Genetic Algorithms

- 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 a set of points in the search space

- Also popular with optimization folks

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

Genetic Algorithm

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

- *Initialize* : $P \leftarrow p$ random hypotheses
- *Evaluate* : for each h in P, compute *Fitness(h)*
- While $[\max_h Fitness(h)] < FitnessThreshold$

  1. *Select* : probabilistically select $(1\text{-}r)p$ 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 pairs of hypotheses from P. For each pair $< h_1, h_2 >$, 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 Ps
  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

(Type=Car $\vee$ *Minivan)* $\wedge$ *(Tires = Blackwall)*

by

Type	Tires
011	10

Represent

IF (Type = SUV) THEN (NiceCar = yes)

by

Type	Tires	NiceCar
100	11	10

Operators for Genetic Algorithms

Parent Strings		Offspring
101100101001 000011100101	*Single Point Crossover*	101111100101 000000101001
101100101001 000011100101	*Two Point Crossover*	101011101001 000100100101
101100101001 000011100101	*Uniform Crossover*	100111100001 001000101101
101100101001	*Point Mutation*	101100100001

Selecting Most Fit Hypothesis

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 probability
- With probability p, select the more fit

Rank selection:

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

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∧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 bitstring hypotheses

Crossover with 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. Choose crossover points for h1, e.g., after bits 1,8
   h_1 : 1[0 01 1 11 1]0 0
2. Now restrict points in h2 to those that produce bitstrings with well-defined semantics, e.g.,
   <1,3>, <1,8>, <6,8>
   If we choose <1,3>:
   h_2 : 0[1 1]1 0 10 01 0
   Result is:

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, 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 bit string to determine whether to allow these:

a_1 a_2 c a_1 a_2 c *AA* *DC*
10 01 1 11 10 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 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% accuracy

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 population at time *t*

Consider Just Selection

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

Probability of selecting h in one selection step

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

Probability of selecting an instances of s in one step

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

Expected number of instances of s after n selections

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

Schema Theorem

$$E[m(s,t+1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t)\left(1 - p_c \frac{d(s)}{l-1}\right)(1-p_m)^{o(s)}$$

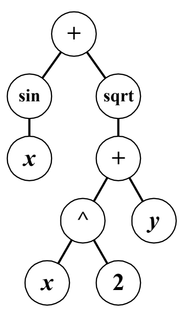
- $m(s,t)$ = instances of schema s in population at time t
- $\bar{f}(t)$ = average fitness of population at time t
- $\hat{u}(s,t)$ = average fitness of instances of s at time t
- p_c = probability of single point crossover operator
- p_m = probability of mutation operator
- l = length of single bit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

Genetic Programming

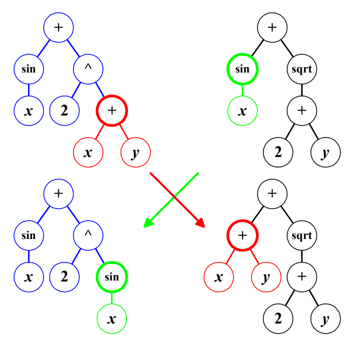
Population of programs represented by trees

Example:

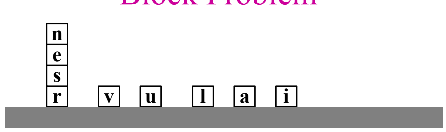
$$\sin(x) + \sqrt{x^2 + y}$$



Crossover



Block Problem



Goal: spell UNIVERSAL

Terminals:

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

Block Problem Primitives

Primitive functions:

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

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 the beginning circuit to a final circuit by adding/subtracting components and connections
- Use population of 640,000, run on 64 node parallel process
- Discovers circuits competitive with best human designs

GP for Classifying Images

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

Lamarck (19th century)

- Believed individual genetic makeup was altered by lifetime experience
- 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 the 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 (Example)

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 on Baldwin Effect

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, white number of trainable weights decreased

Bucket Brigade

- Evaluation of fitness can be very indirect
  - consider learning rule set for multi-step decision making
  - bucket brigade algorithm:
    - rule leading to goal receives reward
    - that rule turns around and contributes some of its reward to its predessor
  - no issue of assigning credit/blame to individual steps