RL Notes!

1 Introduction

The fundamental idea of reinforcement learning that of an adaptive, "hedonistic" system; one that changes it's behaviour based on it's goal and can adapt itself to maximise some signal that it can induce from its environment. Reinforcement learning is learning how to map situations to actions, to maximise a numerical reward function. The agent is not told which specific actions to take, but must discover which actions yield the most reward; not only at the current timestep, but also to suit the next situation end and all subsequent stops and rewards.

RL is essentially a model of the learning problems that an agent, who learns behaviour through trial and interactions with a dynamic environment, encounters in its lifetime.

2 Elements

The main elements of a reinforcement learning setup are a *policy*, *reward function*, *value function* and a model of the *environment*. The policy is what defines the agent's behaviour and actions at a point in time. It is a mapping from perceived states of the environment, to the action that the learning agent takes when in those states.

Policies may be stochastic, and these are what determine the behaviour of the agent.

The reward is what defines the goal of the reinforcement learning problem. For example, the learning agent can receive some real-valued reward at every times tep based on the actions it takes, and it's objective could be to maximise the total reward it receives at the end. In general, these may be stochastic functions of the state of the environment and the actions taken.

The value of a state, is the total amount of reward that an agent can expect to attain when starting from that state. Therefore, while rewards determine immediate desirability of states, values determine the long-term desierability of states, after taking into account what states can follow and what rewards can be attained at the following states.

A model of the environment is the final element, that simulates the behaviour of the environment i.e. for a current state and action, predicts the resultant next state and next reward.

We can have reinforcement learning systems that are both model-based and model-free.

2.1 How is this different from Supervised Learning?

In RL, there is typically no pairing of the input and output, as in the case of SL. An agent in a dynamic environment chooses actions based on a policy, and receives reinforcement. The worlds here are non-deterministic and the the reinforcement learning system must explore the space of actions and determine the optimal policy to maximise reward.

However, reinforcement learning can be equivalent to supervised learning. Consider a world and an agent in the world s.t., only two possible actions can be taken at any point in time. If reinforcement is boolean and given only in the form of a signal, and the world is deterministic, this is now similar to a supervised learning system. When action a is chosen by an agent in state s, we now have a pairing to learn from.

- if $reward(a, s)$ is True, then $f(s) = a$
- if $reward(a, s)$ is False, then $f(s) =\nleftarrow a$

2.2 Models of Optimal Behaviour

Before algorithms that allow optimal behaviour come into play, we have to decide what our models of optimality will be i.e., we have to determine exactly how the agent takes the future and the current world into account while making a decision.

2.2.1 Finite Horizon Model

At a given point in time, an agent must optimise it's expected reward for the next h steps i.e., a finite number of steps.

$$E\left(\sum_{t=0}^{h} r_t\right) \tag{1}$$

The agent only worries about the next h steps, and here r_t refers to the scalar reward received t steps in the future.

The agent can have a non-stationary policy that changes over time. On its first step it takes an h-step optimal action i.e., the best action given the remaining h steps to act and gain reinforcement. It uses the same policy, but the value of h limits how far it can look ahead to determine which action to take. This model therefore, is not always appropriate. Moreover, we may not always know the exact length of the agent's life in advance and whether it falls under h.

2.2.2 Infinite Horizon Discounted Model

This model takes the reward of the agent in the long-run into account, along with a discount factor. Rewards that are received in the future are geometrically discounted in accordance with a discount factor that lies between 0 and 1.

$$E\left(\sum_{t=0}^{\infty} r_t\right) \tag{2}$$

2.2.3 Average Reward Model

This model ensures that the agent takes actions to optimise its average reward in the long run.

$$\lim_{h\to\infty} E\left(\frac{1}{h}\sum_{t=0}^{h} r_t\right) \tag{3}$$

This is a gain optimal policy and is exactly the limiting case of the infinite horizon discounted model as the discount factor tends to infinity. Here however, there is no way to distinguish between two policies i.e., if one starts off by gaining a large reward in the initial phases and the other does not.

Overall, the finite horizon model is appropriate when the agent's lifetime is known and the system has a hard deadline. Infinite horizon discounted and bias optimal models are sometimes preferred when we have no knowledge of the agent's lifetime and do not want to take into account a discount factor, for the bias-optimal policies.

3 Multi-armed Bandits

We should note that an important distinguishing feature of RL from other types of learning is that it is *evaluative* as opposed to just *instructive*. Essentially, this indicates how good the action that was taken was, rather than an instructive approach that indicates the best action to take, independent of the action that is taken. Evaluative feedback therefore entirely depends on the action taken, while instructive is independent of the action taken. The non-associative setting is one in which prior work involving evaluative feedback has been done. Here we explore a particular non-associative, evaluative feedback problem i.e., the k-armed bandit problem.

3.1 A k-armed Bandit Problem

Consider the k-armed bandit learning problem. We are repeatedly faced with a choice among k options, or actions. After each action or choice we receive a numerical reward that is chosen from a stationary probability distribution and our objective is to maximise the total expected reward over some time period i.e., some number of action selections or time steps.

In this problem, each of the k actions has an expected reward given that that action is selected; and this is called the *value* of that action. Let us denote the action selected on time step t as A_t with corresponding reward R_t, then the value of an arbitrary action a, denoted $q_*(a)$, is the expected reward given that action a is selected:

$$q_*(a) = E[R_t|A_t = a] \tag{4}$$

Now if we knew the value of each action, we could simply choose the action with the highest value at each time-step, and solve the k-armed banding problem. Let us assume we do not know action values with certainty, although we have an estimated value of action a at time step t as $Q_t(a)$. We therefore want $Q_t(a)$ to be as close to $q_*(a)$

If we maintain estimates of the action values, then we know that at any time step there is at least one action whose estimated value is the greatest. These are the *greedy* actions. When we select one of these actions, we are *exploiting* current knowledge of values of actions. When we select one of the non-greedy actions, we are *exploring*, which allows us to improve the estimate of the non-greedy action's value. Exploitation can maximise the expected reward at the current step, but exploration may produce the greater total reward in the long run.

Therefore, for any specific case, the choice between exploration and exploitation depends in a complex way, on the values of the estimates, uncertainties and the number of remaining steps. Balancing exploration and exploitation has been studied for a long time, and many sophisticated methods exist to balance this for particular mathematical formulations of the k-armed bandit and related problems. Most of these, however, make strong assumptions about stationarity and prior knowledge. But these are either violated or cannot be verified in the full reinforcement learning problem, and the guarantees of optimality or bounded loss for these methods cannot hold given that the assumptions of their theory do not apply.

3.2 Action-value Methods

Q-Learning Q-Learning aims to learn better quality policies by filling up a Q-table using the Bellman equation as stated below, to improve the quality of the policy (i.e., the correct $(s,\ a)$ pairs that an agent must follow to reach the goal.

$$Q(s, a) = r + \gamma(\mathsf{max}(Q(s', a'))) \tag{5}$$

The Q-table is of size $s \times a$ where $Q(s, a)$ is the current policy of action a from state s, r is the reward for an action, $\mathsf{max}(Q(s', a'))$ defines the maximum future reward and γ is the discount factor (this varies from 0 to 1, where values near 0 give preference to immediate reward and values near 1 give more importance to future rewards). With this, we attempt to formulate the delayed rewards into immediate rewards.

For a given goal or problem we want to solve, we can create an environment matrix that defines the (positive and negative) rewards we can get by taking

actions in states (that potentially lead to the correct goal state). Our q matrix is then updated iteratively by selecting actions at states, updating the q matrix using the Bellman equation and reward from the environment matrix, going to the corresponding next states, until the correct goal state is reached.