Slide 14

Genetic Programming

Reference: Koza, 1992, 1994;

Uses genetic algorithms to generate computer programs represented as parse trees.

Representation: Parse trees of LISP expressions. Requires:

Terminals.

Functions which take other functions or terminals as input.

Example: Representing the absolute value function.

Terminal set: real variables x; real constants; boolean constants.

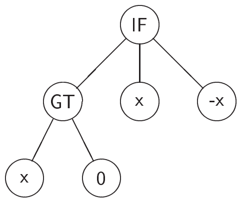
Function set: IF$(\cdot, \cdot, cdot)$ evaluates the second (third) argument if the first evaluates to TRUE (false). GT$(\cdot, \cdot)$ returns true if the first argument is greater than the second.

Slide 15

Representation of program as a functional expression:

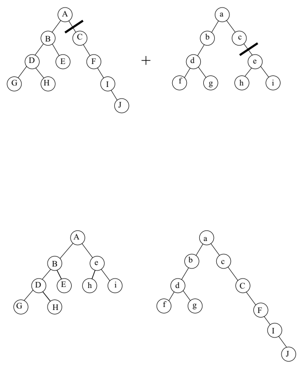
```
(IF (GT (x) (0))
    (x)
    (-x))
```

An expression represented as a tree:



Slide 16

Crossover: One randomly chosen subtree from one parent is replaced by that of another.



Slide 17

Mutation: Take a random subtree and replace it with a randomly generated one (often not used). Generates variable sized structures.

Selection: Fitness proportional selection with 10% elitism.

Slide 18

Example: Simulated Ant

Scenario: A simulated ant can detect food directly in front, can turn left or right, and can move forward. The world is a grid with a trail of food. There are some gaps in the trail.

Task: Collect as much food as possible from the trail. Since there are gaps, the ant must have exploration behaviour for when no food is detected, to find the trail.

Terminals: MOVE, RIGHT, LEFT.

Functions: IF-FOOD-AHEAD$(\cdot, \cdot)$ evaluates the first argument if food is ahead; the second if it is not. PROG2$(\cdot, \cdot)$ evaluates its first argument and then the second. PROG3$(\cdot, \cdot, \cdot)$ evaluates its three arguments in turn.

Examples: This moves towards food if detected, else turns left.

```
(IF-FOOD-AHEAD (MOVE) (LEFT))
```

Slide 19

Fitness: The number of pieces of food found by the ant in a given number of time-steps. Program is iterated until the number of time-steps is exceeded.

Perfect solution: (for the particular trail) found after 21 generations with a population size of 500.

```
(IF-FOOD-AHEAD (MOVE)
              (PROGN3 (LEFT)
                      (PROGN2 (IF-FOOD-AHEAD (MOVE)
                                             (RIGHT))
                              (PROGN2 (RIGHT)
                                      (PROGN2 (LEFT)
                                              (RIGHT))))
                      (PROGN2 (IF-FOOD-AHEAD (MOVE)
                                             (LEFT))
                               (MOVE))))
```

Slide 20

Evolving a cellular automata to perform a task.

References: Mitchell, Crutchfield, Das, 1996

Cellular automata computer:

- One dimensional line of cells, each taking 0 or 1 as its state.
- Each cell takes the state of r of its neighbours on each side and its own state and produces a new state according to a rule.
- All cells use the same rule, and are updated in parallel.
- System input: initial condition of the CA.
- System output: final condition of the CA.
- Programming task: find the rule which implements a given input-output relationship.

Slide 21

Example: Density Classification (Gacs, Krudyamov, Levins, 1978).

- Output: all 1's if more than half the inputs were 1's
- Output: all 0's otherwise.

Slide 22

Genetic algorithm used to search the 2^7 $(r = 3)$ bit rule table.

Selection: Take top 20% (elitism); modify bottom 80%.

Fitness: Number correct on 100 initial conditions with uniformly distributed densities.

Results:
- "Epochs of innovation" — dynamics exhibits punctuated equilibria
- "Information carrying particles". Interpreted the solutions as developing mechanisms to transmit information from one side of the system to the other. This was through domains of particular spatio-temporal patterns, the interfaces of which were interpreted as information carrying particles.

Slide 23

Genetic Algorithms and Neural Networks

References: Schaffer et. al. 1992; Yao, 1995

Genetic algorithms have been used for three general tasks in neural network learning:

1. Use a genetic algorithm to find the weights of a fixed architecture neural network.
2. Use a genetic algorithm to find the structure or architecture of the neural network (using perhaps another algorithm to learn the weights).
3. Use a genetic algorithm to evolve the learning rule for a neural network (Chalmers, 1990).

Slide 24

Genetic Algorithms to Train Fixed-Architecture Neural Networks

For supervised learning, this appears to be a questionable enterprise.

1. If gradient information is available, it is usual best to use it.
2. Genetic algorithms turn all problems into reinforcement learning problems (performance is defined by one number). Reinforcement learning algorithms are generally slower than supervised learning ones.

The argument in favour of GAs is that they are global search algorithms.

A number of researchers have studied this application, but there is little evidence that it is better than the standard algorithms (except for discrete-valued weights).

Slide 25

Genetic algorithms to search Neural Network Architectures

An evolutionary approach can be used to evolve the architecture of the network. Example EPNet (Yao, 1995).

Training the weights: done with a adaptive gradient descent/simulated annealing hybrid. Run for a fixed number of steps each generation.

Mutations: Grow and prune the network.Try in order

- delete a node
- delete a node
- add a node
- add a connection

until something works. (Biased towards pruning.)

Slide 26

Training sets: A training set used to train the weights; a validation set is used to evaluated fitness for architecture construction; a second validation set is used to select a member of the population to use as the final classifier.

Slide 27

Adding crossover

To include crossover, the architecture must be represented as strings. Two methods:

directed encoding: structure of the network is represented directly in the string. For example, the connectivity matrix could be represented as a string by rasterization (Miller et. al. 1989).

indirect encoding: The string encodes some process which generates the network.

- **grammatical encoding** the strings represent a grammar (Kitano, 1990) which generates the network.
- **developmental encoding:** here the strings represent a developmental process which generates the network architecture (Miglino et. al. 1996.)