CHAPTER 6

ELEMENTS OF A LEARNING PROBLEM

By now we have covered some elementary classes of learning problems. Fortunately, these describe a very broad class of applications. But there are some important problem classes that we have not yet discussed, and which we cannot solve using the tools presented so far. However, before we launch into what can seem to be a list of scattered applications, it is useful to lay out a more comprehensive modeling framework that at least hints at how we might deal with the major classes of unsolved problems. This presentation will motivate the problems we consider in the remainder of the volume, but will also serve potentially as areas for further research.

The dimensions of any stochastic, dynamic problem can be organized along five core dimensions:

1) States variables - For learning problems, these capture our belief about unknown parameters.

2) Actions (or decisions or controls) - These are the experimental choices we can make.

3) Exogenous information - This is the new information we obtain as the result of an experiment.

4) The transition function - For learning problems, these are the equations that update our beliefs given the results of an experiment.

5) The objective function - This captures our performance metric, and is the basis for how we evaluate a learning policy.

This presentation will provide a general framework that allows us to approach virtually any sequential decision problem, and certainly any learning problem. This will lay the foundation for the more complex settings we are going to address later in this book.

6.1 THE STATES OF OUR SYSTEM

The state S_t (or S^n) of a system can be described as consisting of all the information needed from history to model the system from time t (or experiment n) onward. This means it has the information we need to make a decision, compute our performance metrics, and compute the transition to the next state. For example, the transition equations for updating beliefs

make a decision, compute the objective (contributions and rewards), and compute the transition to the next state. Elements of a state variable can include the following types of information:

- The physical state - These are variables that describe the locations, quantities and status of people, equipment and goods. In most applications, decisions directly impact the physical state.
- The information state - The information state captures other information needed to model the problem. In most applications, this information arrives exogenously to the system, such as weather, prices, customer demands or changes in technology.
- The belief state - This captures our distribution of belief about quantities that we do not know perfectly. For example, if we are using an independent normal-normal model, the belief state consists of a vector of means and a vector of variances (or precisions). For pure learning problems, the belief state is the only state variable, but later we consider more general problems.

We might designate the state as S_t if it is the information available before we make a decision x_t at time t after which we learn W_{t+1} between t and $t+1$. Or we might use S^n as the state after the nth experiment which is used to make decision x^n, which are the choices needed to run the $n+1st$ experiment. Either way, S_t or S^n is known as the *pre-decision state*.

There are times when it is useful to define the post-decision state S_t^x (or $S^{x,n}$), which is the state immediately after we make a decision, but before any new information arrives. The key difference is that S_t^x no longer includes any information needed to make the decision x_t (or x^n). Instead, S_t^x only carries the information that is needed at time $t+1$, which would always include our belief state.

Up to now, our problems (ranking and selection and the bandit problem) have been characterized by just a belief state, which is used only by the policy. To help with this distinction, we identify two classes of functions (or problems):

State-independent functions These are problems where the function we are optimizing does not depend on our state variable. For example, imagine that we are trying to optimize a newsvendor problem:

$$\max_x \mathbb{E}F(x,W) = \mathbb{E}[p\min(x,W) - cx].$$

Assume that we do not know the distribution of W, but we are allowed to observe actual sales after each iteration. With each iteration, we learn more about the function $\mu_x = \mathbb{E}F(x,W)$, but this information is used only to make a decision. After day t, we would write our (Bayesian) belief about the function $\mathbb{E}F(x,W)$ as

$$S_t = B_t = (\bar{\mu}_{tx}, \beta_{tx})_{x\in\mathcal{X}}.$$

State-dependent functions Now imagine that the prices change from day to day, and that on day t, we are first shown the price p_t before we have to make our decision where we wish to solve

$$\max_x \mathbb{E}\{F(x,W)|p_t\} = \mathbb{E}[p_t\min(x,W) - cx].$$

For this version, our state variable includes both p_t as well as our beliefs $B_t = (\bar{\mu}_{tx}, \beta_{tx})_{x\in\mathcal{X}}$ about the function $\mathbb{E}\{F(x,W)|p_t\}$, so we would write our state as

$$S_t = (p_t, B_t).$$

Some call this a *contextual learning problem* (in some communities this is known as a *contextual bandit problem*) since we have to learn the belief state B_t in the "context" of the price p_t. If p_t is independent of p_{t-1}, then the post-decision state would be

$$S_t^x = B_t,$$

since the belief state does not change until we have observed the outcome of the next experiment, and p_t is irrelevant at time $t+1$. However, we might have a price process that evolves according to

$$p_{t+1} = \theta_0 p_t + \theta_1 p_{t-1} + \theta_2 p_{t-2} + \varepsilon_{t+1}.$$

In this case, our pre-decision state would be

$$S_t = ((p_t, p_{t-1}, p_{t-2}), B_t).$$

The post-decision state would then be

$$S_t^x = ((p_t, p_{t-1}), B_t),$$

since we no longer need p_{t-2} when we advance to time $t+1$.

We could also introduce a physical state by assuming that leftover newspapers are held until tomorrow. Let R_t be the newspapers on hand at the beginning of day t, where

$$R_{t+1} = R_t + x_t - \min(x_t + R_t, W_{t+1}).$$

Returning to the setting where p_t does not depend on history, our state variable now includes both the physical state R_t, and the price p_t, as well as our belief state B_t about the function:

$$\max_x \mathbb{E}\{F(x,W)|p_t\} = \mathbb{E}\{p_t \min(x+R_t, W) - cx\}.$$

Thus, our (pre-decision) state would be

$$S_t = (R_t, p_t, B_t),$$

where B_t is our belief state as it was above.

State-independent functions represent pure learning problems, which is the focus of this book. State-dependent functions are inherently more difficult to solve than standard learning problems. For example, consider the easiest state-dependent problem, where we do not have a physical state R_t, and where p_t does not depend on history. In this setting, the price p_t represents the "context," and it means that we have to design a policy $X^\pi(B_t|p_t)$, whereas in the past we only had to design $X^\pi(B_t)$. Think of the contextual problem as having to design a learning policy $X^\pi(B_t|p_t)$ for each price p_t.

If we introduce a physical state R_t where R_{t+1} depends on x_t, then our policies need to capture the impact of a decision x_t on the future state. The types of learning policies we have seen up to now will not work well in this setting.

Problems with a physical state are quite complex, and generally beyond the scope of this book. We only revisit problems with a physical state at the end of the book.

Most of our attention focuses on problems with a pure belief state. There are three important classes of belief states:

Stationary distribution This is the primary focus of this volume, and describes problems with an unknown parameter characterized by a distribution, where we are trying to reduce the uncertainty by collecting information, typically from noisy experiments.

Nonstationary distribution In this setting, the unknown parameter is changing exogenously over time, which means that our distribution of belief about these parameters will exhibit increasing variance over time. This represents the problems known as restless bandits that were introduced in section 5.5.

Controllable distribution The most difficult problem class is where the uncertainty is controlled (or at least influenced) by decisions. For example, we may be trying to minimize the blood sugar of a patient through the choice of type of medication (which is a decision) that affects how the patient responds to different types of food (in an uncertain way). Alternatively, an athlete (or team) will improve with practice.

We have already seen problems with correlations in the belief model. This is where $Cov(\mu_x, \mu_{x'}) \neq 0$. We have only begun to explore the depth and richness of covariance structures for correlated beliefs. Correlated beliefs allow us to address problems with large (or infinite) numbers of potential alternatives with relatively small experimental budgets. Some examples of correlated beliefs include:

- Evaluations are made of a continuous (and possibly scalar) function. We might be sampling disease within a population, the response due to a particular drug dosage, or the demand response to the price of a product. The results of experiments are correlated inversely proportional to the distance between two experiments.
- Evaluations of multiattribute entities. We might be predicting the importance of a document (based on the attributes of the document) or the likelihood that someone will default on a loan (as a function of an individual's financial characteristics).
- Estimating the time on a path in a network. The observed travel time over one path will be correlated with other paths that share common links.
- Effectiveness of a drug compound. Different drug compounds will share common atomic subsequences which interact and determine the effectiveness of a drug. Drug compounds sharing common atoms (typically in specific locations) may exhibit correlations in their effectiveness.

The distinction between a physical (or resource) state and a belief (or knowledge) state has caused confusion in the dynamic programming community since the 1950s. It is common for people to equate "state" and "physical state," which is problematic when we only have a belief state. When it became clear that our belief about parameters could also evolve, Bellman & Kalaba (1959) used the term "hyperstate" to refer to the combination of physical state (an imperfect term) and belief state. In Chapter 15 we address the problem of learning in the presence of a physical state. Rather than use terms like hyperstate, we prefer to use terms like physical (or resource) state, information state and belief state.

6.2 TYPES OF DECISIONS

The complexity of a problem depends in large part on the nature of the decision we have to make in order to make an experiment. Major problem classes include:

- Binary decisions - We can continue to collect information, or stop; we can decide to show a document to an expert, or not; or we can switch to a new web design, or stay with our existing one (known as A/B testing).
- Discrete choice - Here, we have a set of discrete choices (not too large - dozens, hundreds, perhaps thousands), where at each point in time we have to make a decision to collect information about one of the choices. A discrete choice could be a person to do a job, a technology, a drug compound or a path through a network. So far, we have considered only discrete choice problems under the heading of ranking and selection or bandit problems.
- A scalar, continuous variable - We have to choose a quantity, price, location of a facility or concentration of a chemical that we need to optimize.
- A vector, where the elements can come in several flavors:
    a) Binary - 0's and 1's, which we would use to choose subsets or portfolios of discrete items.

b) Reals - We might have to set a multidimensional set of continuous parameters, such as testing a series of settings on a physical device such as an engine, or parameters governing the performance of a simulator.

c) Integers - Similarly, we may have a vector of discrete variables (but other than 0 or 1), such as testing an allocation of ambulances or locomotives, or a mix of different types of aircraft.

d) Multiattribute - We might choose a complex item (a document, a molecule, a person) characterized by a vector of categorical attributes.

We let x represent a generic "decision." We might have $x \in (0,1)$, or $x = (1,2,\ldots,M)$, or $x = (x_d)_{d \in \mathcal{D}}$ where $d \in \mathcal{D}$ is a type of decision ("fly to Chicago," "try a particular drug compound") where x_d can be binary, discrete or continuous. We let $\mathcal{X}$ be the set of feasible decisions. It is very important from a computational perspective to understand the nature of x, since there are problems where we assume that we can easily enumerate the elements of $\mathcal{X}$. In the list above, we open the door to problem classes where the size of $\mathcal{X}$ is infinite (if x is continuous) or much too large to enumerate.

A different dimension arises when we separate experimental decisions from implementation decisions. Some examples include:

- We may fund research (the experiment) to build a device (the implementation).
- We may observe the performance of links in a network (the experiment) to choose a path (implementation).
- We may wish to visit a company to talk to its management team (the experiment) before deciding to invest in the company (implementation).

For online learning problems, we learn as we go, so the only decisions are implementation decisions, but we can also learn from these. For offline learning problems, we run a series of experiments where x represent experimental design decisions. We have assumed up to now that after testing different $x \in \mathcal{X}$, that we choose one x as our final design, which means that $\mathcal{X}$ is also the set of implementation decisions. However, this is not always the case.

6.3 EXOGENOUS INFORMATION

Exogenous information clearly comes in a wide variety depending on the application. We briefly survey a few common types:

- Stationary, uncorrelated processes - This is the simplest problem, where observations come from a stationary distribution (with unknown mean, and potentially unknown variance), and where observations of different alternatives are uncorrelated.
- Nonstationary processes - We may make an observation of the level of disease in the population, but this can clearly change over time, but it changes in an exogenous way that we do not control.

- Correlated beliefs - We might wish to simulate multiple configurations using what are known as common random variables, which means that we use the same sample realizations to help control statistical error. It is important to separate correlated beliefs (where W_x^n is correlated with $W_{x'}^n$) and correlated beliefs (where $Cov(\mu_x, \mu_{x'}) > 0$).
- Learning processes - We might have a nonstationary process, but one that improves (or potentially deteriorates) as we sample it, rather than through some exogenous processes. For example, imagine observing how well a baseball player can hit; as we observe the player, his hitting may actually improve.
- State and/or action dependent processes - We generally view exogenous information as dependent on an unknown truth, but exogenous information may be a function of the state (this happens with learning processes described above) and possibly even the action (if we decide to sell stock or expensive equipment, this can depress the market price for these assets). When this happens, we will replace our conditional expectation $\mathbb{E}_{W^1,\ldots,W^N|\mu}\ldots$ with $\mathbb{E}^{\pi}_{W^1,\ldots,W^N|\mu}\ldots$, since the observations W^n depend on the policy $X^\pi(S^n)$.

All of these generalizations represent interesting extensions of our learning model.

6.4 TRANSITION FUNCTIONS

If there is a physical state R^n, we are going to assume we are given a function that describes how it evolves over time. We write it as

$$R^{n+1} = R^M(R^n, x^n, W^{n+1}).$$

The notation $R^M(\cdot)$ represents the "resource transition function." In traditional dynamic models (where the belief state is held constant), this would be the transition function, system model or simply "the model." For our purposes, we do not need to get into the details of this function, which can vary widely.

We assume that with each decision x^n, we learn something that allows us to update our belief state. We represent this generically using

$$B^{n+1} = B^M(B^n, x^n, W^{n+1}).$$

For example, $B^M(\cdot)$ could represent the Bayesian updating equations (3.2) and (3.3) in a ranking and selection problem. When we want to be really compact, we write the state variable as $S^n = (R^n, B^n)$ and represent its transition function using

$$S^{n+1} = S^M(S^n, x^n, W^{n+1}).$$

Elsewhere in this volume, we have been using the function $S^M(\cdot)$ to represent the updating of the belief state when there is no physical state because this notation is more familiar and there is no ambiguity. We suggest using the notation $B^M(\cdot)$ in situations where there is both a physical and a belief state, and when we want to refer specifically to the updating of the belief state.

Most of this volume focuses on problems without a physical state. Chapter 15 addresses learning in the presence of a physical state.

6.5 OBJECTIVE FUNCTIONS

There are several dimensions to the design of an objective function:

1) Are we focusing on the best design (terminal reward) or maximizing performance along the way (cumulative reward)? Terminal reward problems tend to arise in a laboratory or test setting (sometimes called offline learning) while cumulative reward problems arise when we have to learn in the field (also known as online learning).

2) Different types of experimental costs.

3) Performance metrics and policy evaluation.

6.5.1 Terminal vs. cumulative reward

Chapters 3 and 4 started with the types of learning problems that arise in a laboratory, computer simulator or a test environment, where we care only about the quality of the final design rather than how well we do while learning this final design. Since this objective typically arises in a testing environment, it is often referred to as offline learning.

Chapter 5 then shifted to learning in the field, a setting that is sometimes referred to as online learning. Here, we have to pay attention to how well we are doing while we are learning, which means we want to maximize the cumulative reward.

Below, we briefly sketch how offline (terminal reward) and online (cumulative reward) objectives might be structured.

Offline learning (terminal reward)

Up to now, we have represented our truth about design x using an unknown μ_x. We can use our learning policy $X^\pi(S)$ to generate decisions about what experiment to perform next, giving us the sequence

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

where $x^n = X^\pi(S^n)$. From the sequence $W^1_{x^0}, W^2_{x^1}, \ldots, W^N_{x^{N-1}}$, we obtain estimates $\bar{\mu}^{\pi,N}_x$, from which we find our best design

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

We can then evaluate the performance of our policy using

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

In a simulator where we simulate μ_x from the prior $N(\mu^0_x, \beta^0_x)$, we can use our simulated truth to evaluate the policy with

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

We are going to later need to represent our unknown function as $F(x,W)$ where W is a random input, from which we can create a noisy observation $\hat{F} = F(x,W)$. We typically do not know the function $F(x,W)$, or we may not know the distribution of W, but either way, we do not know $\mathbb{E}F(x,W)$. If x is discrete, we can still let $\mu_x = \mathbb{E}F(x,W)$. More generally, we can let $\overline{F}^n(x)$ be an approximation of $\mathbb{E}F(x,W)$ after n experiments (where, for example, we have observed $\hat{F}^1,\ldots,\hat{F}^n$).

While there are different ways of approximating $\overline{F}^n(x)$, one that we use later is to assume a parametric form that we represent using $f(x|\theta)$. For example, we might use a linear model

$$f(x|\theta) = \theta_0 + \theta_1\phi_1(x) + \theta_2\phi_2(x),$$

where $(\phi_1(x),\phi_2(x))$ are features derived from x. For example, we may be choosing the best movie, where x specifies the movie, while $\phi_f(x)$ extracts information such as year made, genre, leading actors, and so on.

Using our parametric model, we are going to assume that we can view W as either an input to $F(x,W)$, or as a noisy observation of the parametric function itself as in

$$W = f(x|\theta) + \varepsilon. \tag{6.1}$$

In other words, we can treat a realization of W as a realization of the function.

With a parametric belief model, we assume we know the parametric form, but do not know the parameter vector θ. If we use a linear belief model (as we do in chapter 7), we show how to create a multivariate normal distribution of θ with point estimate $\bar{\theta}^n$, and covariance matrix $\Sigma^{\theta,n}$, where

$$\Sigma^{\theta,n}_{ij} = Cov(\theta_i,\theta_j).$$

In chapter 8 we are going to show how to do optimal learning use a *sampled belief model*, which we first introduced in section 2.4, where the true θ is represented

on nonlinear belief models, we are going to introduce a different way of representing the distribution of belief about the true θ called the sampled belief model.

Using our parametric model, if we use the sampled belief model we introduced in section 2.4 we would obtain a set of probabilities $p^N_k = \mathbb{P}[\theta = \theta_k]$. We could use these probabilities to find an estimate of θ using

$$\bar{\theta}^{\pi,N} = \sum_{k=1}^{K} p^N_k \theta_k.$$

We can then use this estimate to find the best experimental design using

$$x^{\pi,N} = \arg\max_{x\in\mathcal{X}} f(x|\bar{\theta}^{\pi,N}).$$

A better way (if we can compute it) is to compute the best design using

$$x^{\pi,N} = \arg\max_{x\in\mathcal{X}} \sum_{k=1}^{K} f(x|\bar{\theta}_k)p^N_k.$$

We then evaluate our learning policy π by evaluating the performance of the solution using

$$F^{\pi} = \mathbb{E}_{\theta}\mathbb{E}_{W^1,\ldots,W^N|\theta}\mathbb{E}_{\widehat{W}}F(x^{\pi,N},\widehat{W}). \tag{6.2}$$

We can break down the expectation in equation (6.2) as follows. The first expectation $\mathbb{E}_{\theta}$, which would only be used with a Bayesian belief model, is where we take the expectation over different possible values of θ based on some prior (note that W depends on θ as given in equation (6.1)). Then, given θ, the expectation $\mathbb{E}_{W^1,\ldots,W^N|\theta}$ represents the random experiments that we run while learning.

We then have to evaluate how well our design works, and we use $\widehat{W}$ to represent the uncertainty during the testing (while the sequence $W^1,\ldots,W^N$ is the observations used for training). For this reason, we use the final $\mathbb{E}_{\widehat{W}}$ to evaluate our solution $F(x^{\pi,N},\widehat{W})$ by averaging over the random realizations $\widehat{W}$ if we were to implement our final design $x^{\pi,N}$. Of course, we cannot compute these expectations in practice, so we have to turn to the simulation-based estimates that were described in section 3.4.

Online learning (cumulative reward)

In online learning, we are learning as we progress, so we are trying to maximize the cumulative rewards, as we did in section 5.6. Using the same notation as above for offline learning, the value of a policy would be given by

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

Now make the transition to using a function $F(x,W)$ to represent our performance that we wish to maximize.

$$F^{\pi} = \mathbb{E}_{\theta}\mathbb{E}_{W^1,\ldots,W^N|\theta}\sum_{n=0}^{N-1}F(X^{\pi}(S^n),W^{n+1}). \tag{6.3}$$

If we refer back to the offline counterpart of equation (6.3), we see that (6.2) has a final $\mathbb{E}_W$ for evaluating the final design $x^{\pi,N}$. In the online version in (6.3), we evaluate as we go.

If we compare our offline, terminal reward objective in (6.2) to our online, cumulative reward objective in (6.3), we see two differences. The most obvious, of course, is that in when evaluating the cumulative reward in (6.2) we have to sum the rewards as we progress rather than just the performance at the end. Second, when using the terminal reward in (6.2) we have to introduce a testing step, which is where we use the random variable $\widehat{W}$. We do not need this in online learning, since the observations $W^1,\ldots,W^N$ used for learning are the same that are used for evaluating experimental choices as we progress.

6.5.2 Experimental costs

There are several variations that distinguish problems in terms of experimental costs. These include

- Startup costs - There may be a certain amount of time or money required to start measuring an alternative, making the first observation much more expensive than subsequent observations. For example, perhaps we have to invite a baseball player to training camp, or we have to purchase a device to test it, or we have to create a molecule in a lab to determine its properties.
- Switchover costs - Startup costs address the first experiment of an alternative, but there may be switchover costs, where even after the first experiment, there may be a cost from making an observation of x to an observation of x'. For example, perhaps we are simulating a policy, but it takes time to switch from testing one set of parameters to another because we have to reload a software package or restart a model that has been stopped.
- Alternative specific costs - We often represent the budget for running experiments as the number of experiments, which also implies that each alternative "costs" the same. But this may not be true at all. For example, an experiment might involve drilling an exploration well (costs depend on geology) or testing a population for disease (cost might reflect distance to travel, or the type of diagnostic test).

6.5.3 Objectives

Below we provide an indication of some objectives we might use. Perhaps not surprisingly, there is a variety of possible objectives that we might use. This discussion simply hints at the diversity of perspectives.

Expected value

Up to now, we have been choosing a policy to maximize our performance. We can write this as

$$\begin{aligned} V^{\pi} &= \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\mu_{x^{\pi}} \qquad (6.4)\\ &= \mathbb{E}_{\mu}\mathbb{E}_{W^1,\ldots,W^N|\mu}\left\{\max_{x\in\mathcal{X}}\bar{\mu}_x^{\pi,N}\right\}, \qquad (6.5)\end{aligned}$$

where $x^{\pi} = \arg\max \bar{\mu}_x^{\pi,N}$. We remind the reader that the expectation is over all beliefs μ and all experimental outcomes W. We note that in (6.4), we are using our belief μ_x to evaluate the result of our policy x^{π}. In (6.5), we are using our estimates $\bar{\mu}_x^{\pi,N}$. Both objectives produce the same policy, and in expectation they are the same, as we showed in Section 3.6. In both versions of the objective function, we are taking an expectation over truths, so we are not changing our fundamental model. However, in practice it will be statistically more reliable to use (6.4).

It helps to illustrate the approximation of the expectations in equation (6.5). Assume we have L possible values for the truth μ with possible values $\mu(\ell)$, $\ell = 1,\ldots,L$, where $\mu_x(\ell)$ is the value of alternative x when $\mu = \mu(\ell)$. Next assume that we have

K possible experimental paths, where we represent the kth possible experimental sequence as $W^1(k), \ldots, W^N(k)$, where $k = 1, \ldots, K$.

Next let $\bar{\mu}_x^{\pi,N}(k,\ell)$ be the estimate of μ_x when the truth $\mu = \mu(\ell)$ and we are following sample path $W^1(k), \ldots, W^N(k)$ while using experimental policy π. Averaged over all truths and experimental samples gives us estimates

$$\bar{\mu}_x^{\pi,N} = \frac{1}{L}\sum_{\ell=1}^{L}\frac{1}{K}\sum_{k=1}^{K}\bar{\mu}_x^{\pi,N}(k,\ell). \tag{6.6}$$

Our best design is then given by

$$x^{\pi,N} = \arg\max_x \bar{\mu}_x^{\pi,N}. \tag{6.7}$$

We can then evaluate the value of our policy using

$$\overline{V}^{\pi} = \frac{1}{L}\sum_{\ell=1}^{L}\mu_{x^{\pi,N}}(\ell). \tag{6.8}$$

Note that we are using estimated values $\bar{\mu}_x^{\pi,N}$ when we find the best design in equation (6.7), whereas we use the true value $\mu_x(\ell)$ when evaluate the value of the policy in equation (6.8).

Expected opportunity cost (or regret)

It is very common to minimize the *opportunity cost* (also known as *regret*) which measures how much worse we are doing than the optimal (whether we are minimizing or maximizing). This is written as

$$V^{OC,\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\left\{\max_x \mu_x - \mu_{x^\pi}\right\}. \tag{6.9}$$

This objective function is also known as "linear loss," because it represents the suboptimality of the alternative selected by policy π. The best possible loss value is zero, indicating that we have found the true best alternative.

In online problems, it sometimes makes sense to consider the average opportunity cost or regret per time step. Thus, if we have N experiments total, we can compute

$$\overline{V}^{OC,\pi} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\frac{1}{N}\sum_{n=0}^{N-1}\left\{\max_x \mu_x - \mu_{x^n}\right\}, \tag{6.10}$$

where $x^n = X^\pi(S^n)$.

Risk

A largely overlooked aspect of experimentation is risk. We might be running a series of experiments in the hope that we will achieve a certain target, such as killing enough cancer cells, finding the price that produces revenue over a particular target, or designing a material whose strength is over a required level. For such situations, we need to replace the expectation with a risk measure that we designate $\rho(Z)$ which operates on a random variable Z (the objective function) in a way that penalizes unacceptable outcomes. For example, imagine we have a threshold z_α. Our risk measure might be

$$\rho(Z) = \mathbb{E}Z + \eta\mathbb{E}\max\{0, Z - z_\alpha\}. \tag{6.11}$$

This version of $\rho(Z)$ is striking a balance between maximizing the expectation $\mathbb{E}Z$ and maximizing the expectation of Z when it exceeds a threshold z_α. Here, the parameter η puts the weight on outcomes $Z > z_\alpha$, with an implicit penalty on outcomes $Z < z_\alpha$. Evaluating policies using the risk measure $\rho(Z)$ using simulation is the same as with an expectation, with the single change of including the additional penalty term. This said, very little attention has been placed on handling risk in learning problems.

Robust optimization

In some settings, it makes more sense to focus on the worst case rather than an expectation. This is becoming known in the optimization community as *robust optimization*. We illustrate the calculation of a robust objective using the sampled approximation we introduced when computing the expected value of a policy above. There, we let $\mu_x(\ell)$ for $\ell = 1, \ldots, L$ be a set of truths for μ_x, and $W^n(k)$ be the nth realization of W when following sample path k, $k = 1, \ldots, K$. Let $\bar{\mu}^{\pi,N}$ calculated using equation (6.6), and let $x^{\pi,N}$ be the best design computed using equation (6.7).

Above, we computed the expected performance of a policy using

$$\overline{V}^\pi = \frac{1}{L}\sum_{\ell=1}^{L} \mu_{x^{\pi,N}}(\ell).$$

We might be interested in knowing how badly our policy might perform when the truth is some extreme value. Instead of taking an average, we might instead use the truth that produces the worst performance, which we can compute using

$$\overline{V}^{robust,\pi} = \min_\ell \mu_{x^{\pi,N}}(\ell). \tag{6.12}$$

where $x^{\pi,N}$ is computed using equation (6.7).

Equation (6.12) recognizes that some truths may be harder to discover than others. We may also want to consider that some experimental sequences may also produce worse designs. Let

$\bar{\mu}^{\pi,N}_x(k,\ell)$ = Estimate of μ_x when $\mu = \mu(\ell)$ and we observe experimental sequence $W^1(k), \ldots, W^N(k)$.

Now let the resulting best design be computed using

$$x^{\pi,N}(k) = \arg\max_x \bar{\mu}^{\pi,N}_x(k,\ell). \tag{6.13}$$

Finally, we can compute the worst possible performance over truths and sample paths using

$$\overline{V}^{robust2,\pi} \quad = \quad \min_{\ell} \min_{k} \mu_{x^{\pi,N}(k)}(\ell). \tag{6.14}$$

We can now use $\overline{V}^{robust,\pi}$ or $\overline{V}^{robust2,\pi}$ as the value of the policy when comparing the performance of policies. While it is nice to find policies that work best on average, there may be considerable interest in policies that mitigate the worst possible outcomes.

Yet a further generalization is to use the concept of *quantile optimization*. If Q_α is the α quantile of a random variable, we may wish to solve

$$V^\pi_\alpha \quad = \quad Q_\alpha \mu_{x^\pi}. \tag{6.15}$$

Quantile optimization is related to robust optimization, because we might choose to maximize the 10^{th} quantile. Quantile optimization is also useful in the presence of heavy-tailed random variables.

Probability of correct selection

A different perspective is to focus on the probability that we have selected the best out of a set $\mathcal{X}$ alternatives. In this setting, it is typically the case that the number of alternatives is not too large, say 10 or 20, and certainly not 100,000. Assume that

$$x^* = \arg\max_{x \in \mathcal{X}} \mu_x$$

is the best decision (for simplicity, we are going to ignore the presence of ties). As before our we choose as the best design

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

We have made the correct selection if $x^{\pi,N} = x^*$, but even the best policy cannot guarantee that we will make the best selection every time. Let $1_{\{\mathcal{E}\}} = 1$ if the event $\mathcal{E}$ is true, 0 otherwise. We write the probability of correct selection as

$$\begin{aligned} P^{CS,\pi} &= \text{probability we choose the best alternative} \\ &= \mathbb{E}^\pi 1_{\{x^{\pi,N} = x^*\}}, \end{aligned}$$

where the underlying probability distribution depends on our learning policy π. The probability is computed using the appropriate distribution, depending on whether we are using Bayesian or frequentist perspectives. This may be written in the language of loss functions. We would define the loss function as

$$L^{CS,\pi} \quad = \quad 1_{\{x^{\pi,N} \neq x^*\}}.$$

Although we use $L^{CS,\pi}$ to be consistent with our other notation, this is more commonly represented as L_{0-1} for "0-1 loss."

Note that we write this in terms of the negative outcome so that we wish to minimize the loss, which means that we have not found the best selection. In this case, we would write the probability of correct selection as

$$P^{CS,\pi} = 1 - \mathbb{E}^\pi L^{CS,\pi}.$$

Indifference zone selection

A variant of the goal of choosing the best is to maximize the likelihood that we make a choice that is almost as good as the best. Assume we are equally happy with any outcome within δ of the best. This is referred to as the *indifference zone*. Let $V^{n,\pi}$ be the value of our solution after n experiments. We require $\mathbb{P}^{\pi}\{\mu_{d^*} = \bar{\mu}^*|\mu\} > 1 - \alpha$ for all μ where $\mu_{[1]} - \mu_{[2]} > \delta$, and where $\mu_{[1]}$ and $\mu_{[2]}$ represent, respectively, the best and second best choices.

We might like to maximize the likelihood that we fall within the indifference zone, which we can express using

$$P^{IZ,\pi} = \mathbb{P}^{\pi}(V^{n,\pi} > \bar{\mu}^* - \delta).$$

As before, the probability has to be computed with the appropriate Bayesian or frequentist distribution.

Least squared error

A different form of loss function arises when we want to fit a function to a set of data. In this setting, we think of "x" as a set of independent variables which we choose directly or indirectly. For example, we may be able to choose x directly when fitting a linear regression of the form

$$\begin{aligned} Y(x) &= \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_I x_I + \epsilon \\ &= \theta x + \epsilon. \end{aligned}$$

where Y is the observed response and ϵ is the random error explaining differences between the linear model and the responses. We choose it indirectly when our regression is in the form of basis functions, as in

$$Y(x) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(x) + \epsilon.$$

Classical linear regression assumes that we are given a set of observations which we use to fit a model by choosing θ. Let

$$Y^{n+1} = \theta x^n + \epsilon^{n+1}.$$

where θ is the true set of parameters. Our indexing reflects our requirement that x^n be chosen before we observe ϵ^{n+1}. Our measure of performance is given by

$$F(Y^{(N+1)}, x^{(N)}|\bar{\theta}) = \sum_{n=1}^{N} (Y^{n+1} - \bar{\theta} x^n)^2.$$

which is the sample sum of squares given experiments $x^{(N)} = (x^0, \ldots, x^N)$ and observations $Y^{(N+1)} = (Y^1, \ldots, Y^{N+1})$. Ordinary least squares regression fits a model by finding

$$\bar{\theta}^{N+1} = \arg\min_{\bar{\theta}} F(Y^{(N+1)}, x^{(N)}|\bar{\theta}).$$

Let $F^*(Y^{(N+1)}, x^{(N)}) = F(Y^{(N+1)}, x^{(N)}|\bar{\theta}^{N+1})$ be the optimal solution given $Y^{(N+1)}$ and $x^{(N)}$. Sequential estimation starts with $\bar{\theta}^n$, then measures x^n, and finally observes Y^{n+1} from which we compute $\bar{\theta}^{n+1}$. This can be done easily using recursive least mean squares, given by

$$\bar{\theta}^{n+1} = \bar{\theta}^n - H^n x^n (\bar{\theta}^n x^n - Y^{n+1})$$

where H^n is an $I \times I$ scaling matrix that is computed recursively (we cover this in Section 7.2.2).

Our focus is on choosing the experiment x^n. Classical experimental design assumes that we choose $(x^n)_{n=0}^N$ first and then fit the model. This is sometimes referred to as batch design because the entire sample is chosen first. This is equivalent to solving

$$\min_{x^{(N)}} \mathbb{E}F^*(Y^{(N+1)}, x^{(N)})$$

where the expectation is over the random variables in $Y^{(N+1)}$.

We focus on sequential design, where we choose x^n given our state S^n, which includes $\bar{\theta}^n$ and the information we need to update $\bar{\theta}^n$. In a sequential learning problem, we have to use some basis for determining how well we have done. In our optimization problems, we want to maximize our expected contribution. This optimization problem determines the values of x that are most interesting. In the area of adaptive estimation, we have to specify the values of x that are most likely to be interesting to us, which we designate by a density $h(x)$ which has to be specified. In an off-line learning environment, we want to choose $x^1, \ldots, x^n, \ldots, x^N$ according to a policy π to solve

$$\min_{\pi} \mathbb{E} \int_x \left(Y(x) - \bar{\theta}^{\pi} x\right)^2 h(x) dx$$

where $Y(x)$ is the random variable we observe given x, and where $\bar{\theta}^{\pi}$ is the value of $\bar{\theta}$ produced when we select x^n according to π, and when we estimate $\bar{\theta}$ optimally.

This formulation requires that we specify the domain that interests us most through the density $h(x)$. An illustration of the density function arises when we are trying to sample nuclear material over a border or in a region. For such cases, $h(x)$ might be the uniform density over the region in question. When we solve on-line and off-line optimization problems, we do not have to specify $h(x)$ explicitly. The optimization problem, e.g. equation (5.3), determines the region within $\mathcal{X}$ that is of greatest interest.

Entropy minimization

Entropy is a measure of uncertainty that can be used for numeric and nonnumeric data. Imagine that we are trying to estimate a parameter μ that we know with uncertainty. If our distribution of belief about μ is continuous with density $f(u)$, a measure of the uncertainty with which we know μ is given by the entropy of $f(u)$, given by

$$H(\mu) = -\int_u f_u \log(f_u) du.$$

The logarithm is typically taken with base 2, but for our purposes, the natural log is fine. The entropy is largest when the density is closest to the uniform distribution. If the entropy is zero, then we know μ perfectly. Thus, we can try to take experiments that reduce the entropy of the distribution that describes our knowledge about a parameter.

6.6 DESIGNING POLICIES

Up to now, we have introduced a number of different policies to illustrate different methods for learning. Our list at this stage includes

- Decision trees (section 1.6) - Decision trees allow us to optimize decisions and information over some horizon.
- Dynamic programming (section 3.2.2) - We showed that we could set up a learning problem as a dynamic program using Bellman's optimality equation to characterize an optimal policy (which is typically impossible to solve).
- Pure exploration (section 3.2.3) - Here we just choose actions at random.
- Excitation policies (section 3.2.3) - For problems with continuous controls, we add a noise term as in

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

  The noise ε serves as an *excitation* term to encourage exploration.
- Epsilon-greedy (section 3.2.3) - Randomly explore (choose an action at random) or exploit (choose the action that appears to be best).
- Boltzmann exploration (section 3.2.3) - Choose an action x with a probability proportional to the exponential of an estimated value of the action.
- Interval estimation (section 3.2.3) - Choose the action with the highest index which is given by the α-percentile of the distribution of belief about the value of the actcion.
- Upper confidence bounding (sections 3.2.3 and 5.2) - Choose the action with the highest index given by expected reward plus a term that captures how often an action has been tested (there are a number of variations of this).
- Gittins indices (section 5.3) - Choose an action based on the Gittins index, which scales the standard deviation of the measurement noise. This produces an optimal policy for certain problems.
- Value of information policies (chapter 4) - This includes the knowledge gradient and expected improvement.

At this point we are not going to make a distinction between policies for offline learning (where we maximize the final reward) or online learning (where we maximize the cumulative reward). Rather, we are going to take the position that we have a family of policies to consider which have to be evaluated using whatever objective function we choose.

It turns out that this rather daunting list of policies can be organized into two core strategies, each of which can then be further divided into two classes, creating four (meta) classes of policies.

The two core strategies are:

Policy search These are policies that are characterized by a set of parameters θ that have to be tuned either using a simulator (offline) or in the field (online).

Lookahead policies This group tries to build good policies by approximating the impact of a decision now on the future.

6.6.1 Policy search

Policy search involves searching over a parameterized set of policies to find the one that works the best, typically in a simulated setting but possibly in an online, real-world setting. These come in two flavors:

Policy function approximations (PFAs) These are analytical functions that map states directly to actions. Some examples are

- We have a policy to apply a force x based on the speed of a robot and the slope of the surface it is moving over. If S_{t1} is the speed and S_{t2} is the slope, we might choose are speed using the policy

$$X^{\pi}(S_t|\theta) = \theta_0 + \theta_1 S_{t1} + \theta_2 S_{t1}^2 + \theta_3 S_{t2} + \theta_4 S_{t2}^2.$$

- We wish to use a battery to draw energy from the grid when prices are low, and then sell back to the grid when they are high. We can write the policy as

$$X^{\pi}(S_t|\theta) = \begin{cases} \text{-1} & \text{If } p_t \leq \theta^{min} \\ 0 & \text{If } \theta^{min} < p_t < \theta^{max} \\ 1 & \text{If } p_t \geq \theta^{max} \end{cases}$$

where $\theta = (\theta^{min}, \theta^{max})$ are tunable parameters.

Parametric cost function approximations (CFAs) In this class, we are maximizing or minimizing a parametric function, which is typically a parametrically modified cost function. The simplest example of this is interval estimation, where the policy is

$$X^{IE}(S^n|\theta) = \arg\max_x \big(C(S^n, x) + \theta\bar{\sigma}_x^n\big).$$

The distinguishing feature of a CFA policy is the presence of an $\arg\max_x$ or $\arg\min_x$ of a cost that does not model decisions and information in the future.

6.6.2 Lookahead policies

Lookahead policies are based on finding the best decision that maximizes the expected rewards over some horizon. We can divide these into two broad groups:

Value function approximations (VFAs) - In this class we design functions that approximate the value of being in a state resulting from an action made now.

Direct lookaheads (DLAs) - This approach explicitly optimizes decisions now and into the future over some horizon, to help make a better decision now. For learning problems, it is useful to further divide this class into two subclasses:

- Single-period lookahead - We make a decision at iteration n that considers the decision we will make at iteration $n + 1$.
- Multi-period lookahead - We optimize decisions for H time periods into the future.

We first saw a policy using a value function in section 3.2.2 where we showed we could write

$$V^n(S^n) = \max_x \big(C(S, x) + \mathbb{E}_W V^{n+1}(S^{n+1})\big). \tag{6.16}$$

where $S^{n+1} = S^M(S^n, x, W^{n+1})$, and where S^n is our belief state after n experiments. There are settings where our belief state is discrete which allows us to actually compute $V^n(S^n)$, but this is rare. Typically S^n is multidimensional and, typically, continuous, making it impossible to compute $V^n(S^n)$ exactly. An alternative approach is to try to approximate the value function, but as of this writing, this has not emerged as a useful approach for pure learning problems.

By contrast, policies based on one-step lookahead models (and in some cases multi-step) have actually proven to be quite popular.

6.6.3 Classifying policies

Every single policy can be classified as one of our four classes: PFAs, CFAs, VFAs, and DLAs. Below we list all the policies we have seen so far using this grouping:

PFAs Policy function approximations include

- Pure exploration, where we choose an action at random (unrelated to its value).
- Lookup tables which specify "when in discrete state s take discrete action x." This includes any rule-based system.
- Linear models, often called "affine policies" with the form:

$$X^\pi(S^n|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(S^n).$$

- Excitation policies, where we add a noise term to a continuous controller (such as the affine policy above)

- Neural networks. These are popular in the control of engineering systems, but have not been used for learning problems.

CFAs Parametric cost function approximations include

- Pure exploitation, where we choose

$$X^\pi(S^n) = \arg\max_x \bar{\mu}^n_x.$$

- Epsilon-greedy, where we choose an action at random with probability ϵ and use a pure exploitation policy with probability $1 - \epsilon$.
- Bayes greedy. Imagine we have a nonlinear function $f(x|\theta)$ where θ is an unknown parameter that might be multivariate normal $\mathcal{N}(\theta^n, \Sigma^n)$. A basic pure exploitation policy would be

$$X^\pi(S^n) = \arg\max_x f(x|\mathbb{E}^n\theta) = \arg\max_x f(x|\theta^n).$$

  This is a basic greedy (or myopic) policy, but what we really want to do is to maximize the expectation of the function (rather than the function evaluated at the expectation). This would be written

$$X^\pi(S^n) = \arg\max_x \mathbb{E}\{f(x|\theta)|S^n\}$$

  where S^n is the state that the true $\theta \sim \mathcal{N}(\theta^n, \Sigma^n)$. Using the function of the expectation is fine if $f(x|\theta)$ is linear in θ, but not if it is nonlinear. If we maximize the expectation of the function, then this is called a Bayes greedy policy.
- Interval estimation:

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

- Upper confidence bounding, for example

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

- Boltzmann exploration, where we choose an action x with probability

$$p^n_x(\theta) = \frac{e^{\theta\bar{\mu}^n_x}}{\sum_{x'\in\mathcal{X}} e^{\theta\bar{\mu}^n_{x'}}}.$$

  The Boltzmann policy is often referred to as a "soft-max" policy does not have an explicit $\arg\max_x$, but is performing this operation probabilistically.

VFAs This would include our optimal policy (equation (6.16)), where the policy depends on a value function, as in

$$X^{VFA}(S^n) = \arg\max_x \left(\bar{\mu}^n_x + \mathbb{E}_W V^{n+1}(S^{n+1})\right).$$

This is how Gittins indices were computed in section 5.3.

DFAs Direct lookaheads - These come in single-step and multi-step variations:

- Single step - These are very popular in optimal learning applications, and include:
  - Value of information policies such as the knowledge gradient or expected improvement.

    $$X_x^{KG,n} \quad = \quad \arg\max_x \left(\mathbb{E}\left\{ \max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^{n+1}(x) | S^n \right\} - V^n(S^n) \right).$$

    The maximization over x' is actually an optimization over possible decisions x^{n+1}, after the outcome of the experiment $x^n = x$ has been completed.
  - Thompson sampling:

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

    Thompson sampling can be thought of a simulation of what might happen in the $n + 1st$ experiment based on the time n distribution of belief.
- Multistep lookaheads:
  - Decision trees, which we first introduced in section 1.6.
  - The KG(*) policy, where we make a decision based on repeating an experiment n_x times (this is a restricted lookahead).
  - Extensions of KG(*) based on tuning the lookahead. We introduced this in seciton 4.3.3, where the policy is given by

    $$X^{KG}(\theta^{KG}) = \arg\max_x \nu_x^{KG,n}(\theta^{KG})$$

    Here, θ^{KG} is the parameter governing how many times we plan on repeating each experiment.

As of this writing, CFAs (such as upper confidence bounding) and one-step DLAs are by far the most popular. Randomized exploitation policies (a form of PFA) are popular with online applications, where it is tempting to do what appears to be best, but with some recognition that we need to introduce exploration.

Almost all of these policies can be tuned, or converted to a tunable version. Value of information policies are generally used without a tunable parameter by exploiting a Bayesian prior.

6.7 THE ROLE OF PRIORS IN POLICY EVALUATION

The process of evaluating policies is subtle, because it can bias the identification of good policies toward problems with specific properties that may not be clearly identified. Perhaps one of the most difficult issues arises in the interaction between how problems are generated (that is, the "truth") and the relationship between the truth and any prior information that may be used.

Truth from prior

From the Bayesian point of view, arguably the most natural way to evaluate a learning policy is to use the method described in Chapter 3, where we:

1) Generate a truth $\mu(\psi)$ from a prior.
2) Sample observations $W(\omega)$ from the truth $\mu(\psi)$.
3) Evaluate how well we discover the truth using one of the objectives listed above.
4) Repeat many times for different truths and different sample observations, to evaluate how well we discover the truth on average, and how reliably.

This strategy represents a very controlled experiment, which eliminates potential biases which a particular policy might be able to exploit. However, this approach may bias the evaluation process toward policies that work well with good priors.

Truth from an alternative prior

A way of evaluating the robustness of a policy is to generate truths from distributions other than the prior. For example, our prior may be normally distributed, but we might generate truths from a uniform or exponential distribution. Such experiments should focus on the ability of a policy to work with truths that come from distributions with potentially heavy tails, but the expected truth should match the expectation of the priors, which is to say that the prior should be unbiased.

Truth from a fixed prior

In the global optimization literature, where we collect noisy observations in an attempt to discover the maximum of an unknown continuous function, it is common to evaluate policies using a few "test functions." In other words, the truth is set to a fixed value that, for some reason, is believed to be a particularly interesting or difficult test case. For example, the Branin function

$$f\left(x_1, x_2\right) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos\left(x_1\right) + 10$$

is a commonly used test function in two dimensions. Test functions may be highly non-linear with multiple local maxima, and so we may view them as posing an especial challenge for an optimization algorithm. If we are able to successfully find the point (x_1^*, x_2^*) that maximizes this difficult function, we may reasonably believe that our policy will be able to learn other functions.

This is a frequentist approach to policy evaluation, where we consider a few fixed test cases rather than sampling truths from a distribution. However, the strength of this approach is that it presents a very clear and well-defined test suite of problems. In Chapter 14, we mention some of these test functions, such as the Branin function, the Ackley function, and others.

Truth from a biased prior

In any particular learning situation, the sample realization of a specific truth is going to be higher or lower than the mean of any prior distribution. However, a subtle issue arises if the expected truth is higher or lower than the expectation of the prior, an issue that we touched on in Section 4.9. In that section, we discussed the problem with the prior is consistently higher or lower than the truth. But this discussion did not recognize that it may be possible to identify and correct biases. Even more important to the discussion here is the fact that if there is a bias, then certain types of policies will tend to exploit this bias. For example, we might use epsilon-greedy or interval estimation, each of which has a tunable parameter. In the presence of a bias, we only have to tune the parameters to emphasize more exploration, and we are likely to get a better policy.

Expectations versus risks

Most of our presentation focuses on the expected performance of a learning policy. The probability of correct selection and indifference zone criterion can be viewed as risk-based measures, since they focus on finding alternatives that are within a specified tolerance with a specified probability.

For many information collection problems, risk is a major issue. Finding a workable drug that will help to extend lives with a high probability may be much more attractive than finding the best drug, but with a significant risk of not finding anything. The workable drug may not be ideal, but it may be good enough to gain federal approval which can then be marketed.

In real implementations, we are not allowed to run many sample realizations and take an average. We will use a policy on a single sample path. If we are not successful in finding, for example, a workable drug, we may be left asking whether this was just bad luck, or did we have a bad policy. We would like to minimize the possibility that it was bad luck.

Regret bounds

In Section 5.2, we described a policy known as upper confidence bounding. For example, for the case of normally distributed priors and rewards, the UCB policy was given by

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

where N_x^n is the number of times we have observed alternative x after n trials. UCB policies have attracted considerable attention in the community that works on multi-armed bandit problems after it was found that it was possible to bound the number of times that incorrect arms would be tested by $C \log N$, for an appropriately chosen constant C. In fact, it was shown that this was the best possible bound, which is often interpreted to mean that this may be the best possible policy.

In Section 5.4, comparisons were made between a tuned UCB policy and the knowledge gradient policy for problems with normally distributed rewards. Although the knowledge gradient policy outperformed UCB, it is hard to know if this reflects a subtle bias in how the experiments were run. As of this writing, there is insufficient empirical evidence to form any judgments regarding the value of regret bounds in terms of predicting the empirical performance of a UCB policy.

6.8 POLICY SEARCH AND THE MOLTE TESTING ENVIRONMENT

We first addressed the problem of evaluating policies in some depth in section 3.3 in the context of offline (terminal reward) settings. When we are using a simulator, we can use a policy π to find estimates $\bar{\mu}_x^{\pi,N}$ and then find the best choice using

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

We then evaluate our policy using the known truth (which we are simulating), which we write as

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

We often use the opportunity cost, or regret, where we compare how well we do against the best that could be done (which is possible in our simulated environment). This would be written as

$$R^{\pi,N} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu}\big(\max_x \mu - \mu_{x^{\pi,N}}\big).$$

Then, in chapter 5, we noted that we could evaluate policies using an online (cumulative reward) which we can write as

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

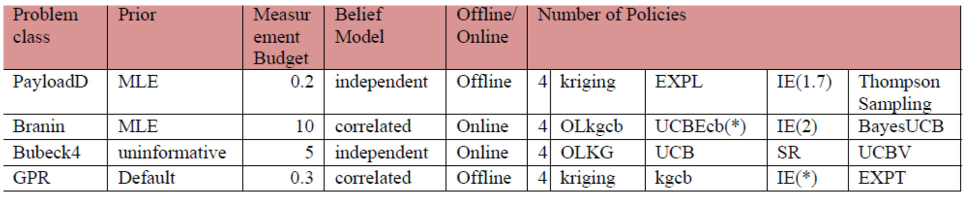
or in terms of its opportunity cost (or regret)

$$F^{\pi,N} = \mathbb{E}_\mu \mathbb{E}_{W^1,\ldots,W^N|\mu} \sum_{n=0}^{N-1} \big(\max_x \mu_x - \mu_{X^\pi(S^n)}\big).$$

Policy search typically involves searching across different classes of policies (UCB, interval estimation, Gittins indices, knowledge gradient), and then tuning parameters within a class (if there are tunable parameters).

We developed a public domain, Matlab-based system called MOLTE (Modular, Optimal Learning Testing Environment) for comparing a variety of policies on a library of test problems. Comparisons can be made on either an offline (terminal reward) or online (cumulative reward) basis. Each policy, and each problem, is encoded in its own .m file.

The front end of MOLTE is a spreadsheet, shown in figure 6.1, where you can specify which problems you want to use for testing, and which policies you want to apply



Problem class	Prior	Measurement Budget	Belief Model	Offline/ Online	Number of Policies				
PayloadD	MLE	0.2	independent	Offline	4	kriging	EXPL	IE(1.7)	Thompson Sampling
Branin	MLE	10	correlated	Online	4	OLkgcb	UCBEcb(*)	IE(2)	BayesUCB
Bubeck4	uninformative	5	independent	Online	4	OLKG	UCB	SR	UCBV
GPR	Default	0.3	correlated	Offline	4	kriging	kgcb	IE(*)	EXPT

Figure 6.1 Snapshot of the spreadsheet front-end of MOLTE. Each line lists a test problem, along with choices on how to construct a prior (maximum likelihood estimation, pre-specified, uninformative), the measurement budget (as a fraction of the number of alternatives), belief model (independent or correlated), and whether it is offline vs. online evaluation. Then there is a list of policies, where the first policy is used as the reference policy against which the other policies are compared. Tunable parameters can be specified, or a "*" indicates the package should do the tuning.

to that problem. Each line of the spreadsheet refers to a different test problem in the library, which is captured by a .m file with the name of the problem (such as "Branin.m"). The user then has to specify

- Prior - This is how to construct a prior. Some choices are
  - MLE - Maximum likelihood estimation -
  - Default - Here we assume that that the prior is encoded within the specification of the problem.
  - Uninformative - An uninformative prior means that we have no information at all about the function (think of infinite variance in the prior).
- Measurement budget - this is specified as a fraction of the number of alternatives.
- Belief model - Independent or correlated beliefs (the system can only handle lookup table belief models).
- Offline vs. online - Offline means we evaluate a policy based on the terminal reward (after the experimental budget has been exhausted). Online means evaluating a policy based on the total rewards earned during the evaluation (the classic bandit setting).

A partial list of the test problems is given in table 6.1.

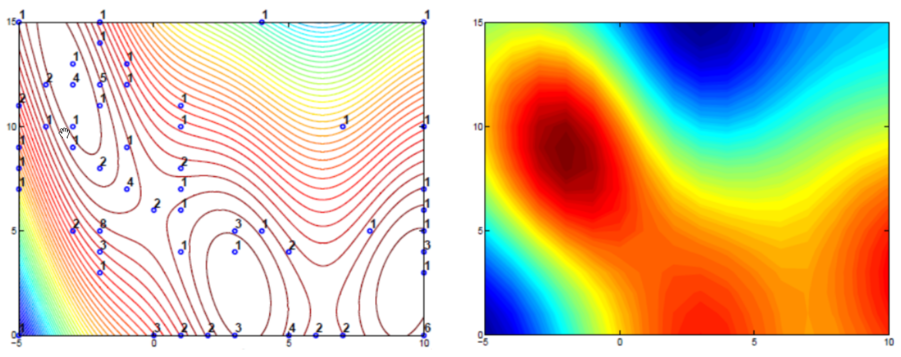
Then, for each problem, the user may list any number of pre-coded learning policies, shown in table 6.2. Many (but not all) of these policies have a tunable parameter. When these exist, the user may specify (for interval estimation) "IE(2.2)" if the parameter should be set to 2.2, or "IE(*)" to request that MOLTE do automated tuning. When automated tuning is requested, MOLTE first searches over the range 10^{-5} to 10^{5}, finds the right order of magnitude, and then refines the search.

MOLTE outputs a number of tabular and graphical reports to summarize the performance of the different policies. Figure 6.2 shows a graphical picture of the sampling pattern for a two-dimensional problem for some policy; these plots provide insights into the nature of the different behaviors. Figure 6.3 provides a frequency distribution of the difference between each policy and the reference policy.

MOLTE also produces tabular performance statistics, including both the opportunity cost relative to the reference policy, and the probability a policy outperforms the

Bubeck1 Bubeck7
Asymmetric Unimodular Functions
Rosenbrock function
Pinter's function
Goldstein function
Griewank function
Braninâ^s function
Axis parallel hyper-ellipsoid function
Rastriginâ^s function
Ackleyâ^s function
Six-hump camel back function
Easom function
Gaussian process regression functions
Payload delivery problem
Nanoparticle design

Table 6.1 A list of some of the test problems in MOLTE.



(a) Sampling distribution (b) Function heatmap

Figure 6.2 Example of the sampling distribution from MOLTE, and the resulting function approximation shown as a heatmap.

reference policy over many simulations. An example is shown in figure 6.3. MOLTE outputs the Latex code for the body of the future.

MOLTE makes it easy to add new problems and new policies by just following the style of other problems and policies. The software and a users manual is available at

Interval Estimation (IE)
Kriging
UCB (independent beliefs)
UCBcb (correlated beliefs)
UCBNormal
UCB-E (independent beliefs)
UCBEcb (correlated beliefs)
UCB-V (independent beliefs)
UCBVcb (correlated beliefs)
Bayes-UCB
KL-UCB
Knowledge gradient policy for offline learning
Knowledge gradient for online learning
Successive rejects
Thompson sampling
Pure exploration
Pure exploitation

Table 6.2 A list of policies that are available in MOLTE.

`http://castlelab.princeton.edu/software/`.

6.9 DISCUSSION

The purpose of this chapter was to provide a more comprehensive perspective of learning problems by covering all the major dimensions of any sequential learning problem. The four classes of policies provides an equally comprehensive roadmap for designing policies that cuts across the different communities that have contributed to solving this important problem class.

One of the challenges of optimal learning is that important contributions have been made in specific subcommunities with a style that reflects both the characteristics of specific problem classes as well as the culture of different research communities. For example, the bandit community, which has attracted the attention of computer science, tends to emphasize online problems with relatively small action spaces, no switchover costs, a Bayesian belief model and a desire to derive provable regret bounds. The simulation optimization community, which tends to focus on finding the best of a small set of designs using discrete event simulation, also deals with

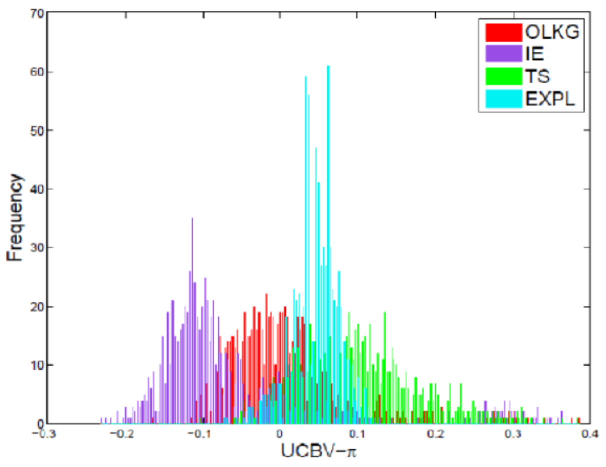


Figure 6.3 Frequency distribution of difference in performance between each policy and the reference policy.

Problem Class	IE		UCBE		UCBV		Kriging		TS		EXPL	
	OC	Prob.	OC	Prob.	OC	Prob.	OC	Prob.	OC	Prob.	OC	Prob.
Goldstein	-0.061	0.81	-0.097	0.92	-0.003	0.45	-0.031	0.73	0.100	0.09	0.041	0.16
AUF_HNoise	0.058	0.40	0.022	0.43	0.037	0.54	0.031	0.39	0.073	0.22	0.047	0.48
AUF_MNoise	0.043	0.29	0.027	0.42	0.343	0.21	0.023	0.28	0.173	0.21	-0.057	0.52
AUF_LNoise	-0.043	0.73	-0.013	0.64	0.053	0.51	0.005	0.53	0.038	0.20	0.003	0.62
Branin	-0.027	0.76	0.025	0.24	0.026	0.26	0.004	0.54	0.041	0.07	0.123	0.00
Ackley	0.007	0.42	0.04	0.41	0.106	0.20	0.037	0.42	0.100	0.23	0.344	0.00
HyperEllipsoid	-0.059	0.73	0.064	0.12	0.08	0.07	0.146	0.22	0.011	0.38	0.243	0.03
Pinter	-0.028	0.56	-0.003	0.51	0.029	0.42	-0.055	0.65	0.122	0.19	0.177	0.04
Rastrigin	-0.082	0.70	-0.03	0.56	0.162	0.04	-0.026	0.57	0.136	0.08	0.203	0.01

Table 6.3 Example of table produced by MOLTE. MOLTE outputs the latex code for the body of the table; headings have to be provided. OC refers to opportunity cost, and Prob. is the probability the policy outperforms the reference policy specified in the spreadsheet.

small choice sets but uses offline learning of a model with significant switching costs, a frequentist belief model, and a desire to show asymptotic optimality.

6.10 BIBLIOGRAPHIC NOTES

Sections 6.1-6.5 - This representation of learning problems is new, but is based on the modeling framework for dynamic programs presented in Powell (2011). The notion of the “hyperstate” to solve the optimal learning problem was put forth by Bellman & Kalaba (1959); further efforts in this direction were undertaken by Cozzolino et al. (1965), Martin (1967) and Satia & Lave (1973).

Section 6.5 - The different objective functions have been proposed by different authors in different communities. Expected opportunity cost lends itself better to the Bayesian approach (Chick & Inoue 2001), but can also be analyzed in a frequentist setting (Chick 2003, Chick & Wu 2005). A review of early research on indifference zone procedures is given in Bechhofer et al. (1968), with a review of somewhat more recent papers given by Bechhofer et al. (1995).

Section 6.7 - The truth-from-prior approach is used in Bayesian problems; see e.g. Ryzhov et al. (2011). Vermorel & Mohri (2005) and Chhabra & Das (2011) also make use of other evaluation strategies such sampling from an alternative prior or from an empirical dataset. Lai & Robbins (1985) provides the original paper on regret bounds for upper confidence bound policies. Regret bounds continue to be popular, usually in connection with upper confidence bound methods; see, for example, Agrawal (1995), Auer et al. (2002), Auer et al. (2008), Bartlett et al. (2008), Kleinberg et al. (2010), or Srinivas et al. (2010).

PROBLEMS

In each of the exercises below, you are given a situation that involves learning. For each situation, you need to describe the five fundamental dimensions of a learning model, including:

- Carefully define the state variable, giving variable names to each component. Clearly distinguish the belief state (what you think about unknown parameters whose beliefs are evolving over time) and any physical state variables.
- Define the experimental decision (how you are collecting information) and the implementation decision (what you are doing with the information). Note that these may be the same.
- Define the exogenous information. What are you observing? What is the source of the information? Is the information possibly changing your belief about a parameter?
- Describe the equations that make up the transition function. Distinguish between the equations used to update your belief state from those that update any physical or informational state variables that may be present.
- Define your objective function. Here is where you are going to distinguish between offline and online learning. Be sure to differentiate costs of measuring from implementation costs.

Note that in our modeling, we are ignoring the dimension of designing policies. This is addressed in all the remaining chapters of this volume. You may feel that you need information not specified in the problem description, so you should just highlight any missing elements, or make up elements to round out your model.

6.1 An entrepreneur would like to market the use of portable solar panels for recharging cell phones in Africa. The idea is to purchase a number of these solar panels and then to let individuals try to start businesses in different regions of Africa. Each region will face its own unique characteristics in terms of need, competition for alternative sources of energy and the ability of the local population to pay. For example, the market responds badly to decisions to raise prices. Focus on the problem faced by a single individual who has to figure out a pricing strategy as quickly as possible.

6.2 A pharmaceutical company is faced with the problem of performing a series of market research studies to determine the best pricing for a drug. The market can be expected to respond to recent pricing behavior, and the performance of the drug. The company would like to strike a balance between maximizing revenue and minimizing costs related to the market research. Market research studies may be run in local regions while the drug is still being marketed nationally.

6.3 The Centers for Disease Control wants to collect information so that it can best understand the scope of an outbreak of a virulent new virus in the northeast of the United States. On a limited budget, the CDC would like to manage a single team of technicians who will set up a tent to test people as they walk by. The tent will typically be set up for 1-3 days before the team moves to another location.

6.4 An analyst is using an expensive computer simulation to model the dynamics of wind and its effect on generating electricity. It is important to understand the impact of very low periods of wind, but obtaining these observations can require simulating years, which can take weeks of time on the computer. The analyst is using the simulator to design the location of wind farms and investments in the power grid. She might run shorter simulations to do quick evaluations to eliminate poor designs, but much longer simulations are needed to obtain accurate estimates of the likelihood that a particular design will be susceptible to blackouts.