Genetic Algorithms & Genetic Programming

Lilly Spirkovska
CMPS290A; Feb. 17, 2000

Based on notes from Stanford University’s CS426: Genetic Algorithms and Genetic Programming, Winter 1998, taught by John Koza
Genetic Algorithms
Motivation

✔ Random Search
  - Does not use acquired information to direct the search

✔ Hill Climbing & Gradient Descent (Ascent)
  - Use acquired information
  - Prone to getting trapped in *local* optima

✔ Genetic Algorithms
  - Use acquired information to direct the search
  - Explore more of the search space so less chance of getting trapped in a *local* optimal
The genetic algorithm transforms a set (population) of mathematical objects (typically fixed-length binary character strings), each with an associated fitness value, into a new set (population) of offspring objects by means of operations based on the Darwinian principle of reproduction and survival of the fittest and naturally occurring genetic operations, such as crossover (sexual recombination) and mutation.
Example

- Population size = 4
- Fixed-length character string, string length = 3
- Alphabet size = 2 \( \{0,1\} \)

<table>
<thead>
<tr>
<th>Generation 0</th>
<th>Fitness</th>
<th>Generation 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>$3</td>
<td>111</td>
</tr>
<tr>
<td>001</td>
<td>$1</td>
<td>010</td>
</tr>
<tr>
<td>110</td>
<td>$6</td>
<td>110</td>
</tr>
<tr>
<td>010</td>
<td>$2</td>
<td>010</td>
</tr>
</tbody>
</table>
Preparatory Steps

- Determine the representation scheme
  - Structure (e.g. fixed-length), alphabet size
  - Mapping from search space to problem space
- Determine the fitness measure
- Determine the control parameters
  - Population size, number of generations
- Determine the method for designating a result (e.g. best-so-far) and the criterion for terminating a run.
Hamburger Restaurant Problem

Problem: Assuming no knowledge of the hamburger business, find the management strategy for operating restaurants to maximize profits.

Three binary decisions:
- Hamburger price: either $10.00 or $0.50
- Accompanying drink: either wine or coke
- Restaurant ambiance: either leisurely service with waiter in tuxedo or fast service
Step 1: Representation Scheme

✔ Fixed-length string of length 3, alphabet size 2

✔ Mapping
  - Left bit: Price
    • 0 = $10.00
    • 1 = $0.50
  - Middle bit: Drink
    • 0 = wine
    • 1 = coke
  - Right bit: ambiance
    • 0 = leisurely service with waiter in tuxedo
    • 1 = fast service
Step 2: Fitness Measure

✔ Profit in dollars
  - $0 - $7 derived from the strategy used

  - Ex: strategy       fitness
    • 010     $2
    • 101     $5
    • 111     $7
Step 3: Control Parameters

✔ Major parameters:
  – Number of restaurants to experiment with = 4
  – Max number of trials to run = 6

✔ Secondary parameters
  – Probability of restaurants crossing strategies = 50%
    {2 members out of size 4 pop.}
  – Probability of a restaurant slightly modifying its strategy = 1
    decision per “generation”
Step 4: Termination Criterion & Result Designation

✔ Termination
  - Global maximum attained ($7) by an individual restaurant or
  - 6 generations have been run

✔ Result Designation
  - Best-so-far strategy from population
Basic Genetic Algorithm

✔ Darwinian reproduction
✔ Crossover (sexual recombination)
✔ Mutation (only occasionally)
✔ Pseudo-code for GA:

- \( t = 0 \)
- initialize population \( P(t) \)
- loop
  - \( t = t+1 \)
  - evaluate individuals in \( P(t-1) \) for fitness
  - select \( P(t) \) from \( P(t-1) \) using FPR (fitness proportionate selection)
  - perform crossover on \( P(t) \)
  - possibly perform mutation
GA Flow Chart

Run = 0
Gen = 0
Create initial random population for run

Termination criterion satisfied for run?

yes

Evaluate fitness of each individual in population

i = 0

Gen ++

no

i = M?

no

Designate result for run

yes

Select genetic operation

Pr

Select one indiv. based on fitness

Select two indiv. based on fitness

Select one indiv. based on fitness

Perform reproduction

Perform crossover

Perform mutation

Insert mutant into new population

Insert two offspring into new population

Designate result for run

End

Pr = prob. of reproduction
Pc = prob. of crossover
Pm = prob. of mutation
N = max number of runs
M = size of population;
Generation 0 = blind random search

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>011</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Worst</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Best</td>
<td>6</td>
</tr>
</tbody>
</table>
# Mating Pool Produced by FPR

<table>
<thead>
<tr>
<th>Gen 0</th>
<th>FPR</th>
<th>Mating Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 011 3</td>
<td>0.25</td>
<td>011 3</td>
</tr>
<tr>
<td>2 001 1</td>
<td>0.08</td>
<td>110 6</td>
</tr>
<tr>
<td>3 110 6</td>
<td>0.50</td>
<td>110 6</td>
</tr>
<tr>
<td>4 010 2</td>
<td>0.17</td>
<td>010 2</td>
</tr>
<tr>
<td>Total 12</td>
<td>Total 17</td>
<td></td>
</tr>
<tr>
<td>Worst 1</td>
<td>Worst 2</td>
<td></td>
</tr>
<tr>
<td>Average 3</td>
<td>Average 4.25</td>
<td></td>
</tr>
<tr>
<td>Best 6</td>
<td>Best 6</td>
<td></td>
</tr>
</tbody>
</table>

- Improved average fitness of population
- Improved worst individual
- Less diversity
- Nothing new

FPR: Fitness Proportionate Reproduction(= fitness value / total)
Fitness Proportionate Reproduction
Crossover Operation

✔ Two parental strings
  - Parent 1
    • 011
  - Parent 2
    • 110

✔ Interstitial point chosen at random
  - Bit 2
  - Fragment 1
    • 01-
  - Remainder 1
    • --1
  - Fragment 2
    • 11-
  - Remainder 2
    • --0

✔ Two offspring produced by crossover
  - Offspring 1
    • 111
  - Offspring 2
    • 010
Mutation Operation

✔ One parental string
  – Parent
    • 010

✔ Mutation point chosen at random
  – Bit 3

✔ One offspring produced by the mutation operation
  – Offspring
    • 011

✔ VERY occasional, maybe 1 bit per generation
### Generation 1 (maybe)

<table>
<thead>
<tr>
<th></th>
<th>Gen 0</th>
<th>FPR</th>
<th>Mating Pool</th>
<th>Gen 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>011</td>
<td>3</td>
<td>011</td>
<td>111</td>
</tr>
<tr>
<td>2</td>
<td>001</td>
<td>1</td>
<td>110</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>110</td>
<td>6</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>4</td>
<td>010</td>
<td>2</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>Total</td>
<td>12</td>
<td></td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>Worst</td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Average</td>
<td>3</td>
<td></td>
<td></td>
<td>4.5</td>
</tr>
<tr>
<td>Best</td>
<td>6</td>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

- Crossover 1,2
- Crossover 1,2
- Reproduction 3
- Mutation 4
GA’s Are Probabilistic

✔ Initial random population
✔ Probabilistic selection of participation for operations based on fitness
✔ Probabilistic selection of crossover or mutation points
✔ Probabilistic selection of operation
✔ Often probabilistic fitness measure
  • 70 ways of choosing 4 individuals out of 8 to create initial population
  • 6 ways of choosing 2 parents for crossover
  • 2 ways of choosing crossover point
  • 4 ways of choosing individual for mutation
  • 3 ways of choosing bit for mutation
  • Total of 10,080 ways just for doing these steps
Some Individuals Are Better

✓ “I think it would be a most extraordinary fact if no variation ever had occurred useful to each being’s own welfare…. But if variations useful to any organic being do occur, assuredly individuals thus characterized will have the best chance of being preserved in the struggle for life; and from the strong principle of inheritance they will tend to produce offspring similarly characterized. This principle of preservation, I have called, for the sake of brevity, Natural Selection.”

» Charles Darwin, 1859, in *On the Origin of Species by Means of Natural Selection*
The Art of GA’s

✔ Finding a chromosomal representation of the problem
✔ Finding a chromosomal representation with the right genetic linkages
✔ Finding a good fitness measure
Common Mistakes

✔ Population is much too small
✔ Mutation rate is much too high
✔ Excessive greed is introduced
  - There’s a tradeoff between exploitation (greed) and additional exploration
✔ Hand-crafted crossover operators cause mutation to be introduced for virtually every crossover
✔ Hand-crafted crossover operators are not in sync with the problem
✔ Misapplying the rules of thumb
  • 90% crossover; 9% reproduction; < 1% mutation
Genetic Programming
Automatic Programming

✓ “How can computers learn to solve problems without being explicitly programmed? In other words, how can computers be made to do what is needed to be done, without being told exactly how to do it?”

» Arthur Samuel, 1959
Genetic Programming

✓ “Genetic programming is automatic programming. For the first time since the idea of automatic programming as first discussed in the late 40’s and early 50’s, we have a set of non-trivial, non-tailored, computer-generated programs that satisfy Samuel’s exhortation: ‘Tell the computer what to do, not how to do it.’”

» John Holland, University of Michigan, 1997
Genetic Programming

✓ An extension of the conventional genetic algorithm in which structures undergoing adaptation are hierarchical computer programs of dynamically varying size and shape.

✓ The search space is the space of all possible computer programs composed of functions and terminals appropriate to the problem domain.
Terminals & Functions Requirements

✔ Sufficiency requirement: set of terminals and functions together must be capable of expressing a solution to the problem.

✔ Closure requirement: each of the functions in the function set should be able to accept, as its arguments, any value that may possibly be returned by any function in the function set and any value that may possibly be assumed by the terminal in the terminal set.
Preparatory Steps

✔ Determine the representation scheme
  - set of terminals (ex: $\{x, y, z, \mathbb{R}\}$)
  - set of functions (ex: $\{=, +, -, *, /\}$)

✔ Determine the fitness measure

✔ Determine the parameters
  - Population size, number of generations
  - Number of atoms in program
  - Probability of crossover, mutation, reproduction

✔ Determine the method for designating a result and the criterion for terminating a run
Initial Random Population

☑ Create programs at random
☑ Example: $y + 0.3 + x - z \times 0.9$
☑ Tree representation

```
         +
        /|
       / +
      +--|
           x
      /|
     / -
    +--|
     /   
    +    
   z 0.9
```
Fitness Measure

✔ Varies with the problem.
✔ Fully defined: capable of evaluating any computer program that it encounters in any generation of the population.
✔ Measured over a number of different fitness cases.
  – Chosen at random over a range of values of the independent variables, or
  – Chosen in some structured way, e.g. at regular intervals over a range of values of each independent variable.
Fitness Measure Examples

✔ Error between the result produced by the computer program and the correct result.

✔ Determined from the consequences of the execution of the program.
  - Controller fitness based on amount of time (fuel, distance, money, etc.) it takes to bring the system to a target state.

✔ Amount of points scored
  - Food eaten, work completed, cases correctly handled, etc.
Fitness Measure Examples, cont.

✔ Number of patterns classified correctly and incorrectly
  – True positives, true negatives, false positives, false negatives.

✔ Multi-objective fitness measure
  – Incorporate a combination of factors, such as correctness, parsimony, or efficiency.
New Generations

✔ The initial generation (gen 0) will generally have very poor fitness.
✔ Some individuals will be somewhat more fit than others.
✔ These differences will be exploited using crossover and mutation operations.
Crossover Operation

✔ Select two parents based on fitness
✔ Randomly pick a node independently for each parental program
✔ Swap them
Crossover Example

Parent 1:

```
+  
/   
+  -
|   |
y 0.3 x *
  
z 0.9
```

Parent 2:

```
+  
/   
+  -
|   |
z 1.9 y x
  
```

y 0.3 *

x

z 1.9
Crossover Example

Parent 1:
+   +
  +   -
 y 0.3 x
 z 0.9

Parent 2:
+   +
  -   *
 z 1.9 y x
Crossover Example

Parent 1:
+  
  +  
  +  
  y  0.3  x

Parent 2:
+  
  -  
  -  
  z  1.9  y

Offspring 1:
+  
  +  
  +  
  y  0.3  *  
  +  
  +  
  *  
  y  0.3  y  x

Offspring 2:
+  
  -  
  -  
  z  1.9  x

  z  0.9
Mutation Operation

✔ Select a parent based on fitness
✔ Pick a point
✔ Delete the subtree at that point
✔ Grow a new subtree at the mutation point in the same way trees were generated for the initial random population
Mutation Example

Parent

+ 

- 

x 1.9 

* 

z 0.9
Mutation Example

Parent

+ 
  -  
  z 1.9
  - 
  * 
  z 0.9
Mutation Example

Parent

```
+  
-   -  
z   1.9  x  *
   
```

Offspring

```
+  
+   -  
+   
+   
+   
|   
|   
|   
|   
y 9  x  *
   
```

z 0.9
Examples of Genetic Programming

✔ Intertwined spirals classification
  - Find a way to tell whether a given point in the x-y plane belongs to the first or second of two intertwined spirals

✔ Artificial ant
  - Find a computer program to control an artificial ant so that it can find all 89 pieces of food located on the Santa Fe trail

✔ Truck backer upper
  - Find a control strategy for backing up a tractor-trailer truck to a loading dock

✔ Broom balancing (aka Inverted Pendulum)
  - Find a control strategy to balance the broom and bring the cart to a stop in minimal time
For More Information

