CHAPTER 5

ONLINE LEARNING

In this chapter we turn to the problem where we have to learn as we go. This means that we no longer have the luxury of making mistakes within our experimental budget before we have to decide which design to implement. Now, every decision is an implementation decision where we have to live with the consequences, but simultaneously learning as we go.

On the surface, the difference between the offline learning problem we introduced in chapter 3, and the online learning problem we address in this chapter, is the objective function. With the offline problem, our objective could be stated as

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\mu_{x^{\pi,N}}, \tag{5.1}$$

where $x^{\pi,N}$ is the alternative we think is best given our estimates $\bar{\mu}_x^N$, given by

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^N.$$

This means we are only interested in the performance $\mu_{x^{\pi,N}}$ of the final design $x^{\pi,N}$ that we learned by following learning policy π after exhausting our experimental budget N.

For an online learning problem, we have to learn as we go. So, if $X^\pi(S)$ is our policy that returns a decision $x^n = X^\pi(S^n)$ when our belief state is S^n. We then receive a

reward W^{n+1}. This means we have to find the policy that solves

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} W_{x^n}^{n+1}, \tag{5.2}$$

where $x^n = X^\pi(S^n)$. If we assume that we have access to the true mean μ_x for the purpose of testing policies (as we did in equation (5.1)), we could write our objective as

$$\max_{\pi} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \mu_{X^\pi(S^n)}. \tag{5.3}$$

However we calculate the objective, we see that our offline problem focuses on the final (or terminal) reward or performance, while our online problem focuses on the cumulative reward.

Viewed this way, the difference between offline (terminal reward) and online (cumulative reward) seems modest, and for some classes of policies, it is. However, online learning problems have a long and rich history where it has evolved as the *multiarmed bandit problem*.

The multi-armed bandit problem (or MAB, as it is often known) is a venerable topic in optimal learning, and has inspired some of the pioneering work in the field. The story that was originally used to motivate the problem (and gave the problem its name) is not really an important application, but is useful for understanding the basic idea behind the problem. The term "one-armed bandit" refers to a slot machine operated by pulling a lever (or "arm") on its side. A multi-armed bandit, then, is a collection of slot machines, each with a different winning probability (or different average winnings).

Suppose that we have M slot machines each with a different but unknown reward distribution (yes, that is the story). If we play slot machine x at time n, we receive random winnings W_x^{n+1}. Suppose that the expected value of these winnings, given by μ_x, is unknown. We would like to estimate μ_x, but the only way to do this is by putting money into the machine and collecting a random observation. For this reason, we must balance our desire to find the slot machine with the highest value with our desire to achieve good results on every play.

There are numerous examples of online learning problems. Some examples are

---

EXAMPLE 5.1

You need to find the best path to work, but the only way to evaluate a path is to actually try it.

EXAMPLE 5.2

A physician has to find the best diabetes medication for a patient, but the only way to see how the patient responds is to try the medication on the patient.

■ **EXAMPLE 5.3**

A company needs to place bids to have its ad appear on Google. The company has to try different bids to find the one that strikes the best balance between cost and response.

■ **EXAMPLE 5.4**

An internet movie provider has to decide which movies to show to maximize the likelihood that a user logging into her account will pick one of the movies to watch.

■ **EXAMPLE 5.5**

An internet retailer has to experiment with different prices for a product to find the price that maximizes revenue.

Both offline and online learning problems require balancing exploration and exploitation, but the tradeoff is more apparent with online problems.

As early as 1970, we have known that we could characterize an optimal policy by setting up Bellman's equation

$$V^n(S^n) = \max_x \left(\bar{\mu}_x^n + \mathbb{E}_W V^{n+1}(S^{n+1})\right),\ n = 0, \ldots, N-1 \tag{5.4}$$

$$V^N(S^N) = \max_x \bar{\mu}_x^N, \tag{5.5}$$

where $S^{n+1} = S^M(S^n, x, W_x^{n+1})$. Here, $S^n = (\bar{\mu}_x^n, \beta_x^n)_{x \in \mathcal{X}}$ is the belief state, x is a decision of which "arm" to try, W_x^{n+1} is what we learn, and $S^M(S, x, W)$ captures the equations for updating the belief model. The problem is that if we have a normally distributed belief model and M alternatives, then S has $2M$ continuous dimensions. As a result, solving equation (5.5) is completely intractable.

Given this, it was quite a breakthrough when J.C. Gittins found, in 1974, that instead of solving a $2M$ dimensional problem across all M alternatives, it was possible to find an optimal policy by solving M individual dynamic programs. Each of these dynamic programs returned an index, later known as the *Gittins index*, which required that we simply find the alternative with the highest index. This became known as an *index policy*. This spawned a steady stream of papers looking for optimal index policies for variations of the basic bandit problem.

The development of Gittins indices generated considerable enthusiasm in the applied probability community, although they still remained a computational challenge in that while they were computable, they were not easy to compute. We illustrate the basic idea of Gittins indices, which also serves to highlight the relative complexity, especially compared to the policies we have seen so far for the offline problem.

A completely different approach was developed by Lai and Robbins in 1985, who found that a very simple idea called upper confidence bounding (which we first saw in chapter 3) produced a policy where it is possible to bound how often it chose the wrong arm. While this does not mean that UCB policies (as they are known) are

optimal, it provided evidence that they might be quite good. In addition, they are trivial to compute, which makes them attractive for high-speed internet applications.

Reflecting the combined contributions of these two communities, the multi-armed bandit problem has attracted a substantial following that has grown past the core niche communities that developed these ideas. In fact, these communities have collectively spawned the study of a wide range of "bandit problems" that has introduced novel problem variations that have been overlooked by other communities working on stochastic optimization problems.

We are ultimately going to see that there are many policies that were designed originally for online learning problems, but which still need to be tuned, and which can be tuned for either online or offline applications. We are going to cover three major classes of policies:

Excitation policy Excitation policies use a simple rule, such as a decision that appears to be best, and introduces some noise to encourage exploration.

Upper confidence bounding UCB policies are trivial to compute and enjoy bounds that suggest that they should work well, although they still have to be tuned in practice.

Gittins indices Gittins indices determine the best decision based on the value of information from the decision over the entire horizon, computed using dynamic programming. These are more difficult to compute, but provide a rare example of an optimal policy when certain assumptions are satisfied.

Lookahead policies While Gittins indices using dynamic programming to compute the value of being in a belief state, another approach is to try to model decisions and learning in the future to help make a decision now. We have already seen this in section 1.6 when we used decision trees to model decisions and information over some horizon. In this chapter we are going to illustrate some different strategies for approximating the future.

5.1 AN EXCITATION POLICY

Consider the problem of pricing a product where we have to set a price x in the hope of maximizing revenue. We might assume that we can approximate the demand $D(p)$ as a function of price p over a reasonable range of prices using a linear function

$$D(p|\theta) = \theta_1 - \theta_2 p.$$

We would like to maximize the revenue $R(p) = pD(p)$. If we knew $\theta = (\theta_1, \theta_2)$ we would just take the derivative of $R(p)$, set it equal to zero and solve for the optimal price, giving us

$$p^* = \frac{\theta_1}{2\theta_2}.$$

The problem is that we do not know θ. Let $\bar{\theta}^n$ be our estimate of θ after n samples. This means that we would compute our price p^n using

$$p^n = \frac{\bar{\theta}_1^n}{2\bar{\theta}_2^n}.$$

Further assume that after setting a price p^n we observe a revenue $\hat{R}^{n+1}$. The revenue $\hat{R}^{n+1}$ presumably comes from

$$\hat{R}^{n+1} = \theta_1 - \theta_2 p^n + \varepsilon^{n+1},$$

where θ_1 and θ_2 are the true (but unknown) parameters, and ε^{n+1} represents noise in the process of generating revenue.

Since we only have an estimate $\bar{\theta}^n$, we would expect to use the data $(\hat{R}^{n+1}, p^n)$ to obtain an updated estimate $\bar{\theta}^{n+1}$ (we learn how to do this with a linear belief model in chapter 7).

This seems reasonable, except that this method will tend to test prices in a fairly narrow range. If we think $\bar{\theta}^n$ is a good estimate of the true θ, this is fine. But what if we do not have a lot of confidence in this estimate? Sampling points in a narrow range is not a good way of generating data to estimate a model.

A simple strategy that is widely used in some communities is to simply add a noise term, ε^p, to induce exploration. Assume that $\varepsilon^p \sim N(0, \sigma_\epsilon^2)$, where the standard deviation σ_ϵ is a tunable parameter. We refer to the resulting policy as an *excitation policy* which is written as

$$P^{exc}(S^n|\sigma_\epsilon) = \frac{\bar{\theta}_1^n}{2\bar{\theta}_2^n} + \varepsilon^{p,n+1}. \tag{5.6}$$

We can think of the rule in equation (5.6) as a type of policy where we use some formula for making a decision, such as choosing the decision that appears to be best, and then adding noise (of course, this only works if the decision is continuous). The variance σ_ϵ^2 is a tunable parameter, where larger values encourage more exploration.

5.2 UPPER CONFIDENCE BOUNDING

A class of policies that has received considerable interest is known as *upper confidence bounding* or UCB policies. These policies are quite simple to implement, and different variants have been developed using this approach for many types of reward distributions. For example, imagine we have a problem where all the rewards are in the interval $[0, 1]$, e.g. if we are using a beta-Bernoulli model. In this setting, one possible UCB policy defines the index of alternative x to be

$$\nu_x^{UCB1,n} = \bar{\mu}_x^n + \sqrt{\frac{2\log n}{N_x^n}}, \tag{5.7}$$

where N_x^n is the number of times we have played arm x up to and including time n. The policy is somewhat analogous to interval estimation: we take our current

estimate of μ_x and add an uncertainty bonus. Just as in interval estimation, this particular uncertainty bonus represents the half-width of a confidence interval. In a sense, (5.7) represents a probabilistic guess of the largest possible value that μ_x could realistically take on. We choose to measure, not the alternative with the highest estimated value, but rather the alternative that *could* potentially have the largest value.

The policy in (5.7) is geared toward the specific case where the rewards are in $[0, 1]$. A UCB policy designed for the normal-normal model defines the index of x as

$$\nu_x^{UCB1-Normal,n} = \bar{\mu}_x^n + 4\sigma_W \sqrt{\frac{\log n}{N_x^n}}. \tag{5.8}$$

There are also many other versions in the literature for both beta-Bernoulli and normal-normal problems, but (5.7) and (5.8) are two of the most prominent. Problems with bounded rewards turn out to be particularly attractive for the UCB approach because, in these settings, a particular proof technique can be applied to create UCB policies with provable bounds on the regret.

There is also a UCB policy for the gamma-exponential model, where W_x follows an exponential distribution with parameter λ_x, and our beliefs about λ_x are represented by a gamma distribution with parameters a_x^n and b_x^n. In this case, we compute

$$\nu_x^{UCB-Exp,n} = \frac{b_x^n}{a_x^n - 1} + \theta \min\left(\sqrt{2\frac{\log n + 2\log\log n}{N_x^n}}, 1\right), \tag{5.9}$$

with θ being a tunable parameter.

The reasons why UCB policies have attracted attention is a) they are easy to compute and b) because they have an optimality property of sorts. If we are able to make N experiments, and we make them by following a UCB-type policy, then the average number of times we will play a sub-optimal alternative (an alternative with $\mu_x < \max_{x'} \mu_{x'}$) can be bounded above by $C \log N$, where C is some constant. This is known as a *regret bound*. Thus, the number of times that we choose any particular sub-optimal alternative is on the order of $\log N$. It has been proven that this is the best possible bound (up to the choice of C) on the number of times a sub-optimal machine is played. Each of the UCB policies given above have this property.

The structure of the policies (5.7), (5.8) and (5.9) all have the general form

$$X^{UCB}(S^n|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta B_x^{unc,n}\right), \tag{5.10}$$

where $B_x^{unc,n}$ is often referred to as the *uncertainty bonus*. The trick is in the design of this bonus. We are looking for a function that starts large (which encourages the logic to try an alternative) and shrinks as the number of times we test an alternative grows. The trick is in designing this bonus. For example, we could use

$$B_x^{unc,n} = 4\sigma_W \sqrt{\frac{\log n}{N_x^n}}, \tag{5.11}$$

or

$$B_x^{unc,n} = 4\sigma_W \frac{\log n}{N_x^n}. \tag{5.12}$$

Both of these exhibit the same behavior of starting with a large bonus, declining as we test alternative x. So why do we use (5.11) instead of (5.12)? The answer is simply that (5.11) enjoys a tighter regret bound than (5.12). At the same time, the coefficient $4\sigma_W$ is typically discarded in practice, and replaced with a tunable parameter, producing a policy of the form

$$X^{UCB}(S^n|\theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB}\sqrt{\frac{\log n}{N_x^n}}\right). \tag{5.13}$$

We have simply replaced the coefficient $4\sigma^W$ with a tunable parameter θ^{UCB}, which is typically tuned in an ad hoc way. We return to the issue of tuning later.

5.3 GITTINS INDICES

The idea behind Gittins indices works as follows. Assume that we are playing a single slot machine, and that we have the choice of continuing to play the slot machine or stopping and switching to a process that pays a reward r . If we choose not to play, we receive r, and then find ourselves in the same state (since we did not collect any new information). If we choose to play, we earn a random amount W, plus we earn $\mathbb{E}\{V(S^{n+1}, r)|S^n\}$, where S^{n+1} represents our new belief state resulting from our observed winnings. For reasons that will become clear shortly, we write the value function as a function of the state S^{n+1} and the stopping reward r.

The value of being in state S^n, then, can be written as

$$V(S^n, r) \quad = \quad \max\left[r + \gamma V(S^n, r), \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\}\right]. \tag{5.14}$$

The first choice represents the decision to receive the fixed reward r, while in the second, we get to observe W^{n+1} (which is random when we make the decision). When we have to choose x^n, we will use the expected value of our return if we continue playing, which is computed using our current belief state. For example, in the Bayesian normal-normal model, $\mathbb{E}\{W^{n+1}|S^n\} = \bar{\mu}^n$, which is our estimate of the mean of W given what we know after the first n experiments.

If we choose to stop playing at iteration n, then S^n does not change, which means we earn r and face the identical problem again for our next play. In this case, once we decide to stop playing, we will never play again, and we will continue to receive r (discounted) from now on. The infinite horizon, discounted value of this reward is $r/(1-\gamma)$. This means that we can rewrite our optimality recursion as

$$V(S^n, r) = \max\left[\frac{r}{1-\gamma}, \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\}\right], \tag{5.15}$$

Here is where we encounter the magic of Gittins indices. We compute the value of r that makes us indifferent between stopping and accepting the reward r (forever), versus continuing to play the slot machine. That is, we wish to solve the equation

$$\frac{r}{1-\gamma} = \mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r)\middle|\, S^n\right\} \tag{5.16}$$

for r. The Gittins index $I^{Gitt,n}$ is the particular value of r that solves (5.16). This index depends on the state S^n. If we use a Bayesian perspective and assume normally distributed rewards, we would use $S^n = (\bar{\mu}^n, \beta^n)$ to capture our distribution of belief about the true mean μ. If we use a frequentist perspective, our state variable would consist of our estimate $\bar{\theta}^n$ of the mean, our estimate $\hat{\sigma}^{2,n}$ of the variance, and the number N^n of observations (this is equal to n if we only have one slot machine).

If we have multiple slot machines, we consider every machine separately, as if it were the only machine in the problem. We would find the Gittins index $I_x^{Gitt,n}$ for every machine x. Gittins showed that, if $N \to \infty$, meaning that we are allowed to make infinitely many experiments, it is optimal to play the slot machine with the highest value of $I_x^{Gitt,n}$ at every time n. Notice that we have not talked about how exactly (5.16) can be solved. In fact, this is a major issue, but for now, assume that we have some way of computing $I_x^{Gitt,n}$.

Recall that, in ranking and selection, it is possible to come up with trivial policies that are asymptotically optimal as the number of experiments goes to infinity. For example, the policy that measures every alternative in a round-robin fashion is optimal for ranking and selection: if we have infinitely many chances to measure, this policy will measure every alternative infinitely often, thus discovering the true best alternative in the limit. However, in the multi-armed bandit setting, this simple policy is likely to work extremely badly. It may discover the true best alternative in the limit, but it will do poorly in the early iterations. If $\gamma < 1$, the early iterations are more important than the later ones, because they contribute more to our objective value. Thus, in the online problem, it can be more important to pick good alternatives in the early iterations than to find the true best alternative. The Gittins policy is the only policy with the ability to do this optimally.

5.3.1 Gittins indices in the beta-Bernoulli model

The Gittins recursion in (5.15) cannot be solved using conventional dynamic programming techniques. Even in the beta-Bernoulli model, one of the simplest learning models we have considered, the number of possible states S^n is uncountably infinite. In other models like the normal-normal model, S^n is also continuous. However, in some models, the expectation in the right-hand side of (5.15) is fairly straightforward, allowing us to get a better handle on the problem conceptually.

Let us consider the beta-Bernoulli model for a single slot machine. Each play has a simple 0/1 outcome (win or lose), and the probability of winning is ρ. We do know this probability exactly, so we assume that ρ follows a beta distribution with parameters α^0 and β^0. Recall that the beta-Bernoulli model is conjugate, and the updating equations are given by

$$\begin{aligned} \alpha^{n+1} &= \alpha^n + W^{n+1}, \\ \beta^{n+1} &= \beta^n + \left(1 - W^{n+1}\right), \end{aligned}$$

where the distribution of W^{n+1} is Bernoulli with success probability ρ. After n plays, the distribution of ρ is beta with parameters α^n and β^n. The belief state for a

single slot machine is simply $S^n = (\alpha^n, \beta^n)$. Consequently,

$$\begin{aligned}\mathbb{E}\left(W^{n+1} \,|\, S^n\right) &= \mathbb{E}\left[\mathbb{E}\left(W^{n+1} \,|\, S^n, \rho\right) \,|\, S^n\right] \\ &= \mathbb{E}\left(\rho \,|\, S^n\right) \\ &= \frac{\alpha^n}{\alpha^n + \beta^n}.\end{aligned}$$

Then, writing $V\left(S^n, r\right)$ as $V\left(\alpha^n, \beta^n, r\right)$, we obtain

$$\begin{aligned}\mathbb{E}\left\{W^{n+1} + \gamma\, V(S^{n+1}, r) \middle| S^n\right\} &= \frac{\alpha^n}{\alpha^n + \beta^n} + \gamma \frac{\alpha^n}{\alpha^n + \beta^n} V\left(\alpha^n + 1, \beta^n, r\right) \\ &\quad + \gamma \frac{\beta^n}{\alpha^n + \beta^n} V\left(\alpha^n, \beta^n + 1, r\right). \qquad (5.17)\end{aligned}$$

For fixed α and β, the quantity $V\left(\alpha, \beta, r\right)$ is a constant. However, if the observation W^{n+1} is a success, we will transition to the belief state $(\alpha^n + 1, \beta^n)$, and if it is a failure, the next knowledge will be $(\alpha^n, \beta^n + 1)$. Given S^n, the conditional probability of success is $\frac{\alpha^n}{\alpha^n + \beta^n}$.

From (5.17), it becomes clear why Gittins indices are difficult to compute exactly. For any value of r and any α, β, we need to know $V\left(\alpha + 1, \beta, r\right)$ as well as $V\left(\alpha, \beta + 1, r\right)$ before we can compute $V\left(\alpha, \beta, r\right)$. But there is no limit on how high α and β are allowed to go. These parameters represent roughly the tallies of successes and failures that we have observed, and these numbers can take on any integer value if we assume an infinite horizon.

However, it is possible to compute $V\left(\alpha, \beta, r\right)$ approximately. For all α and β such that $\alpha + \beta$ is "large enough," we could assume that $V\left(\alpha, \beta, r\right)$ is equal to some value, perhaps zero. Then, a backwards recursion using these terminal values would give us approximations of $V\left(\alpha, \beta, r\right)$ for small α and β.

The quality of such an approximation would depend on how many steps we would be willing to perform in the backwards recursion. In other words, the larger the value of $\alpha + \beta$ for which we cut off the recursion and set a terminal value, the better. Furthermore, the approximation would be improved if these terminal values were themselves as close to the actual value functions as possible.

One way of choosing terminal values is the following. First, fix a value of r. If $\alpha + \beta$ is very large, it is reasonable to suppose that

$$V\left(\alpha, \beta, r\right) \approx V\left(\alpha + 1, \beta, r\right) \approx V\left(\alpha, \beta + 1, r\right).$$

Then, we can combine (5.15) with (5.17) to approximate the Gittins recursion as

$$V\left(\alpha, \beta, r\right) = \max\left[\frac{r}{1-\gamma}, \frac{\alpha}{\alpha + \beta} + \gamma V\left(\alpha, \beta, r\right)\right]. \qquad (5.18)$$

In this case, it can be shown that (5.18) has the solution

$$V\left(\alpha, \beta, r\right) = \frac{1}{1-\gamma} \max\left(r, \frac{\alpha}{\alpha + \beta}\right),$$

and the solution to (5.16) in this case is simply

$$r^* = \frac{\alpha}{\alpha + \beta}.$$

β \ α	1	2	3	4	5	6	7
1	0.7029	0.8001	0.8452	0.8723	0.8905	0.9039	0.9141
2	0.5001	0.6346	0.7072	0.7539	0.7869	0.8115	0.8307
3	0.3796	0.5163	0.6010	0.6579	0.6996	0.7318	0.7573
4	0.3021	0.4342	0.5184	0.5809	0.6276	0.6642	0.6940
5	0.2488	0.3720	0.4561	0.5179	0.5676	0.6071	0.6395
6	0.2103	0.3245	0.4058	0.4677	0.5168	0.5581	0.5923
7	0.1815	0.2871	0.3647	0.4257	0.4748	0.5156	0.5510

Table 5.1 Gittins indices for the beta-Bernoulli model with $\alpha, \beta = 1, ..., 7$ for $\gamma = 0.9$.

We can use this result to approximate Gittins indices for a desired α^n and β^n. First, we choose some large number N. If $\alpha + \beta \geq N$, we assume that $V(\alpha, \beta, r) = \frac{1}{1-\gamma}\frac{\alpha}{\alpha+\beta}$ for all r. Then, we can use (5.15) and (5.17) to work backwards and compute $V(\alpha^n, \beta^n, r)$ for a particular value of r. Finally, we can use a search algorithm to find the particular value r^* that makes the two components of the maximum in the expression for $V(\alpha^n, \beta^n, r)$ equal.

The computational cost of this method is high. If N is large, the backwards recursion becomes more expensive for each value of r, and we have to repeat it many times to find the value r^*. However, the recursion (for fixed r) is simple enough to be coded in a spreadsheet, and r can then be varied through trial and error (see Exercise 5.3). Such an exercise allows one to get a sense of the complexity of the problem.

When all else fails, the monograph by Gittins (1989) provides tables of Gittins indices for several values of γ and $\alpha, \beta = 1, 2, ..., 40$. A few of these values are given in Tables 5.1 and 5.2 for $\gamma = 0.9, 0.95$. The tables allow us to make several interesting observations. First, the Gittins indices are all numbers in the interval $[0, 1]$. In fact, this is always true in the beta-Bernoulli model (but not for other models, as we shall see in the next section). Second, the indices are increasing in the number of successes, and decreasing in the number of failures. This is logical; if the number of successes is low, and the number of trials is high, we can be fairly sure that the success probability of the slot machine is low, and therefore the fixed-reward process should give us smaller rewards.

5.3.2 Gittins indices in the normal-normal model

Let us now switch to the normal-normal model. Instead of winning probabilities, we deal with average winnings, and we assume that the winnings in each play follow a normal distribution. The quantity μ_x represents the unknown average winnings of slot machine x. Every observation W_x^n is normal with mean μ_x and known precision β^W, and every unknown mean μ_x is normally distributed with prior mean $\bar{\mu}_x^0$ and

β \ α	1	2	3	4	5	6	7
1	0.7614	0.8381	0.8736	0.8948	0.9092	0.9197	0.9278
2	0.5601	0.6810	0.7443	0.7845	0.8128	0.8340	0.8595
3	0.4334	0.5621	0.6392	0.6903	0.7281	0.7568	0.7797
4	0.3477	0.4753	0.5556	0.6133	0.6563	0.6899	0.7174
5	0.2877	0.4094	0.4898	0.5493	0.5957	0.6326	0.6628
6	0.2439	0.3576	0.4372	0.4964	0.5440	0.5830	0.6152
7	0.2106	0.3172	0.3937	0.4528	0.4999	0.5397	0.5733

Table 5.2 Gittins indices for the beta-Bernoulli model with $\alpha, \beta = 1, ..., 7$ for $\gamma = 0.95$.

prior precision β_x^0. In every time step, we select an alternative x, observe a random reward W_x^{n+1}, and apply (3.2)-(3.3) to obtain a new set of beliefs $\left(\bar{\mu}^{n+1}, \beta^{n+1}\right)$.

The Gittins index of slot machine x at time n can be written as the function $I_x^{Gitt,n}\left(\bar{\mu}_x^n, \sigma_x^n, \sigma_W, \gamma\right)$. Observe that this quantity only depends on our beliefs about slot machine x, and not on our beliefs about any other slot machines $y \neq x$. This is the key feature of any index policy. However, the index does depend on the problem parameters σ_W and γ. We find it convenient to write the index in terms of the variance rather than the precision, for reasons that will become clear below.

Gittins showed that $I_x^{Gitt,n}$ can be simplified using

$$I_x^{Gitt,n}\left(\bar{\mu}_x^n, \sigma_x^n, \sigma_W, \gamma\right) = \bar{\mu}_x^n + \sigma_W \cdot I_x^{Gitt,n}\left(0, \frac{\sigma_x^n}{\sigma_W}, 1, \gamma\right). \tag{5.19}$$

This equation is reminiscent of the well-known property of normal random variables. Just as any random variable can be written as a function of a standard normal random variable, so a Gittins index can be written in terms of a "standard normal" Gittins index, as long as we are using a normal-normal learning model.

For notational convenience, we can write

$$I_x^{Gitt,n}\left(0, \frac{\sigma_x^n}{\sigma_W}, 1, \gamma\right) = G\left(\frac{\sigma_x^n}{\sigma_W}, \gamma\right).$$

Thus, we only have to compute Gittins indices for fixed values of the prior mean and experimental noise, and then use (5.19) to translate it to our current beliefs. Table 5.3 gives the values of $G\left(s, \gamma\right)$ for $\gamma = 0.95, 0.99$ and $s = 1/\sqrt{k}$ for $k = 1, 2, ...$ This corresponds to a case where the experimental noise is equal to 1, and our beliefs about alternative x have the variance $1/k$ if we make k experiments of x.

The table reveals several interesting facts about Gittins indices. First, $G\left(1/\sqrt{k}, \gamma\right)$ is increasing in γ. If γ is larger, this means that we have a larger effective time horizon. Essentially, a larger portion of our time horizon "matters," more and more of the rewards we collect remain large enough after discounting to have a notable

	Discount factor					
k	0.5	0.7	0.9	0.95	0.99	0.995
1	0.2057	0.3691	0.7466	0.9956	1.5758	1.8175
2	0.1217	0.2224	0.4662	0.6343	1.0415	1.2157
3	0.0873	0.1614	0.3465	0.4781	0.8061	0.9493
4	0.0683	0.1272	0.2781	0.3878	0.6677	0.7919
5	0.0562	0.1052	0.2332	0.3281	0.5747	0.6857
6	0.0477	0.0898	0.2013	0.2852	0.5072	0.6082
7	0.0415	0.0784	0.1774	0.2528	0.4554	0.5487
8	0.0367	0.0696	0.1587	0.2274	0.4144	0.5013
9	0.0329	0.0626	0.1437	0.2069	0.3608	0.4624
10	0.0299	0.0569	0.1313	0.1899	0.3529	0.4299
20	0.0155	0.0298	0.0712	0.1058	0.2094	0.2615
30	0.0104	0.0202	0.0491	0.0739	0.1520	0.1927
40	0.0079	0.0153	0.0375	0.0570	0.1202	0.1542
50	0.0063	0.0123	0.0304	0.0464	0.0998	0.1292
60	0.0053	0.0103	0.0255	0.0392	0.0855	0.1115
70	0.0045	0.0089	0.0220	0.0339	0.0749	0.0983
80	0.0040	0.0078	0.0193	0.0299	0.0667	0.0881
90	0.0035	0.0069	0.0173	0.0267	0.0602	0.0798
100	0.0032	0.0062	0.0156	0.0242	0.0549	0.0730

Table 5.3 Gittins indices $G(s, \gamma)$ for the case where $s = 1/\sqrt{k}$, from Gittins (1989).

effect on our objective value. This means that we can afford to do more exploration, because one low reward early on will not harm our objective value as much. Hence, the Gittins indices are higher, encouraging us to explore more.

Second, $G\left(1/\sqrt{k}, \gamma\right)$ is decreasing in k. This is fairly intuitive. The standard Gittins index $G(s, \gamma)$ represents the uncertainty bonus, and does not depend on our estimate of the value of an alternative. As the variance of our beliefs goes down, the uncertainty bonus should go down as well, and we should only continue to measure the alternative if it provides a high reward.

Unfortunately, even the seminal work by Gittins does not give the values of $G(s, \gamma)$ for all possible s and γ. In general, computing these values is a difficult problem in and of itself. For this reason, a minor literature on Gittins approximation has arisen in the past ten years. To obtain a practical algorithm, it is necessary to examine this literature in more depth.

5.3.3 Approximating Gittins indices

Finding Gittins indices is somewhat like finding the cdf of the standard normal distribution. It cannot be done analytically, and requires instead a fairly tedious numerical calculation. We take for granted the existence of nice functions built into most programming languages for computing the cumulative standard normal distribution, for which extremely accurate polynomial approximations are available. In Excel, this is available using the function `NORMINV`.

As of this writing, such functions do not exist for Gittins indices. However, in the case of the normal-normal model, there is a reasonably good approximation that results in an easily computable algorithm. First, it can be shown that

$$G(s,\gamma) = \sqrt{-\log\gamma}\cdot b\left(-\frac{s^2}{\log\gamma}\right),$$

where the function b must be approximated. The best available approximation of Gittins indices is given by

$$\tilde{b}(s) = \begin{cases} \frac{s}{\sqrt{2}} & s \le \frac{1}{7}, \\ e^{-0.02645(\log s)^2 + 0.89106\log s - 0.4873} & \frac{1}{7} < s \le 100, \\ \sqrt{s}\left(2\log s - \log\log s - \log 16\pi\right)^{\frac{1}{2}} & s > 100. \end{cases}$$

Thus, the approximate version of (5.19) is

$$I_x^{Gitt,n} \approx \bar{\mu}_x^n + \sigma_W\sqrt{-\log\gamma}\cdot\tilde{b}\left(-\frac{\bar{\sigma}_x^{2,n}}{\sigma_W^2\log\gamma}\right). \tag{5.20}$$

Figure 5.1 gives us an idea of the quality of this approximation. In a few select cases where the exact Gittins indices are known (see Table 5.3), we can evaluate the approximation against the exact values. We see that the approximation gives the most error for small values of k, but steadily improves as k increases. The approximation for $\gamma = 0.9$ tends to be slightly more accurate than the one for $\gamma = 0.99$. In general, the approximation is more accurate for lower values of γ.

5.4 THE KNOWLEDGE GRADIENT FOR ONLINE LEARNING

The knowledge gradient approach that we introduced in Chapter 4 is a particularly simple and elegant strategy for collecting information. In the ranking and selection setting, it produces an easily computable algorithm. What is more, it can be adapted to handle correlated beliefs, as well as non-Gaussian learning models. A natural question, then, is whether it can be adapted for the multi-armed bandit problem, which is the online version of ranking and selection.

In this section, we develop a simple relationship between the knowledge gradient for offline and online settings, which also allows us to consider problems with correlated beliefs. We present a few experimental comparisons that seem to suggest that this works quite well for online problems. We then discuss applications to problems with non-normal belief models.

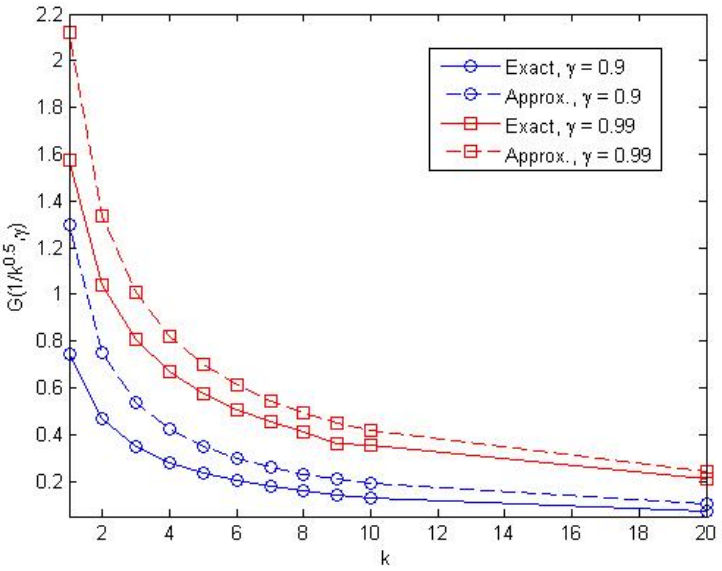


Figure 5.1 Comparison of approximate and exact Gittins indices for $\gamma = 0.9, 0.99$.

5.4.1 The basic idea

Once again, consider the normal-normal Bayesian learning model. Suppose that we can run N experiments, and that $\gamma = 1$. Furthermore, suppose that we have already run n experiments, and have constructed estimates $\bar{\mu}_x^n$ and β_x^n for each alternative x. Now, as a thought experiment, let us imagine that we will suddenly cease learning, starting at time n. We will still continue to collect rewards, but we will no longer be able to use the updating equations (3.2) and (3.3) to change our beliefs. We are stuck with our time-n beliefs until the end of the time horizon.

If this were to occur, the best course of action would be to choose $x^{n'} = \arg\max_x \bar{\mu}_x^n$ for all times $n \leq n' \leq N$. Since we cannot change our beliefs anymore, all we can really do is choose the alternative that seems to be the best, based on the information that we managed to collect up to this point. The expected total reward that we will collect by doing this, from time n to time N, is given by

$$V^{Stop,n}(S^n) = (N - n + 1) \max_x \bar{\mu}_x^n, \tag{5.21}$$

simply because there are $N - n + 1$ rewards left to collect. Because $\gamma = 1$, each reward is weighted equally. For instance, in the example given in Table 5.4, this quantity is $V^{Stop,n}(S^n) = 6 \cdot 5.5 = 33$.

Consider a different thought experiment. We are still at time n, but now our next decision will change our beliefs as usual. Assume we run experiment $x^n = x$ at time n, giving us updated estimates $\bar{\mu}_{x'}^{n+1}(x)$. However, starting at time $n + 1$, we will cease to learn, and from there on we will be in the situation described above. This means that, starting at time $n + 1$, we will always measure the alternative given by

$\arg\max_{x'} \bar{\mu}_{x'}^{n+1}(x)$. The problem thus reduces to choosing one single decision x^n to maximize the expected total reward we collect, starting at time n.

This idea is essentially the knowledge gradient concept from a slightly different point of view. In ranking and selection, we chose each decision to maximize the incremental improvement (obtained from a single experiment) in our estimate of the best value. Essentially, we treated each decision as if it were the last time we were allowed to learn. We made each decision in such a way as to get the most benefit out of that single experiment. In the online setting, we do the same thing, only "benefit" is now expressed in terms of the total reward that we can collect from time n to the end of the time horizon.

The KG decision for the bandit problem is given by

$$X^{KG,n} = \arg\max_x \mathbb{E}\left[\mu_x + V^{Stop,n+1}\left(S^{n+1}\right) \mid S^n, x^n = x\right] \tag{5.22}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1}(x) \mid S^n, x^n = x\right] \tag{5.23}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\mathbb{E}\left[\max_{x'} \bar{\mu}_{x'}^{n+1}(x) - \max_{x'} \bar{\mu}_{x'}^n \mid S^n, x^n = x\right] \tag{5.24}$$

$$= \arg\max_x \bar{\mu}_x^n + (N-n)\,\nu_x^{KG,n}, \tag{5.25}$$

where $\nu_x^{KG,n}$ is simply the knowledge gradient for ranking and selection, given by (4.13). We start with the basic Bellman equation in (5.22). The downstream value is given by $V^{Stop,n+1}$ because we assume that we will cease to learn starting at time $n+1$. Next, we use the fact that $\mathbb{E}\left(\mu_x \mid S^n\right) = \bar{\mu}_x^n$, together with the definition of $V^{Stop,n+1}$ from (5.21) to obtain (5.23). Because the quantity $\max_{x'} \bar{\mu}_{x'}^n$ is constant given S^n, and does not depend on x, we can put it into the expression without changing the arg max, thus arriving at (5.24). Finally, we apply the definition of the knowledge gradient from (4.3) to obtain (5.25).

This line of reasoning has given us a simple and easily computable algorithm for the multi-armed bandit problem. At first, the expression $\bar{\mu}_x^n + (N-n)\,\nu_x^{KG,n}$ that we compute for alternative x may look very similar to the index policies we discussed earlier. Like interval estimation, Gittins indices, and other methods, KG takes $\bar{\mu}_x^n$ and adds an uncertainty bonus $(N-n)\,\nu_x^{KG,n}$. Just as in the other index policies, the uncertainty bonus gets smaller as σ_x^n gets smaller: thus, if the level of uncertainty is zero, the uncertainty bonus is zero as well. Furthermore, all other things being equal, the uncertainty bonus is larger if n is smaller, reflecting the fact that it is more important to learn in the early stages of the problem, while we have more remaining time steps in which we can potentially use the information we collect.

However, the KG policy is not an index policy. Crucially, the knowledge gradient $\nu_x^{KG,n}$ depends not only on $\bar{\mu}_x^n$, but also on $\max_{x' \neq x} \bar{\mu}_{x'}^n$. This cannot happen in an index policy, where the index of x is only allowed to depend on our beliefs about x. The knowledge gradient policy does not decompose the multi-armed bandit problem into many one-armed bandit problems. It considers each alternative relative to the others.

Table 5.4 shows the computations performed by the online KG policy for a particular problem with five alternatives and $N - n = 5$ experiments remaining in the time horizon. This example illustrates the distinction between the online KG policy and

Choice	$\bar{\mu}^n$	β^n	$\tilde{\sigma}^n$	ζ^n	$\nu^{KG,n}$	$\bar{\mu}^n + 5 \cdot \nu^{KG,n}$
1	2	1/2	1.1547	-3.0311	0.0004	2.0020
2	4	1/2	1.1547	-1.2990	0.0527	4.2634
3	3	2/3	0.9487	-2.6352	0.0012	3.0062
4	5.5	1/2	1.1547	-0.8660	0.1234	6.1168
5	4.5	1/3	1.5	-0.6667	0.2267	5.6333

Table 5.4 Calculations for the online KG policy in a bandit problem with $M = 5$, $N - n = 5$, and $\beta_x^W = 1$ for all x.

the offline KG policy from Chapter 4. If this were a ranking and selection problem, we would measure the alternative with the highest KG factor, namely alternative 5. However, even though alternative 4 has a smaller KG factor, our estimate $\bar{\mu}_4^n$ is sufficiently larger than $\bar{\mu}_5^n$ to make the online KG policy choose alternative 4. Thus, the online KG policy favors exploitation more than the offline KG policy.

At the same time, if $N - n = 50$ in the same example, then the online KG policy would prefer alternative 5 to alternative 4, thus agreeing with the offline KG policy. Unlike the offline KG policy, online KG is what is known as a *nonstationary policy*. This means that the decision made by online KG depends on n as well as on S^n. The exact same belief state can lead online KG to measure different alternatives depending on the current time.

The formulation of KG for bandit problems is quite versatile. Suppose that we have a discount factor $\gamma < 1$. It is a simple matter to repeat the above reasoning and arrive at the decision rule

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \gamma \frac{1 - \gamma^{N-n}}{1 - \gamma} \nu_x^{KG,n}.$$

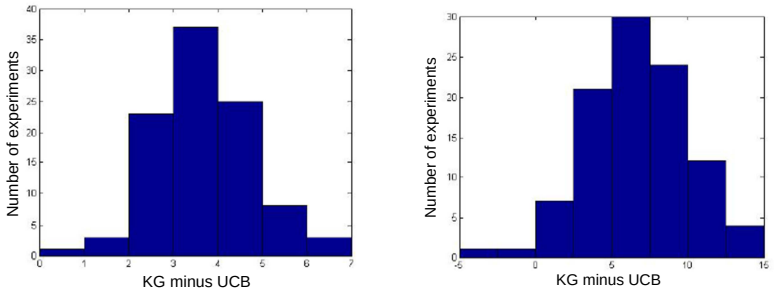
Taking $N \to \infty$, we obtain the knowledge gradient rule for infinite-horizon bandit problems,

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \frac{\gamma}{1 - \gamma} \nu_x^{KG,n}.$$

This is substantially easier to compute than Gittins indices. Keep in mind also that Gittins indices are only designed for infinite-horizon problems. If N is finite, the Gittins policy becomes a heuristic with γ serving as a tunable parameter. On the other hand, the KG policy can be defined for both finite- and infinite-horizon problems, and requires no tuning in either case. Of course, it is not an optimal policy, but it is able to respond to these different environments without the need for any tunable parameters.

5.4.2 Tunable variations

As with other policies, we can introduce tunable parameters that can produce better results when tuned for specific environments. We start with the tunable version of the knowledge gradient we first introduced for the offline (terminal reward) problem in section 4.3.3. There, we made the assumption that we would evaluate an alternative



(a) (b)

Figure 5.2 Histogram showing the average difference in the total discounted rewards collected by KG and approximate Gittins indices across 100 bandit problems with $M = 100$ and $\gamma = 0.9$. (a) uses truth from the prior, and (b) uses truth from an unbiased uniform distribution (from Ryzhov et al. 2011).

with precision $\theta_1 \beta^W$, where θ_1 is a positive integer that can be interpret as repeat factor which anticipates that we might perform the same experiment multiple times. Thus, our updated precision from an experiment would be modeled as

$$\beta_x^{n+1} = \beta_x^n + \theta_1 \beta^W.$$

We let $\nu_x^{KG,n}(\theta_1)$ be the offline KG formula using θ_1 as the repeat factor for the precision β^W. Then, we introduce a second parameter θ_2 which replaces the horizon $N - n$ in our online KG formula. Our tunable online KG formula is now given by

$$X^{OLKG}(\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta_2 \nu_x^{KG,n}(\theta_1)\right),$$

where $\theta = (\theta_1, \theta_2)$ captures our tunable parameters.

Tuning can have a major impact for problems where the noise of a measurement is quite high, and/or where the learning budget is relatively small.

5.4.3 Some experimental comparisons

It is useful to see how the knowledge gradient adapted for online problems compares against some of the popular policies that have been proposed for this problem class. Figure 5.2 shows a histogram of the performance of infinite-horizon KG minus the performance of a policy based on the Gittins approximation from Section 5.3.3 across 100 bandit problems with 100 alternatives. The numbers represent differences in the total discounted rewards (with $\gamma = 0.9$) collected by the two policies. We can see that all the numbers are positive, meaning that KG outperformed the approximate policy in every problem. To be sure, one can create problems where this is not the case, but it does indicate that KG can be competitive with the best existing approximation of the optimal policy.

The main advantage of the KG method, however, is that its nature as a non-index policy makes it well suited to the case where our beliefs about the alternatives are

tb

Comparison	Average difference	Standard error
KG minus approximate Gittins	0.7076	0.0997
KG minus interval estimation	-0.0912	0.0857
KG minus UCB	44.4305	0.6324
KG minus UCB1	1.2091	0.1020
KG minus pure exploitation	5.5413	0.1511

Table 5.5 Comparison between knowledge gradient and competing policies for 100 truth-from-prior experiments (from Ryzhov et al. 2011).

correlated. Suppose that we begin with a multivariate normal prior $\mu \sim N\left(\bar{\mu}^0, \Sigma^0\right)$, and use (2.18) and (2.19) to update our beliefs, but we keep the bandit objective function from (**??**). In this setting, index policies are inherently unsuitable: an index policy depends on our ability to decompose the problem and consider every alternative as if it were the only alternative in the problem, but the whole point of correlations is that the alternatives are inextricably related. Thus, while we can still use index policies such as Gittins and UCB as heuristics, they automatically lose their nice optimality properties in the correlated setting.

However, we can still define a knowledge gradient method in the correlated case. In fact, the KG decision rule is still given by (5.25), with the only change that we replace $\nu_x^{KG,n}$ by $h\left(\bar{\mu}^n, \tilde{\sigma}\left(\Sigma^n, x\right)\right)$ from (4.24). This quantity is then computed exactly as in Chapter 4. As of this writing, KG is the first algorithm that is able to consider bandit problems with multivariate normal priors.

Table 5.5 summarizes the results of a series of experiments on online problems with 100 alternatives with correlated beliefs. The knowledge gradient outperformed approximate Gittins, UCB, UCB1 and pure exploration for all 100 sample realizations. Only interval estimation proved to be competitive, and actually outperformed the knowledge gradient policy 77 percent of the time (although the difference in the average performance was not statistically significant). Recall that interval estimation uses the policy of maximizing the index

$$\nu^{IE,n} = \bar{\mu}_x^n + z_\alpha \sigma_x^n,$$

where $\bar{\mu}_x^n$ is our current estimate of the value of alternative x and σ_x^n is the standard deviation of our estimate $\bar{\mu}_x^n$.

The performance of interval estimation seems surprising, given that the knowledge gradient policy is taking advantage of a covariance structure (which we assume is known in advance), while IE has no such ability. However, IE has a tunable parameter z_α, and it is important that this parameter be tuned carefully. Figure 5.3 shows the behavior of interval estimation as a function of z_α, along with the knowledge gradient (which of course is a constant with respect to z_α). Note that IE outperforms KG only over a narrow range. Even more important, notice the severe degradation of IE as z_α moves away from its best setting.

This experiment demonstrates that there is a tremendous amount of information in a tunable parameter. Statements have been made in the literature that z_α can be safely

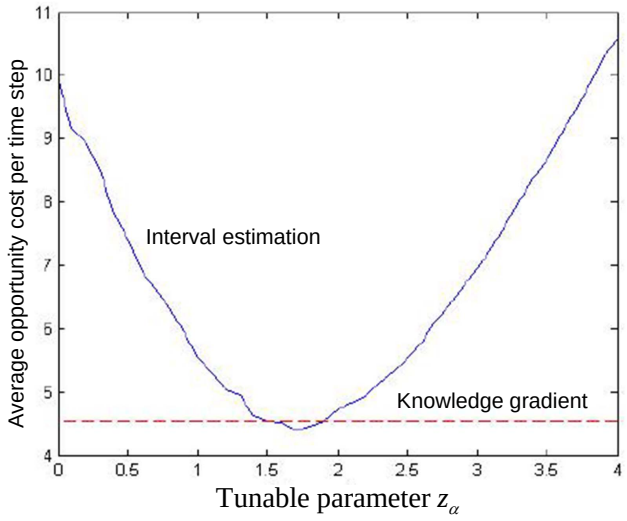


Figure 5.3 Expected opportunity cost for interval estimation as a function of z_α along with the knowledge gradient (from Ryzhov & Powell (2009*a*)).

chosen to be around 2 or 3, but problems have been found where the optimal value of z_α ranges anywhere from 0.5 to 5.0. Furthermore, the solution can be quite sensitive to the choice of z_α, suggesting that tuning has to be performed with care.

5.4.4 Non-normal models

Just as in ranking and selection, KG can also be extended to non-normal learning models, although we should take care to define the knowledge gradient in accordance with our reward structure. Suppose that we are working with the gamma-exponential model, where W_x follows an exponential distribution with parameter λ_x, and each λ_x has a gamma prior distribution with parameters a_x^0, b_x^0. Thus, our prior estimate of every reward is

$$\begin{aligned}\mathbb{E}\left(W_x\right) &= \mathbb{E}\left[\mathbb{E}\left(W_x \mid \lambda_x\right)\right] \\ &= \mathbb{E}\left[\frac{1}{\lambda_x}\right] \\ &= \frac{b_x^0}{a_x^0 - 1},\end{aligned}$$

where $b_x^0 > 0$ and $a_x^0 > 1$. Suppose that our objective function is to maximize the sum of the expected rewards,

$$\max_{\pi \in \Pi} \sum_{n=0}^{N} \gamma^n \frac{1}{\lambda_{x^n}}.$$

Then, the online KG decision rule for $\gamma = 1$ is given by

$$X^{KG,n} = \arg\max_x \frac{b_x^n}{a_x^n - 1} + (N - n)\, \nu_x^{KG,n}.$$

To write the knowledge gradient $\nu_x^{KG,n}$, let $C_x^n = \max_{x' \neq x} \frac{b_{x'}^n}{a_{x'}^n - 1}$ and define the baseline KG quantity

$$\tilde{\nu}_x^n = \frac{(b_x^n)^{a_x^n}}{(a_x^n C_x^n)^{a_x^n - 1}} \left(\frac{1}{a_x^n - 1} - \frac{1}{a_x^n} \right).$$

Then,

$$\nu_x^{KG,n} = \begin{cases} \tilde{\nu}_x^n & \text{if } x \neq \arg\max_{x'} \frac{b_{x'}^n}{a_{x'}^n - 1} \\ \tilde{\nu}_x^n - \left(\frac{b_x^n}{a_x^n - 1} - C_x^n \right) & \text{if } C_x^n \geq \frac{b_x^n}{a_x^n} \\ 0 & \text{otherwise.} \end{cases} \tag{5.26}$$

Once again, we have an easily computable KG algorithm, for a problem where Gittins approximations are not easily available. Note the presence of a penalty term in (5.26), much like in the expression for gamma-exponential KG in ranking and selection that was presented in Section 4.7.

5.5 VARIATIONS OF BANDIT PROBLEMS

One of the notable features of the bandit literature has been the introduction of a wide range of variations. The style of this community is to provide names of the form "XXX bandits" where XXX describes some variation, typically in the nature of the arms (choices), the exogenous information process, and/or the belief model. This style has produced a colorful literature full of creative adjectives, as illustrated in table 5.6. Below we provide more detailed descriptions on a few of the examples.

Restless bandits - The standard bandit model assumes that the truth μ_x for arm x remains the same over time. Restless bandits describe a model where the means are allowed to vary over time.

Arm acquiring bandits - These are problems where the number of alternatives grows over time.

Intermittent bandits - In this setting, not all the arms are available to be tested. A related variation is known as "sleeping bandits."

Continuous armed bandits - While the standard bandit problem involves discrete alternatives $x \in \mathcal{X} = \{x_1, \ldots, x_M\}$, some authors have considered the case where the choices are continuous.

Response-surface bandits - This is an instance of a bandit problem where the adjective is describing the belief model. While the most standard belief model is a lookup table, some authors have considered parametric models ("response surface"). The term linear bandit has also been used for this purpose.

Finite horizon bandits - While the classic problem assumes an infinite horizon, some authors have worked on finite-horizon problems.

Dueling or adversarial bandits - Here we have two agents where one is requesting information (choosing an arm to test), while another is providing information specifically to produce worse results for the first bandit.

Contextual bandits - This describes a setting where we are given exogenous information (call it S^c) that influences the underlying distributions. Instead of learning the best arm x^*, we want to learn the best arm $x^*(S^c)$ in the "context" of the new information S^c.

Despite the popularity of multiarmed bandit problems, there does not seem to be a formal definition of a bandit problem. The original setting of a problem with discrete alternatives where rewards are accumulated while learning is happening has been generalized to problems with continuous alternatives, and for problems where we are only interested in the performance after all experiments have been completed.

Bandit problems appear to be any problems which involve an exploration vs. exploitation tradeoff, which is a property of the policy (or search algorithm), rather than the problem. We note that exploration-exploitation tradeoffs occur with any problem where we maximize cumulative rewards (this applies to both finite and infinite horizon problems), as well as any finite horizon problems where we are optimizing the final reward. A key feature of any procedure that manages an exploration-exploitation tradeoff is the use of a belief model, which is the only way to learn.

5.6 EVALUATING POLICIES

In chapter 3, we introduced a series of heuristic policies such as Boltzmann exploration or interval estimation, but we also introduced a tunable form of upper confidence bounding, with more variations in this chapter. In section 4.3.3, we even introduced a tunable form of the knowledge gradient for problems where the value of information is nonconcave.

Any tunable policy can be tuned for either offline (terminal reward) or online (cumulative reward) settings. Let $X^\pi(S|\theta)$ be a policy where S is our belief state and θ is some tunable parameter (there could be more than one). If we are in an offline setting, we can use $X^\pi(S|\theta)$ to guide a sequence of experiments $x^0, W^1, x^1, W^2, \ldots, x^{N-1}, W^N)$ that produces a set of estimates $\bar{\mu}_x^N$ from which we can identify the best alternative (or design)

$$x^{\pi,N}(\theta) = \arg\max_x \bar{\mu}_x^N,$$

where we retain the explicit dependence on θ. We can then optimize θ by solving (approximately) the optimization problem

$$\max_\theta \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^{\pi,N}}, \tag{5.27}$$

where we are assuming that we can use our known μ (known only to our simulator) to evaluate the performance of the policy. This is the same as our policy search problem we posed in equation (5.1) at the beginning of the chapter, except that now we are writing it explicitly in terms of the tunable parameter θ.

Bandit problem	Description
Multiarmed bandits	Basic problem with discrete alternatives, online (cumulative regret) learning, lookup table belief model with independent beliefs
Restless bandits	Truth evolves exogenously over time
Adversarial bandits	Distributions from which rewards are being sampled can be set arbitrarily by an adversary
Continuum-armed bandits	Arms are continuous
X-armed bandits	Arms are a general topological space
Contextual bandits	Exogenous state is revealed which affects the distribution of rewards
Dueling bandits	The agent gets a relative feedback of the arms as opposed to absolute feedback
Arm-acquiring bandits	New machines arrive over time
Intermittent bandits	Arms are not always available
Response surface bandits	Belief model is a response surface (typically a linear model)
Linear bandits	Belief is a linear model
Dependent bandits	A form of correlated beliefs
Finite horizon bandits	Finite-horizon form of the classical infinite horizon multi-armed bandit problem
Parametric bandits	Beliefs about arms are described by a parametric belief model
Nonparametric bandits	Bandits with nonparametric belief models
Graph-structured bandits	Feedback from neighbors on graph instead of single arm
Extreme bandits	Optimize the maximum of recieved rewards
Quantile-based bandits	The arms are evaluated in terms of a specified quantile
Preference-based bandits	Find the correct ordering of arms
Best-arm bandits	Identify the optimal arm with the largest confidence given a fixed budget

Table 5.6 A sample of the growing population of "bandit" problems.

We can apply the same idea to any online problem by simply changing the objective function to capture the cumulative reward, giving us

$$\max_{\theta} \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \mu_{X^\pi(S^n)}. \tag{5.28}$$

With this observation in hand, it is useful to take a closer look at the policies themselves to contrast how they might behave in offline and online learning settings. We begin by noting that most of our policies, including interval estimation, the UCB

policies, Gittins indices and the knowledge gradient for online learning, all have the basic structure

$$X^{\pi}(S) = \arg\max \left(\bar{\mu}_x^n + \theta U^B(S)\right), \tag{5.29}$$

where U^B is some term that is often referred to as the "uncertainty bonus." Examples of policies that have this structure include:

Interval estimation:
$X^{IE}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\bar{\sigma}_x^n\right)$

Upper confidence bounding (UCB1 variation):
$X^{UCB1}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sqrt{\frac{\log n}{N_x^n}}\right)$

Upper confidence bounding (UCBE variation):
$X^{UCBE,n}(\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sqrt{\frac{1}{N_x^n}}\right)$. Upper confidence bounding (UCBE variation):
$X^{UCB}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sqrt{\frac{\log n}{N_x^n}}\right)$

Gittins indices:
$X^{Gittins}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\sigma^W\right)$

Online knowledge gradient:
$X^{OLKG}(S|\theta) = \arg\max_x \left(\bar{\mu}_x^n + \theta\nu_x^{KG}\right)$

Each of these has been presented in its tunable form. Each strikes a balance between the estimated reward from trying experiment x, given by $\bar{\mu}_x^n$, and a measure of the uncertainty in this estimate. The parameter θ then strikes the balance between exploitation (which emphasizes $\bar{\mu}_x^n$), and exploration, captured by the uncertainty bonus.

We note, however, that these policies are all designed for online learning, since it is only in this setting that $\bar{\mu}_x^n$ is actually our best estimate of what we will receive by choosing $x^n = x$, with the uncertainty bonus encouraging exploration. If our budget is just one experiment, then the optimal $\theta = 0$.

By contrast, a value of information policy such as the knowledge gradient or expected improvement is specifically designed for offline learning, since it only captures the value of information. Recall that the offline knowledge gradient is given by

$$\nu_x^{KG,n} = \mathbb{E}\left\{V^{n+1}(S^{n+1}(x))|S^n\right\} - V^n(S^n).$$

Note that $\bar{\mu}_x^n$ is missing, as it should be, since in an offline setting, we do not care how well an experiment might perform. We only care about what we learn, that contributes to our ability to identify the best choices at the end.

Given this observation, how is it that these policies can be tuned for offline learning (and work reasonably well)? First, they all capture the fundamental tradeoff between exploitation (captured by $\bar{\mu}_x^n$ in the policy), and exploration (captured by some form of uncertainty bonus). However, these are heuristic policies, which is why they all have a tunable parameter. This also hints at the power of tuning.

5.7 BIBLIOGRAPHIC NOTES

Section 5.3 - Gittins index theory is due to Gittins & Jones (1974), Gittins (1979) and Gittins (1989). Berry & Fristedt (1985) also provides a rigorous analysis of bandit problems. This research has launched an entire field of research into the search for index policies for variations on the basic bandit problems. Glazebrook (1982) analyzes policies for variations of the basic bandit model. Glazebrook & Minty (2009) presents a generalized index for bandit problems with general constraints on information collection resources. Bertsimas & Nino-Mora (2000) show how an index policy can be computed using linear programming for a certain class of bandit problems. See the updated version of Gittins' 1989 book, Gittins et al. (2011), for a modern treatment of bandit problems and a much more thorough treatment of this extensive literature. The approximation of Gittins indices is due to Chick & Gans (2009), building on the diffusion approximation of Brezzi & Lai (2002).

Section 5.2 - Lai & Robbins (1985) and Lai (1987) provide the seminal research that shows that the number of times an upper confidence bound policy chooses a particular suboptimal machine is on the order of $\log N$, and that this is the best possible bound. Auer et al. (2002) derives finite-time regret bounds on the UCB1 and UCB1-Normal policies and reports on comparisons against variations of UCB policies and epsilon-greedy on some small problems (up to 10 arms). The UCB policy for exponential rewards comes from Agrawal (1995).

Section 5.4 - The online adaptation of the knowledge gradient is due to Ryzhov et al. (2011). Some additional experimental comparisons can be found in Ryzhov & Powell (2009*b*). Ryzhov & Powell (2011*c*) presents the KG policy for the gamma-exponential model. Rates of convergence for KG-type policies are still an open question, but Bull (2011) is an interesting first step in this direction.

PROBLEMS

5.1 You have three materials, A, B and C that you want to test for their ability to convert solar energy to electricity, and you wish to find which one produces the highest efficiency, which you have to do by learning from field implementations (in other words, online learning). Table 5.7 shows your initial beliefs (which we assume are independent) summarized as the mean and precision. Your prior belief is normal, and testing alternative x produces an observation W_x which is normally distributed with precision $\beta^W = 1$.

a) You follow some learning policy that has you first evaluating A, then B and finally C, obtaining the observations W_x^n shown in Table 5.7 (for example, $W_A^1 = 3$). Give the updated belief (mean and precision) for $\bar{\mu}_A^1$ given the observation $W_A^1 = 3$. Also compute the updated means only (not the precisions) for $\bar{\mu}_B^2$ and $\bar{\mu}_C^3$.

b) Give the objective function (algebraically) to find the best policy after N experiments if this is an off-line learning problem. Compute a sample realization of the objective function for this example.

Iteration	A	B	C
Prior (μ_x, β_x)	(5,.05)	(3,.02)	(4,.01)
1	3	-	-
2	-	2	-
3	-	-	6

Table 5.7 Three observations, for three alternatives, given a normally distributed belief, and assuming normally distributed observations.

c) Give the objective function (algebraically) to find the best policy if this is an on-line learning problem. Compute a sample realization of the objective function for this example.

5.2 Consider a classic multi-armed bandit problem with normally distributed rewards.

a) Is the multi-armed bandit problem an example of an on-line or off-line learning problem?

b) Let $R^n(x^n)$ be the random variable giving the reward from measuring bandit $x^n \in (1, 2, \ldots, M)$ in iteration n. Give the objective function we are trying to maximize (define any other parameters you may need).

c) Let $\Gamma(n)$ be the Gittins index when rewards are normally distributed with mean 0 and variance 1, and let $\nu_x(\mu_x, \sigma_x^2)$ be the Gittins index for a bandit where the mean reward is μ with variance σ^2. Write $\nu_x(\mu_x, \sigma_x^2)$ as a function of $\Gamma(n)$.

5.3 Consider a bandit problem with $\gamma < 1$ where we use the beta-Bernoulli learning model.

a) Suppose that, for a particular choice of α, β and r, we have $V(\alpha, \beta, r) \approx V(\alpha + 1, \beta, r) \approx V(\alpha, \beta + 1, r)$. Show that the Gittins recursion is solved by

$$V(\alpha, \beta, r) = \frac{1}{1-\gamma} \max\left(r, \frac{\alpha}{\alpha+\beta}\right).$$

b) In a spreadsheet, choose values for r and γ (these should be stored in two cells of the spreadsheet, so that we can vary them), and create a table that compute the values of $V(\alpha, \beta, r)$ for all $\alpha, \beta = 1, 2, ...$ with $\alpha + \beta < 200$. When $\alpha + \beta = 200$, use $V(\alpha, \beta, r) = \frac{1}{1-\gamma}\frac{\alpha}{\alpha+\beta}$ as a terminal condition for the recursion.

c) The spreadsheet from part b) can now be used to compute Gittins indices. The Gittins index r^* for a particular α and β with $\alpha + \beta < 200$ is the smallest value of r for which $\frac{r}{1-\gamma}$ is equal to the entry for $V(\alpha, \beta, r)$ in the table. Use trial and error to find r^* for $\alpha, \beta = 1, ..., 5$ with $\gamma = 0.9$. Report the values you find, and compare them to the exact values of the Gittins indices in Table 5.1.

5.4 Consider a bandit problem with $\gamma < 1$. Repeat the derivation from Section 5.4 to show that the KG decision rule is given by

$$X^{KG,n} = \arg\max_x \bar{\mu}_x^n + \gamma \frac{1-\gamma^{N-n}}{1-\gamma} \nu_x^{KG,n}$$

for finite N.

5.5 Consider a finite-horizon bandit problem with $\gamma = 1$ and a gamma-exponential learning model. Show that $\nu_x^{KG,n}$ is given by (5.26).

5.6 This exercise needs the spreadsheet:

`http://optimallearning.princeton.edu/exercises/FiveAlternative.xls`

available on the optimal learning web site. You are going to have to construct a learning policy to choose the best of five options, using the problems that are described in the attached spreadsheet. You are welcome to solve the problem directly in the accompanying spreadsheet. But this is an exercise that will be easier for some of you to solve using a programming environment such as MATLAB, Java or perhaps visual basic in Excel.

The spreadsheet illustrates the calculations. Each time you choose a path, the spreadsheet will show you the time for the path. It is up to you to update your estimate of the average travel time and the variance of the estimate. Use a Bayesian framework for this exercise. You can repeat the exercise different times on the same set of random realizations. If you wish to use a fresh set of random numbers, hit the "Refresh button." You can see the data on the "Data" tab. The data tab uses the data in columns A-F, which will not change until you hit the refresh button. The data in columns H-L use the Rand() function, and will change each time there is a recompute (which can be annoying). If you click on the cells in columns H-L, you will see how the random numbers are being generated. The true means are in row 2, and the numbers in row 3 control the spread. You should use a prior estimate of the standard deviation equal to 10 for all your analyses.

The problem set requires testing a number of exploration policies. For each policy, compute two objective functions (averaged over 100 random number seeds):

1) The online objective function, which is the discounted sum of your performance (for the chosen option) over all 100 experiments, with a discount factor of $\gamma = 0.80$. If $C^n(\omega)$ is the observed value of the measured option for the n^{th} experiment for random number seed ω, your objective function would be:

$$F^\pi = \frac{1}{100} \sum_{\omega=1}^{100} \sum_{n=0}^{100} \gamma^n C^n(\omega)$$

2) The final experiment:

$$G^\pi = \frac{1}{100} \sum_{\omega=1}^{100} C^{100}(\omega)$$

F^π is our online objective function, while G^π is our offline objective function. Also let

$$\begin{aligned} F &= \sum_{n=0}^{100} \gamma^n \bar{\mu}^* \\ G &= \bar{\mu}^* \end{aligned}$$

where $\bar{\mu}^*$ is the true mean for the best choice, if we knew the true means. Where necessary, you may assume that the standard deviation of an experiment is 10. You have to test the following policies:

1) Pure exploitation.
2) Boltzmann exploration, using scaling factor $\theta = 1$.
3) Epsilon-greedy exploration, where the exploration probability is given by $1/n$.
4) Interval estimation. Test the performance of $z_\alpha = 1.0, 2.0, 3.0$ and 4.0 and select the one that performs the best.
5) Gittins indices (use the numerical approximation of Gittins indices given in Section 5.3.3.

Do the following

a) In a graph, report F^π and G^π for each of the policies below. Also show F and G to provide a measure of how well we are doing.

b) Discuss your results. Compare the performance of each policy in terms of the two different objective functions.

5.7 Consider a multi-armed bandit problem with independent normal rewards. In this exercise, you will implement a few online policies.

a) How should the opportunity cost be expressed in the online problem?

b) In exercise 3.4 you implemented the interval estimation policy in an offline setting. Now take your code from before, and adapt it to online problems by changing the objective function appropriately. The parameters of the problem should be the same as before (use the same priors and assume $\gamma = 1$), but now you need to compute opportunity cost differently. How does the best value of the tunable parameter z_α change when the problem becomes online? After you tune z_α, report the confidence interval for the opportunity cost using 200 simulations.

c) Now implement the Gittins index approximation for the independent normal-normal model. You cannot solve the Gittins recursion – just use the approximation function $\tilde{b}$. Our problem has a finite horizon, so you can treat the parameter γ in the Gittins index calculation as another tunable parameter. What value of γ gives you the best results?

d) Now implement the online KG policy. Compare the confidence intervals for KG, Gittins, and interval estimation.