CHAPTER 7

LINEAR BELIEF MODELS

In the ranking and selection problem and the multiarmed bandit problem, we assumed we had a finite set of alternatives $x \in \mathcal{X} = \{1, 2, \ldots, M\}$, with a belief μ_x about each of the finite alternatives. Known as a lookup table representation, this model is very flexible in that it does not require any assumed structure among the alternatives. However, such models become very clumsy when the number of alternatives is large.

In this chapter, we make the transition from a lookup table belief model to one that uses a parametric model that is linear in the parameters. We assume that an experiment x can be written as the vector $x = (x_1, x_2, \ldots, x_K)$ where x_k may be discrete or continuous. Given x, we observe a value y where we assume a linear relationship of the form

$$f(x|\theta) = \sum_{f \in \mathcal{F}} \theta_f \phi_f(x),$$

where $\mathcal{F}$ is a set of features, x is our input data, and $(\phi_f(x))_{f \in \mathcal{F}}$ are features that we draw from the data in x. For example, x could be a movie, while $\phi_1(x)$ could be the genre, $\phi_2(x)$ could be the year the movie was made, and $\phi_3(x)$ could be the number of minutes of high action in the movie. After deciding to display movie x, we then observe the response y which might be the number of times the movie is viewed by online users when it is displayed on their login page.

Instead of observing a sample observation $\hat{\mu}^n_x$ of the unknown truth μ_x, we are going to observe $\hat{y}^n$ of the function $f(x|\theta)$. As before, if we decide to run experiment

$x = x^n$, we then observe $\hat{y}^{n+1}$, staying with our standard convention of using the iteration counter to tell us which experiments the variable is allowed to see.

We are going to need to manipulate our linear model. For this reason, we are going to adopt the standard notation of statistics and let $x_1, \ldots, x_K$ be the features (instead of $\phi_f(x)$). This means that we are going to equate the experiment "x" with the features of the experiment $(x_1, \ldots, x_K)$.

$$\hat{y}^{n+1} = \theta_0 + \theta_1 x_1^n + \theta_2 x_2^n + \ldots + \theta_K x_K^n + \varepsilon^{n+1}$$

where ε^{n+1} is typically assumed to be a normally distributed error term with mean 0 and variance σ^2.

The most common motivation for using a parametric model is that the number of alternatives M may be extremely large (say, 100,000 or more), or even infinite (if x is continuous). However, there are applications where we know something about the structure of the function that we can capture with a parametric model. For example, we may be trying to estimate the maximum of a concave function. A quadratic approximation may capture this quite nicely, whereas a discretized approximation may easily lose the natural concavity of the function.

In classical regression model applications, we are given a series of observations $x^1, x^2, \ldots, x^n$, where for each set of explanatory variables x^m there is an observation $\hat{y}^{m+1}$. (In traditional batch statistics, we would write $\hat{y}^m$ as the observation corresponding to experiment x^m, but in our sequential setting, it makes more sense to choose x^m and then observe $\hat{y}^{m+1}$.) This data is then used to find a parameter vector θ that minimizes the mean squared error between the predicted model and the actual observations. We often do not have any choice in how the experiments $x^1, x^2, \ldots, x^n$ are chosen.

The problem of determining how to choose a set of experiments is a popular topic in the literature on statistical design of experiments. This literature, however, typically focuses on problems where a design (that is, a sequence of experiments) is chosen before any observations are made (for more on this topic, see Chapter 2). This strategy reflects the nature of the objective function that is used in this work. If the goal is to minimize variance, we can exploit the property that the variance of an estimate is a function of the experiments x^n and not the observations $\hat{y}^n$.

In this chapter, we show how the knowledge gradient (for offline or online learning) can be extended to problems where the belief is captured by a model that is linear in the parameters. This result will allow us to tackle problems with thousands of alternatives, capturing correlations in the beliefs between these alternatives but without having to store a covariance matrix with thousands of rows and columns. The complexity of most of the steps in our adaptation of the knowledge gradient will be determined by the number of parameters rather than the number of alternatives.

7.1 APPLICATIONS

It is useful to start with some real applications to provide context to the discussion that follows.

7.1.1 Maximizing ad clicks

Imagine that you are an Internet company. You might be selling a set of products, or perhaps you are Google selling ad space. Either way, you may have a set of ads (and possibly thousands of ads) that you could post on the Internet, and you want to find the ad that generates the most ad clicks (the number of times that someone clicks on an ad, indicating interest).

How do you choose which ads to post? Let D_i be the data describing the ith ad. D_i might include all the text, along with descriptors of any graphics. The hard part of this project requires identifying important *features*, which are often represented using a device called *basis functions*. We let $\phi_f(D)$ be a particular basis function in a set $f \in \mathcal{F}$. Examples of basis functions might include

$$\begin{aligned}
\phi_1(D) &= \text{The number of words in the ad,} \\
\phi_2(D) &= \text{The number of times the word “iphone,” “itouch” or “ipad” appears,} \\
\phi_3(D) &= \text{1 if the ad involves wireless communication, 0 otherwise,} \\
\phi_4(D) &= \text{1 if the ad involves food, 0 otherwise,} \\
\phi_5(D) &= \text{1 if the ad has color graphics, 0 otherwise,} \\
\phi_6(D) &= \text{1 if the ad claims to save money, 0 otherwise,} \\
\phi_7(D) &= \text{1 if the ad involves travel or vacations, 0 otherwise.}
\end{aligned}$$

We first quickly realize that a “basis function” is the same as an independent variable in our linear regression model (different words for the same thing). Also, we can capture both individual features as well as combinations of features. For example, some people might be attracted to ads that involve food, while others enjoy travel, but what about ads that do both? We can reflect the combined contribution by adding the marginal effect of each, but we can also introduce a basis function that capture whether an ad covers food in exotic locations.

Using the notation of basis functions, we would write our linear model as

$$Y = \theta_0 + \sum_{f \in \mathcal{F}} \theta_f \phi_f(D) + \varepsilon.$$

We can make this a bit more compact by introducing the basis function $\phi_0(D) = 1$. Now let $\phi(D)$ be a column vector of basis functions, and θ be a column vector of the coefficients. We can then write

$$Y = \bar{\theta}^T \phi + \varepsilon.$$

There are numerous variations of this basic problem which involve identifying websites (or documents) that achieve a particular purpose. For example:

- Researching information about a company that might predict a change in the stock price - The observable information might be an actual change in stock price (which can be automated) or it might involve showing the website to a domain expert who can assess its importance. Since the domain expert is a limited resource, we have to be careful in our selection of websites to be evaluated.

- Finding documents that provide information on terrorist activity - Similar to the previous item, we can scan millions of websites, but we have to choose which ones we should show to a domain expert for evaluation. This allows us to build up a dataset that can be used to calibrate the importance of different features.

7.1.2 Dynamic pricing

Now consider the same Internet company from the point of view of pricing. Perhaps we are selling textbooks, DVDs, or music downloads. In any case, the Internet offers us a great deal of freedom in setting prices for our product. We can choose a price, wait for a period of time and observe the resulting revenue, then adjust the price in response. Of course, this is an example of online learning, because our objective is to maximize total revenue.

To keep the modeling simple, we are going to assume that demand is a linear function of price which can be written

$$D(p) = \theta_0 + \theta_1 p,$$

where presumably $\theta_1 < 0$ to create a downward sloping demand function. We do not know θ_0 and θ_1, but we assume that we can charge a price p and then make a noisy observation of the demand $D(p) = \theta_0 + \theta_1 p + \varepsilon$. When we charge a price p, the revenue we earn is given by

$$R(p) = pD(p).$$

The pricing problem also arises outside the e-commerce setting. For example, it could apply to a small business that has a lot of room to change its pricing decisions, but is not prominent enough on the market to be able to change the underlying revenue curve. The business then has to adapt to an existing, unknown true curve. An interesting example of such an enterprise is a recharging station for mobile phones in a developing country. For example, cell phone use in rural Africa is currently experiencing prodigious growth, because it is much easier to build a single cell phone tower in a rural area than to stretch a land line out to every home. Mobile phones can serve as versatile and affordable tools even for many who are not connected to the electric power grid. An entrepreneur with a small power generator, such as a car battery or a solar panel, can serve many users; the question is how the service should be priced.

7.1.3 Housing loans

The Small Business Administration (SBA) plays the role of granting loans to people whose homes have been destroyed by hurricanes and other natural events. As with any bank, the SBA has to identify people who are good loan prospects. By granting a loan, the SBA is able to observe which people eventually replay the loan. This information can then be used in a regression model to improve future predictions of whether someone might default on a loan. Explanatory variables might include

- The term (length) of the loan.
- The credit score of the applicant.
- Number of months employed.
- Income.
- Did the applicant own the home that was damaged or destroyed?
- Amount of the loan relative to the size of the loss from the storm.
- Amount of collateral.

Normally the SBA would grant loans to people whose probability of repayment is over some amount. However, it is possible for the SBA to grant some loans to applicants whose probability of repayment might be lower simply for the information that might improve their predictive ability for loan defaults.

7.1.4 Optimizing dose response

It is often the case that people respond differently to a particular dosage of a medication, forcing physicians to experiment with different dosages. It is natural to try to predict a patient's response (lowering blood sugar, controlling high blood pressure, raising the red blood cell count) using a statistical model. However, sometimes it is necessary to experiment with different dosages to help improve the model (which may involve a term unique to an individual patient).

We might envision a statistical model that predicts response as a function of a patient's body mass index, gender, age, ethnicity and other elements of a patient's history. However, it is often the case that we want to fit a nonlinear function that relates patient response to a nonlinear function which might look like

$$P_i = \frac{e^{U(x_i|\theta)}}{1 + e^{U(x_i|\theta)}},$$

where $U(x|\theta)$ is a linear function of independent variables with the general form

$$U(x|\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots .$$

In this model, P_i gives the proportion of people in a group i with a specific set of attributes $x_i = (x_{i1}, x_{i2}, \dots)$ who respond to a particular dosage. This is an example of a parametric model that is *nonlinear* in the parameter vector θ. Belief models that are nonlinear in the parameters cause problems with methods such as the knowledge gradient because we lose conjugacy (see Chapter **??** for an in depth analysis of this particular belief model). However, we can overcome this problem by introducing the transformation

$$\bar{P}_i = \ln\left(\frac{P_i}{1 - P_i}\right).$$

Using this transformation, we obtain the linear regression

$$\bar{P}_i = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots .$$

Now we have the response relationship expressed as a linear regression. We can use this model given estimates of θ to optimize a dosage strategy. This strategy, however, depends on estimates of the parameter vector θ. We may wish to attempt dosages simply to learn more about this relationship.

7.2 A BRIEF REVIEW OF LINEAR REGRESSION

Assume we have n experiments of a vector $x = (x_1, \dots, x_K)$. If x^m is the mth experiment, using the indexing convention we have used throughout this volume, we let $\hat{y}^{m+1}$ be the observation corresponding to x^m. Recall that our indexing system reflects the property that x^m depends on the observations $\hat{y}^1, \dots, \hat{y}^m$, and we observe $\hat{y}^{m+1}$ *after* we have chosen x^m. Let

$$x^n = \begin{pmatrix} x_1^n \\ x_2^n \\ \vdots \\ x_K^n \end{pmatrix}$$

be a K-dimensional column vector of observations. Often we will let $x_1 = 1$ to represent a constant term. Letting θ be the column vector of parameters, we can write our model as

$$\hat{y} = \bar{\theta}^T x + \varepsilon,$$

where we assume that the errors $(\varepsilon^1, \dots, \varepsilon^n)$ are independent and identically distributed. Since $\bar{\theta}^n$ is our estimate of θ after n observations, our best estimate of y is given by

$$\hat{y}^n = (\bar{\theta}^n)^T x^n.$$

7.2.1 The normal equations

We wish to find a parameter vector θ that solves

$$\min_\theta \sum_{m=0}^{n-1} \left(\hat{y}^{m+1} - \sum_{k=1}^{K} \theta_k x_k^m \right)^2 . \tag{7.1}$$

Let $\bar{\theta}^n$ be the optimal solution for this problem. We can solve this problem very simply. Let X^n be the n by K matrix

$$X^n = \begin{pmatrix} x_1^1 & x_2^1 & & x_K^1 \\ x_1^2 & x_2^2 & & x_K^2 \\ \vdots & \vdots & \cdots & \vdots \\ x_1^n & x_2^n & & x_K^n \end{pmatrix}.$$

The vector of observations of the dependent variable is given by

$$\hat{Y}^n = \begin{pmatrix} \hat{y}^1 \\ \hat{y}^2 \\ \vdots \\ \hat{y}^n \end{pmatrix}.$$

The optimal parameter vector $\bar{\theta}^n$ (after n observations) is then given by

$$\bar{\theta}^n \quad = \quad [(X^n)^T X^n]^{-1}(X^n)^T \hat{Y}^n. \tag{7.2}$$

From the normal equations, we can compute the covariance matrix for $\bar{\theta}^n$ using a simple matrix identity. If u and w are scalar random variables where $u = Aw$, then we know that *Var*$(u) = A^2$*Var*(w). If u and v are vectors, and A is a suitably dimensioned matrix, then we can write *Var*(u) (the covariance matrix for the vector u) as

$$Cov(u) = ACov(w)A^T$$

where $Cov(w)$ is the covariance matrix of w. Recall that for matrices A and B, $AB^T = (BA^T)^T$. Also keep in mind that $[(X^n)^T X^n]^{-1}$ is symmetric. Applying this identity to (12.2), where $A = [(X^n)^T X^n]^{-1}(X^n)^T$, we obtain

$$\begin{aligned} Var(\bar{\theta}^n) &= [(X^n)^T X^n]^{-1}(X^n)^T Cov(\hat{Y}^n) \left([(X^n)^T X^n]^{-1}(X^n)^T\right)^T \\ &= [(X^n)^T X^n]^{-1}(X^n)^T Cov(\hat{Y}^n)(X^n)[(X^n)^T X^n]^{-1}. \end{aligned}$$

Since the elements of $\hat{Y}^n$ are independent, $Cov(\hat{Y}^n) = \sigma_\epsilon^2 I$ where I is the identity matrix and σ_ϵ^2 is the variance of our experimental error . This allows us to write

$$\begin{aligned} \Sigma^{\theta,n} &= [(X^n)^T X^n]^{-1}(X^n)^T X^n [(X^n)^T X^n]^{-1}\sigma_\epsilon^2 \\ &= [(X^n)^T X^n]^{-1}\sigma_\epsilon^2. \end{aligned}$$

It is important to realize that the matrix X^n is n by K, so computing $\Sigma^{\theta,n}$ is not too difficult.

7.2.2 Recursive least squares

There is a shortcut that we can use to do this recursively. The updating equation for $\bar{\theta}^n$ can be computed using

$$\bar{\theta}^n = \bar{\theta}^{n-1} + \frac{1}{\gamma^n} B^{n-1} x^n \varepsilon^n, \tag{7.3}$$

where ε^n is the error given by

$$\varepsilon^n = \hat{y}^n - \bar{\theta}^{n-1} x^{n-1}. \tag{7.4}$$

The matrix $B^n = [(X^n)^T X^n]^{-1}$. This can be updated recursively without computing an explicit inverse using

$$B^n = B^{n-1} - \frac{1}{\gamma^n}(B^{n-1} x^n (x^n)^T B^{n-1}). \tag{7.5}$$

The scalar γ^n is computed using

$$\gamma^n = 1 + (x^n)^T B^{n-1} x^n. \tag{7.6}$$

Note that if we multiply (7.5) through by σ_ϵ^2 we obtain

$$\Sigma^{\theta,n} = \Sigma^{\theta,n-1} - \frac{1}{\gamma^n}(\Sigma^{\theta,n-1} x^n (x^n)^T \Sigma^{\theta,n-1}),$$

where we scale γ^n by σ_ϵ^2, giving us

$$\gamma^n = \sigma_\epsilon^2 + (x^n)^T \Sigma^{\theta,n-1} x^n.$$

Note that we had to multiply each B^{n-1} in the second term on the right of (7.5) by σ_ϵ^2 so we also divided the second term by σ_ϵ^2, which we did by scaling γ^n.

Thus, we have compact updating equations dimensioned only by the number of parameters, rather than the number of alternatives.

The recursive updating formulas, aside from allowing us to avoid an expensive matrix inversion, allows us to handle an issue that we have not yet considered. There are problems where the observations are nonstationary, which means they are coming from a process that is changing over time. In such settings, we may not want to give all the observations equal weight. We can do this by replacing the objective function (12.1) with

$$\min_\theta \sum_{m=0}^{n-1} \lambda^{n-m} \left(\hat{y}^{m+1} - \sum_{k=1}^{K} \theta_k x_k^m \right)^2. \tag{7.7}$$

Here, λ is a discount factor that we use to discount older observations. If we use this objective function, our recursive updating equations change only slightly to

$$\gamma^n = \lambda + (x^n)^T B^{n-1} x^n, \tag{7.8}$$

instead of (7.7), while we replace (7.5) with

$$B^n = \frac{1}{\lambda}\left(B^{n-1} - \frac{1}{\gamma^n}(B^{n-1} x^n (x^n)^T B^{n-1}) \right).$$

Setting $\lambda = 1$ gives us the original updating equations. Using smaller values of λ reduces the weight on older observations, but also increases the variance of the estimates.

7.2.3 A Bayesian interpretation

Classical linear regression is, of course, a frequentist method, but we can put linear regression into our standard Bayesian vocabulary. Let μ^i be the true value of alternative i which, in our Bayesian setting, is a random variable given by

$$\mu^i = \theta x^i.$$

Just as μ is a random variable (our truth), so is θ, which is the truth about the effect of each feature. We learned that we get tremendous benefits from exploiting the covariance between two alternatives. The covariance between μ^i and μ^j is given by

$$\begin{aligned}
Cov(\mu^i, \mu^j) &= Cov\left(\sum_k \theta_k X_k^i, \sum_k \theta_k X_k^j\right) \\
&= \sum_{k,k'} X_k^i X_{k'}^j Cov(\theta_k, \theta_{k'}) = \sum_{k,k'} X_k^i X_{k'}^j \Sigma_{k,k'}^\theta \\
&= e_i^T X \Sigma^\theta X^T e_j,
\end{aligned}$$

where $\Sigma_{k,k'}^\theta = Cov(\theta_k, \theta_{k'})$ is the covariance between the regression coefficients θ_k and $\theta_{k'}$. If Σ is our original covariance matrix between our beliefs of different alternatives, Σ and Σ^θ are related by

$$\Sigma = X \Sigma^\theta X^T.$$

This means that we can write our vector of truths about the value of each alternative as

$$\mu \sim N(\bar{\theta}^T X, X \Sigma^\theta X^T).$$

Here, X is a matrix with a row and column for *each* alternative, which means that it has M rows (one for every alternative which may be an extremely large number) and K columns (one for every feature, which is generally not too large). We then let

$$\Sigma^n = X \Sigma^{\theta,n} X^T$$

be the covariance matrix among all M alternatives (hence Σ^n is $M \times M$, while $\Sigma^{\theta,n}$ is $K \times K$). This means that we are getting an estimate of the covariance between every pair of alternatives, including those which we have not tested yet, from the much more compact matrix Σ^θ. Our goal is to avoid having to explicitly compute and store the complete matrix Σ^n.

The important result is that we can compute a covariance matrix among all *possible* alternatives, using only the covariance matrix $\Sigma^{\theta,n}$ among the parameters, and the matrix X of attributes of each possible alternative. Updating $\Sigma^{\theta,n}$ from data is relatively easy using our recursive formulas. Computing Σ^n, given the potentially large number of rows, is still quite manageable given that it involves fairly simple calculations.

7.2.4 Generating a prior

To perform learning in our Bayesian setting, we have to generate a truth, and then apply a learning algorithm (such as the knowledge gradient) to try to discover the truth. When we had a discrete set of alternatives with normally distributed beliefs, we could simply sample from the normal distribution we used as a prior.

There are several ways to generate a prior for these more complex settings. One, of course, is to collect a small sample of observations from an initial training set

$x^0, \ldots, x^{L-1}$, producing observations $\hat{y}^1, \ldots, \hat{y}^L$. If L is larger than the number of parameters to be estimated, we can create an initial estimate of the parameter vector $\bar{\theta}^0$ using the normal equations in equation (12.2). We can then use this initial estimate to obtain an estimate of the variance σ^2 (if this is not known already). In the context of Bayesian learning, this becomes the same empirical Bayes strategy that we have already discussed.

Using an estimate of σ_ϵ^2 obtained from this model, we can estimate an initial covariance matrix for θ from

$$\Sigma^{\theta,0} = [(X^L)^T X^L]^{-1} \sigma_\epsilon^2 \tag{7.9}$$

where X^L is our initial set of observations from the first L observations.

We cannot use equation (7.9) if we have fewer than L observations, and we need at least $L+1$ observations if we are going to obtain a valid estimate of σ_ϵ^2. However, we can still get initial estimates of θ and Σ^θ with fewer than L observations, as long as we have some sort of prior. If we have some starting point, we can use the recursive equations (7.3)-(7.7) for an initial set of training points which may be chosen at random.

A useful strategy for generating an initial set of estimates is to write the regression equation in the form

$$y = \theta_0 + \theta_1(x_1 - \bar{x}_1) + \theta_2(x_2 - \bar{x}_2) + \ldots + \varepsilon,$$

where $\bar{x}_i$ is viewed as an average or typical value of the independent variable x_i. In this model, θ_0 should be an average (or typical) value of the observations y, while θ_i captures the marginal impact of x_i on y. For example, consider our pricing model above where the demand $D(p)$ is given by

$$D(p) \approx \theta_0 + \theta_1(p - \bar{p}).$$

Here, we have written the independent variable as the deviation from what we might call a typical price give by $\bar{p}$. In this model, θ_0 would be a prior estimate of the typical demand, while θ_1 captures the effect of changes in the price on demand.

Estimating $\Sigma^{\theta,0}$ is somewhat tricker. An estimate of the diagonal terms can be found by trying to estimate a confidence interval for each coefficient θ_i. So, we might think the right coefficient for our demand curve is $\theta_1 \approx -0.07$, and we might then be willing to say that the right coefficient is probably in the range (-0.03,-0.10). From this, we could infer a standard deviation of, say, 0.015, and use this to create the diagonal elements of $\Sigma^{\theta,0}$.

Estimating the off-diagonal elements of $\Sigma^{\theta,0}$ is inherently more complicated. A first order approximation is to just set them to zero, but this can be dangerous. Consider the set of potential demand functions shown in Figure 7.1. We may be uncertain about the precise line that describes the demand-price tradeoff, but we may feel that they all go through a certain region that might reflect the current demand and current price. The implication is that smaller values of θ_1 correspond to smaller values of θ_0, so our prior on these coefficients would not be independent.

Once we have an initial estimate of a mean vector $\bar{\theta}^0$ and covariance matrix Σ^0, we can generate a truth μ from a normal distribution using the techniques described in Section 2.7.

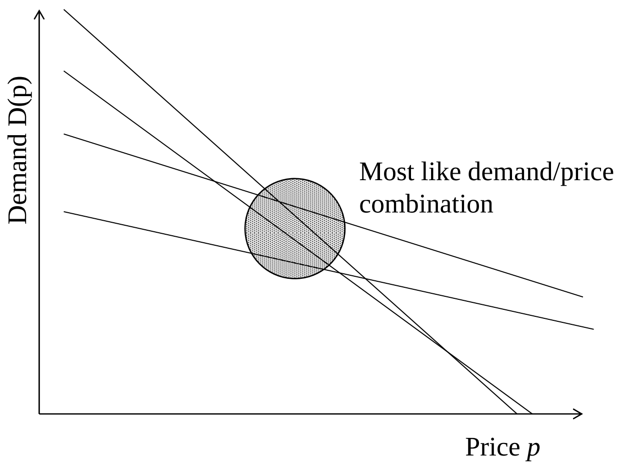


Figure 7.1 Illustration of range of potential demand functions.

7.3 THE KNOWLEDGE GRADIENT FOR A LINEAR MODEL

Now we are ready to derive the knowledge gradient using our linear belief model. We begin with the calculations, which are very similar to the knowledge gradient for correlated beliefs, except that now the correlations are between the parameters. We then contrast the behavior of the knowledge gradient when using a linear belief model versus when we use a lookup table model.

7.3.1 Calculations

Recall that the calculation of the knowledge gradient requires computing the function $h(a, b(j))$ where a is a vector with element $a_i = \bar{\theta}^n_i$ giving the estimate of the value of the ith alternative, and $b(j)$ is a vector with element $b_i(j) = \tilde{\sigma}^n_i(j)$, which is the conditional variance of the change in $\bar{\theta}^{n+1}$ from measuring alternative j. The function $h(a, b)$ is given by

$$h(a, b(j)) = \sum_{i=1}^{M-1} (b_{i+1}(j) - b_i(j)) f(-|c_i(j)|) \tag{7.10}$$

where

$$c_i(j) = \frac{a_i - a_{i+1}}{b_{i+1}(j) - b_i(j)}.$$

Refer to Section 4.5 for the details of computing the knowledge gradient for correlated beliefs.

Keeping in mind that M may be a very large number, we have to use care in how this is computed. We can use $\bar{\theta}^n_i = \bar{\theta}^n x^i$ to compute the expected value, which does not depend on the alternative j that we are measuring.

The harder part is computing $\tilde{\sigma}^n(j)$, which is a vector giving the change in the variance of each alternative i assuming that we have chosen to measure alternative j. Recall that

$$\tilde{\Sigma}^n(j) = \frac{\Sigma^n e_j (e_j)^T}{\sqrt{\Sigma^n_{jj} + \sigma^2_\epsilon}},$$

which gives us the change in the covariance matrix from measuring an alternative j. The matrix $\tilde{\Sigma}^n(j)$ is $M \times M$, which is quite large, but we only need the jth row, which we compute using

$$\tilde{\sigma}^n(j) = \frac{\Sigma^n e_j}{\sqrt{\Sigma^n_{jj} + \sigma^2_\epsilon}}. \tag{7.11}$$

Since $\Sigma^n = X\Sigma^{\bar{\theta}^n}(X)^T$, let X_j be the jth row of X. Then

$$\tilde{\sigma}^n(j) = X_j \Sigma^{\bar{\theta}^n} X^T.$$

We still have to multiply the $K \times K$-dimensional matrix $\Sigma^{\bar{\theta}^n}$ times the $K \times M$-dimensional matrix X^T, after which we have to compute equation (7.10) to find the knowledge gradient for each alternative. Even for problems with tens of thousands of alternatives, this can be executed in a few seconds, since K is much smaller than M.

The real value of the linear belief model is that it allows us to compute the knowledge gradient for much larger sets of alternatives, while still capturing the relationship between our beliefs at different values of x. If we were using our lookup table model with correlated beliefs, it means that we have to manipulate an $M \times M$ matrix, which becomes quite clumsy when M is more than 1,000. With our linear belief model, we only have to manage the parameter covariance matrix Σ^θ, which is much smaller.

In the next section, we take a look into how the parametric model changes the behavior of the learning process.

7.3.2 Behavior

The section above shows how to calculate the knowledge gradient, but does not address the question of how it changes its behavior. Consider, for example, the behavior of a smooth function estimated using a fine-grained lookup table representation as depicted in figure 7.2. The figure to the left shows both the true and estimated function after three experiments. The figure on the right shows the knowledge gradient, which is lowest at each of the three points where experiments have been run. These drops are typical since an experiment reduces the uncertainty about the function at that point. The knowledge gradient can actually rise if the function is sufficiently large.

Now consider what happens when we use a parametric model. Assume that the x is a scalar (such as the price of a product or dosage of a drug) and that the true response is quadratic in x, giving us the belief model

$$f(x|\theta) = \theta_0 + \theta_1 x + \theta_2 x^2. \tag{7.12}$$

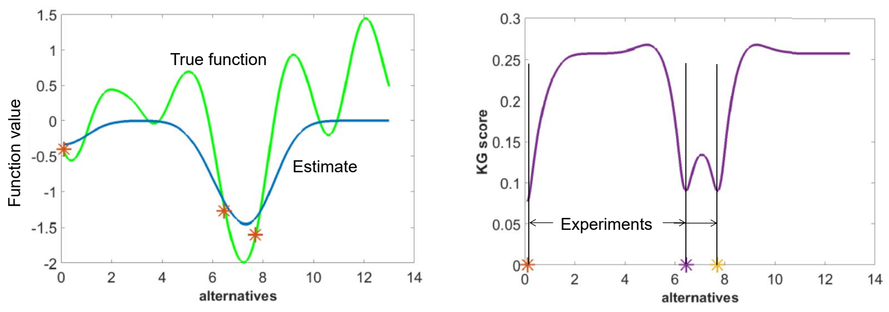


Figure 7.2 Lookup table representation of true function and current estimate after three experiments (left), and the knowledge gradient after three experiments (right) (from **?**).

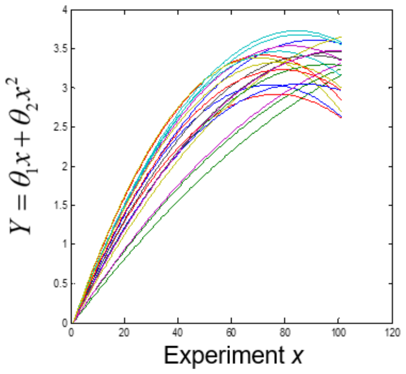


Figure 7.3 A family of possible values quadratic functions assuming $\theta_0 = 0$. (from **?**).

To demonstrate a point, we are going to assume that $\theta_0 =$, but that θ_1 and θ_2 are random (specifically, multivariate normal). Figure 7.4 depicts a sample of these functions, all going through the origin. Our problem is to figure out which experiments to run to learn our function as quickly as possible.

Figure 7.4 shows a plot of the knowledge gradient computed using the equations given above, after 0, 1, 2, and 3 experiments. Initially, the knowledge gradient increases left to right, encouraging the largest value of x so that we experiment with the function where there is the greatest variability. After that, the knowledge gradient becomes bimodal, with the maximum switching between a point at around 1/3 of the maximum, and the maximum. This basic shape and pattern does not change even after a large number of experiments.

The behavior of the knowledge gradient shown in figure 7.4 is quite different than what we saw with a lookup table. It also no longer has the behavior that the knowledge gradient shrinks wherever we run an experiment. In short, working with a parametric belief model produces a completely different behavior.

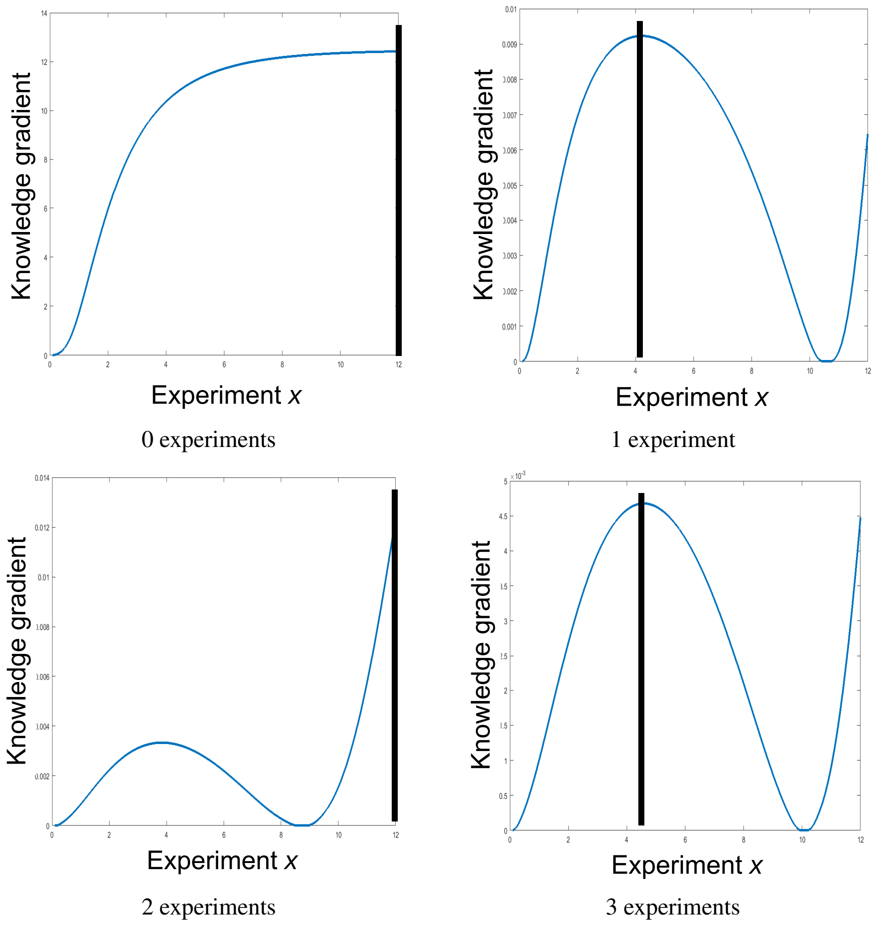
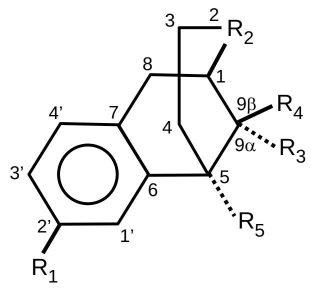


Figure 7.4 The knowledge gradient for the quadratic belief model given in figure **??** after 1, 2, 3 and 4 experiments (top left to bottom right) (from **?**).

7.4 APPLICATION TO DRUG DISCOVERY

A nice area of application of this logic is the design of molecules to accomplish some purpose, such as storing energy or curing cancer. We start with a base molecule such as that shown in Figure 7.5, where there are five locations $(R_1, R_2, \ldots, R_5)$. At these locations, we can add small molecular components called *substituents*, such as those listed to the right of the molecule in the figure. While we may not be able to put every substituent at every location (a chemist would understand which combinations make sense), the number of permutations and combinations can be quite large. For example, this base molecule and associated set of substituents produced 87,000 possible combinations.

The challenge with testing these molecules is that they are fairly difficult to create. It is necessary to design actual chemical processes to produce these compounds. A knowledgeable chemist can create a number of potential molecular compounds on a



Potential substituents:

F	OCH_3
OH	CH
CH_3	NO
$OCOCH_3$	CI
OCOCH	OCOCH

Figure 7.5 Illustration of base molecule with five substituent locations (from Katz & Ionescu 1977).

white board, but once we decide which compound to test next, the actual testing is a project that can take a day to several days. This is the reason why it is important to sequence the experiments carefully.

The problem of designing molecules arises in many settings. The work in this section was motivated by a problem of finding compounds to cure a form of cancer, but the numerical work was done using data taken from the literature which was motivated by a variety of applications. All we require is that we are trying to choose a compound that maximizes (or perhaps minimizes) some metric, such as killing cancer cells.

In practice, the knowledge gradient can be used to produce a ranked list of potential molecules to be tested next. An attraction of this strategy is that it allows a scientist to use expert knowledge to capture dimensions other than the performance of a molecule in terms of a specific metric such as killing cancer cells. For example, a molecule might be toxic, hard to manufacture or insoluble in water.

To put this problem in the context of our model with linear beliefs, let $i = 1, \ldots, I$ be the set of sites (five in our example) and let $j = 1, \ldots, J$ be the set of potential substituents (10 in our example). Let

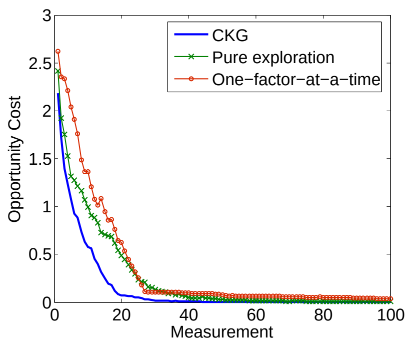
$$X_{ij} = \begin{cases} 1 & \text{If we put the } j\text{th substituent at the } i\text{th site,} \\ 0 & \text{Otherwise.} \end{cases}$$

Now let Y be an observation of the performance of a particular molecule, which might be the percentage of cancer cells that are killed. We might model the performance of a molecule using

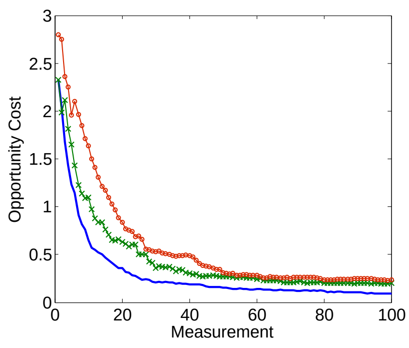
$$Y = \theta_0 + \sum_{i=1}^{I} \sum_{j=1}^{J} \theta_{ij} X_{ij}. \tag{7.13}$$

With this simple model, if we have five sites and 10 substituents, we have 51 parameters to estimate (including the constant intercept θ_0).

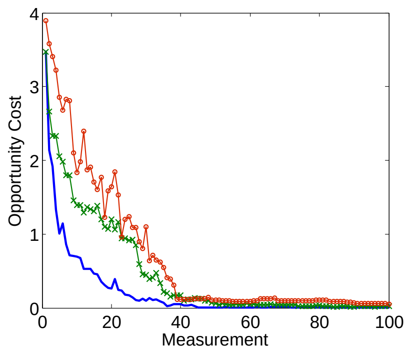
To determine the effectiveness of the knowledge gradient, it is necessary to create a truth (following our standard practice) and then try to discover the truth using different



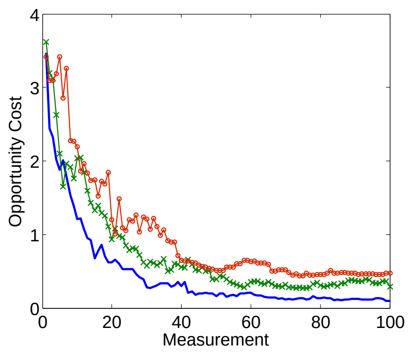
7.6a: Average opportunity cost over 100 runs using a data set of 2640 compounds and a noise standard deviation of 0.1.



7.6b: Average opportunity cost over 100 runs using a data set of 2640 compounds and a noise standard deviation of 0.5.



7.6c: Average opportunity cost over 10 runs using a data set of 87120 compounds and a noise standard deviation of 0.1.



7.6d: Average opportunity cost over 10 runs using a data set of 87120 compounds and a noise standard deviation of 0.5.

Figure 7.6 Comparison of the knowledge gradient to pure exploration and a standard experimental design on a medium and large molecule with different levels of noise (from Negoescu et al. 2010).

methods. The truth was created by using data from an initial set of experiments that was then used to perform an initial fit of the regression model in equation (7.13). This model was then assumed to give us an initial parameter vector $\bar{\theta}^0$ that we used as our prior, and an initial covariance matrix $\Sigma^{\theta,0}$. We made our standard assumption that the true parameter vector $\theta \sim N(\bar{\theta}^0, \Sigma^{\theta,0})$ follows a normal distribution. We would generate a sample realization of a truth $\theta(\omega)$ by sampling from this distribution.

Once we have a truth (which means we now have a "true" parameter vector θ), we run our experiments by choosing a molecule to test (using some policy), and then sampling its performance using equation (7.13). It is important to recognize that this method of testing a learning policy assumes that the linear *model* is accurate, even if we do not know the true parameter vector. But, there is nothing stopping us from generating observations from some other model, and then using linear regression as an approximation.

The knowledge gradient was compared to a pure exploration policy (choosing molecules at random) and a simple experimental design policy which tests one factor (substituent) at a time, cycling through all the substituents. Each of these strategies are used in an off-line setting (since this is laboratory experimentation) to collect data

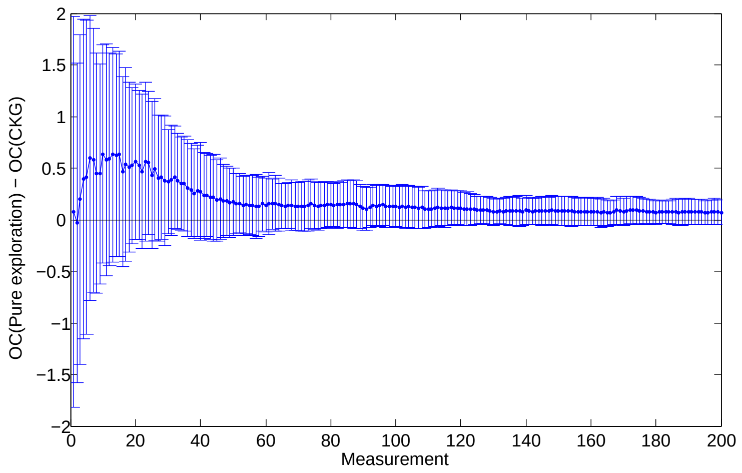


Figure 7.7 Mean and standard deviation of difference in opportunity cost between pure exploration and KGCB using 75 sample paths of 10000 compounds each and a noise standard deviation of 0.38 (from Negoescu et al. 2010).

to fit our regression model. We then use this regression model to decide which of a much larger set of molecules is best.

Figure 7.6 shows the expected opportunity cost for the knowledge gradient, pure exploration and the one-at-a-time factor design on molecules with 2,640 and 87,120 variations, and with experimental noise of 0.1 (lower than that observed from actual data) and 0.5 (higher than that observed from actual data). These experiments suggest that the knowledge gradient consistently outperforms the competition, with a fairly significant improvement if we wanted to stop at around 20 experiments. Note that we were able to find a near-optimal molecule within approximately 30 experiments.

Care has to be used when evaluating the performance of a learning policy based on averages. Figure 7.7 shows the actual spread in the results from 75 sample paths, comparing knowledge gradient against pure exploration (positive means that KG is outperforming pure exploration). After 20 experiments, the knowledge gradient policy is almost always outperforming pure exploration. The difference is much more mixed during the first 20 observations, which should be expected. When there is a lot of uncertainty and not enough information to make sensible choices, random exploration can work just as well as the knowledge gradient on specific sample paths. There is no evidence that the knowledge gradient ever underperforms pure exploration; but it is certainly possible that pure exploration can outperform knowledge gradient in the early iterations. Interestingly, the knowledge gradient significantly outperforms pure exploration *on average* in the early iterations, but the spread is quite high.

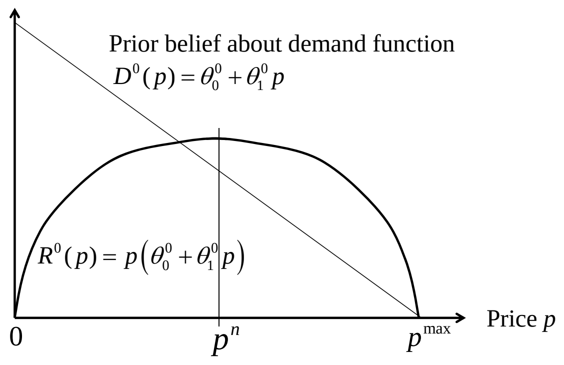


Figure 7.8 Prior demand function and revenue curve.

7.5 APPLICATION TO DYNAMIC PRICING

Let us briefly return to the dynamic pricing application in Section 7.1.2. We are interested in the pricing decision of the African entrepreneur running a cell phone recharging station. This is a small but profitable operation. For example, the entrepreneur can purchase a car battery, charge it in town, and then travel to a village to charge phones for a fee that can be anywhere between $\$0.10$ and $\$0.50$. An alternative with a higher startup cost may be to purchase a small solar system (e.g. a set of solar panels) and then sell the generated power. With a good pricing strategy, the recharging station may see 50–70 customers every day. Assuming that the entrepreneur works five days a week, the resulting revenue can be as high as \$50–\$100 per week.

This problem provides the basic parameters for our example. Every week, the price per use of the recharging station is chosen from the range $[0.10, 0.50]$. The weekly revenue $R\,(p)$ can be as high as $\$50$. In this example, we use a simple quadratic belief model,

$$R\,(p) = p\big(\theta_0 + \theta_1 p\big).$$

where we capture the fact that $R(0) = 0$. Our belief about the revenue for the nth week can be written as $R^n\,(p) = p(\bar{\theta}_0^n + \bar{\theta}_1^n p)$. We would typically start with a prior where $\bar{\theta}_1^0 < 0$, because revenue is likely to be concave (we do not expect it to keep increasing with price forever).

The pricing problem is inherently online: every time we set a price, we collect a revenue. As always, we need to balance the need to maximize revenue with the need to fit a good curve. In this case, we would use the knowledge gradient for online learning, given by

$$X^{KG,n}\,(s^n) = \arg\max_p R^n(p) + (N-n)\,\nu_p^{KG,n},$$

where ν_p^{KG} is the offline knowledge gradient corresponding to the value of observing demand at price p.

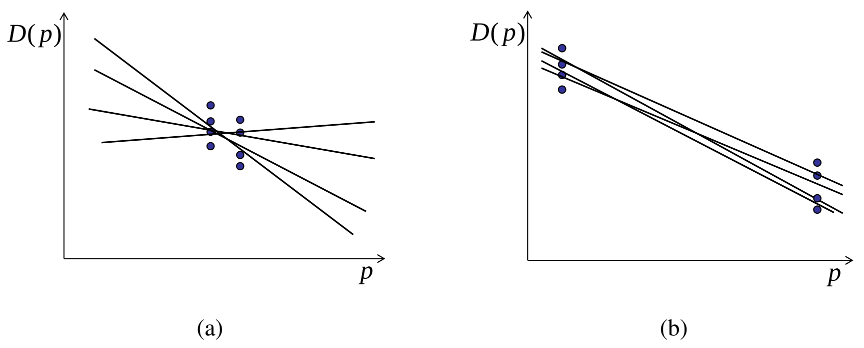


Figure 7.9 Estimating the demand function using (a) observations near the middle and (b) observations near the endpoints.

Let's consider how different learning policies should behave. Figure 7.8 depicts a possible prior on the demand function and resulting revenue curve. Let p^{max} be the price that drives the demand to 0, which means that we should limit our search to prices in the interval $[0, p^{max}]$. Also let p^n be the price that maximizes revenue given our current belief after n experiments, which is to say

$$p^n = \arg\max R^n(p).$$

If we were to stop after n experiments, p^n is the price that we would charge. Otherwise, we would characterize the decision to charge p^n as a pure exploitation policy.

What would we do if we wanted to focus on estimating the demand function? In Figure 7.9(a), we see that if we focus our energies on observations near the middle, we may obtain a wide range of possible functions as a result of our experimental noise. By contrast, if we focus our observations near the endpoints as in 7.9(b), we obtain much more reliable estimates of the demand function. This behavior is well known in the statistics community.

Measuring near the middle offers the potential for maximizing revenue, but we end up learning almost nothing from the observations. By contrast, if we measure near the endpoints, we learn a lot but earn almost nothing. Ideally, we would like to make observations that strike a balance between these two extremes, a property that might be called "sampling the shoulders." This, in fact, is exactly what the knowledge gradient does, in a way that adapts to the number of observations remaining in our budget.

Figure 7.10 illustrates the behavior of a pure exploitation policy (first row), the knowledge gradient policy for offline learning (second row) and the knowledge gradient policy for online learning (third row). These policies were tested for datasets with a moderate level of observation noise (the left column) and a relatively high level of observation noise (right column). The line with solid dots is the initial prior, while the line with open circles is the true demand. The remaining lines represent the estimates of the demand functions for each of the first 20 observations obtained under each policy. The darker lines represent the early iterations, while the lighter lines (which are more clustered) represent the later iterations.

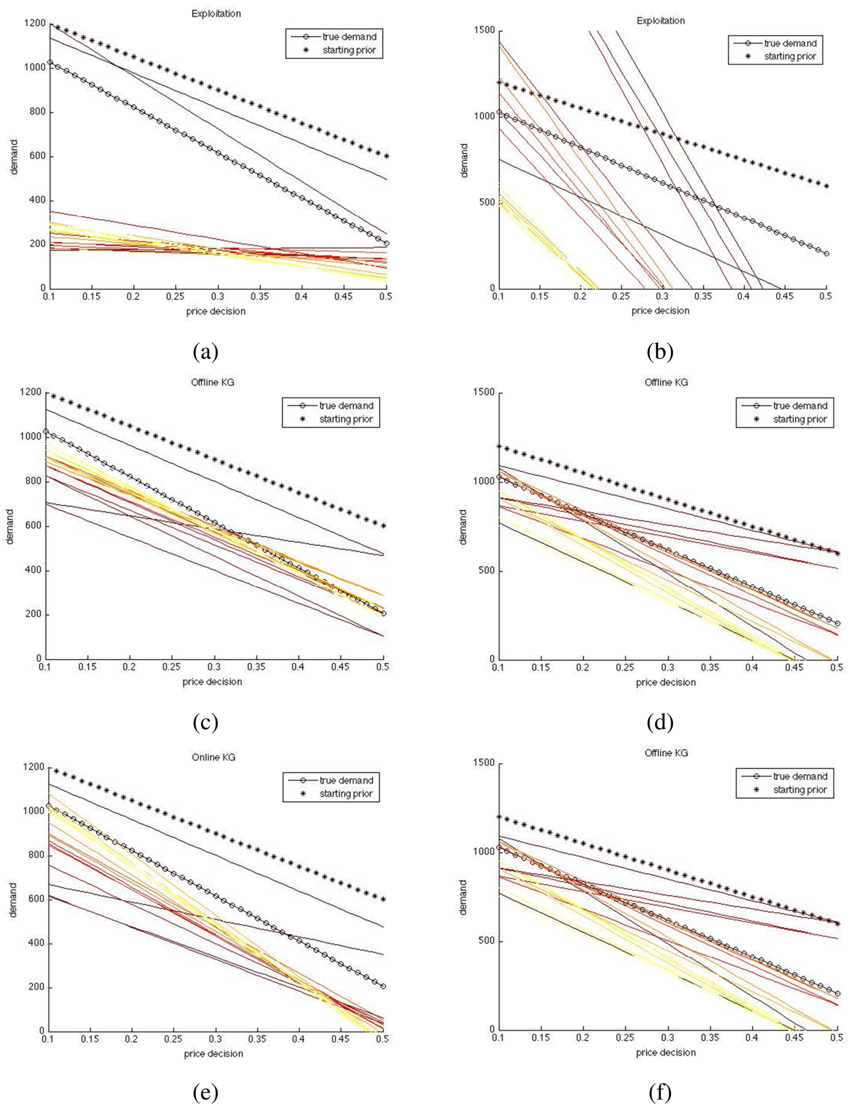


Figure 7.10 The behavior of pure exploitation (first row), offline learning (second row) and online learning (third row) for observations with a moderate level of noise (left column) and a high level of noise (right column).

The results show that using a pure exploitation policy does a terrible job identifying the correct demand curve. By contrast, using an offline policy produces the best results, producing an estimate of the demand curve that is quite close to the truth. The online learning policy strikes a balance between the two, producing an estimate of the demand curve that is clearly not as good as what is obtained using an offline learning policy, but much better than the pure exploitation policy.

The first row of Figure 7.11 shows the actual pricing decisions produced by each policy. The offline knowledge gradient behaves exactly as we would expect given

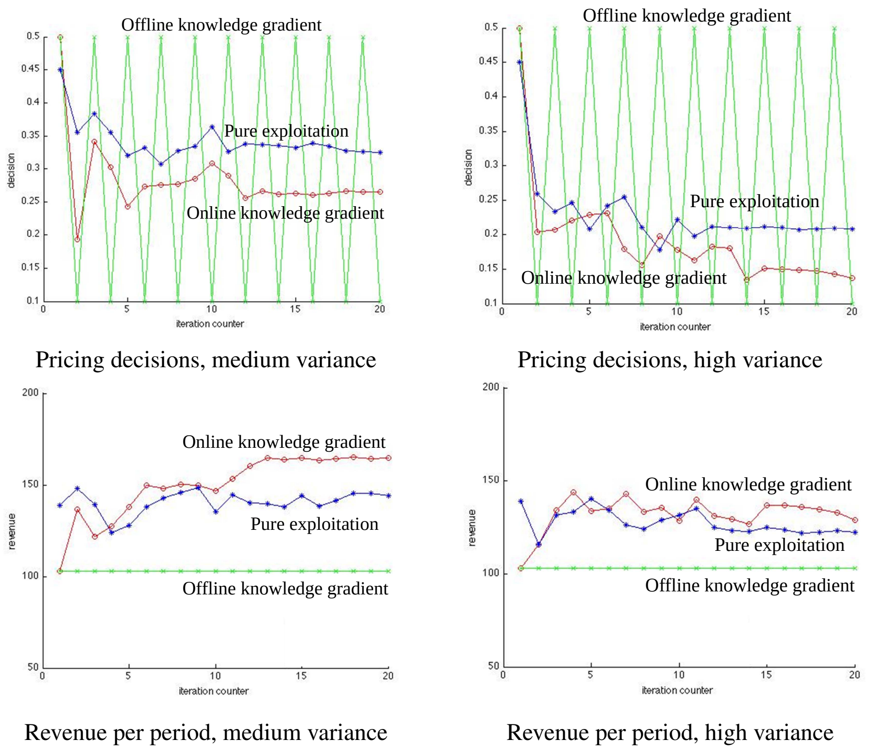


Figure 7.11 The pricing decisions selected by each policy (first row), and revenue earned per period (second row) for moderate and high variance in the experimental noise.

the illustration in Figure 7.9(b). The policy alternates between the two endpoints. This behavior is consistent for both the medium and high variance cases. For the medium variance case, the online knowledge gradient does more exploration in the early iterations before settling in to a single point compared to the pure exploitation policy. Note that the online knowledge gradient approaches a pure exploitation policy toward the later iterations. The only reason that it is measuring different points is that it is converging on a different estimate of the demand curve than pure exploitation, as a result of the exploration in the early iterations.

The second row of Figure 7.10 shows the revenue earned per period. Note that the offline knowledge gradient earns a fixed revenue greater than zero because the endpoints were defined slightly interior to the interval $(0, p^{max})$. The more interesting comparison is between the pure exploitation policy and the online knowledge gradient. For both the medium and high variance cases, the pure exploitation policy produces higher revenues initially, but eventually loses out to the online knowledge gradient policy. Note that the relative improvement of the online knowledge gradient over pure exploitation is largest for the moderate noise case. Again, we think this is to be expected. As the experimental noise increases, it is harder for any policy to learn the correct function. By contrast, we would expect the difference between the two policies to diminish as the experimental noise decreases, because the value of information from learning will be minimal, which again pushes the online knowledge gradient policy toward the pure exploitation policy.

7.6 BIBLIOGRAPHIC NOTES

Section 7.2 - This is classic material that can be found in many statistical textbooks such as Hastie et al. (2005), which can be downloaded from

`https://web.stanford.edu/~hastie/Papers/ESLII.pdf`

Section 7.3 - 7.4 - This material is based on Negoescu et al. (2011). The drug discovery example is taken from Katz & Ionescu (1977).

Section 7.5 - The experimental results presented here come from work by two undergraduate students at Princeton University. The numerical work was performed by Xiaoyang Long as part of her research for the Program in Applied and Computational Mathematics. The application of pricing cell phone charges was studied by Megan Wong as part of her senior thesis.

PROBLEMS

The exercises below require the knowledge gradient algorithm where the belief model is linear in the parameters. This can be downloaded from

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

An example implementation of the algorithm is given in

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

7.1 You are going to replicate the experiments in Section 7.5 where we try to learn the demand as a function of price, while simultaneously trying to maximize revenue. We start by assuming that the demand as a function of price is given by

$$D(p) = \theta_0 + \theta_1 p.$$

Our revenue is given by $R(p) = pD(p)$, and prices are assumed to range between .1 and .5. Assume the true value of $\theta = (\theta_0, \theta_1) = (1233, -2055)$. Normally we would sample this truth from a normal distribution, but we are going to assume a single truth to illustrate how well we discover this truth.

Now assume we start with a high prior $\bar{\theta}^0 = (1350, -1500)$, with a starting covariance matrix of

$$\Sigma^{\theta,0} = \begin{pmatrix} 62500 & 0 \\ 0 & 2{,}500{,}000 \end{pmatrix}$$

a) Set up and run the knowledge gradient with a linear belief model for $N = 20$ iterations assuming a low experimental noise of $\sigma_\epsilon^2 = 62,500$, limiting your search of prices to the range $[.1, .5]$. Do this for both online and offline learning (recall that Section 5.4 describes how to compute the online knowledge gradient from the offline version). Plot the prices chosen and the updated estimate of the demand curve after each iteration.

b) Now sample prices using a pure exploitation policy which maximizes the revenue at each iteration, and compare your results to those obtained using the online and offline knowledge gradient.

c) Repeat (a) and (b) using $\sigma_\epsilon^2 = 625,000$.

d) Repeat (a) and (b) using $\sigma_\epsilon^2 = 6,250,000$.

e) Compare the performance of the three algorithmic strategies under the different noise levels.

7.2 Repeat exercise 7.1, but this time start with a low prior of $\bar{\theta}^0 = (1350, -4500)$.

7.3 Repeat exercise 7.1, but now assume that the truth is normally distributed around the prior with mean $\bar{\theta}^0 = (\bar{\theta}_0^0, \bar{\theta}_1^0) = (1233, -2055)$ and variance

$$\Sigma^{\theta,0} = \begin{pmatrix} 62500 & 0 \\ 0 & 2{,}500{,}000 \end{pmatrix}$$