Reinforcement Learning

Lecture 1

Lecturer: Haim Permuter *Scribe: Omer Sholev*

In this course we will review the field of Reinforcement Learning which has gained significant popularity in many applications in recent years. We will first study the required theoretical background which includes Markov Decision Process and Dynamic Programming to solve it. Later we will discuses the differences between Markov Decision Process problem and Reinforcement Learning problem and explore ways to effectively solve Reinforcement Learning Problem using computer algorithms based upon exploring and interacting with the unknown environment.

I. Introduction

Reinforcement learning is a branch of machine learning which deals with problems involving learning appropriate actions in order to maximize a numerical reward function. The learner is not told which actions to take, it has no prior information such as to learn from such as features/labels pairs in supervised learning problem. The learner (agent) has to learn by itself which action would yield the most reward by trying them out. Reinforcement also differs from unsupervised learning as its purpose is to maximize a reward signal and not trying to find hidden structures in a given data. We therefore consider reinforcement learning to be a third machine learning paradigm, alongside supervised learning and unsupervised. So why use Reinforcement Learning? Collecting data in interactive problem for supervised learning algorithm is often very expensive and impractical to obtain. The quality of the collected Data might also be poor for appropriate supervised learning. Consider for example, supplementing data for collision avoidance to supervised learning algorithm for Autonomous car. Collecting many samples of such data might be highly expensive. So Reinforcement learning comes into play when examples of desired behavior are not available but where it is possible to monitor the response of the actions taken, and asses these actions quality using the quantity of a reward signal.

One of the challenges that arise in reinforcement learning, and not in other kinds of learning, is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. But to discover such actions, it has to try actions that it has not selected before. Therefore agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future.

II. Elements of Reinforcement Learning

A common reinforcement learning algorithm would include the following elements:

- **A Policy:** Defines the agents behavior at a given time. The policy may be a simple function or lookup table, whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it is sufficient to determine the agent's behavior.
- **A Reward Signal:** The objective of an RL agent. On each time cycle the agent receives a reward $r \in \mathbb{R}$ from the environment based on the agent's actions and state. The agents sole objective is to maximize the total reward it receives over the long run.
- **Value Function:** Whereas the reward signal indicates what is good in an immediate sense, a value function specifies what is good in the long run. The value function $V(s)$ is a function of the state and represents the total amount of reward the agent is expected to get starting from that state. We seek actions that bring about states of highest value, not highest reward, because these actions obtain the greatest amount of reward for us over the long run.
- **Environment Model:** The agent's representation of the environment.

III. Markov Decision Processes

Markov decision processes formally describe an environment for reinforcement learning. Most of the RL problems can be formalised as a Markov Decision Processes (MPD),

so any method that is suited to solve MDP problem IS considered to be a reinforcement learning method.

A. Rewards, Returns and the Markov Property

The learner and decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. As described previously, the environment gives rise to a reward that the agent tries to maximize over time by acting with appropriate actions. Therefore, the environment interacts with the agent, the agents selects actions, and gets response (reward) from the environment. Formally, the agent interacts with the environment at discrete time stamps, $t = 0, 1, 2, ...$ where at each time step the agent receives some representation of the environment state $S_t \in \mathcal{S}$ where $\mathcal{S}$ is the set of all possible states, and on that basis selects an action $A_t \in \mathcal{A}(S_t)$ where $\mathcal{A}(S_t)$ is the set of possible actions for state S_t. Formally, we define:

Definition 1 (Reward) The reward at time t:

$$R_t : A_t \times S_t \rightarrow Pr(r)\, for\, r \in \mathcal{R}_t \;\; \forall t,\; S_t \in \mathcal{S},\; A_t \in \mathcal{A}(S_t). \tag{1}$$

For a sequence of rewards received from time step t denoted $R_{t+1}, R_{t+2}, ...$ we define:

Definition 2 (Return) The Return from starting from time t is defined as:

$$G_t = R_{t+1} + R_{t+2} + ... + R_T, \tag{2}$$

where T is the stopping time we usually take $T \rightarrow \infty$

In many cases we consider a discounted return, defined by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{3}$$

where $\gamma \in (0, 1)$ is called $discount\, rate$. The discount rate helps us on 2 ways: if γ is less than 1, the infinite sum has a finite value as long as the reward sequence R_k is bounded. In addition, the discount rate lets us to model a decay in the importance of future reward compared to present reward. Notice that the return in time t can be written as:

$$G_t = R_{t+1} + \gamma G_{t+1}. \tag{4}$$

Ideally, the state signal summarizes all past history in a way that all relevant information is retained. A state signal that succeeds in retaining all relevant information is said to be $Markov$. Consider for example a chess game where the pieces location over chess board represents the state. This state signal has the Markov property as it summarizes all history that led to that position, the information about how we came up with this configuration is lost, but all that really matters for a decision to be made is is retained within this current chess board configuration. We now formally define the Markov property for RL problem. In the most general case, the reward signal R_{t+1} and the next state signal S_{t+1} depend on all previous actions, states and rewards:

$$Pr(S_{t+1} = s', R_{t+1} = r | S_0 = s_0, A_0 = a_0, R_0 = r_0, ..., S_t = s_t, R_t = r_t, A_t = a_t). \quad (5)$$

If the state signal has the Markov property, then the environment response at time $t + 1$ depends only on state and action at time t.

Definition 3 (Markov Property) An environment is said to have the Markov property if:

$$\begin{aligned} Pr(S_{t+1} = s', R_{t+1} = r | S_0 = s_0, A_0 = a_0, R_0 = r_0, ..., S_t = s_t, R_t = r_t, A_t = a_t) \\ = Pr(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a) \quad \forall r, s', s, a. \end{aligned} \quad (6)$$

B. Policies and Value Functions

Like stated in previous section, a typical RL algorithm includes a *Policy* and a *Value Function*.

Definition 4 (Policy) A policy is a mapping from states to probabilities of selecting each possible action If the agent is following policy π at time t, then $\pi(a|s)$ is the probability for choosing $A_t = a$ when $S_t = s$ where $s \in \mathcal{S}$ and $a \in \mathcal{A}(S_t)$

The value function is a function of the state which tells us how good is it for the agent to be in s specific state. Since, future rewards depends upon the agent's actions, the value function is defined with respect to a certain policy - a way of acting.

Definition 5 (Value Function) The value function of state s under policy π is the expected return when the agent is in state $S_t = s$ at time t and acting according to policy π

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi\Big[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s\Big]\ \forall s \in S. \tag{7}$$

where $E_\pi[\dot{]}$ denotes the expected value with respect to the policy π.

Definition 6 (Action Value Function) Similarly, we define the *Action Value Function* which quantifies the expected return for the agent when is found to be in state $S_t = s$ at time t , acting with action $A_t = a$ and following policy π later on

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi\Big[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\Big]. \tag{8}$$

Notice that:

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(a,s). \tag{9}$$

When the environment dynamic, i.e $Pr(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$ is known for all s, a, r the agent's sole objective is to maximize the Return by optimizing its Policy (Can be done using Dynamic Programming methods which will be covered later on). When the the environment dynamic is not known, the agent needs to corporate a learning process in which $q_\pi(s,a), v_\pi(s)$ are estimated from experience using techniques which will also be discussed in future lectures. The Value Function satisfies a very nice property that is being used relates the current Value Function to its immediate expected reward and next state value function:

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] && (10)\\ &= E_\pi\Big[R_{t+1} + \gamma G_{t+1}|S_t = s\Big] && (11)\\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\Big[r + \gamma E_\pi\big[G_{t+1}|S_{t+1} = s'\big]\Big] && (12)\\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]. && (13) \end{aligned}$$

This relation is called *Bellman Equation* which is a fundamental in the implementation of DP methods. Note that the term in (13) is the sum of 2 terms that can be interpreted as the current expected reward and future returns:

- **T**he sum of immediate expected reward:

$$\sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)r = \sum_{a,s',r} p(a, s', r|s)r. \tag{14}$$

- **F**uture expected rewards which are all being weighted by they appropriate probability:

$$\sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)\gamma v_{\pi}(s'). \tag{15}$$

C. Optimal Policies

Since Value Functions depends on the agent's policy π, it is natural to ask whether a specific policy π' might yield better return than policy π. For finite MDP, we define the policy π' to be better than or equal to policy π if its expected return is greater or equal than the expected return of π formally:

$$\pi' \geq \pi \text{ if and only if } v_{\pi'}(s) \geq v_{\pi}(s) \ \forall s \in S. \tag{16}$$

There is always at least one policy that is better than all other policies (there may be more than one) and we define it as follows:

Definition 7 (Optimal Value Function)

$$v_*(s) = \max_{\pi} v_{\pi}(s) \ \forall s \in S. \tag{17}$$

Optimal policies share the same action value function as well:

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a) \ \forall s \in S, a \in A(s). \tag{18}$$

This term quantifies the expected return when acting a when in state s and following optimal policy π_* later on, thus we can write:

$$q_*(s, a) = E\Big[R_{t+1} + \gamma v_*(S_{t+1})|S_t = s, A_t = a\Big]. \tag{19}$$

Definition 8 (Bellman Optimality Equation) For optimal policy, the bellman equation satisfies:

$$v_*(s) = \max_{a \in A(s)} q_{\pi_*}(s,a) = \max_a E_{\pi_*}\big[G_t | S_t = s, A_t = a\big] \tag{20}$$

$$= \max_a E\Big[R_{t+1} + \gamma v_*(S_{t+1} | S_t = s, A_t = a\Big] \tag{21}$$

$$= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]. \tag{22}$$