CSCE 478/878 Lecture 11: Genetic Algorithms

Stephen D. Scott (Adapted from Tom Mitchell's slides)

December 3, 2001

Outline

- Evolutionary computation
- Prototypical GA
- An example: GABIL
- Genetic Programming
- Individual learning and population evolution

Evolutionary Computation

- 1. Computational procedures patterned after biological evolution
- 2. Search procedure that probabilistically applies search operators to set of points in the search space
- Remember, learning is a <u>search</u> for a good hypothesis, so EC can certainly be applied

Biological Evolution

Lamarck and others:

• Species "transmute" over time

Darwin and Wallace:

- Consistent, heritable variation among individuals in population
- Natural selection of the fittest

Mendel and genetics:

- A mechanism for inheriting traits
- genotype \rightarrow phenotype mapping

A Basic GA

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

- Initialize: $P \leftarrow p$ random hypotheses
- *Evaluate:* for each h in P, compute Fitness(h)
- While $[\max_h Fitness(h)] < Fitness_threshold$
 - 1. Select: Probabilistically select (1 r)p members of *P* to add to *P*_S.

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

- 2. *Crossover:* Probabilistically select $\frac{r \cdot p}{2}$ pairs of hypotheses from *P*. For each pair, $\langle h_1, h_2 \rangle$, produce two offspring by applying the Crossover operator. Add all offspring to P_s .
- 3. *Mutate:* Invert a randomly selected bit in $m \cdot p$ random members of P_s
- 4. Update: $P \leftarrow P_s$
- 5. *Evaluate:* for each h in P, compute Fitness(h)
- Return the hypothesis from P that has the highest fitness

Representing Hypotheses

Represent

 $(Outlook = Overcast \lor Rain) \land (Wind = Strong)$ by

Represent

 $\label{eq:interm} \mbox{IF } Wind = Strong \ \ \mbox{THEN } PlayTennis = yes$ by

Outlook Wind PlayTennis 111 10 10



Selecting Most Fit Hypotheses

Fitness proportionate selection:

$$\Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=1}^{p} Fitness(h_j)}$$

... can lead to crowding

Tournament selection:

- Pick h_1, h_2 at random with uniform prob.
- With probability p, select the more fit

Rank selection:

- Sort all hypotheses by fitness
- Prob of selection is proportional to rank

GABIL [DeJong et al. 1993]

Learn disjunctive set of propositional rules, competitive with C4.5

Fitness:

$$Fitness(h) = (fraction correct(h))^2$$

Representation:

IF $a_1 = T \land a_2 = F$ THEN c = T; IF $a_2 = T$ THEN c = F

represented by

a_1	a_2	C	a_1	a_2	С
10	01	1	11	10	0

Genetic operators: ???

- want variable length rule sets
- want only well-formed bitstring hypotheses

Crossover with Variable-Length Bitstrings

Start with

h_1 :	$\begin{array}{c c} a_1 \\ 1 & 0 \end{array}$	a ₂ 01	с 1	a ₁ 11	a₂ 1∣0	с 0
h ₂ :	01	11	0	10	01	0

- 1. Choose crossover points for h_1 , e.g., after bits 1, 8
- 2. Now restrict points in h_2 to those that produce bitstrings with well-defined semantics, e.g. $\langle 1, 3 \rangle$, $\langle 1, 8 \rangle$, $\langle 6, 8 \rangle$.

if we choose $\langle 1, 3 \rangle$, result is

GABIL Extensions

Add new genetic operators, also applied probabilistically:

- 1. AddAlternative: generalize constraint on a_i by changing a 0 to 1
- 2. *DropCondition*: generalize constraint on a_i by changing every 0 to 1

And, add new field to bitstring to determine whether to allow these

So now the learning strategy also evolves!

GABIL Results

Performance of GABIL comparable to symbolic rule/tree learning methods C4.5, ID5R, AQ14

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 ranged from 91.2 to 96.6

Schemas

How to characterize evolution of population in GA?

Schema = string containing 0, 1, * ("don't care")

- Typical schema: 10**0*
- Instances of above schema: 101101, 100000, ...

Characterize population by number of instances representing each possible schema

 m(s,t) = number of instances of schema s in pop at time t

Consider Just Selection

- $\overline{f}(t)$ = average fitness of pop. at time t
- m(s,t) = # of instances of schema s in pop at time t
- $\hat{u}(s,t) = avg$ fitness of instances of s at time t

Probability of selecting h in one selection step

$$Pr(h) = \frac{f(h)}{\sum_{i=1}^{n} f(h_i)}$$
$$= \frac{f(h)}{n\overline{f}(t)}$$

Probability of selecting an instance of s in one step

$$Pr(h \in s) = \sum_{h \in s \cap p_t} \frac{f(h)}{n\overline{f}(t)}$$
$$= \frac{\widehat{u}(s,t)}{n\overline{f}(t)} m(s,t)$$

Expected number of instances of s after n selections

$$E[m(s,t+1)] = \frac{\widehat{u}(s,t)}{\overline{f}(t)}m(s,t)$$

Schema Theorem

$$E[m(s,t+1)] \ge \frac{\widehat{u}(s,t)}{\overline{f}(t)}m(s,t)\left(1-p_c\frac{d(s)}{\ell-1}\right)(1-p_m)^{o(s)}$$

- m(s,t) = # of instances of schema s in pop at time t
- $\overline{f}(t)$ = average fitness of pop. at time t
- $\hat{u}(s,t) = avg$ fitness of instances of s at time t
- $p_c =$ probability of single point crossover operator
- $p_m =$ probability of mutation operator
- ℓ = length of bit strings
- o(s) number of defined (non "*") bits in s
- d(s) = distance between leftmost, rightmost defined bits in s

Genetic Programming

Population of programs represented by trees







Goal: spell UNIVERSAL in one stack

Terminals:

- CS ("current stack") = name of the top block on stack, or F.
- TB ("top correct block") = name of topmost correct block on stack
- NN ("next necessary") = name of the next block needed above TB in the stack

Block Problem (cont'd)

Primitive functions:

- (MS x): ("move to stack"), if block x is on the table, moves x to the top of the stack and returns the value T. Otherwise, does nothing and returns the value F
- (MT x): ("move to table"), if block x is somewhere in the stack, moves the block at the top of the stack to the table and returns the value T. Otherwise, returns F
- (EQ *x y*): ("equal"), returns *T* if *x* equals *y*, and returns *F* otherwise
- (NOT x): returns T if x = F, else returns F
- (DU x y): ("do until") executes the expression x repeatedly until expression y returns the value T

Learned Program

Trained to fit 166 test problems

Using population of 300 programs, found this after 10 generations:

(EQ (DU (MT CS)(NOT CS)) (DU (MS NN)(NOT NN)))

Genetic Programming

More interesting example: design electronic filter circuits

- Individuals are programs that transform begining circuit to final circuit, by adding/subtracting components and connections
- Use population of 640,000, run on 64-node parallel processor
- Discovers circuits competitive with best human designs

GP for Classifying Images

[Teller and Veloso, 1997]

Fitness: based on coverage and accuracy

Representation:

- Primitives include Add, Sub, Mult, Div, Not, Max, Min, Read, Write, If-Then-Else, Either, Pixel, Least, Most, Avg, Variance, Difference, Mini, Library
- Mini refers to a local subroutine that is separately coevolved
- Library refers to a global library subroutine (evolved by selecting the most useful minis)

Genetic operators:

- Crossover, mutation
- Create "mating pools" and use rank proportionate reproduction

Biological Evolution Revisited

Lamark (19th century)

- Believed individual genetic makeup was altered by lifetime experience
- Though current evidence contradicts this view, has been employed in some artificial systems

Baldwin Effect

Assume

- Individual learning has no direct influence on individual DNA
- But ability to learn reduces need to "hard wire" traits in DNA

Then

- Ability of individuals to learn will support more diverse gene pool
 - Because learning allows individuals with various "hard wired" traits to be successful
- More diverse gene pool will support faster evolution of gene pool

 \rightarrow individual learning (indirectly) increases rate of evolution

Baldwin Effect

Plausible example:

- 1. New predator appears in environment
- 2. Individuals who can learn to avoid it will be selected
- 3. Increase in learning individuals will support more diverse gene pool ...
- 4. ... resulting in faster evolution ...
- 5. ... possibly resulting in new non-learned traits such as instintive fear of predator

Computer Experiments on Baldwin Effect [Hinton and Nowlan, 1987]

Evolve simple neural networks:

- Some network weights fixed during lifetime, others trainable
- Genetic makeup determines which are fixed, and their weight values

Results:

- With no individual learning, population failed to improve over time
- When individual learning allowed
 - Early generations: population contained many individuals with many trainable weights
 - Later generations: higher fitness, while number of trainable weights decreased

Summary: Evolutionary Algorithms

- Conduct randomized, parallel, hill-climbing search through H
 - Use of population allows fast, robust search of H
 - H can be space of hypotheses (e.g. sets of rules) or programs
- Approach learning as optimization problem (optimize fitness)