Lecture 8: Genetic Algorithms

Lecture 8, DD2431 Machine Learning

A. Maki

October 2013

Overview of GA

- GAs: Algorithms for search problems
- Pioneered by John Holland in the 1970's
  - Got popular in the late 1980's
- Solve combinatorial optimization
  - Suitable to overly complex problems where little is known about the search space
- Based on ideas from Darwinian Evolution
  - On the origin of species (Charles Darwin 1859)

Evolution in the real world

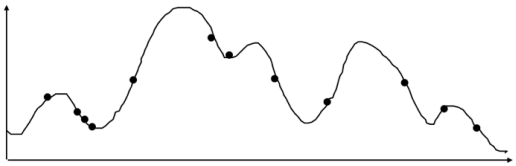
- Each cell of a living thing contains ***chromosomes*** - strings of *DNA*
- Each chromosome contains a set of ***genes*** - blocks of DNA
- Each gene determines some aspect of the organism (like eye colour)
- A collection of genes is sometimes called a ***genotype***
- A collection of aspects (like eye colour) is sometimes called a ***phenotype***
- Reproduction involves recombination of genes from parents and then small amounts of ***mutation*** (errors) in copying
- The ***fitness*** of an organism is how much it can reproduce before it dies
- Evolution based on "survival of the fittest"

(Slide: D. Hales)

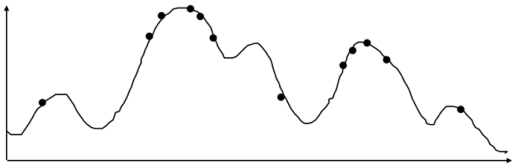
Basic framework of GA

- Generate an initial *set* of N random solutions
  - Test each solution in the set (and score them)
- Generate another (2nd) set of N solutions by
  - Duplicating some good solutions
  - Finding a new solution from a set of two
  - Making small changes to some of them
- Repeat for G times -> select the best solution

Toy Example



Distribution of solutions (individuals) in Generation 0



Distribution of solutions (individuals) in Generation G

Terminologies in GA

- Individuals: feasible solutions
- A population: a set of feasible solutions, size N
- Genotypes: encoded strings of solutions
- Genes: the individual bits of a string

- Fitness: the goodness of a solution
- Selection: selection among solutions
- Crossover, Mutation (names of operations)

Basic framework of GA (cont.)

- Generate an initial *set* of N random individuals
  - Test each solution in the set (and compute fitness)
- Generate 2nd set of N individuals: population
  - Duplicate some good solutions: selection (removing some poor solutions from set)
  - Find a new solution from a set of two: crossover
  - Make small changes to some of them: mutation

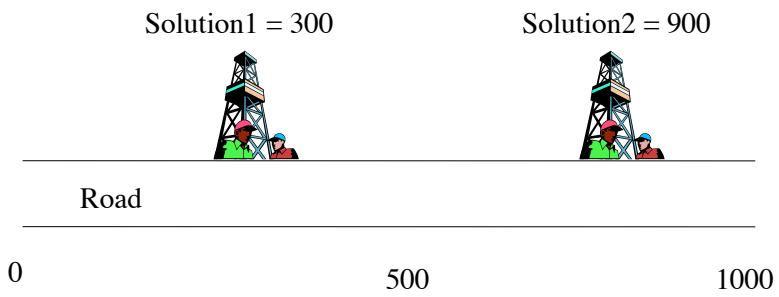
- Repeat for G times -> select the best solution

"Silly" Example - Drilling for Oil

- Imagine you had to drill for oil somewhere along a single 1km desert road
- Problem: choose the best place on the road that produces the most oil per day
- We could represent each solution as a position on the road
- Say, a whole number between [0..1000]

(Slides: D. Hales)

Where to drill for oil?



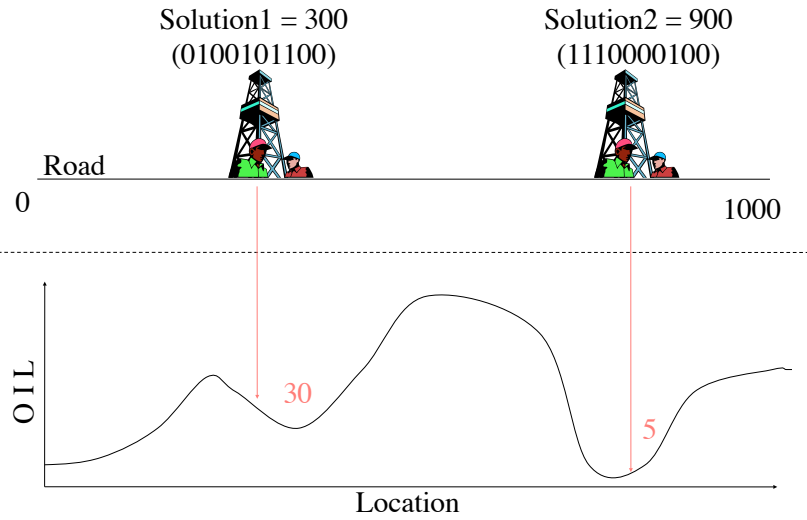
Digging for Oil

- The set of all possible solutions [0..1000] is called the *search space* or *state space*
- In this case it's just one number but it could be many numbers or symbols
- Often GA's code numbers in binary producing a bitstring representing a solution
- In our example we choose 10 bits which is enough to represent 0..1000

Convert to binary string

	512	256	128	64	32	16	8	4	2	1
900	**1**	**1**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
300	**0**	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**
1023	1	1	1	1	1	1	1	1	1	1

Drilling for Oil



Recap: components of a GA

- A. Encoding principles
  - represent feasible solutions as a number
- B. Fitness function
  - Each individual needs be tested and scored
- C. Mechanisms to generate a new population
  - initialization of population (often random)
  - operators: selection, crossover, mutation
- D. Termination conditions

A. Encoding principles

- Encoding each individual *often* as fixed length bitstrings (chromosome)
  - e.g. 101110, 111111, 000101
  - Each bit represents some aspect of the proposed solution to the problem
- Other integers, real numbers etc.
- Programing elements in tree structures
  - genetic programming

B. Fitness function

- Define "Fitness" to indicate how "good" that solution is for the given problem
  - error in a function approximation
  - performance of a simulated robot
- Definition of good fitness is the key to GA
- Ideally a simple design
  - evaluating the fitness function is the most time consuming part of a GA

C-1. Initialization of population

- Randomly generated solutions (individuals)
- Solutions provided by another heuristic
- Solutions given with some knowledge

C-2. Stochastic operators

- Selection
  - replicates the most successful solutions in a population according to their relative scores
- Crossover
  - decomposes two distinct solutions into pieces
  - randomly mixes their part to form novel solutions
- Mutation
  - randomly perturbs a candidate solution

Selection

- Roulette selection
  - Probability of survival proportional to fitness, f
  - May converge too early with dominant individual
- Ranking selection
  - Based on order of f
  - Sorting required at every generation.
- Elitism, Tournaments
  - The best individual prioritized to survive
  - Select individuals with high f in random subsets

Crossover

- Basic idea to produce two new offsprings
  - find a combination of good parts of the genotypes
- One-point crossover

  01001 | **11010** ⇒ 01001 **01011**

  10101 | **01011** ⇒ 10101 **11010**
- Two-point crossover

  010 | **01110** | 10 ⇒ 010 **01010** 10

  101 | **01010** | 11 ⇒ 101 **01110** 11
- Uniform crossover

  0 **1** 0 0 **1** 1 1 **0** 1 **0** ⇒ 0 **0** 1 0 **1** 1 1 **0** 1 1

  1 **0** 1 0 **1** 0 1 **0** 1 1 ⇒ 1 **1** 0 0 **1** 0 1 **0** 1 **0**

Mutation

- Make random changes to the contents of the chromosome at a mutation rate (e.g. 0.05)
- Explore unseen part of the search space
- To avoid for majority of population to get stuck in local minimum

D. Termination conditions

- New generation process can be stopped when reached a termination condition:
  - The highest ranking solution in the population is satisfactory
  - Fixed number of generations reached
  - No or little improvement observed between the best solutions in consecutive generations (reaching a plateau)

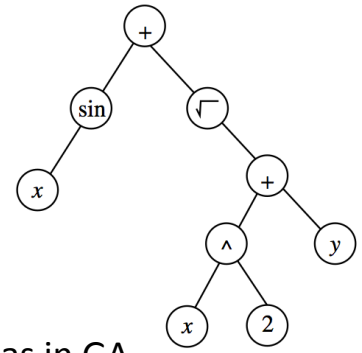
Pros and Cons of GA

- Benefits
  - simple concept, easy to understand
  - suited to parallel implementation
  - always an answer; solution gets better with time
- Issues
  - parameter setting; population size, operations ..
  - early convergence / local minimums
  - optimal answer not guaranteed

Genetic Programming (GP)

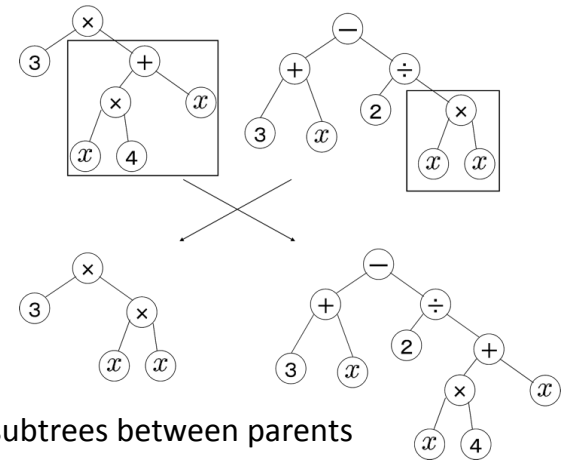
- Population of programs represented by trees (J. Koza, 1990)

$$\sin(x) + \sqrt{x^2 + y}$$



- Search methods same as in GA

Crossover in GP



Swapping subtrees between parents

Applications

Control	gas pipeline, pole balancing, missile evasion, pursuit
Design	semiconductor layout, aircraft design, keyboard configuration, communication networks
Scheduling	manufacturing, facility scheduling, resource allocation
Robotics	trajectory planning
Machine Learning	designing neural networks, improving classification algorithms, classifier systems
Signal Processing	filter design
Game Playing	poker, checkers, prisoner's dilemma
Combinatorial Optimization	set covering, travelling salesman, routing, bin packing, graph colouring and partitioning

(Table: W. Williams)

Further readings / reference slides

- Further readings
  - J.H. Holland, "Adaptation in Natural ..." 1975
  - D.E. Goldberg, "Genetic Algorithm in Search" 1989
- Reference slides
  - Genetic Algorithms: A Tutorial, W. Williams
  - Introduction to Genetic Algorithms, D. Hales http://cfpm.org/~david/talks.html