Optimal Learning

Optimization in the information age

Warren B. Powell
Department of Operations Research and Financial Engineering
Princeton University

Consider a simple optimization problem. You would like to choose the best features for a laptop to maximize total sales. Based on extensive market research, you have identified five configurations which produce weekly sales (in millions) of $4.5, $4.8, $5.0, $5.3, and $5.2. Which configuration should you use? Hmm. Looks like configuration 4 is the winner. Not too hard (and the astute reader will recognize that this is an integer programming problem!). Of course, we realize that there are much more complex deterministic integer programs that are genuinely difficult.

The reality is that the actual sales are random. Assume now that the sales you will achieve for each configuration has a mean given by the numbers above, with standard deviations of $1.3 (million), $1.6, $1.8, $0.9 and $1.3 (the beliefs are depicted in figure 1). Now which would you do? Since we really care about averages over long periods of time, you would again choose the fourth configuration because it is expected to yield the highest average sales. We have just solved a very simple stochastic optimization problem.

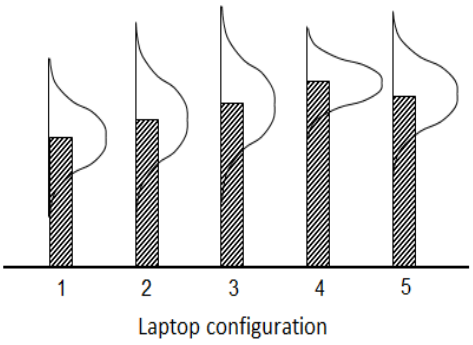

Figure 1-Beliefs about sales from each laptop configuration.

Yet, even this view of the problem is not realistic. In actuality, not only are sales random, we are not even sure of the distribution. If we choose configuration 4 and observe average sales of $4.7 million for a few weeks, we would not treat this as random observations from our distribution. We would probably adjust our distribution downward, since we view our original distribution as just an estimate, while $4.7 is an observation from the true distribution. Given that we are uncertain about our uncertainty, choosing the fourth configuration may not, in fact, be the best because it ignores the value of the information we gain from future sales figures. Note that we have a lot more confidence about the sales from configuration 4 than configuration 3. If we try selling configuration 3, we may learn that the true value for average sales is actually higher than the sales for configuration 4.

This is a simple example of a problem where we have to consider the value of information. Such problems arise in a wide range of settings in business (what is the best price for a product), engineering (what is the best material for a device), science (what is the best molecular compound for curing cancer), politics (where should we run polls for a candidate), policy (what

tax should we put on fertilizer to control runoff into lakes), medicine (what is the best treatment for a disease in a particular patient), the internet (which movies should Netflix show when you log in to maximize rentals), and transportation (how do we find the best path over a network). In each case, we are making decisions that yield information that may influence future decisions.

Given the breadth of applications, it is useful to consider a few major problem classes. It is important to first distinguish problems based on whether they use online learning (learning in the field, as occurs when we observe sales) or offline learning (learning in the laboratory, where we do not care about bad outcomes as long as they lead to a good design). Offline learning problems are typically described under names such as ranking and selection (where the choices are discrete), stochastic search (where we are often optimizing a continuous function), and simulation-optimization.

Online learning problems are perhaps best known as the *multiarmed bandit problem*. This name derives from the story of trying to find the best slot machine (sometimes referred to as "one-armed bandits") by trying out each slot machine. We may try a slot machine that appears to be giving lower payouts in the hope that the actual payout rate is higher than we think. But we run the risk that we were right in the first place, which means we have to live with the potentially lower payout. This is the tradeoff we make when we have to find the best price of a product by varying the price and observing sales. We might learn that a higher price returns higher revenue, but we may learn that the market is unwilling to buy at the higher price, which means that we have to incur lower sales as the price of this information.

The multiarmed bandit problem can be formulated as a multidimensional dynamic program with continuous states, but there is no known algorithm for solving this problem. In the 1970's, J.C. Gittins introduced what are now known as "Gittins indices" where you compute a statistic for each alternative ("bandit") and choose the bandit with the highest value. This was viewed as a breakthrough, except for the fact that computing these indices is also computationally very difficult. A separate community in computer science discovered a class of policies known as "upper confidence bounds" which enjoys a kind of optimality property in the form of a bound on how often you try suboptimal alternatives. The simulation community encounters this problem when testing designs using Monte Carlo simulation, and they have developed policies suited to this setting, often referred to as "optimal computing budget allocation" rules.

It is possible to approach the learning problem using classical and familiar ideas from optimization. The operations research community is very familiar with the use of gradients to minimize or maximize functions. Dual variables in linear programs are a form of gradient, and these are what guide the simplex algorithm. Gradients capture the value of an incremental change in some input such as a price, fleet size or the size of buffers in a manufacturing system. We can apply this same idea to learning.

Assume for the moment that we have the potential to observe ("measure") the value of *M* different alternatives. These may be the sales for different laptop configurations or *M* different prices. Perhaps we are using a business simulator and we have *M* different configurations of a factory layout or design of a logistics system. An observation might consist of a week's worth of sales figures, or a run of our business simulator. After *n* observations spread across the *M* alternatives, we have an estimate of the performance of each alternative, but we recognize that they are imperfect, as depicted by the spreads in the distributions in figure 1.

Given our state of knowledge, we have an *implementation decision,* which captures the decision we need to make with our information. In some cases, the measurement decisions and implementation decisions are the same (such as the best laptop configuration or the price we should charge), but in other cases they are different. For example, we might measure the time to traverse a link in a network (the measurement decision), after which we have to find the best path (the implementation decision) [1]. We make our implementation by choosing the alternative that seems to be the best, which is some form of deterministic optimization problem (such as a shortest path problem).

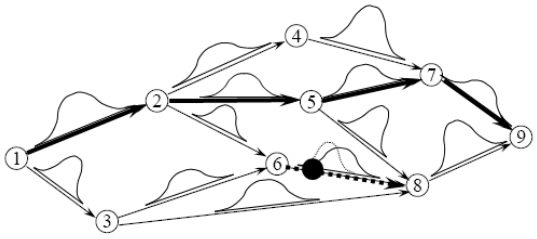

Figure 2-Network showing beliefs about the cost on each link initially and (for link 6-8) after an observation.

How do we estimate the value of a measurement? We might compute the expected marginal value of the implementation decision from a measurement. For example, if we decide to measure the time on link 6-8 in figure 2, the outcome of the measurement *may* identify a shortest path that is better than the path we were going to take. We refer to the *expected marginal value* of this measurement as the *knowledge gradient*. The knowledge gradient policy chooses to make the measurements that offer the highest value.

The term "knowledge gradient" was introduced by Peter Frazier while doing his doctoral research (Peter is now a professor at Cornell University). Peter showed that for offline learning problems, the knowledge gradient is myopically optimal (by construction, it is the best you can do if you can only make a single observation) and it is asymptotically optimal (given an unlimited budget, it will eventually find the optimal solution). These two properties seem to explain why it works well empirically for intermediate budgets.

A second Ph.D. student, Ilya Ryzhov (now a professor at the University of Maryland), then derived the knowledge gradient for online learning problems. The online knowledge gradient is related to the offline knowledge gradient through a simple, one-line equation that captures the tradeoff between the expected value from making a particular choice (such as selling a laptop with a particular configuration, or selling the laptop at a price), and the value of information in the future. With this simple equation, Ilya has provided a bridge between the offline ranking and selection and stochastic search communities, and the online bandit community [2].

With this result in hand, it is useful to stand back and think about the much broader class of learning problems that arise in real applications. We have already talked about problems that have distinct observation/measurement decisions (such as polling a population) and implementation decisions (such as where a candidate spends time campaigning). A separate and fundamental dimension is the *belief model*. Above, we assumed a *lookup table* belief model, where we have a belief (an estimate) of the value of each discrete alternative. The classical literature on learning assumes independent beliefs (if we observe that the sales from a particular laptop configuration is higher than expected, it tells us nothing about other alternatives).

It seems that in most real-world applications, we have the situation of *correlated beliefs*. For example, two laptop configurations may feature the same size screen, which turns out to be a key determinant of sales. If one configuration with a large screen sells well, it is likely that other configurations with the same screen will also sell well. Peter Frazier developed a fast, simple algorithm for computing the knowledge gradient in the presence of correlated beliefs which dramatically enhances the power of this strategy [3]. This simple idea allows us to consider hundreds of laptop configurations, and still make meaningful judgments after just a few dozen observations.

But what if we need to find the value of continuous variables such as prices or the size of buffers in a manufacturing simulator? Or, perhaps we want to find the best set of 15 ice cream flavors (out of 31) to put on display (over 300 million combinations)? We might replace our lookup table belief model with a statistical model which allows us to estimate the value of an alternative in terms of its attributes. This strategy was used in a project on drug discovery by Diana Negoescu, where the knowledge gradient was used to sequence experiments to discover the best molecular compound (out of 87,000) to cure a form of cancer using just a few dozen experiments [4]. Rather than finding flavors of ice cream, this problem required matching molecular fragments onto sites of a base molecule (labeled X and Y in figure 3).

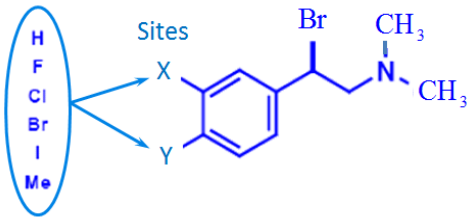

Figure 3-Base molecule with two sites for attaching additional fragments.

The different subcommunities that have addressed the problem of collecting information efficiently have been largely confined to well-trained researchers armed with Ph.D.'s. At Princeton, I teach this material each year to 30 or 40 undergraduates in the Department of Operations Research and Financial Engineering who have all taken a course in probability and statistics. Students learn a range of methods for efficiently collecting information, ranging from simple heuristics to the knowledge gradient, but also covering other optimal learning problems such as Fibonacci search and the secretary problem.

Along the way, we try to identify insights that are separate from any particular algorithm or policy. One example is the paradox of too many choices. If you have 50 alternatives and a budget to perform 50 observations, it makes sense to test each alternative once and then choose

the best. This strategy fails when observations are noisy. If there is enough uncertainty in an observation, a better strategy might be to choose 10 alternatives at random, and then use your budget of 50 observations to choose the best from this set of 10 [5].

The focus of the course, however, is on *modeling*, and learning how to *think* about an optimal learning problem. The course emphasizes basic ideas such as measurement and implementation decisions, belief models, the concept of a learning policy, and how to compare two learning policies. Perhaps most important is developing an appreciation of *when* it is important to think about the value of information. Many learning problems benefit from a prior distribution of belief (reflecting past experience or previous observations), and are naturally set in a Bayesian learning framework. This creates a natural testing environment for a policy, which consists of three steps: 1) assume a truth, 2) use a policy to try to discover the truth, and 3) then repeat 1000 times on many different truths to test the ability of the policy to work over many truths.

New ideas are best learned in terms of an application, and for this reason the course requires that the students (working in two-person teams) identify their own learning problem, and then compare the performance of different policies in the context of this problem. Applications have included topics that include maximizing ad-clicks, designing polling strategies for political campaigns, identifying the best diabetes treatment, pricing apps for smart phones, and solving complex protein folding problems. Each team gives a short presentation of their problem (without numerical results) so that everyone benefits from seeing the range of applications.

This article offers little more than a taste of this emerging field. Much more information is available at the website http://optimallearning.princeton.edu, which provides additional information, downloadable software and papers, a much longer list of applications, and some information on a new book, *Optimal Learning* (co-authored with Ilya Ryzhov), which is the textbook for the course. It is not possible to do justice to the many people who have contributed to the field in this short article; the bibliographic notes that accompany each chapter provide a guided tour of the literature.

References:

[1] I. O. Ryzhov and W. B. Powell, "Optimal learning on a graph," *Operations Research*, Vol. 59, No. 1, pp. 188-201, 2011.

[2] I. O. Ryzhov, W. B. Powell, P. I. Frazier, "The knowledge gradient algorithm for a general class of online learning problems," *Operations Research*, Vol. 60, No. 1, pp. 180-195 (2012).

[3] P. I. Frazier, W. B. Powell, S. Dayanik, "The Knowledge-Gradient Policy for Correlated Normal Beliefs," *Informs Journal on Computing*, Vol. 21, No. 4, pp. 585-598 (2009)

[4] D. Negoescu, P. Frazier and W. B. Powell, "The Knowledge Gradient Algorithm for Sequencing Experiments in Drug Discovery", *Informs Journal on Computing*, Vol. 23, No. 3, pp. 346-363, 2011. http://www.castlelab.princeton.edu/DrugDiscovery.htm

[5] P. I. Frazier, and W. B. Powell, "Paradoxes in Learning and the Marginal Value of Information," *Decision Analysis*, Vol. 7, No. 4, pp. 378-403, 2010.