Genetic Algorithms

Mitchell chapter 9.

- Based loosely on simulated evolution.
- Hypotheses: described in bit strings (subject to interpretation in specific domains).
- Search: population of hypotheses, refined through mutation and crossover to increase fitness.
- Applications: optimization problems, learning the topology and parameters in neural networks, and many more.

Biological Evolution

Lamarck and others:

- Species “transmute” over time (inheritance of acquired trait)

Darwin and Wallace:

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

Mendel and genetics:

- A mechanism for inheriting traits
- genotype $\longrightarrow$ phenotype mapping

Motivation

- Mutation and crossover of hypotheses in the current population.
- Basically a *generate-and-test* beam search.
- Motivating factors:
  - Evolution is known to be successful.
  - GAs can search hypotheses containing complex interacting parts.
  - Easily parallelizable.

Genetic Algorithms

- Population: set of current hypotheses
- Fitness: predefined measure of success
- Elements of GA:
  fitness test $\longrightarrow$ selection $\longrightarrow$ reproduction (mutation, crossover)

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

$$(Outlook = Overcast \vee Rain) \wedge (Wind = Strong)$$

by

$Outlook$	$Wind$
011	10

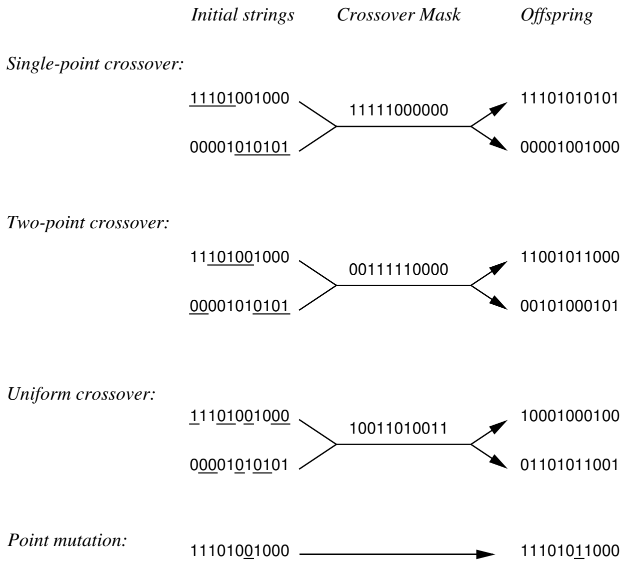
Represent

IF $Wind = Strong$ THEN $PlayTennis = yes$

by

$Outlook$	$Wind$	$PlayTennis$
111	10	10

Genetic Operators



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
- Probability of selection is proportional to rank

Example: GABIL [DeJong et al. 1993]

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

Fitness:

$$Fitness(h) = (percent\_correct(h))^2$$

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 (as number of attributes can change)
- 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 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

	a_1	a_2	c
h_3 :	11	10	0

	a_1	a_2	c	a_1	a_2	c	a_1	a_2	c
h_4 :	00	01	1	11	11	0	10	01	0

Lesson

- Picking a representation for the hypotheses can be tricky.
- Genetic operators need to preserve the semantics of the genetic encoding.

Extensions to GABIL

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

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. (Allowing this increased accuracy.)

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

Characterizing Evolution: 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, ...
- An instance of lenght 4, say 0010, can have 2^4 matching schemas.

Characterize population by number of instances representing each possible schema:

- $m(s, t) =$ number of instances of schema s in pop at time t
- Want to estimate $m(s, t + 1)$ given $m(s, t)$ and other factors.

Factors Influencing Change in $m(s, t)$

$m(s, t)$ can change as t changes, due to the following factors:

- Selection: if individuals representing s get selected more often, $m(s, \cdot)$ will increase.
- Crossover
- Mutation

Schema theorem: gives $E[m(s, t + 1)]$.

Influence of Selection

- $\bar{f}(t) =$ average fitness of pop. at time t
- $m(s, t) =$ instances of schema s in pop at time t
- $\hat{u}(s, t) =$ average fitness of instances of s at time t
- $h \in s \cap p_t$: instances of schema s in the population at time t

Probability of selecting h in one selection step

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

Mean fitness of instances of s at time t:

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

Influence of Selection

Probabilty of selecting an instance of s in one step

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

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

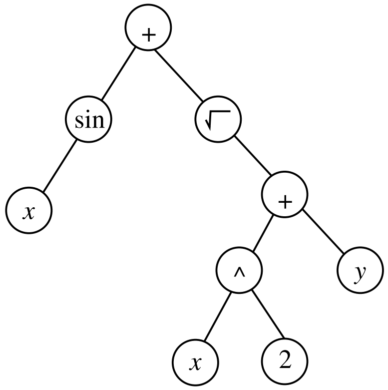
$$E[m(s,t{+}1)] \geq \frac{\hat{u}(s,t)}{\bar{f}(t)} m(s,t) \left(1 - p_c \frac{d(s)}{l-1}\right) (1{-}p_m)^{o(s)}$$

- $m(s,t) =$ instances of schema s in 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

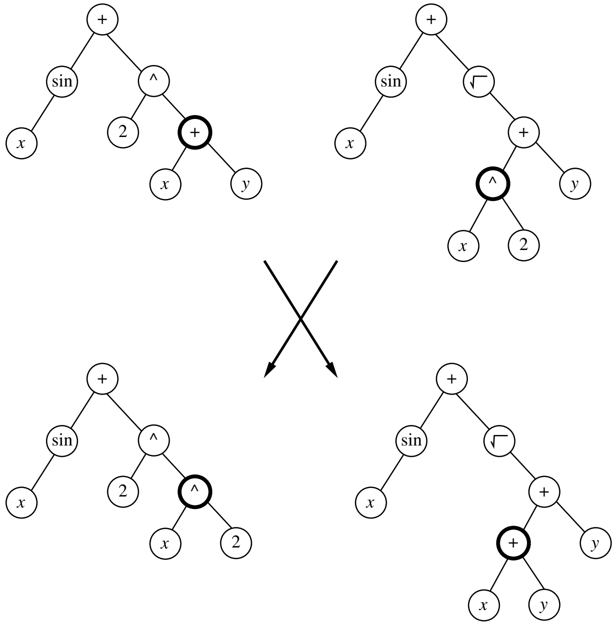
Genetic Programming

Population of programs represented by trees

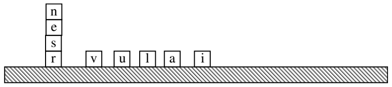
$$\sin(x) + \sqrt{x^2 + y}$$



Crossover: Swap whole subtrees



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)) )
```

Biological Evolution

Lamarck (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 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

Other Considerations

- Coevolution: escalating effect or complementary dependence (insects and flowering plants) between two or more species.
- Cultural transmission: **memes** vs. genes.

Summary: Evolutionary Learning

- 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 individual steps