SOC - Week1

Kshitij Raj Suryavanshi

June 2024

1 Introduction

In the first week of this project, I focused on understanding, what is reinforcement learning? Machine learning can be broadly classified into three type, supervised learning, unsupervised learning and reinforcement learning. Reinforcement learning is a machine learning paradigm which works on the basic concept of trial and error. In reinforcement learning there is an agent in a particular state or environment which takes certain actions and in return receives a reward from the environment. The aim of this agent is to maximise the cumulative rewards which it does by gaining experience from the environment reaction.

2 Basic terms in Reinforcement Learning

Some commonly used reinforcement learning terms about which I learnt this week are as follows:

1. **Agent** - An agent in reinforcement learning is an entity which is present in the environment and makes decision by gaining experience.
2. **Environment** - The surrounding in which the agent is present and which gives feedback to the agent for its actions.
3. **Actions** - Actions are performed by the agent with the motive of changing the state of the environment and receiving feedback's for this.
4. **Reward** - Reward refers to the feedback given by the environment to the agent for the action it has performed. Reward help the agent to gain experience and take optimal actions for maximising cumulative reward over a time period.
5. **Policy** - Policy is the basic strategy which the agent follows given his particular state. The final goal of the agent is to find the optimal policy which can maximise expected future rewards.

3 Multi Armed Bandit Problem

Multi armed bandit problem is a classic problem in reinforcement learning which very clearly shows how the agent in reinforcement learning learns from its environment and takes optimal actions.

PROBLEM - In this problem the agent is provided with multiple actions. The objective of the agent is to choose actions in a certain manner such that the cumulative reward in the end is maximum. Performing a particular action gives a stochastic reward and not deterministic. This means that if the agent decides to perform action A two times then he would receive different rewards. The rewards each action provides is based on a probability distribution which is not known to the agent. As each action has a probability distribution of rewards it gives to the agent, there is a mean or expected reward for each action. This is known as the *value* of that action. None of these probability distributions are known beforehand to the agent. If this was known, then the agent would be aware of which action has the highest *value* and would have performed the same action every time to maximize its reward. This is the catch in Multi armed bandit problem, the agent has to perform actions, gain experience form the rewards and feedback it receives from the environment for performing these actions, the agent then has to calculate the expected reward of each action on its own and then take future decisions based on the previous experience.

SOLUTION - Reinforcement learning utilizes an exploration and exploitation strategy to maximizes its rewards in multi armed bandit problem. The concept used in this strategy is what differentiates reinforcement learning from other paradigms of machine learning. In this strategy the agent spends enough time to explore the action which provides the best rewards and then exploit those actions to maximize the cumulative rewards over time. Depending upon how much we explore and exploit, we use greedy algorithm or epsilon-Greedy Algorithm. In greedy algorithm, the agent performs each action once and the action which gave the maximum reward is performed every time in the future. As would have realized, this strategy is not suitable for maximizing rewards in long-term. It might give good results in short term but not in long term activities. This is because the agent has explore each action just once and as each action gives a stochastic and not a deterministic reward, there is a high chance that the action which gave him the maximum reward is not the ideal action. Epsilon - greedy is a more advisable algorithm. In this epsilon is number between 0 and 1. Epsilon indicates the fraction of times the agent is planning to explore. Here is a simulation shown which depicts how different values are affecting the average reward over time steps. From the simulation result shown below we can make some conclusions:

- When using greedy algorithm in this problem, we observe that it is performing pretty well for the initial time steps but in the end it converges at an average reward much lesser than average rewards from other strategies.

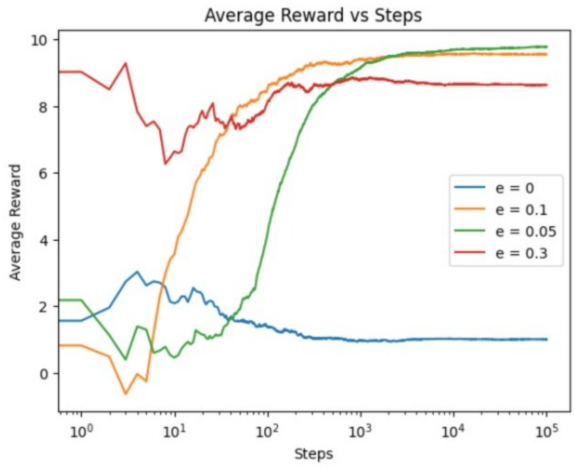


Figure 1: Average rewards vs time steps for different epsilon

- The strategy with epsilon = 0.1 and 0.05 perform the best in the end. In the initial time steps, the reward they produce is less than the greedy algorithm but after many time steps it converges at a greater average reward than others.
- The strategy with epsilon = 0.3 is exploring much compared to others, hence ends up exploiting less and finally less average reward than epsilon = 0.1 and epsilon = 0.05 strategies.
- We can conclude that the best strategy is the one which successfully balances the trade off between exploitation and exploration.

4 Markov Decision Processes and Dynamic programming

Markov Decision processes are a classical formalization of sequential decision making, where actions affect not only immediate rewards but also future states and situation and through these states future rewards. In bandit problems we estimated the value q(a) of each action a, in MDPs we estimate the value q(s, a) of each action a in each state s. The concepts which I learned in Markov Decision Processes and dynamic programming can be summarized as follows:

1. Agent - Environment interface - In Markov decision processes there is a decision maker which is called the agent. It is present in a state of environment at time step t depicted by s_t. Based on its experience it performs an action A_t which changes its state to $s_t + 1$ and gives it a reward $R_t + 1$. At each state s, the system can take a set of possible actions A(s).

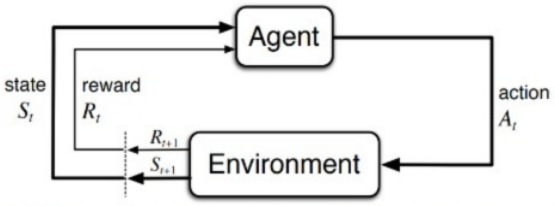


Figure 2: Agent-Environment Interface

2. Transition probability - When the agent takes an action a in the state s then there is a probability $P(s'|s,a)$, the next state will be s'.

3. Goals and Rewards - Similar to rewards in multi armed bandit problem, the reward is a way of environment giving feedback to the agent for the action it has taken. The ultimate goal of the agent is the maximization of the expected value of the cumulative sum of received reward.

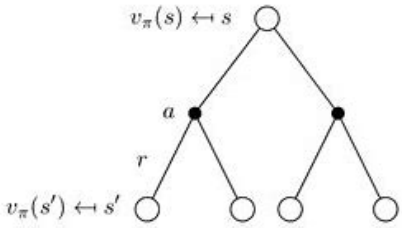
4. Returns and Episodes - Till now we know that the goal of the agent is to maximize cumulative reward but how to do so? Here comes the concept of returns, we try maximizing a mathematical expression which is a function of reward sequence. The simplest case of showing return is the sum of rewards after a time step t.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$

   The above return is when the final time step is T and after that a new episode starts. Tasks where episodes are present are called episodic tasks. There also tasks which do not terminate, these are called continuous tasks. A commonly used return is the discounted return where the future rewards are discounted at a particular rate.

5. Policies and Value function - This is a very important concept which forms the basis of dynamic programming. Value function is something which is common to all reinforcement learning algorithms. Value function tells that how good it is for the agent to be in a particular state or take a particular action being in a particular state by calculating the return for that state-action pair. The rewards and return depends upon the specific actions that the agent takes, so value functions are defined with particular ways of acting called policies. A policy is a mapping from states to probabilities of actions the agent can take.

6. Bellman's Equation - Basically, when an agent is in a particular state then it can take specific actions by following different policies. The agent has to decide that out of many policies which policy is the optimal policy which will maximize the value function for that state. So, with the help of Bellman's equation the agent tries finding the maximum expected return or



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Figure 3: Bellman's equation

maximum value function among all possible actions using value iteration. From this value function the agent extracts the optimal policy to be followed. Many also follow policy iteration which gives them optimal policy and creates the value function corresponding to that.
The aim was to maximize the cumulative reward over time but in some games or activities the possible combinations for all actions are very high. Due to this one cannot plan taking actions keeping in mind the future outcomes. Richard Bellman suggested breaking down the above complex problem into simpler units. We use the Bellman's equation to solve this. He said that instead of focusing the cumulative reward as a whole we can focus on maximizing some closer rewards in the future and use value iteration and policy iteration to approximate the best value function for maximizing the cumulative reward. Bellman's equation is a recursive equation in which the value function int he current state s is a function of the value function in other states. By using value iteration and policy iteration we can find the best value function and the optimal policy which maximizes the reward.