Chapter 9: Genetic Algorithms

CS 536: Machine Learning

Littman (Wu, TA)

Genetic Algorithms

[Read Chapter 9]

[Exercises 9.1, 9.2, 9.3, 9.4]

- 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

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 → phenotype mapping

Watson and Crick:

- Information strings! (DNA)

GA Template

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
  2. *Crossover*: Probabilistically select $rp/2$ 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 *mp* random members of P_s
  4. *Update*: $P \leftarrow P_s$
  5. *Evaluate*: for each *h* in *P*, compute *Fitness*(*h*)

return $\text{argmax}_{h \in P}$ *Fitness*(*h*)

Representing Hypotheses

Represent

(*Outlook* = *Overcast* v *Rain*) ^ (*Wind* = *Strong*)

by

Outlook	*Wind*
011	10

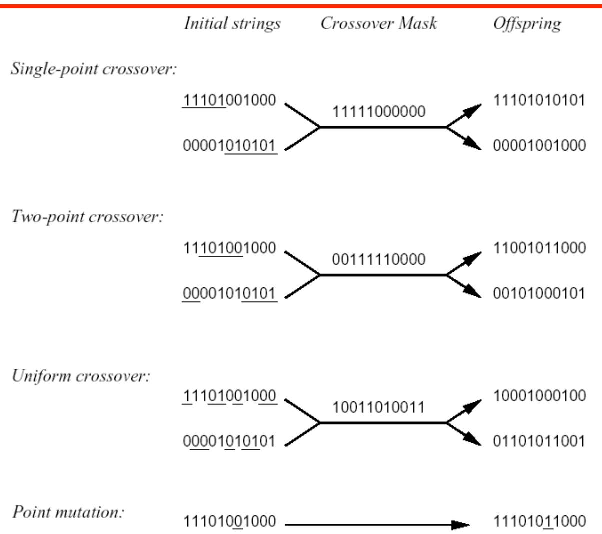
Represent

IF *Wind* = *Strong* THEN *PlayTennis* = *yes*

by

Outlook	*Wind*	*PlayTennis*
111	10	10

Operators for GAs



Selecting Most Fit Hypotheses

Fitness proportionate selection:

$$\Pr(h_i) = Fitness(h_i) / \Sigma_{j=1}^{p} Fitness(h_j)$$

- can lead to crowding, sensitive to *Fitness* magnitudes.

Tournament selection:

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

Rank selection:

- Sort all hypotheses by fitness
- Prob. of selection depends on rank

Learning As Optimization

- One way to view the search for a hypothesis is as an optimization problem.
- We can measure the error for any hypothesis.
- Posit a series of hypotheses to find one with low error.
- Relate low error with high fitness to get a mapping to GAs.

GABIL [DeJong et al. 1993]

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

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

Representation:

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

represented by a_1 a_2 c a_1 a_2 c

10 01 1 11 10 0

Genetic operators: ???

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

Crossover Variable-Length Bitstrings

Start with a_1 a_2 c a_1 a_2 c

h_1 : 10 01 1 11 10 0

h_2 : 01 11 0 10 01 0

1. pick h_1 crossover points: after bits 1, 8
2. restrict points in h_2 to have well-defined semantics: <1, 3>, <1, 8>, <6, 8>.

if <1, 3>: a_1 a_2 c a_1 a_2 c a_1 a_2 c

h_3 : 11 10 0

h_4 : 00 01 1 11 11 0 10 01 0

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

Add new field to bitstring gating these:

a_1 a_2 c a_1 a_2 c AA DC

01 11 0 10 01 0 1 0

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 (accuracy) on a set of 12 synthetic problems:

- GABIL without AA and DC: 92.1%
- GABIL with AA and DC: 95.2%
- symbolic learning: 91.2% to 96.6%

(What's missing to evaluate these results?)

Schemata

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 pop. by number of instances representing each possible schema

- $m(s, t)$ = number of instances of schema s in population at time t

Consider Just Selection (1)

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

Probability of selecting h in one selection step (via fitness proportional selection):

$$\Pr(h) = f(h) / \Sigma_{i=1}^{n} f(h_i)$$
$$= f(h) / (n \bar{f}(t))$$

Consider Just Selection (2)

Probability of selecting an instance of s in one step

$$\Pr(h \text{ in } s) = \Sigma_{h \text{ in } s \cap pt} = f(h) / (n \bar{f}(t))$$
$$= \hat{u}(s, t) / (n \bar{f}(t)) \; m(s, t)$$

Expected number of instances of s after n selections

$$E[m(s, t+1)] = \hat{u}(s, t) / \bar{f}(t) \; m(s, t)$$

Schema Theorem

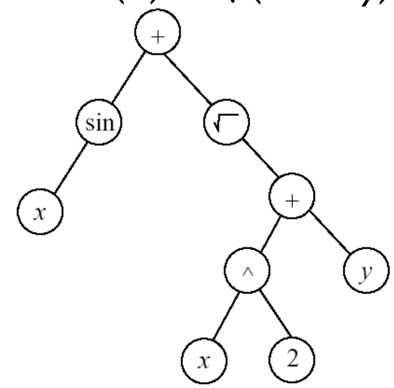
$E[m(s, t+1)]$

$$\geq \hat{u}(s,t) \,/\, f(t)\; m(s,t)\,(1-p_c\, d(s)/(l-1))(1-p_m)^{o(s)}$$

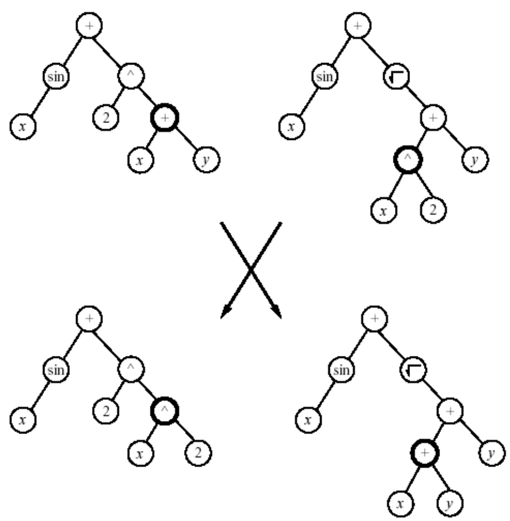
- $m(s, t)$ = instances of schema s in pop. at time t
- $\bar{f}(t)$ = average fitness of pop. at time t
- $\hat{u}(s, t)$ = ave. fitness of instances of s at time t
- p_c = probability of single point crossover operator
- p_m = probability of mutation operator
- l = length of single bit strings
- $o(s)$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

Genetic Programming

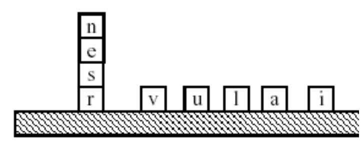
Population of programs represented by trees: $\sin(x) + \sqrt{(x^2 + y)}$



Crossover



Block Problem



Goal: spell UNIVERSAL

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

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 beginning 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, Ave, Variance, Difference, Mini, Library
- Mini refers to a local subroutine that is separately co-evolved
- 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

Lamark (19th century)

- Believed individual genetic makeup was altered by lifetime experience
- But current evidence contradicts this view

What is the impact of individual learning on population evolution?

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

→ 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 instinctive fear of predator

Computer Experiments

[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: EP

- Conduct randomized, parallel, hill-climbing search through *H*
- Approach learning as optimization problem (optimize fitness)
- Nice feature: evaluation of Fitness can be very indirect
  - consider learning rule set for multistep decision making
  - no issue of assigning credit/blame to indiv. steps