CE213 Artificial Intelligence – Lecture 19

Genetic Algorithms

Nature-inspired Problem Solvers or Learning Algorithms:

Can be regarded as problem solvers, directly applied to problem solving

Or regarded as learning algorithms, applied to build problem solvers such as neural networks

Biological Basis

Darwin's Contribution to the Theory of Evolution

Contrary to popular opinion, Darwin did *not* invent the theory of evolution.

He proposed a ***mechanism*** of evolution :

Source of Variation + Selection $\rightarrow$ *Adaptation*

Adaptation will still occur even if the source of variation is completely **random**.

Biological Basis (2)

Mendelian Genetics

Organisms contain **coded descriptions** of all the features of the organism.

They are used to construct the organism during development and passed on to the organism's offspring.

Such coded descriptions are called ***genes***.

Genes are grouped into linear sequences called ***chromosomes.***

Biological Basis (3)

Sources of Variation

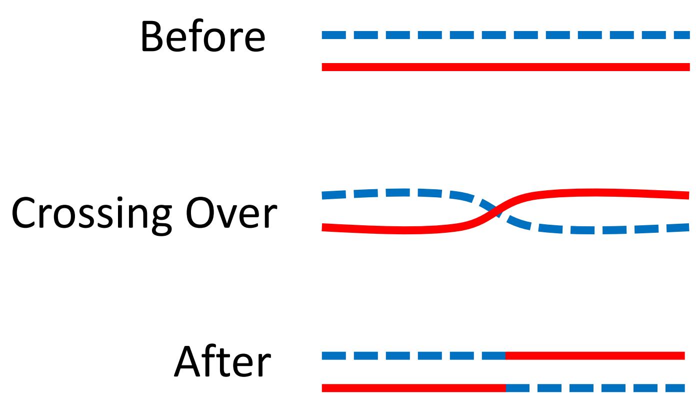
Mutation:

Modifies genes randomly – usually detrimental, occasionally beneficial.

Crossover:

Generates new combinations of genes.

Corresponding portions of **two chromosomes** are exchanged.



Biological Basis (4)

Natural Selection

If

An organism produces more offspring than can possibly survive

And

These offspring are only approximate copies, differing **in random ways**

Then

Those offspring that are best able to survive will be selected

Fitness as selection criterion

The likelihood that an individual will survive to reproduce is called its ***fitness***.

Therefore, it is the "Survival of the Fittest" selection criterion makes improvement through evolution, generation by generation.

A Basic Genetic Algorithm

Suppose

- We want to search for the solution to some problem;
- We can encode candidate solutions to the problem as a string of symbols (***problem formalisation***).

Then we could create a population of candidate solutions by randomly generating a set of such strings that represent ***chromosomes***.

Suppose we also have

- A function that could measure how good each candidate solution is,

  which could be used as a fitness function.

Then we could proceed to construct better solutions using the following procedure: cycles of **"Evaluation – Selection – Reproduction"**

A Basic Genetic Algorithm – Pseudo Code

Initialise a population of chromosomes; //as **candidate solutions**

REPEAT //**generation by generation, 'evaluation-selection-reproduction'**

    Determine fitness of each chromosome; //**evaluation**

    REPEAT

        //**selection**

        Select a pair of chromosomes (parents) with probability proportional to fitness;

        //**reproduction**

        Create two new chromosomes (children) using ***Crossover***;

        Apply ***Mutation*** to randomly change the new chromosomes.

    UNTIL enough children have been generated

    Replace the least fit members of the population with the children;

UNTIL required number of generations has been reached.

The chromosome with the highest fitness represents the optimal solution.

Mutation

The representation scheme (i.e., encoding) will define a set of values for representing genes in chromosomes.

e.g., {0,1} for a binary chromosome

Mutation simply replaces the current value in a gene with a randomly selected member of the value set.

Mutation Rate (to control how often a gene may be replaced.)

Mutation Rate = Probability that mutation will occur at a given gene

Typical value: 10^{-3}

Mutation rates are typically very low since they introduce random change that is usually harmful (occasionally beneficial).

Mutation (2)

Effect of Mutation

Major benefit: It can introduce new values into the gene pool. It plays a sort of role of exploration.

A system in which mutation is the **only source of variation** would adapt, but only very slowly.

Such a system would lack any means to combine two partial solutions to make a better solution.

Crossover

Crossover operators create new chromosomes by combining components of two existing chromosomes.

Uniform One-Point Crossover

There are many varieties of crossover.

The simplest is uniform one-point crossover:

```
Randomly select a single point somewhere along
the chromosome;
Exchange the portions of the two chromosomes
beyond the selected crossover point.
```

The crossover is ***uniform*** if each point in a chromosome is equally likely to be selected.

Crossover (2)

Crossover Rate (to control how often chromosomes may be selected for crossover)

Crossover rate = Probability that a given pair of chromosomes will crossover.

Typical values: 0.5 ~ 0.8

Such values allow

A significant number of chromosomes to be passed on to the next generation without modification

And a significant number of chromosomes to be recombined

Crossover (3)

Effect of Crossover

Major benefit: It can generate new gene combinations (new chromosomes) from the gene pool. It is more about exploitation.

A system in which crossover is the **only source of variation** would adapt, but would have no way of replacing any gene by something absent from the gene pool.

Hence, such a system might not find as good a solution as a system that also includes mutation.

Relationship between Mutation and Crossover

Mutation and crossover are *complementary*.

They do different jobs:

Mutation ensures the whole solution space can be searched.

Crossover accelerates the search process.

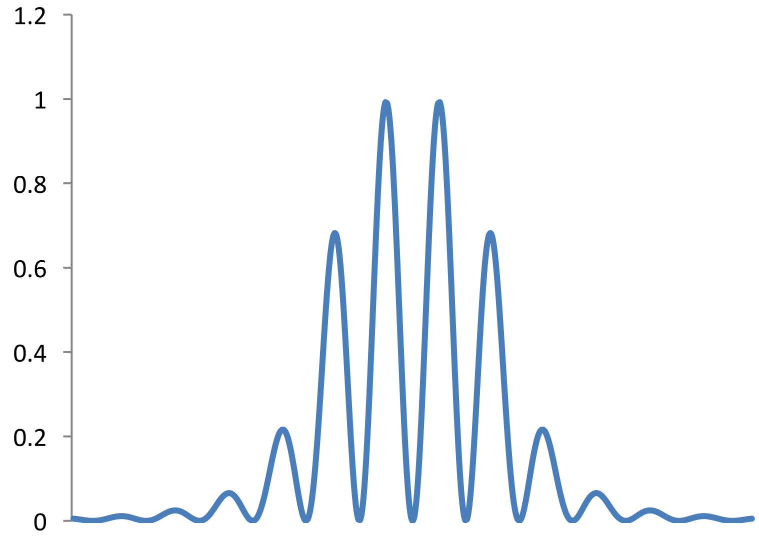
A Genetic Algorithm needs both, so that exploitation and exploration are combined.

An Example (finding the maximum point)

Consider the two argument function:

$$f(x,y) = \frac{\left(\sin\sqrt{x^2+y^2}\right)^2}{1+0.001(x^2+y^2)^2}$$

A plot of this function as x is varied looks something as shown in this figure:



What x and y make f(x,y) maximum?

This function is difficult to maximise because it has a very large number of local maxima and minima.

Problem Formalisation for a GA Solution

Encoding candidate solutions

Assume we know that around the maximum value of $f(x,y)$ the values of x and y are:

$$-100 < x < 100$$

$$-100 < y < 100$$

We could represent a candidate solution as a binary string of length 2L:

The first L bits represent x, and

the remaining L bits represent y.

Fitness function

Simple: Use the actual value of $f(x,y)$.

In general, **problem formalisation** includes: 1) solution representation,

2) fitness function.

Results

Using a population of 100 chromosomes (or candidate solutions represented as strings of 2L bits, L=16), run genetic algorithm for 40 generations, with evaluation-selection-reproduction in each generation.

Fitness of best five chromosomes:

After 1 generation

0.9903, 0.9893, 0.9100, 0.8697, 0.8241

After 4 generations

0.9823, 0.9823, 0.9774, 0.9758, 0. 9758

After 40 generations

0.9930, 0.9926, 0.9925, 0.9925, 0.9923

The corresponding chromosomes give the values of *x* and *y* at the maximum point.

How about Training a Neural Network?

Encoding candidate solutions

Assume there are N connection weights (including thresholds) in the neural network. Use L bits to represent one weight and thus NxL bits to represent all the weights. A chromosome of NxL bits can represent a candidate neural network.

Fitness function

Can be simply the accuracy of the neural network on the training dataset.

Use of genetic algorithm

Initialisation: Generate a population of *m* random chromosomes (candidate neural networks)

Cycles of 'evaluation-selection-reproduction' (*n* generations)

(The values of N, L, *m*, *n* depend on the size of the neural network and the experimental setup for the genetic algorithm)

Genetic Programming (Optional)

Genetic programming (GP) is a development from genetic algorithm (GA), in which chromosomes represent computer programs.

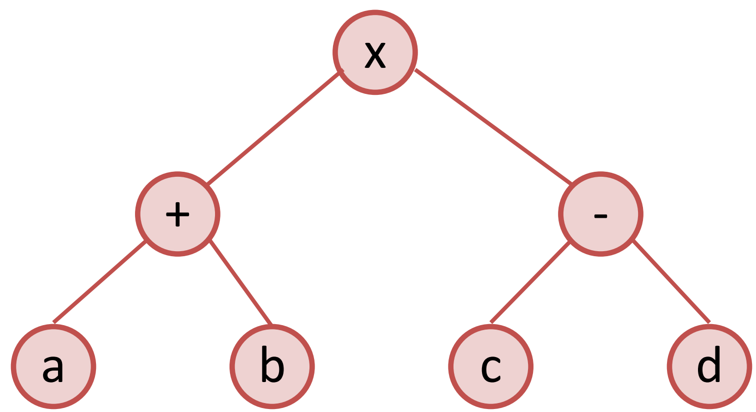
If GA is directly used, a program could be viewed as **a linear sequence of characters**, but this removes most of the structure of the program.

In GP, programs are typically represented as syntax trees.

Consider the following expression:

$$(a + b) \text{ x } (c - d)$$

This could be represented as a tree as shown in the figure on the right.



Therefore, in GP the population that evolves is made up of **trees rather than strings**. (**different problem representations**)

Operators for Genetic Programming

Crossover

The simple biologically inspired mechanism of crossover is fine for **strings** but must be modified to deal with **syntax trees**.

Given two parent trees, T_A and $T_{B.}$

Randomly select a **node** in each tree, N_A and N_B.

Swap the **subtrees** starting at N_A and N_B to make two new trees.

Many nodes of a syntax tree could be leaf nodes.

Swapping these produces only minor change.

Therefore, node selection in GP is not uniform.

It is biased to select non-leaf nodes.

Operators for Genetic Programming (2)

Mutation

Subtree mutation

Mutation point in the tree is replaced by a randomly generated subtree.

Point mutation

Changes the content of the chosen node only, such as compatible operands or arguments.

Summary

Biological Basis of Genetic Algorithm

Gene and Chromosome

Mutation and Crossover

Natural Selection and Fitness

A Basic Genetic Algorithm

Cycles of “Evaluation – Selection – Reproduction”

(a new approach to “generate and evaluate”)

Operators for Reproduction

Mutation

Crossover

Genetic Programming (optional)

Syntax Trees

Operators (mutation and crossover)