# CHAPTER 3

# OFFLINE LEARNING

Offline learning represents the set of problems where we have a budget to run $N$ experiments, after which we have to use the information we gained to make the best decision possible based on what we learned about the function. One of the earliest versions of this problem arose in statistics under the name *ranking and selection*. The two defining characteristics of this problem class are:

- The possible decisions $x$ fall in a finite set $\mathcal{X} = \{x_1, \ldots, x_M\}$, where $x_m$ may be a categorical choice (type of drug or material, type of color), a discretized value of a continuous parameter, or a possibly sampled value of a multidmensional vector.

- We have a finite budget $N$, and we only care about the quality of the final decision rather than how well we do while we are learning. We refer to the performance at the end as the *terminal reward*.

Offline learning problems arise in many settings. We may have to choose a type of cholesterol drug, the parameters of a chemical process (e.g. temperature, relative mix of certain inputs), or the design of a manufacturing process (choice of equipment, size of buffers, routing of jobs). Testing an alternative might involve running a time-consuming computer simulation, or it might require a physical experiment. We assume that in either case, the experiment involves some noise, which means we may need to repeat the experiment several times. We assume that we have a budget that determines how many times we can perform these experiments, after which we have to take the best design and live with it.

We need to pause and comment on different interpretations of the terms “offline” and “online” (a class of problems we turn to in chapter 5). In this book, we interpret offline problems as having two distinct phases:

**The information collection phase** This is when we are running experiments to learn about the properties of a function or process, without regard to how well we are doing.

**The implementation phase** This is when we use what we have learned to design a system or process, and then we have to live with the performance of the resulting design.

By contrast, online problems represent problems that are running in the field, where we learn and implement at the same time. For example, testing medication on a patient would be an online problem; if we misdiagnose the condition, the patient will suffer. Similarly, trying to find the best price for a product requires setting a price and then receiving the associated revenue stream which may be lower than it could be, but we learn from this. This chapter focuses on offline learning. We turn to online learning, which has a separate but very rich history, in chapter 5.

This is not the only interpretation of offline vs. online. The machine learning community (to name one) uses offline learning to refer to using a batch dataset to estimate a function. By contrast, “online” refers to procedures that are sequential in nature. Implementing a process in the field is inherently sequential, hence the association of “online” with “sequential.” But, learning in the lab or a simulation (offline) can also be sequential, and hence the machine learning community will interpret this setting as online as well, while recognizing that we do not care about how well we do.

In this chapter, we are going to show how to model offline learning problems. We then illustrate how to design and evaluate different learning policies for choosing the next experiment to run. The policies we introduce in this chapter are not the most sophisticated, although they can work quite well if properly tuned. Further, we are going to introduce new policies in chapter 5 for online learning that can also be applied to offline problems.

## 3.1 THE MODEL

The learning process proceeds as follows. Let $S^n$ be our belief state, which captures the statistics we have collected from our first $n$ observations using the methods we introduced in chapter 2. We assume we have some policy $X^\pi(S^n)$ that returns a decision $x^n$ given our belief state that is given by $S^n$. We start at time $n = 0$, and make decision $x^0 = X^\pi(S^0)$ using our initial state $S^0$ (that is, our prior). Using the lookup table model we introduced in chapter 2, assume we choose $x^n = X^\pi(S^n)$ and then observe

$$W^{n+1}_{x^n} = \mu_{x^n} + \varepsilon^{n+1}. \tag{3.1}$$

If we are using a Bayesian model, our initial state $S^0$ might be that our prior is $\mu_x \sim N(\bar{\mu}_x^0, \beta_x^0)$. If we are using a frequentist model, our initial state contains no information, and we might be forced to test each $x$ at least once (later in the book, we introduce parametric belief models that require fewer experiments to create a prior).

We then observe $W_{x_0}^1$, and use this information to update our belief state to obtain $S^1$ using the methods we described in chapter 2 (note that we can use either a Bayesian or frequentist belief state). We can write the sequence of states, decisions and information as

$$(S^0, x^0, W_{x^0}^1, S^1, x^1, \ldots, x^{N-1}, W_{x^{N-1}}^N, S^N).$$

We note that our indexing reveals the information content of each variable. Thus, $S^{n-1}$ and $x^{n-1}$ are computed without seeing $W^n$.

Keeping in mind that we do not update beliefs for alternatives that are not observed, if we are using a Bayesian model, we would update our belief state using

$$\bar{\mu}_x^{n+1} = \begin{cases} \frac{\beta_x^n \bar{\mu}_x^n + \beta_x^W W_x^{n+1}}{\beta_x^n + \beta_x^W} & \text{if } x^n = x \\ \bar{\mu}_x^n & \text{otherwise,} \end{cases} \tag{3.2}$$

$$\beta_x^{n+1} = \begin{cases} \beta_x^n + \beta_x^W & \text{if } x^n = x \\ \beta_x^n & \text{otherwise.} \end{cases} \tag{3.3}$$

Our belief state with the Bayesian model would be written

$$S^n = (\bar{\mu}_x^n, \beta_x^n)_{x \in \mathcal{X}}.$$

The goal of our problem is to design a policy that we designate $X^\pi(S^n)$ that returns a decision $x^n$ which specifies the decision of what experiment to run next, which will yield the information $W^{n+1}$. If we are using a Bayesian belief model, our state is $S^n = (\bar{\mu}^n, \beta^n)$, which evolves according to a *transition function* that we denote by

$$S^{n+1} = S^M(S^n, x^n, W^{n+1}), \tag{3.4}$$

where the function $S^M(S, x, W)$ is known by names such as the transition function or the state transition model (hence the superscript "$M$"). For our Bayesian model, the transition function consists of equations (3.2) and (3.3).

If we use a frequentist model, the state variable is given by

$$S^n = (\bar{\mu}_x^n, \hat{\sigma}_x^{2,n}, N_x^n)_{x \in \mathcal{X}}$$

as developed in section 2.2, where $N_x^n$ is the number of times we have tested alternative $x$, and where the transition equations are given by equations (2.4) and (2.5).

To evaluate a policy, we have to average its performance over many different realizations. For a frequentist model, the random variables are the observations $W^1, \ldots, W^N$ which produce the final estimates $\bar{\mu}_x^{\pi,N}$. We use this to choose our best decision (or design), given by

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^{\pi,N}. \tag{3.5}$$

We then choose the policy that works the best on average. Using our frequentist model, we need to recognize two sources of uncertainty: the sequence of observations $W^1, \ldots, W^N$ which occur while we are running our experiments, and the performance of the final design when it is implemented. We can write this as

$$\begin{aligned} F^\pi &= \mathbb{E}_{W^1,\ldots,W^N} \mathbb{E}_W W_{x^{\pi,N}} & (3.6) \\ &= \mathbb{E}_{W^1,\ldots,W^N} \mu_{x^{\pi,N}}. & (3.7) \end{aligned}$$

If we are using a Bayesian model, we have to recognize one more source of uncertainty, which is the prior on the true value of $\mu$. In our Bayesian model, each $W^n$ depends on $\mu$ (as we saw in equation (3.1)). Equations (3.6) - (3.7) become

$$\begin{aligned} F^\pi &= \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mathbb{E}_W W_{x^{\pi,N}} & (3.8) \\ &= \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^{\pi,N}}. & (3.9) \end{aligned}$$

These expectations can look pretty frightening. Later in this chapter we discuss practical ways of estimating these in a straightforward way.

## 3.2 LEARNING POLICIES

Central to the concept of optimal learning is a learning policy. This is a rule that tells us which action $x$ we should take next in order to observe something new. In addition, we may also be receiving rewards or incurring costs, which have to be balanced against the value of the information being gained.

In this section, we contrast deterministic versus sequential policies, and then provide a mathematical framework for finding optimal sequential policies. Unfortunately, this framework does not provide us with practical policies that we can compute. The section closes with a presentation of a number of the more popular heuristic policies that have been used on this problem class.

### 3.2.1 Deterministic vs. sequential policies

Before we begin our presentation, it is important to make a distinction between what we call *deterministic policies* and *sequential policies*:

**Deterministic policies** These are policies that determine all the experiments to run in advance, without learning the results of any of the experiments.

**Sequential policies** These policies use the results of the $n-1st$ experiment before deciding what to do for the $nth$ experiment.

An example of a deterministic policy is where a business may decide to perform four market research studies in different parts of the country before finalizing the pricing and advertising strategy in a full roll-out to the entire country. The decision to do four studies (and their locations) is made before we have any information from any of the studies. We return to these policies in section 12.1.

By contrast, sequential policies assume that we make a decision $x^{n-1}$, then observe the result $W^n$, before making the next decision $x^n$. So, we try one path to work through the city before choosing the path we try the next day. Similarly, we see how a patient responds to one drug before trying another.

There are problem classes where deterministic policies are optimal. For example, we might be interested in running experiments that minimizes some function of the variance of the quantities that we are trying to estimate. If you take a close look at our formula for updating the variance (or equivalently the precision) in equation (3.3), we see that our estimate of the variance is a deterministic function of what we choose to measure. This means that any rule that depends purely on the variance can be solved deterministically.

Our interest is primarily in sequential policies, where the decision of what to measure next may depend on past experiments. For example, when we are trying to find the shortest path, we may decide to continue sampling a path if it remains competitive, or give up on a path if the observed travel times are simply too long. Our decisions of what to measure in this case depend on the outcomes of prior experiments.

### 3.2.2 Optimal learning policies

It is possible to provide a mathematical characterization of an optimal learning policy. We start by considering a simple shortest path problem, illustrated in figure 3.1(a), where we are in state (node) 2, considering a decision to go to state (node) 5. The field of dynamic programming tells us that the value of being in a state $S$, given by $V(S)$, satisfies Bellman's equation which is given by

$$V(S) = \max_x \big(C(S,x) + V(S^M(S,x))\big). \tag{3.10}$$

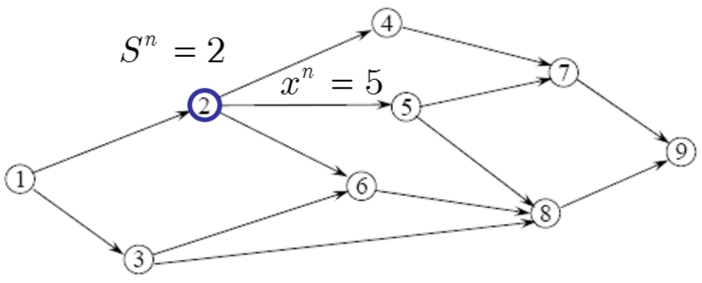
Here, $S^M(S,x)$ is known as the transition function, which determines the state we transition to if we make decision $x$. For our shortest path problem, $x$ (the decision to go to node 5) determines the downstream state deterministically. There are many problems where the downstream state involves a combination of the decision $x$ and random information $W$. For example, we could model time, where every node is a point in space and time. We might model randomness in travel times, which introduces randomness in when we arrive. In this case, we would write our transition as $S' = S^M(S,x,W)$, and write Bellman's equation as

$$V(S) = \max_x \big(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\big). \tag{3.11}$$

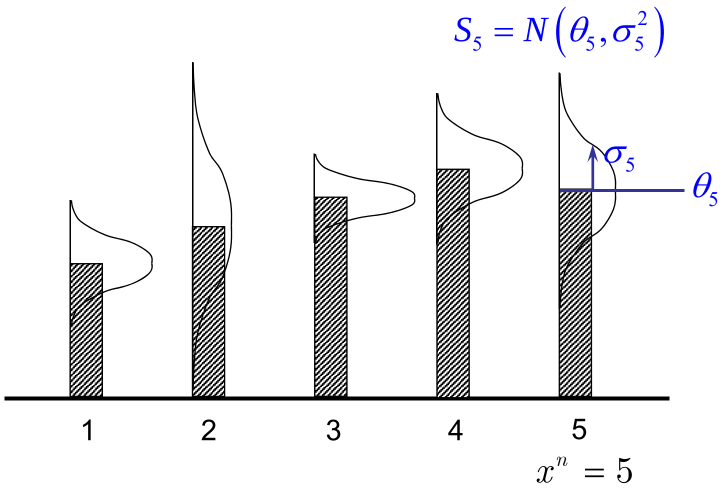
We can solve equation (3.10) or (3.11) by starting at the ending state(s) where $V(S) = 0$, and then stepping backward. This, in fact, is how shortest path problems are actually solved in practice. Once we have the value functions, our policy is given by

$$X^*(S) = \arg\max_x \big(C(S,x) + \mathbb{E}_W V(S^M(S,x,W))\big). \tag{3.12}$$

Now consider the situation in figure 3.1(b), where now our state is the belief state about the performance of each of five choices. These choices might represent five



(a)



(b)

**Figure 3.1** (a) Illustration of a shortest path problem, where we are in state (node) 2, considering going to node 5; (b) Illustration of a learning problem, where the distributions represent our current belief state, and we are considering making a decision to evaluate alternative 5.

medications for lowering blood sugar, and the distributions represent our belief about how each medication will affect the patient. In this context, the transition function $S^M(S, x, W)$ is precisely our belief updating functions (3.2) and (3.3).

The problem with our optimal policy in (3.12) is one of computation. We can solve these problems when the state $S$ is a set of discrete nodes (even if there are a lot of them). But we face a completely different problem if the state is a multidimensional set of continuous variables such as the means and precisions of all the beliefs. However, there are special cases where (3.12) can be solved, and without pointing it out, we already did this in section 1.6 when we set up and solved the decision tree for our problem of finding the next baseball player to send up to bat. The key feature of this problem was that the information was in the form of discrete (binary) random variables, which means that the belief state was also discrete.

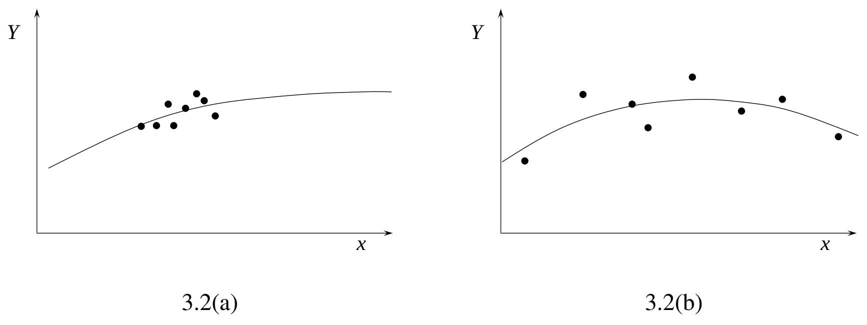


**Figure 3.2** Estimating a function where we sample closest to the point that might be best (a), versus sampling a wide range of points so that we get a better estimate of the function (b).

### 3.2.3 Heuristic policies

Our goal, ultimately, is to find the best possible policies for learning. The reality, however, is that most of the time we are happy to find good policies. Below are some popular methods that have been suggested for problems that are typically associated with discrete selection problems, which is to say that the set of experimental decisions is discrete and "not too large."

***Pure exploration*** A pure exploration strategy might sample a decision $x^n = x$ with probability $1/M$ (the probabilities do not have to be the same - they just have to be strictly positive). We would only use a pure exploration policy if we were focusing purely on estimating the value of each choice, as opposed to making a good economic decision. If we really are trying to find the best value of $\mu_x$, a pure exploration strategy means that we would spend a lot of time measuring suboptimal choices.

Pure exploration can be effective for offline learning problems, especially when the number of choices is extremely large (and especially if the experimental design $x$ is multidimensional). This is often what has been used when we are given a dataset of observations from which we have to fit a model so that we can find the best choice or design. With offline learning, it does not hurt us to observe a poor choice, and extreme choices can give us the best estimates of a function. For example, consider the problem of fitting a linear regression model of the form

$$Y = \theta_0 + \theta_1 x + \theta_2 x^2 + \epsilon.$$

Imagine that $x$ is a scalar between 0 and 10, and we believe the highest values of $Y$ are likely to be found close to the middle of the range. Figure 3.2(a) shows what happens when we draw most of our samples from a narrow range. We may get a lot of data to estimate the function around those points, but we do not get the kind of information we need to get an accurate estimate of the function, which would allow us to do the best job finding the maximum. In Figure 3.2(b), we explore a wider range of points, which allows us to do a better job of estimating the entire curve. For instance, we discover that $Y$ decreases once $x$ is large enough, whereas 3.2(a) leads us to believe that $Y$ always increases with $x$.

***Pure exploitation*** Exploitation means making the best decision given our current set of estimates (we are "exploiting" our knowledge). So, after iteration $n$, we would next measure

$$x^n = \arg\max_{x \in \mathcal{X}} \bar{\mu}_x^n.$$

This strategy would seem to focus our energy on the options that appear to be the best. However, it is very easy to get stuck measuring choices that seem to be the best, especially when we simply had some bad luck measuring the better choices.

Pure exploitation is a common strategy in online problems, where we have to live with the results of each experiment. With pure exploitation, we can always defend our choice because we are doing what we believe is the best, but we may be ignoring errors in our own beliefs.

***Excitation policies*** A popular policy in the control of engineering systems with continuous controls is to assume we have some policy that specifies the action we think is best and then adds a random noise term. For example, we might specify the rate at which gasoline flows into a combustion chamber as a function of the speed $S_t$ of a vehicle. Assume that we use the following policy (known as a control law in engineering):

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2.$$

Presumably we think this is the policy that produces the best gas mileage while maintaining the speed of the vehicle, so we would call this a pure exploitation policy. We can introduce a random noise term to induce exploration using

$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2 + \varepsilon.$$

The noise $\varepsilon$ is known in engineering as *excitation*, and it helps to learn new behaviors.

***Epsilon-greedy exploration*** A simple strategy that avoids the limitations of pure exploration and pure exploitation is to use a mixed strategy, where we explore with probability $\epsilon$ (known as the exploration rate) and we exploit with probability $1 - \epsilon$. The value of $\epsilon$ has to be tuned for each application.

Mixing exploration and exploitation is appealing because it allows you to spend more time evaluating the choices that appear to be best (to make sure this is the case) while still doing a certain amount of exploration. As our experimentation budget goes to infinity, we can still provide guarantees that we will find the best alternative because of the exploration component. But this policy still suffers from the significant limitation that when we do choose to explore, we sample from the entire population of alternatives, which may be extremely large, including choices that are clearly suboptimal.

The problem with a mixed exploration/exploitation strategy with fixed $\epsilon$ is that the correct balancing of exploration and exploitation changes with the number of iterations. In the beginning, it is better to explore. As we build confidence in our estimates, we would prefer to exploit more. We can do this by using an exploration probability $\epsilon^n$ at iteration $n$ that declines with $n$. We have to make sure it does not decline too quickly. We do this by setting

$$\epsilon^n = c/n$$

for $0 < c < 1$. If we explore, we would choose experimental design $x$ with probability $1/|\mathcal{X}|$. This means that in the limit, the number of times we will measure $x$ is given by

$$\sum_{n=1}^{\infty} \frac{c}{n|\mathcal{X}|} = \infty.$$

This assures us that we will estimate each experiment $x$ perfectly, but as the experiments progress, we will spend more time measuring what we think are the best choices.

***Boltzmann exploration*** A different strategy for balancing exploration and exploitation is known as Boltzmann exploration. With this strategy, we run experiment $x$ with probability $p_x^n$ given by

$$p_x^n(\theta^B) = \frac{e^{\theta^B \bar{\mu}_x^n}}{\sum_{x' \in \mathcal{X}} e^{\theta^B \bar{\mu}_{x'}^n}}. \tag{3.13}$$

This policy is also known as the *soft max* policy. If $\theta^B = 0$, we are going to run each experiment $x$ with equal probability (pure exploration). As $\theta^B \to \infty$, we will run the experiment with the highest value of $\bar{\mu}^n$ with probability 1 (pure exploitation). In between, we explore the better options with higher probability. Furthermore, we can make $\theta^B$ increase with $n$ (which is typical), so that we explore more in the beginning, converging to a pure exploitation strategy. Tuning such policies, however, can be tricky; we return to this below in section 3.7.

Care should be used when computing the probabilities using a Boltzmann distribution, especially if you are increasing $\theta^B$ as you progress to focus attention on the best alternatives. The problem is that the exponent $\theta^B \bar{\mu}_x^n$ can become so large as to make it impossible to evaluate $e^{\theta^B \bar{\mu}_x^n}$. A better way is to first compute

$$\bar{\mu}^n = \max_{x \in \mathcal{X}} \bar{\mu}_x^n,$$

and then compute the probabilities using

$$p_x^n(\theta^B) = \frac{e^{\theta^B (\bar{\mu}_x^n - \bar{\mu}^n)}}{\sum_{x' \in \mathcal{X}} e^{\theta^B (\bar{\mu}_{x'}^n - \bar{\mu}^n)}}.$$

This calculation can be further streamlined by excluding any choices $x$ where $\bar{\mu}_x^n$ is sufficiently far from $\bar{\mu}^n$ (for example, where $\theta^B |\bar{\mu}^n - \bar{\mu}_x^n| > 10$).

We write the Boltzmann policy by first generating a random variable $U$ that is uniform on $[0,1]$. To simplify notation, assume that $x_m = m$. Next, create a cumulative distribution $P_x^n(\theta^B) = \sum_{x'=1}^{x} p_{x'}^n(\theta^B)$. Finally, we can write our policy using

$$X^{Boltz}(S^n|\theta^B) = \arg\max_x \{x | P_x^n(\theta^B) \leq U\}. \tag{3.14}$$

Boltzmann exploration is more effective than epsilon-greedy which either chooses the very best, or chooses among the rest purely at random. Boltzmann focuses its efforts more on the choices that are at least somewhat attractive. However, it is missing an important feature that is enjoyed by the policies which follow.

***Upper confidence bounding*** UCB policies (as they are known) were developed primarily for the setting of online learning, so we are going to cover these in more depth in chapter 5. However, they represent an important class of policy, and they can be used in offline settings if they are properly tuned (the same can be said of all of the heuristic policies described in this section).

A basic version of a UCB policy (known as "UCB1" in the research literature) is given by

$$X^{UCB1,n}(S^n|\theta^{UCB}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{UCB}\sqrt{\frac{2\log n}{N_x^n}}.\right) \tag{3.15}$$

The parameter $\theta^{UCB}$ is one that we have introduced, and we will have to tune it for an offline setting (this is discussed below).

***Thompson sampling*** Thompson sampling is a simple policy based on the idea of created a *sampled* estimate of what *might* happen in the next experiment for each possible design. We illustrate using a Bayesian belief model, although a frequentist model could be used as well.

Assume that after $n$ experiments, we characterize our belief about $\mu_x \sim N(\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})$. Now, randomly sample $\hat{\mu}_x^{n+1}$ from the belief $N(\bar{\mu}_x^n, \bar{\sigma}_x^{2,n})$, where $\mathbb{E}\hat{\mu}_x^{n+1} = \bar{\mu}_x^n$. However, $\hat{\mu}_x^{n+1}$ may be higher or lower than $\bar{\mu}_x^n$. The Thompson sampling policy is then given simply by

$$X^{TS}(S^n) = \arg\max_x \hat{\mu}_x^{n+1}. \tag{3.16}$$

The randomization in choosing $\hat{\mu}_x^{n+1}$ is the mechanism by which Thompson sampling explores different alternatives, with a bias toward the choices that appear to be better.

***Interval estimation*** Imagine that instead of running the experiment that we think is best, we choose an alternative that we think might learn is the best if we were to take enough observations. With this idea, we might construct a confidence interval and then value an option based on the upper limit of, say, a $95\%$ confidence interval. Letting $\alpha$ be our confidence level and denoting by $z_\alpha$ the standard normal deviate leaving $\alpha$ in the upper tail, our upper limit would be

$$\nu_x^{IE,n} = \bar{\mu}_x^n + z_\alpha \sigma_x^n \tag{3.17}$$

where $\sigma_x^n = \sqrt{\frac{1}{\beta_x^n}}$ is the standard deviation of the distribution of our belief at time $n$. When we use an interval exploration policy, we choose the experiment $x^n$ with the highest value of $\nu_x^{IE,n}$.

Although the interpretation as the upper limit of a confidence interval is appealing, the confidence level $\alpha$ carries no particular meaning. In fact, the policy we would implement in practice looks like

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\bar{\sigma}_x^n\right), \tag{3.18}$$

where now, $\theta$ is simply a tunable parameter. It has been reported in the literature that values around 2 or 3 work best for many applications, but it is possible to construct problems where the best value may be anywhere from $0.1$ to 100 (the high values arise when the priors are really poor). Furthermore, it has been found that the algorithm can be very sensitive to the choice of $\theta$. However, if properly tuned, this policy can work extremely well in many settings.

Interval estimation introduces two important dimensions relative to policies based on exploration and exploitation. First, like Boltzmann estimation, the likelihood of measuring a choice depends on how well we think the choice may work. This means that we will avoid exploring options that seem to be genuinely bad. Second, we are more willing to explore options which we are more uncertain about. The term $z_\alpha\sigma_x^n$ has been called the "uncertainty bonus." As we explore an option with a high value of $\nu_x^{IE,n}$, $\sigma_x^n$ will decrease, which will often push us to try other options.

IE is not guaranteed to find the best option, even with an infinitely large experimental budget. It is possible for us to get some poor initial estimates of what might be the best option. If $\bar{\mu}_x^n$ is low enough for some choice $x$, we may never revisit this choice again. Our experimental work suggests that this can happen, but rarely.

## 3.3 SIMULATING OUTCOMES

Now that we have covered a menu of policies, we have to address the problem of evaluating them to determine which one is best. We are going to do this by simulating the policy, which means we have to simulate the sequence of experimental outcomes $W^1, \dots, W^N$ that result from using a policy. There are three ways to do this. The first two are independent of the policy, which allows them to be done in advance. The last is a policy-dependent representation which may seem more natural and can be easier to implement, but introduces a slight problem when comparing policies.

### 3.3.1 Policy-independent representations

For Bayesian belief models, there are two ways to represent the outcome of an experiment which are not dependent on the policy. The first models the truth $\mu$ and the experimental outcomes $W^1, \dots, W^N$ separately, while the second blends these. We start with the first method.

We use the notation $\omega$ to denote a sample realization of all the observations made from a sequence of experiments. Think of a matrix $W(\omega)$ of numbers, such that the number in the $n$th row and $x$th column represents a sample realization $W_x^n(\omega)$ of the observation $W_x^n$. Figure 3.1 illustrates this idea with three sample realizations of $W_x^n$ (three different values of $\omega$) for $n = 1, ..., 10$ and $x = 1, 2, 3$. It is useful to think of generating realizations for every possible experiment, but then we are going to choose a learning policy $X^\pi(S)$ that determines which of these realizations that

| | $\omega=\omega_1$ | | | $\omega=\omega_2$ | | | $\omega=\omega_3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ |
| 1 | 37.4 | 64.1 | 59.2 | 38.1 | 66.9 | 55.7 | 33.4 | 66.5 | 40.1 |
| 2 | 24.3 | 65.2 | 56.0 | 28.0 | 57.6 | 59.3 | 25.4 | 60.1 | 59.5 |
| 3 | 30.5 | 64.5 | 56.5 | 38.9 | 59.3 | 50.2 | 22.7 | 59.5 | 48.7 |
| 4 | 20.5 | 63.2 | 55.9 | 34.1 | 57.7 | 57.1 | 24.1 | 59.8 | 58.8 |
| 5 | 29.1 | 64.5 | 44.8 | 36.3 | 60.1 | 56.4 | 37.3 | 65.1 | 57.1 |
| 6 | 36.7 | 62.1 | 59.6 | 37.0 | 53.6 | 42.9 | 20.7 | 51.8 | 42.9 |
| 7 | 27.3 | 53.5 | 44.8 | 27.7 | 60.9 | 48.9 | 32.0 | 53.4 | 53.3 |
| 8 | 26.8 | 57.7 | 48.1 | 28.8 | 68.6 | 54.4 | 34.3 | 53.0 | 48.6 |
| 9 | 40.0 | 58.6 | 42.8 | 39.6 | 52.8 | 53.5 | 31.1 | 64.0 | 59.9 |
| 10 | 21.4 | 51.6 | 56.4 | 33.1 | 54.5 | 42.7 | 33.4 | 56.7 | 42.7 |

**Table 3.1** Three sample realizations of three alternatives over 10 observations.

| | $\omega=\omega_1$ | | | $\omega=\omega_2$ | | | $\omega=\omega_3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | $\mu_1(\omega_1)$ 32.1 | $\mu_2(\omega_1)$ 57.8 | $\mu_3(\omega_1)$ 55.8 | $\mu_1(\omega_2)$ 29.5 | $\mu_2(\omega_2)$ 62.3 | $\mu_3(\omega_2)$ 60.1 | $\mu_1(\omega_3)$ 34.4 | $\mu_2(\omega_3)$ 59.2 | $\mu_3(\omega_3)$ 59.7 |
| $n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ |
| 1 | 33.8 | 57.7 | 53.3 | 32.6 | 62.7 | 60.5 | 29.6 | 55.7 | 62.6 |
| 2 | 37.0 | 55.0 | 59.7 | 26.0 | 60.8 | 55.3 | 36.8 | 62.1 | 59.1 |
| 3 | 36.0 | 54.9 | 51.9 | 34.3 | 65.0 | 58.3 | 38.0 | 60.3 | 60.4 |
| 4 | 30.8 | 53.3 | 58.6 | 30.0 | 57.7 | 57.8 | 36.1 | 59.9 | 63.2 |
| 5 | 30.9 | 58.6 | 58.3 | 30.7 | 59.4 | 61.0 | 38.6 | 58.3 | 63.2 |
| 6 | 29.7 | 56.3 | 55.6 | 26.3 | 62.4 | 60.9 | 33.9 | 54.6 | 64.5 |
| 7 | 32.1 | 57.0 | 57.9 | 31.6 | 57.8 | 59.9 | 30.8 | 63.3 | 55.3 |
| 8 | 31.1 | 53.9 | 54.4 | 29.1 | 59.6 | 59.0 | 31.6 | 58.9 | 60.8 |
| 9 | 35.9 | 55.5 | 50.8 | 31.5 | 65.9 | 60.8 | 36.0 | 62.3 | 62.9 |
| 10 | 35.4 | 60.5 | 54.7 | 33.7 | 62.4 | 55.9 | 38.9 | 54.9 | 64.2 |

**Table 3.2** Three sample realizations of both a truth $\mu$ and a set of sample realizations $W$ drawn from this truth.

we are actually going to see. That is, we only get to see $W_x^n$ if we choose to observe $x = X^\pi(S^n)$.

Since the true values $\mu_x$ are also random variables (in our Bayesian model), we can let $\psi$ be a sample realization of the truth. That is, $\mu(\psi)$ is a particular set of truth values $\mu_{x_1}(\psi), ..., \mu_{x_M}(\psi)$ for the different alternatives. Our Bayesian model makes the fundamental assumption that $\mu_x \sim \mathcal{N}(\bar{\mu}_x^0, \beta_x^0)$, which means that we assume our prior distribution is assumed to be accurate *on average*. Therefore, the sample path $\psi$ is generated from the prior distribution. This is sometimes known as the *truth from prior* assumption. Of course, we have to recognize that our prior may not be accurate, but we generally assume it is unbiased.

| | $\omega = \omega_1$ | | | $\omega = \omega_2$ | | | $\omega = \omega_3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| $n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ | $W_1^n$ | $W_2^n$ | $W_3^n$ |
| 1 | 37.4 | - | - | - | 66.9 | - | - | 66.5 | - |
| 2 | - | 65.2 | - | 28.0 | - | - | - | 60.1 | - |
| 3 | 30.5 | - | - | - | - | 50.2 | 22.7 | - | - |
| 4 | - | - | 55.9 | 34.1 | - | - | - | - | 58.8 |
| 5 | - | 64.5 | - | - | 60.1 | - | 37.3 | - | - |
| 6 | - | 62.1 | - | 37.0 | - | - | - | 51.8 | - |
| 7 | - | - | 44.8 | - | - | 48.9 | - | - | 53.3 |
| 8 | 26.8 | - | - | - | - | 54.4 | 34.3 | - | - |
| 9 | - | - | 42.8 | - | 52.8 | - | 31.1 | - | - |
| 10 | - | 51.6 | - | 33.1 | - | - | - | 56.7 | - |

**Table 3.3** Three sample realizations where we only model actual observations generated by following a policy.

The second strategy for representing the outcome of a set of experiments is to combine the uncertainty about the truth and the uncertainty about experiments into a single space of outcomes. Table 3.2 illustrates how we might represent three sample realizations. Here, $\omega$ represents both a realization of the truth $\mu$ as well as a realization of the observations $W_x^n$ drawn from this truth.

### 3.3.2 Policy-dependent representation

The third approach for representing the outcome of a set of experiments (and perhaps the most natural) is to model the sequence of decisions $x^n$ and corresponding experimental outcomes $W^{n+1}$. If we are using a frequentist representation, a sample realization would be

$$(x^0, W^1, x^1, W^2, \ldots, x^{N-1}, W^N).$$

In this representation, we let $\omega$ be a single sample path of decisions and outcomes. Since the decisions depend on what policy we are following, we would write our set of all possible outcomes as $\Omega^\pi$. One way to represent sample paths is depicted in table 3.3, where we only show the observations of alternatives we actually observe.

### 3.3.3 Discussion

The policy-independent representations require that we pre-generate all the sample realizations and store these in a file (typically). While this requires more work, it allows us to compare different policies on the same sample realizations, which simplifies the comparison of policies. By contrast, it is easier to generate outcomes $W_x^n$ on the fly while following a policy, but this typically means comparing policies using different sample outcomes.

With both methods, if we are using a Bayesian model, we can generate truths $\mu(\psi)$ independently of the experimental outcomes $W^1(\omega), \ldots, W^N(\omega)$, or we can generate them simultaneously, creating a single sample realization $(\mu(\omega), W^1(\omega), \ldots, W^N(\omega)$. For the same number of simulations, the latter approach will produce a better mixture of truths and experimental outcomes.

If we are using a frequentist belief model, we skip the sampling of a truth. Instead, we have to sample the experimental outcomes $W$ from an assumed distribution.

## 3.4 EVALUATING POLICIES

We now want to estimate the performance of policy $X^\pi(S)$. Our challenge is to evaluate a policy which we might do using our Bayesian objective (equation (3.9)) given by

$$F^\pi \quad = \quad \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^{\pi,N}}.$$

The problem is that we cannot actually compute these expectations. For this reason, we have to use Monte Carlo simulation to approximate the value of a policy.

Recall that $x^{\pi,N}$ is the best decision based on our final estimates $\bar{\mu}_x^{\pi,N}$ computed when we follow policy $\pi$ with an experimental budget of $N$ observations (see equation (3.5)). We refine this notation by indexing it by the sample truth and the sample of observations, which gives us

$x_x^{\pi,N}(\psi, \omega)$ = The optimal decision based on the estimates $\bar{\mu}_x^{\pi,N}$ for each alternative $x$ when the truth is $\mu(\psi)$ and the observations are $W^1(\omega), \ldots, W^N(\omega)$.

There are two ways we can evaluate the performance of a policy. The first is based purely on the actual estimates of the value of each alternative. Let

$\bar{\mu}_x^{\pi,N}(\psi, \omega)$ = The estimate of $\mu_x$ if $\mu_x(\psi)$ is the true value of $\mu$, we observe $W(\omega)$ while following policy $X^\pi(S)$ with a budget $N$.

Remember that each observation $W_x^n(\omega)$ is generated using

$$W_{x^n}^{n+1}(\omega) \quad = \quad \mu_{x^n}(\psi) + \varepsilon^{n+1}(\omega)$$

for a given truth $\mu_x(\omega)$. With this, we finally can compute a sample realization of the value of a policy using

$$\begin{aligned} F^\pi(\mu(\psi), W(\omega)) &= \max_{x \in \mathcal{X}} \bar{\mu}_x^{\pi,N}(\psi, \omega) \\ &= \bar{\mu}^{\pi,N}_{x^{\pi,N}(\psi,\omega)}(\psi, \omega). \end{aligned} \tag{3.19}$$

Equation (3.19) is simply using our final estimate of each $\mu_x$ to evaluate how well the policy has performed. Naturally, we need to average over different truths $\mu(\psi)$ and sample observations $W^1(\omega), \ldots, W^N(\omega)$. If we sample $L$ truths $\mu(\psi_\ell)$ and $K$ experiments $W^1(\omega_k), \ldots, W^N(\omega_k)$, then we can compute an estimate of the performance of the policy using

$$\overline{F}^\pi = \frac{1}{L} \sum_{\ell=1}^{L} \left( \frac{1}{K} \sum_{k=1}^{K} F^\pi(\mu(\psi_\ell), W(\omega_k)) \right). \tag{3.20}$$

We can also use the representation illustrated in table 3.2 where we sample the truth $\mu$ and the outcomes $W^1, \ldots, W^N$ simultaneously, rather than in the nested fashion that we use in our estimate in equation (3.20). In this case, we let $\bar{\mu}_x^{\pi,N}(\omega)$ be the estimate of $\mu_x$ when we sample the combination of truth and experimental outcomes $(\mu(\omega), W^1(\omega), \ldots, W^N(\omega))$. We also let $x^{\pi,N}(\omega)$ be the optimal solution given by

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

Using this representation, our estimate of the value of a policy would be written

$$\overline{F}^{\pi} = \frac{1}{K}\sum_{k=1}^{K} F^{\pi}(\mu(\omega_k), W(\omega_k)). \tag{3.21}$$

An important feature of equation (3.20) is that it requires nothing more than sampled observations that could come from any field source. However, it is often the case that we are using simulations to evaluate and compare policies that would then be used in the field. When we are simulating a policy, we do this using a simulated truth $\mu(\psi)$. When this is the case, then we can evaluate the policy using the actual truth, equation (3.19) becomes

$$\begin{aligned} F^{\pi}(\mu(\psi), W(\omega)) &= \max_{x\in\mathcal{X}} \mu_x(\psi,\omega) \\ &= \mu_{x^{\pi,N}(\psi,\omega)}(\psi,\omega). \end{aligned} \tag{3.22}$$

We then use this method for computing $F^{\pi}(\mu(\psi), W(\omega))$ in equation (3.20). Equation (3.22) avoids the noise inherent when using the estimates $\bar{\mu}^{\pi,N}_{x^{\pi,N}(\psi,\omega)}(\psi,\omega)$.

A particularly powerful idea is to compute what is known variously as the *opportunity cost* or the *regret* which calculates how much worse we are doing than the optimal solution. Using our ability to use the known truth (because we are simulating it), we can find the difference between how well we do using our estimates to find the best, and what is truly the best. This is given by

$$\bar{R}^{\pi} = \frac{1}{L}\sum_{\ell=1}^{L}\left(\frac{1}{K}\sum_{k=1}^{K}\left(\mu_{x*}(\psi) - \mu_{x^{\pi}(\mu(\psi_\ell),W(\omega_k))}(\psi)\right)\right). \tag{3.23}$$

where $\mu_{x*}(\psi) = \max_x \mu_x(\psi)$ is the best we can do for a particular truth. The regret has a lower bound of zero, which provides a nice reference point.

There are other ways to evaluate a policy which we discuss in greater depth in chapter 6. However, most of the time we will use either the expected performance or, quite frequently, the opportunity cost (or regret).

We suggest that the best environment for identifying good learning policies is inside the computer, where you can assume a truth and then try to discover the truth. Once you have decided on a good learning policy, you can go to the field where the truth is unknown, and you only have access to the estimates $\bar{\mu}_x^{\pi,N}$.

## 3.5 HANDLING COMPLEX DYNAMICS

It is useful to note that the process of evaluating policies is quite robust to the underlying dynamics or choice of belief model. For example, in chapter 2 we introduced updating equations if we have correlated beliefs. Our process of simulating policies is quite robust to the introduction of correlated beliefs, or more complex information processes.

To see this, imagine that we have a current belief state $S^n = (\bar{\mu}^n, \Sigma^n)$ where $\bar{\mu}^n$ is a vector with element $\bar{\mu}^n_x$ (as we have assumed above), and where $\Sigma^n$ is our estimate of the covariance matrix in our beliefs, with element

$$\Sigma_{xx'} = Cov^n(\mu_x, \mu_{x'}).$$

This is the covariance in our belief about the truths $\mu_x$ and $\mu_{x'}$ after we have observed $W^1, \ldots, W^n$. After choosing $x^n$ and observing $W^{n+1}$, we can then use the updating equations for correlated beliefs (see section 2.3.2) to produce updated estimates $\bar{\mu}^{n+1}$ and $\Sigma^{n+1}$.

Similarly, our information process $W$ may have its own complex dynamics. For example, we might use a time series equation of the form

$$W^{n+1} = \theta_0 W^n + \theta_1 W^{n-1} + \theta_2 W^{n-2}.$$

Incorporating more complex time series models that depend on trailing observations does not affect our process of testing and comparing policies.

## 3.6 EQUIVALENCE OF USING TRUE MEANS AND SAMPLE ESTIMATES*

It turns out that our evaluation of the policy using sampled estimates, given in equation (3.19), is the same (on average) as that computed using the real truth, given in equation (3.22). This result means that

$$\mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \max_x \bar{\mu}^{\pi,N}_x. \tag{3.24}$$

Below, we are going to use $\mathbb{E}$ instead of the nested $\mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}$. When using the single expectation operator, it is easiest to think of this as using the combined sampling of both the truth and the outcomes as we illustrated in table 3.2.

To demonstrate equation (3.24), recall that $\mathbb{E}^N \mu_x = \bar{\mu}^{\pi,N}_x$ for any fixed $x$. We use the tower property of conditional expectation, which simply means that we can nest expectations. For example, imagine that we have random variables $A$ and $B$, Assume that the expectation $\mathbb{E}$ refers to taking the expectation over the combination of $A$ and $B$, which we might write as $\mathbb{E}_{A,B}$. The tower property can be written

$$\mathbb{E}X = \mathbb{E}_{A,B}X = \mathbb{E}_A \mathbb{E}_{B|A} X.$$

For our problem, we can think of the truth $\mu$ as the random variable $A$, and then let the sequence $W^1, \ldots, W^N$ be the random variable $B$. The tower property allows us to write

$$\mathbb{E}\mu_{x^\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \mu_{x^\pi} = \mathbb{E}\bar{\mu}^{\pi,N}_{x^\pi},$$

because $x^\pi = \arg\max_x \bar{\mu}^{\pi,N}_x$ is known at time $N$ (that is, it is fixed from the point of view of our time-$N$ beliefs). However, $\bar{\mu}^{\pi,N}_{x^\pi} = \max_x \bar{\mu}^{\pi,N}_x$ by definition of $x^\pi$.

There is an interesting corollary to this result. Denote by $\chi$ an "implementation policy" for selecting an alternative at time $N$. We can think of $\chi$ as a function mapping the belief state $S^N$ to an alternative $\chi(S^n) \in \{x_1, ..., x_M\}$. Then,

$$\max_{\chi} \mathbb{E}\mu_{\chi(S^N)} = \max_x \bar{\mu}_x^{\pi,N}.$$

In other words, the optimal decision at time $N$ is always to select $\bar{\mu}_x^{\pi,N}$. If we have no more opportunities to learn, the best possible decision we can make is to go with our final set of beliefs. This result addresses the issue of why most of our policies seek to maximize $\max_x \bar{\mu}_x^{\pi,N}$ in some way, even though what we really want to learn is the unknown true value $\max_x \mu_x$.

## 3.7 TUNING POLICIES

Most heuristic policies have a tunable parameter. Perhaps the simplest illustration is the interval estimation policy, which we restate for easy reference as

$$X^{IE}(S^n|\theta^{IE}) = \arg\max_x \left(\bar{\mu}_x^n + \theta^{IE}\sigma_x^n\right).$$

We now have the problem of tuning $\theta^{IE}$ so that the policy performs at its best. When working with a specific policy such as interval estimation, the problem of searching over policies $\pi$ becomes one of searching over values of $\theta^{IE}$.

We generally assume that we are tuning a policy using a simulator, which means that we can evaluate our policy using the actual (simulated) truth as we did in equation (3.22). Instead of writing the performance of the policy as $\overline{F}^{\pi}$, we can write it as $\overline{F}(\theta^{IE})$. Now we can write our tuning problem as

$$\max_{\theta^{IE}} \overline{F}(\theta^{IE}).$$

It should not be lost on us that the problem of designing a good policy for learning is itself a learning problem. An important difference between our tuning problem and the learning problem that we really want to solve is that we assume that we are using a fairly fast simulator to evaluate a policy.

If we do very accurate simulations, we can treat $\overline{F}(\theta^{IE})$ as if it were deterministic and unimodular (which means it has a single maximum), which makes it fairly easy to search. In chapter 10, we review a number of methods for efficiently searching for the maximum of a scalar function, even when the simulations are noisy.

Often, the problem of tuning a policy involves performing a one-dimensional search to find the best value of a single parameter such as $\theta^{IE}$. In some settings (interval estimation is one of them) we have a rough idea of the magnitude of the tunable parameter. In the case of interval estimation, $\theta^{IE}$ is unitless, so we know it is on the order of 1, but might be 0.1 or perhaps as large as 10. However, the tunable parameter for the Boltzmann policy has units, and as a result might be anywhere from $10^{-5}$ to $10^5$. For this reason, it is a good idea to first search across orders of magnitude, and then refine the search over smaller ranges.

To appreciate the importance of tuning, we tuned two policies on a library of nine problems. The first policy was interval estimation, while the second is a class of upper confidence bounding policy called UCB-E, which is given by

$$X^{UCBE,n}(\theta^{UCBE}) = \arg\max_x \left( \bar{\mu}_x^n + \sqrt{\frac{\theta^{UCBE}}{N_x^n}} \right). \tag{3.25}$$

where $N_x^n$ is the number of times we have observed alternative $x$. We note that while the tunable parameter $\theta^{IE}$ is unitless, the parameter for the UCBE policy, $\theta^{UCBE}$, is not.

Table 3.4 shows the tuned values of these parameters for each of nine test problems. We see that the variation in $\theta^{IE}$, which ranges from 10 to $10^{-2}$, is much narrower than the variation of $\theta^{UCBE}$, which ranges from $10^3$ to $10^{-4}$. We also note that tuning matters; the performance of a policy can vary quite a bit as the tunable parameter is adjusted.

| Problem class | IE | UCBE |
|---|---|---|
| P1 | 0.0994 | 2571 |
| P2 | 0.0150 | 0.319 |
| P3 | 0.0187 | 1.591 |
| P4 | 0.0109 | 6.835 |
| P5 | 0.2694 | .00036 |
| P6 | 1.1970 | 1.329 |
| P7 | 0.8991 | 21.21 |
| P8 | 0.9989 | 0.00016 |
| P9 | 0.2086 | 0.001476 |

**Table 3.4** Illustration of tuned parameters for interval estimation (IE) and the UCB policy UCB-E over nine different problems.

We are going to assume that we always have access to a simulator for the purpose of tuning our policy. However, this presumes that our simulator captures the behavior of the real environment in which we are doing learning. Needless to say, this will introduce errors.

There are settings where simulators are simply not available. This means that we have to do tuning in the field, in the same expensive environment where we also have to do learning. As of this writing, policy tuning in the field remains an open research question.

## 3.8 EXTENSIONS OF THE RANKING AND SELECTION PROBLEM

Our basic ranking and selection problem, characterized by a discrete set of alternatives $\mathcal{X} = \{x_1, \dots, x_M\}$ and a lookup table belief model, can be extended in a number of ways to produce an almost unlimited number of problem variations. Below we list some of the more popular variations.

**Decisions** There are a number of different characterizations of decisions:

- Binary $\mathcal{X} = \{0, 1\}$, which arises in A/B testing of websites (old versus new), or stopping problems (continue/stop).
- Finite set, where $\mathcal{X} = \{x_1, \ldots, x_M\}$, which is the setting we have focused on above.
- Subsets, where we have $K$ items (such as members of a basketball team), where we have to choose 5 players. If $K = 10$, some elements of our set might look like:

$$\begin{gathered}(0,1,1,0,0,1,0,1,1,0)\\(0,1,0,0,0,1,1,0,1,0)\\(1,0,1,0,0,1,0,1,0,1)\\(0,0,1,1,0,0,1,1,1,0)\\(1,0,1,0,1,1,0,0,1,0)\end{gathered}$$

  The distinguishing feature of subset selection problems is that the number of alternatives may be extremely large.
- Continuous scalar, as in $\mathcal{X} = [a, b]$. This might describe the price of a product, concentration of an additive or dosage of a drug.
- Vector-valued continuous, which we might write as $x = (x_1, \ldots, x_K)$ where each $x_k$ is a continuous scalar.
- Vector valued discrete, where each $x_k$ has to be 0 or 1. This is similar to the subset selection problem, except that it may be limited by more complex constraints.
- Multiattribute. We might be choosing among people, who are each characterized by a number of attributes (gender, age, weight, ethnicity, education). Alternative, we might need to choose to target a particular population for an ad campaign, where a population group might be characterized by country or region, the device they are using (laptop or smartphone), their gender and age. The number of attributes can be quite large, but we might be able to exploit a hierarchical structure.

**Types of noise** Above we assumed that our random noise $W$ was normally distributed, but we have a library of distributions to choose from, including

**Unknown distribution** - We may start with the realization that we simply do not know the distribution. For example, we may not even know if the proper dosage is .01mg, .1, mg, 1 mg, 10 mg.

**Binomial** - This might describe settings where the outcome is a success or failure, or whether a drug is judged to be a success (allowing us to go to market) or not.

**Exponential family** - This includes a number of distributions such as normal, exponential, log-normal, gamma, chi-squared, beta, Bernoulli, Poisson, and geometric.

**Uniform distributions** - We may know that a random variable is bounded. For example, the energy generated by a 2mw wind turbine may be reasonably approximated as a uniform distribution between 0 and 2mw.

**Heavy-tailed distributions** - This might include the Cauchy distribution (which has infinite variance) or spikes, as occurs with electricity prices, which can jump from \$20 per megawatt-hour to \$300 per mwh in a matter of minutes, and then drop back.

**Mixture distributions** - We randomly choose an observation from one of two distributions, one with a relatively smaller variance, and one with an extremely high variance.

**Bursts** - Bursts can arise when the wind rises during a storm, a product briefly becomes popular through word of mouth, or a disease pops up and spreads before it is contained.

**Rare events** - These are events that may occur and which we have to prepare for, even though it is quite rare. Examples include major floods, a tree falling on your house, or being robbed.

**Offline vs. online learning** In this chapter we have considered only offline learning, where we are only interested in the performance of the decision after all experiments have been conducted. In chapter 5 we make the transition to online learning, where we wish to maximize the cumulative rewards that arise while we are learning. These two problem classes have evolved since the 1950's in very distinct communities, but are actually quite closely related.

**Experimentation costs and budgets** There are settings in the internet where we can try different alternatives quite quickly, often making it possible to consider very high dimensional choice problems (such as which movies to display on a users account). There are many other settings where experiments are quite expensive, sharply limiting the number of experiments. For example, a scientist may be able to test 20 or 30 compounds over the course of a summer before his budget runs out. A business might want to test market features of a product, but it may require several months to observe sales. A patient may require a month before we know how she responds to a new diabetes medication.

**Switching cost** It may be easy to switch from one design $x$ to another $x'$, as an internet retailer might do when testing different prices for a product. There are many settings, however, where there is a switching cost. For example, we may be taking samples from people to test for the presence of a disease in a location. Moving to another location involves a switching cost.

**Transient performance** We have assumed that the underlying truth $\mu$ is unknown but constant. However, the market response to a price may change as a result of decisions made by competitors; the performance of an athlete may improve as he gets more playing time; and the impact of a drug may drop as a disease acquires immunity.

**Transient alternatives** A standard assumption is that the set of alternatives $\mathcal{X}$ is static, but even this is not always the case. We can only try a restaurant if we can get a reservation; we can only learn about the performance of a storm sensor if there is a storm; and we can only test the efficacy of a drug for a particular medical condition if we can find a patient that meets our criteria.

**Belief models** We have illustrated our learning with a lookup table belief model, but these become clumsy when the number of alternatives is very large (or continuous). Later we are going to consider other classes of linear and nonlinear models.

Needless to say, the number of variations of learning problems is quite large.

## 3.9 BIBLIOGRAPHIC NOTES

Section 3.1 - We present here a standard Bayesian framework for ranking and selection: see e.g. Gupta & Miescke (1996) or Chick (2006). Our presentation is based on the measure-theoretic idea of random variables as functions on a space of outcomes or sample paths; although measure theory is far outside the scope of this book, interested readers are directed to Cinlar (2011), a definitive rigorous exposition of measure-theoretic probability.

Section 3.2 - The design of policies for taking observations (or experiments) of noisy functions has its roots in the 1950s and 1960s. It evolved originally under the umbrella of stochastic optimization over a continuous domain from the seminal paper of Robbins and Monro (Robbins & Monro 1951), but this literature did not focus on the issue of maximizing the information gained from each observation (there was more attention on asymptotic convergence than rate of convergence). The ranking and selection community evolved with a focus on the problem of finding the best out of a set of discrete alternatives; see D. R. Barr (1966) for an early review, and Fu (2002) for a more current review. The challenge of collecting information in an optimal way has its roots in DeGroot (1970), which appears to give the first presentation of optimal learning as a dynamic program (but without an algorithm). Interval estimation was introduced by Kaelbling (1993). The adaptation of interval estimation using Chernoff bounds was done by Streeter & Smith (2006). Epsilon-greedy is described in Sutton & Barto (1998), with an analysis of convergence properties given in Singh et al. (2000). There is an emerging area of research which uses the concept of upper confidence bounding (UCB), originally developed for online problems, in an offline setting. We introduce the idea of upper confidence bounding in chapter 5 for online ("bandit") problems. Drawing on this framework, Audibert & Bubeck (2010) present a frequentist approach to ranking and selection with finite alternatives, with provable guarantees.

### PROBLEMS

**3.1** We wish to find a good learning policy to solve the problem in Table 3.5 in an offline setting.

a) Briefly describe the epsilon-greedy policy, the Boltzmann policy and the interval-estimation policy. Evaluate each policy in terms of its ability to capture important characteristics of a good learning policy.

b) Define the expected opportunity cost (EOC). Describe in words how you would approximate the EOC (since computing it exactly is impossible) using Monte Carlo simulation.

c) Let $\omega^n$ be the index for the $nth$ sample realization of the random observations. Give an expression for the EOC for some policy $\pi$ and give a precise formula for the confidence interval for the true performance of a policy $\pi$.

| Iteration | A | B | C |
|---|---|---|---|
| Prior $(\mu_x, \beta_x)$ | (7,.05) | (3,.02) | (4,.01) |
| 1 | 9 | - | - |
| 2 | - | - | 6 |
| 3 | - | 1 | - |

**Table 3.5** Three observations, for three alternatives, given a normally distributed belief, and assuming normally distributed observations.

**3.2** This exercise requires that you test an exploration policy in MATLAB. You will need to download two files from the course website:

```
http://optimallearning.princeton.edu/exercises/exploration.m
http://optimallearning.princeton.edu/exercises/explorationRun.m
```

The MATLAB file `exploration.m` executes a pure exploration policy for a general ranking and selection problem. The file `explorationRun.m` creates the data for a problem with 10 alternatives, where it simulates 1000 truths and 50 samples per truth. The program `exploration.m` computes the average opportunity cost "o_cost" and the standard deviation "se_result."

a) Write out the meaning of "o_cost" mathematically using the notation we have been using in the course.

b) The standard deviation "se_result" is the standard deviation of what variable?

c) Construct a 95 percent confidence interval for the value of the exploration policy, and modify the code to produce this confidence interval.

**3.3** In this exercise you have to implement the Boltzmann learning policy, which you should model after the `exploration.m` routine in the previous exercise.

a) Create a new file `boltzmann.m` which implements the Boltzmann learning policy. Keep in mind that you will have to introduce a tunable parameter $\rho$. The Boltzmann policy gives you the probability that you should sample an alternative. Imagine you have three alternatives, and the Boltzmann distribution gives you sampling probabilities of .2, .5 and .3. To sample from this distribution, generate a random number between 0 and 1. If this number is less than or equal to .2, you choose the first alternative; if it is between .2 and .7, choose the second alternative; otherwise choose the third. You will need to generalize this for an arbitrary number of alternatives.

b) Using $N = 5000$, vary $\theta$ over .001, .01, .1, 1.0, until it appears that you have found the range which bounds the best value of $\rho$. For each value of $\theta$ that you try, record it in a table and report the value of the policy and the standard error. Narrow your range until you begin getting results that are indistinguishable from each other.

c) Compare the performance of the best Boltzmann policy that you can find to a pure exploration policy that was developed in exercise 3.2.

**3.4** Implement the interval estimation policy using the `exploration.m` routine provided in exercise 3.2.

a) Create a file called `ie.m` which implements the interval estimation policy. Again, you will need a tunable parameter $z_\alpha$ which you might call $zalpha$.

b) The optimal value of $z_\alpha$ will be in the range from 0 to 5. Search over this range, first in increments of 1.0, and then in smaller increments, until you again are unable to distinguish between values (continue using $N = 5000$).

c) Compare the performance of the best Boltzmann policy that you can find to a pure exploration policy that was developed in exercise 3.2.

**3.5** This exercise builds on the exploration policy, Boltzmann policy and interval estimation policy that was implemented in exercises 3.2, 3.3 and 3.4.

a) Run each policy using $N = 5000, M = 50$ and report the confidence intervals, using your best estimate of the tunable parameters. Can you conclude that one policy is better than the other two? If so, which one? If not, use the fact that the standard deviation declines inversely with $\sqrt{N}$ (it also decreases inversely with $\sqrt{M}$, but we are going to hold $M$ constant for our study). Use this to determine how small you need the standard error to be, and then how large $N$ needs to be to produce a confidence interval that is small enough to conclude that one policy is best. You have may to repeat this exercise a few times for the numbers to settle down.

b) Now set $N = 1, M = 1$ and run each policy 10 times, reporting the actual outcome (noticed that you will not get a standard error in this case). This simulates the application of a learning policy in a specific situation, where you put it into practice (but we are going to pretend that you can replicate this 10 times). Record how often each policy discovers the best alternative (o_cost = 0). Comment on the likelihood that your best policy would outperform the other two policies on a single sample path.

**3.6** You have to choose the best of three medications to treat a disease. The performance of each medication depends on the genetic characteristics of the individual. From the perspective of this medication, people can be divided into five genetic subgroups. If a doctor knew a patient's subgroup, he would know the medication he should use, but this information is not available. Lacking this information, the doctor has to resort to trial and error. Complicating the process is that measuring the performance of a medication involves a certain level of noise.

Table 3.6 gives the average performance of each type of medication on the five patient types, based on extensive prior records. The five patient types occur with equal probability in the population. The doctor will typically test a medication on a patient for one month, after which a blood test provides a measure of the performance of the medication. But these experiments are not precise; the error between the experiment and the actual impact of the medication is normally distributed with a standard deviation of 2.2 (we assume this is constant for all medications and patient types).

| | Patient type | | | | |
|---|---|---|---|---|---|
| Medication | A | B | C | D | E |
| M1 | 6.2 | 7.3 | 5.4 | 7.2 | 5.4 |
| M2 | 8.4 | 6.9 | 6.8 | 6.6 | 4.2 |
| M3 | 5.2 | 5.8 | 6.3 | 5.5 | 3.7 |

**Table 3.6** The performance of each type of medication for each type of patient.

a) What is the prior vector $\bar{\mu}^0$ that you would use for this problem, given the data in the table?

b) What is your prior probability distribution for the performance of medication M1? Note: It is *not* normally distributed.

c) Test each of the following policies below. You may use the MATLAB routines developed in the previous exercises, or perform the exercise manually with a budget of $N = 10$ observations. If you are using the MATLAB routines, use a budget of $N = 50$ observations.

   1) Pure exploitation.
   2) Boltzmann exploration, using scaling factor $\rho = 1$.
   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.

   For each policy, report the average performance based on 100 trials, and compute a 95 percent confidence interval.