CHAPTER 9

LARGE CHOICE SETS

While there are problems where the number of alternative decisions $\{x_1, \ldots, x_M\}$ is small enough to be enumerated, it is quite easy to encounter problems where the potential set is extremely large (tens of thousands to millions). This typically arises when x is a vector, but sometimes there are simply many choices (imagine choosing ads or movies to show an online user).

In this chapter, we consider three classes of problems where large choice sets arise:

Subset selection These arise when we have to choose L elements out of a set K, such as choosing five basketball players from a team of 12. The number of potential subsets grows very quickly as K and L increase.

Hierarchical models Imagine having to pick the best person to do a job, or the best ad to maximize ad-clicks, where the choice is characterized by a multidimensional attribute vector, producing an extremely large set of choices.

Continuous vectors There are many problems where x is a multi-dimensional vector $(x_1, \ldots, x_K)$, where x_k might be continuous or an integer.

These problems will allow us to illustrate several powerful strategies that should provide a good starting toolbox for a wide range of problem settings.

9.1 SAMPLED EXPERIMENTS

A powerful strategy for handling large experimental spaces is to turn a large set of experiments into a small one by choosing a smaller sample from the original large set. If x is multidimensional, we can sample each dimension independently, rejecting any that are not feasible or acceptable. This is a simple but powerful idea that is easy to implement. How well it works depends on the structure of the problem and the nature of the underlying belief model.

Sampling is effective when we use a belief model that allows us to generalize what we learn. This is the case if we have correlated beliefs (presented in section 4.5), a linear (or nonlinear) belief model (which we presented in chapters 7 and 8), or a hierarchical belief model (which we present below in section 9.3). The core idea is that with some form of generalized learning, we do not have to test every experiment in $\mathcal{X}$. Rather, we just need to sample enough to produce a reasonable estimate of our belief model, from which we will then find the best value in the set $\mathcal{X}$.

It is easy to overlook that most statistics is done using sampled experiments. The most standard is when we are given a dataset consisting of N data points $(y^n, x^n)_{n=1}^N$ from which we have to fit a statistical model. We may have no idea how the inputs x^n were chosen. Often, they come from a process over which we have no control. For example, x^n might be the attributes of a patient and y^n is the cost of the medical procedure. We typically cannot control which patients walk through the door.

The size of $\mathcal{X}$ is much more important for a value of information policy than it is for the other heuristics. Thus, we might need to draw a relatively smaller sample for the purpose of computing the knowledge gradient or expected improvement just to simplify these calculations. Once the experiments are complete, we can use a much larger sample to try to find the best design.

There are different strategies for generating and using sampled experiments:

Static samples We can draw a single sample from the set $\mathcal{X}$ and then proceed using just this sampled set for all calculations.

Iterative resampling Since our belief model provides estimates for all $x \in \mathcal{X}$, and not just the sampled experiments, we can draw an entirely new sample each time we want to find an experiment to run. This reduces the exposure to the vagaries of a single sample.

Using a sampled subset of experiments from $\mathcal{X}$, combined with a generalized belief model, is an exceptionally powerful strategy. The size of the sample depends on the behavior of the belief model.

It is possible to approach virtually any problem regardless of the size of the experimental set $\mathcal{X}$, and regardless of the complexity of the underlying belief model, using the idea of a sampled experimental set. You simply have to choose a sample $x_1, \ldots, x_M$, and then use the tools for discrete alternatives that we have presented in depth. However, even if you can exploit the structure of correlated beliefs, it will still be harder to estimate $\mu_1, \ldots, \mu_M$ (where M might be 1000), when you can reduce this to a parametric belief model $f(x|\theta)$ where $\theta = (\theta_1, \ldots, \theta_K)$.

9.2 SUBSET SELECTION

Consider the problem where we have to pick the four best rowers to man a four-person shell for rowing competitions. The coach can rank the 15 rowers on his team based on how they perform on ergonomic machines, but it is well known that people perform differently in a boat, and people interact in ways that affect their performance. However, there is over 1,365 ways of choosing four people out of a set of 15, so it is not possible to evaluate every possible team.

Our challenge is to find the best subset of a discrete choice set. This problem, known as *subset selection*, arises in numerous settings and has a classical formulation as a mathematical program. For example, we might have M items, where $x_i = 1$ if we have chosen i to be in our subset. Normally we have some sort of budget constraint, so we have to choose the vector x to satisfy this constraint. If we knew that item i made a contribution c_i, we could choose our elements by solving

$$\max_x \sum_{i=1}^{M} c_i x_i$$

subject to

$$\begin{aligned} \sum_{i=1}^{M} a_i x_i &\leq B, \\ x_i &\in (0,1). \end{aligned}$$

If $a_i = 1$, then B is simply a limit on how many items we can have (for example, four rowers). The coefficient a_i might be the cost of using i, in which case B is a monetary budget constraint. In this case, we have an instance of the well-known knapsack problem.

Subset selection problems include forming a team, identifying investment portfolios, prescribing a course of treatment for an illness, and choosing tools to solve a problem. In all of these problems, we have to choose a subset of elements that interact with each other in a relatively complex way. Our specific interest is in problems where we do not have a simple deterministic, linear objective function. Because of the interactions between elements, the problem is harder than simply not knowing the coefficients c_i. However, we may have a learning budget allowing us to experiment with several subsets before we have to choose one. For example, a basketball coach may hold some practice games to learn how well different players work together, before committing to a lineup for the season.

We can apply optimal learning concepts to subset selection. In theory, the tools developed in Chapters 3 and 4 already allow us to handle this problem. We can view subset selection as an instance of a ranking and selection problem with correlated beliefs. Each alternative in the ranking and selection problem corresponds to one possible subset. In the problem of choosing four rowers out of 15 possible candidates, we thus have a total of $M = \binom{15}{4} = 1,365$ alternatives. We model the values of different subsets separately. If we choose rowers $\{\text{Anne}, \text{Mary}, \text{Cathy}, \text{Tara}\}$ and $\{\text{Anne}, \text{Mary}, \text{Susan}, \text{Tara}\}$, these two subsets would be considered as two distinct alternatives, each with its own value. However, our beliefs about these two values are heavily correlated, because these two subsets have three elements in common. Learning about the effectiveness of one subset should also provide information about the other subset.

As long as we do a good job modeling our prior beliefs (particularly our beliefs about the correlations between alternatives), we can then apply one of the learning techniques from Chapter 3, or the knowledge gradient method from Chapter 4, to decide which subset we want to learn about. Conceptually, we can approach the problem of subset selection entirely within the framework of ranking and selection. For smaller problems such as choosing four rowers out of 15, this approach will work fine.

However, one characteristic specific to subset selection problems is that they tend to be quite large. Suppose that we need to choose five items from a list of ten. Five and ten are small numbers, but the number of possible subsets is $\binom{10}{5} = 252$. If we slightly increase the size of the problem, and now choose six items from a list of twelve, the number of subsets is $\binom{12}{6} = 924$. The number of alternatives grows combinatorially as we add more items to our list. Choosing ten items from a list of twenty results in $184,756$ alternatives!

In Chapter 3, we did not really specify a problem size. Rather, we analyzed a generic problem with M alternatives. In theory, the techniques we developed can handle any M, but as a general rule we are limited to M equal to around 1,000. When we are choosing subsets, the number of subsets can quickly grow much larger than this.

Subset selection problems come in many varieties. We are primarily interested in applications where there is some relationship between subsets so we can draw on our tools using correlated beliefs.

- Designing a diabetes treatment - Type 2 diabetes is an illness where the body is unable to properly react to insulin, a hormone used to remove sugar from the bloodstream. Most diabetes drugs fall into one of several categories; for example, sensitizers increase the body's sensitivity to insulin, whereas secretagogues stimulate the body to secrete more insulin. A course of treatment for a single patient may contain drugs from multiple categories, as well as multiple complementary drugs within a single category. Clinical trials can help us learn how well two drugs complement each other.

- Choosing a starting basketball lineup - The coach wants to find five players who provide the best performance over the first quarter of a basketball game. He knows how many points each player scores from past games, and has other statistics such as rebounds, assists and blocks, but the simple reality is that there is no easy formula that predicts how people will interact with each other. Each time he tries out a team, he observes the total score (which might be viewed as a correction to a simpler formula). The coach has 12 players, implying that he has to pick the best set of 5 out of a team of 12, which means evaluating 792 different teams. Fortunately, the observations should be correlated. We would not know the correlations exactly, but we might start with a covariance matrix that is proportional to the number of players in common between two teams.

- Choosing movies to display to a user - We wish to select a sample of movies (perhaps 6 or 8) out of a population of thousands to display to a user in the hope that she will select one to rent or purchase. The value of a set is given by the probability that she selects one, which requires guessing whether she is interested in action, drama, comedy, or is more influenced by the lead actor/actress. We have to find the best subset of movies to display to maximize the probability of a purchase. It is important to try a mixture, because if we display all action movies to someone who prefers drama, we will not be successful.

- Product assortment planning - A retail company has to choose a set of products to offer at its outlet store, or to display in a storefront window. The profitability of a product, or its effectiveness at drawing customers to a store, may also depend on the presence of other products in the assortment (e.g. accessories for an iPod). It is also important to display a diverse set of products that may attract buyers from different demographic groups. We have a short period of time to experiment with different assortments at a particular store in an attempt to discover the best one.

- Shortest path problems - Finding a path through a network with uncertain costs (or travel times) involves finding a subset of links out of the full set of links in the network. Of course, these links have to form a path that joins origin to destination.

9.2.1 Choosing a subset

We show how ranking and selection can be used to find a subset in the context of designing a diabetes treatment. In the latter stages of clinical trials, the search has typically been narrowed to a relatively small set of the most promising drugs. We can create a combination treatment consisting of multiple drugs from this set.

The effectiveness of a diabetes treatment can be measured in terms of blood sugar reduction. Blood sugar level is measured in millimoles per liter, after the patient has not eaten for eight hours; this is known as the "fasting plasma glucose" or FPG level. For a healthy person, the FPG level is about 4-6 mmol/L. A diabetic can show FPG levels of up to 10-15 mmol/L. A single drug can reduce FPG level by an amount between 0.5 and 2 mmol/L.

We design a combination treatment for two reasons. First, two drugs yield a bigger FPG reduction than one, and second, combination treatments tend to have lower risks of side effects. Prescribing three drugs instead of one may give us an FPG reduction between 2 and 5 mmol/L. Due to interactions between drugs, FPG reduction obtained from the combination is not a straightforward sum of the reductions we might observe from individual drugs.

9.2.2 Setting prior means and variances

Let us consider an example with six drugs. Our list may include certain basic elements such as conventional treatment (recommending diet and exercise to the patient), metformin (the first drug of choice in most cases), and insulin injections, as well as newer compounds such as rosiglitazone, glibenclamide and chlorpropamide.

Our goal is to find the best possible combination of three of these drugs. The total number of subsets is still quite small, $\binom{6}{3} = 20$.

Let $\mathcal{S} = \{1, ..., 6\}$ be our set of $S = 6$ drugs, from which we have to choose a subset of size s. (These should be pronounced "script S" and "big S.") Let μ_x be the true value of a subset $x \subseteq S$, that is, the true FPG reduction achieved by treatment x on average. Now suppose that we have a large population of patients in roughly the same poor health condition. We divide them into N groups and then we prescribe a treatment for each group. When we assign the $(n+1)$st group to treatment x, our observation W_x^{n+1} is the average FPG reduction across all the patients in the group. Sample averages are approximately normal, so we make our usual modeling assumption $W_x^{n+1} \sim \mathcal{N}\left(\mu_x, \sigma_W^2\right)$, where the variance σ_W^2 is assumed known (but somehow estimated or guessed in practice).

All we need now in order to apply optimal learning is a prior distribution on the values μ_x. We will use a multivariate normal distribution (accounting for correlations between subsets). The prior mean $\bar{\mu}_x^0$ on the true value of treatment x tends to be easier to choose, because it comes directly from our domain knowledge about the problem. In light of what we know about blood sugar levels, we might let $\bar{\mu}_x^0$ be a number between 2 and 5 mmol/L. The prior variance Σ_{xx}^0 represents a rough guess about the range of our uncertainty about the true value. If we are reasonably sure that the true value is between 2 and 5, we might let the variance be 0.5, meaning that we believe that μ_x falls in the range $\bar{\mu}_x^0 \pm 2 \cdot \sqrt{0.5}$.

9.2.3 Two strategies for setting prior covariances

While prior means and variances can be chosen with the help of knowledge about the problem, covariances between the true values are trickier to estimate. In actual clinical trials, we can observe the effectiveness of a treatment directly, but covariances between treatments can only be inferred from the results of individual trials. Fortunately, we can still obtain good performance with a simple, heuristic covariance structure. Our prior mainly needs to capture the fact that subsets are more heavily correlated if they have more elements in common.

Our first rule of thumb is based purely on the number of such elements. In the problem of choosing three drugs from a list of six, let $x = \{1, 2, 3\}$ and $y = \{2, 3, 5\}$ be two possible treatments. These subsets have $2/3$ elements in common, so we can simply choose $2/3$ as the correlation coefficient of these two subsets in our prior distribution. That is, the prior correlation coefficient is set using

$$\rho_{xy}^0 = \frac{1}{S} \left| x \cap y \right| .$$

The prior covariance is simply

$$\Sigma_{xy}^0 = \rho_{xy}^0 \sigma_x^0 \sigma_y^0. \tag{9.1}$$

We calculate these covariances for every possible x and y. This simple heuristic produces high correlations between subsets with more common elements.

In some problems, we may have more domain knowledge about individual items on our list than about subsets. For example, we may have access to clinical data about the drug rosiglitazone, tested independently of other drugs. Or, we may have some data on the travel time on a particular street or region, but no travel time data for a complex travel route that takes us through multiple regions. In this case, it may be easier to construct a prior on the value of a subset using our information about individual components.

Let μ_i^{ind} be the true value of the individual component $i \in \mathcal{S}$ (e.g. the FPG reduction achieved by the single diabetes drug i). Using our domain knowledge, we construct a prior $\mu_i^{ind} \sim \mathcal{N}\left(\bar{\mu}_i^{ind,0}, \left(\sigma_i^{ind,0}\right)^2\right)$. For the diabetes setting, we may believe that μ_i^{ind} falls in the range 1.25 ± 0.75 mmol/L. Then, our belief about a treatment x is given by

$$\bar{\mu}_x^0 = \sum_{i \in x} \bar{\mu}_i^{ind,0}, \tag{9.2}$$

and the covariance of our beliefs about treatments x and y is given by

$$\Sigma_{xy}^0 = \sum_{i \in x \cap y} \left(\sigma_i^{ind,0}\right)^2. \tag{9.3}$$

This heuristic implicitly assumes that the values of individual diabetes drugs are independent, which may not be the case. For example, the FPG reductions for multiple drugs in the same class will be correlated. If a patient reacts adversely to sensitizers, all drugs of this type will tend to perform poorly. In any case, however, our prior is not able to capture all the nuances of the problem. Instead, the prior is meant to provide some rough guidelines about the values of different alternatives and the interactions between them. Even if we assume that individual items on our list are independent, subsets will still be correlated if they have common elements. The correlation structure in (9.3) once again captures the fact that subsets with many common elements should have similar values. As we make decisions, the information we collect will allow us to obtain more accurate beliefs. To put it another way, we start by assuming a simple linear objective function, then leave it to our learning algorithm to refine these initial beliefs.

We now proceed as usual. Once we have a prior $\left(\bar{\mu}^0, \Sigma^0\right)$, we rely on our standard Bayesian updating equations to change our beliefs from that point. If we choose treatment x for the trial at time n, our beliefs for subsets y will change according to the equations

$$\begin{aligned}
\bar{\mu}_y^{n+1} &= \bar{\mu}_y^n - \frac{W_x^{n+1} - \bar{\mu}_x^n}{\sigma_W^2 + \Sigma_{xx}^n} \Sigma_{xy}^n, \\
\Sigma_{y,y'}^{n+1} &= \Sigma_{y,y'}^n - \frac{\Sigma_{x,y}^n \Sigma_{x,y'}^n}{\sigma_W^2 + \Sigma_{xx}^n}.
\end{aligned}$$

This gives us everything we need to run an algorithm such as knowledge gradient. If there are only 20 possible subsets, it is smooth sailing from here.

9.2.4 Managing large sets

Let us switch gears and consider a slightly larger problem, namely the energy portfolio selection example we introduced above. A recent study by McKinsey & Company (2007) has identified a number of promising technologies for reducing greenhouse gas emissions. These new technologies have a higher energy efficiency, and also help to reduce energy costs. A partial list of these technologies includes: residential lighting; energy-efficient water heaters and appliances; improved ventilation systems and air conditioners; heating systems and furnaces; solar technology; shell improvements for residential buildings (e.g. insulation); and fuel economy packages for cars.

Suppose that we have a total of 12 promising technologies, and our goal is to create a portfolio of six. The McKinsey study provides estimates for the abatement potential (the emissions reduction) of these technologies. The estimates represent the total abatement, measured in gigatons of CO_2 per year (ranging from 0.2 to 3.0 for different technologies), that could be achieved by implementing the technology across the entire United States. We can scale down these estimates and use the strategy from (9.2) and (9.3) to create a prior for the energy efficiency of a single portfolio of six technologies, applied to a single building.

The total number of energy portfolios in this example is 924, about 50 times more than we had in the problem of designing a diabetes treatment. We can still run the correlated KG algorithm from Chapter 4, but we will see that it will run much more than 50 times slower on this problem. The reason is because, in a problem with M subsets, the time required to compute correlated KG factors for all the subsets is proportional to $M^2 \log M$. This computational cost grows faster than the problem size. Considering that subset selection problems are already prone to very large sizes, this is a serious issue. We now discuss a way to reduce this cost using Monte Carlo simulation.

9.2.5 Using simulation to reduce the problem size

We are going to propose an idea which can be viewed as a hybrid of Thompson sampling and any other policy, although this is of greatest value for the knowledge gradient which is harder to compute. We use posterior sampling (as is done in Thompson sampling) to identify a subset of promising alternatives. We are then going to use this more reasonable subset to use tools such as correlated beliefs.

Suppose that we are at time n, with point estimate $\bar{\mu}^n = (\bar{\mu}^n_{x_1}, \ldots, \bar{\mu}^n_{x_M})$ and covariance matrix Σ^n with element $\Sigma^n_{xx'} = Cov^n(\mu_x, \mu_{x'})$. Note that each x is an entire subset, such as

$$x = (0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1).$$

A subset x might also be a path in our network, or a set of drugs in a drug regimen.

What we want to do is to create K samples of the vector μ from the distribution $\mathcal{N}(\bar{\mu}^n, \Sigma^n)$. Let $\hat{\mu}^k$ be the kth sample of this vector, and let

$$\hat{x}^k = \arg\max_{x \in \mathcal{X}} \hat{\mu}^k_x$$

be the best alternative out of this sample of beliefs across all alternatives. Constructing this set might require computing $\hat{\mu}_x^k$ for each alternative x, but not always. For example, if we are finding paths through a graph, we would randomly generate costs on each link, and then solve a shortest path problem for this set of random costs, which has the effect of implicitly enumerating all possible paths without actually enumerating them.

If we repeat this K times, we get a set of alternatives $\hat{x}^1, \dots, \hat{x}^K$, which may have some duplicates which have to be removed (if we insist on having a full set K, we can continue the process until we have K distinct alternatives). This set is likely to consist of K high quality alternatives, but as with Thompson sampling, there is an element of exploration.

To see how this works, consider a numerical example. Suppose that we have five alternatives with $\bar{\mu}^n = (10, 13, 12, 16, 8)^T$ and

$$\Sigma^n = \begin{bmatrix} 22 & 13 & 14 & 15 & 14 \\ 13 & 16 & 18 & 13 & 12 \\ 14 & 18 & 24 & 12 & 14 \\ 15 & 13 & 12 & 14 & 11 \\ 14 & 12 & 14 & 11 & 16 \end{bmatrix}.$$

We decide to reduce the number of alternatives. To that end, we generate $K = 4$ samples from the distribution $\mathcal{N}(\bar{\mu}^n, \Sigma^n)$. Table 9.1 gives the simulated values for all five alternatives on all four sample paths. Observe that, according to our prior, alternative 4 is believed to be the best. This same alternative is also the winner on $2/4$ sample paths. However, alternatives 2 and 3 each win on a single sample path, even though these alternatives are believed to be the second- and third-best according to the prior. The set of winners is thus $\{2, 3, 4\}$. In this way, Monte Carlo sampling can give us a reasonably diverse choice set, which includes many promising alternatives, but does not include alternatives that seem to be hopelessly bad.

It can be shown that the marginal distribution of (μ_2, μ_3, μ_4) is multivariate normal with parameters

$$\bar{\mu}^{MC,n} = \begin{bmatrix} 13 \\ 14 \\ 15 \end{bmatrix}, \qquad \Sigma^{MC,n} = \begin{bmatrix} 16 & 18 & 13 \\ 18 & 24 & 12 \\ 13 & 12 & 14 \end{bmatrix}.$$

x	$\hat{\mu}_x^n(\omega^1)$	$\hat{\mu}_x^n(\omega^2)$	$\hat{\mu}_x^n(\omega^3)$	$\hat{\mu}_x^n(\omega^4)$
1	15.59	11.38	3.73	10.48
2	17.24	18.49	7.30	**18.98**
3	18.06	**20.77**	2.61	18.88
4	**19.32**	17.97	**12.53**	18.73
5	12.37	13.41	4.08	11.55

Table 9.1 Simulated values of five alternatives based on the prior distribution.

That is, if we throw out alternatives 1 and 5, the distribution of our belief about the remaining alternatives is the same as before. We just need to throw out the rows and columns of Σ^n, and the elements of $\bar{\mu}^n$, having to do with alternatives 1 and 5. We obtain the reduced prior $\mathcal{N}\left(\bar{\mu}^{MC,n}, \Sigma^{MC,n}\right)$, where the notation MC refers to the use of Monte Carlo sampling to reduce the prior.

We can formalize this procedure as follows. Let K_0 be the number of unique winners obtained from Monte Carlo sampling (with duplicates removed; this number would be 3 in the preceding example). We can enumerate the winners as $\bar{x}_1^{MC,n}, ...,$ $\bar{x}_{K_0}^{MC,n}$. We can define a matrix A^n of size $M \times K_0$ by

$$A^n = \left[e_{\bar{x}_1^{MC,n}}, ..., e_{\bar{x}_{K_0}^{MC,n}}\right].$$

Recall that e_x is a vector of zeroes with only the xth component set to 1. In the preceding example, we would write

$$A^n = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Then, we can calculate the reduced prior using the equations

$$\begin{aligned} \bar{\mu}^{MC,n} &= \left(A^n\right)^T \bar{\mu}^n, \\ \Sigma^{MC,n} &= \left(A^n\right)^T \Sigma^n A^n. \end{aligned}$$

We can now run the knowledge gradient algorithm using $\bar{\mu}^{MC,n}$ and $\Sigma^{MC,n}$ as the prior instead of the original parameters $\bar{\mu}^n$ and Σ^n. The computational cost is thus at most $K^2 \log K$, and likely less if there are many duplicates. We choose K to be much smaller than M. In a problem with 924 alternatives, we can obtain good results with $K = 30$.

9.2.6 Computational issues

Monte Carlo simulation allows us to greatly reduce the cost of calculating knowledge gradients. However, it carries a new cost for running the simulations. Recall from Section 2.7 that a single sample from a multivariate normal distribution can be created using the equation

$$\hat{\mu}^n = \bar{\mu}^n + C^n Z$$

where $Z = (Z_1, ..., Z_M)$ is a vector of standard normals, and C^n is a "square-root matrix" satisfying $C^n \left(C^n\right)^T = \Sigma^n$. The standard normals are easy enough to generate, but calculating the square root matrix can be costly. In the worst case, the time required to compute C^n is proportional to M^3, which is actually greater than the $M^2 \log M$ needed to compute the full set of knowledge gradients in the first place.

Fortunately, there is an elegant work-around for this issue. The square-root matrix can be computed recursively using the formula

$$C^{n+1} = C^n - \frac{\Sigma^n e_{x^n} e_{x^n}^T C^n}{\left(\sigma_W^2 + \Sigma^n_{x^n x^n}\right)\left(1 + \sqrt{\frac{\sigma_W^2}{\sigma_W^2 + \Sigma^n_{x^n x^n}}}\right)}. \tag{9.4}$$

This equation is entirely analogous to (2.19). Given our decision x^n at time n, C^{n+1} can be calculated directly from the beliefs at time n. The updating equation (9.4) even has a similar form to (2.19), and requires the same computational effort.

Thus, we only need to perform the expensive calculations required to compute the square root of a matrix once, to obtain C^0 from Σ^0. In MATLAB, this can be done using the routine `chol` (for Cholesky factorization, the name for the square root procedure). After that, we keep track of the square root matrix at each time step, and the next square root is computed recursively at a low cost. We are saving a lot of computing time by extending our belief state to include C^n as well as Σ^n.

It may also sometimes happen that, due to numerical rounding errors, the prior covariance matrix Σ^0 that we build using the methods from Section 9.2.3 may be reported by MATLAB as not being positive semidefinite (a necessary condition for computing square root matrices). This issue can often be resolved by simply adding a diagonal matrix $\varepsilon \cdot I$ to Σ^0. Here, I is the identity matrix and ε is a small positive number (e.g. 0.001). This quick fix is often enough for computing C^0, while still preserving the general correlation structure of subset selection.

9.2.7 Experiments

Earlier, we mentioned that the number K of Monte Carlo samples can be chosen to be much smaller than the total number M of subsets. One important question is exactly how many samples we need to take. Figure 9.1 shows how the size K_0 of the reduced choice set (recall that this is the number of winners obtained from Monte Carlo sampling) increases with the sample size K on a variant of the energy portfolio selection problem. Due to the random sampling, the curve is noisy, but we can see that K_0 grows fairly slowly. With $K = 30$ samples, we obtain a reduced choice set of $K_0 \approx 20$ alternatives. This number is roughly doubled when K increases to 300. Thus, as long as our prior presents a roughly accurate picture of the values of the alternatives (in Chapter 6 we referred to this as a "truth-from-prior" problem), we can safely ignore the vast majority of the possible subsets and focus only on several dozen of the most promising choices.

Figure 9.2.7 illustrates the performance, expressed using the average opportunity cost metric from (6.10), of the Monte Carlo KG (MCKG) policy as a function of the sample size K. We see that, once $K \geq 30$, performance begins to level off, and adding more samples does not yield substantially better results. Here, we see that small sample sizes lead to more erratic behavior. In each iteration, our reduced choice set is randomly missing some important, promising portfolios, and it also randomly includes some portfolios that have previously been under-represented. Our policy thus measures more portfolios in an effort to gain enough information about the most relevant alternatives. On the other hand, once the sample size is large enough that all

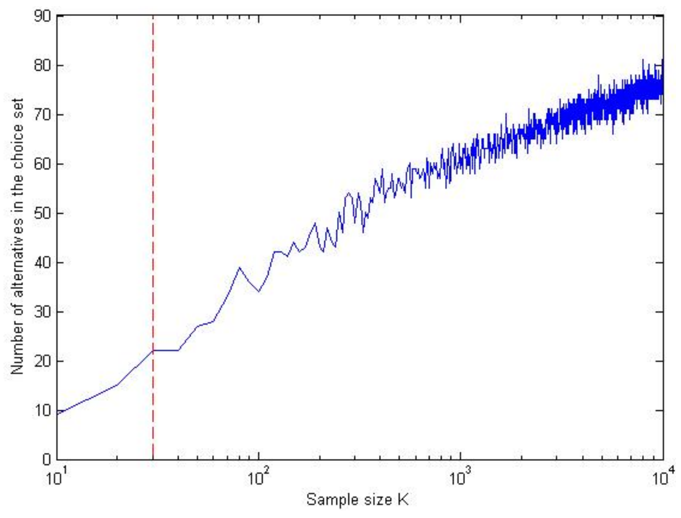

Figure 9.1 Growth of the size K_0 of the reduced choice set as a function of the sample size K.

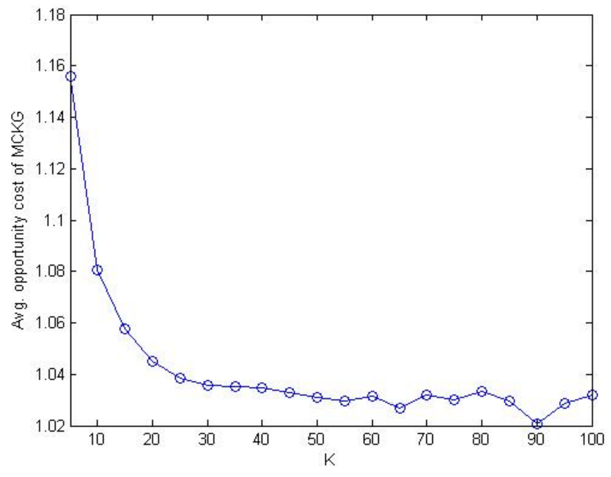

Figure 9.2 Illustration of performance of the Monte Carlo KG (MCKG) policy as the sample size K increases (from Ryzhov & Powell (2009*b*)).

of these important portfolios are consistently included in the reduced choice set, the policy is able to more accurately discern which alternatives need to be measured.

Finally, Figure 9.3 compares the performance of MCKG for a fixed sample size $K = 25$ to several other policies. In addition to our usual mainstays – approximate Gittins indices, interval estimation, and pure exploitation – we also apply the independent KG policy, in which we apply the formula from (4.13) to make a decision, thus temporarily ignoring the correlations between subsets when we make decisions, but incorporating them into our learning model. In Section 4.5, this approach was also called "hybrid KG." Although this idea is clearly not as good as correlated KG, it

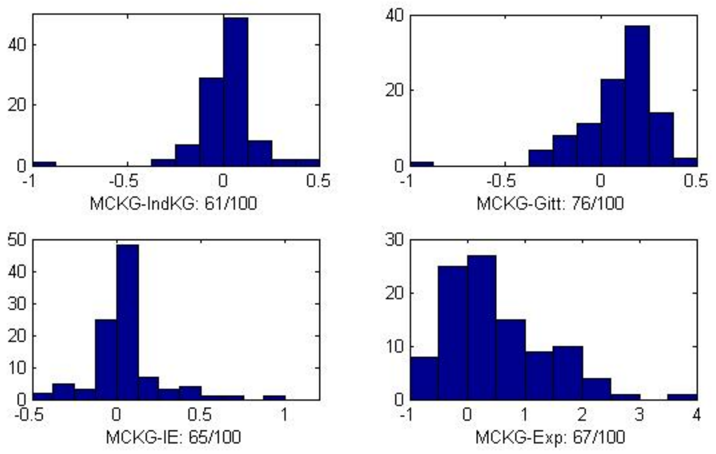

Figure 9.3 Performance of the MCKG policy relative to other policies for $K = 25$. The numbers indicate the number of problems (e.g. 61/100) where MCKG outperformed the competing policy. Results taken from Ryzhov & Powell (2009*b*).

makes sense to use it in this setting because of its much lower computational cost: the time to calculate the formula for each alternative is proportional to M, as opposed to $M^2 \log M$ for correlated KG.

We find that Monte Carlo KG still outperforms the other policies. These results are consistent with what we know about correlated KG from Section 4.5. The Monte Carlo component does not seem to harm performance, as long as K is around 20 or 30. Note that Figure 9.3 does not include a comparison with the full correlated KG policy (computing correlated KG factors for the full choice set). Even in our version of energy portfolio selection, which is still not unreasonably large, it simply takes too much time to run correlated KG. The Monte Carlo modification allows us to work around this computational limitation.

9.3 HIERARCHICAL LEARNING FOR MULTIATTRIBUTE CHOICES

There are situations where we face a large number of possible choices. Some examples are:

■ **EXAMPLE 9.1**

An online video store needs to display a set of movies that a customer is likely to choose. This set can be customized to the particular user, but we need to identify which movies out of thousands that this customer is most likely to choose.

■ **EXAMPLE 9.2**

A company is growing quickly and needs to hire people to fill a variety of new positions. After posting the job requirements on various internet job boards, the human resources director is quickly overwhelmed with thousands of applications.

■ **EXAMPLE 9.3**

The Department of Homeland Security needs to identify websites that suggest terrorist origin or intent. Algorithms can process the websites, but it is necessary to show some of them to experts who can do a better job of assessing risk. Which of the many thousands of websites should we show to the expert?

These are problems with extremely large choice sets, but they lack the nice structure of a subset selection problem. However, each choice is described by a set of attributes which can be exploited to create a natural hierarchy. For example, when organizing movies, we might decide that genre is most important, followed by leading actor/actress, director, and year it appeared. We can then use this structure to create a series of estimates that balance bias and variance.

We have to begin by describing how to compute estimates of a function at different levels of aggregation, since we did not cover this in our first pass through statistical estimation in chapter 9.5. Fortunately, the equations for doing this are relatively simple and quite easy to implement. We do this in two stages. Section 9.3.1 lays the foundation by showing how to estimate bias and variance. Then, section 9.3.2 describes how to perform hierarchical estimation using a simple weighting formula.

Once we have this belief model in hand, we can use this with any of our tunable policies, just as we could when we first encountered correlated beliefs. Value of information policies such as the knowledge gradient, however, are another matter, since we need to capture the updating of the belief model when we calculate the value of information. We show how to calculate the knowledge gradient in section 9.3.3.

9.3.1 Laying the foundations

Aggregation can be achieved in different ways. If x is multidimensional, we can simply ignore dimensions. Example, if estimating the value of posting a movie on a website, we might use genre as the most important attribute, followed by the year in which it was made, and then the acting lead. But we can also aggregate by using coarse representations of a continuous variable, such as the number of words in an ad, or by using advanced statistical models to pick out the most important features.

However we handle aggregation, we can represent our procedure as a mapping that reduces our original set of choices $\mathcal{X}$ to a more aggregate set that we write $\mathcal{X}^{(g)}$, which write write as

$$G^g : \mathcal{X} \rightarrow \mathcal{X}^{(g)}.$$

$\mathcal{X}^{(g)}$ represents the g^{th} level of aggregation of the domain $\mathcal{X}$. Let

$$\begin{aligned}\mathcal{G} &= \text{The set of indices corresponding to the levels of aggregation,}\\ &= \{0, 1, \ldots, G\}.\end{aligned}$$

Throughout we assume that the 0^{th} level of aggregation is the true function. Figure 9.4 illustrates the hierarchical ordering.

$g = 2$	13								
$g = 1$	10			11			12		
$g = 0$	1	2	3	4	5	6	7	8	9

Figure 9.4 Example with nine alternatives and three aggregation levels

We assume that we are sampling the original function μ_x according to

$$\hat{\mu}_x^n = \mu_x + \varepsilon^n,$$

where we may write W_x^n or just W^n if the choice of experimental decision x is clear. We need to characterize the errors that arise in our estimate of the function. Let

$$\begin{aligned}\mu_x^{(g)} &= \text{The true estimate of the } gth \text{ aggregation of the original function } \mu_x\\ \bar{\mu}_x^{(g,n)} &= \text{Our estimate of } \mu_x^{(g)} \text{ after we have run a total of } n \text{ experiments,}\\ \hat{\mu}_x^{(g,n)} &= \text{The observation } \hat{\mu}_x^n \text{ mapped to the } gth \text{ level of aggregation.}\end{aligned}$$

This estimate depends, of course, on the probability that we are going to run experiment x. In our estimation problem, this experimental distribution will be built into our estimates $\bar{\mu}_x^{(g,n)}$.

When we are working at the most disaggregate level ($g = 0$), the experiment x that we run is the observed state $x = \hat{x}^n$. For $g > 0$, the subscript x in $\bar{f}_x^{(g,n)}$ refers to $G^g(x^n)$, or the g^{th} level of aggregation of $f(x)$ at $x = x^n$. Given an observation $(x^n, \hat{f}^n)$, we would update our estimate of the $f^{(g)}(x)$ using

$$\bar{\mu}_x^{(g,n)} = (1 - \alpha_{n-1}^{(g)})\bar{\mu}_x^{(g,n-1)} + \alpha_{n-1}^{(g)}\hat{\mu}_x^n. \tag{9.5}$$

Note that equation (9.5) is performing the process of taking the observation of the function $\hat{\mu}_x^n$ at the disaggregate level x, and is smoothing it into an aggregated estimate $\bar{\mu}_x^{(g,n)}$. The aggregation function is built into the estimate $\bar{\mu}_x^{(g,n)}$.

To illustrate this, imagine that we have a truck driver who is characterized by the attributes

$$x = \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix},$$

where Loc is the current location of the driver, $Equip$ is the type of equipment he is driving, $Home$ is the location of where he lives, HoS stands for "hours of service" which captures how many hours he has worked today, and $Days$ is the number of days he has been away from home. Assume we have an observation of the amount of money the driver makes on day n, where we might write

$$\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{DOThrs} \\ \text{Days} \end{pmatrix} = \text{Money earned on day } n \qquad .$$

We can create estimates at three different levels of aggregation using

$$\begin{aligned}
\bar{\mu}^{(1,n)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \end{pmatrix} &= (1-\alpha^{(1)}_{x,n-1})\bar{\mu}^{(1,n-1)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \end{pmatrix} + \alpha^{(1)}_{x,n-1}\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix}, \\
\bar{\mu}^{(2,n)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \end{pmatrix} &= (1-\alpha^{(2)}_{x,n-1})\bar{\mu}^{(2,n-1)} \begin{pmatrix} \text{Loc} \\ \text{Equip} \end{pmatrix} + \alpha^{(2)}_{x,n-1}\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix}, \\
\bar{\mu}^{(3,n)} \begin{pmatrix} \text{Loc} \end{pmatrix} &= (1-\alpha^{(3)}_{x,n-1})\bar{\mu}^{(3,n-1)} \begin{pmatrix} \text{Loc} \end{pmatrix} + \alpha^{(3)}_{x,n-1}\hat{\mu}^n \begin{pmatrix} \text{Loc} \\ \text{Equip} \\ \text{Home} \\ \text{HoS} \\ \text{Days} \end{pmatrix}.
\end{aligned}$$

We need to estimate the variance of our estimate $\bar{\mu}^{(g,n)}_x$. To do this we are going to need

$(s^{(g,n)}_x)^2 =$ The bias-corrected sum of squares in the estimate of $\bar{\mu}^{(g,n-1)}_x$.

$(s^{(g,n)}_x)^2$ is the estimate of the variance of the observations $\hat{\mu}_x$ when we observe the function at $x = x^n$ which aggregates to $x^{(g)}$ (that is, $G^g(x^n) = x^{(g)}$).

To compute $(s^{(g,n)}_x)^2$, we have to estimate the total variation in the estimate of $\bar{\mu}^{(g,n-1)}_x$, and then separate out the bias. We first define

$\bar{\nu}^{(g,n)}_x \;=\;$ The total variation in the estimate $\bar{\mu}^{(g,n-1)}_x$.

$\bar{\delta}^{(g,n)}_x \;=\;$ The estimated bias in $\bar{\mu}^{(g,n-1)}_x$ due to aggregation.

We compute $\bar{\nu}^{(g,n)}_x$ using

$$\bar{\nu}^{(g,n)}_x = (1-\eta_{n-1})\bar{\nu}^{(g,n-1)}_x + \eta_{n-1}(\bar{\mu}^{(g,n-1)}_x - \hat{\mu}^n_x)^2,$$

where η_{n-1} follows some stepsize rule (which is typically just a constant such as 0.1). We refer to $\bar{\nu}^{(g,n)}_x$ as the total variation because it captures deviations that arise both due to measurement noise (the randomness when we compute $\hat{\mu}^n_x$) and bias (since $\bar{\mu}^{(g,n-1)}_x$ is a biased estimate of the mean of $\hat{\mu}^n_x$).

We next estimate the bias using

$$\bar{\delta}_x^{(g,n)} = \bar{\mu}_x^{(g,n)} - \bar{\mu}_x^{(0,n)}. \tag{9.6}$$

We then separate out the effect of bias to obtain

$$(s_x^{(g,n)})^2 = \begin{cases} 0 & n = 1, \\ \frac{\bar{\nu}_x^{(g,n)} - (\bar{\delta}_x^{(g,n)})^2}{1+\lambda^{n-1}} & n = 2, 3, \ldots, \end{cases} \tag{9.7}$$

We are not done. What we are really looking for is the variance of $\bar{\mu}_x^{(g,n-1)}$, which we define as

$(\bar{\sigma}_x^{(g,n)})^2 =$ The estimate of the variance of $\bar{\mu}_x^{(g,n)}$.

This is given by the formula

$$\begin{aligned} (\bar{\sigma}_x^{(g,n)})^2 &= Var[\bar{\mu}_x^{(g,n)}] \\ &= \lambda_x^{(g,n)} (s_x^{(g,n)})^2, \end{aligned} \tag{9.8}$$

where $\lambda_x^{(g,n)}$ can be computed from the recursion

$$\lambda_x^{(g,n)} = \begin{cases} (\alpha_{x,n-1}^{(g)})^2, & n = 1, \\ (1 - \alpha_{x,n-1}^{(g)})^2 \lambda_x^{(g,n-1)} + (\alpha_{x,n-1}^{(g)})^2, & n > 1. \end{cases}$$

The derivation of this formula is given in section 9.4.1 at the end of the chapter.

The different estimates are illustrated using the function depicted in figure 9.5, where we show a function at two levels of aggregation, allowing us to depict the different estimates and bias. Note that bias is very dependent on the decision x, since our underlying function may be smooth (producing very little bias at the aggregate level), or sloped, in which case the bias may be quite large or, in some places, very small.

9.3.2 Combining multiple levels of aggregation

Our next challenge is to create an estimate $\bar{\mu}_x^n$ of μ_x using the data we have collected, which we are going to do using our family of estimates $\bar{\mu}_x^{g,n}$ over the different levels of aggregation. The most natural strategy (we believe) is to take a weighted average.

$$\bar{\mu}_x^n = \sum_{g \in \mathcal{G}} w^{(g)} \bar{\mu}_x^{(g)}, \tag{9.9}$$

where $w^{(g)}$ is the weight applied to the g^{th} level of aggregation.

While this estimate looks appealing, the weights really need to depend on x since the relative accuracy of each level of aggregation will generally depend on x. For this reason, we write our weighted estimate as

$$\bar{\mu}_x^n = \sum_{g \in \mathcal{G}} w_x^{(g)} \bar{\mu}_x^{(g,n)}. \tag{9.10}$$

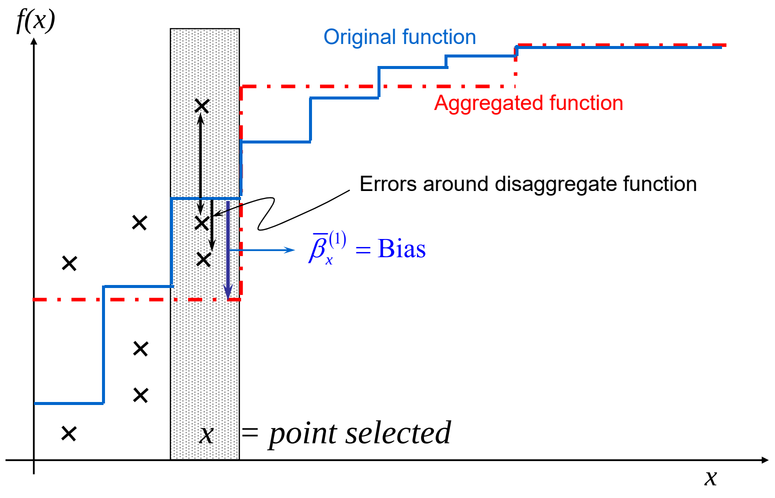
f(x)
Original function
Aggregated function
Errors around disaggregate function
$\bar{\beta}_x^{(1)}$ = Bias
x = point selected
x

Figure 9.5 Illustration of a disaggregate function, an aggregated approximation and a set of samples. For a particular state s, we show the estimate and the bias.

If the estimates $\bar{\mu}_x^{(g,n)}$ were unbiased, we could show that the best weights are proportional to the estimates of the variance of $\bar{\mu}_x^{(g,n)}$, which is to say

$$w_x^{(g)} \propto \frac{1}{(\bar{\sigma}_x^{(g,n)})^2}.$$

Since the weights should sum to one, we obtain

$$w_x^{(g)} = \left(\frac{1}{(\bar{\sigma}_x^{(g,n)})^2}\right)\left(\sum_{g\in\mathcal{G}}\frac{1}{(\bar{\sigma}_x^{(g,n)})^2}\right)^{-1}. \tag{9.11}$$

The problem here is that the estimates $\bar{\mu}_x^{(g,n)}$ are clearly not unbiased, but this is easy to fix using the calculations we presented in the previous section. Instead of using the variance, we instead use the variance plus bias squared, where the bias is given by $\bar{\delta}_x^{(g,n)}$. This gives us the weighting formula

$$w_x^{(g,n)} \propto \frac{1}{\left((\bar{\sigma}_x^{(g,n)})^2 + \left(\bar{\delta}_x^{(g,n)}\right)^2\right)} \tag{9.12}$$

Introducing the normalization constant then gives us

$$w_x^{(g,n)} = \frac{1}{\left((\bar{\sigma}_x^{(g,n)})^2 + \left(\bar{\delta}_x^{(g,n)}\right)^2\right)}\left(\sum_{g'\in\mathcal{G}}\frac{1}{\left((\bar{\sigma}_x^{(g',n)})^2 + \left(\bar{\delta}_x^{(g',n)}\right)^2\right)}\right)^{-1}. \tag{9.13}$$

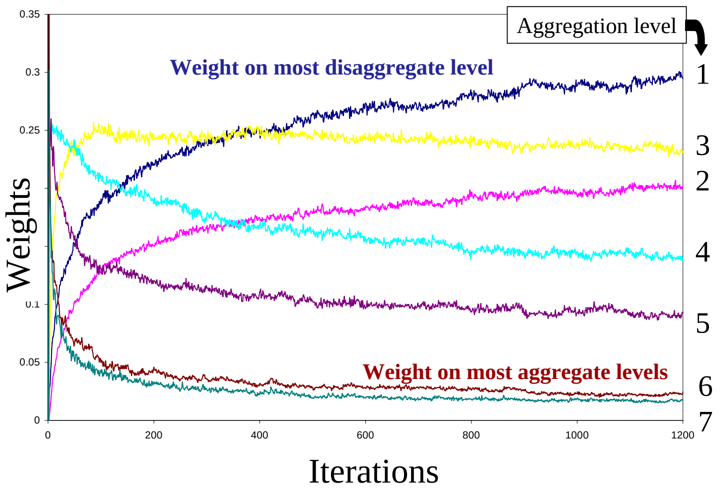

Figure 9.6 Average weight (across all states) for each level of aggregation using equation (9.13).

As might be anticipated, the weights tend to concentrate on the most aggregate levels in the early estimations when there is a lot of noise, which limits our ability to even estimate a bias. As more data comes in, we can put more weight on the more disaggregate levels, although this is not universally true across all experimental decisions x. This behavior can be seen quite clearly in figure 9.6, which plots the average weight for each level of aggregation.

9.3.3 Hierarchical knowledge gradient

We now address the challenge of computing the knowledge gradient, which we restate as

$$\nu_x^{KG}(S^n) = \mathbb{E}\left[\max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^{n+1}(x) | S^n\right] - \max_{x' \in \mathcal{X}} \bar{\mu}_{x'}^n. \tag{9.14}$$

The key to handling our hierarchical belief model in the knowledge gradient is taking advantage of its structure. We begin by repeating the updating equations for the mean and precision at level of aggregation g:

$$\bar{\mu}_x^{(g,n+1)} = \frac{\beta_x^{(g,n)}\bar{\mu}_x^{(g,n)} + \beta_x^W \hat{\mu}_x^{n+1}}{\beta_x^{(g,n)} + \beta_x^W}, \tag{9.15}$$

$$\beta_x^{(g,n+1)} = \beta_x^{(g,n)} + \beta_x^W, \tag{9.16}$$

For reasons that become clear later, we rewrite equation (9.15) in the form

$$\bar{\mu}_x^{(g,n+1)} = \bar{\mu}_x^{(g,n)} + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\hat{\mu}_x^{n+1} - \bar{\mu}_x^{(g,n)}\right).$$

We further rewrite this equation by splitting the second term using

$$\bar{\mu}_x^{(g,n+1)} = \bar{\mu}_x^{(g,n)} + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\hat{\mu}_x^{n+1} - \bar{\mu}_x^n\right) + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right).$$

We are next going to rewrite our updating formula as we did in section 4.5 using the standard normal variable Z,

$$\bar{\mu}_x^{(g,n+1)} = \bar{\mu}_x^{(g,n)} + \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right) + \tilde{\sigma}_x^{(g,n)} Z, \tag{9.17}$$

where, with some work, we can show that the predictive variance is given by

$$\tilde{\sigma}_x^{(g,n)} = \frac{\beta_x^W \sqrt{\frac{1}{\beta_x^{(g,n)}} + \frac{1}{\beta_x^W}}}{\beta_x^{(g,n)} + \beta_x^W}. \tag{9.18}$$

We next define the sets

$\mathcal{G}(x, x')$ Set of all aggregation levels that the alternatives x and x' have in common, with $\mathcal{G}(x, x') \subseteq \mathcal{G}$. In the example in figure 9.4 we have $\mathcal{G}(2,3) = \{1,2\}$, since $x = 2$ and $x' = 3$ have common points at aggregation levels 1 and 2.

$\mathcal{X}^{(g)}(x)$ Set of all alternatives that share the same aggregated alternative $G^{(g)}(x)$ at the g^{th} aggregation level, with $\mathcal{X}^{(g)}(x) \subseteq \mathcal{X}$. In the example in figure 9.4 we have $\mathcal{X}^1(4) = \{4, 5, 6\}$.

We need to capture the impact of a decision to run experiment x on all the estimates $\mu_{x'}$ for $x' \in \mathcal{X}$. With the hierarchical learning logic, running one experiment x can affect the estimates of $\mu_{x'}$ for all the experiments. This will happen whenever x and x' share an aggregation level. This might include everything if the highest level of aggregation is simply a constant baseline over the entire surface. To help with this process, we break our original weighted estimate

$$\bar{\mu}_x^n = \sum_{g \in \mathcal{G}} w_x^{(g)} \bar{\mu}_x^{(g,n)},$$

into two parts:

$$\bar{\mu}_{x'}^{n+1} = \sum_{g \notin \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \bar{\mu}_{x'}^{(g,n+1)} + \sum_{g \in \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \bar{\mu}_x^{(g,n+1)}.$$

After substitution of (9.17) and rearranging gives us

$$\begin{aligned}\bar{\mu}_{x'}^{n+1} \quad = \quad & \sum_{g \in \mathcal{G}} w_{x'}^{(g,n+1)} \bar{\mu}_{x'}^{(g,n)} + \sum_{g \in \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \frac{\beta_x^W}{\beta_x^{(g,n)} + \beta_x^W}\left(\bar{\mu}_x^n - \bar{\mu}_x^{(g,n)}\right) \\ & + Z \sum_{g \in \mathcal{G}(x',x)} w_{x'}^{(g,n+1)} \tilde{\sigma}_x^{(g,n)}. \end{aligned} \tag{9.19}$$

When we run experiment x and observe $\hat{\mu}^n_x$, we will of course have to update $\bar{\mu}^{(g,n)}_x$ for all aggregation levels g. It is tempting to assume that the weights $w^{(g)}_x$ remain constant, but we found that this did not work, primarily in the early iterations. For example, it is natural to initialize $w^{(G)}_x = 1$ for the highest level of aggregation, but then we have to understand that an experiment may change this, since otherwise the value of information may be zero. For this reason, we have to predict the updating weights from running experiment x. We approximate the weights using

$$\overline{w}^{(g,n)}_{x'}(x) = \frac{\left(\left(\beta^{(g,n)}_{x'} + I^{(g)}_{x',x}\beta^W_{x'}\right)^{-1} + \left(\beta^{(g,n)}_{x'}\right)^2\right)^{-1}}{\sum_{g'\in\mathcal{G}}\left(\left(\beta^{g',n}_{x'} + I^{g'}_{x',x}\beta^W_{x'}\right)^{-1} + \left(\beta^{g',n}_{x'}\right)^2\right)^{-1}}, \tag{9.20}$$

where

$$I^{(g)}_{x',x} = \left\{\begin{array}{cc} 1 & \text{if } g \in \mathcal{G}\left(x',x\right) \\ 0 & \text{otherwise} \end{array}\right. .$$

Combining (9.14) with (9.19) and (9.20), gives us the following formula for the knowledge gradient

$$\nu^{KG}_x\left(S^n\right) = \mathbb{E}\left[\max_{x'\in\mathcal{X}} a^n_{x'}(x) + b^n_{x'}(x)Z|S^n\right] - \max_{x'\in\mathcal{X}} \bar{\mu}^n_{x'}, \tag{9.21}$$

where

$$a^n_{x'}(x) = \sum_{g\in\mathcal{G}} \overline{w}^{(g,n)}_{x'}(x)\bar{\mu}^{(g,n)}_{x'} + \sum_{g\in\mathcal{G}(x',x)} \overline{w}^{(g,n)}_{x'}(x)\frac{\beta^W_x}{\beta^{(g,n)}_x + \beta^W_x}\left(\hat{\mu}^n_x - \bar{\mu}^{(g,n)}_x\right), \tag{9.22}$$

$$b^n_{x'}(x) = \sum_{g\in\mathcal{G}(x',x)} \overline{w}^{(g,n)}_{x'}(x)\tilde{\sigma}^{(g,n)}_x. \tag{9.23}$$

We now use the procedure described in section 4.5 for the knowledge gradient for correlated beliefs, where we have to identify the lines $a^n_{x'} + b^n_{x'}z$ which are dominated by the others. We refer to non-dominated lines by $\tilde{a}^n_{x'} + \tilde{b}^n_{x'}z$ over a reduced set of alternatives that are now numbered from $1, \ldots, \tilde{M}$, giving us

$$\nu^{KG,n}_x = \sum_{i=1,\ldots,\tilde{M}-1}\left(\tilde{b}^n_{i+1}(x) - \tilde{b}^n_i(x)\right) f\left(-\left|\frac{\tilde{a}^n_i(x) - \tilde{a}^n_{i+1}(x)}{\tilde{b}^n_{i+1}(x) - \tilde{b}^n_i(x)}\right|\right), \tag{9.24}$$

where (as we saw before in chapter 4),

$$f(\zeta) = \zeta\Phi(\zeta) + \phi(\zeta),$$

where $\Phi(\zeta)$ is the cumulative normal distribution and $\phi(\zeta)$ is the standard normal density, given respectively by

$$\Phi(\zeta) = \int_{-\infty}^{\zeta} \phi(z)dz,$$

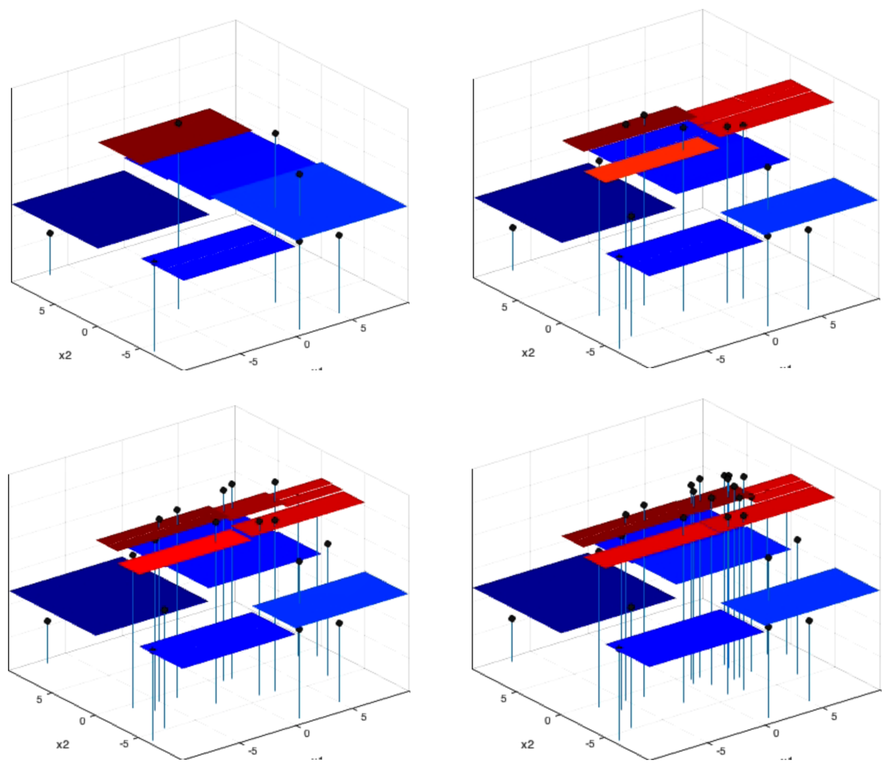

Figure 9.7 Sequence of estimates of a function using the hierarchical knowledge gradient policy, showing initial exploration to a more focused search as it identifies the optimum.

and

$$\phi(\zeta) = \frac{1}{\sqrt{2\pi}} e^{-\frac{\zeta^2}{2}}.$$

Thus, the knowledge gradient policy for hierarchical beliefs is comparable to the KG policy for correlated beliefs.

Figure 9.7 demonstrates the hierarchical knowledge gradient as it learns a smooth function with a single maximum. The first quadrant (upper left) shows initial exploration, steadily narrowing (left to right) to the region of the optimum as it learns to exclude certain regions.

Hierarchical belief models are fairly powerful for strategy for choosing among complex alternatives (such as people, movies and drugs). The speed with which it learns, of course, depends completely on the complexity of the underlying function. For example, if $\mathcal{X}$ is a large discrete set (imagine, for example, the set of all movies that can be watched on the internet, or the set of all possible molecular combinations for a drug), and assume that there is no relationship at all between the value of two different alternatives μ_x and $\mu_{x;}$. If this actually were the situation, no learning policy would perform well without an exceptionally large budget.

Fortunately, most real applications (especially large ones) have a lot of structure. If the belief model captures this structure, then a good learning policy can do quite well. The power of the Bayesian knowledge gradient, of course, is that it assumes that we start the process using some prior knowledge, which we want to exploit.

9.4 WHY DOES IT WORK?

From OUU book - needs rewriting

9.4.1 Computing bias and variance

Before we present our methods for hierarchical aggregation, we are going to need some basic results on bias and variance in statistical estimation. Assume we are trying to estimate a true but unknown parameter μ which we can observe, but we have to deal with both bias δ and noise ε, which we write as

$$\hat{\mu}^n = \mu + \delta + \varepsilon^n. \tag{9.25}$$

Both the mean μ and bias δ are unknown, but we are going to assume that we have some way to make a noisy estimate of the bias that we are going to call $\hat{\delta}^n$. Later we are going to provide examples of how to get estimates of δ.

Now let $\bar{\mu}^n$ be our estimate of μ after n observations. We will use the following recursive formula for $\bar{\mu}^n$

$$\bar{\mu}^n = (1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n.$$

We are interested in estimating the variance of $\bar{\mu}^n$ and its bias δ^n. We start by computing the variance of $\bar{\mu}^n$. We assume that our observations of μ can be represented using equation (9.25), where $\mathbb{E}\varepsilon^n = 0$ and $Var[\varepsilon^n] = \sigma^2$. With this model, we can compute the variance of $\bar{\mu}^n$ using

$$Var[\bar{\mu}^n] = \lambda^n \sigma^2, \tag{9.26}$$

where λ^n can be computed from the simple recursion

$$\lambda^n = \begin{cases} (\alpha_{n-1})^2, & n = 1, \\ (1-\alpha_{n-1})^2\lambda^{n-1} + (\alpha_{n-1})^2, & n > 1. \end{cases} \tag{9.27}$$

To see this, we start with $n = 1$. For a given (deterministic) initial estimate $\bar{\mu}^0$, we first observe that the variance of $\bar{\mu}^1$ is given by

$$\begin{aligned} Var[\bar{\mu}^1] &= Var[(1-\alpha_0)\bar{\mu}^0 + \alpha_0\hat{\mu}^1] \\ &= (\alpha_0)^2 Var[\hat{\mu}^1] \\ &= (\alpha_0)^2\sigma^2. \end{aligned}$$

For $\bar{\mu}^n$ for $n > 1$, we use a proof by induction. Assume that $Var[\bar{\mu}^{n-1}] = \lambda^{n-1}\sigma^2$. Then, since $\bar{\mu}^{n-1}$ and $\hat{\mu}^n$ are independent, we find

$$\begin{aligned}
Var[\bar{\mu}^n] &= Var\left[(1-\alpha_{n-1})\bar{\mu}^{n-1} + \alpha_{n-1}\hat{\mu}^n\right] \\
&= (1-\alpha_{n-1})^2 Var\left[\bar{\mu}^{n-1}\right] + (\alpha_{n-1})^2 Var[\hat{\mu}^n] \\
&= (1-\alpha_{n-1})^2\lambda^{n-1}\sigma^2 + (\alpha_{n-1})^2\sigma^2 \qquad (9.28) \\
&= \lambda^n\sigma^2. \qquad (9.29)
\end{aligned}$$

Equation (9.28) is true by assumption (in our induction proof), while equation (9.29) establishes the recursion in equation (9.27). This gives us the variance, assuming of course that σ^2 is known.

Using our assumption that we have access to a noisy estimate of the bias given by $\hat{\delta}^n$, we can compute the mean-squared error using

$$\mathbb{E}\left[\left(\bar{\mu}^{n-1} - \mu(n)\right)^2\right] = \lambda^{n-1}\sigma^2 + (\hat{\beta}^n)^2. \qquad (9.30)$$

See exercise **??** to prove this. This formula gives the variance around the known mean, $\bar{\mu}^n$. For our purposes, it is also useful to have the variance around the observations $\hat{\mu}^n$. Let

$$\nu^n = \mathbb{E}\left[\left(\bar{\mu}^{n-1} - \hat{\mu}^n\right)^2\right]$$

be the mean squared error (including noise and bias) between the current estimate $\bar{\mu}^{n-1}$ and the observation $\hat{\mu}^n$. It is possible to show that (see exercise **??**)

$$\nu^n = (1+\lambda^{n-1})\sigma^2 + (\beta^n)^2, \qquad (9.31)$$

where λ^n is computed using (9.27).

In practice, we do not know σ^2, and we certainly do not know the bias β. As a result, we have to estimate both parameters from our data. We begin by providing an estimate of the bias using

$$\bar{\delta}^n = (1-\eta_{n-1})\bar{\delta}^{n-1} + \eta_{n-1}\hat{\delta}^n,$$

where η_{n-1} is a (typically simple) stepsize rule used for estimating the bias and variance. As a general rule, we should pick a stepsize for η_{n-1} which produces larger stepsizes than α_{n-1} because we are more interested in tracking the true signal than producing an estimate with a low variance. We have found that a constant stepsize such as .10 works quite well on a wide range of problems, but if precise convergence is needed, it is necessary to use a rule where the stepsize goes to zero such as the harmonic stepsize rule (equation (**??**)).

To estimate the variance, we begin by finding an estimate of the total variation ν^n. Let $\bar{\nu}^n$ be the estimate of the total variance which we might compute using

$$\bar{\nu}^n = (1-\eta_{n-1})\bar{\nu}^{n-1} + \eta_{n-1}(\bar{\mu}^{n-1} - \hat{\mu}^n)^2.$$

Using $\bar{\nu}^n$ as our estimate of the total variance, we can compute an estimate of σ^2 using

$$(\bar{\sigma}^n)^2 = \frac{\bar{\nu}^n - (\bar{\delta}^n)^2}{1+\lambda^{n-1}}.$$

We can use $(\bar{\sigma}^n)^2$ in equation (9.26) to obtain an estimate of the variance of $\bar{\mu}^n$.

If we are doing true averaging (as would occur if we use a stepsize of $1/n$), we can get a more precise estimate of the variance for small samples by using the recursive form of the small sample formula for the variance

$$(\hat{\sigma}^2)^n = \frac{n-2}{n-1}(\hat{\sigma}^2)^{n-1} + \frac{1}{n}(\bar{\mu}^{n-1} - \hat{\mu}^n)^2. \tag{9.32}$$

The quantity $(\hat{\sigma}^2)^n$ is an estimate of the variance of $\hat{\mu}^n$. The variance of our estimate $\bar{\mu}^n$ is computed using

$$(\bar{\sigma}^2)^n = \frac{1}{n}(\hat{\sigma}^2)^n.$$

We are going to draw on these results in two settings, which are both distinguished by how estimates of the bias δ^n are computed:

Hierarchical aggregation We are going to estimate a function at different levels of aggregation. We can assume that the estimate of the function at the most disaggregate level is noisy but unbiased, and then let the difference between the function at some level of aggregation and the function at the most disaggregate level as an estimate of the bias.

Transient functions Later, we are going to use these results to approximate value functions. It is the nature of algorithms for estimating value functions that the underlying process varies over time (see see this most clearly in chapter **??**). In this setting, we are making observations from a truth that is changing over time, which introduces a bias.

9.5 BIBLIOGRAPHIC NOTES

Section **??** - See also Miller (2002) for additional applications of subset selection in statistics, and Horrace et al. (2008) for applications in economics.

Section 9.2.4 - The energy portfolio application is also considered by Ryzhov & Powell (2009*c*) and Ryzhov & Powell (2009*b*). The latter reference also first proposes the Monte Carlo KG policy. The recursive updating equation for the square root matrix can be found e.g. in Kaminski et al. (1971).

PROBLEMS

9.1 There are six primary drugs that can be used to treat diabetes, and these are often prescribed in groups of three at a time. There are 20 ways of choosing three drugs from a set of six (known as a "cocktail"), assuming all possible combinations make sense. Enumerate these 20 cocktails, and create a covariance matrix Σ where entry $\Sigma_{ij} = \sigma^2 N_{ij}$, where N_{ij} is the number of drugs in common between drug cocktail i and drug cocktail j (so, $0 \leq N_{ij} \leq 3$). Let $\sigma^2 = .35^2$ be the variance in our prior distribution of belief about the effect of a drug cocktail. Finally, let $\sigma_\epsilon^2 = .55^2$ be the variance of an observation.

Assume that our prior on the blood sugar level produced by each cocktail is the same and is equal to 5.0.

a) Randomly generate a truth from this prior for each cocktail. Create the covariance matrix and use the knowledge gradient for correlated beliefs to search for the best cocktail using $N = 20$ experiments. You may use the MATLAB code (first introduced in Chapter 4) that can be downloaded from.

   `http://optimallearning.princeton.edu/exercises/KGCorrBeliefs.m`

   An example illustration of the KGCB algorithm is given in

   `http://optimallearning.princeton.edu/exercises/KGCorrBeliefsEx.m`

b) Now repeat (a) for 100 different truths, and summarize how well you discover each truth using 20 observations.

c) Next we are going to use the methods described in this chapter to solve a much larger problem, but we are going to use them on this small test problem. Repeat (a), but this time you are going to randomly sample 5 out of the 20 combinations, and limit your calculation of the KG factor to these three (the easiest way to modify the code is to compute the knowledge gradient for all 20 as you are doing, but then choose a subset of 5, and finally choose the drug cocktail with the best knowledge gradient out of these five). Perform $N = 100$ samples and compare the performance to the results you obtained when you computed the knowledge gradient for all the alternatives.

d) Finally, repeat (d) for all 100 truths and report on the average performance.

9.2 Verify that 9.4 produces a valid square root matrix, that is, $C^{n+1}\left(C^{n+1}\right)^T = \Sigma^{n+1}$.