Lecture 22: Evolutionary computation

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

Evolutionary computation

- Computational procedures patterned after biological evolution
- Search procedure that probabilistically applies search operators to set of points in the search space

Biological Evolution:

- Lamark 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

Typical genetic algorithm

$GA(Fitness, Fitness\_threshold, p, r, m)$

1. *Initialize:* $P \leftarrow p$ random hypotheses
2. *Evaluate:* for each $h \in P$, compute $Fitness(h)$
3. While $\max_h Fitness(h) < Fitness\_threshold$
   (a) *Select:* Probabilistically select $(1-r)p$ members of P to add to P_S.
   (b) *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.
   (c) *Mutate:* Invert a randomly selected bit in $m \cdot p$ random members of P_s
   (d) *Update:* $P \leftarrow P_s$
   (e) *Evaluate:* for each $h \in P$, compute $Fitness(h)$
4. Return the hypothesis from P that has the highest fitness.

Example: Binary classification

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Discover a “rule” for the PlayTennis predicate!

Representing hypotheses

Represent

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

by

Outlook	*Wind*
011	10

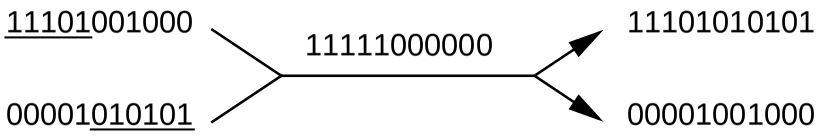
Represent IF $Wind = Strong$ THEN $PlayTennis = yes$ by

Outlook	*Wind*	*PlayTennis*
111	10	10

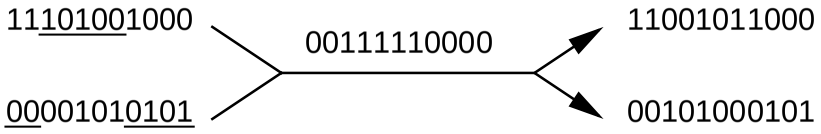
Operators for genetic algorithms

Initial strings *Crossover Mask* *Offspring*

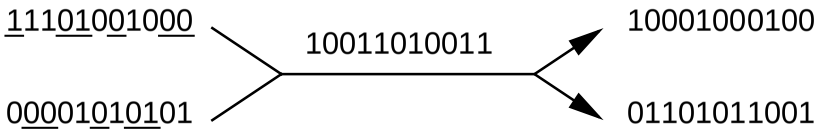
Single-point crossover:



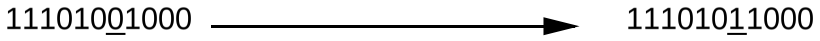
Two-point crossover:



Uniform crossover:



Point mutation:



Kinds of crossover

- Single-point crossover: pick a single point in each parent, cut the strings there and recombine
- Uniform crossover: pick a number of points at random in each parent. In the offspring, alternate the fragments of the two parents
- If the chromosomes have fixed length, the cut points have to be identical in both parents

Selecting the most fit hypotheses

- Fitness proportionate selection:

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

  ... can lead to *crowding* (multiple copies being propagated)

- Tournament selection:
  1. Pick h_1, h_2 at random with uniform probability
  2. With probability p, select the more fit.
- Rank selection:
  1. Sort all hypotheses by fitness
  2. Probability of selection is proportional to rank
- Softmax selection:

$$P(h_i) = \frac{e^{Fitness(h_i)/T}}{\sum_{j=1}^{p} e^{Fitness(h_j)/T}}$$

Example: GABIL (DeJong et. al, 1993)

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

- Fitness: $Fitness(h) = (Correctness(h))^2$
- 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

- *What genetic operators should we use?*
  - variable length rule sets
  - only well-formed bit-string hypotheses as a result

Crossover with Variable-Length Bit-strings

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. Restrict points in h_2 to those that produce bit-strings with well-defined semantics, e.g., $\langle 1,3 \rangle$, $\langle 1,8 \rangle$, $\langle 6,8 \rangle$.

If we choose $\langle 1,3 \rangle$, the 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

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 operators:

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

The performance of GABIL was comparable to symbolic rule/tree learning methods such as C4.5

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
- The symbolic learning methods ranged from 91.2 to 96.6

Schemas

How can we characterize the evolution of population in GA?

Schema = string containing 0, 1, * (“don’t care”)

Example:

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

The population can then be characterized by the number of instances representing each possible schema:

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

Consider just selection

- The probability of selecting h in one selection step is:

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

  where $\overline{f}(t)$ is the average fitness of the population at time t
- The probability of selecting an instance of s after one time step is:

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

  where $\hat{u}(s,t)$ is the average fitness of the instances of s at time t
- The expected number of instances of s after n selections is:

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

Schema theorem

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

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

Genetic programming

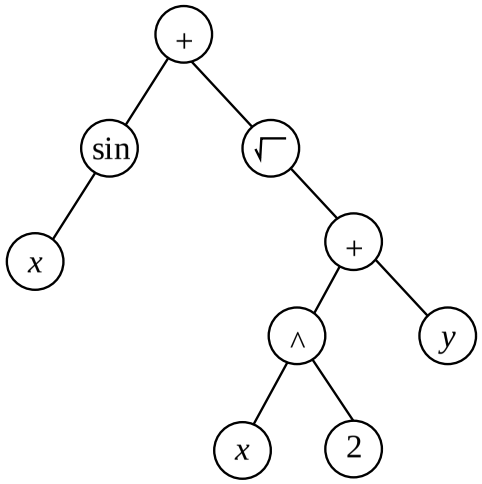
Goal: Evolve entire programs, in order to improve their performance.

The population of programs is typically represented by trees.

For instance a function computing:

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

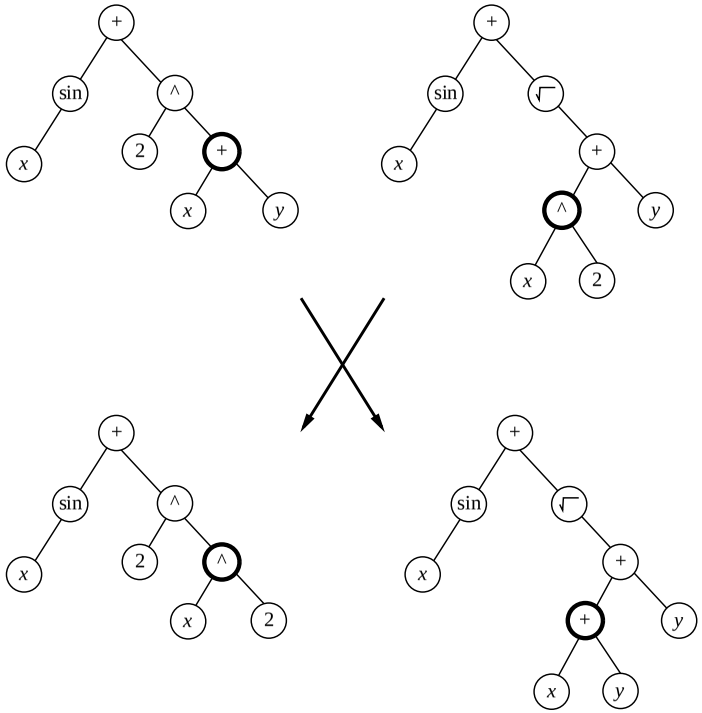
would be represented by:



Crossover

A lot of crossover operations would result in programs with syntactic mistakes, or programs that would not run.

Often the crossover needs to be restricted and the offspring need to be checked. E.g.:



Example: Designing 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

Biological vs. individual 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

So individual learning (indirectly) increases rate of evolution!

Example of Baldwin effect

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, resulting in faster evolution
4. Possibly resulting in new non-learned traits such as instinctive fear of predator

mputer experiments on Baldwin effect (Hinton & Nowlan, 1987)

Evolve simple neural networks:

- Some network weights fixed during lifetime, others trainable
- Genetic makeup determines which networks 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 hypothesis space
- Approach learning as optimization problem (optimize fitness)
- But unlike in reinforcement learning, there is no attempt to assign credit/blame directly to particular traits.
- Fitness evaluation can be very indirect
- Can be combined with other learning methods (e.g. decision trees, neural networks), especially in order to determine the structure of function approximators
- Can be very slow (just like natural evolution)