The Jungle of Stochastic Optimization

SIAM News, November, 2015

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

There is a vast range of problems that fall under the broad umbrella of making sequential decisions under uncertainty. While there is widespread acceptance of basic modeling frameworks for deterministic versions of these problems from the fields of math programming and optimal control, sequential stochastic problems are another matter.

Motivated by a wide range of applications, entire fields have emerged with names such as dynamic programming (Markov decision processes, approximate/adaptive dynamic programming, reinforcement learning), stochastic optimal control, stochastic programming, model predictive control, decision trees, robust optimization, simulation optimization, stochastic search, model predictive control, and online computation. Problems may be solved offline (requiring computer simulation) or online in the field, which opens the door to the communities working on multi-armed bandit problems[i]. Each of these fields has developed its own style of modeling, often with different notation (x or S for state, u/x/a for decision/action/control), and different objectives (minimizing expectations, risk, or stability). Perhaps most difficult is appreciating the differences in the underlying application. A matrix K could be 5x5 in one class of problems, or 50,000x50,000 in another (it still looks like K on paper). But what really stands out is how each community makes a decision.

Despite these differences, it is possible to pull them together in a common framework that recognizes that the most important (albeit not the only) difference is the nature of the *policy* being used to make decisions over time (we emphasize that we are only talking about *sequential problems*, consisting of decision, information, decision, information, …). We start by writing the most basic canonical form as

$$\min_{\pi\in\Pi} \mathbb{E}^{\pi} \sum_{t=0}^{T} C(S_t, U_t^{\pi}(S_t)) \tag{1}$$

where $S_{t+1} = S^M(S_t, u_t, W_{t+1})$. Here, we have adopted the notational system where S_t is the state (physical state, as well as the state of information, and state of knowledge), and u_t is a decision/action/control (alternatives are x_t, popular in operations research, or a_t, popular in operations research as well as computer science). We let $U_t^{\pi}(S_t)$ be the decision function, or policy, which is one member in a set Π where π specifies both the type of function, as well as any tunable parameters $\theta \in \Theta^{\pi}$. The function $S^M(S_t, u_t, W_{t+1})$ is known as the transition function (or system model, state model, plant model, or simply "model"). Finally, we let W_{t+1} be the information that *first* becomes known at time t+1 (control theorists would call this w_t, which is random at time *t*).

Important problem variations include different operators to handle uncertainty; we can use an expectation in (1), a risk measure, or worst case (robust optimization), as well as a metric capturing system stability. We can assume we know the probability law behind W_t, or we may just observe S_{t+1} given S_t and u_t (model-free dynamic programming).

While equation (1) is well-recognized in certain communities (some will describe it as "obvious"), it is actually quite rare to see (1) stated as the objective function with anything close to the automatic writing of objective functions for deterministic problems in math programming or optimal control. We would argue that the reason is that there is no clear path to computation. While we have powerful algorithms to serve over real-valued vector spaces (as required in deterministic optimization), equation (1) requires that we search over spaces of functions (policies).

Lacking tools for performing this search, we make the argument that all the different fields of stochastic optimization can actually be described in terms of different classes of policies. In fact, we have identified four fundamental (meta) classes, which are:

1. Policy function approximations (PFAs) – These are analytical functions that map states to actions. PFAs may come in the form of lookup tables, parametric, or nonparametric functions. A simple example might be

$$U^{\pi}(S_t \mid \theta) = \sum_{f \in F} \theta_f \phi_f(S_t) \tag{2}$$

   where F is a set of features, and $\left(\phi_f(S_t)\right), f \in F$ are sometimes called basis functions.

2. Cost function approximations (CFAs) – Here we are going to design a parametric cost function, or parametrically modified constraints, producing a policy that we might write as

$$U^{\pi}(S_t \mid \theta) = \arg\min_{u \in \mathcal{U}_t^{\pi}(\theta)} C_t^{\pi}(S_t, u \mid \theta) \tag{3}$$

   where $C_t^{\pi}(S_t, u \mid \theta)$ is a parametrically modified set of costs (think of including bonuses and penalties to handle uncertainty), while $\mathcal{U}_t^{\pi}(\theta)$ might be a parametrically modified set of constraints (think of including schedule slack in an airline schedule, or a buffer stock).

3. Policies based on value function approximations (VFAS) – These are the policies most familiar under the umbrella of dynamic programming and reinforcement learning. These might be written as

$$U_t^{\pi}(S_t \mid \theta) = \arg\min_{u \in \mathcal{U}_t^{\pi}(\theta)} C(S_t, u) + \mathbb{E}\left\{\overline{V}_{t+1}^{\pi}(S^M(S_t, u, W_{t+1}) \mid \theta) \mid S_t\right\} \tag{4}$$

   where $\overline{V}_{t+1}^{\pi}(S_{t+1})$ is an approximation of the value of being in state $S_{t+1} = S^M(S_t, u, W_{t+1})$, where π captures the structure of the approximation and $\theta \in \Theta^{\pi}$ represents any tunable parameters.

4. Lookahead policies – Lookahead policies start with the basic observation that we can write an optimal policy using

$$U_t^{\pi}(S_t \mid \theta) = \arg\min_u \left(C(S_t, u) + \min_{\pi \in \Pi} \mathbb{E}^{\pi} \left\{ \sum_{t'=t+1}^{T} C(S_{t'}, U_{t'}^{\pi}(S_{t'})) \mid S_t, u \right\} \right) \tag{5}$$

   The problem is that the second term in (5) is not computable (if this were not the case, we could have solved the objective function in (1) directly). For this reason, we create a *lookahead model* which is an approximation of the real problem. Common approximations are to limit the horizon (e.g. from T, which might be quite long, to t+H for some appropriately chosen horizon

H), and (most important) to replace the original stochastic information process with something simpler. The most obvious is a deterministic approximation, which we can write as

$$U_t^\pi(S_t \mid \theta) = \arg\min_{u_t, \tilde{u}_{t,t+1},\ldots,\tilde{u}_{t,t+H}} \left(C(S_t, u_t) + \sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}, \tilde{u}_{tt'}) \right) \qquad (6)$$

To make the distinction from our original *base model* in (1), we put tildes on all our variables (other than those at time t), and we also index the variables by t (to indicate that we are solving a problem at time t), and t' (which is the point in time within the lookahead model).

A widely used approach in industry is to start with (6) and then introduce modifications (often to the constraints) so that the decisions made now are more robust to uncertain outcomes that occur later. This would be a form of (hybrid) cost function approximation.

We may instead use a stochastic lookahead model. For example, the stochastic programming community most often uses

$$U_t^\pi(S_t \mid \theta) = \arg\min_{u_t, \tilde{u}_{t,t+1},\ldots,\tilde{u}_{t,t+H}} \left(C(S_t, u_t) + \sum_{\omega \in \tilde{\Omega}_t} p(\omega) \sum_{t'=t+1}^{t+H} C(\tilde{S}_{tt'}(\omega), \tilde{u}_{tt'}(\omega)) \right) \qquad (7)$$

Here, we would let θ capture parameters such as the planning horizon, and the logic for constructing $\widehat{\Omega}_t$.

Other variations include a robust objective (which minimizes over the worst outcome rather than the expected outcome), or a chance-constrained formulation, which approximates the costs over all the uncertain outcomes using simple penalties for violating constraints.

All of these policies involve tunable parameters, given by θ. We would represent the policy π as the policy class $f \in \mathcal{F}$, and the parameters $\theta \in \Theta^\pi$. Thus, the search over policies π in equation (1) can now be thought of as the search over policy classes $f \in \mathcal{F}$, and then over the tunable parameters $\theta \in \Theta^\pi$.

No, this is not easy. But with this simple bit of notation, all of the different communities working on sequential stochastic optimization problems can be represented in a common framework.

Why is this useful? First, a common vocabulary facilitates communication and the sharing of ideas. Second, it is possible to show that each of the four classes of policies can work best on the same problem, if we are allowed to tweak the data. And finally, it is possible to combine the classes into hybrids that work even better than a pure class.

And maybe some day, mathematicians will figure out how to search over function spaces, just as Dantzig taught us to search over vector spaces.

[i] http://www.castlelab.princeton.edu/jungle.htm