Genetic Algorithms & Genetic Programming

Senlin Liang

April 19, 2005

Outline

- Evolutionary Computation
- Basic GA
- An example: GABIL
- Genetic Programming
- Individual Learning & Population Evolution

Evolutionary Computation

- Computational procedures patterned after biological evolution. Operators:
  - –Inherit
  - –Crossover
  - –Mutation
- Based on probability theory

GA(Fitness, Fitness_threshold, p, r, m)

- Initialize: P ← p random hypotheses
- Evaluate: for each h in P, compute Fitness(h)
- While max(Fitness(h)) < Fitness_threshold
  - –Select: probabilistically select (1-r)*p members of P to add to Ps.
    - $Pr(h_i) = Fitness(h_i) / Sum(Fitness(h_k))$
  - –Crossover: probabilistically select r*p/2 pairs of hypotheses from P. For each pair, $\langle h_1, h_2 \rangle$, produce two offerspring by applying the Crossover operator. Add all offspring to Ps.

GA(Fitness, Fitness_threshold, p, r, m)

–Mutate: invert a randomly selected bit in m*p random members of Ps

–Update: P $\leftarrow$ Ps.

–Evaluate: for each h in P, compute Fitness(h)

- Return the hypothesis from P that has the highest fitness

Representing Hypotheses

•Represent

(Outlook = Overcast OR Rain) AND (Wind = Strong)

By	Outlook	Wind
	011	10

•Represent

IF Wind = Strong THEN PlayTennis = yes

By	Outlook	Wind	PlayTennis
	111	10	10

Note: Outlook: Sunny, Overcast, Rain

Wind: Strong, Weak

PlayTennis: yes, no

Operators for GA

	Initial strings	Crossover Mask	offspring
Single-point crossover	11101001000	11111000000	11101010101
	00001010101		00001001000
Two-point crossover	11101001000	00111110000	11001011000
	00001010101		00101000101

Operators for GA

	Initial strings	Crossover Mask	offspring
Uniform crossover	11101001000	10011010011	10001000100
	00001010101		01101011001
Point mutation	11101001000	→	11101011000

Select most fit hypotheses

- Fitness proportionate selection
  - –$Pr(h_i)=Fitness(h_i)/Sum(Fitness(h_k))$
  - –Can lead to crowding
- Alternatives
  - –Tournament selection
    - Pick h1 and h2 randomly
    - With probability p, select the more fit one from h1 and h2
  - –Rank selection
    - Sort all hypotheses by their fitness
    - Prob. of selection is propositional to its rank
- Complexity and generality

GABIL [Dejong et al. 1993]

- Learn disjunctive set of propositional rules
- Fitness:
  - –Fitness(h)=(correct(h))^2
- Representation:
  - –IF a1=T AND a2=F THEN c=T; IF a2=T THEN c=F

By:	a1	a2	c	a1	a2	c
	10	01	1	11	10	0

- Genetic operators:
  - –Variable length rule set
  - –Well-formed bit string hypotheses

GABIL [Dejong et al. 1993]

- Crossover
  - a1 a2 c a1 a2 c
  - h1: 1[0 01 1 11 1]0 0
  - h2: 0[1 1]1 0 10 01 0

•Choose crossover point for h1 as <1,8>

•Restrict the crossover points in h2: <1,3>, <1,8>,<6,8>.

•If <1,3>,Results:

- 1[1 1]0 0
- 0[0 01 1 11 1]1 0 10 01 0

GABIL Extensions

- Add new genetic operators, also applied probabilistically:
  - AddAlternative: generalize constriant on a_i by changing a 0 to 1
  - DropCondition: generalize constriant on a_i by changing every 0 to 1

•Add new fields to bit string:

- a1 a2 c a1 a2 c AA DC
- 01 11 0 10 01 0 1 0

GABIL Results

- Average performance on a set of 12 synthetic problems:
  - –GABIL without AA and DC operators: 92.1% accuracy
  - –GABIL with AA and DC operators: 95.2% accuracy
  - –Symbolic learning methods (C4.5, ID5R, AQ14) ranged from 91.2 to 96.6% accuracy

Schema

- How to characterize the evolution of population in GA?
  - –Schema: string containing 0,1 *
  - –0*1, representing 001, 011
- Characterize population by number of instances representing each possible schema
  - –m(s, t): number of instances of schema s in population at time t

Schema

- E[m(s, t+1)] >=

  u(s, t) * m(s, t) / f(t)

  *(1 - p_c * d(s)/(l -1))

  *(1 – p_m)^(o(s))

•f(t): average fitness of population at time t

•u(s,t): average fitness of schema s at time t

•p_c: prob. of single point crossover operator

•p_m: prob. of mutation operator

•l: length of single bit strings

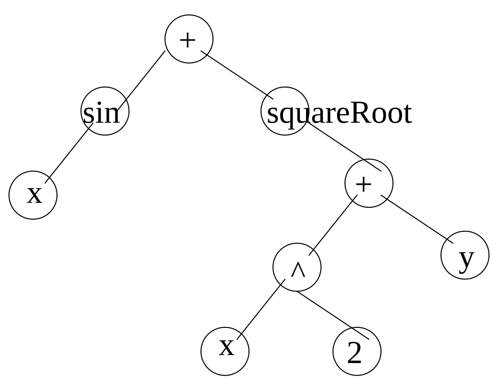
•o(s): #of defined bits in schema s

•d(s): distance between leftmost and rightmost defined bits in schema s

Genetic Programming

- population of programs represented by trees:

  sin(x) + squareRoot(square(x) + y)



Crossover

Biological Evolution

- Lamark (19th century)
  - Individual genetic makeup was altered by lifetime experience
  - Current evidence contradicts this view
  - But it improve efficiency in GP
- What is the impact of individual learning on population evolution?

Baldwin Effect

- Assume:
  - Individual learning has no direct effect on individual DNA
- Then:
  - Ability of individuals to learn will support more diverse gene pool
  - More diverse gene pool will support faster evolution of the gene pool
- So, individual learning indirectly increases the evolution rate

Baldwin Effect

•Plausible example:

–New predator appears in environment

–Individuals who can learn (to avoid it) will be selected

–Increase in learning individuals will support more diverse gene pool

–Resulting in faster evolution

–Possibly resulting in new non-learned (or genetic) traits such as instinctive fear of the predator

Experiments on Baldwin Effect [Hinton & Nowlan, 1987]

- Evolve simple neural networks:
  - Some networks weights fixed during lifetime, while others trainable
  - Genetic makeup determines which are fixed, and their weight values
- Results:
  - With no individual learning, population failed to improve overtime
  - With individual learning
    - Early generations: population contained many individuals with many trainable weights
    - Later generations: higher fitness, while number of trainable weights decreased

Usage

• huge search space

• avoid the problem of local minimal, so after several generations, the solution is very near to the optimal one.

Acknowledgement

•Based on Tom M. Mitchell's slides

•From "Machine Learning", Tom M. Mitchell