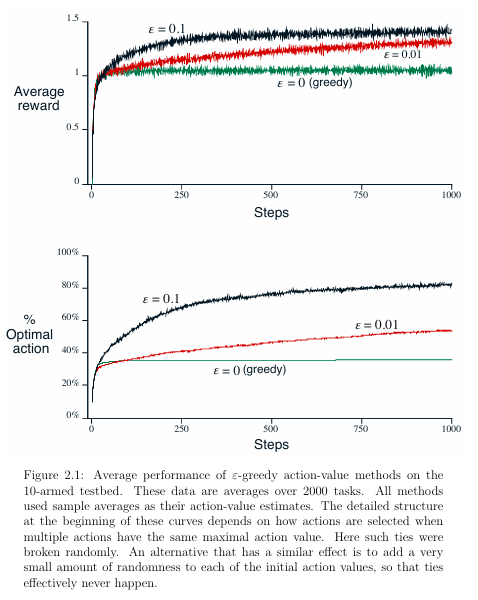
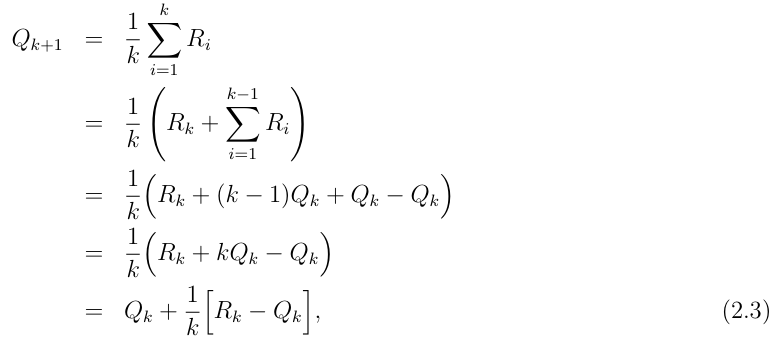
**Chapter 2 Multi-arm Bandits**

2.2 Action-Value Methods



2.3 Incremental Implementation (Tracking a Stationary Problem)

Given this average and a kth reward for the action, Rk, then the average of all k rewards can be computed by:



which holds even for k = 1, obtaining Q2 = R1 for arbitrary Q1. This simple mentation requires memory only for Qk and k, and only the small computation (2.3) for each new reward.

The update rule (2.3) is of a form that occurs frequently throughout this book. The general form is:



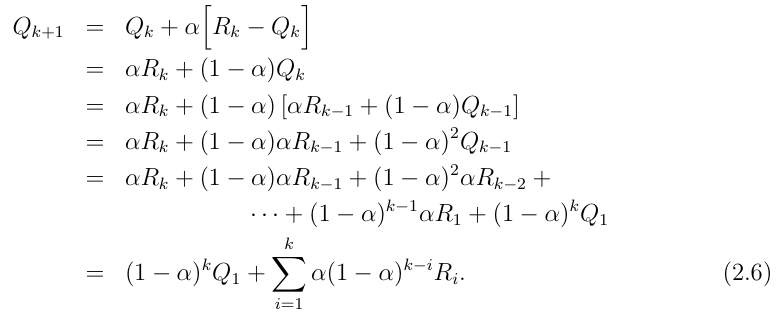
We sometimes use the informal shorthand = 1 k to refer to this case, leaving the dependence of k on the action implicit.

2.4 Tracking a Nonstationary Problem

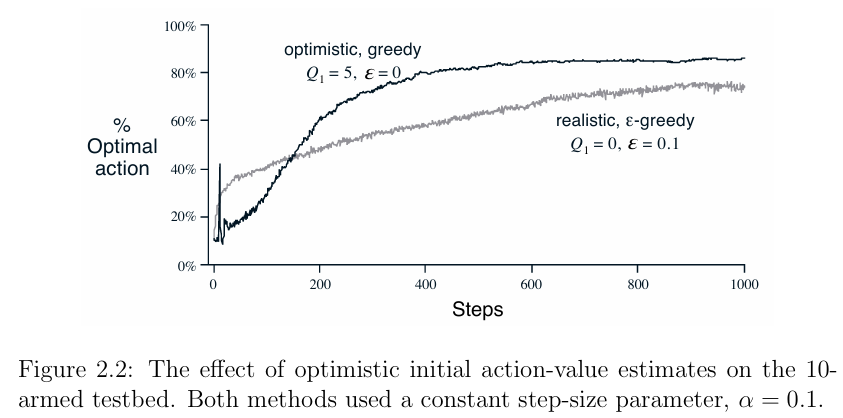
the incremental update rule (2.3) for updating an average Qk of the k 1 past rewards is modified to be:



where the step-size parameter is constant. This results in Qk+1 being a weighted average of past rewards and the initial estimate Q1:



2.5 Optimistic Initial Values



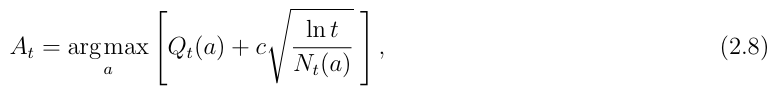
For the sample-average methods, the bias disappears once all actions have been selected at least once, but for methods with constant e , the bias is permanent, though decreasing over time as given by (2.6).

Figure 2.2 shows the performance on the 10-armed bandit testbed of a greedy method using Q1(a) = +5, for all a. For comparison, also shown is an-greedy method with Q1(a) = 0. Initially, the optimistic method performs worse because it explores more, but eventually it performs better because its exploration decreases with time. We call this technique for encouraging exploration optimistic initial values.

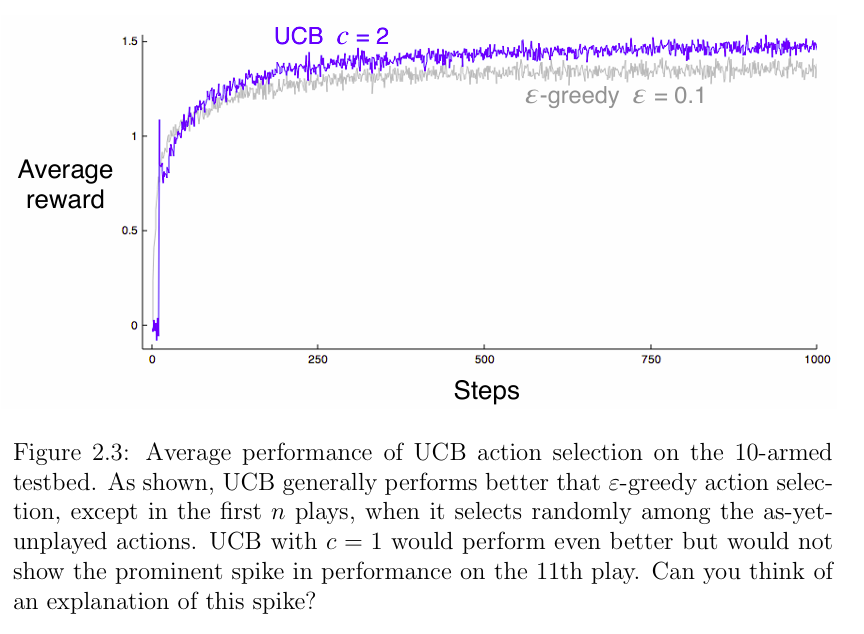
2.6 Upper-Con dence-BoundAction Selection

Exploration is needed because the estimates of the action values are uncertain. The greedy actions are those that look best at present, but some of the other actions may actually be better.-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. One e ective way of doing this is to select actions as:

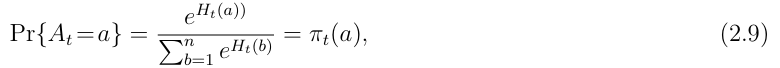


where lnt denotes the natural logarithm of t (the number that e 271828 would have to be raised to in order to equal t), and the number c > 0 controls the degree of exploration. If Nt(a) = 0, then a is considered to be a maximizing action.

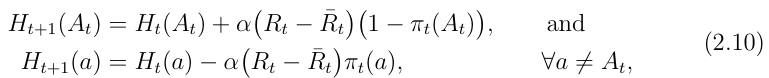


2.7 Gradient Bandits

we consider learning a numerical preference Ht(a) for each action a. The larger the preference, the more often that action is taken, but the preference has no interpretation in terms of reward.



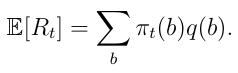
There is a natural learning algorithm for this setting based on the idea of stochastic gradient ascent. On each step, after selecting the action At and receiving the reward Rt, the preferences are updated by:

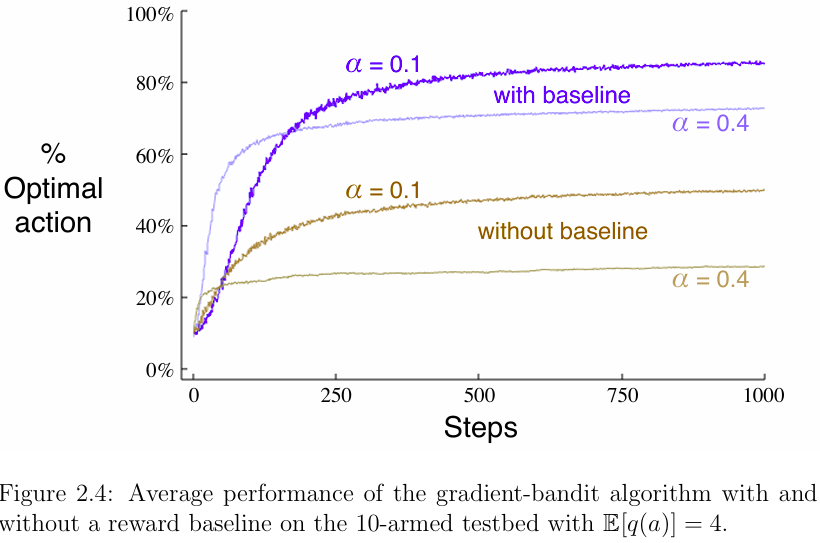


In exact gradient ascent, each preference Ht(a) would be incrementing proportional to the increments exact on performance:



where the measure of performance here is the expected reward:





2.8 Associative Search (Contextual Bandits)

**Multi-Armed Bandit with ε-Greedy Strategy**

**Explanation:**

1. **Bandit Class**:
   * Initializes the number of arms and their true values.
   * Contains methods to pull an arm and update estimates.
2. **Epsilon Greedy Agent Class**:
   * Chooses an arm to pull based on the ε-greedy policy.
3. **Simulate Function**:
   * Runs the simulation for a specified number of steps, pulling arms and updating estimates.
4. **Parameters**:
   * You can adjust the number of arms, steps, and the epsilon value for exploration.

The multi-armed bandit program serves several purposes, particularly in the context of reinforcement learning and decision-making. Here are some key uses:

### 1. ****Exploration vs. Exploitation****

* The program illustrates the fundamental trade-off in reinforcement learning: balancing exploration (trying new options) and exploitation (choosing the best-known option).

### 2. ****Algorithm Testing****

* It provides a platform for testing different strategies (like ε-greedy) for action selection, which can be compared against other methods (like UCB, Thompson Sampling).

### 3. ****Performance Evaluation****

* By running simulations, you can evaluate the performance of the bandit algorithm, such as how quickly it converges to the optimal arm and how well it balances exploration and exploitation.

### 4. ****Application in Real-World Scenarios****

* The multi-armed bandit approach is applicable in various fields, such as:
  + **Online Advertising**: Selecting which ad to display to maximize click-through rates.
  + **A/B Testing**: Determining the best version of a webpage or product by comparing different variations.
  + **Recommendation Systems**: Choosing which items to recommend to users based on their preferences.

### 5. ****Educational Tool****

* It serves as an educational example for understanding the concepts of reinforcement learning, helping learners grasp the mechanics of bandit algorithms and their implementations.

### 6. ****Dynamic Decision-Making****

* The program can be adapted to solve dynamic decision-making problems where the environment changes over time, allowing the agent to continuously learn and adapt its strategy.

### Summary

Overall, the program is a practical demonstration of reinforcement learning principles, providing insights into decision-making processes across various domains.