**A logo of a university

Description automatically generated**

***MA4270 Data Modelling and Computation***

***Markov Decision Process in Reinforcement Learning***

***By:***

***Woo Kean Jin Brandon, A0233835A***

Introduction

Reinforcement Learning stands out as a potent machine learning approach in the realm of artificial intelligence, enabling machines to learn from their surroundings. Unlike supervised learning, it doesn't rely on labelled data; instead, it utilizes rewards and punishments to guide the model toward achieving optimal outcomes. To illustrate, consider a child warned by parents about the dangers of touching a hot stove. While some kids grasp the concept through verbal warnings, others need the first-hand experience of getting burnt to associate the hot stove with pain—a parallel to the punishment aspect in reinforcement learning. Conversely, the pleasure of tasting sugary or oily foods triggers dopamine release in the brain, acting as a reward and encouraging the pursuit of high-caloric sources. Despite the imperfections in the human reward and punishment system, we can control and influence the model's behaviour.

Markov Chain (MC)

Before delving into the intricacies of Markov Decision Processes, it's beneficial to understand the fundamental concept of a Markov Chain (MC). A Markov Chain comprises two key components: the state space, which encompasses all conceivable states within the system, and the transition probability, dictating the likelihood of transitioning from one state to another over a unit time interval. The key attribute of a Markov Chain lies in its memoryless nature. This property asserts that the probability distribution of future states is dependent solely upon the current state, rendering past states irrelevant for predicting future transitions.

0.5

0.4

0.8

0.5

0.6

0.2

Example of a MC with three states (green circle) and the transitional probability (blue arrows)

In this report, we will operate under three fundamental assumptions regarding Markov Chains (MC):

1. Discrete Time: We assume that time progresses in discrete steps, rather than continuously.
2. Finite State Space: The set of possible states within the system is finite, allowing for a well-defined and manageable exploration of state transitions.
3. Stationary Transition Probability: The transition probabilities between states remain constant over time, meaning that the probability distribution governing state transitions remains unchanged throughout the process.

Markov Decision Process (MDP)

The key distinctions between a Markov Decision Process (MDP) and a Markov Chain (MC) lie in how they handle decision-making. In an MDP, unlike an MC, the agent has the autonomy to choose actions at each state. For instance, an agent at state S1 can choose between action a1 or a2. This flexibility allows for dynamic decision-making tailored to the current state of the environment.

Unlike a MC where the transitional probability is from a state to another state, the transitional probability in MDP is from an action to a state, i.e. choosing action a1 can lead to few possible states depending on the transitional probability. By incorporating rewards into an MDP framework, we can explore the concept of optimizing the agent's actions to maximize its cumulative reward under given conditions.

+1

1

0.5

0.3

0.8

0.2

0.5

1

0.7

1

Example from a MDP with three states (green circle), 2 actions (orange circles), 2 rewards (orange arrow).

-2

Reward and Policy

The primary goal of the agent is to optimize its decision-making process to maximize the expected return, which is achieved by summing the expected returns at each time interval. The expected return at time t, denoted as Rt, is weighted by an exponentially decreasing factor known as the 'discount factor'.

The discount factor plays a crucial role in guiding the agent's decision-making process, balancing between immediate rewards and long-term gains. A discount factor of 1 signifies a forward-looking, long-term strategy, where the agent values future rewards as much as immediate ones. Conversely, a factor of 0 emphasizes immediate rewards, indicating a myopic approach. Factors between 0 and 1 represent a blend of short-term gratification and long-term planning. The optimal expected reward has an upper bounded of where .

Incorporating a discount factor into the algorithm enhances its ability to mirror real-world scenarios accurately. Consider a starving animal: food available now holds significantly greater value than food obtainable in the distant future. Conversely, for an investment banker managing client funds, short-term market fluctuations hold less weight than long-term returns. However, the expected return is an infinite sequence, leading to infinitely large optimal reward when . There are several solutions to combat:

1. Incorporating termination states, where the agent is forced to conclude its ongoing episode. (episode is the current run the agent is going through).
2. Introducing an upper limit on the number of steps allowed before episode termination.
3. Utilizing a discount factor less than 1 to attenuate the influence of distant rewards, mitigating the issue of unbounded optimal rewards.

Determining the appropriate reward allocation under specific conditions is pivotal in constructing a proficient algorithm. Consider a bot undergoing training to play chess: overly prioritizing the capture of enemy pieces might result in a chess bot fixated solely on capturing opponent pieces, disregarding the goal of achieving checkmate. Thus, striking the optimal balance in reward distribution, tailored to desired outcomes, requires careful parameter tuning. This process ensures that the algorithm aligns with the overarching objective, whether it's winning the game or executing strategic manoeuvres leading to victory.

The policy is essentially a rule book that instructs the agent on which action to take given its current state. In reinforcement learning, the policy is tuned iteratively to get the optimal set of actions to take.

Bellman Equation

So far, we've explored the Markov property and its significance in reinforcement learning, particularly in the context of Markov Decision Processes (MDPs), where rewards are integrated into state transitions. We've also discussed the main objective of an agent to maximize its cumulative reward and the methodologies for computing such rewards. In the following section, we'll delve into the approach used to evaluate the value of a state. As previously mentioned, reinforcement learning algorithms strive to optimize the expected reward. Consider the scenario illustrated below: a lion, seeking food, encounters a tempting deer within its vicinity. However, lurking nearby is a poacher, poised to strike if the lion ventures into their area. The lion's movement is restricted to four directions: up, down, left, and right, with each movement governed by a specific probability distribution, as depicted in the accompanying diagram. In this scenario, how can the lion effectively enhance its chances of capturing the deer while simultaneously evading the threat posed by the poacher?

80%

Desired direction

Lion’s movement probability distribution

10%

10%

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Elk on a white background |
|  |  |  | Hunter | Human - Recreation |
| Side view of a lion's head |  |  |  |

Firstly, let's designate the grids occupied by the deer and the poacher as termination states. In this setup, we'll assign a reward of +1 for reaching the deer and a punishment of -1 for encountering the poacher.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | +1 |
|  |  |  | -1 |
|  |  |  |  |

The value function gives the expected utility at and following policy . As discussed previously, the expected utility at is the expected sum of future rewards weighted by the discount factor . Let’s denote the optimal value function to be by following optimal policy .

The action-value function slightly differs from the value function by having an action input at and then subsequently following policy .

If we assume that the policy from onwards is optimal can be determined by finding the action that maximizes .

Due to Markov property,

by substituting into , rewriting and incorporating the reward function ,

It is also clear to see that with optimal a is equal to , giving us the bellman equation,

Or alternatively,

Value Iteration

The Bellman equation's usefulness lies in its self-contained nature, wherein the optimal value function encompasses itself within the equation, facilitates an iterative solution approach. Initially assigned an expected value of 0 to all unknown states, then systematically updating the value of each state using the Bellman equation until convergence is achieved.

To prove its convergence, let’s measure the difference between 2 value functions using the -norm, i.e.

Define the value iteration as recursively applying Bellman optimality operator ,

The Bellman optimality is a contraction mapping which in layman terms, it brings points closer to each other, i.e. there exists such that

Another point to note is that is a fixed point on T

By the contraction mapping of T

Applying recessively until gives

As , and thus the iteration converges.

Let’s use the grid example to illustrate the method.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 0 | 0 | +1 |
| 0 | 0 | 0 | -1 |
| 0 | 0 | 0 | 0 |

To initiate the calculation of the first value iteration for the blue grid, let's simplify the scenario by setting the reward function to 0 and the discount rate to 0.9. Specifically, let's focus on determining the action-value function for moving upward.

Repeating the calculations for the other 3 directions yield

It can be calculated to show that the value function is 0 for the rest of the cells. Let’s run it 20 times to see the result.

|  |  |  |  |
| --- | --- | --- | --- |
| 0.64 | 0.73 | 0.85 | +1 |
| 0.57 | 0.63 | 0.58 | -1 |
| 0.5 | 0.54 | 0.49 | 0.29 |

As anticipated, cells closer to the -1 cell and farther away from the +1 cell tend to have lower expected values. While we now have the expected value for each state, the crucial question remains: How does the agent discern which action to undertake?

The policy function, dictates how the agent interact with the environment which in the example shown is the direction to take.

For the lion starting position (1,3),

Repeating this for the other 3 directions yield {0.56, 0.504, 0.539, 0.507} and we can see that moving north returns the agent the max expected reward. Iterating throughout the rest of the grid gives us the policy function that can be represented as a grid of the same shape.

|  |  |  |  |
| --- | --- | --- | --- |
| East | East | East | +1 |
| North | North | North | -1 |
| North | North | North | West |

Value Iteration Performance

In this section, we are going to explore the agent’s performance by running simulations and recording the average result. Firstly, to visually see the rate of convergence of the bellman equation. We are going to use the to measure the improvement between iterations.

A graph with a red line

Description automatically generated

Given the small grid, the bellman update algorithm converges exponentially fast and there is no visually detectable improvement after 13 iterations. There is wasted computation effort for the extra iterations and the optimal policy is already obtained at an earlier iteration. As such it is useful to set a maximum iteration or set an improvement threshold. How would the speed of convergence be affected by the size of the grid? The starting point as previously at and the two termination states at and for a grid.

A graph of a number of rows

Description automatically generated

The number of iterations with respect to the exponential increase in state space only increases linearly but the number of computations per iteration grows linearly with number of cells. To model a realistic grid, let’s include a constant punishment of for every action that can be interpreted as energy utilized by the lion to move about. The reward for catching the deer is and the punishment for being caught by the hunter is . After running 1000 simulations, the lion finds the deer 987 times and caught by the hunter 13 times. The lion performs an average of 17.4 actions before reaching a termination state. If we were compared it to the minimum possible number of actions, the fewest number of steps is 15. The discrepancies between the result and the theoretical most efficient path are due to the probabilistic nature of the actions. What is interesting is if we were to increase the punishment for the lion’s movement,

Running 1000 Simulations

A graph with a blue line

Description automatically generatedA graph with blue lines

Description automatically generated

Two prominent spikes are evident in the graph at points and . The larger phenomenon arises because the anticipated reward for encountering the hunter exceeds that for finding the deer, leading the agent to prioritize ending the episode in the shortest steps possible. However, the secondary, smaller spike poses a less intuitive problem. Examining the policy grid with a focus on cells neighbouring termination states may shed light on this phenomenon.

|  |  |  |  |
| --- | --- | --- | --- |
| East | East | East | +1 |
| East | East | North | -1 |
| North | North | North | North |

|  |  |  |  |
| --- | --- | --- | --- |
| East | East | East | +1 |
| East | North | North | -1 |
| North | North | North | North |

The difference between the 2 policy grids is the purple cell where the policy changed from north to east. As such, the frequency of the agent reaching the -1 state increases and intuitively, we should expect to see more jumps when cells in the second row of policy switches from north to east. However, the jump in frequency gets increasing smaller the further away the switched cell is from the termination states and as such the jumps are indistinguishable from the random fluctuations of the graph.

Previously, the value of discount factor was discussed but the ramification of the absence of one was not. If the discount factor was 0, the policy grid for the original grid example is

|  |  |  |  |
| --- | --- | --- | --- |
| -0.1 | -0.1 | -0.1 | +1 |
| -0.1 | -0.1 | -0.1 | -10 |
| -0.1 | -0.1 | -0.1 | -0.1 |

With the value of every state being the same, there is no meaningful policy that the agent can follow to reach the desired termination state. Depending on the algorithm implemented, the policy grid could contain cycles where the agent is stuck in a loop even when the reward function is set to be largely negative. A positively small discount factor also leads to this homogeneity of the states’ values and as a result, a suitable discount factor is required to prevent cycles in the policy.

Closing Remarks

Hopefully, the basic explanation of reinforcement learning with a known action probability distribution is simple to understand. Reinforcement learning can be further extended of other environments such as continuous time, unknown action probability distribution and reward function. Additionally, most use cases like creating a bot to play Mario or chess have too many or sometimes infinitely many states to explicitly record. Moreover, in many practical scenarios, such as developing a bot to play games like Mario or chess, the sheer number of states—sometimes even infinite—renders traditional methods like value iterations computationally impractical for achieving convergence. Deep reinforcement learning integrates reinforcement learning with deep neural networks, enabling the capture of intricate patterns across layers of nodes. There are many extensions of reinforcement learning that goes deeper into the rabbit hole and provide us with many use cases.

Reference List

Gangwani, Tanmay, et al. *Lecture 16: Value Iteration, Policy Iteration and Policy Gradient*. 17 Sept. 2019. <https://yuanz.web.illinois.edu/teaching/IE498fa19/lec_16.pdf>

Grosse, Roger, et al. *CSC 311: Introduction to Machine Learning Lecture 11 -Reinforcement Learning*. 2020. <https://www.cs.toronto.edu/~rgrosse/courses/csc311_f20/slides/lec11.pdf>

Han, Xintian. *A Mathematical Introduction to Reinforcement Learning*. <https://cims.nyu.edu/~donev/Teaching/WrittenOral/Projects/XintianHan-WrittenAndOral.pdf>

Sutton, H. “Peter Morgan Sutton.” *BMJ*, vol. 348, no. mar31 11, 31 Mar. 2014, pp. g2466–g2466, web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf, https://doi.org/10.1136/bmj.g2466. <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>