Machine Learning

Assignment #3

Due: Monday, February 20th

Genetic Programming

Since some students in this course do not yet know a functional programming language such as Lisp or Scheme, it's not feasible to assign a genetic programming project for this course. So instead, we'll do a pen and paper experiment together.

Consider the task of "learning" a Boolean circuit consisting only of AND and OR gates. To streamline our design, we will use only simple gates that take only two inputs each. Our goal is to learn a Boolean circuit that can decide the Majority-of-Three Problem.

The Majority-of-Three Problem is just as is sounds: given three inputs which we will call **1**, ***2*** and **3**, the output of our circuit should be TRUE if at least two of the inputs are TRUE and FALSE otherwise.

Experiment Design

FITNESS FUNCTION:

The fitness of each population member is the number of times it correctly produces the desired output for each of the 8 possible inputs <***A***, ***B***, ***C***>. In other words, the circuit is "run" for each and every combination of true/false values of ***A***, ***B*** and ***C***.

SELECTION:

We will use a modified version of rank order selection for this problem. At each generation, all chromosomes are ranked by their fitness. To start the next generation,

- The most fit individual gets copied into the next generation twice and will become chromosomes ***1*** and ***2*** in the next generation.
- The second fittest individual gets copied into the next generation twice and will become chromosomes ***3*** and ***4*** in the next generation.
- The third fittest individual gets copied once into the next generation and will become chromosome ***5*** in the next generation.
- The fourth fittest individual gets copied once into the next generation and will become chromosome ***6*** in the next generation.
- The two least individuals become extinct.

In case of fitness ties, the winner is the lowest numbered chromosome.
REPRODUCTION:

After selection, all population members are mated as follows.

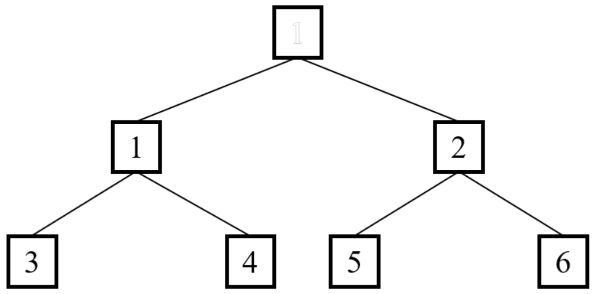
- Crossover occurs between *1* and *4*.
- Crossover occurs between *2* and *5*.
- Crossover occurs between *3* and *6*.

After crossover, all chromosomes are then mutated.

CROSSOVER:

Crossover occurs by selecting a locus in each of the two chromosomes, and then exchanging the subtrees with the gene at that locus as the root (see lecture slides).

For this experiment, we will number the possible crossover loci by doing a level-order traversal of a chromosome's tree as follows, skipping the tree's root:



You will keep a running a count of possible crossover loci starting with locus 1.

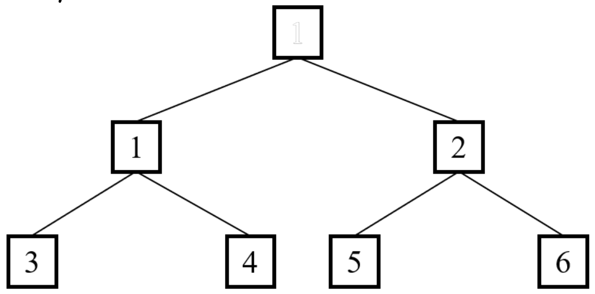
- The first crossover will use locus 1 of the first parent and locus 2 of the second parent.
- The second crossover will use locus 3 of the first parent and locus 4 of the second parent.
- The third crossover will use locus 5 of the first parent and locus 6 of the second parent, and so on…

When at any point there is no gene at a given locus, the running count is reset to 1 again.

For example, if chromosome *1* was (OR (AND 2 1) (AND 3 (OR 2 4))) and the running count was 4, then gene 1 would be selected as the crossover root in chromosome *1* and the running count would be incremented to 5. If chromosome *2* was (AND (OR 3 2) 1), the running count would be reset to 1 (as there is no gene at locus 5) and gene OR would be selected as the crossover root. The running count would then be incremented to 2. The resulting offspring would be the new chromosome *1* (OR (AND 2 (OR 3 2)) (AND 3 (OR 2 4))) and the new chromosome *2* (AND 1 1)

MUTATION:

Since we have no easy way to simulate random subtree generation by hand, we will instead use a simple form of gene mutation. As with crossover, you will keep a running count of possible mutation loci to try.



Each time you mutate a chromosome, you will need to first check if there is a gene at that locus. If not, you will need to reset the running count to 1 as above.

Note: this mutation running count is separate from the crossover running count!

Mutation will then work as follows:

- AND is changed to OR
- OR is changed to AND
- 1 is changed to 2
- 2 is changed to 3
- 3 is changed to 1

INITIAL POPULATION:

1: (OR (AND 2 1) (OR (AND (OR 2 3) 1) (OR 1 3)))
2: (OR 1 3)
3: (AND (AND 2 3) (OR (AND 1 2) (AND 2 3)))
4: (OR (OR (OR 1 2) 1) (AND 1 3))
5: (AND (AND 1 3) 2)
6: (OR (AND 1 1) (OR 3 2))

ASSIGNMENT:

1. Show each of the first 10 generations of this genetic programming experiment. It's probably easiest to draw the trees for each chromosome if you're not fluent in Lisp.
2. What issues did you encounter with this exercise? You may need to consult the genetic programming literature. I am expecting an answer worthy of a graduate student, and not just a sentence or two.
3. What could be done to improve the performance of this simulation? I am expecting an answer worthy of a graduate student, not just a sentence or two or something copied from the web. Think about your answer to 2., and what you could do to address the issues you encountered.