Image recognition using the Genetic Algorithm

Dominik Chodounský

25.5.2019

1 Assignment

The task was to use the Genetic Algorithm to find a set of five pixels, which would be able to distinguish between a dataset of 26 images. The given images were 16x16px large, black and white and represented a set of all the letters of the alphabet. The classic Genetic Algorithm was also to be modified by at least three advanced methods in order to improve it's qualities as much as possible.

2 Implementation

The assignment was implemented in Python 3. The main part of the project is made up of four functions, each of which starts a different evolution cycle. These are Basic evolution, Evolution by catastrophe, Evolution by deterministic crowding and Island evolution.

2.1) Basic evolution

This function implements the classical Genetic Algorithm in its most simple form. The algorithm consists of five stages: initialization, selection, crossover, mutation and replacement. The first generation of individuals is randomly initialized with random sets of five pixels. The population is then tested with the fitness function, in order to figure out each individual's recognition ability. Afterwards, the process of selection chooses two individuals to act as parents. They mate and as a result they crossover their genotypes using the one-point-crossover method, which uses a randomly selected intersection point. The resulting offspring can undergo a random mutation of its genotype with the probability of 7 % (value achieved through testing). This process is repeated until a new generation of unique individuals is created, after which it replaces the previous one.

The fitness function used for evaluating an individual's recognition ability can acquire values in range from 1 to 26 with 26 being the optimal fitness for our given problem. The calculation is performed as follows:

Coordinates of the individual's five pixels are compared with values which they represent in each image from the data set. Based on the values a pattern is constructed, which represents a binary number. If the grayscale value of the pixel is 255, the pattern gets appended with a '1', otherwise if the value is 0, the pattern is appended with a '0'. These created patterns are stored in a distinct set, which is afterwards checked for its length. The number of patterns it stores is the fitness of the individual.

Selecting individuals for crossover is done by the tournament method, where the fittest contender out of randomly chosen *k* gets considered.

Each implemented method is also enhanced by a Novelty search element, which adds 3 fitness points to every mutated individual. This bonus gradually wears off (-0.1 points for each following generation), making it effective only for around 30 generations.

2.2) Evolution by catastrophe

This evolution technique extends the previous method by making sure the generations don't converge to a local optimum too quickly. At a point when the progress of a population is stale for the duration of 200 generations, the entire population aside from its fittest *k* individuals (testing lead me to use k = 10) gets wiped out and replaced by

randomly initialized new genotypes. This method turned out to be useful around the mark, where generations began to reach fitness 24 and stay in this local optimum. After a few catastrophes, the random element of new individual initialization found a novelty, which boosted the fitness score to 25.

2.3) Evolution by deterministic crowding

In order to ensure a balanced population, the method of deterministic crowding was used. The method is applied after a crossover and mutation. The offspring fight with the more similar of their parents for survival and only the fittest one survives. The similarity of an offspring and its parents is determined by a distance function, which calculates by how many pixels the genotypes differ.

2.4) Island model evolution

For the purposes of maximizing the evolution process, all of the previous methods were combined and used in the Island evolution model. There are three different populations, each evolving on its own using its own defined evolutionary technique (one has basic evolution, other has catastrophes and the last has deterministic crowding). The islands evolve together in cycles, but for each cycle, independent populations evolve into a new generation. Every cycle, there is a slim chance of island crossover (3 %). When this happens, each of the three populations gets sampled and 7 individuals are picked out to be sent to the other two island to replace random 7 individuals there. This allows for new and interesting genes to be created within our ecosystem.

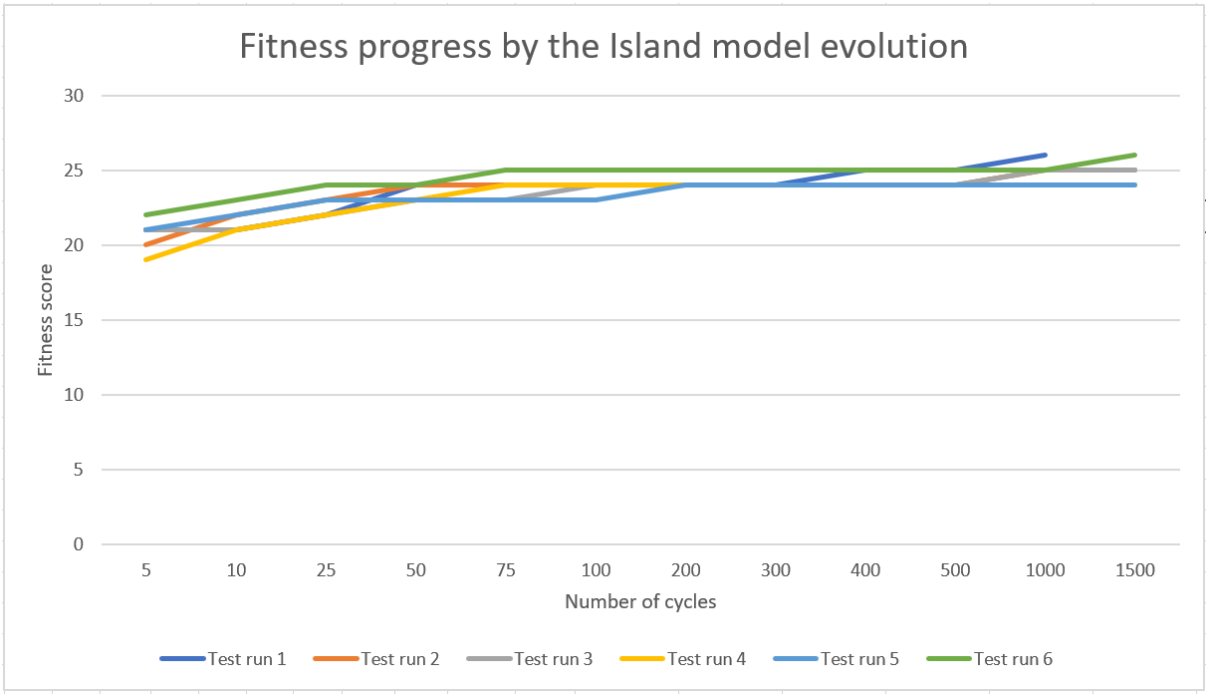
3) Experiments

The basic evolution method and the deterministic crowding method were unable to reach the optimum during my tests. The basic evolution method is a simple hill climbing technique and is prone to getting stuck in a local optimum. In contrast, the catastrophe evolution was able to swing itself out of these local extremes, however, important genes were sometimes lost in the process. The run of the deterministic crowding method was interestingly very different to the others. The other methods usually found the fitness of around 24 fairly quickly and then abruptly slowed down, but the crowding method took longer and slower steps in searching for the higher scores. The best method was probably the island model evolution, which combines the advantages (and disadvantages) of each of the previous methods. It reaches the several highest points of fitness very quickly and over time continues to make further progress until it reaches the global optimum. Fitness of 24 is usually reached within 80 cycles, 25 tends to be within several hundred cycles and the optimum takes significantly longer, some results were found to take over 1000 cycles.

Here are the results of the conducted experiment. Used population size was 120 individuals, mutation rate = 7 %, crossover rate = 50 %, selection round participant count = 5, survivor of catastrophe count = 10, island crossover rate = 3 % and island crossover count = 7. All values were a result of adjusting parameters while trying to increase the results of the experiment.

Number of cycles	5	10	25	50	75	100	200	300	400	500	1000	1500
Test run 1	21	21	22	24	24	24	24	24	25	25	26	
Test run 2	20	22	23	24	24	24	24	24	24	24	25	25
Test run 3	21	21	22	23	23	24	24	24	24	24	25	25
Test run 4	19	21	22	23	24	24	24	24	24	24	24	24
Test run 5	21	22	23	23	23	23	24	24	24	24	24	24
Test run 6	22	23	24	24	25	25	25	25	25	25	25	26

3.1 - Table shows the reached fitness score (yellow part) of each of 6 performed test runs in relation to the cycle in which the island evolution method was.



3.2 – Line graph visualizing the data collected in the previous table 3.1.