Notes on *Reinforcement Learning: An Introduction, Second Edition* by Richard S. Sutton and Andrew G. Barto

Luke Renchik

April 14, 2024

1 Session Outline

1.1 Sections Covered

- 2.1 - A k-armed Bandit Problem
- 2.2 - Action Value Methods
- 2.3 - The 10-armed Testbed
- 2.4 - Incremental Implementation
- 2.5 - Tracking a Non-stationary Problem

2 Chapter 2: Multi-armed Bandits

2.1 Important Notes

The distinguishing feature for reinforcement learning from other types of learning is that it uses training information that *evaluates* the actions taken rather than *instructs* correct actions. This chapter focuses on the evaluative aspect of reinforcement learning in a simple setting. This *non-associative* setting avoids the complexity of the full reinforcement learning problem, this case enables us to see how evaluative feedback differs and can be combined with instructive feedback.

The K Armed Bandit Problem: An agent is faced repeatedly with a choice of k different actions, after each one there is a numerical reward chosen from a stationary probability distribution that depends on the action selected (see Figure 1). The objective is to maximize the expected total reward over some trial period, like 1000 actions, or *time steps*.

This bandit problem can be thought of as k slot machines placed in front of the agent, each one having an unknown expected value range, the agent's job is to maximize the winnings/reward from these machines.

Greedy Actions - The action with the highest expected value at any time step, selecting this action would be *exploiting* the current knowledge of the value of the actions.

Exploring - Selecting one of the *non-greedy* actions, ideally on an action with a low confidence of its value range.

ϵ-greedy methods become more effective as the reward signals begin to have more noise involved. In completely static situations where reward signals are known, a purely greedy algorithm will be the most effective. This only holds true for the most rigid of cases, imagine a case where the values are non-stationary, then exploration would be necessary to understand the shift in value gained over time.

Non-Stationary Problems - Reward probabilities change over time, in such a case it makes more sense to give more weight to recent rewards than long-past rewards.

2.2 Key Concepts

Action-Value Methods

The *sample-average* method for estimating action values.

$$Q_t(a) \doteq \frac{\text{sum of rewards when a taken prior to t}}{\text{number of times a taken prior to t}} = \frac{\sum_{i=1}^{t-1} R_i * A_{i=a}}{\sum_{i=1}^{t-1} 1_{A_{i=a}}}$$

Where $1_{predicate}$ denotes the random variable if *predicate* is true and 0 if it is not. If the denominator is 0, set $Q_{t(a)}$ to some default value.

Incremental Implementation

How to utilize sample=averages in a computationally efficient manner (constant memory, constant

per-time-step).

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

$= Q_n + \frac{1}{n}[R_n - Q_n]$

This implementation requires only memory for Q_n and n.

NewEstimate ← OldEstimate + StepSize[Target - OldEstimate].

Non-Stationary Problems

Exponential Recency Weighted Average:

$Q_{n+1} \doteq Q_n + \alpha[R_n = Q_n]$ where α is some constant (0,1]

$= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$

2.3 Important Formulas

Mean Reward = Value
Action Select on Timestep t = A_t
Reward on Timestep t = R_t
Value of an Arbitrary Action = $q_*(a)$
Expected Reward Given a is selected: $q_*(a) \doteq \mathbb{E}[R_t|A_t = a]$
The estimated value of action a at time step t as $Q_t(a)$
Want $Q_t(a)$ to be close to $q_*(a)$
Greedy Action Selection - $A_t \doteq argmax Q_t(a)$

2.4 Personal Insights

The "simple" models of adjusting step-size and tracking non-stationary problems despite being conceptually straightforward require a great deal of care and consideration to express mathematically and implement in code.

2.5 Exercises

Exercise 2.1 - In ϵ-greedy action selection, for the case of two actions and $\epsilon = 0.5$, what is the probability that the greedy action is selected? Half of the time.

Exercise 2.2: Bandit Example - Consider a k-armed bandit problem with k = 4 actions, denoted 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ϵ-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is $A_1 = 1, R_1 = -1, A_2 = 2, R_2 = 1, A_3 = 2, R_3 = -2, A_4 = 2, R_4 = 2, A_5 = 3, R_5 = 0$. On some of these time steps the ϵ case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which times steps could this possibly have occurred?

With the assumption that the expected value is utilizing the *sample-average* that have a default value of 0, A_4 and A_5 are ϵ cases. This is apparent because after R_3 returns -2, the expected value for A = 2 is below 0, thus selecting it again would have to be an exploratory action, if k = 3 and k = 4 are both expected values of 0. Finally the same logic can be applied once the expected value of k = 2 rises above 1 again due to the result of R_4.

Exercise 2.3 - In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

If we are inspecting the case in which trials approach infinity and we are able to converge on the true expected value of each action, $\epsilon = .01$ will have a Optimal Action % of 99% and beat out $\epsilon = 0.1$ by 9%. In the case where we are not able to reach a convergence in a reasonable number of trials the $\epsilon = 0.1$ will outperform, reaching an Optimal Action % of 80% in 1000 steps, 22% higher than $\epsilon = 0.01$.

Exercise 2.4 - If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters?

The weighting on each prior reward is exponentially decreasing due to the step size parameters having an exponent of n. A larger step size parameter accelerates this exponential decay of the initial expected values.

Exercise 2.5 (Programming) - Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for non-stationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (say by adding a normally distributed increment with mean 0 and standard deviation 0.01 to all the $q_*(a)$ on each step). Prepare plots like Figure 2.2 for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\epsilon = 0.1$ and longer runs, say of 10,000 steps.

This is a complex problem that will require careful consideration to solve, code will be on my github.

3 Closing Thoughts

It seems we are getting into the meat and potatoes of this thing here, I suppose this program where I build my first model that utilizes sample-average and recency weighted averages will reveal much about the complexity of these problems and implementation level details. I think understanding these models well will provide a much needed foundation when progressing to the more tailored methods of learning.

4 Appendix

4.1 General Notation

Random Variables - Upper Case
Instantiated Variables - Lower Case
At time step t:
State = S_t
Action = A_t
Reward = R_t
The specific values would be s, a, r.
Value functions - lower case (e.g., v_π)
Tabular Estimates - upper case (e.g., Q_t(s,a))
Approximate value functions are deterministic functions of the random parameters and written in lowercase ($\hat{v}(s, \mathbf{w_t}) \approx v_\pi(s)$)
Bold Letters represent vectors
Uppercase Bold is used for matrices
Transition for State Function - $p(s', r|s, a)$
$\doteq$ - Equality relationship that is true by definition
$\propto$ - Proportional To
$\Pr\{X = x\}$ - Probability that a random variable X takes on the value x
$X \sim p$ - random variable X selected from distribution $p(x) \doteq \Pr\{X = x\}$
$\mathbb{E}[\mathbb{X}]$ - Expectation of a random variable X, i.e., $\mathbb{E}[\mathbb{X}] \doteq \sum_x p(x)x$
$\ln x$ - Natural Log of X
e^x, exp(x) - The base of the natural log, e carried to the power x; $\mathrm{e}^{\ln x} = \mathrm{x}$
$\mathbb{R}$ - Set of real numbers
$f : X \to y$ - Function f from elements of set X to elements of set y
$\leftarrow$ - Assignment
ϵ - Probability of takin a random action in an ϵ greedy policy.
α, β - step-size parameters
γ - Discount Rate Parameter λ - decay-rate parameter for eligibility traces
$\mathbb{I}_{predicate}$ = indicator function ($\mathbb{I}_{predicate} \doteq 1$ if the *predicate* is true, else 0)

4.2 Markov Decision Process Notation

TODO: Add the MDP Notation to Template.

4.3 Important Figures

Figure 1 shows an example image.

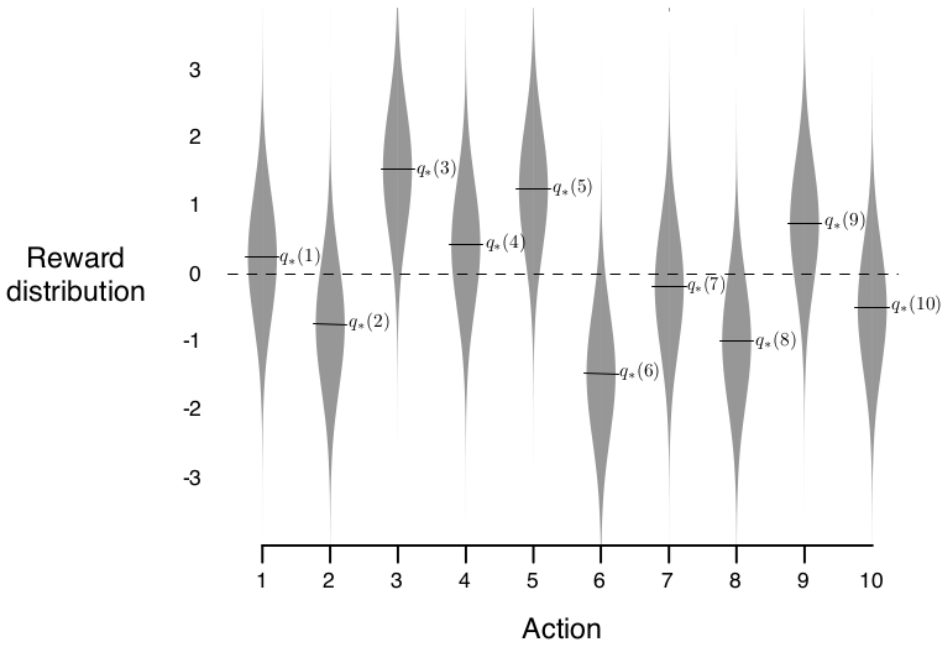

Figure 1: An example of a 10-Armed Testbed

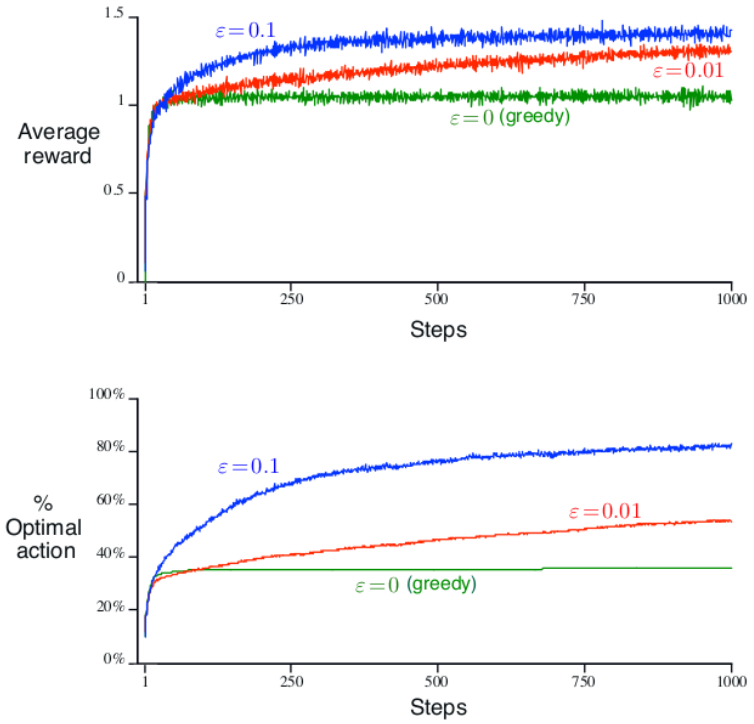

Figure 2.2: Average performance of ε-greedy action-value methods on the 10-armed testbed. These data are averages over 2000 runs with different bandit problems. All methods used sample averages as their action-value estimates.

Figure 2: The impact of different exploratory action rates

4.4 Code Snippets

A simple bandit algorithm

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

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

Figure 3: Bandit Algorithm Pseudocode