Reinforcement Learning

Lidor Erez

June 2024

1 Preface

This paper represents my attempt to learn reinforcement learning as I am preparing for my M.Sc in Data Science which will focus on reinforcement learning as its main topic. All of the information given in this paper is taken from [1] and [2].

2 Introduction

Reinforcement learning is a method of learning that enables an agent to determine the optimal actions to take in various situations to maximize its cumulative reward. Unlike other forms of machine learning, where the correct actions are typically provided, reinforcement learning requires the agent to explore and experiment with different actions to discover which ones yield the highest rewards. This trial-and-error approach means that the agent's decisions not only influence immediate rewards but also impact future situations and their associated rewards. Besides the agent and the environment it interacts with, there are four main sub-elements of the reinforcement learning system. The first is a policy, which is a function that takes the state of a given environment and returns the action that needs to be taken when the agent is in that state. The second is a reward function, which defines the amount of immediate reward the agent receives for a given state-action pair. The objective of the reinforcement learning agent is to maximize these rewards. The third is a value function, which defines what is good in the long run. Specifically, the value of a state is the total amount of rewards an agent can expect to accumulate over the future. The fourth, though optional, is a model of the environment. This model allows the agent to simulate and predict the outcome of actions, thereby aiding in planning and decision-making. The goal in reinforcement learning is to balance between the exploration which is the way of an agent to take some actions that might not yield the highest reward and exploitation which is the way of an agent to maximize it's total reward. This balance is crucial because an agent needs to explore sufficiently to gather information and refine its understanding of the environment (which can be uncertain initially) while also exploiting its current knowledge to achieve the highest possible rewards.

3 Background

Reinforcement learning is built upon Markov Decision Process (MDPs), which provide mathematical model for decision-making in environments where outcomes are partly random and partly under the control of the agent. We can define an MDP by the tuple (S, A, P, R, γ):

- S is the set of states an agent can be in.
- A is the set of actions an agent can take.
- $P(s'|s, a)$ is the state transitioning. Namely, the probability of transitioning to state s' given the previous state s and an action a that has been taken by our agent in s.
- $R(s, a)$ is the reward function, which gives the immediate reward for every state-action tuple.
- $\gamma \in [0, 1]$ is the discount factor which scales the importance of future rewards.

3.1 Polices

As mentioned above, a policy $\pi(a|s)$ is a mapping function that maps a given state to the action that needs to be taken. Alternatively, a policy can be stochastic, returning a distribution over actions for each state, indicating the likelihood of each action.

3.2 State - Value Function

The state-value function $V^\pi(s)$ defines what is beneficial in the long run given a state and providing the agent with the expected accumulated reward over the future regardless the action he decided to take. It can be defined as follows:

$$V^\pi(s) = E_\pi[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s]$$

where s_0 is the current state, and (s_t, a_t) are the state-action tuple in time t. From what I understand, we raise γ by the power of t to ensure immediate and near-term rewards are more beneficial, and being more focused by our agent.

3.3 Action - Value Function

The action-value function $Q^\pi(s, a)$ also defines what is beneficial in the long run. However, it provides the agent with the expected accumulated reward over the future regarding the action he decided to take in a given state. It can be defined as follows:

$$Q^{\pi}(s,a) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s_0 = s, a_0 = a]$$

where a_0 is the current action the agent decided to take in the current state s_0.

3.4 Action - Value Methods

Action-Value methods are used to estimate the values of actions which are crucial for selecting actions in a given state. The true value of action, which is typically unknown, is denoted as $Q^*(a)$. The estimated value of an action at time t is denoted as $Q_t(a)$. We can estimate the true value of an action using various of methods which will be defined in the next sub-sections.

3.4.1 Averaging Rewards

The averaging rewards method estimates the value of an action a by averaging the rewards the agent acquired when that action has been taken. Namely, if action a has been taken k_a times up to time t, the estimated reward value is calculated as:

$$Q_t(a) = \frac{r_1 + r_2 + r_3 + ... + r_{k_a}}{k_a}$$

where r_i is the reward obtained for a at time i. This method gives us the true value of an action by applying the law of large numbers:

$$\lim_{k_a \to \infty} Q_t(a) = Q^*(a)$$

When $k_a = 0$ then we use a default value of $Q_t(a)$ to be the estimated initial value. The problem with averaging rewards that it's memory and computational requirement can grow infinitely. To solve this problem, we can compute averages of the rewards acquired by our agent using incremental implementation.

Stationary Incremental: Let Q_k be the average of the first k rewards acquired then given this average and (k+1)st reward r_{k+1}, we can compute Q_{k+1} by the following formula:

$$Q_{k+1} = Q_k + \frac{1}{k+1}[r_{k+1} - Q_k]$$

where $\frac{1}{k+1}$ is the step-size parameter that changes from time step to time step. The step-size parameter is denoted as α or $\alpha_k(a)$. For example, the step-size parameter in the averaging rewards method is $\alpha_k(a) = \frac{1}{k_a}$.

Non-stationary Incremental: One way to create a non-stationary incremental is by using $0 \leq \alpha \leq 1$ as the step-size parameter in the stationary incremental formula and show that:

$$Q_k = Q_{k-1} + \alpha[r_k - Q_{k-1}] = (1-\alpha)^k Q_0 + \sum_{i=1}^{k} \alpha(1-\alpha)^{k-i} r_i$$

where Q_0 is the initial estimate. For non-stationary incremental methods, the choice of the step-size parameter α or more generally $\alpha_k(a)$ must satisfy the following conditions:

- $\sum_{k=1}^{\infty} \alpha_k(a) \to \infty$
- $\sum_{k=1}^{\infty} \alpha_k^2(a) < \infty$

These conditions ensure that the step-size adapts appropriately over time, allowing the learning process to balance between exploring new information and exploiting existing knowledge effectively.

4 Selection Methods

We defined our way to estimate the rewards r for a given action a but how can we decide which action to take? One way of selecting an action is using the greedy method, which simply takes the action that gives the highest reward in a given state. However, this approach may not balance between exploration and exploitation. By always choosing the action with the highest current reward, we risk missing out on exploring other actions that could potentially yield higher rewards in the long term. This limitation can lead to sub-optimal performance when the environment is dynamic or uncertain.

The second way of selecting an action is called $\epsilon - \textit{greedy}$. The $\epsilon - \textit{greedy}$ greedy approach selects the action that currently appears to have the highest reward most of the time (exploitation), but occasionally (with probability of ϵ), it chooses a random action (exploration). This method allows us to balance between exploring different actions and exploiting the best-known actions for maximizing rewards over time. However, a challenge with the $\epsilon - \textit{greedy}$ is that it assigns equal probability to each action when choosing randomly. This means that the agent has the same probability of selecting actions that have been consistently rewarding (best-case actions) as it does of choosing actions that have historically led to poor outcomes (worst-case actions). This uniform random exploration can be inefficient and may not effectively prioritize actions that are more likely to lead to higher rewards.

The third method, which addresses the uniform exploration issue of ϵ-*greedy*, is called Softmax Action Selection. This method utilizes the softmax function to create a probability distribution of actions at time t, thereby varying the

probabilities of selecting each action. The formula for Softmax Action Selection is:

$$\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau}}$$

where τ is the temperature parameter. When τ is high, the probabilities for selecting each action become nearly equal. As τ decreases, the differences in probabilities between actions become more pronounced. When $\tau \to 0$, the Softmax method approaches the Greedy method, where the action with the highest estimated reward $Q_t(a)$ is most likely to be selected. Softmax Action Selection thus allows for controlled exploration and exploitation: higher values of τ encourage exploration by giving more uniform probabilities across actions, while lower values emphasize exploitation by favoring actions with higher estimated rewards. This flexibility makes Softmax Action Selection a versatile approach in reinforcement learning, capable of adapting to different environments and learning scenarios. Although, the choice between the Softmax Action Selection and ϵ-*greedy* depends on the problem you're trying to solve.

References

[1] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[2] Stanford University. (2019). *Reinforcement Learning Course Winter 2019* [YouTube playlist]. YouTube. Retrieved from `https://www.youtube.com/playlist?list=PLoROMvodv4rOSOPzutgyCTapiGlY2Nd8u`