Genetic Algorithms

Peter Swire

**9.1** Design a genetic algorithm to learn the conjunctive classification rules for the *Play-Tennis* problem described in in Chapter 3. Describe precisely the bit-string encoding of hypotheses and a set of crossover operators.

Assuming that the day does not matter, we have four attributes and a target:

(outlook:3, temperature:3, humidity:2, wind:2, playTennis:2)

Since the greatest number of values that a single attribute can take is three, we’ll represent each part of the tuple with 3 bits. The third bit, in cases of only two values, is a no-op. An attribute with multiple on bits has each of those values joined by “or” statements. The separate attributes are assumed to be joined by “and” statements.

({sunny OR overcast} AND {hot} AND {high OR normal} AND {weak})

=

110 100 011 001

Fitness would be measured by the percent of instances in the training set that satisfy the rule.

I can’t think of a reason not to use regular single-point crossovers: pick a point at random, have everything to the left of it come from one parent, and everything to the right from another.

Mutation would be regular point mutation: flip a single bit.

**9.4** Consider applying GAs to the task of finding an appropriate set of weights for an artificial neural network (in particular, a feedforward network identical to those trained by backpropagation (Chapter 4)). Consider a 3 x 2 x 1 layered, feedforward network. Describe an encoding of network weights as a bit string, and describe an appropriate set of crossover operators. Do not allow all possible crossover operations on bit strings. State one advantage and one disadvantage of using GAs in contrast to backpropagation to train network weights.

Network weights are easily represented by matrices. Partitioning matrices keeps similar weights together, and so for a single-point crossover mutation I would use a row or column selected randomly. Backpropagation would use less space: you only have to keep one network, as opposed to a population of them. Backpropagation would also work better in smaller state-spaces: with a network with a small topology like this one, it might be faster to backpropagate throught it. Genetic algorithms will perform better on larger networks, or on networks that cannot be trained with backpropagation.

Dear professor: has anybody done work on learning *topologies* with genetic algorithms?

## Project

For my project, I tried to learn speech through blind additive synthesis and genetic algorithms. My training set was a single vector: a ten second clip of Neil Armstrong speaking from the moon. Encoding was a list of “Tone generators”: tuples of amplitude, frequency, phase, starting location and duration that described how to add a sine wave to a buffer. My fitness metric was the Euclidean distance between the buffer and the Armstrong clip. I did not use any Fourier analysis to guide my search.

I used single-point crossovers and multipoint mutations: at each mutation step, each tone generator had a 5% chance of mutating. If it was chosen to mutate, one of its attributes (amplitude, frequency, phase, starting location, duration) would change in either direction (randomly) by 2% of its previous value.

The single-point crossovers were done between the fittest member of a generation and everyone else.

Since each chromosome could find its fitness independent of every other one, I parallelized the code. I’ve included both the final Java code which I used to run these experiments, and the unparallelized python code I used to draft them.

Two kinds of elitism were tried: first where the fittest member was reintroduced after the mutation step, and second where every past fittest member is reintroduced after the mutation step. This second method leads to a geometric increase in the size of a population, but avoids the “Buster Douglass” effect where two very similar and very strong members end up destroying each-other, ultimately weakening the whole pool.

Unfortunately, none of these led to notable speech patterns before they converged. Still, it was an interesting idea and I’m glad I tried it.