Artificial behavioural ecology:

Using artificial intelligence to study how brains, bodies, and environment produce behaviour

Recent advancements in artificial intelligence have the potential to make a great impact on how we study animal behaviour. However, the tools in AI and the types of questions it can open up are relatively new. This book will introduce AI tools and guide a reader through how they can be used to address questions in both animal behaviour and AI. Ideally, at the end the book, the reader will be better equipped to see AI from an animal behaviour perspective, and understand how AI can be used to better understand how brains, bodies, and environments all contribute to the evolution and development of behaviour.

Table of contents

Preface: Why artificial behavioural ecology (ABE)?

**Chapter 1: building behaviour from the ground up**

*What is BE and AI, and their synergy:*

* 1. What is Behavioural ecology (BE)?
  2. What is Artificial Intelligence (AI)?
  3. Synergies

**Chapter 2: Setting up an ABE workshop**

*What are the important components:*

* + 1. Brains
    2. Bodies
    3. Environments
    4. Tutorial: Setup CleanRL

**Chapter 3: Shaping behaviour with reinforcement**

*Understanding the basics of reinforcement learning algorithms*

* 1. Introduction to reinforcement learning
  2. Value based approach
  3. Policy based approach
  4. Actor-critic algorithms
  5. Associative learning algorithms
  6. Tutorial: A2C

**Chapter 4: Generalized learning from current states**

*Learning in complex worlds with elaborate bodies and one lifetime*

* 1. Functional approximation of the A-learning algorithm
  2. Tutorial: test different A2C neural architectures
  3. Short term memory - eligibility traces
  4. Tutorial: A2C add eligibility traces

**Chapter 5: Providing the benefit of sight**

*Deep Reinforcement Learning:*

* 1. Providing sight
  2. Benefits of learning generalizations
  3. Costs of computation
  4. Tutorial: letting our agents see the world

**Chapter 6: When is learning hard?**

*How to think about behavioural complexity:*

* 1. Chains of behavioural sequences
  2. Favourable entry points
  3. Role of the environment
  4. Tutorial: foraging for pinecones

**Chapter 7: Using evolution and development to shape learning**

*How to inherit contextual learning:*

* 1. Within vs. between generation learning - how much prepared learning is needed? Why?
  2. How to monitor development of behaviour
  3. How to use evolution to shape an agent
  4. Hyperparameter tuning using evolutionarily algorithms
  5. Questions: Growing a network
  6. Tutorial: evolving to learn fast or slow

**Chapter 8: Prepared learning**

*Adding in context dependencies:*

* 1. The problem of learning fast/slow: AI and BE perspectives – millions of examples needed?
  2. Role of evolution in prepared learning: i.e., no blank slate here!
  3. Inherited body
  4. Inherited context dependent behaviours
  5. Inherited context dependent learning
  6. Inherited perception
  7. Inherited context dependent rewards
  8. Inherited context dependent short-term memory
  9. Interaction and interconnections (i.e., between perception and motor?)
  10. Tutorial: between generation perceptual learning

**Chapter 9: Social learning**

*Multi agent DRL:*

* 1. Social learning in BE and animal cognition
  2. Different mechanistic explanations: social brain hypothesis, cognitive buffer hypothesis, elaborate sensory-motor hypothesis
  3. Imitation – theory of vertical associations (links between perceptual and motor states)
  4. MADRL – multi-agent deep reinforcement learning
  5. Tutorial: competing/collaborating animal troops

**Chapter 10: Dealing with complexity and false equivalency**

12.1 What/How can we learn from our simulated agents?

12.2 Fallacy of affirming the consequent

12.3 Simplicity vs complexity

12.4 Reductionism vs constructivism

12.5 An answer from cybernetics

**Chapter 11: Conclusions**

*Take away messages:*

* 1. Artificial Behavioural Ecology: overview of what we learnt
  2. Questions in ABE: Tinbergen’s four questions extended/new light
  3. Collaborating in ABE: a zoo for simulated organism with known training/evolution histories
  4. Collaborating in ABE: simulated skinner boxes for standardized testing
  5. Exporting to robotic bodies: spot quadruped
  6. What’s missing? Is the concept of the mind missing? Evolutionary psychology perspective / cognitive and behaviourist takes
  7. Future considerations

**Appendix 1: Common computational challenges and pitfalls**

*Keeping track of problems:*

A1.1 Hardware needs

A1.2 Software options

A1.3 …

Preface

There are many ways that we can study the behaviour of animals. In this book we will focus on the approach taken in behavioural ecology. The field of behavioural ecology studies the evolution and development of behaviours that allow animals to thrive in their environment.

What can AI bring to the study of behavioural ecology? Traditionally, behavioural ecology has focused on the evolution of inherited behaviours, assuming the learning mechanisms responsible for the development of these behaviours are not central. This has been termed the behavioural gambit, and it is a widespread assumption. The up-side to this assumption is that we can study how behaviours evolve over time in a population. The down-side is that we assume that animals can always learn the optimal behaviour, i.e., that there is no cognitive limit to what an animal can learn. It also assumes relatively static behavioural strategies and limits the impact of changing environments. Understanding how individuals adapt their behaviour to changing environments becomes important if we want to understand how animals might respond to changes not seen by that species in the past. Rather there is a need to incorporate both learning and inheritance when trying to understand animal behavior.

This is where recent advancements in AI can be particularly useful. A set of algorithms called re-enforcement learning (RL), allow for both within lifetime learning as well as inheritance to play crucial roles when it comes to the development of behaviour. These RL algorithms have a long history, and closely approximate associative learning theories that were developed in animal behaviour and psychology [ref Thorndike 1898, Pavlov 1927, Skinner 1938, 1963].

In this book we’ll see how we can use these RL algorithms to build simulated animals (i.e., agents) that are not subject to the behavioural gambit. These simulated animals will inherit RL-structures that will shape their learning of behaviour during their lifetime. We’ll also see that by allowing behaviour to be influenced by within lifetime learning and inheritance, we can tackle old questions in behavioural ecology from a new angle, and ask completely new questions.

I’ve termed this approach artificial behavioural ecology (ABE), as it simply adds a new simulation-based approach to an old field. The use of simulation is common in maturing fields in science [ref Margaret Morrison – reconstructing reality], and represents a long ark from the development of associative learning theory at the turn of the century, to the mathematization of these theories in mathematics/computer science, and finally to their practical implementation in complex and changing environments today. The ability to implement these associative learning theories in simulation allows us to better study the development of behaviour, something that has remained somewhat opaque due to the complex interaction between brains, bodies, and environments. It will also allow us to study AI algorithms from a new point of view, applying insights and methods from behavioural ecology to better understand how AI agents develop behaviour that allow them to thrive in complex and changing environments.

Chapter 1: Behaviour from the ground up

*What is behavioural ecology and artificial intelligence?*

* 1. What is behavioural ecology?

In the deserts of Western Iran, a bird sits on a tree and watches a spider move back and forth in a shaded area. The spider climbs and circles a rock. From past hunting experience the bird waits patiently for the right moment to swoop down. The spider turns to move away. The bird strikes. The spider, however, is not a spider, but the tail of a viper – a spider tailed horned viper – used to lure prey withing striking distance (ref Marlene Zuk: dead man test book). The vipers’ imitation of a spider walking and climbing around has been a success, and now it settles down for lunch.

As a behaviour, the use of a tail, that resembles a spider, and the ability to move in a convincing way is a spectacular achievement. The question that naturally arises when witnessing the performance, is how did that behaviour evolve? Scientists that research this fall into the field of animal behaviour, and apply evolutionary theory to try and understand how behaviours like this come to be. Animal behaviour has a long history, and has grown many sub-fields that focus on specific types of questions, or take a particular approach to studying behaviour. Ethology is particularly interested in documenting and recording behaviour in the wild. Animal cognition is particularly interested in understanding the cognitive mechanisms used by animals to make decisions and has laboratory and field components. While behavioural ecology (BE) is interested in how behaviours of animals evolve and how these behaviours allow animals to thrive in their specific environments.

These fields of study have done much to understand how a viper could evolve over many generations to have a spider like appendage at the end of its tail, and how it can learn to use the tail in a way to attract a bird within striking distance. To gain this knowledge, these fields use observation of animals in the wild and in captivity and monitor how behaviour changes and responds to changes in the environment. These observed changes can be compared to predictions based on competing theories, allowing for better and more precise theories to be developed. Similarly, under more controlled conditions it has been possible to keep the environment relatively fixed and change one thing to see how an animal might respond. This experimental approach helps pinpoint the consequences that a change in the environment has on behaviour. These different approaches to studying animal behaviour has moved the field forward.

It is, however, increasingly possible to use a constructionist approach to complement the diverse fields in animal behaviour. A constructivist approach can be understood from the oft repeated saying from Richard Fineman: “what I cannot create, I do not understand.” The idea is that through trying to build something we often find insight into how and why something doesn’t work the way it should, or simply does not work at all. And if we can create something, it often suggests that we understand something about it. Though that is not always the case. Developments in artificial intelligence (AI) have made it possible to build simple artificial animals (i.e., agents), and use these simple agents to develop insights in the development of behaviour in specific environmental conditions. The addition of this constructivist approach is likely to benefit both animal behaviour and artificial intelligence fields.

* 1. What is Artificial Intelligence (AI)?

How is it that a snake “knows” how to move its tail like a spider? Or that its behaviour will attract prey? Cognitive scientists and philosophers have developed theories of how animals, including humans, develop knowledge and use that knowledge to make decisions and take actions. The field of artificial intelligence has taken a constructivist approach to these long running fields of scientific investigation. AI can be seen as a general attempt to create objects/agents that are able to perform intelligent behaviour. A lot of work and thought has gone into what “intelligent” means, but here we’ll use intelligent as the ability of an organism or agent to adapt its behaviour to take advantage of an environment. A highly intelligent foraging agent then is one that can be placed in an environment and effectively learn behaviours to gather food, even in the face of environmental changes. While a low intelligence foraging agent is one that cannot develop behaviours to gather food in static or dynamic environments.

*Intelligence ~ the ability of an organism or agent to adapt its behaviour to take advantage of an environment*

AI, just like animal behaviour, has many sub-fields. One field in particular has relevance for studying animal behaviour, that is, re-enforcement learning (RL). This field largely emerged in the 1980s, and was influenced by work on associative learning being developed in psychology and animal behaviour (ref Sutton & Barto). Work in RL explicitly states associative learning theories in mathematical notation rather than verbal statements. This mathematization resulted in a plethora of RL algorithms that were put to work on ever increasing and complex environments, with increasingly impressive results. From winning against humans in tick-tack-toe and backgammon, to playing video games, and beating world masters at chess and the game of GO [ref chess, go, atari]. Most recently, the use of images as inputs to these algorithms have allowed these algorithms to play video games better than humans, and learn in more complex environments with limited input from a human. These newer algorithms that can take images as inputs and use neural networks, a mathematical structure that transforms inputs into ever more abstract and generalized forms, to learn what to use from an image to complete its task. The combination of RL with neural networks has been termed deep reinforcement learning algorithms, and is a very active field within AI [ref 2020 overview paper].

With the added complexity of deep neural networks, however, it becomes ever more difficult to understand how agents using these algorithms learn, as well as how to build these algorithms to produce optimal behaviour in specific as well as varied environments. In AI it is possible to create these agents, however, it becomes increasingly difficult to understand them. This is a somewhat similar problem, though in reverse, when thinking about how behaviour develops from a behavioural ecology perspective (Fig. 1): i.e., in AI we can create intelligent agents, but it is becoming more difficult to understand how intelligent behaviour develops? While in behavioral ecology we have tools to understand how behaviour develops and evolves, but we have difficulty understanding how intelligent behaviour is created.

Table 1: The roots of artificial behavioural ecology

|  |  |
| --- | --- |
| **Research field** | **Goals & limitations** |
| Behavioural ecology | We understand how behaviour develops and evolves in natural settings; we don’t understand how intelligent behaviour is created |
| Artificial intelligence | We create agents with intelligent behaviour, but have trouble understand how behaviour develops |
| Artificial behavioural ecology | We create agents with intelligent behaviour, and study the development and evolution of behaviour |

A close-up of a sign

Description automatically generated

Figure 1: A visual diagram of how artificial intelligence and behavioural ecology can be seen as complimentary when it comes to studying animal behaviour. We call this synergy: artificial behavioural ecology.

* 1. Synergies

The complimentary goals of AI and behavioural ecology suggest that a synergy between these fields could be very advantageous to answering questions in both. In behavioural ecology there are generally four main questions that are asked.

Two generative:

1) How a behaviour evolves through successive generations?

2) How a behaviour develops during a lifetime?

Two mechanistic:

3) What triggers a behaviour?

4) What function does the behaviour serve?

We will see in this book that by building agents using RL we can allow for prepared learning to pass between generations of agents, and how prepared learning can speed up and guide learning during the lifetime of one agent. Using this approach, we’ll learn how to setup agents and their environments to answer all 4 of these questions (Table 2).

Table 2: Tinbergen’s 4 questions in behavioural ecology.

|  |  |  |  |
| --- | --- | --- | --- |
| **Level of question** |  | **Object of study** | |
|  | **Contemporary**  (Focus on the present day) | **Historical**  (Focus on the past) |
| **Proximate**  (How?) | Mechanism  (How is the behaviour triggered?) | Development  (How did the animal develop the behaviour?) |
| **Ultimate**  (Why?) | Function  (Why does the animal perform the behaviour?) | Evolution  (Why did the behaviour evolve?) |

From an AI perspective we will test our agents in a number of controlled environments to see how they compare to studies in animal cognition. We’ll also use observational techniques in ethology and behavioural ecology to better map what kinds of RL structures will produce optimal behaviour in specific environments.

Finally, from the point of view of both fields, we will use intelligent agents to better understand the sometimes-opaque implications of associative learning outcomes under various environmental context, i.e., build a “what if” simulator. E.g., what if we had agents who were capable of social learning and lived in a patchy resource environment, would they develop coalitions to protect food? If so, under what conditions is this the case, and when not? The idea here is that we as humans are limited in our ability to work through the implications of theory in complex real-world environments, and that we can use simulation to help develop intuition and test our theories.

Chapter 2: *Setting up a workshop*

*What are the important components that produce behaviour?*

|  |  |
| --- | --- |
|  | *“It becomes harder and harder to say where the world stops and the person begins.”*   * *Andy Clark* |

* 1. Brains

To think about what a brain is and does there have been many metaphors used over the centuries. They tend to relate to the most complex objects of the day: e.g., clocks, machines with springs and pressure values, governors of a steam engine, telegraph, and telephone exchanges. It’s not surprising that today’s dominant metaphor is the computer. It takes some inputs, stores it in memory and processes it following rules and functions, before returning some outputs. Metaphors can be very useful and help with our intuition, however, there are some issues with the brain as a computer metaphor that can lead us astray.

The first is that of storage, a brain, cannot be thought of as storing memories in discrete relatively permanent locations like a computer. E.g., one memory cannot be located in a specific neuron. Rather memory, as it is currently understood, is a stored process, that is likely better thought to be triggered and relived rather than simply retrieved [ref: organism as waves; self-assembling brain].

Another issue that arises from the computer metaphors is the idea that the brain creates representations of the world internally. In other words, you are creating a virtual representation of the world while you read this book. Rodney brooks showed that you can accomplish a lot with fast, cheap, and out of control robots [book ref]. Here it’s important to think about what brains are for, and how through evolution brain function was selected for, i.e., surviving and reproducing. Making detailed representations of the outside world is likely not a goal unto itself, and can be extraordinarily expensive and time consuming – something a mouse avoiding a snake does not have time to do. A good example here is [ref robot that takes 5 minutes to calculate before each move and can get easily lost]). Rather than thinking about brains as representing or re-creating the external world, we can focus on brains as producing behaviour in response to the environment that leads to survival and reproduction. A slight change in focus but likely an important one.

Of particular relevance for our purposes is the idea that the brain is hardware that stores software. This idea has led to many debates about how we differ from other animals. Do we simply have completely different hardware? When we look across the animal kingdom, we find that humans do have large brains, for our body size, and a relatively larger percentage of neocortex and cerebellum tissues [ref Barton]. However, when looking at the primates, and the cetaceans (whales), we find that hardware wise, we aren’t that different [ref Barton]. Could it be that the small differences in hardware (i.e., biological structure) make the difference between humans and other animals? One line of evidence that is emerging points to the similarities in which the brain is used. Empirical evidence suggests that associative learning abilities are found widely across many animals: from bees, to birds, to cats, dogs, dolphins, and humans [useful refs]. Understanding how associative learning works has led to being able to train animals for movies, treat mental health in humans, and in general to predict behaviour [refs].

Associative learning allows individuals to learn during their lifetime linking states/actions to outcomes. The classical work by Pavlov, who showed that dogs would associate the ringing of a bell to feeding time, showed that dogs learned to anticipate feeding from the sound of a bell. This shows that they learned the association between a state in the environment (ringing bell) and outcome (food). This is called classical conditioning. While much works on rats/birds/humans has shown that individuals can also learn which actions taken in the world lead to desirable outcomes: like a rat pressing a lever that dispenses food. Learning associations between actions and outcomes is called instrumental (or operant) conditioning.

In both cases, the individual has something that they find rewarding, i.e., being fed. This is said to be an unconditioned stimulus. By unconditioned it just means the animal did not have to learn to like food/eating. While the action (i.e., pressing a lever) or state (ringing bell) is a conditioned stimulus. That is, they learnt what actions or states led to a reward in a particular environment.

In these simple examples, we only focused on the action or state directly preceding the rewarding state, but associative learning has been shown to be very effective at chaining together many actions/states for an individual to reach some rewarding state. Think of a juvenile chimp learning to extract ants from an underground nest. They must find a good stick/grass, learn to lick it, place it down into an ant hole, then extract their tasty treat. Associative learning can learn this sequence of steps, though the longer the sequence, and the more potential behaviours that could be linked together, the more difficult learning is. This is where associative learning is thought to get some help from evolutionary history.

Animals that have been shown to possess associative learning are generally not assumed to be blank slates waiting to learn everything without any guidance or biases. Evolutionary past can shape animals’ attention as well as what actions are more likely. We’ll see in chapters 3-4 how we can take these verbal descriptions of associative learning and express them in mathematical terms to give our agents associative learning. Then in chapters 7-8 we’ll also show how it is possible to allow them to evolve and take advantage of inherited biases in perception, bodies, learning, and actions. But first we need to think a little more about how simply inheriting a specific body can also simplify and guide associative learning.

The focus on associative learning in this book, represents a view that cognition cannot be separated from actions and states, or more generally, sensory-motor feedbacks. And that brains evolved to control sensor-motor feedbacks so that organisms can produce adaptive behaviour. From this point of view brains are embedded within an environment, and embodied, rather than abstract entities that process inputs based on logical rules [ref 4Es book].

* 1. Bodies

|  |  |
| --- | --- |
|  | *Person 1: “Human brains are amazing, just look around at what they can build. Look at that bridge!”*  *Person 2: “I’d really like to see a brain build that bridge without arms and legs…”*   * ***During a meeting of psychologists and neuroscientists*** |

It seems rather trivial, and a small point, to say that a brain on its own is rather useless. But often this is something overlooked when thinking about what a brain does. Learning doesn’t happen in a vacuum. There are many ways that bodies have been shown to facilitate learning. A good example is the bush-crickets. From an outsider’s perspective, the female bush-cricket faces an almost impossible task of locating a high-quality mate. Waiting on a grass leaf, the wind, the croaking of frogs, songs of birds, and 100s of other grasshoppers from other species making mating calls fills the environment with sound. How can she hear potential mates? There seems to be a high volume of input for the female to process, yet her brain is quite simple. If we could figure out how she processes such a magnitude of information with seemingly limited resources it might provide insights on how to process high volume audio data, and get a better understanding of the efficiencies of brains more generally. It turns out, however, that the formation of her ears is the solution to the processing puzzle. The ears are formed in such a way to preferentially admit certain frequencies, reducing what looks like a very challenging information processing problem to a much simpler one [ref those structural bush cricket papers]. To her, all she hears is her potential mates in an otherwise rather quiet world.

Bodies are directly relevant for how animals learn. They can alter what stimuli are available, but can also reduce the types of behaviours possible, or even promote certain behaviours. We’ll see in chapter 7-8 how we can let evolution shape the bodies of our artificial animals to help guide them in learning. But we need to look first at another piece of the puzzle when it comes to how behaviour is learnt, and that is the environment.

* 1. Environments

The environment is as obvious a requirement as the body, and again is sometimes overlooked when thinking about how animals learn. From an associative learning perspective, the environment: the objects in it, how fast it’s changing, all other animals and their behaviour, forms the stimuli that an animal can receive. The actions the animal takes can sometimes have drastic immediate impacts on that environment (e.g., birds spreading fire), or those actions might be delayed for a long period of time (e.g., birds spreading seeds). Or the actions by an animal might have no impact of consequent. From the point of view of an animal learning in a specific environment, immediate outcomes can be easier to learn than ones that only lead to long term outcomes. If they can learn those associations at all.

A good example of how the environment can impact learning is how squirls learn to eat from pinecones (ref Enquist book). Again, this is another puzzle of how an animal achieves a rather complex looking behaviour, in this case how a juvenile squirl learns to extract pinenuts from pinecones. To successfully extract the pinenuts from pinecone, the juvenile squirl must climb a tree, find a pinecone, get it to the ground (or keep it in the tree!), remove the outer protected layer, then get the pinenut at the base. This appears like a very difficult sequence of tasks the squirl must learn to perform. Similar to the bush cricket. However, we’ve made some pretty big assumptions about the environment. For one we’ve assumed that all the pinecones are in trees, and that they are all intact. There could be many pinecones on the ground, and if other squirrels are around, there are likely pinecones in various states of being taken apart. This, suddenly richer environment, now presents the young squirl a much easier learning task. Within this environment a juvenile can learn that taken-apart pinecones have pine nuts in them, then they can learn that fully intact pinecones can be taken apart, then they can learn where to find fully intact pinecones. The environment in which this squirl is living provides favourable entry points that makes learning a longer sequence of behaviours much easier. We’ll see at the end of this chapter how to build richer environments, and in chapter 6 how to think about behavioural sequence, and how environments can setup favourable entry points to facilitate learning.

* 1. Synergy

To better understand learning in animals we need to then think about brains, bodies, and environments in which learning takes place. They are intrinsically linked, and it is difficult to treat them separately if our goal is to learn how behaviours develop and evolve in complex and changing environments.

* 1. Tutorial: setup an ABE workshop

In this tutorial section we’ll build our workshop to start to put some of these ideas into practice. We’ll go in reverse of how we did this chapter, showing you how you can use a free game development software (Unity) to build an environment. We’ll then add in a body, and give it a simple brain that can perform associative learning. At the end of these tutorials, you’ll be able to create artificial animals that we’ll use throughout this book to gain insight about how brains, bodies, and the environment produce the amazing behaviours we see every day in the world around us.

*Getting up and running – CleanRL – Show how to run default project with a few environments, how to “see” the results. Maybe test learning of different body complexities, and environmental complexities?*

* 1. Tutorial: Install and run CleanRL

– Show how to change the training environment

– Show how to change the learning algorithm

– Visual examples

* 1. Tutorial: Build custom gymnasium environment:

Similar to: <https://www.youtube.com/watch?v=AoGRjPt-vms>

– How observations, action spaces, rewards work

– How reset and step functions work

* 1. Tutorial: Build custom gymnasium bodies and environments:

Similar to: <https://www.youtube.com/watch?v=OqvXHi_QtT0>

– mujoco

You should now have everything setup to build bodies and let them behave within an environment. We’ll expand on these simple examples as we move through this book, but first we need to know a little more about how these agents can be setup to learn. In particular, we need to know a little more about reinforcement learning to build in the brains!

Chapter 3: Shaping behaviour with reinforcement

*Understanding the basics of reinforcement learning algorithms*

|  |  |
| --- | --- |
|  | *“It's a dangerous business, Frodo, going out your door. You step onto the road, and if you don't keep your feet, there's no knowing where you might be swept off to.”*   * ***J.R.R. Tolkien*** |

Associative learning can be thought about as a process that brings an agent along a path, and where they end up, and what behavioural sequences they learn, depends on their bodies, brains, and environment. But how exactly does associative learning work, and can we reproduce this in a simulation? If we could reproduce an associative learning like process, it would be possible to explore a range of questions that are difficult or impossible to explore using observation and experiment alone [ref reconstructing reality]. This is where RL algorithms come in.

RL is a branch of artificial intelligence and a type of machine learning. It, however, has a particular focus. It is primarily interested in how an agent can learn behaviours to perform in an often noisy and chaotic world which maximizes rewards. This can be a challenging task as the agent might not know how the world works (e.g., if I drop something will it fall down/up?), the world could be stochastic (e.g., I might try an action and get a different consequence each time), and the world could be changing (e.g., food might not always be in the same spot). The assumption in RL, is that if the agent receives some kind of reward (or punishment) from the environment (e.g., it might feel good to eat food), then algorithms that explore different behaviours, and exploit the behaviours that work, can eventually allow the agent to learn how to behave in a range of challenging environments. Even when the rewards are sparse or takes a long time to occur (e.g., winning at chess might only occur after many moves).

To build algorithms that can effectively explore alternative behaviours, and exploit the ones that work, it has been useful to think about the general class of problem that an agent faces. Depending on the class of problem, one of a number of RL algorithms have been shown to be more effective. We’ll spend a bit of time here thinking about the characteristics of the class of problems we are interested in this book: i.e., an organism born into a world, and must learn to behave within only one lifetime. And often quickly too, otherwise they might become prey for another animal that has already learnt rewarding behaviours. This type of problem has been explored within ecological reinforcement learning [ref ecoRL paper].

One of the first things we can think about with this class of problems is how much information the agent is receiving. That is, can we make the assumption that the agent can see all of the environment, or is the agent limited to only see a portion of the environment. In our case we must make the assumption that animals only experience part of the environment, i.e., animals only have partial observation of the environment. Sometimes to great advantage: e.g., bush cricket. What part of the environment, and how, an animal sees the world is called the *umwelt* [ref Uexkull, ethology]. Did you note my human biases towards the visual senses in writing the last sentence: “*sees* the world!” The idea of the umwelt will play an important role in setting up our virtual worlds and ultimately in understanding behaviour more generally.

A second characteristic of the class of problems we will cover in this book, is that the agent can only have one lifetime. That is, we can’t assume that the agent will have many attempts at the same problem always restarting back to the same starting point (i.e., learning through repeated episodes). Rather the agent will live one life, where all its behaviour and learning accumulates over time. This characteristic will have important implications when it comes to adding other agents into the same environment. Here each agent will learn separately, though we’ll see in chapters 10 how agents might learn by imitation of other agent’s behaviours. Similarly, with only one lifetime to work with we’ll see that inheritance from parents becomes a way for our simulated animals to guide and speed up learning: i.e., we’ll see more in chapter 7-8 about prepared learning and inheritance.

A third characteristic is that the animal only has real-world experience to use. That means that the animal is directly affected by their actions as they are trying to learn. This might sound a little strange, but there is a class of RL problems that can be solved through offline-learning, where the agent is learning how to behaviour, but does not necessarily put that learning to use when choosing how to behave at that moment. In our case we are interested in online-learning where agents must learn actions in real-time.

So, the class of problems we are interested in are cases where individual agents only have partial observations from the environment, live only one lifetime, can be prepared to learn through inheritance from their parents, and apply what they learn to help them choose how to behave in real-time. That is, we are interested in evolution and development of behaviour in conditions similar to that of real animals.

* 1. Some quick history

The roots of reinforcement learning lie between psychology, neuroscience, mathematics, and computer science. Historically, the founders of reinforcement learning have used, or been trained in, theories of associative learning [Barto history RL]. From this background, algorithms that approximate associative learning have been proposed: both for classical conditioning and instrumental conditioning. However, in the RL literature, algorithms that estimate the value of states (i.e., classical conditioning; think Pavlov’s dogs) are called value-based algorithms, and algorithms that estimate the value of actions within states (i.e., instrumental conditioning; think rat pushing a leaver to get food) are called policy-based algorithm. In this chapter, we’ll first go over how to think about states, actions, and rewards, then take a look at each of these types of algorithms (i.e., value-based, and policy-based), before introducing one class of algorithm that combines them. The combined class of RL-algorithm has been proposed as a good approximation of associative learning when modeling animal behaviours and we’ll introduce one algorithm called A-learning in particular that we’ll get to know well [ref girlanda, enquist, lind]. As a field, RL has a rich variety of algorithms that is growing every year, however, in this book we’ll focus on the A-learning algorithm as its goal meets our own; understanding how brains, bodies, and environment produce behaviours.

* 1. Markov Decision Process

When thinking about an RL problem it has been useful to think about it from the point of view of states, actions, and rewards. One way to formalize this is to use a Markov Decision Process (MDP) which defines the probabilities of transitioning from one state to another, how likely actions are in each state, how those actions influence transitions between states, and the rewards that the agent accumulates. Let’s break this down a little:

State (s): A state is the set of values that describe the current context. This can capture things like the hunger levels of the animal (internal), the current visual image (external), and past visual images. A state can be made up of any combination of internal, external, or past/present variables.

Transition probabilities (p): The probability of going from one state (s1) to another state (s2) is the transition probability. By interacting with the environment the agent can learn these transition probabilities.

Action (a): An action is the set of potential behaviours that the animal can produce. These actions are motor based, can be discrete or continuous, and can depend on the current state.

Rewards (r): When an agent is interacting with the environment it can receive a positive or negative signal. These signals can be thought about as internal to the agent (i.e., release of dopamine when eating), and can be a result of the agent’s interaction with the environment.

Policy (): a policy is a plan on how to behave to maximize rewards. Ideally, the policy will define the actions to take when in a particular state, to achieve a maximum accumulation of rewards.

By breaking down a RL problem into these discrete parts it can help to better understand the challenges faced by an agent. From the point of view of the agent, the goal is to learn the combination of states and actions that will maximize the return of positive rewards.

This description of an MDP requires states and actions to be discrete. For example, the agent choses to take action ‘eat,’ while a continuous action space would simply adjust the force applied to the legs, arms, torso of the animal to produce eating behaviour. We’ll see in chapter 7 how to think about the more continuous case of an MDP using Functional Approximation. But the main ideas of this simplified MDP still apply, and can be very useful when understanding how RL works more generally.

* 1. Value-based algorithms (Pavlov’s Dog)

Now that we understand the components of a RL problem, let’s see how an agent might go about learning what states are valuable and lead to rewards. As this is our first bit of math, we’ll take our time to introduce all the pieces. We’ll see these equations again and again throughout so there is lots of time to get used to them. The first thing we need is to define the value of a particular state, let’s call it w(s). This is what we want our agent to learn, but how do we allow the agents experience to update the value of a state. We’ll need something that takes the old expected value of a state w(st), and after interacting with the environment for a time update it w(st+1) to hopefully be closer to the actual reward of that state. We’ll need something like the following:

or more simply, we can model the change i.e., in w(s):

There might be many ways to choose to update this expected value of a state, however work in RL suggests that an efficient way is to use the observed rewards of the next step minus the expected value of the current state. It might seem a little strange, but by using the next state to inform the current state, we allow the agent to learn sequences of behaviours that lead to a rewarding state. To build our intuition, let’s take a look at how that works first using a simple example.

Let’s look at a case where each state is represented by a box (Fig. 1). We’ll force an agent to start in the left most box (i.e., starting state), and to always move from the left to the right one box each time step. When it reaches the last box on the right (i.e., end state), the episode ends, and we’ll make the agent jump back to the starting state. Starting another episode. Finally, let’s add a reward at this end state, where the agent will receive some food. Ideally, we want our agent to not only learn that the end state is rewarding, but to also learn that states that reliably lead to this rewarding state are also valuable. Let’s see how the agent perceives each state when it uses the total rewards of the next state to update the estimated value of its current state.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Episode 1: the agent finds a rewarding state! It updates the expected value of that state | | | | | |
| Start 🡪 | 🡪 | 🡪 | 🡪 | 🡪 | End |
| Episode 2: the agent learns the preceding state leads to a valuable state! | | | | | |
| Start 🡪 | 🡪 | 🡪 | 🡪 | 🡪 | End |
| Episode 3: the agent learns the preceding state leads to a valuable state! | | | | | |
| Start 🡪 | 🡪 | 🡪 | 🡪 | 🡪 | End |
| Episode 4: the agent learns the preceding state leads to a valuable state! | | | | | |
| Start 🡪 | 🡪 | 🡪 | 🡪 | 🡪 | End |
| Episode 5: the agent learns the preceding state leads to a valuable state! | | | | | |
| Start 🡪 | 🡪 | 🡪 | 🡪 | 🡪 | End |

Figure 1: Diagram showing how updating the value of a state by using the total rewards of the next state can help an agent learn which states reliably lead to a rewarding state.

The idea to take away here is that that if an agent finds a rewarding state, and another state reliably leads to that rewarding state, the reward essentially spreads to those other states. Using this approach, the agent can learn behavioural sequences that lead to a rewarding state. In our very simple example, we forced the agent through states, from left to right, and so it’s not too surprising that the agent learns that each state reliably leads to the next state, and ultimately to the rewarding state at the end. From an associative learning perspective, the first rewarding state can be considered as an *unconditioned stimulus*, i.e., something that is inherently rewarding to the animal. While the subsequent rewarding states are *conditioned stimuli*, i.e., the animal learns that these states lead to rewards.

But how do we let those states that reliably lead to rewarding states capture those potential future rewards. Let’s draw out a simple MDP to help our intuition:

A blue circles with white letters on it

Description automatically generated

We’ll take a second here to go over each part of the diagram of the MDP above. First, there are two states: and . When in the agent always goes to (i.e., the probability of going from A to B = pA,B = 1.0), and when in the agent always goes to (i.e., pA,B = 1.0). We are again fixing the transition probabilities in this little made-up example, so the agent cannot take an action and stay in a particular state. When the agent transitions to it eats and receives a reward of +5. Let’s use this simple example to walk through how an agent learns that state leads to rewards, and what value it should hold.

Let’s use a gerbil as our example agent, set the expected value of each state to 0, and start the gerbil off in SA:

A mouse and a mouse with letters and symbols

Description automatically generated

To update value of the current state ( we could focus on using the observed reward at the next state . We can write that as

However, if we do this the expected rewards for will just keep increasing each time the gerbil reaches . E.g., if the gerbil receives a reward of +5 each time it reaches then will just increase by 5 each time: e.g., 5, 10, 15, 20. That does not make sense if the actual reward received from transitioning from to is 5.

Rather, we could use the difference between the current estimated value of and the reward it sees in the next time step:

This way when the observed reward in is 5, and the gerbil starts with w()=0 and w()=0, when it takes its first step going from to the agent will update:

So w() = 0 + 5 = 5.

Now that the gerbil is in , let’s move it back to .

A mouse and a mouse in a circle

Description automatically generated

When the gerbil goes from to it will update the value of by:

So now w() = 0 + 0.

A mouse and a mouse with letters and symbols

Description automatically generated

The next time step the gerbil repeats going from to it will update the value of by:

w() = 5 + 0 = 5. So, the values stabilize, and remain w() = 5 and w() = 0.

A mouse and a mouse in a circle

Description automatically generated

We can see here that the gerbil learns that state A leads to rewards, and learns that state B does not. We might be able to allow the agent to see a little farther into the future, i.e., state B can lead to state A and then lead back to state B, which brings a reward!! Seeing further into the future could help in more complicated environments where behavioural sequences that lead to a reward might be longer.

The trick to accomplish this is to include the expected value of the next time step in the total expected rewards using a discount factor (). When this discount factor is 1 the agent values rewards it might receive in the distant future as much as immediate rewards, i.e., it is looking far ahead. While a discount factor of 0 means the agent will only value the rewards in the next time step, i.e., just like in the example above with the gerbil. We will learn how to tune this parameter and how agents can inherit this parameter from their parents in chapter 7. For now, let’s see how our agent does with a discount factor of = 0.5.

To see how this works let’s start over with w()=0 and w(s2)=0, and the gerbil in SA.

A mouse and a mouse with letters and symbols

Description automatically generated

When the gerbil takes its first step going from to it will update the value of SA:

So, w() = 0 + 5 = 5.

A mouse and a mouse in a circle

Description automatically generated

When the gerbil then goes from to it will update w() by:

So now w() = 0 + 2.5 = 2.5.

A mouse and a mouse with letters and symbols

Description automatically generated

The next time step the gerbil repeats going from to the gerbil will update using:

So now w() = 5 + 1.25 = 6.25.

A mouse and a mouse in a circle

Description automatically generated

If we keep going for a few more iterations the values start to stabilize, and remain w() = 8.0 and w(sB) = 3. So now our gerbil learns that state B does have some future reward.

One final parameter that we need to learn is the learning rate (). That is how fast should the agent update its estimated rewards? This parameter can be though about as how quickly the agent adapts to reward signals, and can lead to very erratic behaviour when set too high (the agent changes its reward predictions with every change) or lead to slow learning when set too low. From the point of view of an animal, learning too slow or constantly switching behaviours is not ideal, and likely result in poor lifetime outcomes compared to other animals. We’ll see later on how this learning rate can be inherited through generations, and how it’s possible for animals to inherit contextual dependent learning rates that can help guide learning during their lifetime (when to learn fast vs slow) in chapter 8. But first to complete our value-based equation let’s add in the learning rate parameter:

You can try out different levels of learning rates (e.g., 0.1) with the examples above if you want to see how this changes the results.

What we’ve learnt here is called the *temporal differencing* approach to RL. Where the difference in prediction about rewards between time points is used to “bootstrap” an estimate of the expected reward of each state, i.e., the agent “pulls itself up with its bootstraps” by using its own experience to learn values of states. Given enough time, agents exploring an environment can learn these state values.

Next, we’ll see that the same temporal differencing approach can be used when estimating the value of actions within states.

* 1. Tutorial: Value based reinforcement learning

– Revisit the grid and mujoco examples from above

– How training is setup (SARSA?)

– How training is monitored (tensorboard)

* 1. Policy based (Rats pushing levers)

Now that we know how our agent learns what states are valuable, and how they learn which preceding states lead to rewards. Let’s use the same temporal differencing approach to see how agents can learn what actions they should take in each state. That is, in the example above we always forced the agent transitioned from state A to B and B to A. But what if the agent had more “agency” and could choose to move or to eat or to sleep?

To start off let’s allow the agent to track the value of an action in each state, let’s call it:

We can read the above as v for value, s for state, and the arrow pointing out that it’s the value of a behaviour B in state s.

Now we just need to allow the agent to update this value as it interacts with the environment. So, we’ll need something like:

As you might have guessed we are going to use the same update approach as we did with the value-based method above! That will look like:

In this case the value of performing behaviour B inside state A gets updated based on the observed reward in the next state , plus the discounted rewards of expected future rewards , minus the expected value of performing behaviour B in the current state A . As the agent interacts with the environment, choosing different actions and seeing what states they end up in, they will learn which state and action combinations lead to higher rewards. This combination of state-action values can then be thought of as a policy, that is, which state-actions the agent should perform to maximize rewards.

The agent can then use this policy to explore the environment. One way to use this policy is to use the current value associated with a state-action pair to determine which actions an agent will perform. So, when the agent starts out and all the = 0, then all actions are equally likely in that particular state. We can then choose one of these actions at random. However, as the agent interacts with the environment, if some of those actions lead to more rewarding states, these actions should be favoured. But by how much? If we always just choose the action that has the highest expected return of rewards our agent might just always choose the first action that led to any reward, and then never explore other options. So, to allow our agents to explore other options, and still exploit those that lead to rewards well use a weighting scheme. This scheme essentially compares the expected rewards of state-actions pairs and weights them based on relative expected return of rewards. This looks like:

This is just saying that the probability of a behaviours within a state is equal to the exponentiated expected return of that action , divided by the sum of all the other expected rewards from other actions . This allows the agent to explore different behaviours (i.e., not always choosing the top behaviour), and still exploit the behaviours that lead to rewards (i.e., more likely to choose the top behaviours). This approach is called a soft-max approach.

To provide some flexibility in how exploratory or how exploitative our agent is let’s add a parameter , that can be used to shift the agent to be explore more or exploit more:

We’ll see later in chapter 8 how agents can inherit this parameter from their parents, and how they can inherit the ability to shift between more exploration or more exploitation depending on the context.

It’s worth noting here that the agent is starting to track many values! If the agent can perform 5 behaviours and there are 5 discrete states the agent could be in, then there is a total number of 5\*5=25 state-action values to track. The number of possible state and action combinations can grow quite quickly. This presents a challenge for the agent to effectively explore all behaviors in all states to exploit the best combination of states and behaviours to receive the maximum rewards. We’ll see in the next section that we can use something called functional approximation to help the agent explore and exploit the environment when the number of states and behaviours are excessively large (i.e., when states and behaviours are continuous and cannot be assumed to be discrete).

It's also worth noting that the agent is learning a policy and applying what it’s learning right away. This is called online-learning. There are other approaches in RL in which the agent learns a policy but chooses actions in some other way (e.g., randomly choosing actions). When the agent learns a policy, but does not use that policy to decide which behaviour to use, it is called offline-learning. In our case, based on the class of problems we are looking to solve, where we are modeling the evolution and development of behaviours of animal-like agents we focus only on online-learning.

Table 3: Parameters that can contribute to agent learning

|  |  |  |
| --- | --- | --- |
| Parameter | Name | Description |
|  | Discount factor | How much the agent values future rewards [ref Julie’s work] |
|  | Learning rate | How quickly agents change their estimated values of states and action based on their experience |
|  | Explore/exploit actions | How agents trade-off exploring vs. exploiting potential actions |

* 1. Tutorial: Policy based reinforcement learning

– Revisit the grid and mujoco examples from above

– How training is setup (Reinforce?)

– How training is monitored (tensorboard)

* 1. Associative learning models: Actor-critic

Now that we understand how value-based and policy-based algorithms work let’s see how they might work together. The idea behind combining these two approaches comes from associative learning in animals. By including value-based algorithms, we can allow our agents to learn through classical conditioning, where the expected rewards of different states are learnt through experience. By including policy-based algorithms, we can allow our agents to also learn through instrumental conditioning, where the expected rewards of different actions are learnt through experience. Their combination can let agents learn both! Let’s take a look at the two approaches side by side:

We can see that their updates are very similar. We can also see that the learnt value of each state is used in updating the value of actions within states. That is the agent might learn that state B is rewarding, then learn that behaviour C in state A results in transitions to state B. So, the state-action policy directs the agent’s actions (i.e., called the actor) and the value of states determines whether those actions lead to rewarding’s states (i.e., called the critic).

Actor-critic algorithms are very effective at learning, allowing the agent to explore and exploit a range of environments. It will form the backbone for the rest of the book, and will act as a general learning mechanism that approximates associative learning in animals [ref a-learning girlanda]. Using this learning model, we’ll be able to test the limits of associative learning, and better understand how learning shapes behavioral outcomes.

Now that we know how actor-critic models work, let’s see how to use this model with our agents in Unity. The agent we built in Chapter 2, however, does not have a discrete behaviour called “walk” or “eat”. It simply applies force to its limbs, and learns to walk or to reach for food. The actions in this case are continuous, they are just more or less force applied to specific limbs/joints. How can we go from what we learnt above about value- and policy-based algorithms to getting an agent to learn a policy with continuous behaviours and states? The answer is to use something called functional approximation.

* 1. Tutorial: Actor-critic

-- revisit grid and mujoco with A2C

Chapter 4: Generalized learning from current states

*Learning in complex worlds with elaborate bodies in one lifetime*

To go from discrete behaviours, defined by us, to continuous behaviours, defined by the agent, we need to learn about functional approximation. We’ll also find out how to let our agent use more past information to help the agent learn. This section will help us go from the theory presented in chapter 3 to implementing this theory in practice.

* 1. Functional approximation

Functional approximation will solve two problems for us, and cause a new problem. The first problem it solves, is how to move from assumptions about discrete behaviours to continuous behaviours. We’ll also see that it helps the agent generalize it’s learning between similar states. This is powerful, as we saw the number of states can grow fast, and the ability to apply learning from ‘similar’ states is helpful when learning in complex environments. To solve these problems, however, we’ll have to solve the problem of building a model to approximate what state the animal is in, and what actions should be taken. This model building problem will mostly result in computational hurdles, but we’ll see as we go along that this model building aspect to functional approximation has some interesting conceptual implications when it comes to prepared learning in chapter 8.

So, if functional approximation is about building models, what exactly are we modeling? We will start with states. When using the actor-critic approach, we need to estimate the expected value of a state: i.e., . However, in cases where there is no discrete state (e.g., hungry/not hungry, and hot/not hot), and we have a continuous state (e.g., energy level 0.24, body temperature 21.34), we cannot assign a value to each state: e.g., ). So, to estimate the expected value of a continuous state we use a model. By using a model, we can estimate an expected reward for any combination of state variables. In other words, we want a model that can do:

Here, we have a model f(), that is estimating the value of a state f(w), given values for energy and temperature f(w|energy, temperature). The agent then must learn a model that take state variables (e.g., energy levels and temperature) as inputs and predict expected rewards. To build these models we will use neural networks.

* 1. Neural networks

While the name suggests that neural networks are closely linked to neurons in the brain, they are maybe more usefully thought about as a combination of edges and nodes that take some input (e.g., energy level, and temperature), and transform that into some useful outputs (e.g., the value of a state). Neural networks are also nicely visualized (Fig. 2).

A close up of a diagram

Description automatically generated

A neural network accomplishes this transformation from inputs to outputs using a series of simple weights and biases. These weights and biases are usually randomly chosen to start. These weights and biases then transform the input values (Fig. 2a) to update the value in the next layer of the network (Fig. 2b). Let’s see how this works between two nodes taken from the network in figure 2.

A blue and white circle with a black line

Description automatically generated

First thing lets provide random starting values for the weight and bias assigned to the edge connecting the two nodes:

A blue and white circle with a black line

Description automatically generated

Now if the input node received a temperature value of 23.1 we can use the edge to update the value contained in the hidden layer node. Let’s do that:

We can see that to transform the value from the input layer to the hidden layer we just used a simple linear equation. We can do the same for the second input node and add that value to the value we calculated coming from the first input node. Using this same linear equation, we can then calculate the new values in the nodes of the hidden layer, and continue this process until we reach the output layer of the neural network. Neural networks use this simple approach to transform input values into output values.

You might have noticed that all the equations are linear. This can limit the ability of a neural network to transform the inputs into non-linear patterns. So, another step is required. What we need is an *activation function*. There are many activation functions out there, but we’ll start out with a very simple but effective one, the rectified linear unit. Or RELU for short. This function simply ensures that if the value of any node calculated is negative (e.g., you sum up all the transformations from the previous nodes and it ends up being negative) it will automatically be given a value of 0. This introduces non-linearity to the neural network that can help transform the inputs into a wider range of useful outputs. You can think about this RELU function as setting a threshold for the nodes to be “active” and essentially informs the rest of the transformation process. If the node is “passive” it has little impact on subsequent transformations.

Going from the input layer to the last layer in the neural network is called a forward pass. But what we need to do now is to learn how to update the weights and biases in this neural network so that the transformations it performs to the inputs is useful, i.e., so that these transformations take the state variables (e.g., energy and temperature levels) and output an accurate estimate of the value of that state.

To do so we first need to figure out if the outputs of the model are good or bad, and if so by how much. We need a *loss function* to measure this. The idea behind a loss function is that it can measure the distance from the model outputs to something that we’d like it to output. This is supervised learning, where examples are provided that can help the neural network update the weights and biases in a way that hopefully makes its outputs closer to the desired outputs. Let’s look at this visually to get some intuition (Fig. 3):

A screenshot of a phone

Description automatically generated

Figure 3: diagram comparing the outputs of a neural network to those of the desired outputs.

From the point of view of our agent trying to learn weights and biases to best transform inputs about state to expected rewards, it now needs to compare it’s predicted rewards to observed rewards .

Now that we can feed our neural network inputs (energy level, temperature), it transforms those inputs into outputs using weights and biases, and we can estimate how different the outputs are from a desired output, we next need to update the weights and biases to better transform the input data. To do this we will use an *optimizer*:

A screenshot of a computer

Description automatically generated

While optimizers are for the most part outside the scope of this book, we’ll get introduced to how to choose optimizers, and loss functions, in the tutorials. Though for a more throughout introduction to neural networks see Francois Chollet (ref: deep learning in python).

What is important to know now, is that we can use the agents experience with the environment to learn a model that can approximate the value of the current state. We can apply the same approach to estimate the value of continuous actions within continuous states.

With the functional approximation approach, we can estimate the values of states and state-behaviour pairs and use these estimates in the actor-critic algorithm. Now that the initial concept of functional approximation is in place, let’s go into Unity and build the functional approximation of the actor-critic algorithm. This will give us a better idea of how functional approximation works in practice, and it will eventually allow us to give our agents sight!

Link to online tutorial book: chapter 3.1

### Link to tutorial on functional approximation of the actor-critic algo ###

Maybe this is where the mujoco tutorial can come in (i.e., continuous action spaces)

* 1. Short Term memory

In the actor critic method above we use the expected reward of the next state to update the value of states and actions. Why only the next state? Here we’ll learn how to add in short-term memory to allow our agents to use more than just the next state to learn the value of states and actions.

Expending the backwards view of updating.

The idea that TD learning above relies on discrete steps, but there aren’t really discrete steps… if the steps are too small then the differences in expected and current values become quite small. To avoid this, and allow the agent to use more continuous steps we’ll use eligibility traces. These traces keep track about what actions led to increased rewards, and forms a sort of short term memory. (Note: see missing pieces section about how memory here is not what memory is in the biological brain)

Eligibility traces

We will use eligibility traces to allow our agent to use recent updates (i.e., where there was a TD error) to have an impact on subsequent learning. The ability to use this extra information makes our agent a more efficient learner. We’ll also see that this approach can also lead to context dependent learning rates (when to use more/less short-term memory when learning), and how this prepared learning can be inherited through generations to help guide learning during the agent’s lifetime.

* 1. Tutorial: Eligibility traces (adding some short term memory)

Link to online tutorial book: chapter 3.2

### Link to tutorial on eligibility traces for the actor-critic algo ###

After this chapter we now have a general-purpose learning algorithm that is tailored to learn when states and actions are continuous, and agents only live once. We saw that functional approximation allows us to extend the actor-critic algorithm setup for continuous behaviours and states, and that eligibility traces incorporate short-term memory to take advantage of the online learning that an agent experiences during its limited lifetime.

In the next chapter we will add structures around this general-purpose learning algorithm that shape the inputs it receives. To start, we’ll give our agents the ability to see their environment!

Chapter 5: Providing the benefit of sight

*Deep Reinforcement Learning and vision:*

* 1. Providing sight

Sight is something we are good at… hard to get away from this bias.

(Ed Young) Many sensory abilities out there. Motivational examples: xxx.

How to use this sensory information with our sequential learning agent? One way to approach this is to add another neural network that transforms the images experienced by the agent into something useful for our agent. This something useful will be learnt by our agent, and gives us a way to avoid defining what is useful in the environment. Through interactions with its world the transformed images define the observations that the agent learns to use to choose actions that lead to rewards. Let’s draw this out to keep a high-level understanding of what we are trying to do (Fig).

### Figure here with the NN for the value function and the action selection, and now another one for vision###

To transform images experienced by the agent into something useful for the agent we need to gain some insights from what we know about neuron formation in the biological brain. Namely, we’ll need to learn about convolutional neural networks.

* 1. Convolutional neural networks

When we build our first neural networks above, we used what are called densely connected layers. These layers consist of layers of nodes where each individual node is connected with an edge to all other nodes in the next layer. These densely connected layers work well for transforming data through subsequent layers into values that are somehow useful to our agents. With images, however, we run into a problem. To build the first layer, which is just a 1-demensial line of nodes, whereas images are inherently 2- or even 3-dimensional (or more if you consider color and time!). An approach that can take advantage of the spatial, color, and temporal dimensions of images is convolutional layers. These convolutional layers are layers just like the dense layers just with a different structure. Instead of having 1d layers, we have layers that are 3d. We can think of them as cubes, where each layer of 2d can be an image: e.g., image in blue, green, and red. Instead of using edges to transform values in one node to another node as we did with dense layers, we use a moving window approach. This moving window centers on each pixel in an image layer, looks at the surrounding pixel values, and learns values that can convert the center pixel into a new value in the next convolutional layer. These leant values of a moving window are called a kernel and is analogues to the transformation done by the linear function in the dense layers.

### Figure showing convolution ###

These convolutions can reshape an initial image that is a wide shape (180x180 pixels with 3 color layers) into a long shape (16x16x64). In doing so each convolution tends to capture more and more higher-level features:

### Figure showing the higher-level transformations ### edges, corners 🡪 Eyes, mouth…

By tying this convolutional network, the idea is that the agent will learn transformations of the initial input images that will be useful when estimating the value of particular states and actions. For example, it might learn that green is associated with positive rewards when taking the eating actions. By using an approach that uses raw images as inputs, we avoid the potential biases of choosing what in the image the agent should pay attention to. And rather leave it up to the agents to figure out what is meaningful and what is not.

We’ll see in the tutorial that it is also possible to dissect these convolutional neural networks after our agents have trained to see what it is they are paying attention to. Though often useful there are limitations of the extent that we can explain what they have learnt, and will become a feature that will come back to again and again as we increase the complexity of our agents.

* 1. Benefits of generalization / costs of computation

We learnt in the functional approximation chapter that using a model that takes states in a continuous format and outputs a predicted state and action value can help generalize between similar states. I.e., if we are in a new state we’ve never seen before we can use similar states to help estimate what’s going to happen. A similar thing applies with the use of convolutional neural network layers. By using these layers, the idea is that the agent will be able to make predictions about values of states and actions when viewing the environment. The more the agent is exposed to the less likely the visual inputs will be drastically different than anything it has seen before. This suggests that the training history of the agent, and the complexity of the environment can play an important role in the development of behaviour of our agents.

* 1. Open questions

BE: Sufficiency of a sensory ability to complete a task. Using multiple sensory signals.

Cognitive movement ecology: E.g., home simply might look more like home and be more familiar to the agent, even without prominent features.

AI: network architectures (encoder-decoder approach for developing convolutional layers), linking visual NN with the actor-critic (recurrent layers?)

* 1. Tutorial: Adding a sense of sight (CNN)

Chapter 6: When is learning hard?

*How to think about behavioural complexity:*

* 1. Chains of behaviour (Thinking in terms of sequences)

Up to now we’ve built an agent that can interact with a complex body in a rich world using the sense sight, and can learn what features in the world are useful to pay attention when figuring out what states and actions lead to rewards. Key to this learning is that the agent has to spend time and energy interaction with its world. That is, it needs to generate experiences, then learn from that experience, altering its behaviour, and then generating new experiences. And so on and so on. The ratcheting up of learning here, and the sequential nature of this learning is a key feature that matches on with some of the challenges faced by the agent.

Often animals must learn tasks that have many steps. These steps often need to be done in order, where the longer the number of steps the harder it is to learn. We’ve already seen this challenge in Chapter 1 when talking about the environment: with the squirrels and pinenuts. If we remember, the squirrels had to learn that pinecones have food, then to remove the outer layer of the pinecone, then to eat the pinenuts inside. This chain while not comparatively as long as say building a microwave oven, still requires learning on the part of a juvenile squirrel and an explanation on our part. We saw that part of the answer must come from the environment, and the presence of other experienced squirrels. The results of which is that juvenile squirls are likely to have access to all the stages of the 3 part sequence to get pinenuts.

* 1. Favourable entry points

Thinking about chains of behavioural sequences the agent is required to learn will be our main way of computing the difficulty of the learning task. In particular we will be interested in the length of the chain, the ability to start the chain at any point, and whether the combination of the body and environment results in favourable entry points into the chain.

Longer chains are inherently more difficult to learn (Magnus book, figure of empirical evidence). Example of the range.

Longer chains where the only entry point is the first step, and are not able to start anywhere in the chain makes learning even more difficult. Example of one of these:

Favourable entry points can be due to the environment, body, and the interaction between the two. Examples of each of these: e.g., with human babies

* 1. Measuring behavioural chains

How to measure sequences of behaviours when behaviours are continuous… i.e., predictability / entropy measures... How do we know when an agent has learned a chain of behaviours (e.g., we can see that they are walking, by chaining together many actions to their body parts… but how do we measure the magnitude of chaining and see it grow/stall). We can use the action sequences, just like in the discrete case, and identify times where sequences are more predictable than others and for how long that is the case? For the simplest case, let’s say we have a fish in water, which pushes back and forth on a tail to move and is rewarded for moving forward. Initially, the actions sent to the tail are rather random. Through experience the fish slowly learns to push one way, then back the other, in long rhythmic motions. We could monitor the actions sent to the tail and get an erratic pattern that turns into a sin wave. There is a measure of sample entropy for sequences that might capture the change in predictability… but is it the best way to go… might be better options… emphasize this is not done yet, and will need work… maybe add it as a question section not its own section?…

* 1. Role of the environment
  2. Tutorial: foraging for pinecones

Chapter 7: Using evolution and development to shape learning

*Forget top-down designing; how to inherit and grow learning structures.*

|  |  |
| --- | --- |
|  | *“… next time you see a marvelous and complex behavior—such as a border collie herding sheep or birds flying south for the winter—try to resist the temptation to label it as instinctive, hardwired, genetic, or innate. By foregoing a label and digging deeper, you will open yourself to consideration of the myriad of factors that shape who we are and why we behave the way we do.”*   * ***Mark Blumberg*** *(Development evolving: The origins and meanings of instinct)* |

Now that we have developed the tools to create agents capable of adaptable behaviour, if we are to use these agents to better understand the development and evolution of complex behaviours, we need to think about how these agents can evolve. It might seem somewhat trivial to implement evolution, but it turns out there are some issues that arise with a simple view of evolution, and a quick look into evolutionary biology and artificial life might be useful.

Let’s start with genes and their relationship with a behaviour. The notion that a gene, or a particular set of genes, contain within them a particular behaviour is a common one. However, when a behaviour is looked into, and in particular when we watch the development of some behaviour, it quickly becomes obvious that genes themselves do not contain a behaviour, and that the experience of interacting with the world plays a large role. A good example of this is the idea of instincts, i.e., that animals are hardcoded to perform certain behaviours, and that these behaviours are relatively fixed and not learned. The idea of having learned and unlearned behaviours is not very useful (ref basic instinct). Rather there is room for interactions between the world and a developing agent (a debate called the nature vs nurture). It used to be thought that birds knew their songs from birth due to genetics. Recent studies looking at the development of bird song has changed much of that. Let’s take an example of ducks (species name…). When these ducks are isolated from other ducks they were found to produce species-specific calls. That sounds very much like genetically determined songs. However, when isolated ducks were prevented from singing while in embryo, they were not able to later sing their species-specific calls. Here the development of a species-specific call depends on the duck interacting with its world in embryo for the species-specific song to be learnt. Similarly, in cowbirds, who lay their eggs in nests of other bird species to get out of parental duties, their species typical songs were also thought to be instinct, i.e., genetically hardwired. How could these birds growing up in nests of other species, learn their song if not by genetics. However, when the behaviour is observed in more details, and experiments are run where the birds are X, it turns out again that interactions with others of their species are necessary to learn these songs. So, in both examples, what was once thought of as instinct, i.e., unlearned, genetically determined, really turns out to be a little more complicated. Therefore, we can’t simply encode behaviour in genes.

To think about how to inherit behaviour in our agents we must think more generally about how complex behaviours are inherited, i.e., beyond just genes.

* 1. Inheritance is more than just genes

Usually when talking about inheritance we automatically think of genes as the mechanisms. In evolution, however, selection operates on the whole organism manifold. What we mean by this is that reliable environmental features can allow for inheritance between generations similar to that of genetics. Considering these reliable environmental features and their interactions with genetics can be very important with thinking about (and modeling) how behaviours are inherited from generation to generation. In the duck example, the developing embryo has a reliable and stable environment in which to develop. Where genetics plays a role in production of initial sounds that are then fed back upon to allow a developing duck to learn a species-specific call.

As we build our simulation agent-environment systems, we need to pay attention not only to inherited structures like DNA/genetics but also to predictable and reliable features of the environments in which individuals are developing. And crucially how that might vary between individuals we are simulating.

* 1. A history of focusing on learning over inheritance in AI

It is often assumed that when building AI agents using neural networks, that the initial weights of a network are random. When we look at the development of the brains, we find anything but random (ref: self-assembling brain). Let’s take a look at a relatively naive way to incorporate evolution in our simulations, then use that as a springboard to dive deeper into how evolution shapes organism behaviour, and how we can build these insights into our simulations.

* 1. Genes as weights

To incorporate both genetics and reliable environmental features we will take the approach of having genes determine the initial weights of our neural networks. These weights start out as random, and in each generation there is some mutation in these initial weights. We can think of these changes are modifying initial reflexes of our agents. All these weights are then subject to updates using associative learning when interacting with the environment. This approach allows for initial weights to produce reliable initial behaviours that when performed within an environment, can produce feedbacks between agent and environment, in a way that can help guide learning. This way we are not specifying genes to specific behaviours, and we are not unrealistically, allowing individuals to take what they learnt during their lifetime and passing that onto the next generation.

* 1. The problem of low evolvability

What is missing in this account? Well for one, when compared to how evolution is understood to take place, it is somewhat limited in its ability to evolve a wide variety of brains. The magnitude of flexibility of an artificially generated evolutionary program is known as its evolvability. Using weights of a neural network as inherited genes limits the range of brains that could be produced. This can somewhat be ameliorated by allowing the number of weights, hyperparameters, and the architecture of the neural network to be inherited as well. To appreciate how this approach is still somewhat limited we need to dive a little deeper into evolutionary biology and artificial life.

To go beyond this limited view of inherited structures, we might need a different metaphor for our DNA. Rather than genes in our DNA being for something, they can be seen as resources for a cell to do something (ref: how life works, Philip Ball). Evolutionary biology is primarily interested in the frequency and distribution of genes in populations, while developmental genetics is more interested in the idea of genes as resources for the growth and development of organisms. The resource metaphor allows us to focus more on the role of self-organisation in the development of behaviour. Where inheritance and experience are a two-way street, with environment impacting how cells use their inherited DNA. Building agents that inherit structure used in the self-organized development of agents is a promising avenue for allowing greater flexibility in the contextual development of behaviour, potentially allowing evolution to prepared development under a wider range of potential contexts. However, this avenue is still in its infancy, and we will not cover these methods in this book (ref: neurological developmental programs). Rather we will rely on defining initial weights and architectures for our neural networks, with these limitations in mind.

A fact here might be useful to keep in mind so that we appreciate what we are excluding here when taking this track, and to potentially motivate future work. Much of the human DNA looks to be regulatory, and is not genes that produce a specific protein. In most bacteria greater than 90% of DNA is composed protein producing genes, for nematodes it’s ~20%, while for humans it is less than 2%. Suggesting that much of the resources provided in the human genome that allows for more complex behaviour is the result of regulation of what DNA does give a specific context (ref Ball). This in no way suggests that humans are somehow “more evolved” than bacteria or nematodes, just that eventually we are likely to need to build this contextual resource into the inheritance models used for our agents if we wish to allow for complex contextual dependence in our agents’ behaviors.

* 1. For rewards, is complexity the answer?

We need to touch on one more thing while talking about evolution and inheritance. That is that in RL, choosing the reward function to use is deceptively tricky. Often in simple maze cases a reward is given for reaching the end of the maze, or getting a piece of food. But the simplicity of the reward can often lead to strange behaviour. For example, while building the worm tutorial you might give the worm a positive reward for getting to the food, and to try to incentivise the worm to get the food faster you might give the worm a small negative reward for every second that the worm has not yet reached the food. However, even with the relative simplicity of the number of actions to move the worm body, and the challenge of moving them in a way to find food, it might be difficult to for the worm to find the absolute best behaviour. Rather it might find some behavior that leads to pretty good results, that maybe you were not expecting. In this case the worm might learn to jump out of the arena… wrom-suicide!! It found a local optimum, in terms of maximizing its reward, if it jumped to its death it wasn’t alive long enough to accrue negative rewards. In this somewhat simple case, it can be easy to see, but not so for some of the examples we’d like to try in this book. Evolution here offers us an opportunity to shape a reward function that might be complex enough, or robust enough, to ensure that an agent won’t just kill itself at the first possible moment. Rather we can build between generation prepared learning again by allowing successful agents to pass down their reward functions to their offspring with some mutation.

We must note here too that the reward function only specifies the unconditioned stimulus, and that the agents themselves, depending on their environments, can find all sorts of conditions stimuli that might bring them to the unconditioned stimulus. So, agents might show drastically different behaviours to meet the same unconditioned stimulus.

* 1. Tutorial: learning fast vs slow

-This tutorial should show how to build in evolution (multiple generations, with selection, with inherited resources of the initial weights)

-It should also show that through generations agents can be prepared to learn (i.e., the population gets adapted to the environment, and shows some kind of improvement in terms of “optimality”.

Chapter 8: Prepared learning

*Adding in contextual dependencies*

|  |  |
| --- | --- |
|  | “*A moth will never know what a zebra finch hears in its song, a zebra finch will never feel the electric buzz of a black ghost knifefish, a knifefish will never see through the eyes of a mantis shrimp, a mantis shrimp will never smell the way a dog can, and a dog will never understand what it is to be a bat. We will never fully do any of these things either, but we are the only animal that can try*.”   * **Ed Yong**  (An immense world) |

In the last chapter we discussed how we might build in inheritance into our simulations to allow selection pressures to evolve our population of our agents. But what kinds of inheritance structures can our agents inherit? We’ll see in this chapter we can allow the A-learning algorithm to use context and prepared learning to create agents that can be well prepared to learn in their specific environments.

* 1. Role of evolution in prepared learning: i.e., no blank slate here!

Conceptually, approaches that focused on using associative learning to understand animal behaviour have often been criticized as blank slate models, i.e., where an animal only learns from experience during its lifetime and learning is not influenced by genetics. Equating associative learning with blank slate learning, is largely a misunderstanding of associative learning, as there are lots of avenues for prepared learning to play a role (ref enquist and lind). The blank-slate approach is largely true, however, in DRL, where an agent generally uses random initial values for their neural network.

In this chapter we’ll take a look at how we can allow evolution to prepare our agent to learn within their environment. Specifically, we’ll look at:

* Inheriting a body
* Inheriting context dependent behaviours
* Inheriting context dependent learning rates
* Inheriting perception
* Inheriting context dependent rewards
* Inheriting context dependent short-term memory
* Inheriting latent states
* Interactions between inheritance structures (i.e., between perception and motor?)

Each of these routes for inheritance to prepare learning correspond to specific parts of the A-learning algorithm. We’ll start with the body.

* 1. Inheriting a body

Example of body as cognition (cricket example again? And another?), Lou’s book (brain without a body?!), embodied cognition, RL paper on how body can speed up learning, then how we can add this in our models.

* Optional tutorial (prepared learning through the body)
  1. Inheriting context dependent behaviours

(emotions: anger, sexual, maternal, etc) Examples in animal behaviour about the link between emotional states and pre-disposed behaviours. RL example paper. How we can build a separate network that will influence the exploratory vs explanatory behavioural parameter in A-learning (math (standardized to zero + some sd, adjustments on a mean) and how this neural network can result in prepared learning and predisposition to explore/exploit). [Unsure: how can we promote complex actions (e.g., multiple continuous actions result in the behaviour) that are not at a higher level in the network… second network that directly influences those continuous actions spaces based on context, i.e., no learning influence here… ].

* Optional tutorial (prepared learning predisposed behaviours and/or exploratory/exploitative behaviours)
  1. Inheriting context dependent learning rates

When should you learn and when should you ignore. Examples of context dependent learning rates in animals and humans. DRL hyperparameter selection sensitivity analysis/lit review. How we can have a third network that uses contextual states to adjust from a mean learning rate in the A-learning algorithm.

* Optional tutorial (prepared learning fast vs slow rates)
  1. Inheriting perception

Umwelt and how we “see” the world. Human bias towards vision. Primates (Amanda’s work on trichromat and dichromate vision and frugivores/folivores). What we see can guide our learning. How perceptual networks can be inherited (convolutional neural networks vs vision transformers). How we can inherit the first layers and allow learning to adjust the final few layers (fine tuning in DRL).

* Optional tutorial (prepared learning through vision)
  1. Inheriting contextual dependent rewards

We saw in the last chapter that evolution might be an important method to avoid coding reward functions manually. Rather we expect reward functions to be contextual (reward for food *when* hungry), and that rewards can be complex. Animal example of non-simplicity of rewards? Selection pressures will promote the replication of agents that reproduce more, and reinforcement is likely a very effective method to guide agents towards reproducing in noisy dynamic environments.

* Optional tutorial (prepared learning through contextual rewards)
  1. Inheriting working and reference memory

There is often a distinction between working memory, that which we use at the moment, and reference memory, that which we can recall from previous sensory inputs and states. Memory is still quite difficult to define when it comes to how it works in the brain at the neurological level. In animal behaviour studies, however, it has been useful to separate working and reference memory when studying and explaining observed behaviour. In the A-learning algorithm with eligibility traces we can setup another network to use context to adjust how much working memory to use when learning. By increasing eligibility traces we can increase the amount of working memory, and allow rewards to be associated with sequences further back. By adjusting this working memory the agent can be prepared to learn longer and shorter sequences of behaviours depending on the context. Animal example? Then how we can do this in terms of math (again NN with mean zero adjusting some inherited mean eligibility trace decay parameter).

* Optional tutorial (prepared learning of sequences: working/reference memory)
  1. Inheriting latent states

As part of the observations, an agent has access to many state variables: e.g., hunger, vision, temperature, etc. Many of these variables are continuous, dynamic, and noisy. It is possible that an inherited structure might recognize a lower number of latent states. We can incorporate an encoder-decoder neural network within the A-learning algorithm to learn to embed observations into a lower-level state. The encoder part of the network can feedforward these latent states to help define values and actions to take on the next step. The hyperparameters to the encoder-decoder can guide learning of the agent.

* 1. Interactions between inherited structures

Given the many ways that inheritance can prepare our agents to learn through the A-learning algorithm it begs the question, how might they interact. So far, we’ve looked at using separate networks to connect contextual states (which ones?) to adjust some parameters or weights in the A-learning algorithm. It is possible to add residual connections between these contextual networks, however, it is currently not known if this would improve an agents ability to learn complex behaviours in noisy and dynamic environments. Or, if the models can even find optima that are close to optimal or at least lead to better reproduction relative to agents without interactions.

To help keep track of what inheritance structures are being applied and how they might be interconnected we present a diagrammatic approach. The first diagram outlines where in the traditional diagram of RL these inheritance structures fit (Fig. X). While the second uses a specific notation to represent the interconnections between inheritance structures within the A-learning algorithm.

* Optional tutorial (prepared learning through residual connections, relative performance?)
  1. Tutorial: prepared learning

-This tutorial should show how evolution of some prepared learning structure (contextual learning rate?)

-It should also show that through generations agents can perform better than the non-contextual learning rate (from tutorial based on evolution chapter).

-this might be better for each of the sections

Chapter 9: Social learning

*Multi agent DRL*

* 1. Social learning in BE and animal cognition

– Social niche construction, embodied and embedded cognition (4E)

* 1. Different mechanistic explanations

– social brain hypothesis, cognitive buffer hypothesis, elaborate sensory-motor hypothesis

* 1. Imitation

– theory of vertical associations (links between perceptual and motor states)

* 1. MADRL

– multi-agent deep reinforcement learning

* 1. Tutorial: competing/collaborating animal troops

– multilevel selection