Genetic Programming and Symbolic Regression

Finding the laws in data

Manuel Pita | May 2020

UNIVERSIDADE
LUSÓFONA

Today

- Recap: Genetic Algorithms, recap
- Recap: As always, the way we represent knowledge is so important
- Recap: The *EvoSoup*: populations, fitness, fittest elite, offspring, crossover, mutation
- Side story: artificial consciousness
- The qualitative jump in representation again, Genetic Programming
- Operators, variables, coefficients
- Finding laws in data: symbolic regression
- Revision Q&A

The Dynamics of Genetic Algorithms

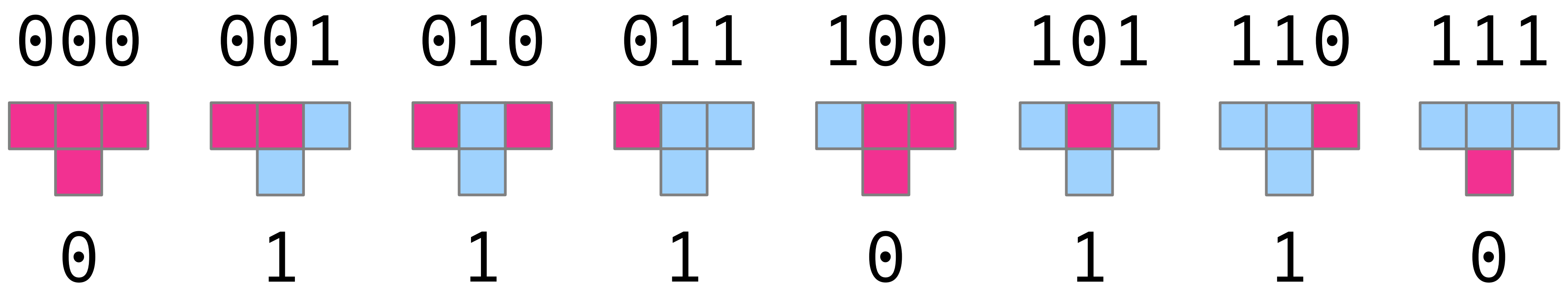
- Define a **genotype template** for solutions to a problem (like CA rules)
- **Create a seed population** of P random different such solutions
- **Give them a fitness value** from zero to one
- Fitness can be **single or multi-objective**!
- **Pick the best E individuals** and **clone them** to next generation
- Complete the **remaining P-E individuals** with **offspring** from the Elite
- **Allow** offspring to be subjected to some exogenous **mutation**
- Let **evolution** do its thing

Genotype representations

Don’t forget GAs operate on solutions that are represented in genotype templates

Individual parts (genes) where each gene can have a range of limited values

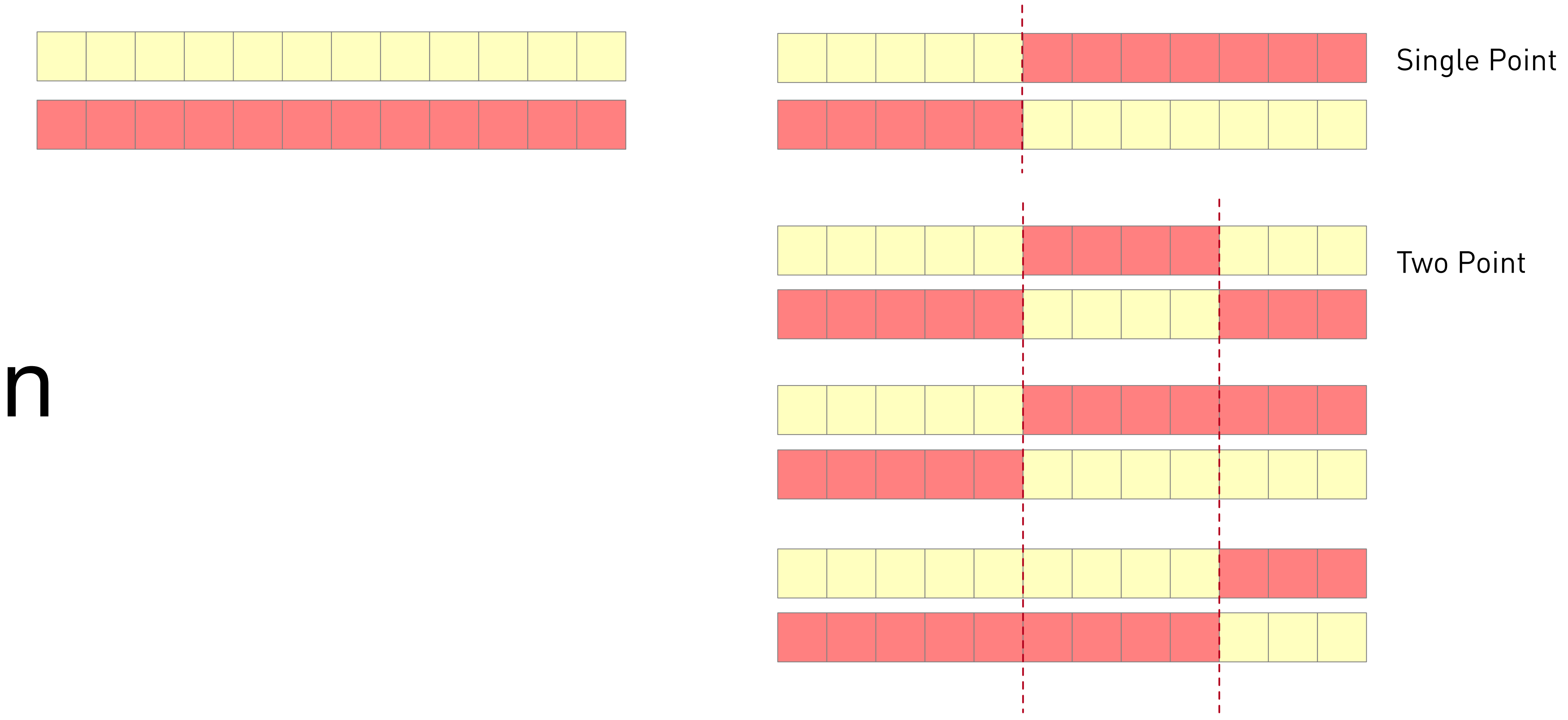
Cellular Automata rules have this property



What is the function of crossover?

Recombine best features

- Allow for the generation of new solutions that combine features from other existing good solutions (like it happens in natural selection)
- The production of **controlled diversity**
- This is how we make our search agent explore the vicinity of good solutions
- If we increase the number of recombination points two parents can produce more offspring (make sure you can explain this with a graphical example)



What is the function of mutation?

Add external sources of diversity

- Exogenous population diversity
- Diversity is the GA way to avoid getting stuck in local maxima/minima
- Otherwise if just a few good solutions are found early on, their genotypes will dominate the entire population.

Side Story

Exploring machine consciousness | Dr. Susan Schneider

- Did you know that…
- We don’t have conscious access to the part of our reasoning that is most computationally powerful
- David Chalmers: The hard problem of consciousness
- That consciousness is very slow. It feels, introspects, contemplates
- Subjective experience, our inner movie
- Most of what consciousness seems to do is not very computational (feeling, introspecting)
- There is a lot of debate: computational view / quantum mechanics view

https://www.youtube.com/watch?v=VypuNfR2GE0

https://en.wikipedia.org/wiki/Susan_Schneider

This is how machines may learn to program themselves

But still they need building blocks, an fitness functions

```
Operators = [ +, -, *, / ]
Numbers = [0-9]
Variables = [X,Y,Z]
```

We do not have fixed arrays of genes like in GAs. In Genetic Programming we work with operator trees that can have any structure

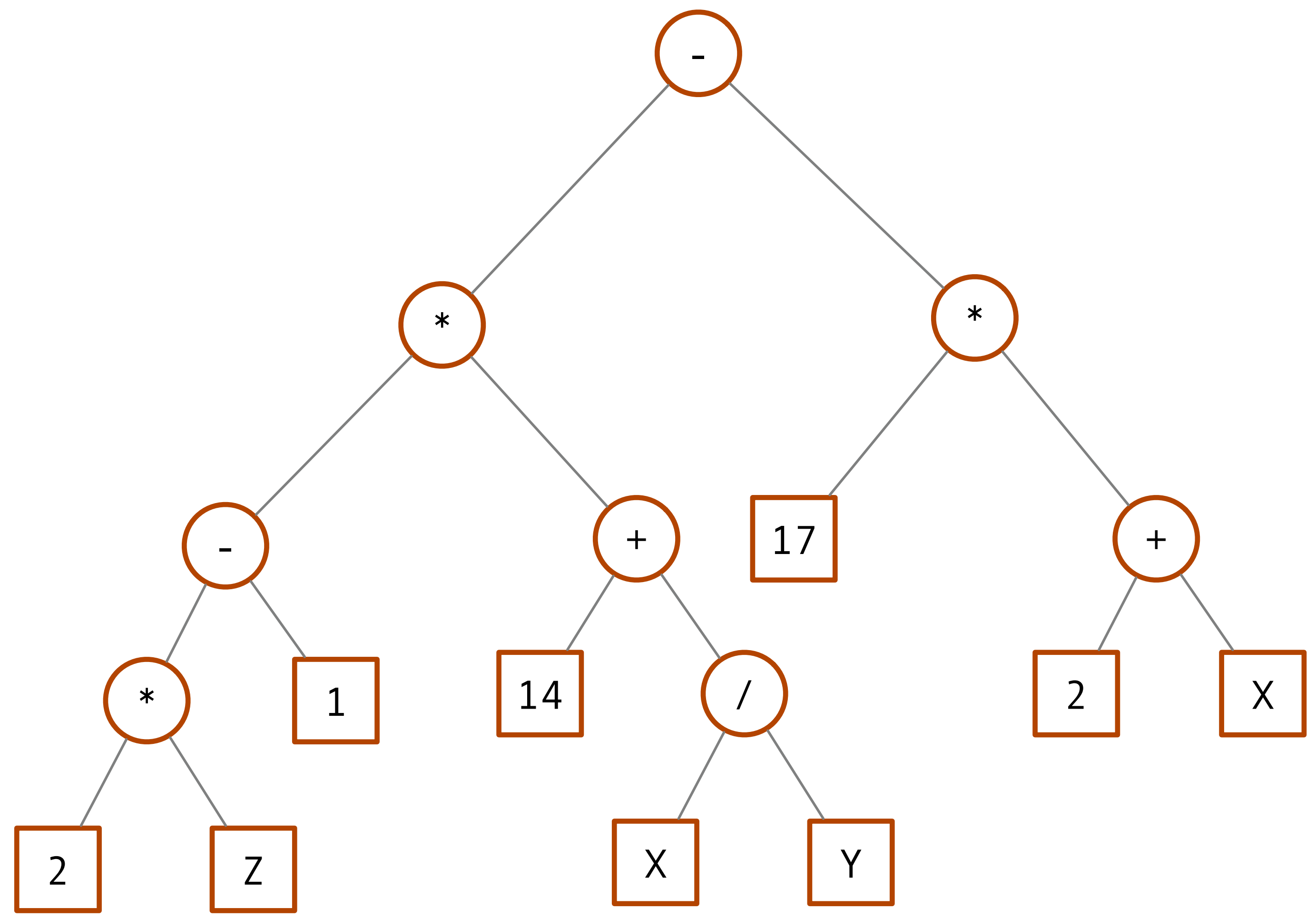
We can also represent programs like this

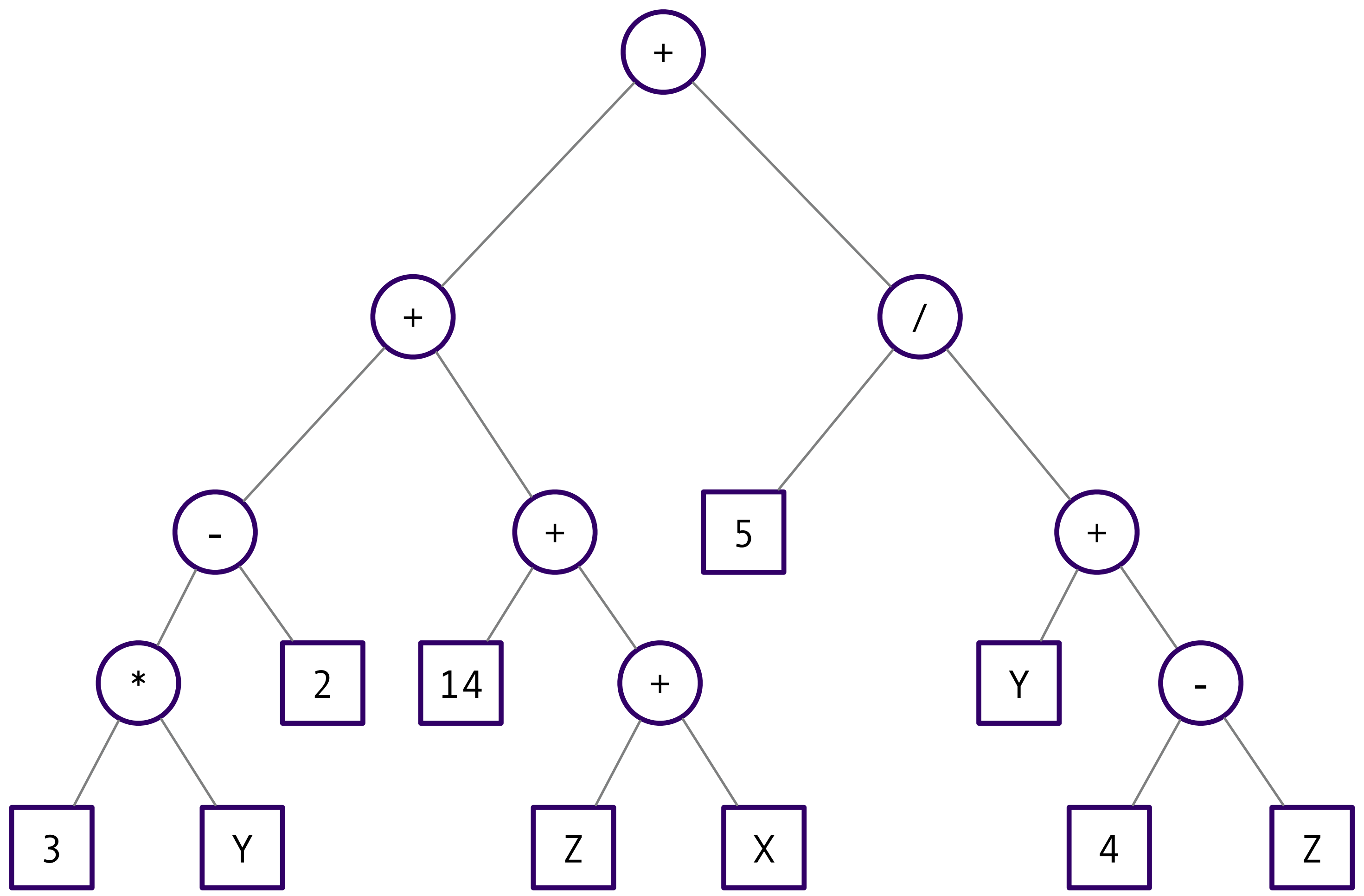
```
(17 * (2 + X) - (2Z -1) * (14 + (X / Y))
```

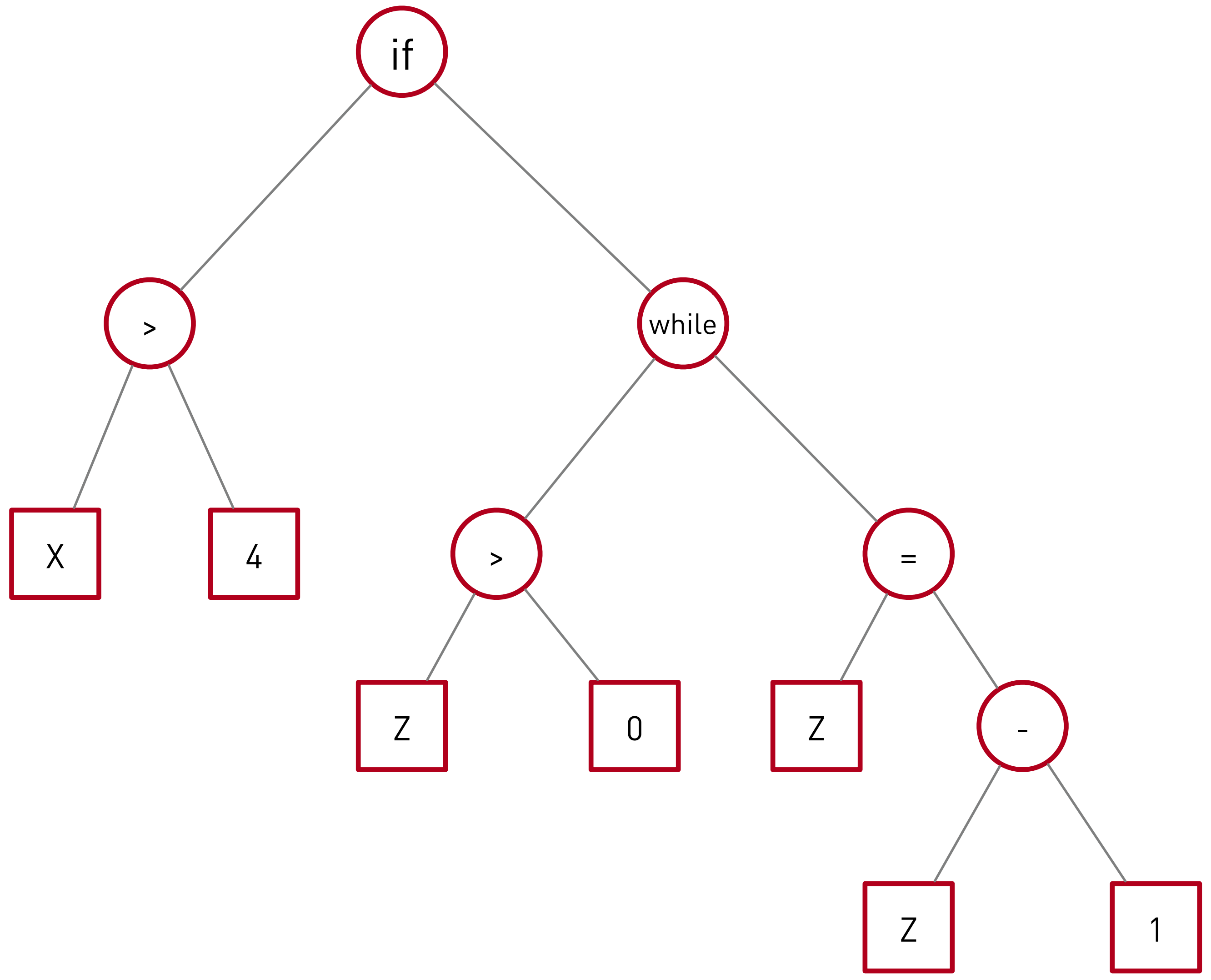
```
(- (* 17 (+ 2 X))
   (* (- (* 2 Z) 1)
      (+ 14 (/ X Y))))
```

```
(+ (/ 5 (+ Y (- 4 Z))
   (+ (- (* 3 Y) 2)
      (+ 14 (+ Z X))))
```

```
(if (< X 4)
   (while (> Z 0)
      (= Z (- Z 1)))
```

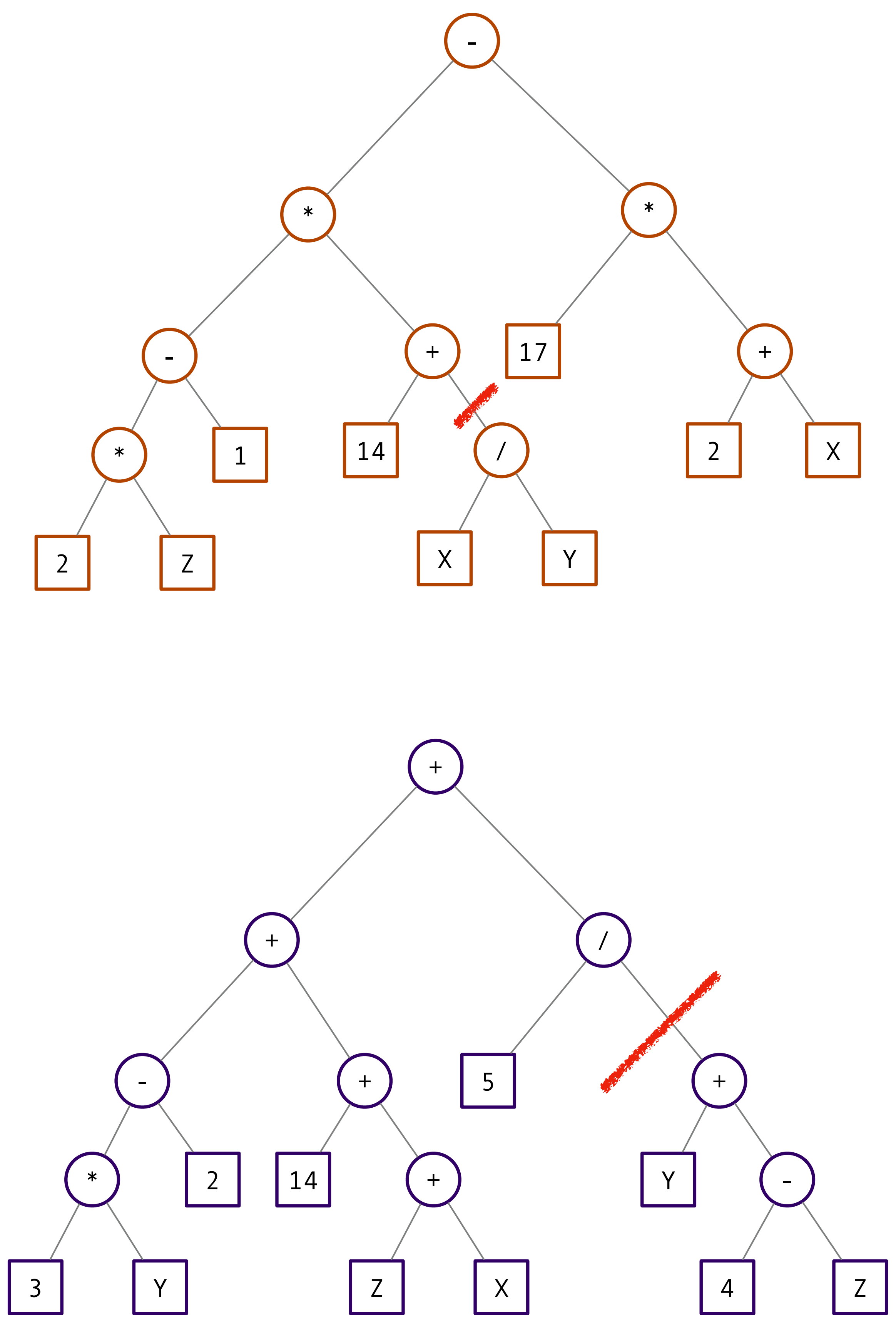






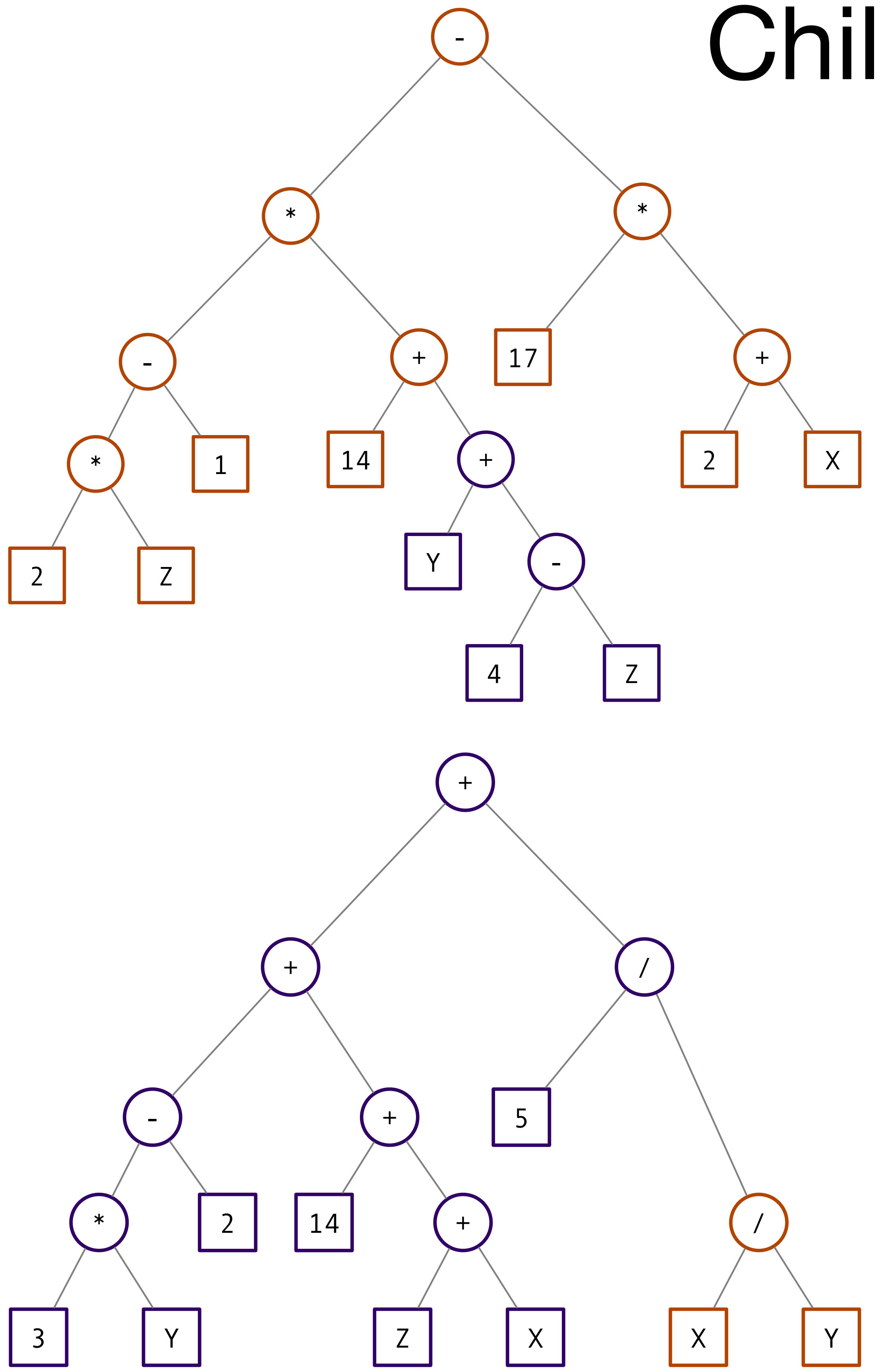
Tree-structure representation

Makes crossover and mutation easy to implement

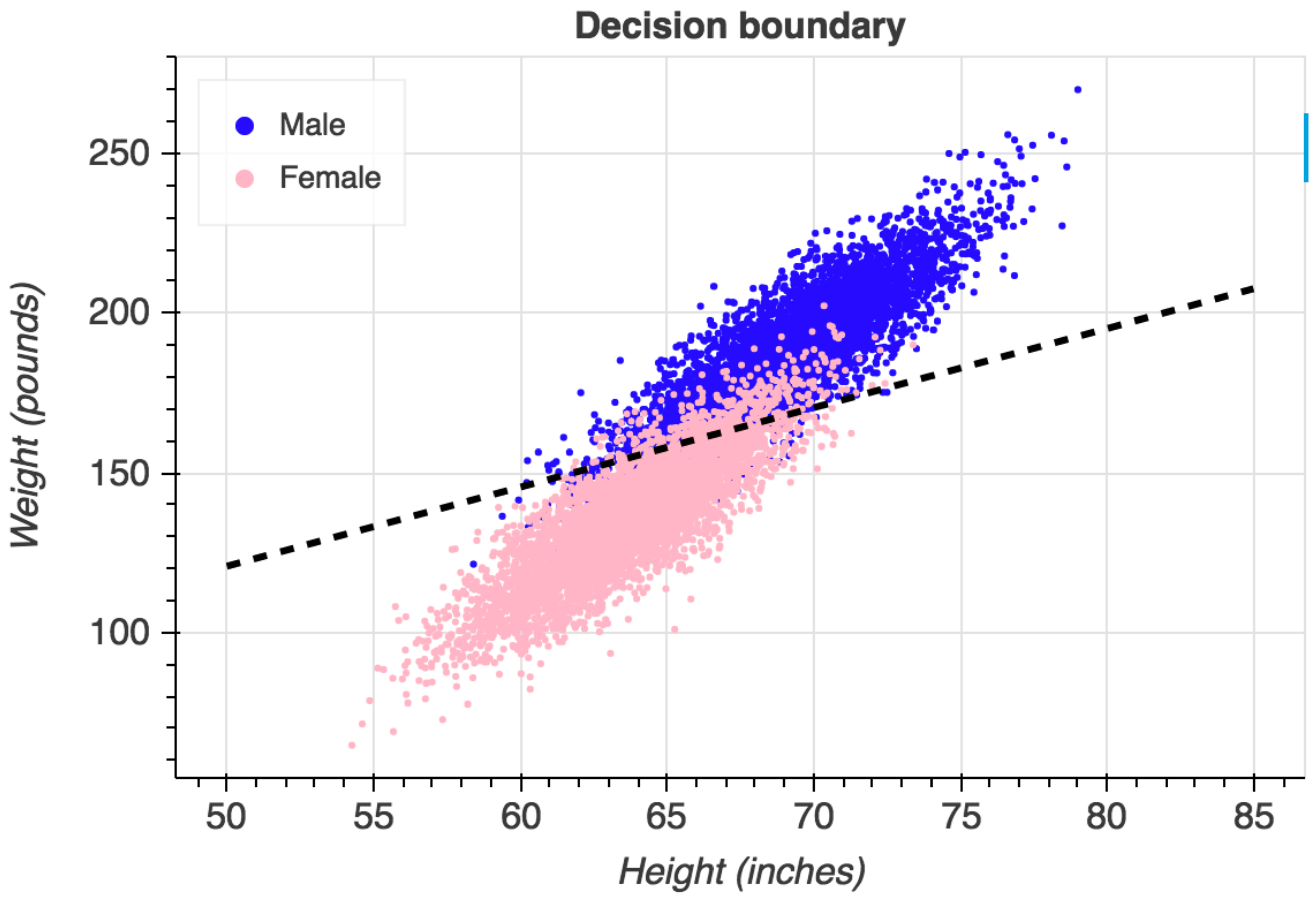


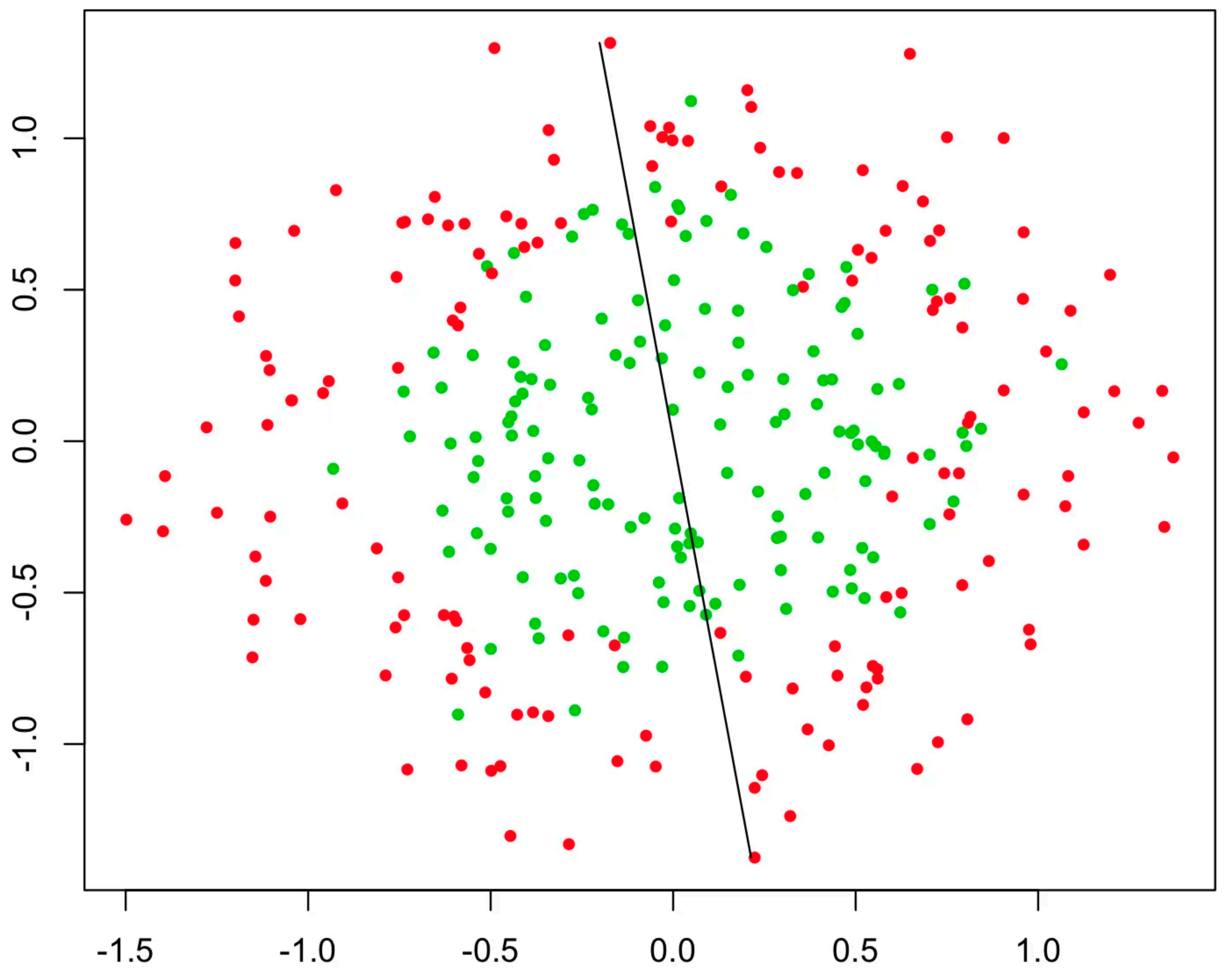
Parents

Children



Remember Logistic Regression?





Symbolic Regression

Finding the laws in data

- Sometimes we have no idea about the shape of the decision boundary
- Many times it is not straightforward linear, or quadratic, etc.
- Symbolic regression uses genetic programming to find good fits
- We will work with `gplearn`, a free, open source implementation

https://gplearn.readthedocs.io/

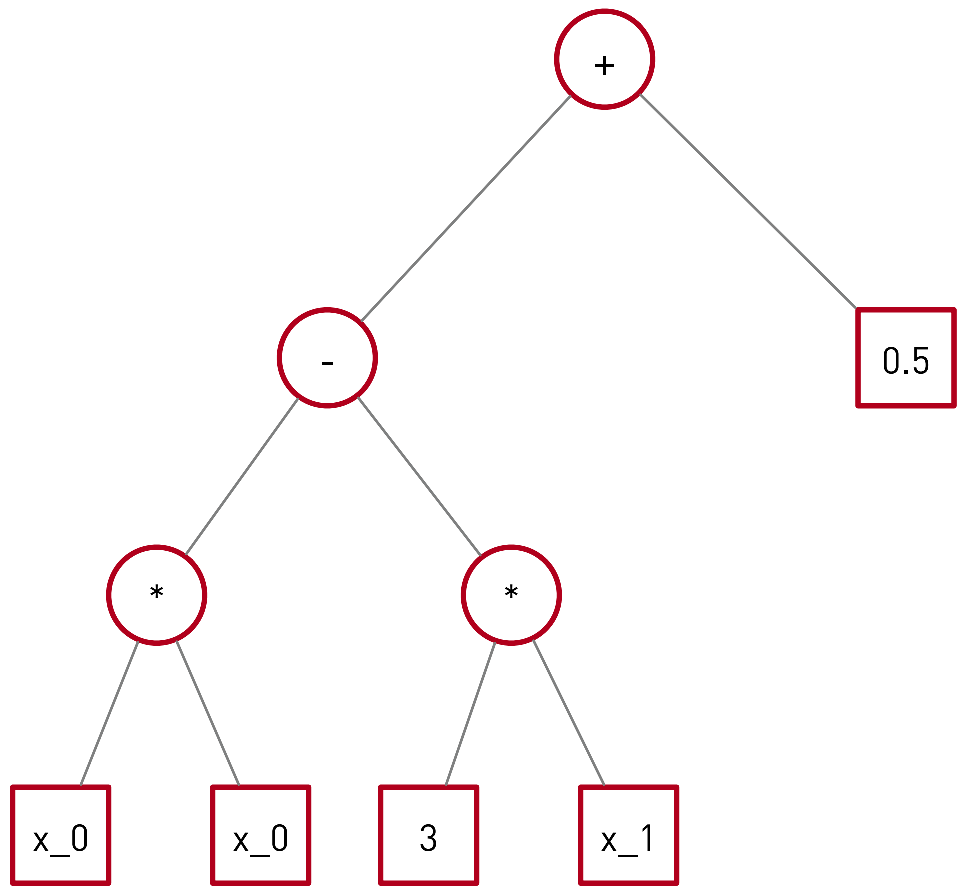
Representation

$$y = x_0^2 - 3x_1 + 0.5$$

$$y = x_0 \times x_0 - 3 \times x_1 + 0.5$$

$$y = (+\ (-\ (\times\ x_o\ x_0)(\times\ 3\ x_1))\ 0.5)$$

```
y = np.add(np.subtract(np.multiply(X0, X0), np.multiply(3., X1)), 0.5)
```



Operators and Closure

- ‘add’ : addition, arity=2.
- ‘sub’ : subtraction, arity=2.
- ‘mul’ : multiplication, arity=2.
- ‘div’ : division, arity=2.
- ‘sqrt’ : square root, arity=1.
- ‘log’ : log, arity=1.
- ‘abs’ : absolute value, arity=1.
- ‘neg’ : negative, arity=1.
- ‘inv’ : inverse, arity=1.
- ‘max’ : maximum, arity=2.
- ‘min’ : minimum, arity=2.
- ‘sin’ : sine (radians), arity=1.
- ‘cos’ : cosine (radians), arity=1.
- ‘tan’ : tangent (radians), arity=1.

GPLearn has methods to avoid evolution to test problematic formulae that includes, e.g. divisions by zero, square roots of negative numbers and similar

Sufficiency and initialisation

- Study the problem and determine how to bootstrap evolution
- Choice of the right operators
- Variable value ranges
- Standarisation
- Initialisation has rules similar to those in GAs, like population size
- But since GP representations are variable in size, we need to tell our program how deep can the trees be.

Homework

- How does GPLearn implement Selection and Evolution?
- In other words, how does it do Elite?
- Reproduction via crossover?
- Mutation?

https://gplearn.readthedocs.io/

For the MinTerm test

Turing machines, definitions, how it works

McCulloch and Pitts networks

Machines and State-Transition Diagrams

Uninformed Search: DFS/BFS

Informed Search Dijkstra and A*

Simulated Annealing and travelling salesman problem (theory only)