Bio-Inspired Artificial Intelligence - EMATM0029 Exercises - Artificial Evolution

TED talks

Watch the following videos related to cellular systems, reflect on the potential and limitations of the approaches described:
https://youtu.be/D3zUmfDd79s?list=PLPwd13OaGbndtCwvT1hUieedttfKEbMmX
https://youtu.be/MZGOr94468w?list=PLPwd13OaGbndtCwvT1hUieedttfKEbMmX

Exploring artificial evolution

Evolving a cellular automaton

Parameters used in artificial evolution influence the speed at which solutions are found, and if the optimal solutions are found at all! To explore this, run the genetic algorithm demonstration here for 20 years:
http://math.hws.edu/eck/jsdemo/jsGeneticAlgorithm.html

Store the "high score" and "average score" values in a vector in Matlab.

Run the experiment again with the mutation and crossover probabilities set to 0. Store the new "high score" values in Matlab.

Plot the "high scores" from both experiments. Is there a difference and why?

Plot the "average scores" from both experiments. Why is the average score of the population increasing in the case where there are no mutations or crossovers (think about selection)?

Run the experiment again with the mutation probability set to 10 and crossover probability set to 100. Store the new scores in Matlab.

Plot the "high scores" from all experiments, is there a difference for high mutation and crossover probabilities and why? Would you expect the diversity to be higher or lower than in previous experiments?

Are high-scoring individuals conserved from year to year? What type of selection could be used to prevent the loss of good individuals.

Evolving a walker

Run this simulation about genetic walkers:
http://rednuht.org/genetic_walkers/

Set the motor noise to 0 and the "Champions to copy" to 20.

Do the genomes change over time (check the names of the walkers)? Why is that?

Set the "Champions to copy" to 2, does the performance improve over time?

Run the simulation with the default parameters (refresh your browser). Set the simulation speed to 1000.

After 20 generations, does it look like the solution has converged (e.g. similar genomes are receiving the highest score)?

How would you change your parameters to decrease diversity? How would you change the parameters to increase diversity? Try these parameters in the simulator without restarting it. Does evolution behave as predicted?

Designing an evolutionary algorithm

For each of the objectives below: 1) describe a genome representation and how you would map it to a phenotype; 2) describe two fitness function that could lead to the desired behaviour.

- evolve a neural controller for a robot that solves a maze - going from point A to point B. The robot will need to make at most 10 turns. Favour fast solutions.
- evolve a program that can compute the sum of 3 numbers.
- evolve a cellular automaton which allows a team of 10 agents in a 100x100 grid to aggregate in the same location.

Based on your answers above, ask yourself the following questions: Does your genome allow for a solution that could solve the problem?

Is your fitness, explicit or implicit? Is there an obvious solution that optimizes your fitness function and is undesirable? Can you imagine the fitness landscape in your head?

Impact of parameters

In your own words, what is the effect of the following changes on the parameters of an evolutionary algorithm:

- Does making your problem more difficult for most agents increase or decrease selection pressure?

- How does increasing mutation probability impact diversity? And your ability to find a good solution?
- How does proportional selection and rank-based selection impact diversity and your ability to find a good solution?
- How does the tournament size (k) impact selection pressure when using tournament selection.
- How does increasing your population size improve your ability to explore rugged fitness landscapes?

Debugging evolutionary algorithms

Imagine you're evolving a neural network to control one of the players in the game of mario kart. It's been 100 generations and the cart is still quite useless - you're not seeing any improvement to the fitness.

What are some things you can try to improve your chances of finding a solution?

Optional: Evolving laws of physics

In matlab, write a simulator that models a ball with mass $m = 1kg$, being thrown with angle $a = 45\,\text{deg}$ and with velocity $v = 10m/s$. Gravity is also acting on the ball with $g = 9.81m/s$. For each time t in $[0 : 0.1 : 100]s$, calculate $x = vcos(a)t$, and $y = vsin(a)t - 1/2gt^2$.

Here's an example of what your matlab script could look like:

```
function simulateforce()
t=[0:1:100]
a=deg2rad(45)
v=10
g=9.81
x=v*cos(a)*t
y=v*sin(a)*t-1/2*g*t.^2
values = [x',y',t']
save('values.txt','values','-ascii')
```

Program an evolutionary algorithm that predicts the laws of physics above. Assume that you know about the necessary operators($*, +, cos, sin,$) and operands(constants, t, and a). Start by only using evolution to predict x.