Reinforcement Learning Notes

Kiran Kannar

July 19, 2025

Contents

1 Introduction to Reinforcement Learning

Computational approach to **goal-directed** learning by an agent from interactions with an environment.

- Trial-and-error search
- Delayed reward
- Exploration vs exploitation
- Model-based vs model-free

Broadly, maximize reward signal, despite uncertainty about the environment $\rightarrow$ Optimization problem.

1.1 Elements of RL

A Markov Decision Process (MDP) involving:

- Policy : Mapping from states to actions
- Reward Signal : Scalar signal that indicates the **desirability** of the state i.e. immediate value
- Value Function : Expected cumulative reward from a state i.e. far-sighted judgment of value of starting from a particular state.
- Model (of the environment): Predicts the next state and reward given the current state and action; used for planning.

Value function is computed without **explicit search** over possible sequences of future states and actions. Focus is on highest value, and not on highest reward, even though value is computed from rewards.

2 Multi-Armed Bandit Problems

Setting: A single state (aka situation) with multiple actions, one of which must be selected repeatedly. Each arm is slot machine with a stationary probability distribution of rewards, say, $\mathcal{N}(\mu_i, \Sigma_i)$. The goal is to maximize the expected reward over some time horizon.

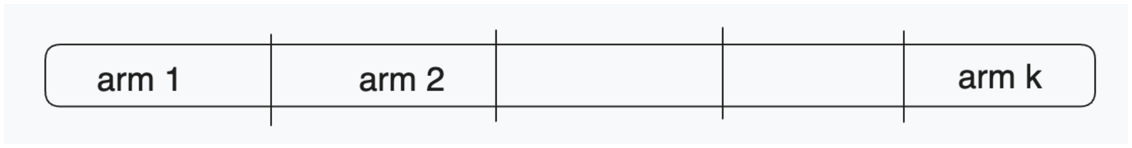

Figure 1: A single state with multiple actions.

The expected reward of taking an action is given by:

$$q_*(a) = \mathbb{E}[R_t | A_t = a] \tag{1}$$

This is the value of taking that action. We do not however know the value of each action, but we can estimate it by sampling the reward. Thus, we approximate the value of an action as $Q_t(a) \approx q_*(a)$.

2.1 Action-Value Methods

Methods to estimate the value of an action and then use that estimate to select actions.

2.1.1 Sample-Average Method:

Each estimate is the average of all the sample of rewards collected up to that point.
Suppose $Q_t(a) = \frac{1}{N_t(a)} \sum_{i=1}^{N_t(a)} R_i$, where $N_t(a)$ is the number of times action a has been selected up to time t.

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot 1(A_i = a)}{\sum_{i=1}^{t-1} 1(A_i = a)} \tag{2}$$

Action Selection:

- Greedy: $A_t = \text{argmax}_a Q_t(a)$
- ϵ-greedy: $A_t = \begin{cases} \text{argmax}_a Q_t(a) & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$

With ϵ-greedy, as $t \to \infty$, each action is sampled infinite times, and therefore the estimate $Q_t(a)$ converges to $q_*(a)$. The greedy selection, however, does not have strong convergence properties.

With ‘k’ arms in ϵ-greedy method, the probability of selecting the greedy action is,

$$\begin{aligned} P(\text{greedy action}) &= (1-\epsilon).P(\text{action is greedy}) + \epsilon.P(\text{action is random}) \\ &= (1-\epsilon).1 + \epsilon.\frac{1}{k} \\ &= 1-\epsilon+\frac{\epsilon}{k} \end{aligned} \tag{3}$$

The expected reward using ϵ-greedy method is $(1-\epsilon).q_*(a^*)$, because of the strong convergence properties of the sample-average method. That is, in the long run, the expected reward will be equal to the expected value of selecting the greedy action.

- If the variance of rewards increases across arms, it’s always good to explore quickly. ϵ-greedy method is better than greedy method.
- If the task is non-stationary, then the ϵ-greedy method is good because we can explore previously explored actions whose rewards have potentially increased.

Incremental Implementation:

$$Q_{t+1}(a) = Q_t(a) + \frac{1}{N_t(a)}(R_t - Q_t(a)) \tag{4}$$

Simple bandit algorithm:

Algorithm 1 Simple Bandit Algorithm

```
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$
**for** $t = 1$ to $T$ **do**
    $A_t \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with probability } 1-\epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$
    $R_t \leftarrow \text{reward}(A_t)$
    $Q(A_t) \leftarrow Q(A_t) + \frac{1}{N(A_t)}[R_t - Q(A_t)]$
    $N(A_t) \leftarrow N(A_t) + 1$
**end for**
```

2.1.2 Non-Stationary Problems: Exponential Recency Weighting

In the case of non-stationary problems, we would want to give more weight to recent rewards.

$$\begin{aligned} Q_{t+1}(a) &= Q_t(a) + \alpha[R_t - Q_t(a)] \\ &= (1-\alpha)^t Q_1(a) + \sum_{i=1}^{t} \alpha(1-\alpha)^{t-i} R_i \end{aligned} \tag{5}$$

Note that all of the above methods are biased by the initial estimate $Q_1(a)$ (aka prior knowledge). For sample average methods, the bias disappears once all actions are explored at least once. $Q_t(a) >> R_t$ allows us to explore the actions more quickly. Wild optimism is a good strategy. For non-stationary problems, however, the bias is persistent but decreasing over time. So this drive for exploration is temporary.

2.2 Gradient Bandit

Instead of estimating the value of each action, we estimate the numerical preference for each action, $H_t(a)$. With exact gradient ascent,

$$H_{t+1}(a) = H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} \tag{6}$$

$$\pi_t(A_t = a) = P(A_t = a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \tag{7}$$

Hence, $\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$; Given that we do not know $q_*(x)$, we can show that this is an instance of stochastic gradient ascent with robust convergence properties.

$$H_{t+1}(a) = \begin{cases} H_t(a) + \alpha(R_t - R_{avg_t})(1 - \pi_t(A_t)) & \text{if } a = A_t \\ H_t(a) - \alpha(R_t - R_{avg_t})\pi_t(a) & \text{for all } a \neq A_t \end{cases} \tag{8}$$

2.3 Contextual Bandits

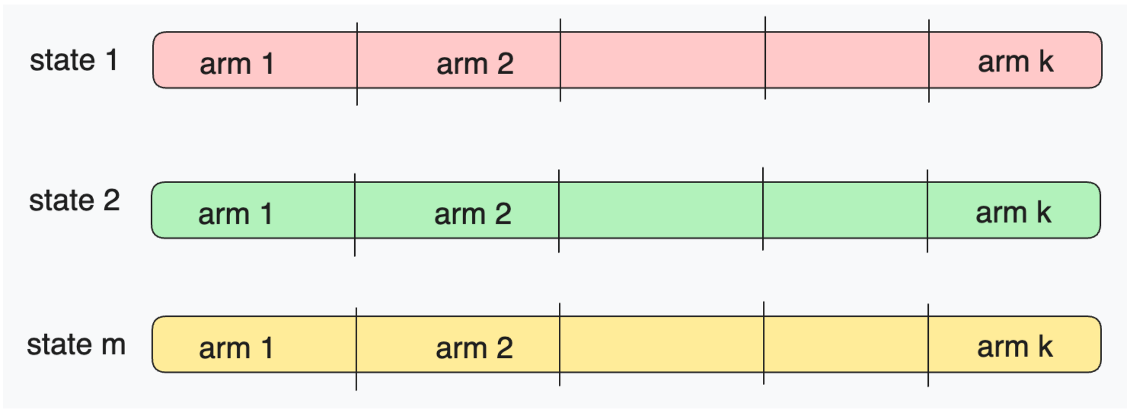

Figure 2: Contextual Bandit with many states, each with multiple actions.

Associative search is a contextual bandit problem where we do trial-and-error learning to

- search for the best action, and,
- the association of these actions with the best situation (state) to be in.

Each (s, a) pair results in a reward $R_{s,a}$. However, it does not change the state s. If the state changes, we are in a full reinforcement learning problem.

3 Finite Markov Decision Processes

MDP Formulation: $s_t, a_t \to r_{t+1}, s_{t+1}$

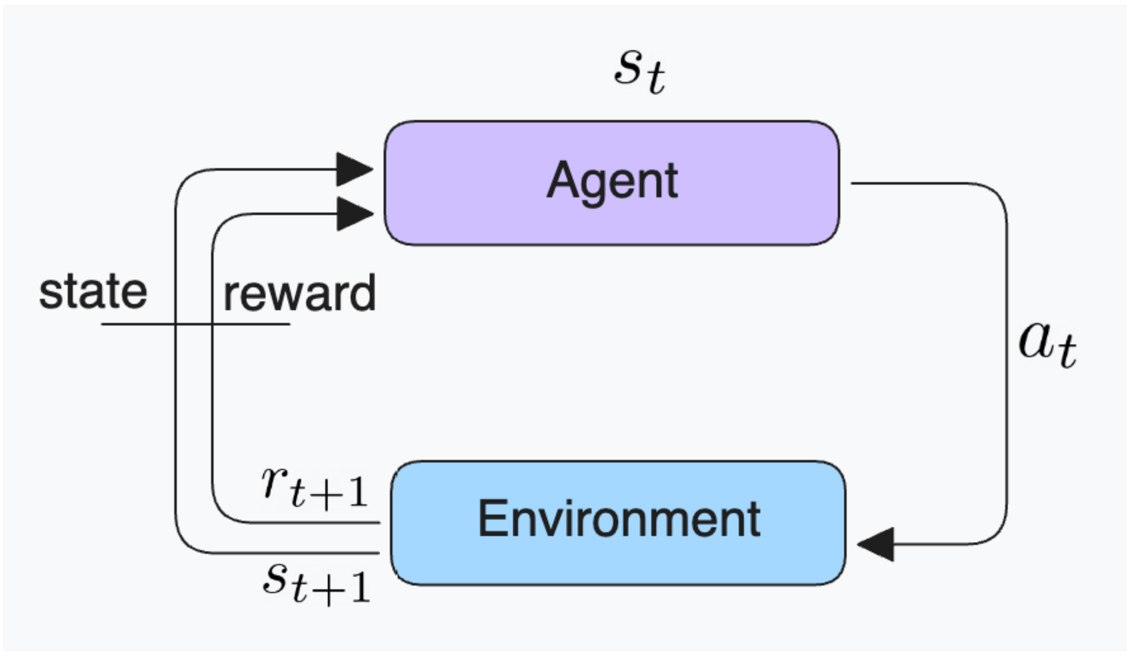

Figure 3: MDP Formulation.

$p(s', r|s, a)$ is the probability of transitioning to state s' and receiving reward r given that we are in state s and take action a. The current state s_t includes all the information about the past. (i.e., the Markov property)

The state transition probability is given by:

$$p(s'|s, a) = \sum_r p(s', r|s, a) \tag{9}$$

The reward function is given by:

$$\begin{aligned} r(s, a) &= \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \\ &= \sum_r r \sum_{s'} p(s', r|s, a) \end{aligned} \tag{10}$$

The MDP is defined by the tuple (S, A, p, r, γ), where S is the state space, A is the action space, p is the state transition probability, r is the reward function, and γ is the discount factor.

Goal: Maximize expected (discounted) return over time, with discount factor $\gamma \in [0, 1]$.

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ G_t = R_{t+1} + \gamma G_{t+1} &\quad \textit{(recursive definition)} \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad \textit{(infinite sum)} \end{aligned} \tag{11}$$

3.1 Policy and Value Functions

A policy $\pi(a|s)$ is a mapping from states to actions indicating the probability of taking action a in state s.

The **state-value function** $v_\pi(s)$ is the expected return starting from state s and following policy π.

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s] \quad \text{for all } s \in S \tag{12}$$

The **action-value function** $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and following policy π.

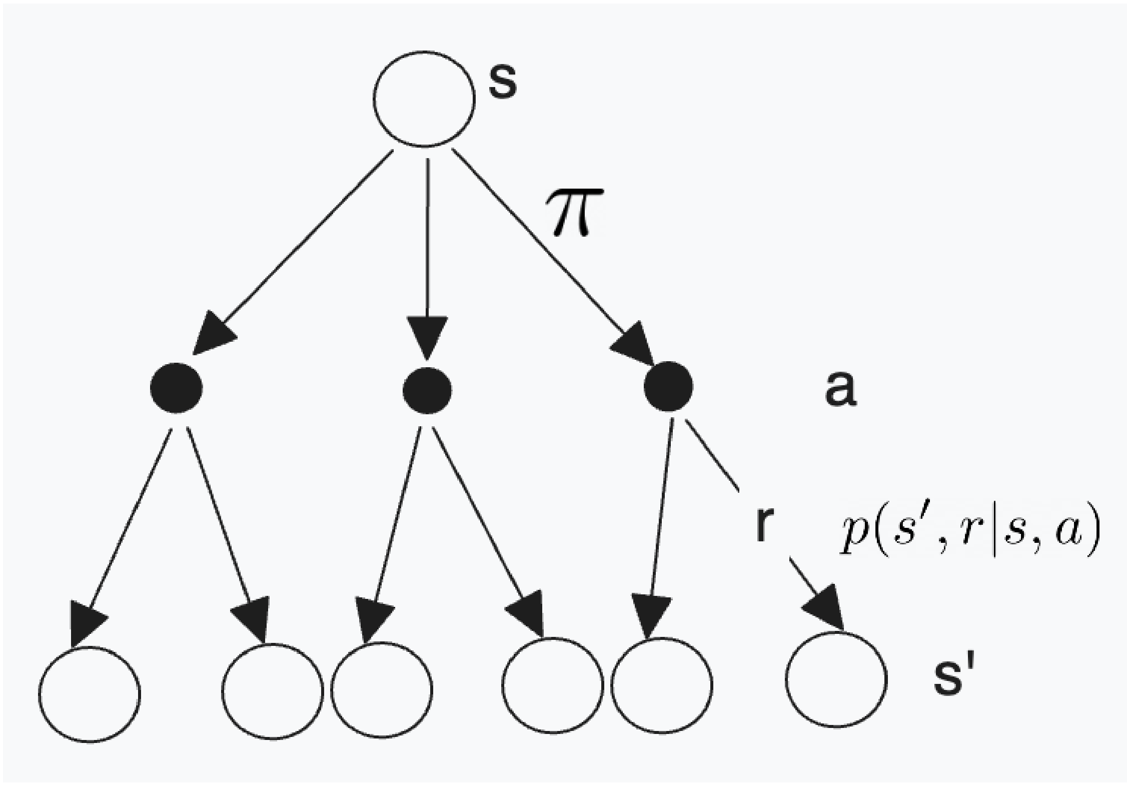

Figure 4: General setting of MDP.

$$q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a] \tag{13}$$

Expected reward at time t is given by:

$$\begin{aligned} \mathbb{E}[R_{t+1}|S_t = s, A_t = a] &= \sum_r r \sum_{s'} p(s', r|s,a) \\ &= \sum_a \pi(a|s) \sum_{s',r} r.p(s', r|s,a) \end{aligned} \tag{14}$$

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t|S_t = s] \\ &= \sum_{G_t} G_t.p(G_t|S_t = s) \\ &= \sum_{G_t} G_t.p(R_t|S_t = s) \\ &= \sum_{G_t} G_t. \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \\ &= \sum_{G_t} G_t. \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \\ v_\pi(s) &= \sum_a \pi(a|s) q_\pi(s,a) \end{aligned} \tag{15}$$

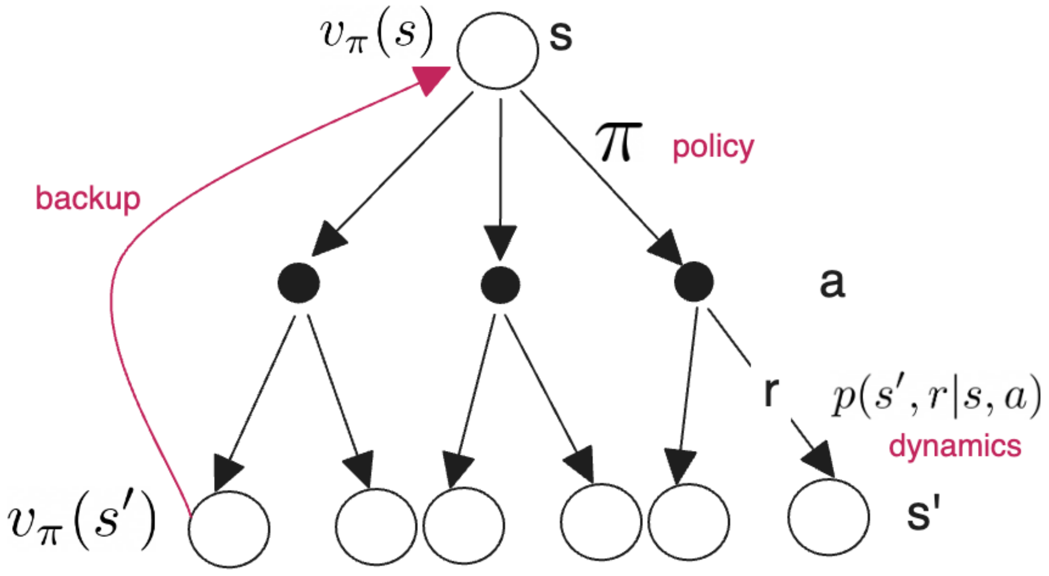

Figure 5: Back-up diagram for the value function.

Furthermore, the value function can be decomposed into the immediate reward plus the discounted value of the next state:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ v_\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{16}$$

This is the **Bellman equation** for the value function v_π, averaging over all possible next actions, weighted by the probability of occuring.

The **Bellman equation** for the action-value function $q_\pi(s, a)$ is given by:

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ q_\pi(s, a) &= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{17}$$

Derivation below:

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \mathbb{E}[R_{t+1} | S_t = s, A_t = a] + \gamma \mathbb{E}[G_{t+1} | S_t = s, A_t = a] \\ &= \sum_{r,s'} r.p(s', r|s, a) + \gamma \sum_{s'} v_\pi(s') p(s'|s, a) \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{18}$$

In summary,

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) q_\pi(s, a) \\ v_\pi(s) &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \\ q_\pi(s, a) &= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \\ q_\pi(s, a) &= \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')] \end{aligned} \tag{19}$$