Reading of theoretical aspect of reinforcement learning – Chapter 5

Zhe Rao

- In this chapter we consider our first learning methods for estimating value functions and discovering optimal policies.
  - Unlike the previous chapters, here we do not assume complete knowledge of the environment.
  - Monte Carlo methods require only ***experience*** – sample sequences of states, actions, and rewards from actual or simulated interaction with an environment.
    - Learning from actual experience is striking because it requires no prior knowledge of the environment's dynamics, yet can still attain optimal behavior.
    - Learning from simulated experience is also powerful. Although a model is required, the model need only generate sample transitions, not the complete probability distributions of all possible transitions that is required for **dynamic programming**.

- Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns.
  - To ensure that well-defined returns are available, here we define Monte Carlo methods only for episodic tasks.
  - That is, we assume experience is divided into episodes, and that all episodes eventually terminate no matter what actions are selected.
  - Only on the completion of an episode are value estimates and policies changed.
  - Monte Carlo methods can thus be incremental in an episode-by-episode sense, but not in a step-by-step sense.

- Monte Carlo methods sample and average ***returns*** for each state-action pair much like the bandit methods that sample and average ***rewards*** for each action.
  - The main difference is that now there are multiple states, each acting like a different bandit problem, and the different bandit problems are interrelated. That is, the return after taking an action in one state depends on the actions taken in later states in the same episode.

- Because all the action selections are undergoing learning, the problem becomes ***nonstationary*** from the point of view of the earlier state.
  - To handle the nonstationary, we adapt the idea of general policy iteration (GPI) developed in chapter 4 for DP
  - Whereas there we computed value functions from knowledge of the MDP, here we learn value functions from sample returns with the MDP

- As in the DP chapter, first we consider the prediction problem (the computation of v_π and q_π for a fixed arbitrary policy π) then policy improvement, and, finally, the control problem and its solution by GPI
- Each of these ideas taken from DP is extended to the Monte Carlo case in which only sample experience is available.

1. Monte Carlo Prediction

- We begin by considering Monte Carlo methods for learning the state-value function for a given policy
  - Recall that he value of a state is the expected return – expected cumulative future discounted reward- starting from that state
  - An obvious way to estimate it from experience, then, is simply to average the returns observed after visits to that state.
  - As more returns are observed, the average should converge to the expected value.
  - This idea underlies all Monte Carlo methods

- In particular, suppose we wish to estimate $v_\pi(s)$, the value of a state s under policy π, given a set of episodes obtained by following π and passing through s.
  - Each occurrence of state s in an episode is called a ***visit*** to s.
  - Of course, s may be visited multiple times in the same episode; let us call the first time it is visited in an episode the ***first visit*** to s.
    - The ***first visit*** MC method estimates $v_\pi(s)$ as the average of the returns following first visits to s, whereas the ***every-visit*** MC method averages the returns following all visit to s.
    - These two MC methods are very similar but have slightly different theoretical properties

- First-visit MC is shown in procedural form in the box. Every-visit MC would be the same except without the check for S_t having occurred earlier in the episode

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average($Returns(S_t)$)

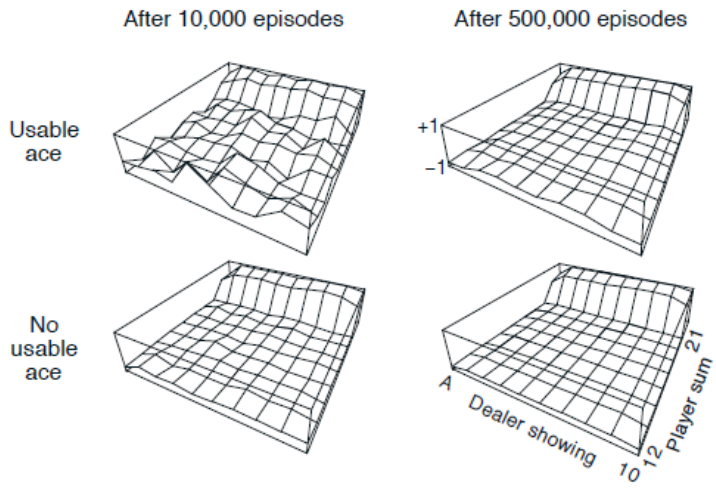
  - Both first-visit MC and every-visit MC converge to $v_\pi(s)$ as the number of visits (or fist visits) to s goes to infinity

Example 5.1: Blackjack

- The object of the popular casino card game of blackjack is to obtain cards the sum of whose numerical values is as great as possible without exceeding 21.
  - All face cards count to 10, and an ace can count either as 1 or as 11.
  - We consider the version in which each player competes independently against the dealer.
  - The game begins with two cards dealt to both dealer and player.
    - One of the dealer's cards is face up and the other is face down.
  - If the payer has 21 immediately (an ace and a 10-card), it is called a *natural*.
    - He then wins unless the dealer also has a natural, in which case the game is a draw.
  - If the player does not have a natural then he can request additional cards, one by one (hits), until he either stops (sticks) or exceeds 21 (go bust)
    - If he goes bust, he loses; if he stocks, then it becomes the dealer's turn.
    - The dealer hits or sticks according to a fixed strategy without choice: he sticks on any sum of 17 or greater, and hits otherwise.
    - If the dealer goes bust, then the player wins; otherwise, the outcome – win, lose, or draw – is determined by whose final sum is closer to 21

- Playing blackjack is naturally formulated as an episodic finite MDP.
  - Each game of blackjack is an episode.
  - Rewards of +1, -1, 0 are given for winning, losing, and drawing.
    - All rewards within a game are zero, and we do not discount: therefore these terminal rewards are also the returns.
  - The player's actions are to hit or to stick.
  - The states depend on the player's cards and the dealer's showing card.
  - We assume that cards are dealt from an infinite deck (i.e. with replacement) so that there is no advantage to keeping track of the cards already delt.
  - If the player holds an ace that he could count as 11 without going bust, then the ace is said to be ***usable***.
    - In this case, it is always counted as 11 because counting it as 1 would make the sum 11 or less, in which case there is no decision to be made because, obviously, the player should always hit.
  - Thus, the player makes decisions on the basis of thee variables
    - His current sum (12-21), the dealer's one showing card (ace-10), and whether or not he holds a usable ace.
    - This makes for a total of 200 states

- Consider the policy that sticks if the player's sum is 20 or 21, and otherwise hits.
  - To find the state-value function for this policy by a Monte Carlo approach, one simulates many blackjack games using the policy and averages the return following each state.
  - In this way, we obtained the estimates of the state-value function shown in Figure 5.1. The estimates for states with a usable ace are less certain and less regular because these states are less common.
  - In any event, after 500,000 games the value function is very well approximated.



- Although we have complete knowledge of the environment in the blackjack task, it would not be easy to apply DP methods to compute the value function
  - DP methods require the distribution of next events – in particular, they require the environments dynamics as given by the four-argument function p – and it is not easy to determine this for blackjack.
    - For example, suppose the player's sum is 14 and he chooses to stick. What is his probability of terminating with a reward of +1 as a function of the dealer's showing card?
  - All of the probabilities must be computed before DP can be applied, and such computations are often complex and error-prone.
  - In contrast, generating the sample games required by Monte Carlo methods is easy

- We can generalize the idea of backup diagram to Monte Carlo algorithms.
  - The general idea of a backup diagram is to show at the top of the root node to be updated and to show below all the transitions and leaf nodes whose rewards and estimated values contribute to the update
  - For Monte Carlo estimation of v_π, the root is a state node, and below it is the entire trajectory of transitions along a particular single episode, ending at the terminal state, as shown here.
  - Whereas the DP diagram shows all possible transitions, the MC diagram shows only those sampled on the one episode,
  - Whereas the DP diagram includes only one-step transitions, the MC diagram goes all the way to the end of the episode.
  - These differences in the diagrams accurately reflect the fundamental differences between the algorithms

- In important fact about MC methods is that he estimates for each state are independent. The estimate for one state does not build upon the estimate of any other state, as is the case in DP
  - In particular, note that the computational expense of estimating the value of a single state is independent of the number of states.

- This can make MC methods particularly attractive when one requires the value of only one or a subset of states.
- One can generate many sample episodes starting from the states of interest, averaging returns from only these states, ignoring all others.
- This is a third advantage MC methods can have over DP methods (after the ability to learn from actual experience and from simulated experience)

2. Monte Carlo Estimation of Action Values

- If a model is not available, then it is particularly useful to estimate action values (the values of state-action pairs) rather than state values.
  - With a model, state values alone are sufficient to determine a policy; one simply looks ahead on step and choose whichever action leads to the best combination of reward and next state, as we did in the chapter on DP.
  - Without a model, however, state values alone are not sufficient
    - One must explicitly estimate the value of each action in order for the values to be useful in suggesting a policy.
  - Thus, one of our primary goals for Monte Carlo methods is to estimate q_*.
    - To achieve this, we first consider the policy evaluation problem for action values

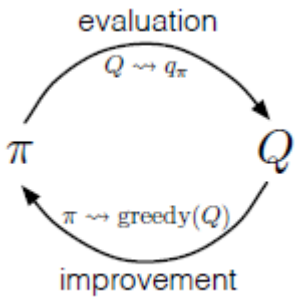
- The policy evaluation problem for action value is to estimate $q_\pi(s, a)$, the expected return when starting in state s, taking action a, and thereafter following policy π.
  - The Monte Carlo methods for this are essentially the same as just presented for state values, except now we talk about visits to a state-action pair rather than to a state.
  - A state-action pair s, a is said to be visited in an episode if ever the state s is visited and action a is taken in it
  - The ***every-visit*** MC method estimates the value of a state-action pair as the average of the returns that have followed all the visits to it.
  - The ***first-visit*** MC method averages the returns following the first time in each episode that the state was visited and the action was selected.
  - These methods converge quadratically, as before, to the true expected values as the umber of visits to each state-action pair approaches infinity

- The only complication is that many state-action pairs may never be visited.
  - If π is deterministic policy, then in following π one will observe returns only for one of the actions from each state
  - This is a serious problem because the purpose of learning action values is to help in choosing among the action available in each state.
    - To compare alternatives we need to estimate the value of all the actions from each state, not just the one we currently favor.

- This is the general problem of ***maintaining exploration***
  - For policy evaluation to work for action values, we must assure continual exploration.
  - One way to do this is by specifying that the episodes *start in a state-action pair*, and that every pair has a nonzero probability of being selected as the start
  - We call this the assumption of ***exploring starts***

3. Monte Carlo Control

- We now ready to consider how Monte Carlo estimation can be sued in control, that is, to approximate optimal policies
  - The overall idea is to proceed according to the same pattern as the DP chapter, that is, according to the idea of generalized policy iteration (**GPI**).
  - In GPI one maintains both an approximate policy and an approximate value function.
    - The value function is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function, as suggested by the diagram.



- To begin, let us consider a Monte Carlo version of classical policy iteration.
  - In this method, we perform alternating complete steps of policy evaluation and policy improvement, beginning with an arbitrary policy π_0 and ending with the optimal policy and optimal action-value function

$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*,$$

  - The first kind of arrow denotes a complete policy evaluation and second kind denotes a complete policy improvement.
  - Policy evaluation is done exactly as described in the preceding section.
  - Many episodes are experienced, with the approximate action-value function approaching the true function asymptotically.

- Policy improvement is done by making the policy greedy with respect to the current value function.
  - In this case we have an action-value function, and therefore no model is needed to construct the greedy policy.
  - For any action-value function q, the corresponding greedy policy is the one that, for each state, deterministically chooses an action with maximum action-value

$$\pi(s) \doteq \arg\max_a q(s,a). \tag{5.1}$$

  - Policy improvement then can be done by constructing each π_{k+1} as the greedy policy with respect to q_{π_k}
  - The policy improvement theorem then applies to π_k and π_{k+1} because, for all states

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s). \end{aligned}$$

- As discussed in the previous chapter, the theorem assures us that each π_{k+1} is uniformly better than π_k, or just as good as π_k, in which case they are both optimal policies.
- This in turn assures us that the overall process converges to the optimal policy and optimal value function.
- In this way Monte Carlo methods can be used to find optimal policies given only sample episodes and no other knowledge of the environment's dynamics

- For now we focus on the assumption that policy evaluation operates on an infinite number of episodes
  - This is relatively easy to be removed
  - DP has the same issue that it converges only asymptotically to the true value function.
  - There are two ways to solve the problem
    - One is to hold firm to the idea of approximating q_{π_k} in each policy evaluation.
      - Measurements and assumptions are made to obtain bounds on the magnitude and probability of error in the estimates, and then sufficient steps are taken during each policy evaluation to assure that these bounds are sufficiently small
      - This approach can probably be made completely satisfactory in the sense of guaranteeing correct convergence up to some level of approximation.
      - However, it is also likely to require far too many episodes to be useful in practice on any but the smallest problems
    - The second approach is to give up trying to complete policy evaluation before returning to policy improvement
      - On each evaluation step we move the value function toward q_{π_k}, but we don't expect to actually get close except over many steps.
      - We used this idea when we first introduced the idea of GPI in section 4.6. One extreme form of the idea is value iteration, in which only one iteration of iterative policy evaluation is performed between each step of policy improvement.
      - The in-place version of value iteration is even more extreme; there we alternate between improvement and evaluation steps for single states

- For Monte Carlo policy iteration it is natural to alternate between evaluation and improvement on an episode-by-episode basis.
  - After each episode, the observed returns are used for policy evaluation, and then the policy is improved at all the states visited in the episode.
  - A complete simple algorithm along these lines, which we call ***Monte Carlo ES***, for Monte Carlo with Exploring Starts, is given below

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:
$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
Append G to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

- In Monte Carlo ES, all the returns for each state-action pair are accumulated and averaged, irrespective of what policy was in force when they were observed.
  - ➢ It is easy to see that Monte Carlo ES cannot converge to any suboptimal policy
  - ➢ Stability is achieved only when both the policy and the value function are optimal

Examples 5.3 – Solving Blackjack

- It is straightforward to apply Monte Carlo ES to blackjack
  - ➢ Because the episodes are all simulated games, it is easy to arrange for exploring starts that include all possibilities.
    - ✧ In this case one simply picks the dealer's cards, the player's sum, and whether or not the player has a usable ace, all at random with equal probability.
  - ➢ As the initial policy we use the policy evaluated in the previous blackjack example, that which sticks only on 20 or 21.
  - ➢ The initial action-value function can be zero for all state action pairs.

- Figure 5.2 shows the optimal policy for blackjack found by Monte Carlo ES.
  - ➢ This policy is the same as the "basic" strategy of Thorp with the sole exception of the leftmost notch in the policy for useable ace, which is not present in Thorp's strategy.
  - ➢ We are uncertain of the reason for this discrepancy, but confident that what is shown here is indeed th optimal policy for the version of blackjack we have described

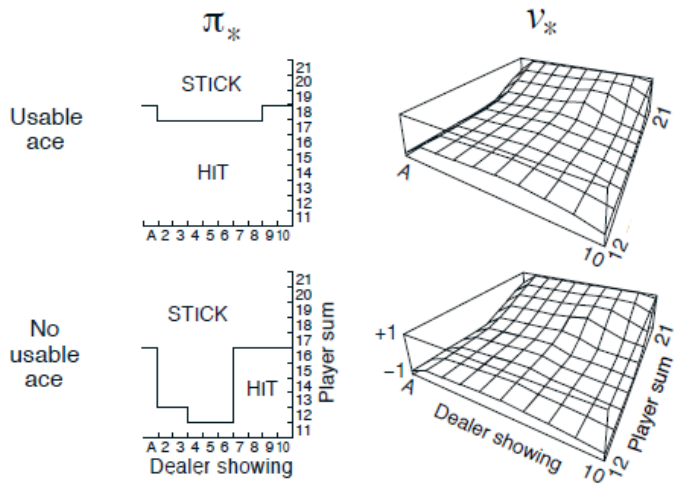


Figure 5.2: The optimal policy and state-value function for blackjack, found by Monte Carlo ES. The state-value function shown was computed from the action-value function found by Monte Carlo ES. ■

4. Monte Carlo Control without Exploring Starts

- How can we avoid the unlikely assumption of exploring starts? The only general way to ensure that all actions are selected infinitely often is for the agent to continue to select them
  - There are two approaches to ensuring this, resulting in what we call ***on-policy*** methods and ***off-policy*** methods.
    - On-policy methods attempt to evaluate or improve the policy that is used to make decisions
    - Whereas off-policy methods evaluate or improve a policy different from that used to generate the data.
  - The Monte Carlo ES method developed above is an example of an on-policy method. In this section we show how an on-policy Monte Carlo control method can be designed that does not use the unrealistic assumption of exploring starts

- In on-policy control methods the policy is generally ***soft***, meaning that $\pi(a|s) > 0$ for all states and all actions, but gradually shifted closer and closer to a deterministic optimal policy
  - The on-policy method we present in this section use $\varepsilon - greedy$ policies, meaning that most of the time they choose an action that has maximal estimated action value, but with probability ε they instead select an action at random
  - The $\varepsilon - greedy$ policies are examples of $\varepsilon - soft$ policies, defined as policies for which $\pi(a|s) \geq \frac{\varepsilon}{|A(s)|}$ for all states and actions, for some $\varepsilon > 0$.
  - Among $\varepsilon - soft$ policies, $\varepsilon - greedy$ policies are in some sense those that are closest to greedy

- The overall idea of on-policy Monte Carlo control is still that of GPI.
  - As in Monte Carlo ES, we use first-visit MC methods to estimate the action-value function for the current policy.
  - Without the assumption of exploring starts, however, we cannot simply improve the policy by making it greedy with respect to the current value function, because that would prevent further exploration of nongreedy actions.
  - Fortunately, GPI does not require that the policy be taken all the way to a greedy policy, only that it be moved *toward* a greedy policy.
  - In our on-policy method we will move it only to an $\varepsilon - greedy$ policy.
  - For any $\varepsilon - soft$ policy π, any $\varepsilon - greedy$ policy with respect to q_π is guaranteed to be better than or equal to π.
  - The complete algorithm is given in the box below

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$
Initialize:
  $\pi \leftarrow$ an arbitrary ε-soft policy
  $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
  $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):
  Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
      Append G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
      $A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
      For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

- That any $\varepsilon - greedy$ policy with respect to q_π is an improvement over any $\varepsilon - soft$ policy π is assured by the policy improvement theorem.
  - Let π' be the $\varepsilon - greedy$ policy.
  - The conditions of the policy improvement theorem apply because for all states

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s,a) \\ &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\varepsilon) \max_a q_\pi(s,a) \qquad (5.2) \\ &\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1-\varepsilon} q_\pi(s,a) \end{aligned}$$

(the sum is a weighted average with nonnegative weights summing to 1, and as such it must be less than or equal to the largest number averaged)

$$\begin{aligned} &= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) - \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + \sum_a \pi(a|s) q_\pi(s,a) \\ &= v_\pi(s). \end{aligned}$$

- Thus, by the policy improvement theorem, $\pi' \geq \pi$. We can show that equality can hold only when both π' and π are optimal among the $\varepsilon - soft$ policies.

5. Off-policy Prediction via Importance Sampling

- All learning control methods face a dilemma: they seek to learn action values conditional on subsequent optimal behavior, but they need to behave non-optimally in order to explore all actions.
  - How can they learn about the optimal policy while behaving according to an exploratory policy?
  - The on-policy approach in the preceding section is actually a compromise – it learns action values not for the optimal policy, but for a near-optimal policy that still explores.
  - A more straightforward approach is to use two policies, one that is learned about and that becomes the optimal policy, and one that is more exploratory and is used to generate behavior.
    - The policy being learned about is called the ***target policy***, and the policy used to generate behavior is called the ***behavior policy***.
    - In this case we say that learning is from data "off" the target policy, and the overall process is termed ***off-policy learning***.

- On-policy methods are generally simpler and are considered first.
  - Off-policy methods require additional concepts and notation, and because the data is due to a different policy, off-policy methods are often of greater variance and are slower to converge
  - On the other hand, off-policy methods are more powerful and general. They include on-policy methods as the special case in which the target and behavior policies are the same. Off-policy methods also have a variety of additional uses in applications.

- In this section we begin the study of off-policy methods by considering the ***prediction*** problem, in which both target and behavior policies are fixed.
  - That is, suppose we wish to estimate v_π or q_π, but all we have are episodes following another policy b, where $b \neq \pi$. In this case, π is the target policy, b is the behavior policy, and both policies are considered fixed and given
  - In order to use episodes from b to estimate values for π, we require that every action taken under π is also taken, at least occasionally, under b
    - That is, we require that $\pi(a|s) > 0$ implies $b(a|s) > 0$.
    - This is called the assumption of ***converge***.
  - It follows from converge that b must be stochastic in states where it is not identical to π.
  - The target policy π, on the other hand, may be deterministic, and, in fact, this is a case of particular interest in control applications
  - In control, the target policy is typically the deterministic greedy policy with respect to the current estimate of the action-value function.
  - This policy becomes a deterministic optimal policy while the behavior policy remains stochastic and more exploratory, for example, an ε-greedy policy.
  - In this section, however, we consider the prediction problem, in which π is unchanging and given

- Almost all off-policy methods utilize ***importance sampling***, a general technique for estimating expected values under one distribution given samples from another.
  - We apply importance sampling to off-policy learning by weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies, called the ***importance-sampling ratio***.
  - Given a starting state S_t, the probability of the subsequent state-action trajectory, A_t, S_{t+1}, A_{t+1}, …, S_T, occurring under any policy π is

$$\begin{aligned}&\Pr\{A_t, S_{t+1}, A_{t+1}, \dots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\ &\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\ &\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),\end{aligned}$$

  - Where p is the state-transition probability function.
  - Thus, the relative probability of the trajectory under the target and behavior policies (the importance-sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}. \tag{5.3}$$

  - The importance sampling ratio ends up depending only on the two policies and the sequence, not the MDP

- Recall that we wish to estimate the expected returns under the target policy, but all we have are returns G_t due to the behavior policy.
  - These returns have the wrong expectation $E[G_t|S_t = s] = v_b(s)$ and so cannot be averaged to obtain v_π.
  - This is where importance sampling comes in.
  - The ratio transforms the returns to have the right expected value

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s). \tag{5.4}$$