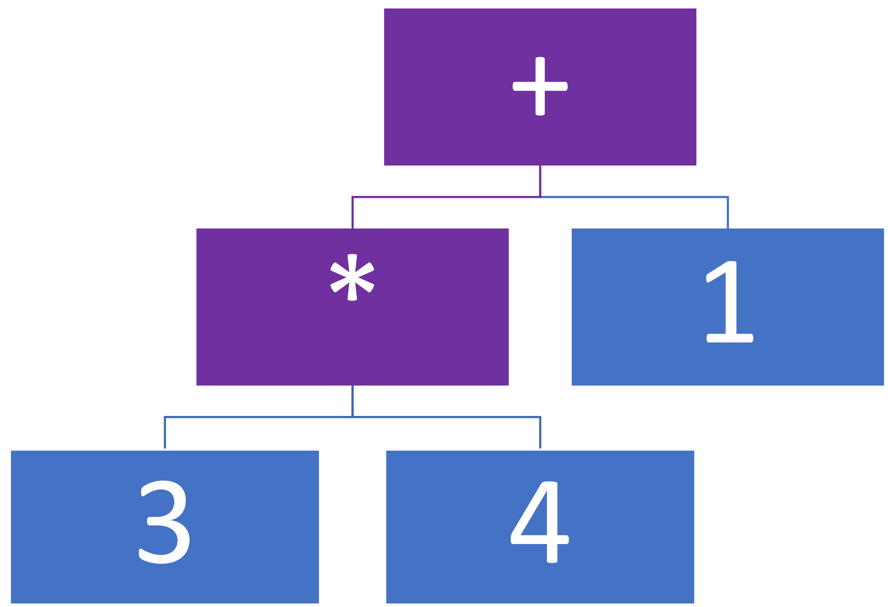


Tree Representation: Example

- What's the function?

f(X) = 3*4 + 1

Note: it's a constant



How is the Tree Stored?

- The tree is converted to a lisp preordered parse tree.
  - Operator followed by inputs
- The tree for f(X) = 3*4 + 1 can be written as:

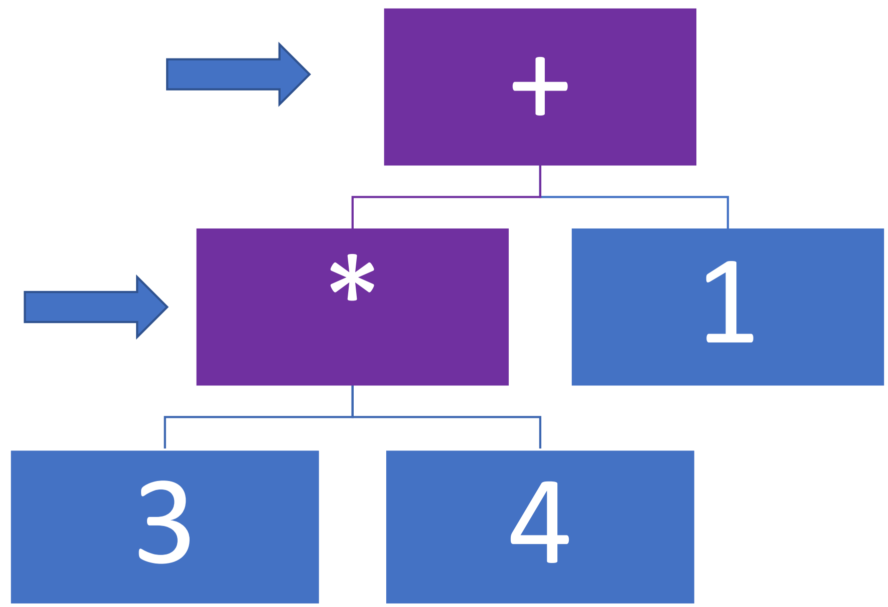
[+, *, 3, 4, 1]

- This comes from using the root first and then expanding:

[+, input1, input2]

[+, *, input3, input4, 1]

[+, *, 3, 4, 1]

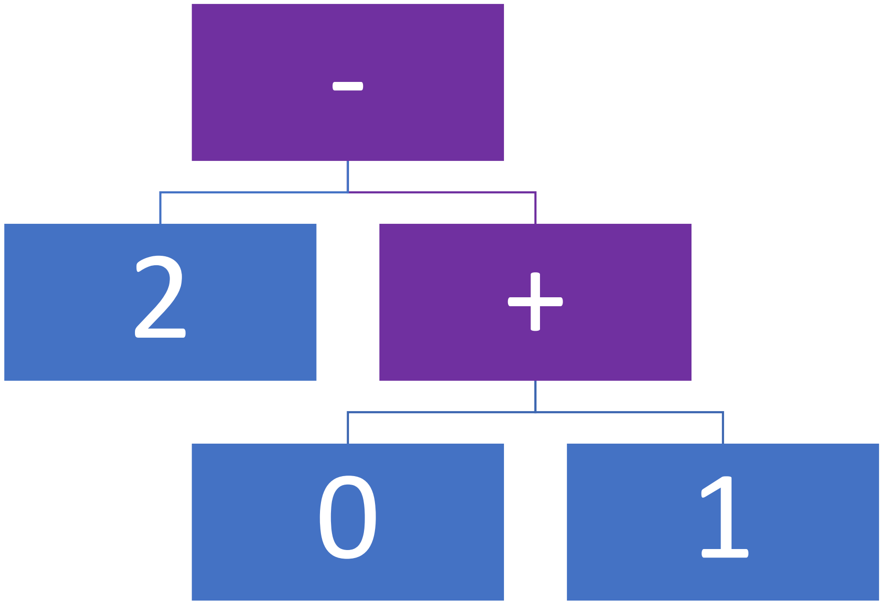


Tree Representation: Example 2

- What's the function?

f(X) = 2 – (0+1)

Note: it's a constant

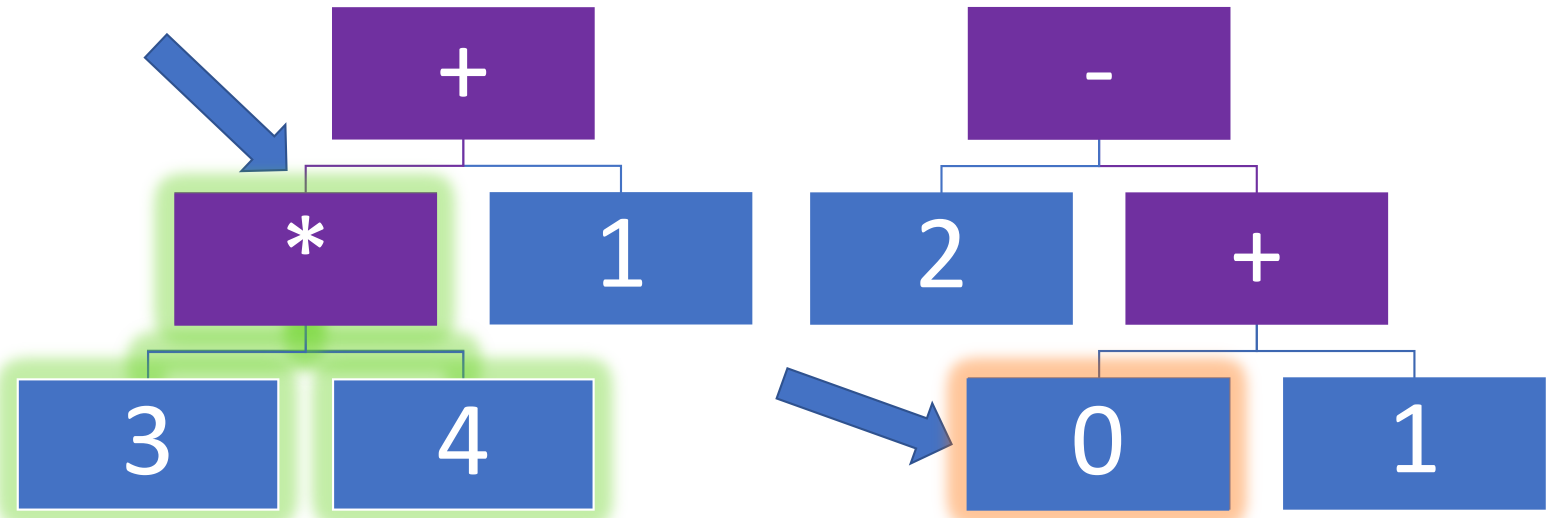


- What's the parse tree?

[-, 2, +, 0, 1]

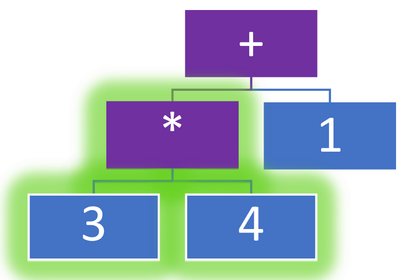
Crossover in GP

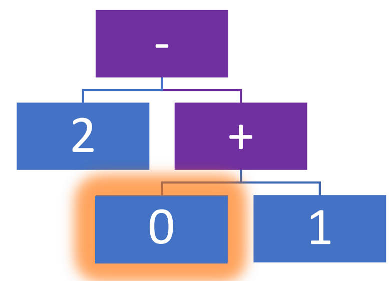
- Crossover in tree-based GP is simply exchanging subtrees
- Start by randomly picking a point in each tree
- These points and everything below create subtrees

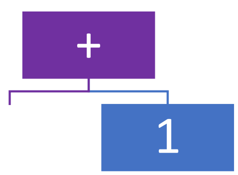


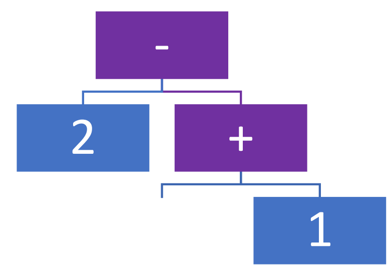
Crossover in GP Continued

- The subtrees are exchanged to produce children









- What are these new trees?

  f(X) = 0 + 1

  [+, 0, 1]

  f(X) = 2 – [(3*4) + 1]

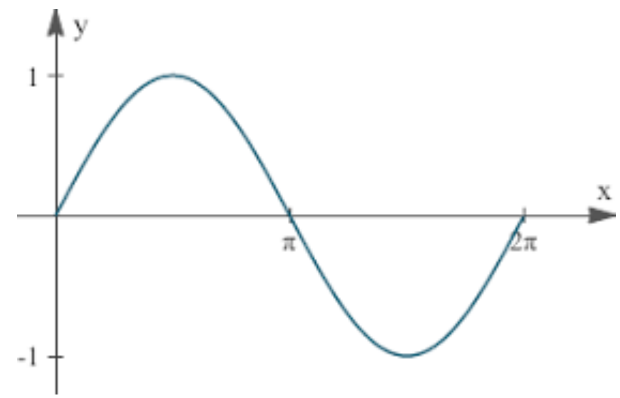
  [-, 2, +, *, 3, 4, 1]

Mutation in GP

- Mutation can involve…
  - Inserting a node or subtree
  - Deleting a node or subtree
  - Changing a node

Example: Symbolic Regression

- Using simple primitives, use genetic programming to evolve a solution to y=sin(x)
- Primitives include:
  - +, *, -, /
- Terminals include integers and…?
  - X
- How did Calc I solve this?
  - Taylor series!



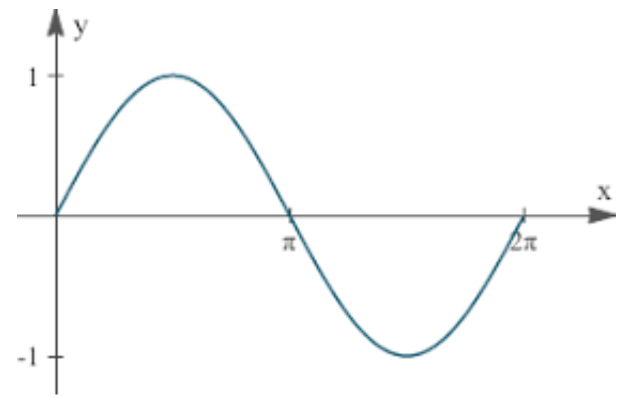
Example: Symbolic Regression

- Taylor series for sin(x)

$$\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k+1)!} x^{2k+1}$$

- What's the equation for third-order?

$$f(x) = x - \frac{x^3}{3!}$$



Third Order Expression

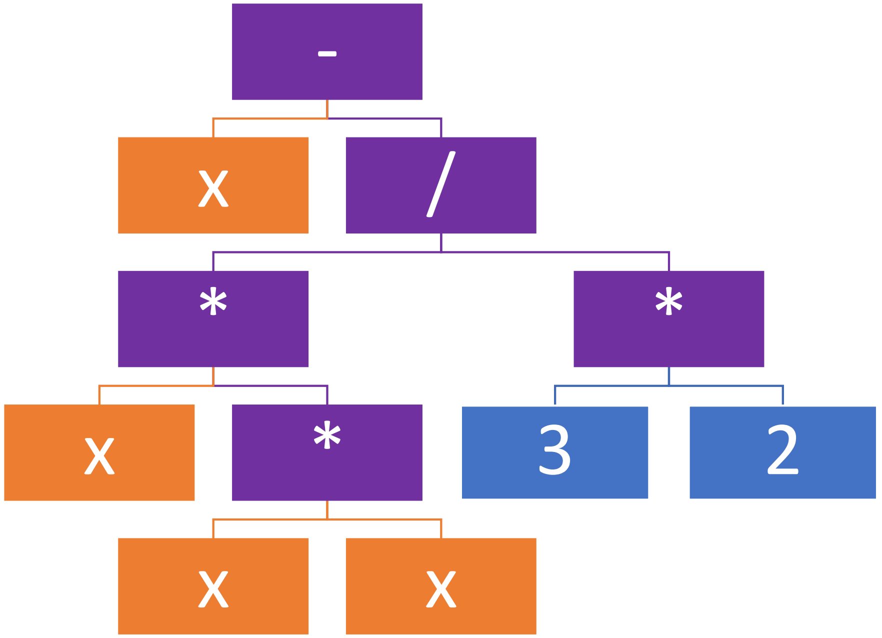
[-, input1, input2]

[-, x, /, input3, input4]

[-, x, /, *, input5, input6, *, input7, input8]

[-, x, /, *, x, *, input9, input10, *, 3, 2]

[-, x, /, *, x, *, x, x, *, 3 , 2]



Evaluating a tree

- We can feed a number of input points into the function to get outputs
  - $X = [0..2\pi]$
- Run f(X)
- We can measure error between outputs and truth, for example sum square error could be computed as

$$Error = \sum_i (f(x_i) - \sin(x_i))^2$$

What Primitives Could Make This Evolution Easier?

- Power()
- Factorial()
- Sin()
- Cos()
- Tan()

This is the idea behind EMADE!
But more on that later!