CM3020: Artificial Intelligence

Summary

Arjun Muralidharan

28th November 2021

Contents

1 Evolving creatures with genetic algorithms **3**

1.1 Introduction to evolution theory . . . 3

1.1.1 Bio-Inspired Computing . . . 3

1.1.2 Evolution Theory . . . 3

1.2 Artificial Evolution and Genetic Algorithms . . . 4

1.2.1 Why do genetic algorithms work? . . . 6

1.2.2 Karl Sims Creatures . . . 6

2 Intelligent Agents **7**

2.1 Properties of Task Environments . . . 7

2.2 Agent Programs . . . 8

2.3 Pybullet & URDF . . . 8

1 Evolving creatures with genetic algorithms

Learning Outcomes

- ✓ Describe and discuss key issues in a range of branches of Artificial Intelligence.
- ✓ Select appropriate artificial intelligence algorithms for particular scenarios and evaluate their adequacy and efficiency using appropriate techniques
- ✓ Create working computer programs that implement specific artificial intelligence techniques
- ✓ Describe how information can be represented in an artificial intelligence system and select appropriate representations for different types of information

1.1 Introduction to evolution theory

1.1.1 Bio-Inspired Computing

Bio-inspired computation has emerged as one of the most studied branches of AI during the last decades.

Cybernetics (1950s) The science of control and communication in the animal and the machine. Co-ordination, regulation and control will be its themes, for these are of the greatest biological and practical interest.

Connectionism (1960s) The perceptron is a minimally constrained "nerve net" consisting of logically simplified neural elements, which has been shown to be capable of learning to discriminate and to recognize perceptual patterns. *This is the birth of deep neural networks.*

Artificial Life (1980s) Simulation of evolution, bio-primordial chemistry, completely artificial complex systems and more. *"Life as it could be".*

Genetic Algorithms Procedures that allow you to solve problems with iterative solutions.

1.1.2 Evolution Theory

Darwin proposed three generalisations:

1. Variation: no two individuals are identical within a population.
2. Hereditary characteristics: the variation expressed as individual characteristics is in heritable from parents to offspring.
3. Multiplication and competition: Malthus's observation — about population increase where no competition for resources exists —— means that as resources are always finite, competition must act as a brake on population growth.

If these forces act, then hereditary variations within a population that allowed an organism to survive were more likely to be passed down. This process is called natural selection.

A **genotype** encodes the characteristics of the individual. This is expressed in DNA as the strands of the double-helix. A **phenotype** is the actual individual that manifests in the world. It is the actual expression of the genotype when taking environmental factors into consideration.

DNA is made up of four nucelobases, *adenine (A)*, *guanine (G)*, *cytosine (C)*, and *thymine (T)*. DNA gets converted into proteins that then define how an individual works. These combine with sugars and phosphates to create *nucelotides*, which are the molecules making up DNA.

The nucelotides of an individual then gets mapped to 21 different amino acids, which then form proteins. In 2021, DeepMind was able to model and predict which amino acids get expressed as what proteins (AlphaFold).

The environment applies **selective pressure** to phenotypes. **Fit** phenotypes are more likely to reproduce.

Fitness landscapes describe the peaks and thoughs of evoluationary success within a population.

- An individual within a species is represented as a string of genes that defines its genotype. The string itself has a real number associated with it.
- This number defines the fitness of the string in terms of the phenotype it produces.
- The distribution of fitness values over the space of all genotypes gives the fitness landscape, and all members of the population map onto that landscape according to their fitness value.
- Selection and mutation causes individuals to "walk" to the nearest peak.

Simple fitness landscapes only map two traits in relation to each other and can be visualised in a 3D surface graph (with the height being the level of fitness).

1.2 Artificial Evolution and Genetic Algorithms

Genetic algorithms are *probabilistic search procedures* designed to work on large spaces involving *states* that can be represented by *strings*.

With regards to evolution theory, probabilistic search represents *selection*, while the states relate to *phenotypes* and the strings are the *genotypes*.

The *search space* is a space of possible solutions to a problem. If we can search the space, we can find a solution.

The space can be represented by the length for a *bit string*, with the possible values being the genotypes, and an actual value for the bit string would be the phenotype.

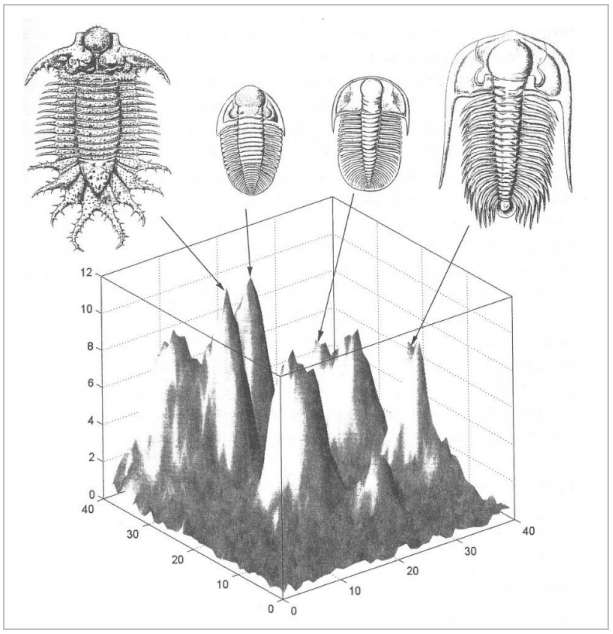

Figure 1. *Fitness Landscapes*

Each phenotype that is found in the space is given a **fitness score**. The fitness score is then used to determine the likelihood of this individual being selected for breeding.

An analogy is to map the fitness scores to the angles of a **roulette wheel**, spinning the wheel and selecting two parents. The fittest invidiuals are more likely to get selected, but it is still possible for others to get selected.

Breeding consists of two steps: **Mutation** and **Crossover**.

Crossover Takes parts of one parent's genome and the other part from the other parent. A basic version is *single point crossover* where the entire sequence up to a digit is taken from parent A, and the rest from parent B. This could be selected at random.

Mutation Once the crossover is done, the new genotype gets some bits flipped randomly, which represents a mutation. This is then expressed in a new phenotype with crossover and mutation elements.

Based on this theory, a canonical form of a **genetic algorithm (GA)** was formulated:

1. Set $t = 0$
2. Initialise chromosome population $P(t)$.
3. Evaluate $P(t)$ using fitness criteria
4. **while** termination condition not satisfied **do**
5. (a) Select best individuals from $P(t)$. Let P1(t) be the set of selected chromosomes. Choose individuals from $P_1(t)$ to enter mating pool (MP).

(b) Recombine chromosomes in MP forming populations P_2. Mutate chro- mosomes in P_2 forming P_3.

(c) Select replacements from P_3 and P (t) forming $P(t+1)$.

(d) Set $t = t + 1$

1.2.1 Why do genetic algorithms work?

Genetic algorithms try to find a **global maximum** of the fitness function in a multi-dimensional space.

Hyperplane sampling is a technique that breaks down a problem into components that can be recombined over and over so we can test different solutions.

Schema theorem says that there are patterns (schemas) like a simple regular expression. For example, we could freeze some bits of the genotype and iterate other bits by defining a schema to use (e.g. 100****1 would use the fixed bits but try out different bits for the *).

Implicit parallelism takes the population and compares two individuals in sequence, one at a time.

Computational parallelism applies this to a computing environment, where you can evaluate multiple pairings individuals in parallel computing threads, depending on how many cores your processor has.

1.2.2 Karl Sims Creatures

Evolving Morphology *Dawkin's Biomorphs* express the idea that evolution can create very complicated morphologies over time through the process of iteration. Using small mutations of biomorphs - abstract figures made to look vaguely like creatures in nature, Dawkins clearly showed how the process of evolution can create very complicated organisms if given a long period of time.

It is presented as a counter-argument to the idea of 'intelligent design' - where the complexity of life is said to attest to the existence of a higher creator-being, commonly denoted as 'God'.

Sims Creatures Karl Sims' paper from 1994 describes a system for creating virtual creatures that move and behave in sumulated 3D physical worlds.

Creature morphology The morphology or genotype is represented as a **directed graph** where the nodes are similar to joints and the vertices are the movable "arms" of the creature.

The graph can contain *multiple connections*, meaning that a node with multiple connections to another node means that the directed node is repeated multiple times.

The graph can also contain *recursion*, so an edge from a node to itself means that the node is connected to another identical node.

Creature neural architecture Mapping from sensor input to actuator output. The graph as a whole percepts some input and produces an output for that input.

2 Intelligent Agents

Summary of Chapter 2 from Russell/Norvig

In AI, we generally want to build *rational agents*. Rationalaity is defined as acting in a way that achieves the best expected outcome.

An agent is anything that *perceives* its *environment* through *sensots* and takes action upon the environment through *actuators*.

The **percept sequence** is a history of all percepts of an agent, and the **agent function** maps any given percept sequence to an action.

Definition of a rational agent For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowlege the agent has.

Note: Rationality does not mean omniscience, rather, rationality deals with expected performance, not absolute certainty.

A **task environment** is defined as consisting of:

- **Performance Measure:** The desired outcome of our agent.
- **Environment:** The full, partially or un-observable environment of the agent.
- **Actuators:** The means by which the agent can act on the environment.
- **Sensors:** The means by which the agent can perceive the environment, and store a percept sequence.

2.1 Properties of Task Environments

Fully observable vs. partially observable How much of the environment can a single percept through sensors perceive?

Single-agent vs. multi-agent How many agents are acting, and are they **competitive** or **co-operative** (or neither).

Deterministic vs. non-deterministic Is the next state of the environment completely determined by the action of the agent? If non-deterministic, is the probability of outcomes known (is it *stochastic*)?

Episodic vs. Sequential Do actions impact future actions?

Static vs. Dynamic Does the environment change while the agent is deciding on the action?

Discrete vs. continuous Does the agent need to act on discrete environments (e.g. single objects) or a continuous environment through the flow of time?

Known vs. unknown How much has the designer of the agent built-in knowledge about the world?

2.2 Agent Programs

A basic agent program is defined as taking a *percept*, appending this to the *percept history*, looking up an action in a table representing the agent function, and returning the action. The percept history and table are persistent elements of such an agent, and would be programmed as persistent data of an object.

Simple Reflex Agents These agents select an action based solely on the *current* percept and refer to a set of *condition-action rules* to decide the action. A common problem is running into infinite loops (because the agent doesn't use previous percepts and might try the same thing over and over again), which can be overcome by **randomization** of actions (e.g. randomly turn left or right as a vacuum).

Model-based reflex agents These agents maintain an *internal state* of the environment to overcome partial observability, by storing a *transition model*, which describes how the actions of the agent impact the environment and how the environment evolves independently, and a *sensor model*, which describes how the current state is reflected in sensors. This agent updates its internal state using the current percept, the current state, the most recent action and the two models. It then checks the state against a set of rules and chooses the action.

2.3 Pybullet & URDF

Pybullet is a physics engine that can simulate objects in a 3D environment using Python. We also use a file format called the **unified robot description format** (URDF) to describea "creature" in the simulation, a phenotype or specific expression in our evolutionary space.

In creating 3D simulations, it is useful to have **separate collission and visual geometry**, meaning that complex shapes are simplified when calculating collisions (e.g. a creature is surrounded by an invisible box, and the box is used for detecting collisions). This is done to manage how complex the computations will be, as more complex shapes need more computation to detect ongoing collissions.

URDF An XML-based filed format for specifying robot models, including their motors and other dynamics. It's useful for programmatically generating models, and easily share models. Pybullet provides some premade models via a `data` module.

Listing 1 Basic URDF File

```
<?xml version="1.0"?>
    <robot name="myfirst">
        <link name="base_link">
            <visual>
                <geometry>
                    <cylinder length="0.6" radius="0.2"/>
                </geometry>
            </visual>
        </link>
    </robot>

```

URDF files define **links** (which are the nodes) and **joints** which are the edges between the nodes. This expresses a robot as a **directed graph**.