Reading of theoretical aspect of reinforcement learning – Chapter 2

Zhe Rao

- The most important feature distinguishing reinforcement learning from other types of learning is that it uses training information that evaluates the actions taken rather than instructs by giving correct action
  - This is what creates the need for active exploration, for an explicit search for good behavior
  - Purely instructive feedback, on the other hand, indicates the correct action to take, independently of the action actually taken
  - In their pure forms, these two kinds of feedback are quite distinct: evaluative feedback depends entirely on the action taken, whereas instructive feedback instructive feedback is independent of the action taken

- In this chapter we study the evaluative aspect of reinforcement learning in a simplified setting, one that does not involve learning to act in more than one situation.
  - This *non-associative* setting is the one in which most prior work involving evaluative feedback has been done, and it avoids much of the complexity of the full reinforcement learning problem.

- The particular non-associative, evaluative feedback problem that we explore is a simple version of the k-armed bandit problem.

1. A k-armed Bandit Problem

- Considering the following learning problem. You are faced repeatedly with a choice among k different options, or actions. After each choice you receive a numerical reward chosen from a stationary probability distribution that depends on the action you selected. Your objective is to maximize the expected total reward over some time period, for example, over 1000 action selections, or time steps.
  - Each of the k actions has an expected or mean reward given that that action is selected; let's call this ***value*** of that action.
    - We denote th action selected on time step t as A_t, and the corresponding reward as R_t.
    - The value then of an arbitrary action a, denoted $q_*(a)$, is the expected reward given that a is selected

$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a].$$

  - If you knew the value of each action, then it would be trivial to solve the k-armed bandit problem: you would always select the action with highest value.
  - We assume that we don't know the action values with certainty, although you may have estimates.
    - We denote the estimated value of action a at time step t as $Q_t(a)$.

- ✧ We would like $Q_{t(a)}$ to be close to $q_*(a)$
- ➢ If you maintain estimates of the action values, then at any time step there is at least one action whose estimated value is greatest.
  - ✧ We call these the *greedy* actions
- ➢ Exploitation is the right thing to do to maximize the expected reward on the one step, but exploration may produce the greater total reward in the long run
  - ✧ Because it is not possible both to explore and to exploit with any single action selection, one often refers to the "conflict" between exploration and exploitation
- ➢ There are many sophisticated methods for balancing exploration and exploitation for particular mathematical formulations of the k-armed bandit and related problems.
  - ✧ However, most of these methods make strong assumptions about stationarity and prior knowledge that are either violated or impossible to verify in most applications and in the full reinforcement learning problem that we consider in subsequent chapters
- ➢ In this book we do not worry about balancing exploration and exploitation in a sophisticated way; we worry only about balancing them at all.

2. Action-value methods

- We begin by looking more closely at methods for estimating the values of actions and for using the estimates to make action selection decisions, which we collectively call ***action-value methods***.
  - ➢ Recall that the true value of an action is the mean reward when that action is selected.
  - ➢ One natural way to estimate this is by averaging the rewards actually received

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}, \qquad (2.1)$$

  - ➢ If the denominator is zero, then we instead define $Q_t(a)$ as some default value, such as 0.
  - ➢ As the denominator goes to infinity, by the law of large numbers, $Q_t(a)$ converges to $q_*(a)$
    - ✧ We call this the ***sample-average*** method for estimating action values

- The simplest action selection rule is the ***greedy actions***
  - ➢ If there is more than one greedy action, then a selection is made among them in some arbitrary way, perhaps randomly.
  - ➢ We write the greedy action selection method as

$$A_t \doteq \arg\max_a Q_t(a), \qquad (2.2)$$

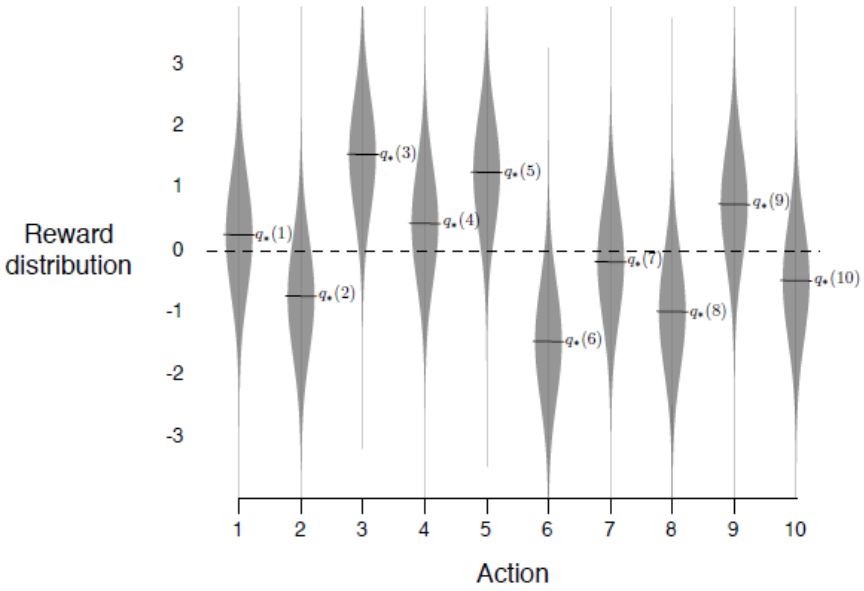
  - ➢ Greedy action selection always exploits current knowledge to maximize immediate reward; it spends no time at all sampling apparently inferior actions to see if they might really be better.

- A simple alternative is to behave greedily most of the time, but every once in a while, say with small probability ε, instead select randomly from among all the actions with equal probability, independently of the action-value estimates.
  - ➢ We call methods using this near-greedy action selection rule ***ε-greedy methods***.

- An advantage of these methods is that, in the limit as the number of steps increases, every action will be sampled an infinite number of times, thus ensuring that all the $Q_t(a)$ converge to their respective $q_*(a)$.
- This of course implies that the probability of selecting the optimal action converges to greater than $1 - \varepsilon$

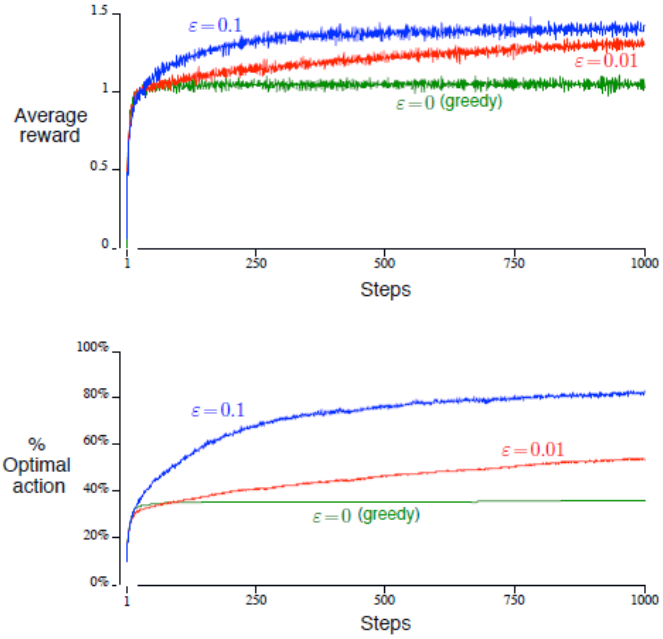
3. The 10-armed Testbed

- To roughly assess the relative effectiveness of the greedy and ε-greedy action-value methods, we compared them numerically on a suite of test problems



- The action values $q_*(a), a = 1, \ldots, 10$ were selected according to a normal (Gaussian) distribution with mean 0 and variance 1.
  - Then, when a learning method applied to that problem selected action A_t at time step t, the actual reward, R_t, was selected from a normal distribution with mean $q_*(A_t)$

- We call this suite of test tasks the ***10-armed testbed***.
  - For any learning method, we can measure its performance and behavior as it improves with experience over 1000 time steps when applied to one of the bandit problems.
  - This makes up on ***run***.
  - Repeating this for 200 independent runs, each with a different bandit problem, we obtained measures of the learning algorithm's average behavior



- All the methods formed their action-value estimates using the sample-average technique (with an initial estimate of 0)
  - The upper graph shows the increase in expected reward with experience.
    - The greedy method improved slightly faster than the other methods at the very beginning, but then leveled off at a lower level.
  - The lower graph shows that the greedy method found the optimal action in only approximately one-third of the tasks.
    - The ε-greedy methods eventually performed better because they continued to explore and to improve their chances of recognizing the optimal action.
    - The $\varepsilon = 0.1$ method explored more, and usually found the optimal action earlier, but it never selected that action more than 91% of the time
    - The $\varepsilon = 0.01$ method improved more slowly, but eventually would perform better tan the $\varepsilon = 0.1$ method on both performance measures shown in the figure.
    - It is also possible to reduce ε over time to try to get the best of both high and low values

- The advantage of ε-greedy over greedy methods depends on the task.
  - For example, suppose the reward variance had been larger, say 10 instead of 1. With noisier rewards it takes more exploration to find the optimal action, and ε-greedy methods should fare even better relative to the greedy method. But if the event is deterministic, exploring doesn't bring any additional reward
  - But even in the deterministic case there is a larger advantage to exploring if we weaken some of other assumptions. For example, suppose the bandit task were nonstationary, that is, the true values of the actions changed over time.

4. Incremental Implementation

- We consider how the average of observed rewards can be estimated or computed in a computationally efficient manner.
  - To simplify notation, we concentrate on a single action. Let R_i now denote the reward received after the ith selection of this action, let Q_n denote the estimate of this action value after it has been selected n-1 times, which we can now write as

$$Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}.$$

- The obvious implementation would be to maintain a record of all the rewards and then perform this computation whenever the estimated value was needed.
  - However, if this is done, then the memory and computational requirements would grow over time as more rewards are seen

- As you might suspect, this is not really necessary. It is easy to devise incremental formulas for updating averages with small, constant computation required to process each new reward.
  - Given Q_n and nth reward, R_n, the new average of all n rewards can be computed by

$$\begin{aligned} Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\ &= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\ &= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\ &= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\ &= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big], \end{aligned} \tag{2.3}$$

    - Which holds even for n=1

- This update rule is of a form that occurs frequently throughout this book. The general form is

$$NewEstimate \leftarrow OldEstimate + StepSize\,\Big[Target - OldEstimate\Big]. \tag{2.4}$$

- Pseudocode for a complete bandit algorithm using incrementally computed sample averages and ε-greedy action selection is shown in the box below.
  - The function *bandit(a)* is assumed to take an action and return a corresponding reward.

A simple bandit algorithm

Initialize, for $a = 1$ to k:
$Q(a) \leftarrow 0$
$N(a) \leftarrow 0$

Loop forever:
$A \leftarrow \begin{cases} \text{argmax}_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$
$R \leftarrow bandit(A)$
$N(A) \leftarrow N(A) + 1$
$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

5. Upper-Confidence-Bound Action Selection

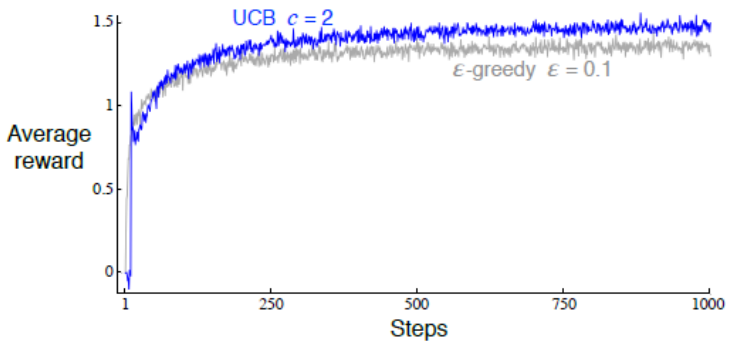
- Exploration is needed because there is always uncertainty about the accuracy of the action-value estimates.
  - The greedy actions are those that look best at present, but some of the other actions may actually be better.
  - ε-greedy action selection forces the non-greedy actions to be tried, but indiscriminately, with no preference for those that are nearly greedy or particularly uncertain.
  - It would be better to select among the non-greedy actions according to their potential for actually being optimal, taking into account both how close their estimates are to being maximal and the uncertainties in those estimates.

$$A_t \doteq \underset{a}{\text{argmax}} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right], \qquad (2.10)$$

    - t denotes the natural logarithm of t
    - $N_t(a)$ denote the number of times that action a has been selected prior to time t
    - $c > 0$ controls the degree of exploration
    - If $N_t(a) = 0$, then a is considered to be a maximizing action

- The idea of this *upper confidence bound* (UCB) action selection is that the square-root term is a measure of the uncertainty or variance in the estimate of a's value.
  - The quantity being mixed over is thus a sort of upper bound on the possible true value of action a, with c determining the confidence level.
  - Each time a is selected the uncertainty is presumably reduced: $N_t(a)$ increments, and, as it appears in the denominator, the uncertainty term decreases.
  - On the other hand, each time an action other than a is selected, t increases but $N_t(a)$ does not
  - The use of natural logarithm means that the increases get smaller over time, but are unbounded; all actions will eventually be selected, but actions with lower value estimates, or that have already been selected frequently, will be selected with decreasing frequency over time.

- Results with UCB on the 10-armed testbed are shown below



- UCB often performs well, as shown here, but is more difficult than ε-greedy to extend beyond bandits to the more general reinforcement learning settings considered in the rest of this book
  - One difficulty is in dealing with nonstationary problems
  - Another difficulty is dealing with large state spaces, particularly when using function approximation as developed in Part II of this book